Savremene Komunikacione

tehnologije i mre

EDICIJA STVARNI SVET

William A. Shay

Savremene komunikacione tehnologije i mreže

STVARNI SVET

VVILLIAM A. SHAY

UNIVERSITY OF WISCONSIN - GREEN BAY

kompjuter biblioteka

THOIVISOIM

LEARINING

Izdavač:

Kompjuter Biblioteka
Vladana Šićevića 19
32000 Čačak
tel: 032/320-140, 232-322
fax: 032/232-322,
Beograd, Vojvode Stepe 34A-5
tel: 011/309-69-66

Tekući računi:

155-847-88 i 205-8174-10

e-mail:

kombib@eunet.yu

internet:

www.kombib.co.yu

Urednik:

Mihailo J. Solajić

Za izdavača, direktor:

Mihailo J. Solajić

Prevod:

Dijana Ivanišević

Lektura:

Miloš Jevtović

Korice:

Saša Prudkov

Slog:

Zora Radojević
Ana Pešić
Ivana Petronijević

Znak Kompjuter biblioteke:

Miloš Milosavljević

Stampa:

"Svetlost" Čačak

Godina izdanja:

2004.

Izdanje:

Prvo

ISBN: 86-7310-310-X

UNDERSTANDING DATA COMMUNICATIONS AND NETWORKS

WILLIAM A. SHAY
UNIVERSITY OF WISCONSIN - GREEN BAY

Sadržaj

Predgovor

Namena

Drugo izdanje ove knjige je izašlo pre par godina, a od tada se promenilo mnogo štošta na polju razmene podataka i kompjuterskih mreža.

- Softver baziran na Webu postao je sasvim uobičajen.
- Potreba za bezbednim konekcijama na Web sajtovima je opšta.
- Razmatraju se bezbednosni aspekti i razvijaju alatke za borbu protiv sve većih pretnji koje ugrožavaju bezbednost.
- Kreirani su novi standardi za šifrovanje i napravljen je kompromis sa starima.
- Zbog zahteva za kvalitetnim servisima, javila se potreba za novim protokolima koji će se pokretati zajedno sa postojećima.
- Nove tehnologije za uspostavljanje konekcija, kao što su DSL, USB i FireVVire, postaju sasvim uobičajene.
- Ethernet mreže su dostigle gigabitske brzine.
- Šeme za kompresiju audio zapisa, kao što je MP3, i mogučnost igranja igara preko mreže zauvek su promenili način na koji ljudi koriste Internet.
- Bežične tehnologije počinju da predstavljaju prihvatljivu alternativu za mnoge korisnike.
- Retko se koriste protokoli koji su ranije bili uobičajeni, ili se mislilo da imaju perspektivu u budućnosti.

Mi više ne živimo u istom svetu u kome smo živeli kada je izašlo prethodno izdanje ove knjige; ovo izdanje odslikava promene koje su se desile u meduvremenu.

Iako je veliki deo sadržaja knjige promenjen, njena namena je, u osnovi, ostala nepromenjena. Napisana je za studente na nižim godinama studija na odseku za kompjutersku tehniku koji su odslušali najmanje dva semestra o izradi softvera i imaju dobru osnovu iz matematike, uključujući i diskretnu matematiku. Obuhvaćene su standardne teme na uvodnom kursu o komunikacijama i kompjuterskim mrežama, kao što su mediji za prenos, analogni i digitalni signali, prenos podataka, metodi za kompresovanje i šifrovanje, mrežne topologije, zaštita mreža, LAN protokoli, Internet protokoli i aplikacije, tehnologije sa komutacijom kola i Web aplikacije. Cilj nam je da pomognemo čitaocu da razume:

- razlike, prednosti i nedostatke različitih medija za prenos
- analogne i digitalne signale, tehnike modulacije i demodulacije i način funkcionisanja uredaja za modulaciju, kao što su modemi, kablovski modemi i DSL modemi
- efekat šuma u toku prenosa i kako protokoli detektuju da su informacije promenjene
- kako protokoli reaguju u situacijama kada šum izaziva oštećenja, ili gubitak informacija
- standarde kao što su AES, ATM, DES, EIA-232, HDLC, IEEE 802.3, IEEE 802.5, IEEE 802.11, IPv6, JPEG, MP3, MPEG, OSI, SONET, TCP/IP i X.25, organizacije za uspostavljanje standarda i razloge zašto su standardi neophodni
- tehnike za kompresovanje podataka, tipove podataka koji se mogu kompresovati i poredenje različitih metoda koji se danas koriste
- "crve", viruse i ostale pretnje umreženim kompjuterima
- potrebu za zaštitom i efektivnim metodima šifrovanja
- razlike izmedu sistema šifrovanja javnim i privatnim ključem
- kako se uspostavljaju bezbedne konekcije između udaljenih sajtova
- potrebu za kontrolom toka i različite načine za njenu implementaciju
- protokole koji se koriste na lokalnim mrežama i strategije nadmetanja na deljenim medijima za prenos podataka
- bežične standarde
- metode povezivanja lokalnih mreža
- strategije rutiranja
- potrebu za protokolima koji podržavaju real-time video aplikacije i koji zadovoljavaju zahteve za kvalitetnim servisima
- kako se dizajniraju i postavljaju različite funkcionalne klijent/server aplikacije
- kako sve veće korišćenje VVeba i multimedijalnih aplikacija utiče na postojeće protokole i šta je učinjeno da bi se prevazišle eventualne poteškoće.

Sadržaj i organizacija

U trećem izdanju su izvršene velike promene - neke su usledile nakon komentara čitalaca, a ostale su rezultat razvoja tehnologije. Mnoge slike su detaljnije razjašnjene i poboljšana su objašnjenja pojedinih protokola. Uključene su i brojne teme koje su u današnje vreme sasvim uobičajene, zajedno sa novim dostignućima razvoja, i izostavljene su stare teme koje danas više nemaju nikakvu značajnu ulogu na ovom polju.

Najočiglednija promena je možda u strukturi knjige koja je sada organizovana u sledećih trinaest poglavlja:

Poglavlje 1 Uvod u komunikacije, standarde i protokole

Poglavlje 2 Mediji za prenos i kodovi

Poglavlje 3 Analogni i digitalni signali

Poglavlje 4 Uspostavljanje konekcija

Poglavlje 5 Kompresovanje podataka

Poglavlje 6 Integritet podataka

Poglavlje 7 Zaštita podataka

Poglavlje 8 Kontrola toka

Poglavlje 9 Lokalne mreže

Poglavlje 10 Povezivanje mreža

Poglavlje 11 Internet protokoli i aplikacije

Poglavlje 12 Intemet programiranje

Poglavlje 13 Tehnologije sa komutacijom kola

Ova restruktuirana knjiga predavačima treba da obezbedi bolje fokusiranje na specifične teme koje trenutno obraduju u okviru svojih predavanja. Neka od ovih poglavlja odgovaraju delovima poglavlja iz prethodnog izdanja, a nekim je predstavljen potpuno novi materijal. Najznačajnije promene uključuju nove, ili proširene teme u vezi:

- medija, uključujući provodne metale, optički prenos, bežične i satelitske komunikacije
- DSL tehnologija
- Universal Serial Bus (USB) i FireWire (IEEE 1394 standard) protokola
- Synchronous Optical Network (SONET) protokola
- tehnike aritmetičke, faksimil i MP3 kompresije
- Advanced Encryption Standard (AES) i Rijndaelovog algoritama
- programa Pretty Good Privacy
- Secure Sockets Layer, Transport Layer Security i X.509 sertifikata
- Firewalla
- sigurnosnih pretnji
- standarda za Ethernet, brzi (Fast) Ethernet i Gigabit Ethernet (dat je opšti pregled 10-gigabit Ethernet standarda)
- 802.11 Wireless LAN standarda
- komutiranog Etherneta
- virtuelnog LAN-a
- protokola na slojevima 3 i 4, uključujući besklasno rutiranje izmedu domena, rutiranje i rutere, rutiranje paketa ka više odredišta istovremeno, pitanja obezbedivanja kvalitetaih servisa, Real-Time Transfer protokol i IPSec
- Internet aplikacija
- CGI programiranja, uključujući radne primere sistema za naručivanje preko Weba pomoću Linuxa i Perl Scripts

- Frame Relay protokola

Iako bi bilo teško (skoro nemoguće) obuhvatiti sve moguće teme u okviru jednosemetralnog kursa, opseg predstavljenih tema omogućava predavačima da izaberu koje su najvažnije za njihove studente.

U ovoj knjizi je ponudena mešavina teorije i praktičnih aplikacija. Teorija obezbeduje solidnu osnovu za dalje studije, a aplikacije približavaju studente stvarnim komunikacionim sistemima i mrežama - oni, na ovaj način, stiču i dragocena iskustva, Svi studenti će imati koristi od praktičnih aplikacija, doksu teorijska objašnjenja, pre svega, namenjena ambicioznijim studentima. Osim toga, u Poglavlju 12 je predstavljen stvarni model radnih programa za klijent i server.

Svako poglavlje predstavlja osnovu na koju se nadovezuje naredno. Na primer, kada proučavate multipleksiranje, nadmetanje za pristup zajedničkom mediju, ili kompresiju, pre toga bi trebalo da proučite načine na koje se signali prostiru kroz različite medije. Kada proučavate lokalne mreže, trebalo bi da znate kakvi problemi mogu da nastanu zbog nadmetanja na linijama sa višestrukim pristupom, kakve probleme stvaraju kanali sa šumovima i kako se vrši kontrola toka. Kada proučavate protokole za WAN mreže, trebalo bi da razumete protokole za lokalne mreže i zašto ti protokoli nisu prikladni za veće mreže. Sledi kratak rezime svih poglavlja.

Poglavlje 1 obezbeduje uvod u oblast kojom ćemo se baviti ("dodirnuti" su tekući problemi i aplikacije na polju komunikacija i mreža). Opisana je potreba za uspostavljanjem standarda, navedene se relevantne organizacije za uspostavljanje standarda, a, zatim, sledi pregled standardizovanog modela protokola Open Svstem Interconnect. Poglavlje se završava predvidanjima o budućim dešavanjima.

U Poglavlju 2 predstavljeni su različiti tipovi medija za prenos (kabl, žičani, bežični, satelitski, optički fiber), njihove prednosti i nedostaci i različiti kodovi koji se koriste za dodeljivanje značenja podacima. U Poglavlju 3 proučavamo tipove analognih i digitalnih signala, tehnike modulacije koje su neophodne za njihovo konvertovanje i efekat šuma na bitskim brzinama. Osim toga, predstavljeni su modemi, kablovski modemi i DSL tehnologije.

Poglavlje 4 se fokusira na uspostavljanje konekcija, na modove prenosa, na komunikacione nosače (telefonski sistem, SONET i T1), na standarde interfejsa (EIA-232, USB i FireWire) i na način kako više uređaja pristupa zajedničkom mediju (metodi multipleksiranja i različiti protokoli za uspostavljanje konekcije).

U Poglavlju 5 su predstavljene tehnike za kompresovanje podataka i objašnjeno je kako se u tim tehnikama koriste različiti tipovi redundantnosti podataka. U Poglavlju 6 se bavimo integritetom prenetih podataka, detektovanjem grešaka i tehnikama za korigovanje grešaka, kao što su parnost, CRC i Hamingovi kodovi.

Poglavlje 7 posvećeno je zaštiti podataka, uključujući tehnike šifrovanja (i sa javnim i sa privatnim ključem), standarde za šifrovanje, algoritme za razmenu ključa, metode za autentifikaciju, X.509 sertifikate i bezbedne konekcije, firewalle i različite pretnje (viaise, "crve", hakere i napade odbijanja servisa).

U Poglavlju 8 je prikazan algoritam kontrole toka koji opisuje kako uredaj rukuje razmenom informacija i šta se dešava prilikom gubitka, ili oštećenja podataka. Opisane su i neke tehnike koje se koriste za formalnu verifikaciju tačnosti protokola.

Nakon toga, u Poglavlju 9 predstavljeni su LAN protokoli, uključujući nekoliko vrsta Etherneta - originalni, brzi (Fast) Ethernet i Gigabit Ethernet, Token Ring i IEEE 802.11 VWireless LAN standard.

U Poglavlju 10 se bavimo načinima povezivanja mreža. Obradeni su konekcije sa Sloja 2 (mostovi i komutatori), učenje adresa, algoritam otvorenog stabla, komutirani Ethernet i VLAN mreže. Osim toga, u ovom poglavlju se bavimo konekcijama Sloja 3 i predstavljamo različite algoritme rutiranja (Dijkstra, Bellman-Ford, RIP, BGP i mnoge druge). Opisani su I problemi zagušenja mreže i "samrtni zagrljaj".

Poglavlje 11 je posvećeno Internetu. Obuhvaćene su verzije 4 i 6 Internet protokola, kvalitet servisa, rutiranje ka više odredišta i drugi protokoli koji su dizajnirani da bi bili ispunjeni zahtevi nekih real-time servisa na Internetu. Osim toga, obraden je TCP (upravljanje konekcijama, kontrola toka i upravljanje zagušenjem), a dat je i opis nekoliko uobičajenih Internet aplikacija (Telnet, SSH, FTP i SMTP).

Poglavlje 12 je namenjeno onima koji u okviru kursa nameravaju da odrade i neke projekte. Obezbedeni su radni primeri klijent/server aplikacija. Primeri uključuju soket programiranje, CGI programiranje korišćenjem C-a i Perla i primer koda koji ilustruje kako se vrši transfer fajlova i kako funkcionišu pretraživačka mašina i sistem za online narudžbine.

U Poglavlju 13 se bavimo tehnologijama sa komutacijom kola, kao što su ISDN, X.25, Frame Relay i ATM.

Pitanja na kraju svakog poglavlja su podeljena u dve grupe. Prva grupa (Pitanja za proveru) sadrži pitanja na koja se odgovori mogu dati direktno na osnovu sadržaja poglavlja - ona treba da ohrabre čitaoca da se vrati na tekst i da izabere ono što su autor i predavač smatrali najbitnijim. Smatram da je ovaj metod bolji sa pedagoškog stanovišta, u odnosu na pristup kod koga se na kraju poglavlja jednostavno navedu najvažnije teme, jer se ovako student ohrabruje da čita knjigu kao da je reč o romanu-lineamo. Ipak, učenje složenog materijala često zahteva ponovna iščitavanja da bi se razvrstali i razumeli različiti koncepti. Jedan kolega mi je ispričao da je ranije imao problema sa nekim studentom koji je stalno zaostajao za ostalima zato što je imao neki honorarni posao, ali je na poslu imao i nešto slobodnog vremena; umesto da se dosaduje, odlučio je da ponese ovu knjigu na posao i da čita kad god mu se ukaže pogodna prilika. Kasnije je u toku semestra značajno napredovao i rekao je svom predavaču da mu je nakon četvrtog, ili petog čitanja sve bilo jasno.

Pitanja za proveru nisu dovoljna. Druga grupa (Vežbe) sadrži pitanja koji čitaoca navode da primeni ono što je naučio i da vrši poređenja, donosi logičke zaključke i razmatra moguće alternative. Odgovori nisu uvek jednostavni i to su najčešće problemi sa kojima biste še sretali u praksi.

Dopune za predavače

- Instructor's Solutions Manual, uputstvo u kojem možete da pronadete odgovore na pitanja iz provere i vežbi; na raspolaganju je predavačima koji dobiju odobrenje od izdavača
- primeri koji su na raspolaganju predavačima na osnovu zahteva (email: shayw@uwgb.edu)

- dodatne instrukcije možete da pronadete i na autorovom Web sajtu http://www.uwgb.edu/shayw/udcn3. Tu se nalaze slike knjige u pdf formatu, ispravke grešaka koje su otkrivene nakon štampanja knjige, sve kopije koda koji je predstavljen u Poglavlju 12 i brojni linkovi ka korisnim Web sajtovima, organizovani po temama iz poglavlja.

Zahvalnost

U pisanju jedne ovakve knjige retko može da učestvuje samo jedna osoba. Mnogi ljudi su doprineli nastanku ove knjige - dali su mi dragocene ideje i informacije i pružili svesrdnu podršku u toku realizacije ovog projekta. Dragocene savete prilikom pisanja prva dva izdanja ove knjige pružili su mi sledeći ljudi kojima se posebno zahvaljujem:

Abdullah Abonamah
Universit}' of Akron

David Kieper
University of Winsonsin-Green Bay

George W. Ball
Alfred University

Lance Leventhal

Mehran Basiratmand
Florida International University

Judith Molka
Univeristy ofPittsburgh

Ron Bates
DeAnza College

Dan O'Connell
Fredonia College-SUNY

Bruce Derr

John L. Spear
Syracause University

Mohammad El-Soussi
Santa Barbara City College

Janet M. Urlaub
Sinclair Communit}' College

James E. Holden
Clarion University

David Whitney
San Francisco State Universitf

dr Sub Ramakrishnan
Boivling Creen State University

dr J. Archer Harris
James Madison University

dr Seyed H. Roosta
Mount Mercy College

dr Paul H. Higbee
University of North Florida

Dr. Brit Williams
Kennesam State University

dr Gene Hill Price
Old Dominion University

Sten Wine
Hunter College i
Neui Era of Netutorks, Inc.

dr J. Mark Pullen
George Mason Universit)'

Zahvaljujem se onima koji su mi obezbedili korisne sugestije za poboljšanje drugog izdanja i koji su imali vremena da pregledaju moj rukopis za treće izdanje. Pažljivo sam razmotrio sve komentare i sugestije i mnoge od njih uključio u konačni rukopis.

Najiskrenije se zahvaljujem recenzentima ovog izdanja:

Irvinu Jay Levyu
Gordon College

Marku Pullenu
George Mason University

Abyu Tehranipouru
Eastern Michigan University

Cameliji Zlatea
De Paul University

Takođe se zahvaljujem ljudima u Brooks/Cole, uključujući mog urednika Kallie Swanson i njenog pomoćnika Aartija Jayaramana, kao i Penmarin Books za produkciju, Cindy Kogut za copy editing i George Barlie iz Accurate Art za ilustraciju ovog novog izdanja. Njihov doprinos i zalaganje su omogućili pretvaranje mog rukopisa u knjigu. Mojoj porodici - Judy, Danu i Timu dugujem posebnu zahvalnost. Oni su se žrtvovali da bih ja mogao da koristim svoje "slobodno vreme" za pripremu rukopisa. Obećavam da ću im to nadoknaditi. Konačno, veoma cenim mišljenje svih onih koji će pročitati ovu knjigu. Slobodno mi pošaljite svoje komentare na adresu Bill Shay, Department of Information and Computing Sciences, University of Winsconsin - Green Bay, Green Bay, WI 54311-7001, ili e-mailom shayw@uwgb.edu.

BILL SHAY

POGLAVLJE 1

Uvod u komunikacije, standarde i protokole

Za Ljubav prema učenju, skrivena skrovista i slatko blaženstvo knjiga

— **Henry Wadsworth Longfellow** (1807-1882), američki pesnik

Postoje dve vrsta znanja. Ili znamo sve o nečemu, ili znamo gde možemo da pronađemo informacije o tome.

— **Samuel Johnson** (1709-1784), britanski autor

1.1 Zašto uopšte proučavamo komunikacije?

Zašto bismo uopšte proučavali kompjutere i razmenu podataka? Postoje brojni razlozi, od onih tipa "Apsolutno sam očaran tom oblašću" do " Moram da znam kako da povežem svoj kompjuter na mrežu moje kompanije". Jedan od najvažnijih razloga je činjenica da su komunikacione tehnologije prodrle apsolutno u sve aspekte našeg života, od profesionalnih i obrazovnih okruženja, do "čiste" rekreacije. Ove tehnologije su imale toliko jak prodor da se često uzimaju "zdravo za gotovo" i uopšte nismo ni svesni svih njihovih primena.

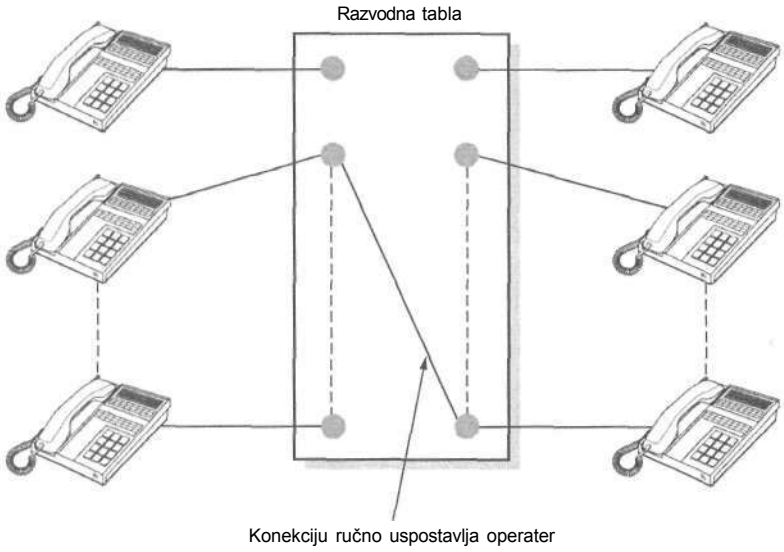
Kratak istorijat

Polje komunikacija nije novo: ljudi su komunicirali još od svog postanka najprimitivnijim načinima sporazumevanja i crtanjem na zidovima pećina. Hiljadama godina su komunicirali koristeći reči, pergament, kamene blokove i dimne signale. Primarni oblici slanja informacija zasnivali su se na čulu sluha i vizuelnom predstavljanju. Ili čujete nekoga da govori, ili vidite slova i simbole koji definišu poruku.

Komunikacije su se drastično promenile 1837. godine, kada je Semjuel Morze (Samuel Morse) izumio telegraf. Zahvaljujući ovom izumu, bilo je moguće poslati informacije pomoću električnih impulsa preko bakarne žice. Poruke su slate tako što se svaki karakter prevodio u niz dugačkih, ili kratkih električnih impulsa, ili, ako ne koristimo tehničke izraze, u nizove tačaka i crtica - ti nizovi su prenošeni preko žice. Pridruženi skup karaktera i električnih impulsa naziva se Morzeov kod. Mogućnost slanja informacija bez očiglednog verbalnog, ili vizuelnog medijuma predstavljala je kamen temeljac mnogim izumima koji će zauvek promeniti načine komunikacije između ljudi.

Aleksander Graham Bel (Alexander Graham Bell) je 1876. godine pomerio telegraf jedan korak unapred. Pokazao je kako glas može direktno da se konvertuje u električnu energiju i prenosi preko žice korišćenjem naizmeničnog napona. Na drugom kraju žice električni signali su konvertovani nazad u zvuk. Rezultat je bila mogućnost prenosa govora elektronskim putem izmedu dve tačke, čije je rastojanje zavisilo samo od mogućnosti fizičkog povezivanja tih tačaka. Za ljude čiji su životi zavisili samo od toga šta su mogli da vide i čuju ovaj izum je bio apsolutno neverovatan i delovao je nestvarno.

Najraniji telefoni su zahtevali poseban par žica za svaki telefon na koji je neka osoba htela da se poveže. Da bi nekoga pozvala, ta osoba je najpre morala da poveže svoj telefon na par žica i da se nada da je neko na drugom kraju sluša. Nije bilo nikakvog zvona, ili uređaja za signaliziranje koji bi osobu na drugom kraju obavestio o pozivu. To se promenilo pronalaskom razvodne table (switchboard), razvodnog uređaja (slika 1.1) koji je povezivao linije između telefona. Kada je neko hteo da pozove nekoga, jednostavno je podizao slušalicu i "redtovao" broj osobe koju želi da pozove. Telefoni tada još uvek nisu bili došli do tačke u kojoj bi ljudi sami obavljali aktivnosti kao što su okretanje brojčanika, ili pritiskanje dugmadi. Uspostavljanje konekcija je aktivirano glasom. Konkretno, operater bi čuo broj, a zatim bi koristio razvodnu tablu za povezivanje linija telefona osobe koja upućuje poziv sa linijama telefona tražene osobe.



SLIKA 1.1 *Razvodna tabla*

U narednih 70 godina telefonski sistem je toliko napredovao da je telefon postao sasvim uobičajeni aparat u svakoj kući. Većina nas nikada se nije ni zapitala kako telefonski sistem funkcioniše. Znamo da okrenemo neki broj i jednostavno sačekamo da uspostavimo vezu sa bilo kojim delom sveta.

Sledecl značajan dogadaj na polju komunikacija desio se 1945. godine, kada je izmišljen prvi elektronski kompjuter ENIAC (Electronic Numerical Integrator and Calculator). Dizajniran je za balističke proračune u Drugom svetskom ratu i predstavljao je prvi uredaj koji je mogao da obraduje informacije elektronskim putem. Iako ENIAC nije imao direktnu ulogu u kompjuterskim komunikacijama, pokazao je da se izračunavanja i donošenje odluka mogu izvesti elektronskim putem, što je jedna od polaznih osnova današnjih komunikacionih sistema.

Kompjuteri i komunikacije počinju da "izbijaju na površinu" odmah nakon pronalaska prvog tranzistora (1947. godine), koji je omogućavao kreiranje manjih i jeftinijih kompjutera. Nova generacija kompjutera se pojavila 60-ih godina prošlog veka; sa njima je olakšano procesiranje i rutiranje telefonskih poziva. Osim toga, sve više kompanija kupuje kompjutere i razvija aplikacije za njih, tako da narasta i potreba za prenosom informacija izmedu njih.

Prvi komunikacioni sistem izmedu kompjutera bio je jednostavan, ali pouzdan. U osnovi, uključivao je zapisivanje informacija sa jednog kompjutera na magnetnu traku, a zatim se sa tom trakom odlazilo do drugog kompjutera (neki ljudi i danas rade isto, mada su magnetne trake zamenjene diskovima, CD-ROM-ovima i DVD-em). Na drugom kompjuteru je bilo moguće pročitati informacije sa trake. Ovo je bio pouzdan oblik komunikacije, uz pretpostavku da je traka mogla bezbedno da se prenese do svog odredišta.

Sledeći značajan pomak u elektronskim komunikacijama desio se sa razvojem prvog personalnog kompjutera (PC-ja). Postojanje kompjutera koji se nalazi na radnom stolu otvara potpuno novi svet mogućnosti za smeštanje i pribavljanje informacija. Ogroman broj PC-ja uveden je 80-ih godina u skoro sva poslovna okruženja, kompanije, škole i organizacije, ali i u brojne domove. Cinjenica da je veliki broj ljudi imao kompjutere uslovila je potrebu za još lakšim načinima za razmenu informacija.

World Wide Web, aplikacija koja je informacije iz bilo kog dela sveta učinila lako dostupnim sa bilo čijeg PC-ja, nastao je u narednoj deceniji. Pomoću klikova mišem korisnici kompjutera piogu da pristupaju fajlovima, programima, video klipovima i zvučnim zapisima. Online servisi, kao što su America Online, ili Yahoo, obezbeduju pristup mnoštvu usluga za svoje korisnike, kao što su čet sobe (prostorije za ćaskanje), oglasne table (bulletin boards), sistemi za rezervaciju avionskih karata i još mnogo štošta.

Da napomenemo da mnogi ljudi pogrešno misle da koncept mreža za razmenu podataka potiče sa kraja 90-ih godina prošlog veka, od pojave Interneta. Mnogi se često iznenade kada čuju da je mreža za razmenu podataka prvi put kreirana u Francuskoj u 18. veku, oko 200 godina pre nastanka Interneta! Bio je konstruisan niz tornjeva, koji su imali časovnike sa klatnom i panele koji su na jednoj strani bili crni, a na drugoj beli.

Osoba na prvom tornju bi u skladu sa časovnikom postavljala panel tako da bude vidljiva crna, ili bela strana. Druga osoba bi na udaljenom tornju teleskopom posmatrala kako je postavljen panel na prvom tornju i u skladu sa tom postavkom definisan je položaj lokalnog panela. Poruke su kodirane u skladu sa nizovima crnih i belih slika i prenošene su od jednog tornja do sledećeg u nizu. Prva poslata poruka je prelazila otprilike 16 kilometara, za šta su bila potrebna otprilike četiri minuta. Referenca [RHo94] obezbeđuje fascinantno štivo o toj mreži i nekim motivima koji su inspirisali konstrukciju ovakve mreže.

I na ulasku u 21. vek nove tehnologije neprestano menjaju način našeg rada i sliku sveta koju trenutno imamo. Integracija medijuma i komunikacionih servisa, zajedno sa eventualnom konverzijom u digitalne prenose, obećava potpuno novi svet interaktivne zabave i nove mogućhosti za obrazovanje. Pristup Intemetu koji obezbeduju kompanije koje se bave uvodenjem kablovskih sistema u kucha okruženja nudi sve veće brzine preuzimanja informacija. Zahvaljujući tome, na raspolaganju imamo sve veći broj informacija. Palmtop kompjuteri i bežične tehnologije omogućavaju fleksibilnije korišćenje kompjutera, učenje i zabavu koje ranije nije bilo moguće obezbediti. Sve veći broj ljudi koristi prednosti ovih tehnologija, što, sa druge strane, nameće sve veći broj etičkih i pravnih problema. Sada je sve teže kontrolisati pornografiju i materijal sa eksplicitnim nasiljem. Problem zaštite autorskih prava (copyright) eskalirao je 2000. godine, kada je tehnologija toliko napredovala da su ljudi mogli da razmenjuju popularnu muziku u digitalnom, kompresovanom formatu. Trenutno su u tu "priču" uključeni i problemi u vezi videa.

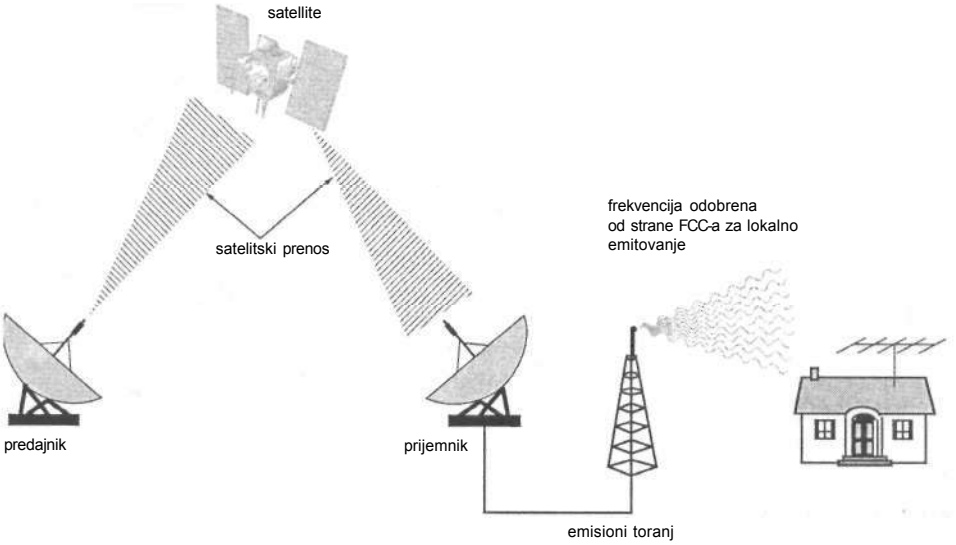
Kompjuteri i komunikacije su napredovali do te mere da je danas skoro nemoguće zamisliti funkcionisanje kompanija, škola, pa, čak, i mnogih individua bez kompjutera. Naša potpuna zavisnost od kompjutera nameće nam potrebu da ih razumemo, kako sa stanovišta prednosti koje pružaju, tako i u pogledu ograničenja.

Primene

Prenos podataka između kompjutera predstavlja samo jednu oblast komunikacija. Na primer, većina ljudi je svesna da je za televizijski prenos neophodno imati antenu i kabl koji će dovesti signal u kuću. Medutim, to je samo poslednji korak u velikom svetskom komunikacionom sistemu, koji je nastao 1962. godine uvođenjem Telstara, komunikacionog satelita dizajniranog za prenos televizijskih i telefonskih signala izmedu Sjedinjenih Američkih Država i Evrope. Telstar je pokazao da je prenos informacija izmedu kontinenata i tehnološki izvodljiv i ekonomski opravdan.

Danas se televizijski signali pomoću preko brojnih komunikacionih satelita. Na slici 1.2 prikazan je klasičan sistem. Predajnik na jednom delu sveta šalje signal do satelita u orbiti, koji prenosi taj signal do prijemnika na drugom kraju sveta. Signali se od prijemnika šalju do emisionih tornjeva i prenose lokalno pomoću frekvencije koju je odobrio FCC (Federal Communications Commision). Antena prima signal i prenosi ga do televizijskog uredaja u našim domovima.

Televizijske antene su sve ređa pojava u današnje vreme, jer se mnogi ljudi pretplaćuju na usluge kablovske televizije, koja signale dovodi direktno u domove pomoću optičkih fiber kablova i koaksijalnih kablova. Osim toga, mnogi ljudi kupuju sopstvene satelitske antene i direktno primaju satehtske signale.

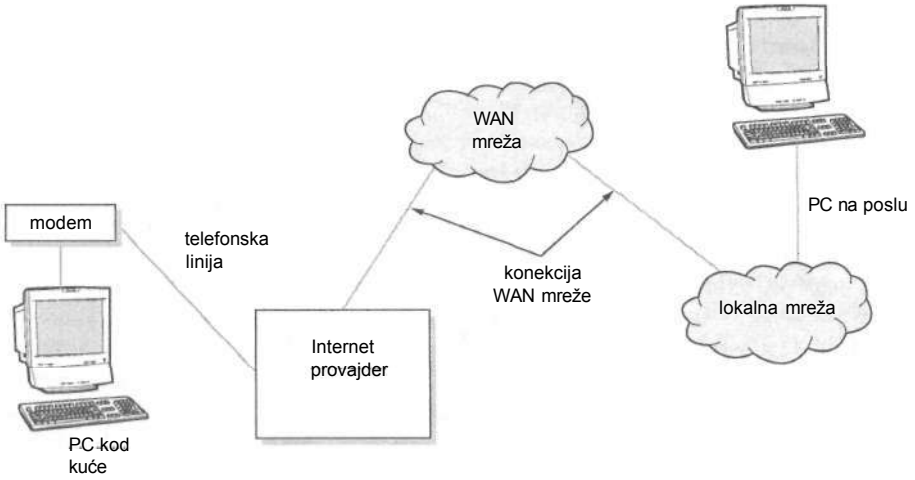


SLIKA 1.2 *Prijem televizijskih signala*

Ostale varijante korišćenja komunikacija uključuju lokalne mreže (LAN - local area networks) i mreže šireg geografskog područja (WAN - wide area networks), sisteme koji većem broju kompjutera omogućavaju da komuniciraju na kraćim (LAN), ili većim (WAN) rastojanjima. Kada se povežu, korisnici mogu da šalju, ili primaju fajlove sa podacima, mogu da se loguju na udaljene kompjutere, da šalju poštu (email - elektronsku poštu), ili da se povezuju na World Wide Web. Zahvaljujući emailu, bilo koja osoba može da pošalje privatne, ili poslovne poruke, tabelarne obrasce, baze podataka, pa, čak, i porodične fotografije sa jednog kompjutera na drugi. Sistem za email smešta poruke na disku kompjutera, tako da drugi korisnik može da ih pročita.

Zbog neverovatno povećanog korišćenja emaila, koji poruke šalje i prima elektronskim putem, neki ljudi predvidaju da će u nekoj budućnosti on zameniti klasičnu poštu. To se neće desiti baš u najbližoj budućnosti, mada danas ogroman broj ljudi koristi email, kako u poslovne, tako i u privatne svrhe, zahvaljujući sve većoj prisutnosti Weba.

Pomoću emaila je moguće poslati poruku na udaljenu lokaciju iz privatnosti doma; na slici 1.3 ilustrovano je jedno moguće uredenje. Osoba sa PC-jem i modemom može da pristupi svom Internet provajderu preko telefonske linije, kablovskog servisa, ili, čak, satelitske antene. Taj kompjuter se povezuje na WAN mrežu, koja omogućava slanje poruke širom zemlje, ili, čak, do drugih zemalja i eventualno do udaljenog Internet provajdera, ili možda kompjutera u kompanijinoj mreži preko LAN-a. Rezultat je elektronski transfer izmedu dve tačke, koje se potencijalno nalaze na velikim udaljenostima.



SLIKA 1.3 *Konekcije elektronske pošte*

Slede kratki opisi dodatnih komunikacionih primena. Neke od ovih tema detaljnije ćemo obraditi u kasnijim poglavljima.

- **Faksimil mašine (faks)** Faks mašina kreira elektronski ekvivalent slici na parčetu papira, pa tu sliku šalje preko telefonskih linija. Faks mašina na drugom kraju ponovo kreira sliku sa originalnog papira. Faks se koristi za slanje pisama, grafikona i dijagrama za svega par minuta, ili, čak, za nekoliko sekundi.

- **Glasovne i video komunikacije** **LAN** mreže su originalno korišćene za povezivanje PC-ja i ostalih uredaja prvenstveno radi transfera podataka i softvera. Cesto su komunikacioni sistemi razvijani isključivo radi prenosa glasa i video slika.

  Neke kompanije su imale sopstvene telefonske sisteme, ili privatne centrale (PBX - private branch exchange), o kojima će više reči biti u Poglavlju 4. Video komunikacije mogu da se koriste za puštanje video zapisa, ili za prijem videa iz spoljašnjeg izvora i prenos signala u okviru kompanije, ili organizacije. Video komunikacije imaju specijalne potrebe, jer obično zahtevaju prenos 30 slika u sekundi, a za svaku sliku je neophodna velika količina informacija da bi bila sačuvana kristalno čista sa pravim bojama. Medutim, sa pojavom novih tehnologija koje koriste gigabitske brzine (milijardu bitova u sekundi), ovakav prenos postaje sasvim uobičajeni deo saobraćaja u okviru LAN mreže. Time je otvoren potpuno novi svet mogućih aktivnosti korišćenjem PC-ja i LAN okruženja. Pomoću slušalica sa ugradenim mikrofonom svaka osoba može da se priključi na PC, izabere neki broj telefona i inicira konverzaciju sa osobom na udaljenom telefonu. Mini kamere postavljene na monitoru PC-ja mogu da prenose slike osobe koja govori. Digitalno kreiranje slika se koristi i za prenos video slika ka korisnicima PC-ja. To ima brojne primene. Na primer, kompanija može da sponzoriše niz programa za obuku. Odeljenje koje je angažovano može da objavi da će prenositi video snimak sa uputstvima preko odgovarajućeg nosača i da će ga održavati u određenom vremenskom periodu.

Zainteresovane osobe treba samo da koriste PC softver za selektovanje kanala u to vreme i moći će da prate emisiju u vreme kada se emituje.

- **Mobilni telefoni** Telefonski sistem je nesumnjivo najrasprostranjeniji komunikacioni sistem. **Medutim,** sve do 60-ih godina prošlog veka učesnici u komunikaciji su se morali fizički povezivati. U to vreme telefonski sistem je počeo da koristi satelite i mikrotalasne tomjeve za slanje signala. Ipak, i u to vreme osobe koje učestvuju u razgovoru morale su fizički da se vezuju na lokalne centrale. To je promenjeno pronalaskom mobilnih (celularnih) telefona, uredaja koji se povezuju na telefonski sistem preko radio talasa. Tako je ljudima omogućeno da pozivaju druge brojeve iz svojih automobila, dok su na pauzi za ručak, na utakmici, ili, čak, iz udaljenih delova zemlje - iz bilo kog mesta na kome je moguća komunikacija sa predajnim i prijemnim tornjevima. Mobilni telefoni su korisnicima omogućili i pristup Webu, kao i mogućnost slanja tekstualnih poruka. Detaljnije ćemo ih predstaviti u odeljku 4.2.

- **Informacioni servisi** Oni koji imaju PC i modem mogu da se pretplate na različite informacione servise. Oglasne table (banke podataka) omogućavaju besplatnu razmenu nekih softverskih proizvoda, fajlova i drugih informacija. Ostali servisi korisnicima omogućavaju uvid u berzanske izveštaje, elektronske transakcije, ili proučavanje rasporeda avionskih letova i rezervisanje karata. Pretraživačke mašine na Webu omogućavaju pretraživanje baza podataka za dokumente u kojima se nalazi zadata ključna reč, ili na osnovu zadate tematske oblasti. Korisnicima se vraćaju linkovi, tako da pomoću jednog klika mišem mogu da pristupe željenim informacijama. Diskusione grupe (newsgroups) omogućavaju pojedincima da postavljaju pitanja i dobijaju odgovore o nekim konkretnim temama. Ovo je postao značajan resurs za ljude koji traže tehničke savete u vezi razlićitih softverskih paketa, ili, u stvari, u vezi bilo čega.

- **E-komerc** Internet i razvoj različitih programskih alatki promenili su način na koji funkcionišu brojne kompanije. Naručivanje sa udaljenih lokacija nije više nikakva novina, jer su Ijudi decenijama koristili kataloge za naručivanje različitih proizvoda. Medutim, e-komerc aplikacije su dovele do pojave ogromnog broja sajtova koji se "takmiče" za svaki dolar potencijalnih korisnika. Skoro sve - od knjiga, CD-ova, odeće i nekada popularnih medvedića (Beanie Babies), do automobila - može da se kupi povezivanjem na sajt, popunjavanjem formulara (naravno, u njemu navodite i broj svoje kreditne kartice) i potvrdom unosa. U decembru 2002. godine grad Bridgeville u Severnoj Kaliforniji, zajednica sa dugom tradicijom logovanja na Internet, imao je najveću ponudu na eBayu, od otprilike 1,8 miliona dolara. Iako to ne može u potpunosti da se uporedi sa kupovinom Luizijane, sigurno je promenilo moguće aspekte kupovine preko Interneta. Mnogi ljudi ovakav način kupovine smatraju najprikladnijim za poslovanje; naravno, ima ih onih koji sa manje entuzijazrna gledaju na sve ovo. U jednoj anketi medu studentima na pitanja u čemu je najveća prednost kupovine preko Interneta i koji je najveći nedostatak kupovine preko Interneta najčešći odgovor je glasio: "Ne morate da razgovarate ni sa kim."

- Peer-to-peer umrežavanje Peer-to-peer umrežavanje je umrežavanje u kojem grupa kompjutera može medusobno da komunicira bez posredovanja centralizovanog servera. Ova tehnologija je privukla pažnju pre nekoliko godina kada je zbog Napstera postavljeno pitanje zaštite autorskih prava (copyright) u vezi razmene muzičkih fajlova. Mnogi korisnici se oslanjaju na peer-to-peer umrežavanje za interaktivno igranje igara preko Intemeta, a servisi kao što je Kazaa omogućavaju korisnicima da dele audio fajlove, video klipove, pa, čak, i cele filmove (često i pre nego što budu predstavljeni širokoj publici).

"Otvorena" pitanja

Novi tehnološki napredak je "otvorio" brojna pitanja koja je bilo neophodno ozbiljno razmotriti. Na primer, u prethodnoj diskusiji često smo koristili reč povezivanje i njene različite oblike. Ali, kako se povezujemo? Šta koristimo da bismo uspostavili konekciju? Da li koristimo žicu, kabl, ili optički fiber? Možemo li da se povežemo i bez njih? U Poglavlju 2 su predstavljene razne opcije.

Komunikacione tehnologije su poput planiranja saobraćaja. Putevi omogućavaju da stignete do željenog odredišta, a moraju da budu sposobni da izdrže veliki saobraćaj, posebno u velikim gradovima. Projektanti moraju da pronađu ravnotežu između toka i cene. Autoput sa 10 traka koji kruži kroz grad može da obezbedi bolji tok saobraćaja od autoputa sa šest traka, ali da li su dodatne trake vredne povećanja cene autoputa? Odgovor je verovatno porvrdan ako je reč o većim, a negativan ako je reč o manjim gradovima. Situacija je slična i kod komunikacionih sistema. Oni moraju da podrže prenos određene količine informacija, ali sama količina zavisi od konkretne primene. Količina informacija koju treba da prenesemo određuje način povezivanja uređaja. U Poglavlju 10 predstavićemo različite načine za povezivanje uredaja.

Kada izaberemo način povezivanja, moramo da uspostavimo pravila komunikacije. Gradske ulice moraju da imaju saobraćajne znakove i pravila za kontrolu saobraćaja. Isto važi i za komunikacione sisteme. Bilo da je primarni medijum kabl, bilo da ste se odlučili za bežični prenos, morate da znate koliko će izvora primati poslate informacije. Potrebno je uspostaviti neka pravila koja će sprečiti koliziju poruka, ili će definisati postupak koji se primenjuje u slučaju kolizije.

Lakoća korišćenja je sledeći aspekt. Većina ljudi neće koristiti neku tehnologiju ako nije laka za upotrebu. Na primer, mnogi kupci video rekordera nikada nisu naučili kako da ih programiraju, bar dok se nije pojavio "VCR plus". Sada se mnogi video rekorderi mogu programirati glasom. Da bi komunikacioni sistem, ili mreža imali mogućnost funkcionisanja i daljeg razvoja, informacije moraju biti lako dostupne. Medutim, u kojoj meri želimo da budu dostupne? Da li svako može da vidi, na primer, informacije o uplatama u penzioni fond, ili o investicijama?

Komunikacioni sistemi moraju da budu bezbedni. Moramo da shvatimo da lakoća razmene mformacija omogućava neautorizovanim licima zloupotrebu tih informacija. Kako informacije učiniti lako dostupnim za one kojima je pristup dopušten i sprečiti sve ostale da ih vide? Ovo je veoma teško izvesti kada neautorizovani ljudi imaju na raspolaganju brojne resurse i ulažu značajne napore za narušavanje bezbednosnih mera. Kako se osetljivost informacija povećava, mere zaštite postaju sve sofisticiranije. Ipak, ni jedan sistem nije savršeno bezbedan. Zato se u sprečavanje ovakvih aktivnosti uključio i zakon sa raznim kaznenim merama. U Poglavlju 7 detaljnije ćemo obraditi oblast zaštite.

Čak i ukoliko uspemo da rešimo sve ove probleme i upravljamo povezanim kompjuterima na najefikasniji, najisplativiji i najbezbedniji način, sa lakim transferom informacija, ostaje jedan problem: nisu svi kompjuteri kompatibilni. U nekim situacijama prenos informacija sa jednog kompjutera na drugi nalik je prelasku sa jednog automobila na drugi. Ako je reč o dva "ford escorta" proizvedena iste godine, to će biti jednostavno, ali ako je jedan "escort", a drugi "grand prix", imaćete problema.

Otvoreni sistemi su oblast koja privlači veliku pažnju. Ako su potpuno implementirani dopuštaju razmenu informacija izmedu dva povezana kompjutera. Zbog različitosti medu sistemima, ovo nije trivijalan postupak. U poslednjih nekoliko godina učinjen je ogroman korak napred u postizanju ovog cilja. U odeljku 1.4 predstavićemo otvorene sisteme i model poznat pod nazivom **Open Svstem Interconnect (OSI model).** Iako se ovaj model nije dobro pokazao za komercijalne svrhe, mnogi ga i dalje smatraju značajnim, jer opisuje strukturu komunikacionih sistema i obezbeduje značajan uvid u način zajedničkog funkcionisanja različitih komponenata komunikacionog sistema.

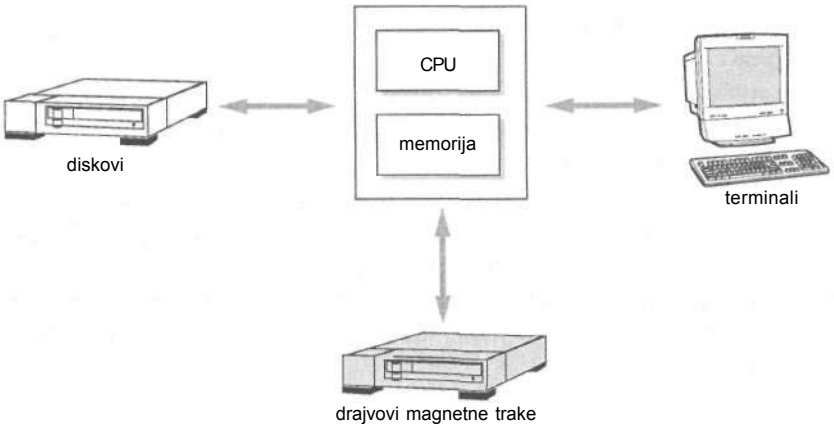
Konačno, vraćamo se na pitanje zašto proučavati komunikacione tehnologije i umrežavanje. Jednostavno, to je polje koje je imalo i imaće ubuduće neverovatan razvoj. Ljudi rnoraju da ih razumeju i da pomognu njihovo uobličavanje u budućnosti.

1.2 Kompjuterske mreže

Tokom 50-ih godina prošlog veka većina kompjutera je bila slična sa jednog aspekta. Imali su glavnu memoriju, centralnu procesorsku jedinicu (CPU) i periferije (slika 1.4). Memorija i CPU su predstavljali centralni deo sistema i imali su konekcije sa uredajima kao što su disk, magnetne trake i kompjuterski terminali. Otada su se razvile nove generacije kompjutera, kod kojih su obrada i smeštanje podataka distribuirani između više različitih uredaja. Korisnik može da pribavlja program sa jednog mesta, pokreće ga na različitim procesorima i šalje rezultate na treću lokaciju.

Sistem koji povezuje različite uredaje, kao što su PC-ji, štampači i skeneri, predstavlja mrežu. Obično svaki uredaj na mreži ima specifičnu namenu za jednog, ili više korisnika. Na primer, PC može da se nalazi na radnom stolu da bi zaposleni imao pristup potrebnim informacijama i soflveru. On može da bude rezervisan i za upravljanje diskom na kome se čuvaju deljeni fajlovi. Takav kompjuter nazivamo fajl server. Mreža često "pokriva" manju geografsku oblast i povezuje uredaje u jednoj zgradi, ili grupi zgrada. Takva mreža se naziva *lokalna mreža* (LAN - *local area network).* Mreža koja "pokriva" veargeografsku oblast, kao što je jedna država, ili svet, naziva se WAN rnreža *(wide area network).*

Većina mreža uključuje veći broj ljudi koji koriste PC-je i svi ti korisnici mogu da pristupaju raznim štampačima, ili serverima.



SLIKA 1.4 *Komunikadoni uređaji u kompjuterskom sistemu*

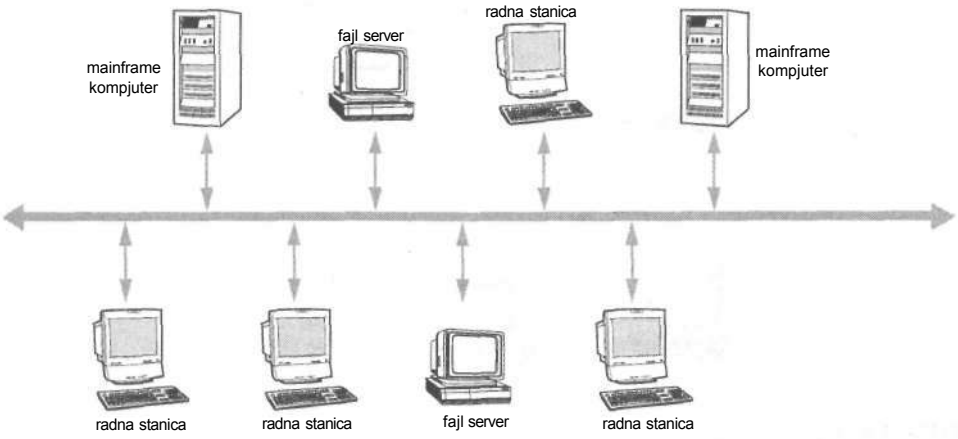
Postojanje velikog broja korisnika sa mogućnošću postavljanja zahteva za odredenim informacijama neminovno dovodi do konflikata. Zbog toga, uredaji moraju da se povežu tako da se obezbedi ispravan transfer informacija za sve učesnike u komunikaciji. Kao analogiju, ponovo možemo da iskoristimo primer ulica u velikom gradu. Kada vozi samo jedna osoba, nije mnogo bitno gde se ulice nalaze, koje su jednosmerne, gde se nalazi saobraćajna signalizacija, ili kako se vrši sinhronizacija. Međutim, kada na ulicama imate hiljade automobila u jutarnjim časovima, loš raspored može da dovede do zagušenja koja uzrokuju ozbiljna kašnjenja. Isto važi i za kompjuterske mreže. Moraju da se povežu tako da se omogući prenos podataka izmedu većeg broja korisnika, sa malim, **ili** bez ikakvog kašnjenja. Strategiju povezivanja nazivamo **mrežna topologija.** Izbor topologije zavisi od tipova uredaja i od potreba korisnika. Ono što može dobro da funkcioniše za jednu grupu može da bude veoma loše za neku drugu.

Topologija zajedničke magistrale

Na slici 1.5 prikazana je tradicionalna topologija zajedničke magistrale (ili, jednostavno, topologija magistrale), koja povezuje uređaje kao što su radne stanice, mainframe kompjuteri i fajl serveri. * Oni komuniciraju preko jedne magistrale (na primer, koaksijalnog kabla). Tradicionalni pristup obezbeduje interfejs za svaki uredaj pomoću koga se magistrala osluškuje i ispituje se saobraćaj na njoj. Ako interfejs utvrdi da su podaci namenjeni uredaju koji ih trenutno opslužuje, podaci se čitaju sa magistrale i prenose do odgovarajućeg uredaja. Slično tome, ako uredaj treba da prenese neke podatke, kola u interfejsu "osluškuju" kada je magistrala slobodna i tada započinju prenos podataka. Ovo nije ništa drugačije od čekanja na rampi za uključivanje na autoput u vreme saobraćajnog špica - proveravate kada je pogodan trenutak da se "ubacite", u zavisnosti od toga da li vozite manji automobil, ili veliki kamion.

Ponekad se dešava da dva uredaja istovremeno pokušavaju da prenesu podatke. Oba detektuju odsustvo saobraćaja i započinju prenos, ne registrujući prenos drugog uredaja.

* U Poglavlju 10 su predstavljeni alternativni načini za implementiranje topologije zajedničke magistrale korišćenjem uredaja kao što su komutatori i habovi.



SLIKA 1.5 *Topologija zajedničke magistrale*

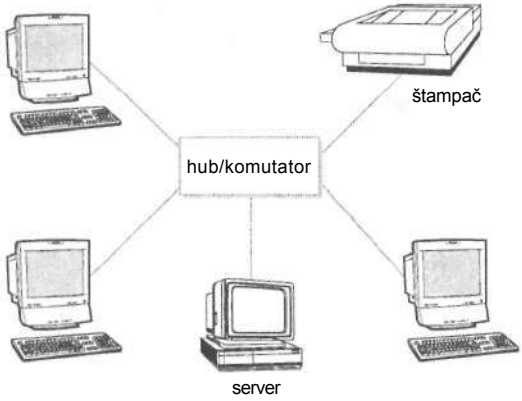
Rezultat je kolizija signala. Dok prenose podatke, uredaji nastavljaju osluškivanje magistrale i detektuju šum koji nastaje zbog kolizije. Kada uredaj detektuje koliziju, prestaje da prenosi podatke, čeka nasumice izabrani period i ponovo pokušava da prenese podatke. Ovaj proces, poznat pod nazivom **Carrier Sense Multiple Access with Collision Detection (CSMA/CD),** predstavljen je detaljno u poglavljima 4 i 9, zajedno sa ostalim načinima pristupa zajedničkom medijumu.

Primer mreže sa zajedničkom magistralom (i originalni standard za LAN mreže) je Ethernet. Njegova originalna konfiguracija koristi zajedničku magistralu onako kako smo je opisali; medutim, najnovije promene u tehnologiji obezbedile su brojne načine za povezivanje Ethernet uređaja, a da se, pri tom, i dalje sačuva logika magistrale. U Poglavlju 9 detaljnije ćemo predstaviti različite verzije Etherneta. Bez obzira na konkretnu implementaciju, glavna prednost Etherneta je mogućnost lakog dodavanja novih uredaja na mrežu.

Topologija zvezde

Sledeće uobičajeno uredenje je **topologija** zvezde (slika 1.6).* Koristi centralnu komponentu koja omogućava povezivanje drugih uredaja radi međusobne komunikacije. Ovakvi uredaji se obično nazivaju hubovi (hubs), ili komutatori (switches); razlike izmedu njih objasnićemo u Poglavlju 10. Kontrola je centralizovana: ako uredaj želi da komunicira, to može da izvede samo pomoću centralnog komutatora. Taj komutator usmerava podatke do njihovog odredišta.

* Topologija zvezde može da se posmatra i kao hijerarhijska topologija kod koje centralni čvor igra ulogu "korena" u stablu. U Poglavlju 10 pokazaćemo kako se veći broj različitih uređaja može povczati na jedan komutator, ili hub, tj. na uredaj koji obezbeduje hijerarhjjsko povezivanje.



SLIKA 1.6 *Topologija zvezde*

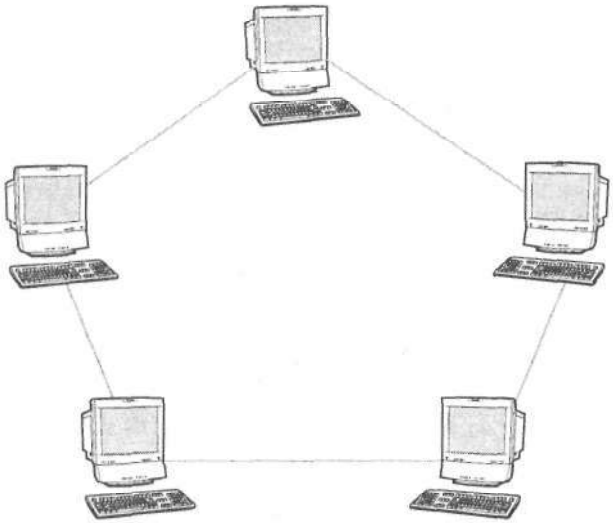
Centralizacija obezbeduje fokusiranje odgovomosti u jednoj tački, što je prednost topologije zvezde. Kod prvih mreža topologija magistrale je imala neke prednosti u poređenju sa topologijom zvezde. Nedostatak centralnog uređaja je olakšavao dodavanje novih uređaja, jer ni jedan uredaj nije morao da bude "svestan" ostalih uredaja na mreži. Osim toga, kvar, ili uklanjanje jednog uredaja na mreži sa magistralom nisu izazivali prestanak rada mreže. Kod topologije zvezde kvar na centralnom komutatoru prekida konekciju. Medutim, sa promenom tehnologije i razvojem pouzdane opreme stvoreni su tehnički i ekonomski uslovi za primenu topologije zvezde u većim topologijama.

Topologija prstena

Kod **topologije prstena** (slika 1.7) uređaji se povezuju kružno. Svaki uredaj komunicira direktno i jedino sa svojim "susedima". Ako "želi" da komunicira sa udaljenim uredajem, on šalje poruku koja se prosleđuje preko svih ostalih uredaja koji se nalaze izmedu njih.

Mreža u obliku prstena može da bude jednosmema i dvosmerna. Pod **jednosmernom mrežom** podrazumeva se mreža kod koje se sav prenos odvija u istom smeru (na primer, na slici 1.7 koristi se smer kretanja kazaljki na časovniku). U tom slučaju svaki uređaj može da komunicira samo sa jednim "susedom". Kod dvosmernih mreža prenos podataka može da se vrši u bilo kom smeru i uređaj može direktno da komunicira sa oba "suseda".

Prva topologija prstena je bila IBM-ova Token Ring mreža, koja je korišćena za povezivanje PC-ja u jednoj kancelariji, ili odeljenju. Kod token ring mreže komunikacija se koordinira prosleđivanjem tokena (preddefinisane sekvence bitova) izmedu svih uredaja u prstenu. Uredaj može nešto da pošalje samo kada primi token. Tako aplikacije sa jednog PC-ja mogu da pristupaju podacima smeštenim na drugim kompjuterima (fajl serverima) bez učešća posebnog centralnog uredaja koji koordinira komunikaciju. Nedostatak topologije prstena je to što je komplikovana sa stanovišta održavanja. Na primer, šta se dešava ako je token izgubljen, ili je oštećen? Svi uredaji koji traže token neće moći da ga dobiju i prenos podataka neće biti moguć. Postoje načini za rešavanje ovakvih problema i njih ćemo predstaviti, zajedno sa ostalim aspektima token ring mreža, u odeljcima 4.7 i 9.S. Topologija prstena ima i svoje prednosti.



SLIKA 1.7 *Topologija prstena*

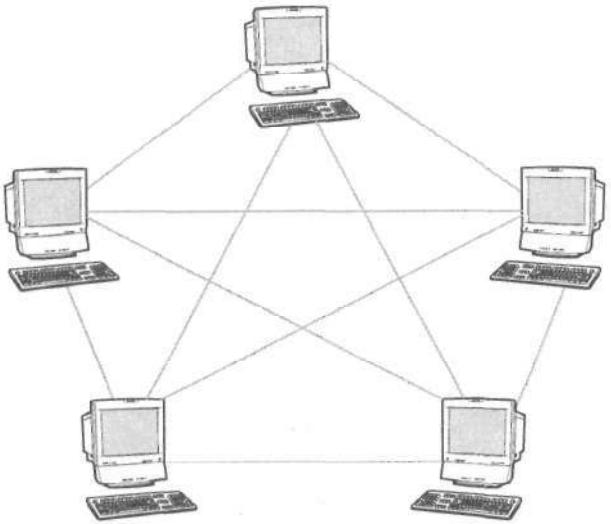
Na primer, činjenica da uređaj mora da čeka na token onemogućava istovremeni prenos iz više uredaja, tako da su kolizije nemoguće. To je karakteristika Ethernet protokola (predstavićemo ga u odeljku 4.7 i u Poglavlju 9. Kako su se mreže razvijale, prednosti Ethemeta su nadvladale njegove nedostatke; kao rezultat tog procesa, danas na tržištu LAN mreža dominiraju različite verzije Etherneta.

Potpuno povezana topologija

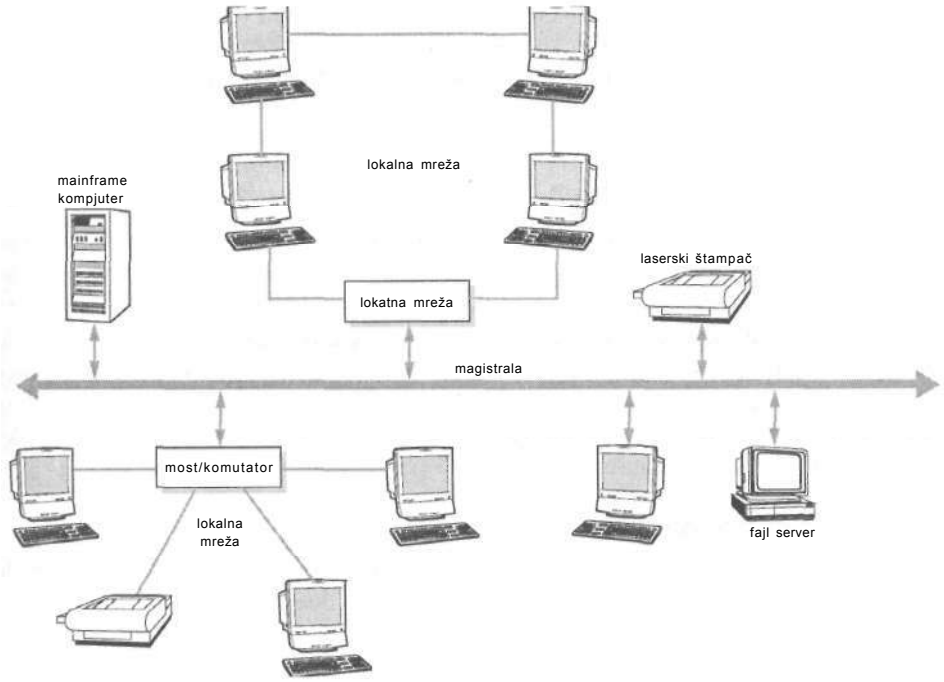
Potpuno povezana topologija (slika 1.8) ima direktne konekcije izmedu svih parova uredaja na mreži. To je ekstremni način dizajniranja mreže. Komunikacija postaje veoma jednostavna, jer nema nadmetanja za dobijanje komunikacionih linija. Ako dva uređaja žele da komuniciraju, to rade direktno, bez uključivanja ostalih uređaja na mreži. Ipak, cena direktnih konekcija izmedu svakog para uredaja je veoma visoka. Osim toga, kod ovakve konfiguracije mnoge konekcije neće biti dovoljno iskorišćene. Ako dva uredaja retko komuniciraju, fizička konekcija između njih se veoma retko koristi. U takvim slučajevima ekonomičniji pristup je indirektna komunikacija, tako da se neiskorišćene linije eliminišu.

Kombinovane topologije

Mnoge kompjuterske mreže koriste kombinacije različitih topologija. Na slici 1.9 prikazana je jedna moguća kombinacija - ima zajedničku magistialu, koja direktno povezuje više uredaja.



SLIKA 1.8 *Potpuno povezana topologija*



SLIKA 1.9 *Kombinovana topologija*

Grupe korisnika kao što su istraživači, računovođe, ili osoblje u prodajnom odeljenju imaju specijalizovane potrebe i žele zasebne LAN mreže u okviru kojih će obavljati najveći deo svog posla. Ipak, povremeno im je neophodan pristup informacijama sa dmgih LAN mreža.

U okviru mogućeg dizajna postoji nekoliko LAN mreža koje povezuju PC-je i ostale uredaje u topologiju prstena, zvezde, ili magistrale. Uređaji u okviru LAN-a komuniciraju u skladu sa pravilima topologije koja je korišćena za njihovo povezivanje. Ako PC mora da komunicira sa uredajem u drugoj LAN mreži, to radi preko mosta, ili komutatora koji povezuje te dve LAN mreže. Mostovi i komutatori su uredaji koji obezbeđuju tehnologiju za povezivanje uredaja unutar i izmedu LAN mreža; detaljnije ćemo ih. predstaviti u Poglavlju 10.

1.3 Standardi i organizacije za uspostavljanje standarda

Potreba za uvođenjem standarda

Možda ste pomislili da je primarni problem kod uspostavljanja komunikacija izmedu dva kompjutera jednostavno obezbeđivanje prenosa podataka sa jednog kompjutera na drugi. Medutim, pošto se kompjuteri često dosta razlikuju, proces prenosa može da bude mnogo složeniji, recimo poput prelaska sa "kadilaka" na "tojotu". Svi automobili se zasnivaju na istim prindpima, ali različiti modeli imaju jedinstvene karakteristike sa različitim stilovima tako da ciljaju na različita tržišta i različite kupce; isto važi i za kompjutere, osim što korisnici mogu da ispoljavaju veću dozu fanatizma (korisnici PC-ja "mrze" Mac, a korisnici Maca "mrze" PC). Komapnije dizajniraju i proizvode kompjutere sa različitim stilovima i za različite aplikacije. Većina sledi iste opšte principe, a specifičnosti odražavaju razmišljanja i filozofiju različitih Ijudi. Kompjuteri imaju različite arhitekture, razumeju različite jezike, smeštaju podatke u različitim formatima i komuniciraju na različitim brzinama. Zbog toga, postoji velika nekompatibilnost, što značajno otežava komunikaciju.

Ova nekompatibilnost dovodi do osnovnog pitanja kako je uopšte moguće uspostaviti komunikaciju izmedu kompjutera. Oni komuniciraju koristeći model sličan onome koji se koristi, na primer, u trgovini izmedu različitih zemalja. Svi učesnici govore drugim jezikom, tako da su im neophodni prevodioci. Osim toga, moraju da se pridržavaju protokola, koji defmiše pravila i način na koji se diskusija započinje i nastavlja. Ako se svi ne pridržavaju protokola, diskusija će postati haotična. Ispravna diskusija se odvija kada svi učesnici poštuju uspostavljena pravila. Slično tome, da bi kompjuteri komunicirali, potrebni su im protokoli pomoću kojih će biti utvrdeno koji kompjuter "govori" i prevedeno to što "kaže" na druge jezike. Sledeći korak je definisanje protokola. Tu leži sledeći problem: protokoli su sjajni, ali će se diskusija, ako učesnici slede različite protokole, odvijati kao da i nema protokola. Ako se ljudi dogovore o zajedničkom protokolu, to postaje standardni protokol i svako može da ga koristi. Nažalost, ovo pomalo liči na situaciju u kojoj bi svi trebalo da usvoje istu arhitekturu, a znamo da se to nikada ne može desiti. Usaglasiti mišljenje više grupa ljudi o nečemu nije nimalo jednostavno. Različite grupe imaju razlidte ciljeve i ideje o tome koji protokol najbolje obezbeduje ispunjavanje tih ciljeva. Zbog toga su godinama nastajali i bili korišćeni razni standardi.

Postoje dve grupe standarda. **De facto standardi** postoje zbog opšte upotrebe - postali su toliko rasprostranjeni da prodavci i proizvođači prepoznaju da će sa takvim proizvodima sebi obezbediti veliko tržište. Mnogi IBM-ovi proizvodi su postali de fakto standardi. Drugi tip su standardi

koje formalno prepoznaju i usvajaju organizacije za uspostavljanje standarda na osnovu nacionalnih, ili svetskih kriterijuma. Oni koji žele da njihovo delo postane standard pišu predlog i prosleduju ga organizaciji za uspostavljanje standarda radi razmatranja. Tipično, ako predlog naide na nesumljivu i opštu podršku, organizacije za uspostavljanje standarda daju sugestije i šalju ih kreatorima radi daljih modifikacija. Nakon nekoliko krugova sugestija i modifikacija, predlog se usvaja, ili odbija. Ako se odobri, standard obezbeduje model na osnovu koga proizvodači mogu da dizajniraju nove proizvode.

Organizacije za uspostavljanje standarda

Uvodenje organizacija za uspostavljanje standarda defmitivno je uvelo red na polju komunikacija koje se razvijaju neverovatnom brzinom. Odobreno je nekoliko stotina standarda za različite aspekte komunikacija, što je, pak, dovelo do nekompatibilnosti izmedu različitih tipova uredaja. Na primer, mnogi korisnici PC-ja kupuju modem (uredaj koji omogućava slanje i prijem signala sa kompjutera preko telefonske linije) za povezivanje na kompjutere u univerzitetskim mrežama, ili mrežama kompanija, ili na Internet provajdere. Problem je što postoji desetine standarda koji opisuju različite načine za slanje i prijem signala preko telefonske linije; ako modemi koriste različite standarde, komunikacija izmedu njih nije moguća. Ipak, proizvodači su prepoznali ovaj problem i obično proizvode modeme tako da se implementiraju odredeni standardi koji zadovoljavaju zahteve tržišta. Ovaj problem je u potpunosti objašnjen u Poglavlju 3.

Na polju kompjuterskih mreža i savremenih komunikacija relevantne su sledeće organizacije:

- **American National Standards Institute (ANSI)** ANSI (www.ansi.org/) je privatna, nevladina agencija, čiji su članovi proizvoclači, korisnici i druge zainteresovane kompanije. Ima skoro 1.000 članova, a predstavlja deo ISO-a (International Organization for Standardization), koji ćemo objasniti kasnije. ANSI standardi su česti **na** brojnim poljima. Neki od konkretnih primera su Fiber Distributed Data Interface (FDDI) i Sinhrona optička mreža (SONET - Synchronous Optical Netowrk) za optički fiber. Sledeći standard (predstavićemo ga u Poglavlju 2) je American Standard Code for Information Interchange (ASCII), koji se koristi na mnogim kompjuterima za smeštanje informacija.

- **International Electrotechnical Commision (IEC)** IEC (www.iec.ch/) je nevladina agencija koja "izmišlja" standarde za obradu podataka i interkonekcije i bezbednu opremu. llključena je u razvoj Joint Photographic Experts Group (JPEG), grupe koja je "izmislila" standard za kompresovanje slika.

- **International Telecommunications Union (ITU)** raniji naziv bio je **Comite Consultatif International de Telegraphique et Telephonique (CCITT)** Engleski ekvivalent bio bi International Consultative Committee for Telephony and Telegraphy. ITU (www.itu.int/) je agencija Ujedinjenih nacija, koja ima tri sektora: ITU-R se bavi radio komunikacijama, ITU-D je razvojni sektor i ITU-T (relevantan za ovu knjigu), koji se bavi telekomunikacijama. Članovi ITU-a su razne naučne i industrijske organi-. zacije, telekomunikacione agencije, autoritativna tela za telefoniju i ISO. ITU "stoji iza" brojnih standarda za mreže **i** telefonske komunikacije. Medu opštepoznate standarde ubrajaju se standardi V **i** X serija. V serija je namenjena telefonskim komunikacijama.

U Poglavlju 3 detaljnije ćemo predstaviti neke V standarde koji opisuju način na koji modem generiše i interpretira analogne telefonske signale. X serija se bavi interfejsima mreža i javnim mrežama. Poznatiji standardi ove serije su X.25 za interfejse u mrežama sa komutacijom paketa (predstavljen je u Poglavlju 13), zatim X.400 za sisteme elektronske pošte i X.509 za digilalne sertifikate (predstavljen je u Poglavlju 7). Postoje još mnogi drugi X i V standardi.

- **Electronic Industries Association (EIA)** Članovi EIA-a (www.eia.org), koji je član ANSI instituta, su brojne elektronske firme i proizvođad telekomunikacione opreme. Primarne aktivnosti EIA asocijacije su elektronske konekcije i fizički prenos podataka izmedu različitih uređaja. Najpoznatiji standard EIA asocijacije je RS-232 (poznat i **kao** EIA-232), koji su PC-ji dugo koristili za komuniciranje sa drugim uređajima, kao što su modemi, ili štampači. Standard EIA-232 predstavićemo u Poglavlju 4.

- **Telecommunications Industry Associatioo (TIA)** TIA (www.tiaonline.org) predstavlja provajdere proizvoda komunikacionih i informacionih tehnologija i servisa na globalnom tržištu. Ta asocijacija, koju akredituje ANSI, bavi se razvojem standarda za široki opseg komunikacionih proizvoda. Neki od primera su optički kablovi, žičani provodnici i konektori koji se koriste u lokalnim mrežama.

- **Internet Engineering Task Force (IETF)** IETF (www.ietf.org) je internacionalna zajednica, čiji su članovi dizajneri mreža, proizvodači i istraživači, koji kao zajednički interes imaju uspostavljanje stabilnog funkcionisanja Interneta i njegov razvoj. Podeljena je u radne grupe koje se bave raznim aspektima Interneta, kao što su aplikacije, operacije, upravljanje, rutiranje, zaštita i transportni servisi. Ove radne grupe su zadužene za razvoj i ocenu specifikacija koje treba da postanu Internet standardi. Jedan od značajnih rezultata rada IETF zajednice je sledeća generacija Internet protokola, koja je predstavljena u Poglavlju 1**1**.

- **Institute of Electrical and Electronic Engineers (IEEE)** IEEE (http://standards.ieee.oig) je najveća svetska profesionalna organizacija koju čine profesionalci u oblasti računarstva i inženjeringa. Objavljuje razne časopise, organizuje konferencije i ima grupu za razvoj standarda. Verovatno njeno najpoznatije delo na polju komunikacija su Project 802 LAN standardi. Standardi 802, koji su predstavljeni u Poglavlju 9, definišu komunikacione protokole **za** mreže sa topologijom magistrale, prstena i bežične mreže.

- **International Organization for Standardization (ISO)** ISO (www.iso.ch) je svetska organizacija, koju čine tela za uspostavljanje standarda iz različitih zemalja, medu kojima se nalazi i ANSI iz SAD. Jedna od najznačajnijih aktivnosti ISO organizacije je rad na otvorenim sistemima, koji definišu protokole koji omogućavaju komunikaciju nezavisnu od arhitekture kompjutera. Opštepoznati model je Open System Interconnect, organizovan u sedam slojeva.

Neki od nas su verovali da će se OSI model koristiti u svim budućim komunikacijama, što je, medutim, nakon razvoja Interneta i Web aplikacija, malo verovatno. Ipak, često se proučava kao model za uslojavanje protokola. OSI model predstavićemo u narednom odeljku.

- **National Institute of Standards and Technology (NIST)** Ranije poznat kao National Bureau of Standards (NBS), NIST (www.nist.gov) je agencija Ministarstva trgovine SAD. Uspostavlja standarde koje federalna vlada koristi prilikom kupovine opreme. Osim toga, razvija standarde za razne fizičke veličine, kao što su vreme, dužina, temperatura, radioaktivnost i radio frekvencije. Jedan značajan standard za bezbedne aplikacije je Data Encryption Standard (DES), metod šifrovanja, ili promene informacija u formu koja ne može da se razume. DES standard je izraden u čipovima koji se koriste u komunikacionim uredajima. Standard je izuzetno složen i kontraverzan; neki veruju da ga je Nacionalna agencija za bezbednost (National Security Agency) namerno oslabila da bi se sprečilo korišćenje tehnika za šifrovanje koje ne može da se dešifruje. Pošto je standard "razbijen", više se ne koristi kao funkcionalni metod šifrovanja, ali postoje neke druge tehnike koje se zasnivaju na ovom standardu. DES detaljnije predstavljamo u Poglavlju 7.
- **International Business Machines (IBM)** Iako nije organizacija za uspostavljanje standarda, naveli smo je zbog ogromnog udela u nastajanju de fakto standarda. Istaknuti primeri su System Network Architecture (SNA) i Extended Binary-Coded Decimal Interchange Code (EBCDIC). SNA je model protokola koji omogućava komunikaciju IBM kompjutera i opreme. Nastao je pre OSI modela i danas nema opštu upotrebu, mada je po mnogo čemu sličan OSI modelu. EBCDIC kod (prikazaćemo ga u Poglavlju 2) predstavlja alternativu ASCII kodu za smeštanje podataka i obično se koristi na IBM mainframe kompjuterima (iako IBM-ovi PC-ji obično koriste ASCII kod).

Ove organizacije nisu jedina tela koja mogu da uspostavljaju standarde, ali su svakako najmerodavnije u oblasti savremenih komunikacionih tehnologija i mreža.

1.4 Otvoreni sistemi i OSI model

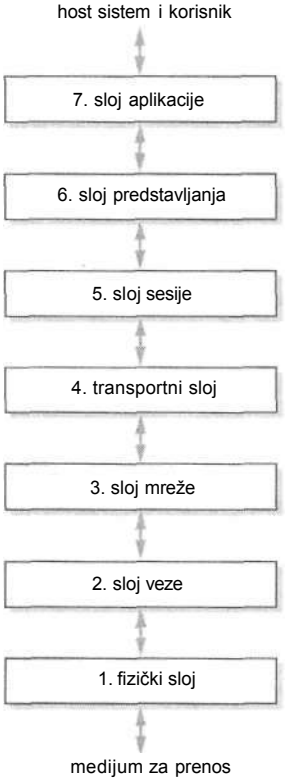
Rekli smo da protokoli omogućavaju komunikaciju izmedu nekompatibilnih sistema. Kada imamo dva specifična sistema, definicija protokola je sasvim jednostavna. Problem postaje ozbiljniji i teži dok se povećava broj različitih tipova sistema. Skup protokola koji omogućava komunikaciju izmedu bilo koja dva sistema, bez obzira na njihovu arhitekturu, naziva se *otvoreni sistem*. ISO se bavi problemom obezbedivanja komunikacije izmedu više uređaja, a razvio je Open System Interconnect (OSI) model. Da je potpuno razvijen, omogućio bi komunikaciju izmedu bilo koja dva povezana kompjutera.

OSI model nije doživeo komercijalni uspeh, jer su ga "zasenili" protokoli na kojima se zasniva Internet. Zbog toga, neki smatraju da je OSI model "mrtav" i da se više ne koristi, a drugi, pak, da, iako nije imao komercijalnog uspeha, ipak definiše radni okvir u kome je moguće proučavati i razumeti protokole. Specijalno, omogućava proučavanje širokog spektra komunikacionih protokola i razumevanje njihovih medusobnih odnosa.

Kao i svi ostali složeni programi, ili sistemi, ima tačno definisanu strukturu na kojoj su izgradene komponente. Osim toga, razumevanje pojedinačnih komponenata sasvim se razlikuje od razumevanja načina na koji te komponente zajedno funkcionišu da bi bio omogućen efikasan komunikacioni sistem. Zbog toga, dajemo opšti prikaz OSI modela.

OSI model ima sedam slojeva (slika 1.10). Svaki sloj izvršava specifične funkcije i komunicira sa slojevima koji se nalaze direktno iznad i ispod njega. Viši slojevi su zaduženi za korisničke servise, aplikacije i aktivnosti, dok se niži slojevi bave stvarnim prenosom informacija.

Svrha uslojavanja protokola je razdvajanje specifičnih funkcija, tako da njihova implementacija bude transparentna sa stanovišta drugih komponenata. Osim toga, organizovanje po slojevima omogućava nezavisno dizajniranje i testiranje svih komponenata. Na primer, sloj veze i fizički sloj izvršavaju zasebne funkcije. Fizički sloj obezbeduje servise za sloj veze. Sloj veze uopšte ne vodi računa o tome kako se servis izvršava, već je bitno samo da li je servis izvršen.

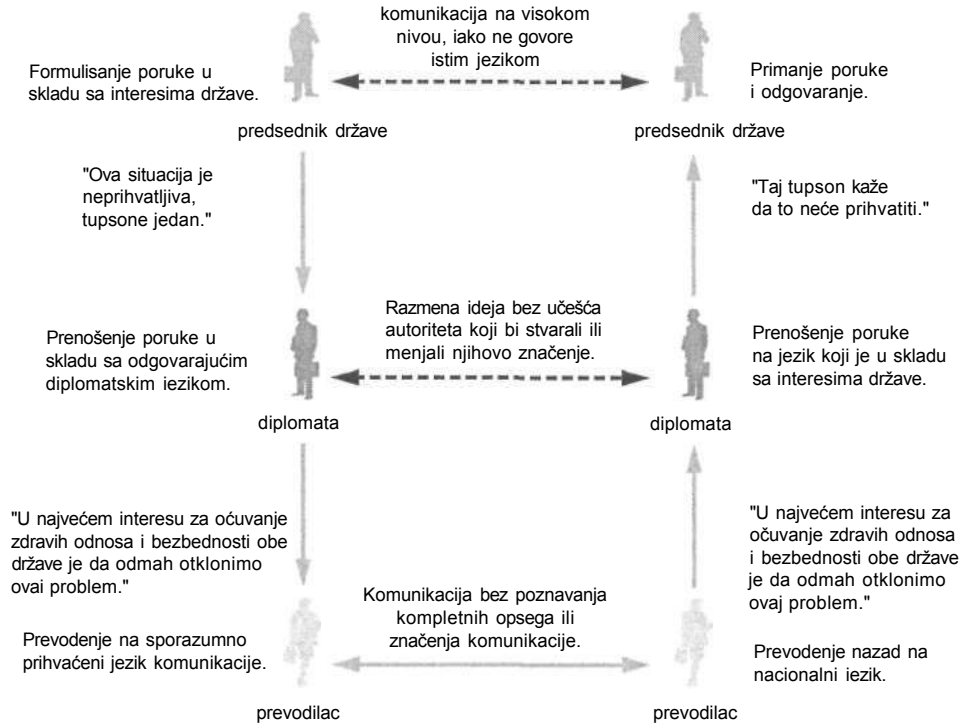


SLIKA 1.10 *Slojeviti OSI model ISO organizacije*

Ako dode do nekih promena u implementaciji fizičkog sloja, to neće uticati na sloj veze (i sve ostale više slojeve). Ovakav odnos važi izmedu bilo koja dva susedna sloja, a obezbeduje apstraktnu analogiju koja se može primeniti u raznim softverskim dizajnima.

Možemo da iskorislimo poređenje sa sastankom predsednika dve države. Svaki lider izlaže svoja razmišljanja, mada se te ideje moraju preneti na odgovarajućem diplomatskom jeziku da bi se izbegle eventualne uvrede. Osim toga, ako govore drugačijim jezikom, jedan jezik mora da se izabere kao primarni oblik komunikacije. Na slici 1.11 ilustrovan je mogući troslojni protokol koji uključuje pokušaj razrešavanja krize. Jedan lider odmah ističe da on neće tolerisati nastalu situadju. Diplomata prenosi poruku u manje pretećem tonu, a prevodilac prevodi poruku na izabrani jezik. Na drugoj strani, drugi prevodilac prevodi poruku u specifični jezik te strane. Diplomata prima pomku i govori predsedniku države šta to, u stvari, znači.

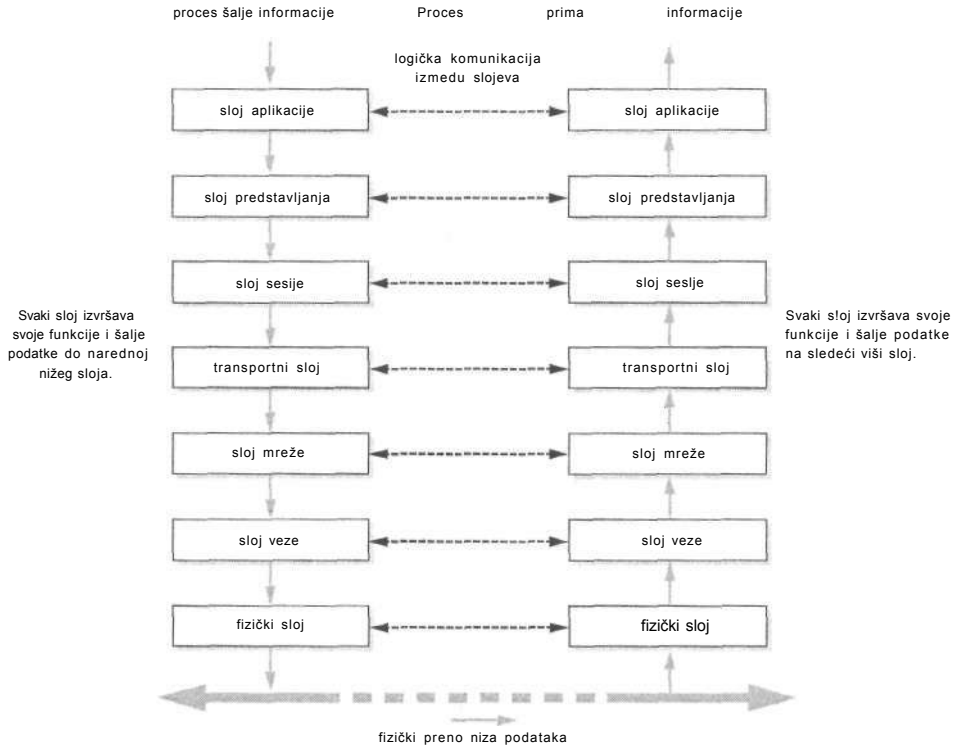
U suštini, predsednici država komuniciraju direktno, iako se poruke, zapravo, prenose posredstvom drugih osoba. OSI model funkcioniše na sličan način. Najniži je fizički sloj, koji je zadužen za stvarni prenos podataka. Najviši sloj je zadužen za kompjuterski sistem koji je povezan na mrežu. Svaki sloj u okvim modela odgovara različitim nivoima apstrakcije u toku komunikacije i definiše određene funkcije i protokole.



SLIKA 1.11 *Komunikacioni protokol između predsednika dve države*

Dve inače nekompatibilne strane, ako koriste OSI model, mogu medusobno da komuniciraju (slika 1.12). Logično, svaki sloj komunicira direktno sa istim slojem na drugoj strani. Fizički, svi slojevi komuniciraju sa slojevima koji se nalaze odmah ispod, ili iznad njih. Kada proces želi da pošalje informacije, počinje da ih prepušta sloju aplikacije. Taj sloj izvršava svoje funkcije i šalje podatke na sloj predstavljanja. Hakon toga, izvršavaju se funkcije sloja predstavljanja i podaci se prosleduju do sloja sesije. Ovaj proces se nastavlja sve dok se podaci ne prenesu do fizičkog sloja, koji, u stvari, prenosi podatke.

Proces se na prijemnom kraju izvodi obrnutim redosledom. Najpre fizički sloj prima niz bitova i predaje sloju veze. Na sloju veze se izvode odredene funkcije, a zatim se podaci šalju do sloja mreže. Ovaj proces se nastavlja sve dok se ne stigne do sloja aplikacije, koji eventualno proslednje podatke do prijemnog procesa.



SLIKA 1.12 *Komunikacija ostvarena pomoću sedmoslojnog OSI protokola*

Izgleda kao da procesi direktno komuniciraju, tako što svaki sloj komunicira direktno sa identičnim slojem na drugoj strani. U stvarnosti, svi podaci se prevode u niz bitova i prenose se između fizičkih slojeva.

Ovaj proces pomalo podseća na slanje pisma, gde sa primaocom pisma komunicirate tako što adresirate kovertu i ubacujete je u poštansko sanduče. Što se Vas tiče, aktivnost je završena: komunikacija ne zavisi od toga kako se pismo usmerava, ili da li putuje kamionom, avionom, vozom, brodom, ili ga nosi golub pismonoša. Znate da će pismo stići i jednostavno možete da čekate na odgovor.

Opšti pregled modela

SIoj aplikacije, najviši sloj, radi direktno sa korisnikom, ili sa programskim aplikacijama. Napomenimo da to nije isto što i programska aplikacija. Sloj aplikacije obezbeduje korisničke servise, kao što su elektronska pošta, ili transfer fajlova. Na primer, kod protokola za transfer fajlova sloj aplikacije na jednom kraju treba da pošalje fajl direktno do sloja aplikacije na drugom kraju, nezavisno od korišćene mreže, ili od arhitektura uključenih kompjutera.

Sloj aplikacije defmiše i protokole koji omogućavaju pristup tekstualnom editoru na udaljenom serveru. Razlog za to je činjenica da razlidti tipovi editora koriste različite kontrolne sekvence za kontrolu kursora. Na primer, samo pomeranje kursora može da zahteva kursorske tastere, ili specijalne kombinacije tastera. Voleli bismo kada bi ovakve razlike bile transparentne za korisnika.

Sloj predstavljanja je odgovoran za predstavljanje podataka u formatu koji korisnik može da razume. Na primer, pretpostavimo da dva različita kompjutera koriste različite numeričke formate i formate za karaktere. Sloj za predstavljanje prevodi podatke iz jedne reprezentacije u drugu i izoluje korisnika od tih razlika. Da bi se to izvelo, sloj za predstavljanje najpre utvrduje razlike izmedu podataka i informacija. Na kraju krajeva, mreže i postoje zato da bi korisnici mogli da razmenjuju informacije, a ne nizove bitova. Korisnici ne treba da vode računa o različitim formatima, većda se koncentrišu na sadržaj informacija i ono šta one znače za njih.

Sloj predstavljanja može da obezbedi i bezbednosne mere. Može da šifruje podatke pre nego što se proslede do nižih slojeva radi transfera. Sloj predstavljanja na drugom kraju u tom slučaju dešifruje primljene podatke. Korisnik nikada i ne mora da zna da je vršena bilo kakva promena podataka. Ovo je posebno važno u WAN mrežama (koje

"pokrivaju" velika geografska područja), gde neautorizovani pristup predstavlja ozbiljan problem.

Sloj sesije omogućava aplikacijama na dva različita kompjutera da uspostave **sesiju,** ili logičku konekciju. Na primer, korisnik može da se uloguje na udaljeni sistem i da komunicira naizmeničnim slanjem i primanjem poruka. Sloj sesije pomaže koordinaciju procesa tako što se svaki kraj obaveštava kada može da šalje podatke, ili kada mora da "osluškuje". Ovo predstavlja jedan oblik sinhronizacije.

Sloj sesije je zadužen i za ispravljanje grešaka. Na primer, pretpostavimo da korisnik šalje sadržaj velikog fajla preko mreže na kojoj iznenada dolazi do kvara. Kada se funkcionalnost mreže ponovo uspostavi, da li korisnik mora da počne ponovni prenos fajla od samog početka? Odgovor je negativan, jer sloj sesije korisniku omogućava umetanje kontrolnih tačaka u dugačkom nizu. Ako mreža padne, biće izgubljeni samo podaci koji su preneti iza poslednje kontrolne tačke.

Osim toga, sloj sesije je zadužen za zatvorene operacije koje se sa stanovišta korisnika izvode kao jedinstvene transakdje. Uobičajeni primer je brisanje zapisa iz baze podataka. Iako korisnik vidi brisanje zapisa kao jedinstvenu operaciju, ona, u stvari, može da uključuje nekoliko operacija. Najpre se mora pronaći odgovarajući zapis, a zatim se pristupa brisanju promenom pokazivača i adresa i eventualnim upisom u indeks, ili heš (hash) tabelu. Ako korisnik pristupa bazi podataka preko mreže, sloj sesije mora da obezbedi da se sve operacije nižeg nivoa izvedu pre nego što počne konkretno brisanje. Ako se operacije u bazi podataka izvode u vreme kada stigne zahtev za brisanje, zbog eventualnih otkaza mreže možda će doći do narušavanja integriteta podataka, jer je moguće da se izbrišu samo neki pokazivači (setite se uvodnih predavanja na kojima ste učili nešto više o strukturama podataka, kada je bilo reči o programirna koji ne menjaju sve pokazivaće), ili može doći do brisanja zapisa, a da se, pri tom, ne izbriše referenca na taj zapis.

Cetvrti je **transportni sloj.** To je najniži sloj koji se bavi komunikacijama izmedu dva kraja (niži slojevi rade sa samom mrežom). Transportni sloj može da utvrdi koja se mreža koristi za komuniciranje. Kompjuter može da bude povezan na više mreža, koje se razlikuju po brzini, ceni i tipu komunikacije, a izbor često zavisi od više faktora. Na primer, da li su informacije predstavljene u obliku dugačkog kontinuelnog niza podataka? Hi, da li se komunikacija odvija sa brojnim prekidima? Telefonska mreža je dobra za dugačke, kontinuelne prenose podataka. Kada se konekcija uspostavi, ona se održava sve dok se ne završi prenos podataka.

Sledeći pristup podrazumeva deljenje podataka na manje pakete (podskupove podataka) i njihov naizmenični prenos. U takvim situacijama nije neophodno održavati konstantnu ko nekciju. Umesto toga, svaki paket može nezavisno da se prenosi kroz mrežu. Kada na prijemnoj strani budu primljeni svi paketi, oni se moraju ponovo sastaviti pre nego što se proslede do viših slojeva. Problem se javlja kada paketi koriste drugačije rute na putu do svog odredišta, jer ne postoji nikakva garancija da će biti primljeni istim redosledom kojim su i poslati (baš kao što ne postoji nikakva garancija da će pismo koje je poslato u ponedeljak sigurno stići pre pisma koje je poslato u utorak), ili da će uopšte i stići do svog odredišta. Ne samo da prijemna strana mora da utvrdi tačan redosled paketa, već mora da proveri i da li su stigli svi paketi.

Sloj mreže je zadužen za strategije rutiranja. Na primer, kod bidirekcione mreže sa topologijom prstena postoje dve putanje između svake dve tačke. Složenija topologija može da uključuje više različitih mta izmedu pojedinih tačaka. Koja od njih je najbrža, najjeftinija, ili najbezbednija? Koje su otvorene, a koje su zagušene? Da li cela poruka treba da se šalje istom rutom, ili se njeni delovi mogu slati nezavisno?

Sloj mreže kontroliše komunikacionu podmrežu (communications subnet), kolekciju medijuma za prenos i elemenata za komutaciju, koji su neophodni za rutiranje i prenos podataka. Sloj mreže je najviši sloj podmreže. On može da sadrži i knjigovodstveni softver za ispostavu računa kupcima. Zapamtite da mreža postoji da bi omogućila komunikaciju korisnika. Kao i u slučaju većine servisa, to neko mora da plati. Cena zavisi od količine podataka koja se prenosi i eventualno od doba dana. Sloj mreže može da rukuje takvim informacijama i da kontroliše naplatu.

Sloj veze nadgleda tok informacija izmedu susednih čvorova u mreži. Koristi tehnike za detekciju i korigovanje grešaka da bi bio obezbeden prenos bez grešaka. Ako se detektuje greška na linku, može da se zahteva novi prenos, ili, u zavisnosti od implementacije, ispravljanje greške. Osim toga, kontroliše se količina informacija koja se šalje u odredenom trenutku; isuviše mala količina informacija izaziva preterano čekanje i na predajnom i na prijemnom kraju.

Sloj veze prepoznaje formate. Podaci se često prenose u okvirima (frames), koje čine grupe bitova organizovanih u skladu sa specifičnim formatom. Sloj veze označava početak i kraj svakog odlazećeg okvira sa jedinstvenim uzorkom bitova i te uzorke prepoznaje i prilikom definisanja dolazećeg okvira. Zatim se okviri koji provereno ne sadrže greške šalju do sloja mreže.

Konačno, fizički sloj prenosi bitove podataka preko mreže. Bavi se fizičkim i električnim aspektima prenosa podataka. Na primer, da li se kao medijum koriste bakarni kabl, optički fiber, ili satelitske komunikacije? Kako se podaci mogu fizički preneti od tačke A do tačke B? **Fizički sloj** prenosi nizove bitova podataka primljene od sloja veze, bez razumevanja njihovog značenja, ili formata. Slično tome, bitovi se primaju bez ikakvog analiziranja i prosleduju se do sloja veze.

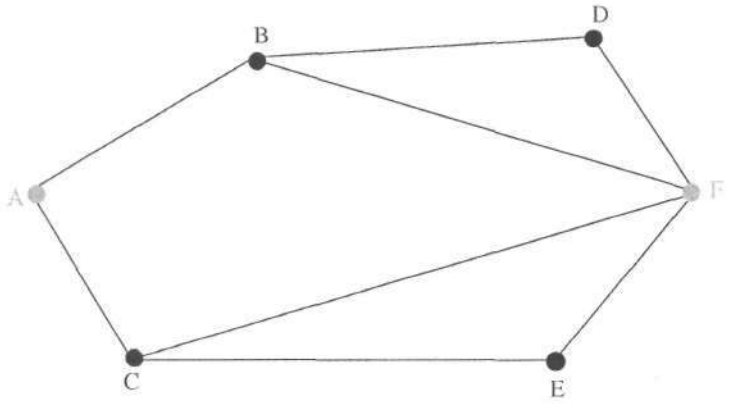
Dakle, najniža tri sloja se bave isključivo mrežnim komunikacijama. Oni zajedno obezbeduju servise za više slojeve koji su zaduženi za komunikacije između krajnjih tačaka. Oni defmišu komunikacione protokole izmedu dva korisnika, ali, pri tom, ne vode računa o detaljima nižeg nivoa u prenosu podataka. Neke mrežne implementacije ne koriste svih sedam slojeva, ili mogu da kombinuju neke funkcije iz različitih slojeva. Zapamtite da je OSI model (i to važan) i da mnogi mrežni protokoli nisu kompatibilni sa njim. Ipak, on zauzima važnu polaznu tačku u proučavanju, jer pomaže razjašnjavanje razloga zbog kojih su neke mrežne funkcije postavljene u okviru protokola. U tabeli 1.1 dat je pregled funkcija koje smo do sada predstavili.

Strategije povezivanja

Pre nego što predemo na proučavanje konkretnih slojeva, dajemo opšti pregled mrežnih operacija. Znamo da se dva kompjutera moraju povezati (žicom, optičkim fiberom, satelitskom vezom, ili nekom drugom bežičnom tehnologijom) da bi bila moguća komunikacija između njih. Način na koji informacije putuju kroz mrežu ie problem dizaina. Na primer, razmotrite mrežu sa slike 1.13.

Tabela 1.1: Pregled slojeva OSI modela

Sloj	Funkcije
7. Aplikacije	Obezbeduje elektronsku poštu, transfer fajlova i druge korisničke servise.
6. Predstavljanja	Prevodi formate podataka, šifruje i dešifruje podatke.
5. Sesije	Sinhronizuje učesnike u komunikciji, vrši oporavljanje od grešaka i zaokružuje operacije.
4. Transportni	Utvrduje mrežu i može da sakuplja i ponovo sastavlja pakete.
3. Mreže	Utvrduje rute i upravlja informacijama za naplatu.
2. Veze	Detektuje i ispravlja greške i definiše okvire.
I.Fizički	Prenosi fizičke podatke.



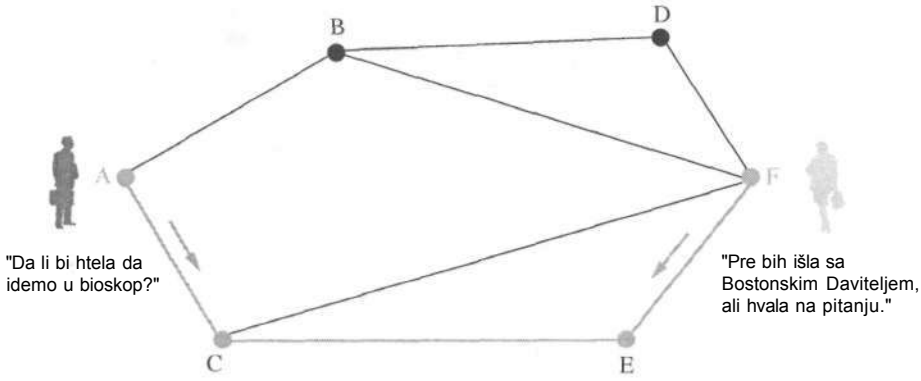
SLIKA 1.13 *Primer kompjuterske mreže*

Ako čvor A želi da komunicira sa čvorom F, kako će informacije putovati od A do F? Morate da razlikujete ovaj problem od utvrđivanja mte, ili mrežne putanje. Ako linije predstavljaju fizičke konekcije, postoje četiri rute duž kojih informacije mogu da putuju od čvora A do čvora F (možete li da ih nabrojite?). Sloj mreže utvrduje koja je ruta najbolja, ali pitanje je kako informacije putuju preko izabrane rute. Ovo se ponekad naziv *strategija povezivanja.*

Postoje tri strategije: komutacija kola **(circuit** svvitching), komutacija poruka (message switching) i komutacija paketa (packet switching). Kod komutacije kola, kada se konekcija uspostavi izmedu dva čvora, održava se sve dok je jedan od čvorova ne okonča. Drugim rečima, konekcija je rezervisana za komunikaciju izmedu ove dve strane. Komutacija kola je uobičajena u telefonskim sistemima (slika 1.14), jer se kanal koji se dodeli jednom telefonu ne može koristiti za druge telefone.

Kako ovo funkcioniše? Osoba u čvom A želi da razgovara sa osobom u čvoru F. Osoba A zahteva aspostavljanje konekcije sa osobom F. Ako je reč o telefonskoj mreži, konekcija se uspostavlja nakon biranja broja. Kod kompjuterske mreže korisnik unosi odgovarajuće komande za povezivanje na spedfičnu lokaciju. U svakom slučaju, logika u čvoru A mora da utvrdi sledeđ čvor u ruti koji vodi ka ćvoai F. Ovaj proces uključuje razne faktore, kao što su cena konekcije i raspoloživost različitih putanja. Na primer, telefonski poziv iz San Franciska u Los Anđeles normalno se ne mtira preko Majamija. Ipak, ako su linije između dva grada zagušene, konekcija može da bude indirektna - na primer, preko Sakramenta.

Na slici 1,14 čvor A je utvrdio da je za put do čvora F čvor C bolji izbor nego čvor B. Zato se čvor A povezuje na čvor C. Nakon toga, čvor C nastavlja u sličnom stilu. Može da izabere čvor F, ili može da odluči da ide preko čvora E. Na odluku utiču cena i postojeće konekcije. U ovom slučaju se čvor C povezuje na čvor E. Čvor E se konačno vezuje na čvor F. Konekcija je uspostavljena i čvor F može da bude spreman za prihvatanje poziva. Kod telefonskog sistema prihvatanje konekcije podrazumeva podizanje slušalice na prijemnoj strani i izgovaranje pozdrava.



SLIKA 1.14 *Rezervisano kolo za povezivanje čvora A sa čvorom F*

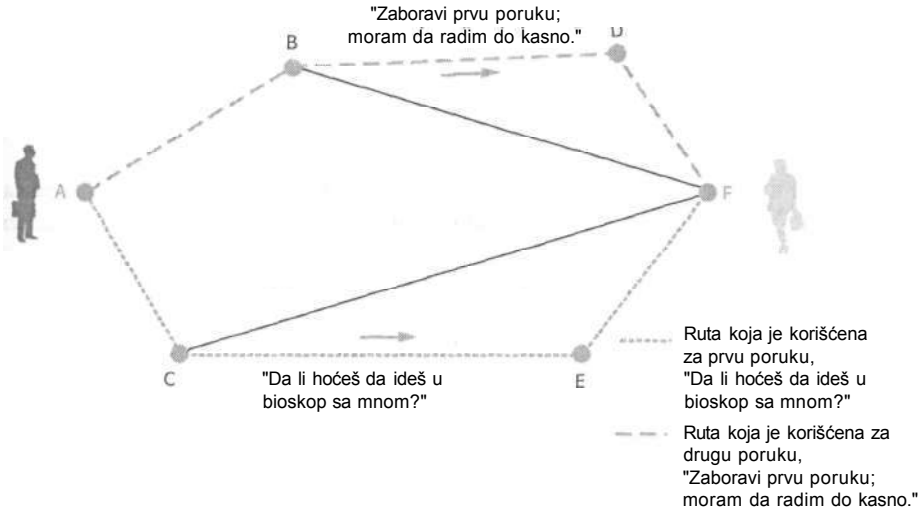
Kod kompjuterskih mreža za prihvatanje konekcije koriste se odgovarajuće komande. Ako čvor F ne odgovori (na primer, dobijete znak zauzeća, ili nema nikakvog odgovora), čvor A okončava zahtev.

Ako čvor F prihvati konekciju, razmena informacija može da otpočne. Osoba u čvoru A pita: "Da li želiš da idemo u bioskop?", a osoba u čvoru F odgovara: "Rado bih išla sa tobom, ali jutros mi je uginio kanarinac i trenutno sam u žalosti. Pozovi me neki drugi put kada nisam u gradu."

Komutacija kola zahteva utvrdivanje rute i uspostavljanje konekcije pre nego što otpočne prenos informacija. Osim toga, mreža održava konekciju sve dok je jedan čvor ne okonča. Ovakav tip komunikacije je najefikasniji kada je komunikacija izmedu dva čvora kontinuelna, tj. kada čvor A "kaže" nešto, a čvor F to "čuje" skoro trenutno, skoro bez ikakvog kašnjenja u prenosu. Ipak, ovaj pristup nije uvek najbolji. Prvo, ako čvor A poziva čvor F, F mora da odgovori. U suprotnom, A ne može da šalje informacije. Drugo, pretpostavimo da čvorovi A i F retko razmenjuju informacije (da li ste ikada "iskusili" dugačke periode ćutanja za vreme telefonskog razgovora?). U tom slučaju, konekcija nije dovoljno iskorišćena.

Komutacija poruka je alternativa komutaciji kola. Mreža je koristi za uspostavljanje rute kada se pošalje poruka (jedinica informacija). Na primer, pretpostavimo da čvor A šalje sledeću poruku do čvora F: "Da li želiš da ideš u bioskop sa mnom?". Čvor A u poruku uključuje i adresu, ili lokaciju čvora F i traži prvi sledeći čvor na ruti. Kao što je prikazano na slici 1.15, čvor A bira čvor C. Kao i ranije, izbor čvora zavisi od cene i raspoloživosti konekcije. Cvor A šalje poruku (zajedno sa adresom čvora F) ka čvoru C. Tu se poruka privremeno smešta, dok logika čvora C ne pronade sledeći čvor. Poruka se, zatim, šalje do čvora E, gde se ponovo privremeno smešta. Konačno, logika u čvoru E locira čvor F i šalje poruku na njeno finalno odredište. Pošto se poruka u potpunosti smešta u svakom čvoru, mreže koje koriste ovaj metod nazivaju se i **store-and-forward** (snimi-i-prosledi) mreže.

Po čemu se komutacija poruka razlikuje od komutacije kola?



SLIKA 1.15 *Mreža sa komutacijom poruka*

- Kod komutacije poruka poruka se privremeno smešta u svakom čvoru. Kod komutacije kola čvor se jednostavno ponaša kao preklopni uređaj za rutiranje podataka. Na primer, Vaš telefonski razgovor se ne smešta na posredničkim lokacijama (osim ako neko prisluškuje i snima taj razgovor). Kašnjenje u prenosu koje je neophodno za donošenje odluka kod komutacije poruka čini ovu strategiju povezivanja neprihvatljivom kada je reč o telefonskim mrežama. Kašnjenja u prenosu glasa otežavaju konverzaciju.
- Kod komutacije kola jedna ruta se rezerviše za razmenu svih poruka izmedu dva čvora. Kod komutacije poruka različite poruke mogu da putuju različitim rutama. Pretpostavimo da čvor A želite da pošalje drugu poruku: "Zaboravi prvu poruku; moram da radim do kasno." do čvora F. Pošto je rutiranje često zavisno od vremena, A može da izabere B kao prvi čvor u ruti. U tom slučaju, poruka ide preko čvora D do čvora F. Različite poruke mogu da dele iste konekcije u vremenu, čime se postiže bolja iskorišćenost.
- Komutacija kola zahteva da oba učesnika budu spremna u trenutku kada se podaci pošalju. To nije neophodno kod komutacije poruka. Poruka može da se pošalje i snimi radi kasnijeg preuzimanja.

Treća konekcija povezivanja, *komutacija paketa,* minimizira efekte problema koje stvaraju dugačke poruke. Dugačke poruke mogu da premaše kapacitet bafera u čvoru, ili konekcije izmedu dva susedna čvora mogu biti "zagušene" u dužim periodima. Otkaz konekcije može da znači gubitak cele poruke. Zbog toga su mreže sa komutacijom poruka uglavnom zamenjene mrežama sa komutacijom paketa. Pogledajmo kako funkcionišu mreže sa komutacijom paketa.

Pretpostavimo da čvor A želi da pošalje poruku do čvora F. Ako je poruka dugačka, ona se deli na manje jedinice, koje se nazivaju **paketi.** Velidna paketa zavisi od dizajna.

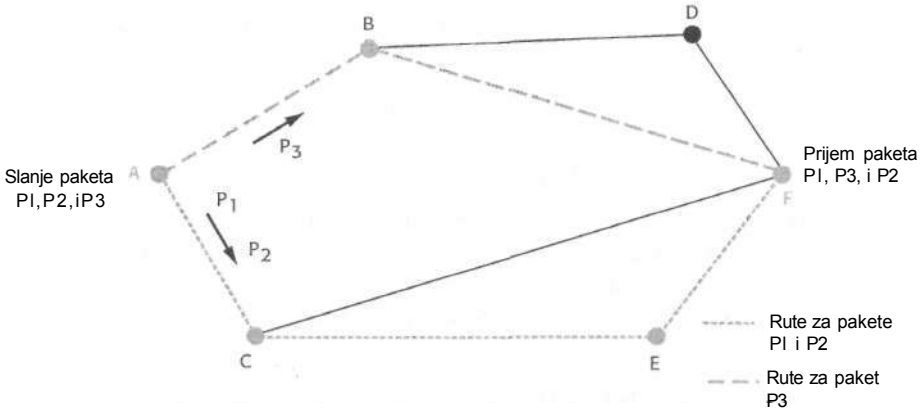
Svaki paket sadrži odredišnu adresu i neke druge naznake koje ukazuju kome je poruka namenjena i poruka se rutira pomoću mrežnih protokola. Kada svi paketi stignu na odredište, ponovo se sastavljaju u originalnu poruku. Kao i kod komutacije poruka, ne održava se fizička konekcija izmedu krajnjih tačaka. Manja veličina paketa predstavlja olakšicu kod neophodnog baferovanja u posredničkim čvorovima mreže.

Dva uobičajena metoda za rutiranje u mrežama sa komutacijom paketa su metod datagrama i metod virtuelnog kola. Kod pristupa datagrama svaki paket se prenosi nezavisno - mrežni protokol rutira svaki paket kao da je reč o zasebnoj poruci. Tako je strategiji za rutiranje omogućeno razmatranje promene stanja u okviru mreže. Zbog zagušenja na nekim rutama može se doneti odluka o promenama ranije utvrdenih mta (u Poglavlju 10 detaljnije ćemo predstaviti strategije rutiranja).

Kod pristupa **virtuelnog kola** mrežni protokol uspostavlja rutu (virtuelno kolo) pre nego što počne slanje paketa. Paketi se isporučuju istom rutom, tako da je osigurano da se primaju u ispravnom redosledu bez grešaka. Proces je sličan komutaciji kola, ali sa jednom značajnom razlikom: ruta nije rezervisana - različita virtuelna kola mogu da dele zajedničku mrežnu konekciju. Logika u svakom čvoru mora da smešta primljene pakete i da planira njihovo dalje slanje.

Metod datagrama ima nedostatke zbog toga što nezavisno rutiranje paketa može da poveća troškove. U nekim slučajevima je efikasnije koristiti virtuelna kola. Sledeći nedostatak datagrama je to što paketi verovatno ne stižu istim redosledom kojim su poslati. To nije prihvatljivo kod real-time audio, ili video zapisa kod kojih se zvuk čuje, ili slika vidi odmah čim se paketi pojave na odredištu. Za takve aplikacije je obično neophodno primati pakete istim redosledom kojim su i poslati.

Na slici 1.16 prikazan je problem. Pretpostavimo da čvor A želi da pošalje poruku do čvora F, koja sadrži tri paketa. Logika u čvoru A odlučuje da rutira pakete Pl i P2 do čvora C. Medutim, kada je proučavana moguća ruta za P3, utvrdeno je da bi ruta preko čvora C mogla da se "zaguši". Zato se P3 šalje do čvora B. Paketi Pl i P2 putuju do čvorova E i F, dok paket P3 ide direktno od B do F. U zavisnosti od saobraćaja na mreži, F može da primi pakete u redosledu P1, P3, P2 i onda mora da ih sastavi u ispravnom redosledu.



SLIKA 1.16 *Mreža sa komutacijom paketa*

Tabela 1.2: Poređenje strategija za povezivanje

Strategija	Prednosti	Nedostaci
Komutacija kola	Brzina. Prikladna je u situacijama kada su kašnjenja u prenosu neprihvatljiva.	Pošto su mrežne konekcije rezervisane, sve druge rute moraju da ih izbegavaju. Oba korisnika moraju da budu prisutna u vreme komunikacije, kao za vreme telefonskog razgovora.
Komutacija poruka	Rute nisu rezervisane i mogu se koristiti odmah nakon prenosa poruke. Primalac ne mora odmah da prihvati poslatu poruku.	U opštem slučaju, porukama je potrebno više vremena da stignu do svojih odredišta. Problemi mogu da se jave i kod dugačkih poruka, jer moraju da se baferuju u posredničkim čvorovima. Poruka putuje preko ranije izabranog čvora na osnovu uslova koji su mogli u meduvremenu biti promenjeni.
Komutacija paketa (datagrami)	Ako dode do "zagušenja", pristup datagrama kod komutacije paketa može da izabere alternativne rute za preostale delove poruke. Na taj način je obezbedeno bolje iskorišćenje mrežnih ruta.	Veći su troškovi, jer se svaki paket rutira zasebno. Odluke o rutiranju moraju da se donesu za svaki paket pojedinačno. Paketi ne moraju da stižu istim redosledom kojim su poslati.
Komutacija paketa (virtuelna kola)	Svi paketi koriste istu rutu, tako da je obezbeđeno pristizanje u istom redosledu kojim su i poslati. Ova strategija je veoma korisna za real-time aplikacije. Osim toga, niži su troškovi rutiranja paketa.	Rute kojima se paketi prenose posle nekog vremena ne moraju da budu optimalne. Potrebno je pregovarati o novim rutama.

Sa druge strane, osetljivost na promenu uslova može da bude prednost. Rutiranje paketa P3 drugom rutom može da uzrokuje primanje paketa u pogrešnom redosledu, ali zbog toga P3 stiže ranogo ranije nego što bi, inače, stigao. Kod mreža sa velikim saobraćajem dobra ruta može da postane loša ako svi čvorovi pokušavaju da prenesu pakete preko nje, kao što i najšire ulice koje vode ka centru velikog grada predstavljaju dobar izbor u 5 sati izjutra, a postaju zakrčene dva sata kasnije.

Strategije rutiranja i protokole za mreže sa komutacijom paketa detaljnije ćemo prikazati u kasnijim poglavljima. Za sada, u okviru tabele 1.2 možete da vidite poredenje do sada predstavljenih strategija povezivanja.

Fizički sloj

Fizički sloj je prvenstveno zadužen za medijum za prenos podataka i načine na koje se signali prenose. Najčešće korišćeni medijumi su kablovi sa upredenim paricama, koaksijalni kabl, optički fiber, sateliti, mikrotalasni tornjevi, infracrveni zraci i radio talasi. Svaka opcija ima specifična električna, elektromagnetna, ili optička svojstva koja su prikladna u određenim situacijama. Ova svojstva stvaraju i ograničenja u pogledu količine informacija koju je moguće preneti u

jedinici vremena i odreduju verovatnoću da će doći do smetnji iz spoljašnjih izvora. Sledeće pitanje je da li treba preneti analogne, ili digitalne signale. Većina ljudi zna da se danas sve više koriste digitalrri prenosi, ali šta to, u stvari, znači? Kompletan opis medijuma za prenos i srodnih tema u vezi analognog i digitalnog prenosa, širine propusnog opsega, odnosa signal-šum, širokopojasnih konekcija, osnovnog opsega, kvaliteta prenosa glasa, pa, čak, i Furijeove analize može da zauzme cele tomove. Neke od ovih tema detaljnije ćemo obraditi u Poglavlju 3.

Sloj veze

Dok se fizički sloj bavi prenosom i prijemom podataka, sloj veze se nalazi na nivou iznad njega - njegov zadatak je da obezbedi ispravno funkcionisanje fizičkog sloja. Na primer, šta se dešava ako dva čvora istovremeno pokušavaju da pošalju podatke preko iste linije (nadmetanje)? Kada čvor zna da li su podaci koje je primio tačni (detekcija grešaka i korigovanje)? Da li električne smetnje koje izazivaju električna oluja, ili promene u naponu mogu da dovedu do promene nekih bitova? Ako smetnje promene odredište paketa, kako čvor zna da nije primio nešto što je trebalo da primi?

Nadmetanje se javlja kada dva, ili više čvorova žele istovremeno da pošalju podatke preko istog medijuma. Postoji nekoliko načina da se to kontroliše. Neke mreže sa topologijom magistrale koriste metod pod nazivom **detekcija kolizije.** Detekcija kolizije ne sprečava čvorove da istovremeno iniciraju prenos preko zajedničkog medijuma. Umesto toga, ovaj metod predstavlja odziv na simultane prenose, ili kolizije. Obično, ako neki uredaj pošalje podatke koji dodu u koliziju sa nekim drugim podacima, kolo za "osluškivanje" detektuje koliziju i uredaj kasnije ponovo pokušava da pošalje iste podatke.

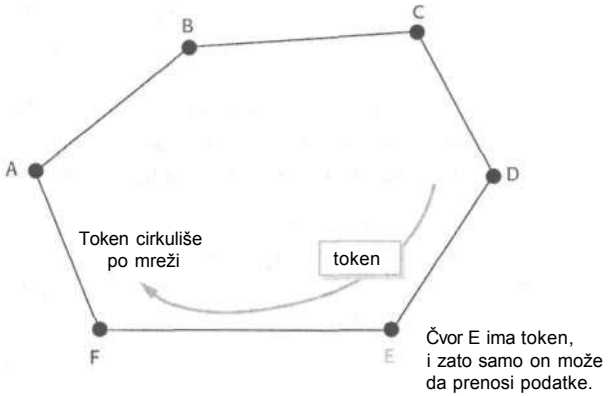
Ponekad uredaji pokušavaju da izbegnu kolizije "osluškivanjem" aktivnosti na magistrali. Ako je magistrala zauzeta, uredaj ne inicira prenos. Ako kola detektuju da nema aktivnosti na magistrali, uredaj inicira prenos. Ako dva uređaja "osluškuju" magistralu i detektuju odsustvo aktivnosti, istovremeno iniciraju prenos podataka i ponovo dolazi do kolizije. Ovaj metod razrešavanja nadmetanja na zajedničkom medijumu nosi naziv *Canier Sense Multiple Access ivith Collision Detection* (CSMA/CD). On ne razrešava problem kolizije, većga samo svodi na minimalnu moguću meru. U odeljku 4.7 data su detaljnija objašnjenja ovog metoda.

Prosledivanje tokena je sledeća šema za rešavanje nadmetanja na zajedničkom medijumu, koja spečava nastanak kolizija. U ovom slučaju jedinstvena sekvenca bitova, nazvana *token,* kruži preko čvorova na mreži. Ako čvor ima podatke koje treba da prenese, mora da čeka sve dok ne primi token, koji postavlja na kraj poruke. Osim toga, menja kontrolne bitove u tokenu da se pokaže da se token trenutno koristi. Poruka se šalje do njćnog odredišta i tada se token nalazi u odredišnom čvoru. U zavisnosti od protokola, prijemna stanica može da uzme token i da ga iskoristi za slanje sopstvenih podataka, ili može da prosledi token do sledećeg čvora.

Mreže sa topologijom prstena često koriste metod prosledivanja tokena, kao što je prikazano na slici 1.17. Token kruži kroz prsten u smeru kretanja kazaljki na časovniku. Cvor E trenutno ima token; zbog toga, samo on može da pošalje poruku. Pošto je token jedinstven, ne može da dode do kolizije. Ipak, ovaj metod ima i svoje nedostatke: token može da se izgubi, ili duplira, ili može da ga zadržava jedan isti čvor. Rešavanje ovakvih problema je detaljnije prikazano u Poglavlju 9.

Prosledivanje tokena nije ograničeno samo na mreže sa topologijom prstena. Može da se koristi i u svim ostalim topologijama ako su čvorovi numerisani, tako da token kruži na osnovu numeričkog redosleda. Ipak, numeracija čvorova se najlakše izvodi u linearnim mrežama, ili kod

mreža sa topologijom prstena. U odeljku 4.7 biće reči o prosledivanju tokena i nekim njegovim varijacijijama, dok je u odeljku 9.6 ovaj metod predstavljen sa stanovišta token ring standarda.

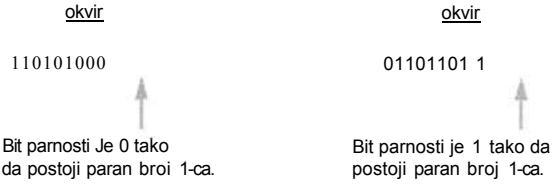


SLIKA 1.17 *Token Ring mreža*

Fizički sloj prenosi nizove bitova preko mreže. Kako primalac zna da li je primio tačne podatke? Loše konekcije, neispravne linije, ili električne smetnje mogu da utiču na prenos. Sloj veze izvršava detekciju grešaka i primenjuje algoritme za korigovanje eventualnih grešaka. Pomoću detekcije grešaka sloj veze na prijemnoj strani utvrđuje da li je došlo do greške; ako jeste, obično zahteva ponovno slanje informacija. Pomoću korekcije grešaka sloj veze ima mogućnost ispravljanja oštećenih bitova.

Najjednostavniji metod detekcije je verovatno bit parnosti, dodatni bit koji se pridružuje svakoj sekvenci bitova, ili okviru. Na primer, parna parnost obezbeduje paran ukupan broj jedinica (uključujući i sam bit pamosti). Zbog toga, ako okvir ima neparan broj bitova sa vrednošću 1, bit parnosti je 1. Ako je ukupan broj paran, bit parnosti je 0 (postoji analogna definicija i za neparnu parnost). Razmotrite okvire na slici 1.18. Prvi okvir ima četiri bita sa vrednošću 1. Zato je bit parnosti O. Drugi okvir ima pet bitova sas vrednošću 1, tako da je bit parnosti I. Bit parnosti se prenosi zajedno sa okvirom i primalac proverava parnost. Ako otkrije da postoji neparan broj bitova sa vrednošću 1, došlo je do greške.

Problem kod korišćenja bitova parnosti je to što neke greške mogu da ostanu nedetektovane. Na primer, ako se u vreme prenosa promene dva bita, broj bitova sa vrednoscu 1 ostaje paran. Zbog toga, parnost može da detektuje samo jednostruke, a ne i dvostruke greške. Za detektovanje višestrukih grešaka postoje sofisticiranije tehnike. O tome je rečeno nešto više u Poglavlju 6.



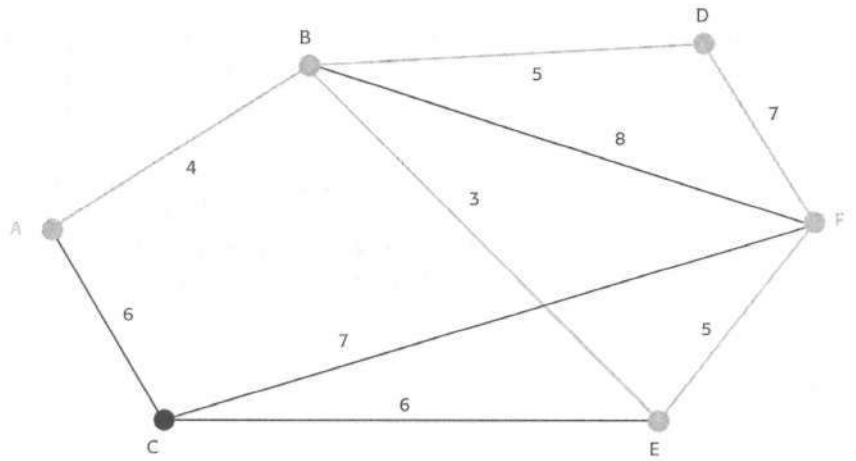
SLIKA 1.18 *Bitovi parnosti*

Sloj mreže

Sloj mreže obezbeduje transportnom sloju mogućnost uspostavljanja komunikacije izmedu krajnjih tačaka. Na osnovu ove mogućnosti transportni sloj može da obavlja svoje zadatke bez potrebe da vodi računa o detaljima razmene informacija između dve stanice. To može da se uporedi sa telefonskim pozivom, kod koga ne morate da vodite računa kako funkcioniše telefonska oprema. Ovaj proces, koji često zahteva komunikaciju izmedu više posredničkih čvorova, može da bude prilično komplikovan. Postoji nekoliko protokola, ali isuviše su komplikovani da bismo ih ovde prikazali; opisaćemo ih u kasnijim poglavljima.

Sloj mreže sadrži algoritme koji su dizajnirani za pronalaženje najbolje rute izmedu dve tačke. Utvrdivanje ruta smo pomenuli u okviru rasprave o tehnikama komutacije. Rekli smo da utvrđivanje mte zavisi od različitih faktora, kao što su cena konekcije i raspoloživost linija, jer se pokušava pronaći najbrža i najjeftinija ruta do odredenog čvora. Na primer, na slici 1.19 prikazana je mreža na kojoj postoji nekoliko mogućih ruta od čvora A do čvora F, Svaka linija koja povezuje dva čvora ima odredenu cenu, koja je naznačena kao broj linije. Polazeći od čvora A ka čvom F, preko čvorova B i D, daje ukupnu cenu 16. Ako se ide preko čvorova B i E, ukupna cena je samo 12. U opštem slučaju, sloj mreže utvrduje koja je ruta najbolja. Hi lačnije, protokoli na sloju mreže koji se izvršavaju u čvorovima kolektivno utvrđuju koja je ruta najbolja. Postoje brojni pristupi rutiranju; njih ćemo predstaviti u Poglavlju 10.

Uspešno rutiranje je često mnogo teže nego što izgleda na prvi pogled. Na predavanjima iz diskretne matematike i struktura podataka obično se obraduju algoritmi za pronalaženje najbolje, ili najjeftinije rule preko grafova. Ipak, tu se uzimaju neke pretpostavke koje često nisu tačne u praksi. Pretpostavlja se da se čvorovi grafa i cene veza ne menjaju. U dinamičkim okmženjima ovakve pretpostavke nisu tačne. U mrežu mogu da se uključuju novi uredaji i to automatski utiče i na postojeće mte i na njihove cene. Algoritmi moraju da budu robustni da bi reagovali na promenu uslova.



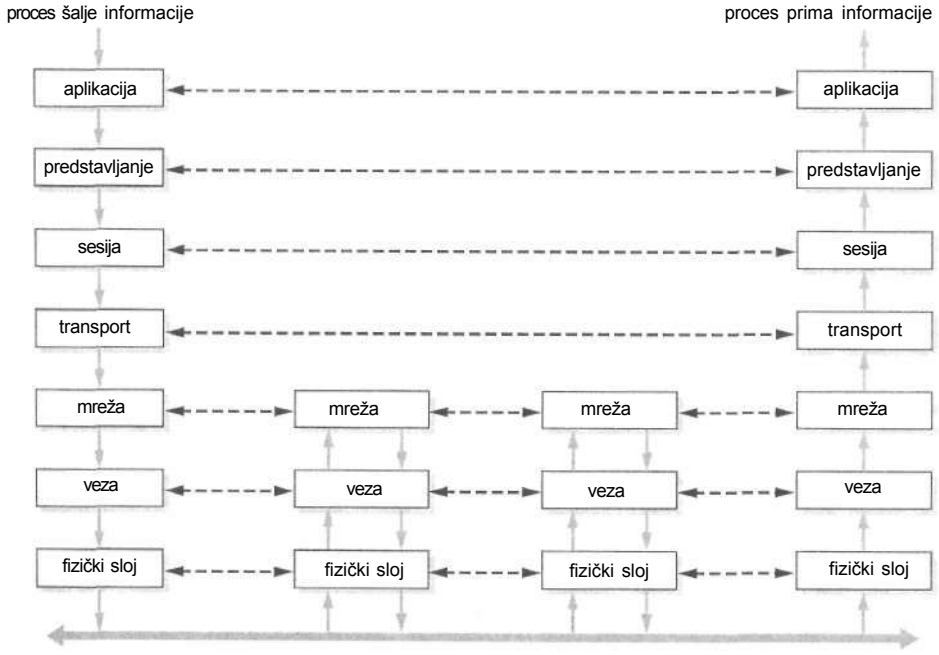
SLIKA 1.19 *Cene ruta*

Čak i kada algoritmi odgovore na promenu uslova, mogu se javiti drugi problemi. Dobra ruta može da privuče veliku količinu saobraćaja i da tako preoptereti kompjutere koji su priključeni na nju. Rezultujuće zagušenje često dovodi do eliminisanja odredene količine saobraćaja. Ovo deluje kao isuviše strogo rešenje, ali kada čvor mora da prenese preveliku količinu informacija, ne postoji mnogo altemativa. U tom slučaju, protokol na sloju mreže mora da bude sposoban da obavesti pošiljaoca da je poruka izgubljena.

Drugi problemi mogu da se jave kada se informacije koje se prenose preko jedne aite preusmere zbog promene uslova, koja je dovela do defmisanja nove najbolje rute. U tom slučaju, informacije mogu da putuju tom mtom, samo da bi se kasnije ponovo preusmerile. U ekstremnim slučajevima, informacije mogu kontinuelno da se preusmeravaju, tako da odredene informacije beskonačno "odskaču" od jednog čvora do drugog. To je elektronski ekvivalent za prosjaka koji traži mesto koje bi nazvao dom.

Transportni sloj

Transportni sloj predstavlja prelazni sloj. Tri sloja ispod transportnog zadužena su za mrežne komunikacije (slika 1.20). Svaki ćvor izmedu predajnog i prijemnog čvora izvršava svoje protokole da bi bio obezbeđen ispravan i efikasan prenos informacija.



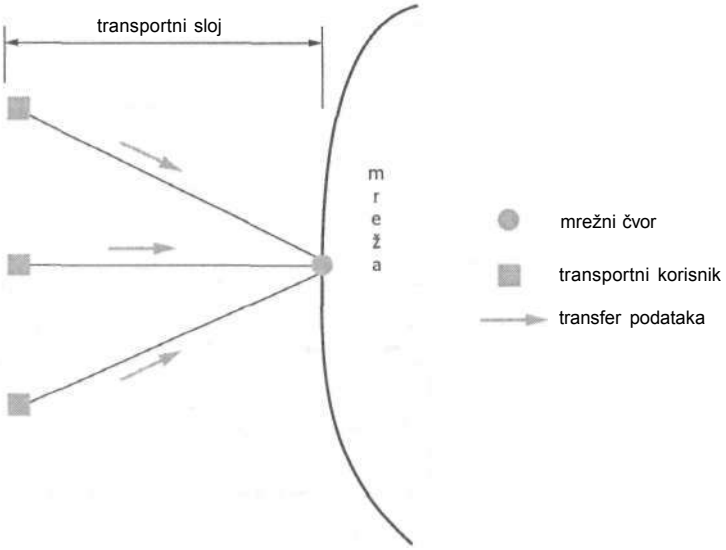
SLIKA 1.20 *Povezivanje krajnjih tačaka pomoću posredničkih čvorova*

Transportni sloj i tri sloja iznad njega obezbeđuju korisničke servise. Njihovo izvršenje je fokusirano prvenstveno na predajne i prijemne čvorove da bi bili osigurani pristizanje informacija na njihovo odredište i potvrda o prijemu informacija koja se šalje do predajnog čvora.

Jedna od funkcija transportnog sloja je obezbedivanje pouzdane i efikasne mrežne konekcije. Zahvaljujući njemu, gornja tri sloja mogu da izvršavaju svoje zadatke nezavisno od specifične mrežne arhitekture. Istovremeno, transportni sloj se oslanja na donja tri sloja prilikom kontrole konkretnih mrežnih operacija. On obezbeduje dolazak informacija od izvora do željenog odredišta. Glavna odgovornost ovog sloja je obezbedivanje mrežne konekcije, ili transportne konekcije za slojeve sesije. Najvažniji zadatak je obezbedivanje pouzdanih i efikasnih komunikacija. U praksi, mreže su ponekad nepouzdane - ne postoji garancija da neće doći do otkaza konekdje. Sta se dešava ako se neki paket podataka izgubi? Sta ako paket značajno kasni? Kako ovakvi problemi utiču na korisnike? Specifičnosti uobičajenog transportnog protokola (Transmission Control Protocol - TCP) predstavićemo u Poglavlju 11.

Među funkcije transportnog sloja ubrajaju se multipleksiranje, baferovanje i upravljanje konekcijom. Pomoću *multipleksiranja* (slika 1.21) nekoliko transportnih korisnika deli isti čvor. Na primer, korisnik može da uspostavi više konekcija na mreži preko jedne radne stanice. Jedna konekcija služi za logovanje na udaljenu mašinu, druga za preuzimanje fajla, a treća za pristup Web sajtu. Svaka intereaguje sa transportnim slojem radi slanja i prijema podataka, a transportni sloj osigurava da se svi dolazeći podaci usmere do odgovorajućeg korisnika.

Transportni sloj kontroliše i baferovanje u mrežnim čvorovima. Interesantno je da se baferovanje može desiti ili u odredišnom, ili izvornom čvoru.



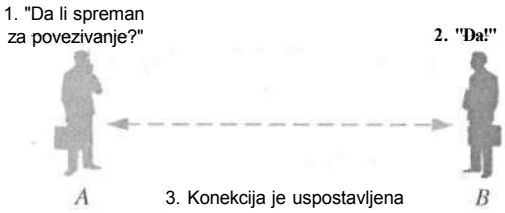
SLIKA 1.21 *Multipleksiranje*

Kada transportni sloj na strani pošiljaoca primi podatke od strane sloja sesije namenjene za prenos, on ih deli u jedinice koje se nazivaju jedinice transportnog protokola **(transport protocol data units** - TPDU). Transportni protokol ih šalje do transportnog sloja na prijemnoj strani (preko nižih slojeva), gde se eventualno rutiraju ka sloju sesije na prijemnoj strani.

Transportni protokol obično zahteva potvrdu za prijem svakogTPDU-a. Ova potvrda je posebno korisna na mrežama sa komutacijom paketa, kod kojih paketi mogu da kasne, ili, čak, i da se izgube zbog nekih otkaza na mreži. Pretpostavimo da transportni korisnik A želi da pošalje nekoliko TPDU-a do korisnika B. U nekim slučajevima TPDU-i se baferuju u izvoru i formira se red čekanja za slanje sa nižih slojeva. Transportni sioj čeka na potvrdu za svaki TPDU, ali ga ne uklanja iz bafera. Ako protokol zahteva potvrdu za svaki TPDU, transportni sloj na strani pošiljaoca čeka na tu potvrdu. Ukoliko potvrda ne stigne u odredenom vremenskom periodu, pošiljalac ponovo šalje TPDU, a to je moguće samo ako se on još uvek nalazi u baferu.

Kada transportni sloj na prijemnoj strani primi TPDU, on ga zadržava sve dok sloj sesije ne bude spreman da ga prihvati. Imajte na umu da ne postoje nikakve garancije da će sloj sesije odmah prihvatiti TPDU. Ako primalac zna da pošiljalac baferuje sve TPDU-e, može da izabere korišćenje samo jednog bafera radi uštede prostora. Nedostatak ovog izbora je što primalac nema dovoljno prostora za prihvatanje narednih TPDU-a sve dok se čuva TPDU koji je ranije isporučen. U tom slučaju, primalac ignoriše TPDU-e i odbija da pošalje potvrdu za njihov prijem. Pošiljalac ne prima nikakvu potvrdu i eventualno ponovo inicira prenos TPDU-a.

Upravljenje konekcijom (connection managament) je protokol na osnovu koga transportni sloj uspostavlja i oslobada konekcije između dva čvora. Na prvi pogled, uspostavljanje i oslobađanje konekcija deluju jednostavno, ali su, u stvari, prilično "nezgodne". Na primer, pretpostavimo da transportni sloj pokušava da uspostavi konekciju između korisnika A i B. Konekcija može da bude jednostavna kada (I) korisnik A zahteva uspostavljanje konekcije sa korisnikom B i kada (2) korisnik B ukazuje na spremnost za prihvatanje konekcije, nakon čega se (3) konekcija uspostavlja (slika 1.22). To je protokol dvosmernog usaglašavanja **(two-way handshake protokol)** za uspostavljanje konekcije. Problem je što ovo ne funkcioniše uvek zbog potencijalnih kašnjenja, ili zahteva korisnika A, ili odgovora korisnika B. Medutim, ove probleme i njihova rešenja moći ćemo da opišemo na najbolji način tek u odeljku 11.4, nakon što predstavimo i neke druge protokole.

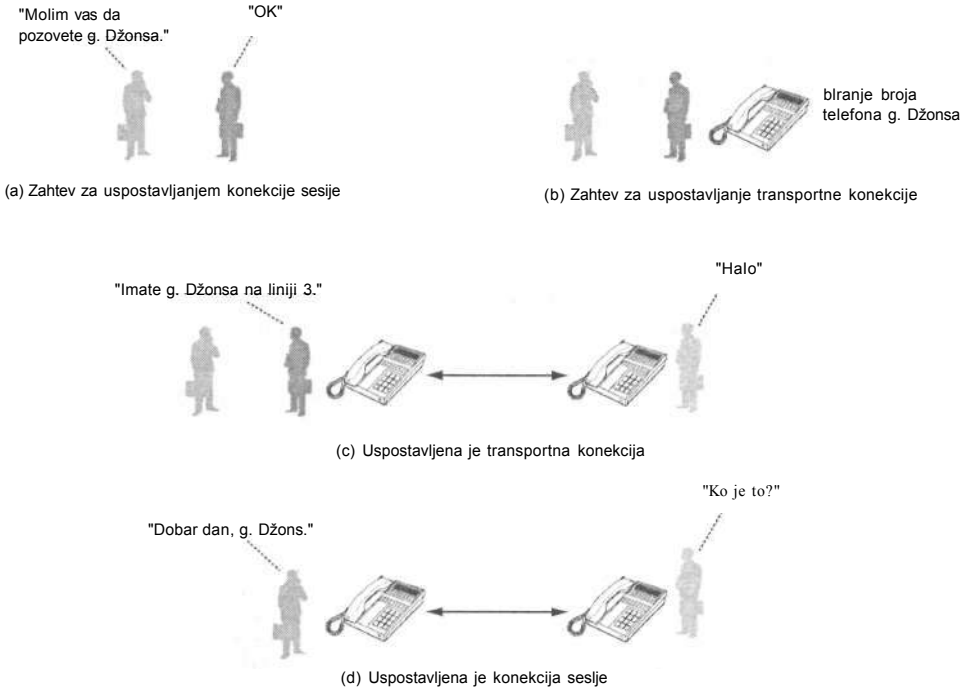


SLIKA 1.22 *Dvosmerno usaglašavanje*

Sloj sesije

Sledeća tri sloja se bave prvenslveno korisničkim servisima i funkcijama (prethodni slojevi su se fokusirali na komunikacije). Sloj sesije sadrži protokole koji su neophodni za uspostavljanje i održavanje konekcije, ili sesije izmedu dva krajnja korisnika. Razlika izmedu transportnog sloja i sloja sesije često nije jasna na prvi pogled. Transportni sloj obezbeduje konekciju izmedu dva čvora koja je neophodna sloju sesije, a upravo smo rekli da sloj sesije obezbeđuje konekciju izmedu korisnika. U čemu je razlika? Na slici 1.23 prikazana je analogija, na osnovu koje bi trebalo da razjasnite eventualne zabune.

Na slici, direktor traži od sekretarice da pozove klijenta. Direktor je analogan sloju sesije, a sekretarica transportnom sloju. Znači, zahtev u primeru (a) odgovara zahtevu za uspostavljanje sesije. Direktor zahteva da se nspostavi konekcija, ali ne zalazi u tehničke detalje, kao što je traženje telefonskog broja, ili njegovo biranje. U primeru (b) sekretarica poziva klijenta i tako inicira procedure za uspostavljanje transportne konekcije.



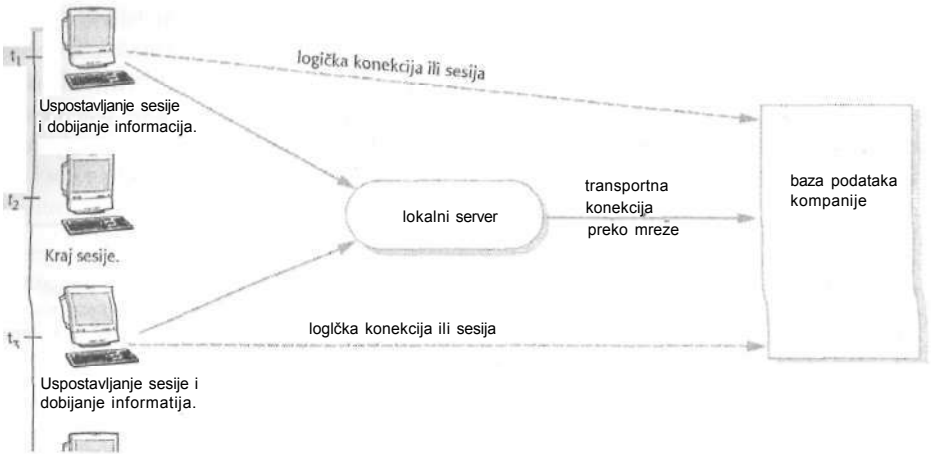
SLIKA 1.23 *Zahtev i uspostavljanje konekcije sesije*

Proces biranja i iniciranja konekcije je nezavisan od načina na koji telefonska kompanija vrši komutaciju kola koja, u stvari, usmerava poziv. Transportni sloj ne vodi računa o tim detaljima. Kada se poziv kompletira u primeru (c), uspostavljena je transportna konekcija. Medutim, sesija nije uspostavljena sve dok direktor konačno ne uzme telefon kao u primeru (d).

Na slici 1.24 dat je još jedan primer koji se tiče kompjuterskih komunikacija. Velike kompanije koje imaju predstavništva u celom svetu uglavnom sve najvažnije podatke čuvaju u bazi podataka na centralnoj lokaciji. Svako predstavništvo (sistemi za rezervadju avionskih karata, velike brokerske kompanije, banke i tako dalje) ima sopstveni server koji komunicira sa bazom podataka kompanije preko mreže. Zaposleni u lokalnim predstavništvima često preko servera pristupaju bazi podataka u kratkim vremenskim intervalima. To su sesije koje obezbeduje sloj sesije. Zbog čestog pristupa, transportni sloj održava jednu transportnu konekciju izmedu servera i baze podataka. Svaka sesija koristi istu transportnu konekciju i sistem funkcioniše mnogo efikasnije nego kada bi se svaki put ugovarala nova konekcija prilikom novog zahteva za pristup bazi podataka.

Ne interpetirajte sliku 1.24 kao oblik multipleksiranja. Transportna konekcija opslužuje samo jednu sesiju u jednom trenutku. Druga sesija može da koristi transportnu konekciju tek kada se prva sesija završi.

Ako se transportna konekcija prekine zbog otkaza mreže, sloj sesije može da zahteva dmgu transportnu konekciju bez prekida sesije. Ovakav oporavak je analogan situaciji kada se prekine veza direktora sa slike 1.23 sa klijentom i dok on čeka da sekretarica ponovo pozove klijenta.



SLIKA 1.24 *Više sesija koristi jednu transportnu konekciju*

Sloj sesije obezbeđuje i druge mogućnosti, ali ovde ih nećemo detaljnije opisivati. Na primer, sloj sesije može obema stranama da dopusti istovremeno slanje informacija (full duplex), ili da ih primora da vrše naizmenično slanje (half duplex). Osim toga, može da organizuje informacije po jedinicama; ako dode do narušavanja konekcije, korisnik ne mora da ponavlja slanje poruke od početka. Ovo je korisno ako dođe do prekida za vreme slanja velikog fajla. Kada se konekcija ponovo uspostavi, transfer može da se nastavi od tačke u kojoj je prekinut. Osim toga, sloj sesije može da defmiše aktivnosti koje se moraju izvesti po principu "sve, ili ništa." Primer je ažuriranje udaljene baze podataka koje uključuje promenu nekoliko zapisa. U slučaju prekida veze za vreme ažuriranja moguče je da se neki zapisi promene, a neki da ostanu nepromenjeni, zbog čega bi došlo do nekonzistentnosti podataka. Sloj sesije može da obezbedi da se izvedu sve promene, ili, u slučaju problema, da se ne izvede ni jedna.

Sloj predstavljanja

Umrežavanje kompjutera može da bude mnogo jednostavnije ako svi "govore" istim jezikom. Ovde ne govorimo o jezicima kao što su C, Ada, Java, C++, ili COBOL. Mislimo na način na koji kompjuter predstavlja entitet koji nazivamo informacija.

Moramo da istaknemo razliku izmedu informadje i podataka, jer je reč o veoma bitnoj razlici. Kada govorimo o *podadma,* mislimo na gomile cifara, heksadecimalnih brojeva, ili stranica sa slovima, ili specijalnim simbolima. Ukratko rečeno, kompjuter ne smešta informacije, već podatke. Informacija je značenje pridruženo zapamćenim podacima. U suštini, podaci predstavljaju uređene nizove bitova i bajtova i drugih nespomenutih "stvari". *Informacija* je ljudska interpretacija tih podataka. Problem nastaje zbog toga što različiti kompjuteri imaju različite načine za predstavljanje istih informacija. Zbog toga, nije dovoljno samo definisati efikasnu razmenu podataka. Moramo da definišemo i efikasnu razmenu informacija. To je zadatak sloja predstavljanja.

Na primer, razmotrimo mrežu koja prenosi podatke između dva kompjutera (slika 1.25). Jedan smešta informacije pomoću ASCII koda, a drugi pomoću EBCDIC koda, koristeći različite načine za smeštanje podataka (u Poglavlju 2 detaljnije ćemo predstaviti ASCII i EBCDIC). Kada ASCCII kompjuter kaže: "HELLO", mreža prenosi ASCII poruku. EBCDIC kompjuter prima i smešta podatke. Nažalost, nakon interpetiranja primljnih podataka dobija se "«!" jer se na prijemnoj strani vrši drugačija interpretacija bitova.

SLIKA 1.25 *Razmena podataka između dva kompjutera*

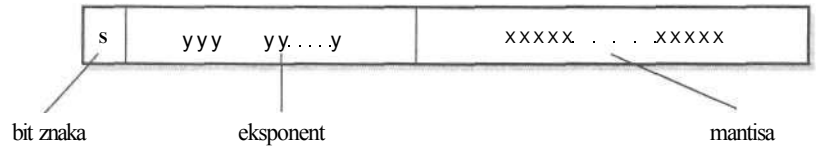
SLIKA 1.26 *Razmena informacija izmedu dva kompjutera*

Mi želimo, u stvari, da izvršimo razmenu informacija onako kako je prikazano na slici 1.26. Preneta je ASCII verzija reči "HELLO". Pošto prijemnik koristi EBCDIC, podaci se moraju konvertovati. U ovom slučaju se prenose heksadecimalni karakteri 48 45 4C 4C 4F, ali se primaju karakteri C8 C5 D3 D3 D6. Ova dva kompjutera nisu razmenila podatke; umesto toga, što je još značajnije, razmenila su informacije u formi reči "HELLO".

Medutim, problem nije samo konverzija koda. Mogući su problemi i prilikom prenosa brojeva. Na primer, kompjuteri mogu da smeštaju celobrojne vrednosti pomodi formata sa jednim, ili dva komplementa. Razlika je minorna, ali mora se uzeti u obzir. Osim toga, razlikuje se broj bitova koji se koristi za predstavljanje celobrojnih vrednosti. Obično se koriste formati sa 16, 32, 64, ili 80 bitova. Bitovi moraju da se dodaju, ili da se uklone da bi se omogudle različite veličine. Ponekad je translacija nemoguća. Na primer, maksimalna celobrojna vrednost u 16-bitnom formatu je 32.767. Šta se dešava ako kompjuter koristi 32-bitne celobrojne vrednosti i pokušava da prenese vrednost 50.000 na kompjuter na kome je moguće koristiti samo 16-bitne celobrojne vrednosti?

Osim toga, probleme stvaraju i brojevi u pokretnom zarezu. Na slici 1.27 prikazan je uobičajeni forraat, mada su moguće razne varijacije. Razlikuje se broj bitova koji se koristi za eksponent i mantisu. Broj bitova može da se razlikuje i unutar same mašine, jer se koriste ili jednostruka, ili dvostruka preciznost. Isto tako, eksponent može da se intepretira na različite načine. Ponekad je to stepen broja 2; u nekim drugim situacijama reč je o stepenu broja 16. Ponekad se mantisa ne pamti kao kontinuelni niz bitova. Kada se uzmu u obzir sve moguće razlike, pravo je čudo da kompjuteri uopšte mogu da razmenjuju informacije.

Problem se proširuje kada se u celu "priču" uključe sofisticiranije strukture podataka, kao što su nizovi, zapisi i lančane liste. Sloj predstavljanja mora da uzme u obzir način na koji se polja zapisa smeštaju.



SLIKA 1.27 *Generički format brojeva u pokretnom zarezu*

Na primer, da li svako polazi od granice reči, ili granice bajta? Gde se čuvaju linkovi na polja? Koliko bajtova zauzimaju? Da li se višedimenzionalni niz smešta po redovima, ili po kolonama?

Sloj predstavljanja mora da zna kakav sistem opslužuje. Osim toga, mora da poznaje format podataka koje prima iz drugih izvora i da obezbedi ispravan transfer informacija kroz mrežu.

Sledeća funkcija sloja predstavljanja je **kompresija podataka,*** pomoću koje se redukuje broj bitova bez gubljenja značenja informacija. Ako je prenos skup, kompresija može značajno da snizi cenu prenosa i omogući prenos veće količine informacija u jedinici vremena. Na primer, pretpostavimo da se veliki fajl u potpunosti sastoji od niza velikih slova - to može da bude lista ključnih reči, ili prezimena zaposlenih. Koliko bitova podataka morate da prenesete? Ako su karakteri kodirani ASCII kodom, broj je 8n, gde Je n broj karaktera. Ako sloj predstavljanja redefiniše kod tako što se O pridmži A, 1 za B i tako redom, do 25 za Z, svaki karakter u alfabetu može da se predstavi pomoću pet bitova (jer je za 25 karaktera potreban manji broj bitova). Tako se, u suštini, šalje oko 38 odsto manje bitova. Postoje mnogi metodi za kompresovanje podataka; detaljnije ćemo ih predstaviti u Poglavlju 5.

Bezbednost je sledeći razlog za promenu bitova (za njihovo šifrovanje) pre slanja. Kada neautorizovana osoba presretne poruku, neće imati nikakve koristi od nje ako je šifrovana. U Poglavlju 7 detaljnije je opisano šifrovanje podataka.

Sloj aplikacije

Sloj aplikacije, najviši sloj u OSI modelu, komunicira sa korisnikom i programskim aplikacijama. Naziva se tako zato što sadrži mrežne aplikacije, koje se razlikuju od aplikacija za obračuna plata, ili knjigovodstvenih aplikacija, paketa za grafički dizajn, jezičkih prevodilaca, ili programa za rad sa bazama podataka. LI tipične mrežne aplikacije ubrajaju se Web aplikacije, elektronska pošta, transfer fajlova, protokoli virtuelnog terminala (sve ćemo predstaviti u Poglavlju 11) i distribuirani sistemi.

Većina fakulteta i velikih organizacija povezana je na Intemet, koji omogučava razmenu privatnih i poslovnih poruka preko elektronske pošte. Primera radi, brojni komentari koje su autor i urednik razmenili u vreme nastajanja ove knjige preneti su pomocu elektronske pošte. Niži slojevi modela obezbeđuju sredstva za izražavanje poruke i njen prenos do odredišta. Protokol za email na sloju aplikacije definiše arhitekturu sistema elektronske pošte. Pošta se smešta u poštansko sanduče - mailbox (u stvari, fajl), u okviru koga korisnici mogu da organizuju i čitaju svoje poruke i obezbeduju odgovore na njih.

Protokol za transfer fajlova korisnidma takođe omogućava razmenu informacija, mada na drugačiji način. Obično korisniku omogućava povezivanje na udaljeni sistem, ispitivanje sadržaja nekih direktorijuma i fajlova na udaljenom sistemu, a, zatim, njihovo kopiranje na svoj lokalni sistem.

•Najefikasnija kompresija podataka može da se izvede na ovom nivou, zato što je na njemu obezbeden najveci siepen razumevanja podataka i njihove eventualne upotrebe. Ipak, kompresija se često izvodi i na nižim nivoima, umesto na ovom nivou.

Za korisnika to je sasvim jednostavno, mada uz nekoliko problema. Jedan je struktura fajla. Neki fajlovi sadrže jednostavne sekvence bitova; u drugim fajlovima se nalaze linearne sekvence zapisa. Hašovani fajlovi dopuštaju nasumični pristup zapisima preko vrednosti ključeva. Hijerarhijski fajlovi* mogu da organizuju sve pojave polja ključa u strukturi stabla. Ove razlike stvaraju probleme prilikom transfera fajlova.

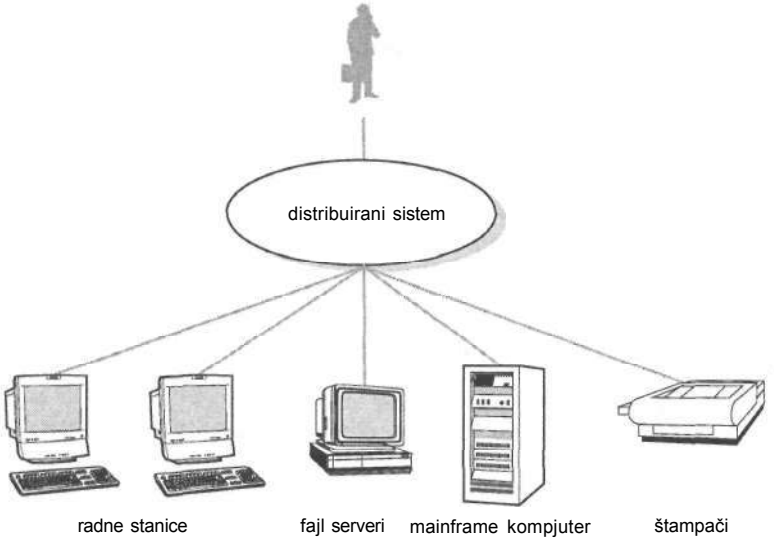
World Wide Web omogućava korisnicima da vide dokumente smeštene na kompjuterima koji se nalaze na različitim lokacijama širom sveta. Osim toga, korisniku je omogućeno da sledi link koji je smešten u jednom dokumentu, a koji ukazuje na neki drugi dokument, ili sajt. Prated linkove, korisnik može da prelazi sa jedne lokacije na drugu jednostavnim klikovima miša. U žargonu se ova aktivnost naziva pretraživanje Weba, ili surfovanje Webom. Web i neke njemu orijentisane aplikacije detaljnije ćemo predstaviti u Poglavlju 12.

Protokol virtuelnog **terminala** omogućava korisniku jedne radne stanice da se uloguje na udaljeni kompjuter preko mreže. Kada se poveže, korisnik može da radi na isti način kao kada radi na svom lokalnom kompjuteru. Mogućnost pristupa različitim kompjuterima sa različitih radnih stanica dovodi do raznih problema, zbog razlika u softveru i opremi.]edan problem su programi koji se često pišu za specifične hardverske komponente (na primer, drajveri za određene grafičke kartice). Primer su tekstualni editori koji koriste ceo ekran. Editor prikazuje tekst na ekranu i omogućava korisniku pomeranje kursora i izvodenje promena u tekstu. U zavisnosti od radne stanice na kojoj radite, postoje razlike u broju prikazanih redova i kolona. Osim toga, komande za pomeranje kursora, ili brisanje i umetanje teksta mogu da zahtevaju različite kontrolne sekvence koje zavise od kompjutera na koji ste povezani. Ostali problemi mogu da budu rezultat različitih formata ekrana koji se koriste za različite softverske pakete (pod ovim podrazumevano organizacioni razmeštaj ekrana, koji je prethodio današnjim grafičkim korisničkim interfejsima). Cesto se pomoću dobrog organizacionog razmeštaja sprečava "zatrpavanje" prikaza korisničkim opcijama. Razmaci, tabulatori i selektovanje pomažu korisniku da radi sa softverom. Ponovo naglašavamo da ove karakteristike zavise od lokacije sa koje pokrećete softver.

Distribuirani sistemi su drugi oblik sve prisutnije aplikacije kompjuterskih mreža. **Distribuirani sistem** omogućava pokretanje istog softvera sa više uređaja i pristup zajedničkim resursima. Medu resurse se ubrajaju radne stanice, fajl serveri, mainframe kompjuteri i štampači (slika 1.28). Kod pravog distribuiranog sistema korisnik se loguje na sistem i ne zna ništa o njegovoj osnovnoj strukturi. Korisnik zahteva određeni resurs i dobija ga bez potrebe da vodi računa odakle resurs potiče, ili koji je tip radne stanice koju koristi. Distribuirani sistem skriva te detalje.

Distribuirani sistemi predstavljaju veliki izazov. Pretpostavimo da korisnik zahteva fajl, navodeći njegov naziv. Problem je što različiti sistemi koriste različita pravila za imenovanje fajlova. Kako da ta pravila budu transparentna za korisnika, a da se i dalje pridržavaju sistema koji kompjuter zahteva? Često neki fajlovi postoje na više različitih kompjutera. Kako distribuirani sistem zna koji od tih fajlova korisnik zahteva? Pošto se koriste različiti kompjuteri, kako distribuirani sistem rešava problern dupliranih naziva fajlova? Distribuirani sistemi uvode brojne probleme.

* Morate da razlikujete strukturu fajla od njegove implementacije. Na primer, fajl može da bude hijerarhijski, ali postoje razne implementacije hijerarhijske strukture.



SLIKA 1.28 *Distribuirani sistem*

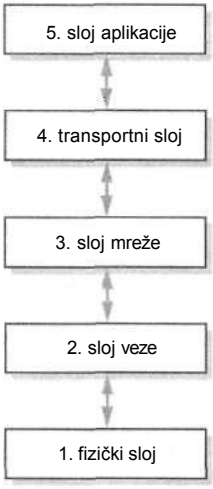
Internet slojevi

Kao što smo istakli na početku ovog odeljka, OSI model nije doživeo komercijalni uspeh. Ogromari napredak emaila i Web aplikacija tokom 90-ih godina prošlog veka učinio je Internet i njegove protokole dominantnim mrežnim modelom.

Po čemu se Internet razlikuje od OSI modela? Glavna razlika je to što se Intemet zasniva na petoslojnom sistemu (slika 1.29). Te slojeve ćemo detaljnije obraditi kroz ovu knjigu, ali sada ističemo da prva četiri mogu da se grubo uporede sa prva četiri sloja OSI modela. Medutim, pošto se Internet razvio tako da može da funkcioniše na ogromnom broju različitih mreža, ne definiše način na koji funkcionišu slojevi 1 i 2. Sloj 3 defmiše formate paketa, procedure rutiranja, tipove servisa i, u zavisnosti od verzije, neke bezbednosne mere. Sloj 4 odgovara ili TCP-u (protokol ne može da funkdoniše bez uspostavljanja veze izmedu krajnjih tačaka za komunikaciju izmedu krajnjih tačaka), ili UDP-u (User Datagram Protocol - protokol za komunikaciju između krajnjih tačaka koji ne zavisi od veze). U Poglavlju 11 predstavićemo oba protokola. Peti sloj, sloj aplikacije, odgovora protokolima kao što su HTTP (za navigaciju Webom), FTP (za transfer fajlova), email i Telnet (protokoli virtuelnog terminala). Funkcionalnosti slojeva 5 i 6 OSI modela nisu podržane na Internetu, ili su uključene u različite Internet aplikacije.

Zaključak

U ovom odeljku smo opisali svih sedam slojeva OSI modela. Svaki definiše komunikacione protokole za kompjuterske mreže i izoluje sloj od detalja slojeva koji se nalaze ispod njega. Zajedno, ovi slojevi izoluju korisnika od detalja samog prenosa informacija u toku komunikacije.



SLIKA 1.29 *Internet slojevi*

Da je OSI model u potpunosti implementiran ("doživeo" je komercijalni neuspeh) omogućio bi komunikaciju izmedu nekompatibilnih uređaja.

Najniža tri sloja se bave prvenstveno mrežnim komunikacijama. Fizički sloj šalje i prima nizove bitova, bez obzira na njihovo značenje. On ne zna značenje podataka, niti da li su oni tačni. Postoje i tri različite strategije povezivanja. Komutacija kola kreira i održava rezervisane linije izmedu čvorova. Komutacija poruka rutira poaike kroz mrežu. Ne postoje rezervisane linije za povezivanje čvorova. Komutacija paketa deli poruku na pakete radi nezavisnog rutiranja.

Sloj veze obezbeduje detekciju grešaka za fizički sloj. Tehnike za detekciju grešaka uključuju bitove parnosti i ostale kodove za detekciju i korigovanje grešaka. Neke od ovih tehnika mogu da detektuju samo jednostruke greške; druge mogu da detektuju kada šum narušava više bitova. Sloj veze sadrži i strategije za rešavanje nadmetanja. Detekcija kolizije omogućava simultani prenos iz više uređaja, ali sa detekcijom nastalih kolizija. Svaki uredaj nakon detektovane kolizije čeka da protekne nasumice izabrani period pre nego što ponovo pokuša da prenese podatke. Prilikom prosleđivanja, token kruži između čvorova na mreži. Čvor može da prenosi podatke samo ako ima token.

Najviši od donja tri sloja je sloj mreže. On sadrži strategije za rutiranje. Algoritmi mogu da utvrde najjeftiniju rutu izmedu dva čvora i svaki čvor zna koji je njegov sledbenik u okviru rute. Medutim, pošto najjeftinija ruta na mreži zavisi od promena uslova u okviru mreže, mogu se koristiti adaptivne strategije rutiranja radi detektovanja promena na mreži; u skladu sa tim promenama, menjaju se ranije utvrđene rute.

Gornja četiri sloja služe korisniku. Najniži je transportni sloj. On obezbeduje baferovanje, multipleksiranje i upravljanje konekcijom. Upravljanje konekcijom obezbeduje da kašnjenje poruka ne ugrožava zahteve za uspostavljanje, ili oslobadanje konekcija.

Sloj sesije kontroliše dijalog između korisnika. Kod half-duplex komunikacija sloj sesije prati ko može da govori, a ko mora da sluša. Osim toga, omogućava nastavak transfera od tačke u kojoj je došlo do prekida i može da upravlja aktivnostima koje moraju da se izvode po principu "sve, ili ništa."

Sloj predstavljanja uzima u obzir razlike u načinu predstavljanja podataka. Omogućava razmenu informacija izmedu dva sistema, mada oni možda ne koriste isti način za smeštanje podataka. Osim toga, kompresuje podatke radi smanjenja broja prenetih bitova. Sloj predstavljanja rnože da implementira i šifrovanje i dešifrovanje.

Poslednji i najviši sloj je s]oj aplikacije, koji sadrži korisničke servise, kao što su elektronska pošta, softver za transfer fajlova i virtuelni terminali. On komunicira direktno sa korisnikom, ili programskom aplikacijom.

Internet je sastavljen od pet slojeva: fizičkog sloja, veze, mreže, transportnog sloja i sloja aplikacije. Elementi 5 i 6 sloja OSI modela sadržani su u Internet aplikacijama, ili u transportnim protokolima.

1.5 Budućnost savremenih komunikacionih tehnologija

Izdvojili smo dosta vremena za opisivanje istorije komunikacija i tekućih tehnologija i modela. Sledeće logično pitanje je kuda dalje. Drugim rečima, šta možemo da očekujemo u budućnosti? Postoje neka predvidanja koja su egzotična i fantastična, mada verovatno nisu ništa više fantastična nego što su ljudima pre nekoliko decenija izgledale tehnologije koje su nama danas na raspolaganju: koncept satelita koji kruži oko zemlje i šalje televizijske signale zvučao je kao naučna fantastika, mogućnost korišćenja svetlosti za prenos slike i zvuka delovala je neverovatno, a ideja o "razgovoru" izmedu kompjutera bila je apsurdna i možda pomalo i zastrašujuća za ljude koji su gledali isuviše filmova u kojima kompjuteri preuzimaju kontrolu nad svetom.

Činjenica je da mi većsada znamo za mnoge "stvari" koje će se sigurno koristiti i u budućnosti. Postoji veliki broj proizvoda koji su već sada tehnološki izvodljivi, ali još uvek nisu isplativi u velikim razmerama. Vratimo se na naše pitanje šta možemo da očekujemo. Evo nekih mogućnosti.

- Ugrađivanje Internet tehnologije u uredaje koji se svakodnevno koriste Mnoge dtaoce će možda iznenaditi činjenica da se 98 odsto procesora ne nalazi u tradicionalnim PC-jima [EsOO], već uređajima koje svakodnevno koristimo (vozilima, časovnicima, kućnim aparatima, fabričkim mašinama, itd). Nemojte da se iznenadite ako u doglednoj budućnosti dode do velikog prodora bežičnih tehnologija u uredaje koji se ne koriste u računarstvu. Zamislite kućni termometar koji može da pristupi telefonu i pozove vatrogasce ako se detektuje visoka temperatura. Ako se uređaj nalazi u stanu, umesto u kući, može da obavesti druge stanare u zgradi i da preduzme i druge aktivnosti osim alarma, kao što je, recimo, uključivanje svih svetala (potencijalni spas za osobe oštećenog sluha). U veoma velikim stambenim kompleksima može da postoji protokol na osnovu koga jedan stan poziva drugi stan, koji, pak, poziva ostale i tako se pravovremeno obaveštavaju svi stanari. Iako takve mogućnosti obaveštavanja već postoje u vidu požarnih alarma, mogu se poboljšati na ovakav način. Medusobno povezivanje uredaja koji se nalaze u stanovima u zgradi može da obezbedi sigurniji i

ažurniji sistem upozorenja, koji će odmah obavestiti ljude pre nego što se problemi "otrgnu" kontroli.

Drugi potencijalni primer mogu da budu uređaji koji će se ugradivati u automobile, a koji mogu da registruju poziciju drugih vozila na autoputu i da sprečavaju akcije koje dovode do saobraćajnih nesreća. Senzori ugrađeni u zemlju mogu da detektuju podrhtavanja u trusnim područjima i da komuniciraju sa drugim uredajima u susednim zgradama koji aktiviraju zvučne alarme. Postoje i neka razmišljanja o ugradnji atmosferskih senzora koji će upozoriti na dolazak tomada u odredena područija, Kada. senzor cMektuje uslove za nailazak tornada, svoje podatke može da šalje preko satelita. Kao odgovor, satelit šalje koncentrisani mikrotalasni zrak u tom smetu, koji može da naruši razlike u temperaturi neophodne za formiranje tornada.

Drugi primer je imitacija prisustva na daljinu (telepresence), koncept projeklovanja na udaljenu lokaciju. Osnovna ideja je da se preko Internet konekcije upravlja robotizovanim uredajem na udaljenom mestu. Medutim, postoje i druge mogućnosti. Pomoću uredaja nalik onima koji generišu prikaz virtuelne stvarnosti korisnik bi mogao da vidi, čuje, ili možda oseti dogadaje koji se dešavaju na udaljenim mestima i da ima mogućnost reagovanja na njih, a bilo bi idealno ako bi se sve to izvodilo u realnom vremenu. Korisnik bi imao isti osećaj kao da se nalazi na lokaciji udaljenoj i nekoliko hiljada kilometara. Značajna primena bi mogla da bude u robotizovanim uredajima koji reaguju na nesreće na lokacijama nedostupnim za čoveka. Koristeći ovaj koncept, korisnik bi mogao sa bezbedne lokadje da učestvuje u operacijama spašavanja, ili čišćenja.

Na polju zabave, vodeći proizvodač gitara Gibson razvio je tehnologiju pod nazivom *MaGlC* (Media-accelerated Global Information Carrier), koja omogućava digitalizaciju muzike i prenos preko Ethernet konekcije koja je uspostavljena sa gitarom. Ova tehnologija je dizajnirana da bi zamenila analogne priključke i ima svoju primenu prilikom snimanja i montiranja muzike. Verovatno ćete moći da svirate gitaru i da, istovremeno, to snimate na kompjuteru.

- **Elektronski telefonski direktorijumi** Iako će trenutni formati koji se koriste za obične i poslovne imenike biti primarni izvor informacija za većinu ljudi u budućnosti još mnogo godina, elektronski telefonski imenici većpodnju da se naziru. Ideja je jednostavna: koristili bi se PC, ili radna stanica za unos imena osobe, ili kompanije, a zatim bi na ekranu bio prikazan odgovarajući broj telefona. Druga opcija je za unos poslovne kategorije (restorani, honorarni rad, kompjuterske prodavnice, itd) i prikaz liste odgovarajućih kompanija. Selektovanjem bilo koje stavke iz liste dobile bi se kompletne informacije, uključujući broj telefona i adresu.

  Iako je široko dostupan, ovaj servis se još uvek ne koristi masovno (mada će se to naravno promeniti). U stvari, AT&T obezbeduje Web sajtove (na primer, www.att.com/directory) koji obezbeduju ovakve direktorijumske servise. Koristeći ove servise, biraju se gradovi, države, imena osoba, ili poslovne kategorije i sajt nudi tražene informacije, uz pretpostavku da su pronađene u bazi podataka. Osim toga, dobijate pristup svim vrstama ostalih informacija koje se tradicionalno ne nalaze u klasičnim telefonskim imenicima.

Klikom na različite linkove možete da preuzmete mapu prikazane lokacije, ili da dobijete dodatne informacije o sajtu koji ste pronašli, kao što su susedne firme, ili osobe. Ako je reč o firmama, možete da sakupite i osnovne informacije, kao što su podaci o bankrotima, pamicama, ili zalozima. Slične informacije se pronalaze i o individualnim osobama (ponekad uz dodatnu cenu), kao što su bračno stanje, ili podaci o eventualnim razvodima, ili suđenjima. Ako imate slušalice sa ugrađenim mikrofonom, možete da kliknete ikonicu koja aktivira softver za pozivanje tog broja telefona. Možete čak da kliknete link koji obezbeduje slanje cveća preko FTD-a. Pomoću samo jednog klika dobijate više informadja nego što većina ljudi uopšte može da zamisli, mada sve to zahteva ozbiljne zaštitne mere i sprečavanje potencijalnih zloupotreba.

- **Prenosivi telefoni** Iako se teško mogu smatrati novom tehnologijom, mobilni telefoni su danas sasvim uobičajeni za mnoge ljude, koji ih koriste ili u poslovne svrhe, ili ih smatraju nekom vrstom statusnog simbola. Naravno, mnogi imaju mnoštvo drugih razlo ga za njihovo korišćenje. Zamislite da Vam se auto pokvari u nekoj "nedodiji" u 2 sata noću, desetak kilometara od najbližeg telefona, ili da ste svedok neke nesreće na udaljenoj lokaciji. Mobilni telefon bukvalno može da spasi život u nekim situacijama. Eventualno, jednog dana će se telefonske govornice moći videti samo u muzejima tehničkih dostignuća. Prenosivi telefoni omogućavaju i prenosive faks mašine. Ako zbog gužve u saobraćaju kasnite na neki poslovni sastanak, iz automobila možete faksom da pošaljete u kancelariju svoje beleške, ili grafikone prodaje. Nove bežične i satelitske tehnologije su uslovile razvoj nove generacije komunikacione opreme, koja, ne samo da može da uspostavlja pozive, većmože i da pristupa Internetu preko prenosivog uredaja. Vreme u kome ćete ovakve uređaje koristiti za paljenje svetla u kucl, ili da počnete spremanje (ili podgrevanje) večere je verovatno pri kraju rada. Sateliti sada omogućavaju korišćenje mobilnih telefona na lokacijama na kojima nema prijema signala sa tornjeva, kao što su područja visokih planina, ili sredina okeana.

- **Elektronska pošta** Internet već koristi elektronsku poštu (email) za razmenu informacija širom sveta. U prošlosti su ovu tehnologiju koristili samo kompanije, vladine agencije i univerziteti. Medutim, danas je to jedan od najčešđh razloga zbog kojih se ljudi odlučuju na pretplaćivanje na Internet servise. Ipak, sigurno je da postoji veliki broj ljudi koji verovatno nije zainteresovan za ove servise, ili, jednostavno, ne želi da menja svoje navike slanja rukom pisanih pisama. Po njihovom mišljenju, slanje čestitki povodom bilo kog dogadaja ne može da ima istu emotivnu notu kao tradicionalne razglednice. Medutim, sa stasavanjem generacija koje i ne znaju kako je izgledao svet bez kompjutera, društvo će verovatno doći do tačke u kojoj će se začudeno zapitati kako je sve funkcionisalo bez kompjutera (baš kao što veđna nas ne može da zamisli život bez telefona). Očekujemo da će se u budućhosti gro poslovne korespodencije obavljati pomoću emaila. Na primer, email se sve više koristi za komunikaciju izmedu lekara i pacijenata. Pacijenti mogu da pošalju email svom lekaru radi izdavanja recepata, opštih dijagnostičkih pitanja, zakazivanja pregleda i svega onoga zbog čega nije neophodno dolaziti u ordinaciju. Email može da obezbedi praćenje odziva koji treba da redukuju pogrešnu interpretaciju onoga što lekar kaže. Čak će i digitalni potpisi biti čitljivi.

- **Digitalna revolucija** Ovaj koncept je pomalo teško objasniti bez predočavanja razlika između analognih i digitalnih signala, o kojima će biti reči u Poglavlju 3. Za

sada, možete da smatrate da se u komunikacijama koriste isti tipovi signala kao i u kompjuterima. Telefon je još jedan primer kod kog je digitalna tehnologija promenila funkcionisanje. U stvari, osim što signali idu direktno u kuću i od kuće, telefonski sistem je u potpunosti digitalan. Iako veđnu ljudi uopšte i ne interesuje da li se njihovi telefonski razgovori prenose kao analogni, ili digitalni signali (ako me neko pozove i kaže mi da sam dobio na lutriji, definitivno se nedi pitati kakva je tehnologija korišćena za prenos signala), digitalni signali su imali i imaće velikog uticaja na servise koji nam stoje na raspolaganju. Na primer, digitalni signali prenose informacije do odredišnog telefona. Za korisnika koji se pretplaćuje na servis identifikacije poziva na telefonskom uredaju se prikazuju ime i broj telefona osobe koja ga zove. Iako mnogi ove informacije koriste samo za prikazivanje, profesionalci kao što su lekari, advokati, ili brokeri mogu da koriste softver za rutiranje dolazećeg broja telefona do baze podataka za klijente, ili pacijente, iz koje mogu da pribave potrebne informacije pre nego što odgovore na poziv. Ne samo da znaju od koga potiče poziv, već pre započinjanja razgovora mogu da imaju sve relevantne informacije. Sistem može čak da prikaže i predlog odgovora, kao što je: "Dobar dan gospodo Smit, kako su Džek i dečaci?". Tako se postiže prisniji kontakt sa osobom koja zove. Sistem za obaveštavanje policije 911 može da mnogo bolje pomogne ljudima u nevolji, jer se na ekranu ispisuje broj osobe koja nerazgovetno govori, ili ako poziv dolazi od malog deteta. Prikazivanje pozivaočevog broja telefona umnogome redukuje broj poziva kojima se ljudi uznemiravaju. Nedostatak ovog sistema je u ugrožavanju privatnosti. Mnogi ljudi nemaju svoje brojeve telefona zabeležene u imenicima i ne bi trebalo da se ispisuju njihovi pozivi. Naravno, u takvim slučajevima je moguće onesposobiti identifikaciju poziva. Sistem će utvrditi da li je potrebno prikazati broj sa koga se upućuje poziv.

Za nekoliko godina svi televizijski prenosi biće digitalni. Jedna od najvažnijih prednosti ove tehnologije je poboljšanje kvaliteta signala, tako da je obezbeden visok vizuelni i zvučni kvalitet. Naravno, nedostatak je to što će današnji televizori biti zastareli kada nova generacija televizije preuzme primat. I kablovski sistemi će prenositi isključivo digitalne signale, što će dovesti do bolje integracije kompjuterske tehnologije i tehnologija prenosa. Naravno, to će omogućiti brojne interaktivne servise. Gledaoci koji gledaju reklamne programe moći će da naručuju predstavljene proizvode pomoću svojih televizijskih prijemnika.

- **Pristup elektronskim medijumima** Mnogi ljudi redovno odlaze u video klubove da iznajme video kasete, ili odlaze u biblioteke da bi pozajmili neku knjigu, ili potražili nešto u enciklopediji. Sada postoji tehnologija koja može da zameni televiziju dvosmernim komunikacionim uredajem.* Moď ćete da je koristite za pristup bibliotekama filmova, specijalnih izdanja, ili dokumentaraca. Plaćanje po prikazivanju je već uobičajeno na brojnim lokacijama, tako da mogu da se selektuju različiti filmovi u naznačeno vreme. Razlika leži u tome što ćete moći da gledate fllmove koje želite da vidite u vreme koje sami izaberete. Slično tome, moći ćete da pristupate elektronskim knjigama iz biblioteke (uz pretpostavku da su rešeni problemi zaštite autorskih prava). Elektronski pristup verovatno neće zameniti opuštanje uz novu knjigu, ali može da bude korisno za one koji žele činjenične informacije, poput onih koje se nalaze u endklopedijama, ili u vladinim dokumentima. Mnoge poznate novine i magazini u potpunosti mogu Vam biti dostupni u elektronskoj formi, ukoliko imate Intemet konekciju.

Doći će do spajanja pretraživanja Interneta i televizijskih tehnologija. Kompanije će moći da dizajniraju reklame koje neće samo govoriti zašto treba da kupite njihove proizvode, već ćete moći odmah i da naaičite te proizvode koristeđ isti uređaj na kome ste videli reklamu. Marketinške agencije će moći da prate gledanost jednostavnim pristupom TV kanalima koje gledate. Naravno, ovde dolazi do izražaja problematika zaštite privatnosti i etike i mnogi će opravdano zahtevati da im se obezbedi pravo na privatno gledanje programa. Možda ćemo dočekati dan kada ćemo glasati na izborima iz naših domova.

- Satelitski radio Možda deluje čudno, ali poslednji značajni događaj kada je reč o radiju bilo je kreiranje FM opsega još pre 50 godina. Otada se nije promenilo mnogo u navikama ljudi koji slušaju radio. To je u skorije vreme počelo da se menja. Internet radio omogućava osobi za kompjuterom da se pomoću svog pretraživača poveže na radio stanicu i da sluša živu emisiju. Satelitski radio je tehnologija koja omogućava kompanijama da emituju radio emisije preko satelita. Emitovanje se vrši preko više kanala, preko kojih se prenose različiti formati, kao što su popularna, ili klasična muzika, sportski prenosi, ili razgovor. Značajna prednost je u činjenici da korisnici na putu više ne moraju da brinu o dometu signala kada napuštaju lokalna područja. Osim toga, više neće biti ograničeni samo na nekoliko stanica kada putuju u udaljene oblasti. Da bi se olakšalo funkcionisanje satelitskog radija, potrebno je poslati mrežu satelita u orbitu i obezbediti njihovu komunikaciju i prenos do svih tačaka na zemlji.

  Postoje neki problemi u vezi toga da li će ljudi biti spremni da plaćaju usluge satelitskog radija (trenutno je slušanje satelitskog radija moguće samo uz pretplatu) kada mogu besplatno da slušaju konvencionalni radio. Sa druge strane, ovo pitanje se ne razlikuje mnogo od analognog pitanja koje je pre mnogo godina bilo postavljeno u vezi plaćanja TV-a. Pravo pitanje je da li će ljudi biti spremni za plaćanje dodatnih opcija; ako je istorija dobar pokazatelj, odgovor je verovatno potvrdan.

- Video konferencije Svakog dana milioni ljudi provode deo svog radnog dana (ako ne i previše) na raznim sastancima. Ponekad moraju da prelaze velika rastojanja, uz visoke troškove, da bi prisustvovali nekim sastancima. Video konferencije omogućavaju ljudima iz raznih delova sveta da vide i čuju jedni druge kao da se nalaze zajedno. Mogu da razgovaraju i prikazuju grafikone kao da su u istoj prostoriji. Sve do skora, video konferencije nisu bile isplative, jer je za video slike bila neophodna ogromna količina informacija. Medutim, sa napretkom video desktop tehnologije, grafike u visokoj rezoluciji, PC aplikacija, brzine prenosa u mrežama i tehnike kompresije, mnogo štošta je promenjeno u ovoj oblasti. Ono što se nekada smatralo luksuzom, koji su sebi mogle priuštiti samo najveće kompanije, danas mogu da koriste mnoge manje firme. Nije daleko ni dan kada će full-motion video biti dostupan svakom korisniku na mreži.* Ova tehnologija može da se koristi i za programe obuke preko mreže.

*Preuzimanje video klipova sa Interneta je već uobičajeno, ali to nije ono na šta ovde mislimo. Takvi video klipovi se preuzimaju i puštaju lokalno. Mislimo na mogućnost prenosa pokretne slike u realnom vremenu, u trenutku kada se radnja stvarno dešava. To zahteva neka real-time ograničenja koja neki mrežni protokoli još uvek ne mogu da obezbede.

Odeljenje za obuku može da isplanira emitovanje različitih programa za obuku tako da ih osobe rnogu pratiti iz svojih kancelarija, povezujući se na odgovarajući kanal i prateći sesiju koja se prikazuje na ekranu.

- **Trodimenzionalni prikazi** Video slike su dvodimenzionalne. Stari "3-D" filmovi su prikazivani pomoću dve razlidte slike iste stvari, a zatim su posmatrači nosili specijalne naočari. Svako sočivo je filtriralo jednu sliku, tako da je svako oko videlo neznatno drugačiju sliku. Mozak te signale drugačije interpretira, dajući perspektivu onoga što mi nazivamo rastojanje. Možda će jednog dana televiziju zameniti holografske kutije, u kojima će biti prikazivane stvarne trodimenzionalne slike. Medutirn, to se verovatno neće razviti u bJiskoj budućnosti.

  Ali, zašto se zaustaviti na slikama? Ljudi nisu bili zadovoljni samo prenosom glasa. Morali su da osmisle kako da prenesu dvodimenzionalne video slike. Nakon toga su počeli da razmišljaju o trodimenzionalnim slikama. Šta je sa prenosom stvarnih objekata, ili u ekstremnim slučajevima, samih ljudi? Zvuči kao naučna fantastika? Naravno da jeste i to zna svako ko je ikada gledao film "Zvezdane staze". Ali, tako se razmišljalo i o lehnologijama koje danas koristimo. Ipak, ne budite apsolutno sigurni da će tako biti i u ovom slučaju. U članku u referenci [ZeOO] predstavljeno je kvantno teleportovanje, što je većuradeno sa fotonima (česticama od kojih se sastoji svetlost). U tom članku su opisani principi po kojima su "čestice istog tipa u istom kvantnom stanju neraspoznatljive". To nameće logično pitanje da li je moguće skenirati nešto na atomskom nivou, zabeležiti tipove i kvantna stanja svih čestica i preneti te informacije na udaljenu lokaciju, gde bi atomi izvučeni iz materije bili ponovo sastavljeni u skladu sa prenetim instrukcijama. Neki sumnjaju da se teleportovanje nije desilo, već da je kreirana kopija. Drugi mogu da tvrde da ospore njihovo mišljenje tvrdnjom da, ako je objekat na udaljenoj lokaciji zaista identičan sa originalnim do atomskog nivoa, onda se sa svih aspekata može smatrati da Je objekat isti. Nema sumnje da ovakvo pitanje može da inicira dugotrajne rasprave. Bez obzira na sve, prenos fizičkih objekata svakako predstavlja krajnji domet komunikacije.

- **Elektronski lokatori** Sledeća vrsta naizgled naučno-fantastičnih uredaja omogućava ljudima da nose torbu koja centralnom kompjuteru šalje informacije o tačnoj lokaciji. Danas se koriste slični uređaji. Na primer, neke transportne kompanije imaju instalirane odašiljače u svim svojim kamionima. Sistemi su toliko precizni da mogu da lociraju kamion u dornenu od nekoliko metara u bilo kom delu zemlje.

  Ovi uredaji se ugrađuju i u automobile. U automobilu se postavi mali video displej koji prikazuje mapu puta. Korišćenjem sistema satelitskog praćenja može da se prikaže pulsirajuća oznaka odredene lokacije. Ovakav sistem može da bude koristan za nekog ko se izgubio na nekom neobeleženom seoskom putu. Slični sistemi su još korisniji za navigacione sisteme na brodovima koji plove na otvorenim morima (naravno, bez mape puta).

  Ova tehnologija je našla svoju primenu i u mobilnoj telefoniji. Neke kornpanije prodaju mobilne telefone sa mobilnim servisom za pozicioniranje. Zamislite šta bi bilo da se izgubite u divljini, ili da se povredite i ne možete da govorite.

Poziv servisu 911 omogućava spasiocima da otkriju Vašu lokaciju preko mobilnog telefona i da Vam brže ukažu pomoć. Mobilni servisi za pozicioniranje mogu da se koriste i u manjim uređajima i da se postave na ogrlice kućnih ljubimaca, ili, čak, deci na odeću, tako da možete lako da ih pronadete ako se izgube.

- **Glasovne komunikacije** Trenutno se razmena podataka najviše oslanja na kompjuterske uređaje, kao što su radne stanice, skeneri, ekrani za prikaz, ili disk drajvovi. Sofisticirani procesori danas mogu da se nauče da prepoznaju govome šablone. Ovo je izuzetno korisno za hendikepirane osobe, koje bi mogle glasom da kontrolišu invalidska kolica, ili ruku robota. Trenutna tehnologija već omogućava programiranje video rekordera glasom, ili kreiranje tekstualnih dokumenata diktiranjem teksta. Sledeća već primenljiva tehnologija omogućava korisniku kompjutera da govori u mikrofon priključen na softver koji digitalizuje glas i prenosi ga na udaljenu lokaciju preko mreže, gde može da se smesti u mailbox. Osim toga, PC može da se koristi na isti način kao i telefon.

- **NASA Deep Space Netvvork (DSN)** DSN trenutno ima tri komunikaciona postrojenja u Kaliforniji, Spaniji i Australiji. Zbog Zemljine rotacije, ove lokacije su neophodne za neprestano praćenje sondi za svemirska ispitivanja bez posade, a mogu da se koriste za vraćanje slika i slanje komandi za navigaciju. Mreža može da se koristi i za prikupljanje naučnih podataka iz astronomskih opservatorija, kao dodatak naučnim ekperimentima i kao podrška misijama u Zemljinoj orbiti. Pošti su signali od sondi za svemirska ispitivanja veoma slabi, DNS koristi osetljive reflektujuće antene veličine fudbalskog terena.

- **Komuniciranje mislima** Može li kompjuter da čita misli? Pre nego što se podsmehnete ovom pitanju, setite se da ljudski mozak stvara jedinstvene moždane talase koji se mogu detektovati elektroencefalografom. Autor ove knjige je jednom prilikom gledao dokumentarac u kome je čovek bio povezan na takav uredaj dok je gledao sliku lopte koja se pomerala gore, dole, levo i desno; njegovi moždani talasi su zabeleženi i zapamćeni na kompjuteru radi kasnijeg referenciranja. Zatim je ta osoba gledala loptu projektovanu na ekranu i pokušavala da je pomeri zamišljanjem jednog od četiri moguća smera. Kompjuter je uporedio generisane moždane talase iz ove vežbe sa onima kada su izdavane komande za projektovanu sliku u odgovarajućem smeru. Funkcionisalo je! Da li kompjuter može da ćita misli? Procenite sami, ali pratite šta se dešava, jer živimo u uzbudljivom vremenu.

Komunikacione tehnologije predstavljaju velike izazove. Možda se najveđ tiču zaštite privatnosti i bezbednosti. Ogromna količina informacija koja se prenosi kroz vazduh "provocira" beskrupulozne ljude da pokušaju ilegalno da presretnu poslate informacije. Kako sprečiti te ljude da ne vide informacije koje ne treba da vide, ili koje nisu platili? Kako ih sprečiti da ometaju televizijske prenose slanjem sopstvenih poruka? Ovo nisu hipotetičke situacije; većsu se desile.

Šta je sa različitim vladinim polisama u vezi prenosa i razmene informacija? Ne možemo da postavimo kontrolne tačke na nacionalnim granicama radi kontrole informacija.

Da li treba ograničavati i filtrirati informacije? Ako treba, ko definiše polise koje odreduju prikladnost informacija? Kako to razdvojiti od cenzure?

Izazovi su brojni i moramo se pripremiti za suočavanje sa njima. Najpre, moramo da razumemo i njih i tehnologije koje su uslovile njihov nastanak. Vreme je da počnemo.

Pitanja i zadaci za proveru

1. Šta je Morzeov kod?
2. U čemu je razlika izmedu nadmetanja i kolizija?
3. Nabrojte pet organizacija za uspostavljanje standarda i bar po jedan standard za koji je svaka od nabrojanih organizacija odgovorna.
4. Šta je razvodna tabla?
5. Nabrojte pet komunikacionih aplikacija i objasnite kako se mogu koristiti.
6. Sta je otvoreni sistem?
7. Nabrojte pet nadna za organizovanje lokalne mreže.
8. Šta je de fakto standard (dajte primer)?
9. Šta se podrazumeva pod slojevitim protokolom? Zašto se protokoli organizuju po slojevima?
10. Uparite funkcije u pratećoj tabeli sa slojem OSI modela koji ih izvršava.

Sloi OSI modela	Funkcija
Fizički	Definicija aktivnosti
Veze	Baferovanje
Mreže	Nadmetanje
Transportni	Kompresija podataka
Sesije	Definicija električnih karakteristika signala
Predstavljanja	Upravljanje dijalogom
Aplikacije	Elektronska pošta
	Šifrovanje i dešifrovanje
	Detekcija grešaka
	Upostavljanje i oslobadanje konekcije
	Transfer fajlova
	Konvertovanje formata
	Multipleksiranje
	Rutiranje
	Komutacija
	Sinhronizacija
	Prosleđivanie tokena

11. Navedite razliku izmedu parne i neparne parnosti.
12. Navedite razliku izmedu komutacije poruka i komutacije paketa.

13. Da li su sledeće izjave tačne, ili netačne (i zašto)?
    a. Prvi kompjuter je razvijen kao pomoću uspostavljanju komunikacionih sistema.
    b. Telstar je bio satelit dizajniran radi prenosa televizijskih i telefonskih signala preko Atlantskog okeana.
    c. Otvoreni sistem omogućava slobodan pristup različitim računarskim i informacionim servisima.
    d. Kompjuterska mreža može da poveže različite tipove uredaja za skladištenje podataka, iako oni smeštaju podatke u različitim formatima.
    e. Dva para uređaja mogu da komuniciraju istovremeno preko zajedničke magistrale, ako se nalaze na suprotnim krajevima magistrale.
    f. LAN sa potpuno povezanom topologijom je najčešće korišćeni oblik mreže, jer omogućava direktan prenos informacija između bilo koja dva uredaja.
    g. Standardi eliminišu nekonzistentnosti izmedu računarskih uredaja.
    h. Sedmoslojni OSI model bi, da je u potpunosti implementiran, dopustio komunikaciju izmedu bilo koja dva uredaja, obezbedujući fizički prenos informacija izmedu njih.
    i. Slojeviti protokoli omogućavaju implementaciju nižih slojeva nezavisno od viših slojeva i obrnuto.
    j. Datagrami predstavljaju bolji pristup od virtuelnih kola prilikom rešavanja "zagušenja" na mrežama.
    k. Uspostavljanje OSI transportne konekcije zahteva da jedna strana postavi zahtev za konekciju i da druga strana da potvrdu.
14. Koje mrežne topologije omogućavaju prosledivanje tokena?
15. Koji su slojevi OSI modela zaduženi prvenstveno za mrežne operacije?
16. Po čemu se OSI model razlikuje od Internet modela?
17. Prokomentarišite prednosti i nedostatke prikazivanja broja telefona sa koga se upućuje poziv svaki put kada zvoni telefon na pozvanoj strani.
18. U čemu je razlika izmedu razmene podataka i razmene informacija?

Vežbe

1. Skicirajte topologiju LAN mreže u Vašoj školi, ili firmi.
2. Koje ste od moguđh primena navedenih u odeljku 1.1 i sami koristili? Zašto ste ih koristili?
3. Koji su tipovi uredaja povezani na LAN mrežu u Vašoj školi, ili radnom mestu? Da li se verovatno nalaze i u drugim LAN mrežama? Zašto se nalaze, ili ne nalaze?
4. Pretpostavimo da bidirekciona token ring mreža povezuje osam uredaja numerisanih od 1 do 8 u smeru kretanja kazaljki na časovniku. Na kojim uređajima otkazi sprečavaju prenos poruke od uredaja 1 do uredaja 4?

5. Pretpostavimo da mreža iz prethodne vežbe ima n uredaja. Da li je moguće da, i pored otkaza dva uredaja, svi ostali uredaji mogu da komuniciraju? Ako je moguće, pod kakvim se uslovima javlja takva situacija?

6. Navedite četiri rute sa slike 1.9 preko kojih čvor A može da komunicira sa čvorom F. Koliki je ukupan broj ruta (uz pretpostavku da ruta samo jednom prolazi kroz isti čvor)?

7. Razmotrite sledeće okvire:

   011010001010001 x
   100111000101101 X
   100001100011000 X

   Pretpostavimo da je *x* bit parnosti. Koju vrednost mora da ima *x* da bi se uspostavila parna parnost, a koju za nepamu parnost?

8. Objasnite zašto jednostavna provera parnosti ne može da detektuje grešku ako u toku prenosa dođe do promene vrednosti dva bita.

9. Kada u opštem slučaju jednostavna provera parnosti može da detektuje grešku? Kada se greške neće detektovati?

10. Navedite primer video konferencije koja je korišćena u Vašoj školi, ili firmi.

11. Koji će aspekat savremenih komunikacija imati najviše uticaja na Vaš lični i profesionalni život?

12. U odeljku 1.5 opisan je primer u kome osoba izdaje komande kompjuteru za pomeranje slike lopte običnim zamišljanjem podizanja, spuštanja i pomeranja ulevo i udesno. Mnogi mogu opravdano da kažu da je ovo daleko od čitanja misli. Kakvo je Vaše mišljenje?

13. Koliko direktnih konekcija postoji u potpuno povezanoj topologiji sa n čvorova?

14. Navedite neke dodatne primene u kojima uredaji koji ne služe u računarske svrhe mogu da zahtevaju uspostavljanje komunikacije.

15. Pronadite svoj broj telefona u online direktorijumskom servisu, poput onog koji se može pronaći na adresi www.att.com/directory. Možete li da pronadete mapu svog susedstva?

Reference

[EsOO] Estrin, D., R. Govindan and J. Heidemann, "Emedding the Internet." *Communications ofthe ACM,* vol. 43, no. 5 (May 2000.), 39-41

[Ho94] Holtzmann, G., and B. Pehrson. "The First Data Networks". *Scientific American,* January 1994, 124-129.

[ZeOO] Zeilinger, A. "Quantum Teleportation". *Scientific Ameriam,* April 2000, 50-59

POGLAVLJE

2 Medijumi za prenos i kodovi

Informacije su poput nerava koji se prostiru kroz Ijudsko telo i animiraju ga. Senzori i monitori su analogni Ijudskim čulima na osnovu kojih ostvarujemo kontakt sa svetom oko nas. Baze podataka odgovaraju memoriji; informacioni procesori izvršavaju funkciju Ijudskog rezonovanja i razumevanja. Kada se postmoderna infrastruktura integriše na odgovarajući način, umnogome će premašiti Ijudsku inteligenciju u bogatstvu, oštrini, kapacitetu i preciznosti.

— Albert Borgman, američki predavač i autor

Podaci nisu informacije
informacije ne predstavljaju znanje
znanje nije razumevanje
razumevanje nije mudrost

—Cliff Stoll i Gary Schubert

2.1 Uvod

U ovom poglavlju predstavićemo neke osnovne aspekte i modove komunikacija. Počinjemo osnovnom raspravom o signalima i nastavićemo predstavljanjem različitih medijuma koji se koriste za transfer podataka. Nakon toga će biti razmotreni razlozi za postojanje velikog broja različitih načina za prenos podataka. Drugim rečima, objasnićemo koje su prednosti i nedostaci svakog metoda.

Videćemo da mnogi faktori pomažu utvrdivanje koji je najbolji način za povezivanje uređaja:

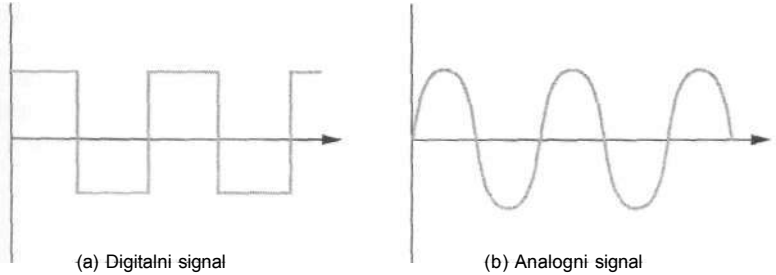
- Cena konekcije
- Količina informacija koje se mogu preneti po jedinici vremena (bitska brzina)
- Imunost na spoljašnje smetnje (šum)
- Osetljivost na neautorizovano "osluškivanje" (bezbednost)

- Logistika (kako se radna stanica povezuje na različite servere u zavisnosti od toga da li se server nalazi na drugom spratu u istoj zgradi, ili u drugoj zgradi preko autoputa sa 12 traka)
- Mobilnost (Da li su komunikacioni uređaji statični? Da li prelaze mala, ili velika rastojanja? Da li ih treba ograničiti na fizičke konekcije kao što su žičani kablovi?)

Kada se uredaji povežu, možda ćete pomisliti da je završen najteži deo posla i da oni mogu lako da komuniciraju. Nažalost, nije tako. Veliki problem leži u načinu na koji se informacije reprezentuju i šalju. Na primer, zamislite dva čoveka koji stoje zajedno i razgovaraju licem u lice. Ako govore istim jezikom, obično dolazi do komunikacije. Ako govore različitim jezicima i ni jedan ne razume šta onaj drugi govori, neće uvek doći do komunikacije.

Razlidti uređaji možda ne reprezentuju, ili ne šalju informacije na isti način. Tako, na primer, ako jedan pokuša da prenese informacije direktno do drugog, drugi možda neće razumeti primljene informacije. Rešenje problema može da bude uspostavljanje komunikacionih standarda. Na primer, mnogi uredaji prenose podatke koristeći **digitalne signale,** koji se mogu predstaviti električnim svojstvima,* tj. sekvencama specifičnih naponskih nivoa. Grafički, često se predstavljaju pomoću četvorougaonih talasnih oblika; na slici 2.1a horizontalna osa predstavlja vreme, a vertikalna nivo napona. Naizmenična promena naponskog nivoa sa visoke na nisku vrednost može simbolično da se predstavi nulama i jedinicama u određenom periodu. Svaka nula, ili jedinica naziva se bit; različiti kodovi kombinuju takve **bitove** i koriste ih za reprezentaciju informacija smeštenih na kompjuteru. LJ odeljku 2.5 predstavićemo neke standardne načine za reprezentovanje informacija, a u odeljku 2.2 različite elektronske medijume preko kojih se signali šalju.

Personalni kompjuteri često komuniciraju pomoću modema, za šta koriste telefonske linije preko kojih se prenose **analogni signali** (slika 2. Ib), ^ koji formiraju kontinuelno promenljive naponske nivoe.



SLIKA 2.1 *Analogni i digitalni signali*

" Digitalni signali imaju i optičku reprezentaciju, koju ćemo predstaviti u odeljku 2.3.

^ lako Je veliki broj telefonskih sistema digitalan, većina telefonskih uredaja šalje i prima analogne signale. Kompjuter obično koristi modem za konvertovanje digitalnih u analogne signale pre prenosa. Modeme cemo predsiaviti u Poglavlju 3.

Korišćenje analognih signala za reprezentovanje podataka i konvertovanje između analognih i digitalnih signala je složeno. Ovaj problem je detaljnije razmotren u Poglavlju 3.

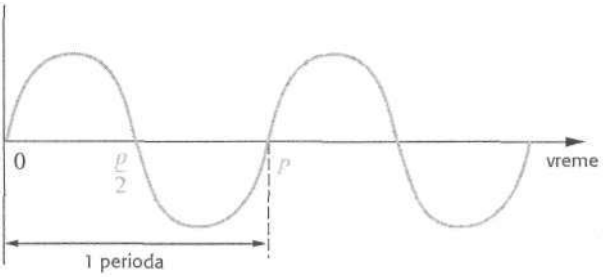
Sledeći korak je utvrdivanje načina na koji signali putuju od jednog mesta do drugog. Postoje tri tipa medijuma za prenos - svaki ima svoje specifičnosti. Prvi tip su provodni metali, kao što su bakar, ili gvožđe, preko kojih se prenose analogni i digitalni signali na prethodno opisani način. Uobičajeni primeri su koaksijalni kabl i kablovi sa upredenim parovima žica. U odeljku 2.2 obradićemo njihove karakteristike. Drugi medijum, koji predstavljamo u odeljku 2.3, je transparentni stakleni kabl (optički fiber), koji prenosi podatke koristeći svetlosne talase. Treći tip ne zahteva flzičke konekcije i u potpunosti se oslanja na elektromagnetne talase, kao što su oni koji se koriste u televizijskim i radio prenosima. Ovde postoje brojne opcije; detaljnije su predstavljene u odeljku 2.4.

Faktor koji se obavezno razmatra prilikom izbora komunikacionog medijuma je cena. Na primer, žičani kablovi, koaksijalni kablovi i optički kablovi imaju različite cene izrade. Osim toga, uredaji na koje se priključuju takode izazivaju dodatne troškove. Sledeći faktor koji treba razmotriti je broj bitova koji komunikacioni medijum treba da prenese u jedinici vremena. Veće cene mogu da budu opravdane ako rezultujući sistem brže prenosi veću količinu informacija.

Bitna su dva merila: bitska brzina i opseg signala. Bitska brzina **(bit** rate) je broj bitova koji se može preneti u jednoj jedinici vremena. Tipična jedinica mere su bitovi u sekundi *{bps - bits per seconds).* U zavisnosti od medijuma i aplikacije, bitske brzine se obično kreću od nekoliko stotina do milijarde bps (gigabita u sekundi), a granice se pomeraju ka opsezima sa terabitima (trilion bitova) u sekundi.

Pre definisanja opsega signala, pogkdaćemo pažljivije kako izgleda signal. Na primer, mnogi analogni signali se predslavljaju pomoću sinusoidalnog uzorka, ili kontinuelnog ciklusa. **Perioda signala** je vreme potrebno za kompletiranje jednog ciklusa. **Perioda** signala sa slike 2.2 je *p.*

Frekvencija signala, /, predstavlja broj ciklusa kroz koje signal može da oscilira u jednoj sekundi. Frekvencija i perioda su "vezane" sledećom formulom:



SLIKA 2.2 *Periodični signal*

Jedinica mere je broj ciklusa u sekundi, ili herc (Hz). Na primer, pretpostavimo da p sa slike 2.2 iznosi 0.5 mikrosekundi (psec). Pošto je 0.5 psec isto što i 0.5 x10 sekundi, frekvencija signala je 1/(0.5 x 10-6), ili 2.000.000 Hz = 2 megaherca (MHz).

Konkretni medijum za prenos može da prenosi signale u konkretnom frekventnom opsegu. Opseg signala (bandvvidth) predstavlja razliku između najviše i najniže frekvencije koju je moguće preneti. Na primer, telefonski signal može da kontroliše frekvencije izmedu 300 i otprilike 3.300 Hz, što daje opseg signala od 3.000 Hz. Sa stanovišta čujnih zvukova, to znači da veoma visok, ili nizak zvuk ne može da prode kroz telefon. Najveći deo ljudskog govora uklapa se Li telefonski opseg i zato se lako prepoznaje. Medutim, gubitak zvukova visoke, ili niske frekvencije stvara probleme onima koji žele da slušaju, na primer, muziciranje Njujorške filharmonije preko telefona.

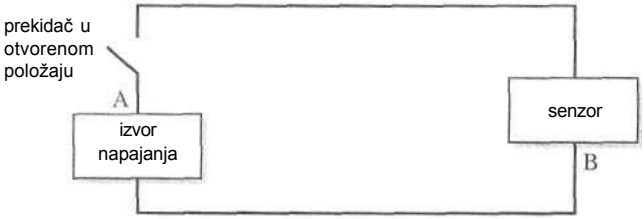
Ponekad se termin opseg signala koristi kada se govori o broju bitova koji može da se prenese. Tehnički, opseg signala i bitske brzine su različiti termini, ali izmedu njih postoji značajna veza. Ova složena relacija detaljnije je opisana u odeljku 3.4.

Da bi se bitovi prenosili izmedu dva uredaja, mora da postoji put kojim će signal putovati između tih uredaja. Tipično, ovo zahteva fizičku konekciju, ili mogućnost korišćenja elektromagnetnih talasa, kao što su oni koji se koriste za radio i teieviziju. U narednih nekoliko odeljaka predstavićemo nekoliko načina za uspostavljanje fizičkih konekcija: kablovi sa upredenim paricama, koaksijalni kabl i optički fiber. Osim toga, prikazaćemo i bežične komunikacije u kojima se koriste infracrveni talasi, mikrotalasni satelitski prenosi i Iaserske tehnologije.

2.2 Provodni metal

Upredene parice

Jedan od najstarijih tipova medijuma za prenos je provodni metal, koji se koristi za prenos informacija još od 1837. godine, kada je Semjuel Morze izmislio telegraf. U osnovi, to je kolo koje ima izvor napajanja, prekidač i senzor (slika 2.3). Prekidač na lokaciji A može da se otvara i zatvara ručno, čime se kontroliše da li struja teče kroz kolo. Senzor na lokaciji B detektuje struju i kreira kliktajući zvuk koji ste imali priliku da čujete na TV-u i u vesternima. Na slici 2.3 prikazan je telegrafski sistem koji omogućava prenos samo u jednom smeru.



SLIKA 2.3 *Jednosmerni telegraf*

Neki drugi dizajni omogućavaju prenos u oba smera (ref. [Sh90]). Otvaranje i zatvaranje prekidača u različitim obrascima kontrolišu frekvenciju i trajanje signala koji se šalju do lokacije B. Poznati Morzeov kod, koji predstavljamo u odeljku 2.5, pridružuje podatke različitim uzorcima signala.

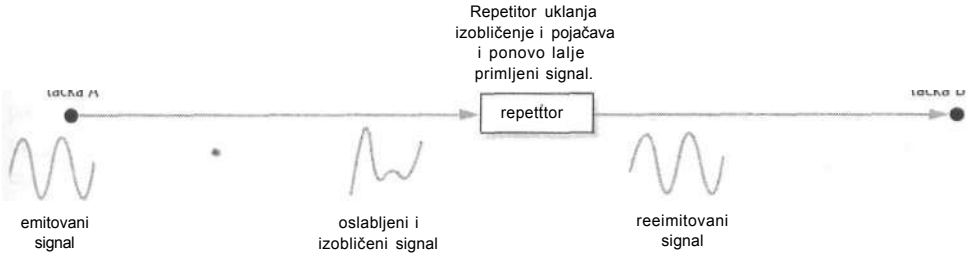
Bakarna žica je verovatno najuobičajeniji način za povezivanje uredaja. Bakar se koristi zbog njegovih električnih provodnih svojstava (elektricitet putuje kroz ovaj metal uz manji otpor nego kroz druge materijale). Osim toga, otporniji je na koroziju nego drugi provodni metali, što je izuzetno bitno ako će kablovi biti izloženi uticaju vlage.

Najčešća primena bakra je u kablovima sa upredenim paricama **(twisted pair),** kod kojih su dve izolovane bakarne žice upredene. Izolacija onemogućava kratke spojeve izmedu provodnika. Kablovi sa upredenim paricama se najčešće koriste za prenos balansiranih signala. To znači da svaka žica prenosi struju, ali su signali fazno pomereni za 180°.* Efekti spoljašnjih elektromagnetnih izvora na struju se skoro anuliraju, tako da je degradacija signala značajno redukovana. Upredanjem žica redukovana je interferenca iz spoljašnjih izvora. Da žice nisu upredene, jedna žica bi mogla biti izložena smetnjama u većoj meri od druge. Upredanjem se interferenca ravnomerno raspoređuje na obe žice i signal, pošto je balansiran, nastoji da ukloni interferencu.

Upredene parice se često omotavaju zaštitnim omotačem, koji omogućava ukopavanje kablova. Takvi kablovi mogu da se označavaju u skladu sa brojem navoja po jedinici dužine žice. Veći broj uplitanja značajno redukuje kapacitivnost, a pomaže i redukovanje preslušavanja (crosstalk), elektromagnentnih smetnji izmedu susednih parica. Ovaj medijum je dosta dugo bio primarni tip za telefonske komunikacije i još uvek se koristi prilikom povezivanja kučnih telefonskih sistema sa najbližom telefonskom centralom. Često se koristi i za povezivanje radnih stanica na mreže.

Iako je bakar dobar provodnik, električni otpor ipak postoji i neophodno je postaviti repetitore izmedu dve tačke (slika 2.4). **Repetitor** je uredaj koji presreće emitovani signal pre nego što se previše izobliči, ili oslabi, a zatim ga ponovo generiše i prenosi dalje ka njegovom odredištu. Ovde se nameće logično pitanje na kom rastojanju je neophodno postaviti repetitore. Rastojanje zavisi od karakteristika, kao što su tip signala, opseg signala i kapacitet žice za prenos struje odredene jačine. Mnogi signali mogu da se prenose miljama pre nego što bude neophodno regenerisanje signala. Sa repetitorima nema nikavog ograničenja u pogledu rastojanja na koje signal može da se prenese.

* Sa druge strane, nebalansirani signali koriste jednu žicu za prenos električne struje, a drugu kao uzemljenje. Nebalansirani signali su obično osetljiviji na interferencu od balansiranih. O tome će biti više reči u odeljku 4.4.



SLIKA 2.4 *Dve povezane tačke između kojih je korišćen repetitor*

Postoje dva tipa kablova sa upredenim paricama: nezaštićene upredene parice (UTP - unshielded twisted pair) i zaštićene upredene parice (STP - shielded twisted pair). Zaštićeni upredeni par ima upleteni metalni omotač preko sloja izolacije, tako da je obezbeđena bolja zaštita od spoljašnje elektromagnetne interference. Osim toga, signali iz jednog para se štite od elektromagnetnog polja koje generiše struja iz drugog para. Nezaštićene upredene parice nemaju takvu zaštitu, ali obično predstavljaju jeftimju varijantu i za proizvodnju i za instalaciju. Ovaj tip je sličan žicama koje se koriste za povezivanje telefona na zidnu utičnicu i danas se često koristi u mnogim mrežnim okruženjima.

Različite kategorije UTP kablova su obeležene numerički. Na primer, ljudi često koriste termin **Cat** 5 **kabl** kada se misli na UTP kabl kategorije 5. Tipično, žice više kategorije imaju veći broj upredanja po jedinici dužine, tako da se redukuje interferenca i postižu se veće bitske brzine. Osim toga, obezbedena je bolja otpornost na **near-end crosstalk** (NEXT), fenomen kod koga signali koji putuju duž jedne žice stvaraju elektromagnetne talase i ometaju signale koji se prostiru duž druge blisko upredene žice. NEXT obično postaje problematičan kod visokofrekventnih signala. U tabeli 2.1 navedene su različite kategorije UTP kablova i njihova tipična primena.

Koaksijalni kabl

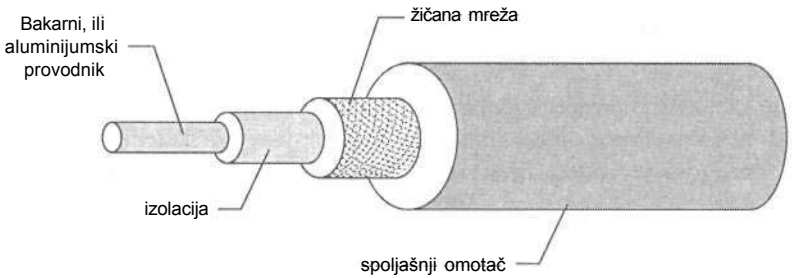
Sledeći tip uobičajenog medijuma je **koaksijalni kabl** (slika 2.5), koji se sastoji od četiri komponente. Prva je unutrašnji provodnik, za koji se, obično, koristi bakarna žica. Kao i kod upredenih parica, jezgro prenosi signal. Izolacioni sloj okružuje jezgro i sprečava provodnik da dode u kontakt sa trećim slojem, obično gusto upredenom žičanom mrežom.* Žičana mreža štiti jezgro od elektromagnetne interference. Osim toga, štiti ga i od gladnih glodara koji traže nezaštićene metalne žice. Poslednji sloj vidite na kablovima pomoću kojih priključujete video rekorder na televizijski uredaj: plastični zaštitni omotač.

Koaksijalni kabl obično prenosi informacije ili u okviru osnovnog opsega, ili u širokopojasnom opsegu. U modu osnovnog opsega **(baseband** mode) opseg signala je rezervisan za jedan niz podataka. Zbog toga, veći opseg signala omogućava prenos sa većim bitskim brzinama.

Tabela 2.1: Kategorije UTP kablova

Kategorija	Primena
Cat 1	Dizajnirana je za primenu u ranim telefonskim sistemima, gde je prvenstveno bio neophodan prenos glasa.
Cat 2	Koristi se u telefonskim sistemima i nekim starijim lokalnim mrežama, kod kojih bitske brzine ne prelaze 4 Mbps.
Cat 3	Postala je standard za kvalitetne govome telefonske komunikacije, a podržava LAN komunikacije do 10 Mbps. Mora da prode testove za preslušavanje susednih signala (near-end crosstalk) do 16 MHz.
Cat 4	Koristi se za Ethernet i token ring mreže kada bitske brzine dostižu 16 Mbps.
Cat 5	Obično se koristi u današnjim mrežama i može da podrži bitske brzine od 100 Mbps na rastojanjima od 100 metara, ili manje. Osim toga, Ethernet mreže sa gigabitskim brzinama razvijene su za korišćenje sa četiri para UTP kablova kategorije 5, u opsegu od 100 metara (ref. [StOO]). Tu je i poboljšani Cat 5 kabl (ponekad se označava kao Cat 5E), koji, u suštini, predstavlja kvalitetniji kabl - podržava bitske brzine do gigabita u sekundi. Korisnici mreža mogu da prepoznaju Cat 5 kabl po plavom omotaču, koji se postavlja od kompjutera do utičnice u zidu. Mora da prode testove na pres lušavanje susednih signala do 100 MHz.
Cat 6	TIA je skoro odobrila standard za Cat 6 kablove, koji obezbeduje dvostruki veći opseg signala u poređenju sa Cat 5E kablovima i ima poboljšane karakteristike koeficijenta signal-šum.
Cat 7	Trenutno se priprema donošenje standarda za kategoriju 7, koji bi obezbedio veće bitske brzine (u vreme kada je ova knjiga pisana, to još uvek nije bilo završeno).

Ovo je tipično za lokalne mreže, gde se u jednom trenutku prenosi samo jedan niz podataka. Kod širokopojasnih opsega (broadband) opseg signala se deli po pojasevima. U svakom pojasu se obično prenose zasebne kodne informacije, tako da je omogućen istovremeni prenos više nizova podataka preko istog kabla. Za kombinovanje signala na izvoru i njihovo razdvajanje na odredištu koristi se specijalna oprema. Kablovska televizija je primer istovremenog prenosa višestrukih signala (po jedan za svaki kanal) preko istog kabla. Predstavljanje kombinovanja signala nastavićemo u odeljku 4.5, kada budemo govorili o multipleksiranju.



SLIKA 2.5 *Koaksijalni kabl*

Dva tipa koaksijalnih kablova često se nazivaju **ThickNet** i **ThinNet.*** ThinNet je tanji, fleksibilniji kabl, koji je ranije korišćen za povezivanje uredaja u kompjuterskim Iaboratorijama. Ova fleksibilnost ga je učinila idealnim izborom u situacijama kada je potrebno postaviti kabl po ćoškovima i oko stolova. ThickNet je više korišćen u starijim mrežama i još uvek može da se pronade u nekim instalacijama. Pošto je manje fleksibilan, njegova primena je ograničena samo na povezivanje opreme koja se nalazi u različitim zgradarna, ili kada je potrebno povezati opremu na različitim spratovima u jednoj zgradi.

Koaksijalni kablovi odavno mogu da postignu veće opsege signala od kablova sa upredenim paricama i dopuštaju veća rastojanja. Osim toga, obezbedena je bolja zaštita od električne interference. Kao i kod kablova sa upredenim paricama, maksimalne bitske brzine i rastojanja menjaju se u skladu sa razvojem novih tehnologija koje omogučavaju proizvodnju sofisticiranije opreme i bolje tehnike signalizacije. Iako su ranije korišćeni kao primami nosači za prenos signala preko glavnog stuba lokalnih mreža, koaksijalni kablovi se danas obično zamenjuju optičkim fiber kablovima (predstavićemo ih u sledećem odeljku), ili, čak, UTP kablovima. Zbog zaštitnog omotača, koaksijalni kabl ima bolju statistiku u pogledu grešaka od UTP kablova, mada su UTP kablovi jeftiniji i lakši za rad, što je značajna stavka kod IAN mreža kada je neophodno uklopiti se u ograničena budžetska sredstva. Medutim, koaksijalni kabl može da bude poželjan izbor kada se manji broj uredaja smešta na manjem podmčju. LI takvim slučajevima zaštitni omotač koaksijalnog kabla obezbeduje bolju zaštitu od interference koja potiče od susednih kablova.

2.3 Optički fiber

Nekoliko problema (ili bar ograničenja) prati prenos podataka preko provodnog metala. ledan problem je činjenica da su električni signali osetljivi na smetnje iz spoljašnjih izvora, kao što su elektromotori, udari munje i drugi kablovi. U odeljku 3.4 ćete videti da interferenca ograničava količinu podataka koju je moguće preneti. Osim toga, kablovi su teški i nezgrapni, posebno ako ih je potrebno zajedno omotati. Zamislite kako izgleda prenos kablova od nekoliko hiljada stopa. U stvari, prema AT&T, namotaj optičkog fibera od 4,5 funti može da prenese istu količinu informacija kao i 200 kalemova žičanog kabla koji teži više od 1.600 funti.

Ova težina predstavlja ograničenje prilikom instalacije kablova u prometnim oblastima, ili na teško dostupnim mestima, kao što su ormari, ili dugački uski prolazi. Osim toga, karakteristike električnih signala i svojstva otpornosti provodnog metala postavljaju ograničenja i u pogledu tipova signala koje je moguće prenositi i u pogledu rastojanja koje signali mogu da predu bez izobličavanja. Ova ograničenja postavljaju granicu i u količini informacija koje je moguće poslati u jedinici vremena.

Alternativa je optički fiber. On za prenos informacija koristi svetlost, a ne elektricitet. Telefonske kompanije široko primenjuju **optički fiber,** posebno za pmžanje usluga na velikim rastojanjima. Otklonjene su mogućnosti za električni šum, a postoji kapacitet za prenos ogromnih količina informacija. Osim toga, optički fiber je veoma tanak (u poredenju sa običnim kablovima), tako da može da se vezuje veća količina kablova na manjem prostoru, nego što je moguće sa tradicionalnim kablovima.

U situacijama kada je potrebno postaviti kablove kroz ispuste, iznad plafona, ili između zidova ovo je velika prednost. Optički fiber se danas standardno koristi u mnogim kompjuterskim mrežama. CD-i visokog kapaciteta, DVD tehnologija i sve veća integracija kompjutera i videa postavljaju zahteve za mrežama sa većim opsegom signala.

Principi koji su omogućili realizovanje optičkog fibera potiču iz fizike, posebno optike i teorije o elektromagnetnim talasima. Ovde nemamo nameru da detaljnije obradimo takve teme, mada smatramo da je bitno da se istaknu neke osnovne ideje da biste bolje razumeli način na koji optički fiber funkcioniše.

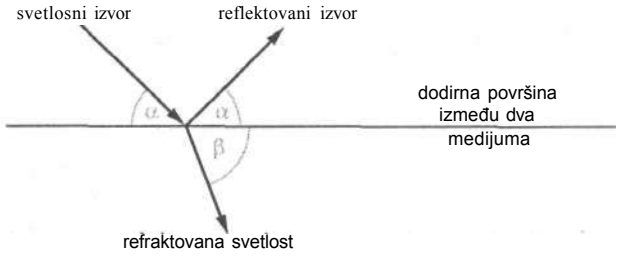
Za početak, razmotrimo svetlosni izvor usmeren ka nekoj površini (slika 2.6). Površina predstavlja granicu izmedu dva medijuma, kao što su vazduh i voda. Neka a bude ugao pod kojim svetlo seče graničnu oblast. Jedan deo svetlosti se reflektuje nazad pod uglom a u odnosu na ravan, a drugi deo prolazi kroz granicu do drugog medijuma. To je refrakcija (prelamanje). Medutim, na granici dolazi do promene ugla pod kojim se svetlost prostire. Drugim rečima, ako je p ugao pod kojim svetlosni talasi putuju od granice, $\beta \neq \alpha$. Zatim, možete da se zapitate da li je β veće, ili manje od α. Ako ste to pomislili, čestitamo. To je dobro pitanje.

Ako je $\beta > \alpha$ (kao na slici 2.6), kažemo da drugi medijum ima veću optičku gustinu od prvog (kao što voda ima veću gustinu od vazduha). Medutim, ako prvi medijum ima veću optičku gustinu, onda je $\beta < \alpha$. Refrakcija objašnjava zašto sočiva u naočarima izobličavaju normalni prikaz, ili zašto objekti koji se nalaze ispod vodene površine izgledaju izobličeno kada se gledaju iznad površine vode. Svetlo koje se reflektuje od objekata je izobličeno i zato oni izgledaju drugačije.

Relacija izmedu β i α je interesantna. Fizičari za njeno opisivanje koriste meru koja je poznata kao indeks refrakcije (koeficijent brzine svetlosti u vakuumu i brzine svetlosti u specifičnom medijumu). Osim toga, dobro poznati rezultat u fizici Šnelovo (Snell) pravilo "kaže" da je koeficijent indeksa refrakcije dva različita medijuma (prikazana na slici 2.6) jednak odnosu

$$\frac{\cos(\alpha)}{\cos(\beta)}$$

Ako je ovaj odnos manji od 1, svetlost putuje u medijum sa manjom optičkom gustinom, a ako je odnos veći od 1, znači da svetlost putuje u medijum sa većom optičkom gustinom.



SLIKA 2.6 *Refrakcija i refleksija svetlosti*

Zapamtite da su β i α uglovi izmedu 0° i 90°, tako da cos(α)<cos(β) znači da je α < β i obratno.

Sledeći interesantan fenomen se javlja kada je ovaj koeficijent manji od 1 (α > β). Kada je a manje od određenog kritičnog ugla, nema refraktovane svetlosti. Drugim rečima, sva svetlost se reflektuje. Zahvaljujući ovom fenomenu, omogućeno je funkcionisanje optičkog fibera.

Tri glavne komponente fiber optičkog vlakna su jezgro, obloga i zaštitni omotač. Jezgro je sastavljeno od čistog stakla, ili plastičnog materijala. Obloga okružuje jezgro. I ona je od stakla, ili plastike, ali sa manjom optičkom gustinom od jezgra. Koliko je jezgro čisto? Kao što ćete videti, optički fiber funkcioniše tako što omogučava prenos svetlosti kroz jezgro. U nekim slučajevima fiber može da bude dužine i do 20 milja. Pošto se svetlost prenosi sa jednog kraja na drugi, možemo da smatramo da je jezgro debljine 20 milja (zamislite stakleni blok koji je toliko čist da je kriška debljine 20 milja skoro transparentna).

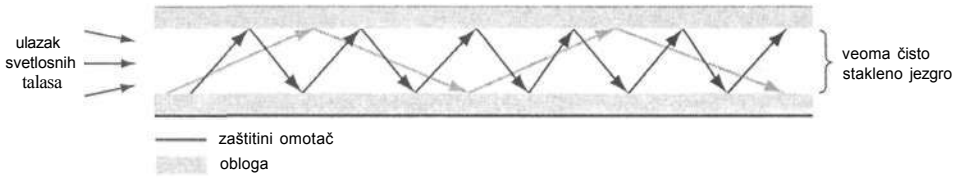
Sledeće pitanje je kako svetlost dospeva u fiber. Na jedan kraj fibera se postavlja svetlosni izvor, kao što su **LED** dioda **(light emitting diode),** ili laser. U oba slučaja je reč o uredaju koji reaguje na električni naboj za stvaranje svetlosnog impulsa, koji se, obično, nalazi blizu infracrvenog opsega od 10^{14} Hz. Laser stvara veoma čist* i uzak mlaz. Osim toga, ima visok izlazni naboj, tako da svetlost može da se prenese na veće daljine, nego u slučaju LED dioda. LED diode proizvode manje koncentrisanu svetlost koja ima širi spektar talasnih dužina. One su jeftinije i u opštem slučaju traju duže. Laseri se koriste tamo gde je neophodan veliki kapacitet prenosa podataka na velikim rastojanjima, kao u slučaju telefonskih linija koje povezuju udaljena područja.

Svetlosni izvor emituje kratke, ali brze svetlosne impulse koji ulaze u jezgro pod različitim uglovima. Svetlost pogada granicu jezgro/obloga pod uglom koji je manji od kritičnog ugla i u potpunosti se reflektuje nazad u jezgro, gde eventualno "pogada" granicu na njegovoj drugoj strani. Efekat je odbijanje svetlosti od granice do granice, dok se prostire niz jezgro. Eventualno, svetlost postoji unutar jezgra i može da se detektuje senzorom. Svetlost koja "pogađa" granicu pod uglom većim od kritičnog ugla parcijalno se refraktuje u oblozi i apsorbuje je zaštitni omotač. Tako se sprečava odavanje svetlosti i apsorbovanje u susednim fiberima.

Prilikom prostiranja svetlosti kroz jezgro fibera može doć do jednog problema. Ako je jezgro prilično debelo u odnosu na talasnu dužinu svedosti, svetlost "pogada" različita mesta pod različitim uglovima. Jedan deo svetlosti se prostire kroz centar jezgra, a preostali deo "pogada" granice jezgra (slika 2.7). Proučavanje elektromagnemih talasa (posebno Maksvelovih jednačina) ukazuje da dolazi do interference izmedu nekih refiektovanih svedosnih talasa. Zbog toga, postoji konačan broj uglova pod kojim se zraci reflektuju i prostiru duž fibera. Svaki ugao defmiše putanju, ili mod. Fiber koji prenosi svedost na ovakav način naziva se **step-index multimode fiber.**

Svetlo koje se reflektuje pod većim uglovima (mereno u odnosu na horizontalu) reflektuje se češće i prelazi veća rastojanja od svetlosti koja se reflektuje pod manjim uglom. Zato je potrebno više vremena za dolazak svetlosnog impulsa na drugi kraj fibera.

* Svetlost se često sastoji od većeg broja različitih talasnih dužina, ili boja, što se vidi na primeru svetlosti koja prolazi kroz prizmu i deli se na dugine boje. Laser može da proizvede "čistu" svetlost, ili svetlo koje se sastoji od svega nekoliko taiasnih dužina.



SLIKA 2.7 *Step-index multimode fiber*

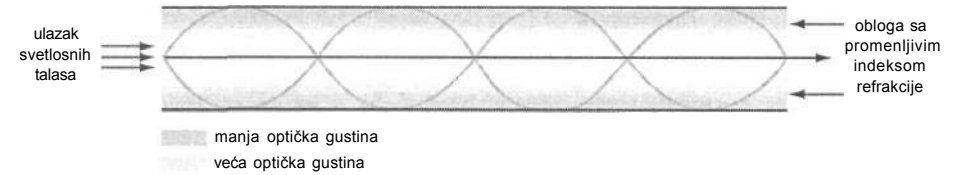
Ovaj fenomen se naziva modalna disperzija. Zamislite grupu ljudi koja trči kroz hodnik. Neki trče pravo kroz centar, dok drugi, koji imaju povez preko očiju, udaraju o zidove. Zna se ko će pobediti u ovakvoj trci.

Modalna disperzija je problem ako je fiber isuviše dugačak. Svetlost iz jednog impulsa (koja se reflektuje pod malim uglovima) može da zahvati svetlost emitovanu iz prethodnih impulsa (koja se reflektuje pod većim uglovima), čime se eliminišu razmaci koji se stvaraju izmedu njih. Senzori više ne vide svetlosne impulse; za njih je to ravnomeran svetlosni zrak i uništavaju se sve informacije koje su bile kodirane u impulsima.

Jedan od načina za rešavanje problema modalne disperzije je korišćenje graded-index multimode fibera. To omogućava korišćenje drugog fenomena koji se tiče činjenice da brzina svetlosti zavisi od medijuma kroz koji se prostire; svetlost brže putuje kroz medijum sa manjom optičkom gustinom.

Graded-index multimode fiber (slika 2.8) takode ima jezgro, oblogu i zaštitni omotač. Razlika je u tome što granica između jezgra i obloge nije strogo definisana. Dmgim rečima, dok se radijalno udaljavate od jezgra, materijalu se postepeno smanjuje optička gustina. Zbog toga se sva svetlost koja se radijalno prostire od jezgra vraća nazad ka centru i, eventualno, reflektuje nazad. Pošto se materijalu smanjuje optička gustina, svetlost brže putuje. Krajnji rezultat je da svetlost, iako prelazi veća rastojanja, putuje brže i tako se redukuje modalna disperzija.

Drugi način da se reši problem modalne disperzije je njena eliminacija. Verovatno se pitate kako to izvesti. Ranije smo rekli da postoji konačan broj modova za prostiranje svetlosnih zraka. Tačan broj zavisi od prečnika jezgra i talasne dužine svetlosnog zraka. Redukovanjem prečnika jezgra smanjuje se broj uglova pod kojima svetlost "pogada" granice jezgra.



SLIKA 2.8 *Graded-index multimode fiber*

Naravno, na taj način se smanjuje i broj modova. Ako redukujemo prečnik u dovoljnoj meri, fiber će imati samo jedan mod. U tom slučaju, dobijamo fiber sa jednim modom (single-mode fiber), kao na slici 2.9.

Do koje mere možemo da smanjujemo prečnik? Drugi princip fizike tiče se mogućnosti za reflektovanje elektromagnetnih talasa (kao što je svetlost) do veličine refleklora. Znači, da bi se svetlost reflektovala na način koji smo opisali, reflektor mora da bude veći od talasne dužine reflektovane svetlosti. Pošto je reflektor obavijen oko jezgra, njegova veličina zavisi od prečnika jezgra. Izmedu frekvencije i talasne dužine postoji sledeća relacija

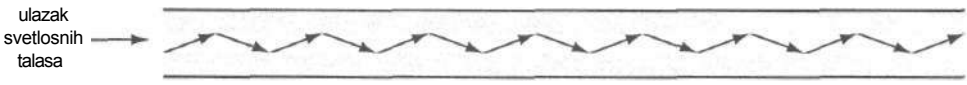
talasna dužina = brzina svetlosti/frekvencija

Svetlost koja se prostire kroz optičke fiber kablove ima frekvenciju aproksimativno od *W"* Hz, tako da talasne dužine iznose otprilike $2x10^{-6}$ metara, ili 2 mikrona (1 mikron = 10^{-6} metara). Dakle, single-mode fiberi obično imaju prečnike koji se mere u mikronima (obično 4-8, mada ponekad i manje) i veoma su tanki (mnogi fiber kablovi su debljine ljudske vlasi). Ovako mali prečnici čine fiber krhkijim i težim za uplitanje. Zato se najčešće koriste za dalekovode, gde su manipulisanje i uplitanje minimalni.

Optička fiber tehnologija ima brojne prednosti u odnosu na provodne metale:

- Podaci mogu da se prenose na velikim brzinama.
- Postoji veoma nizak otpor. Zbog toga, signal može da prelazi veća rastojanja bez postavljanja repetitora. Na primer, repetitori mogu da se postave na svakih 30 milja; trađicionalni kablovi sa provodnim metalom zahtevaju postavljanje repetitora na znatno kraćim rastojanjima.
- Na optički fiber ne utiče elektromagnetna referenca, jer se signali prenose pomoću svetlosti.
- Postoji veoma visoka otpornost na nepovoljne faktore okruženja, kao što je vlaga. Zbog toga je ovo poželjna opcija u priobalnim područjima.

Sa druge strane, današnji kompjuteri su elektronski uredaji i zato korišćenje optičkih fiber kablova zahteva konvertovanje električnih signala u svetlosne zrake i obratno. Ovaj proces uvodi dodatni nivo složenosti. Osim toga, optički fiber kablovi se teže priključuju i upliću nego bakarni vodovi. Komponente se lako dodaju priključivanjem na bakarne vodove, dok je za priključivanje na stakleni fiber potrebna mnogo veća pažnja. U referenci [RoOl] možete da pročitate detaliniji uvod u optičke fiber komunikacije.



SLIKA 2.9 *Single-mode fiber*

2.4 Bežične komunikacije

Svi komunikacioni modovi koji koriste provodni metal, ili optičke fiber kablove imaju nešto zajedničko: uredaji koji komuniciraju moraju da budu fizički povezani. Fizičke konekcije su sasvim zadovoljavajuće u brojnim slučajevima, kao što su povezivanje personalnih kompjutera u kancelariji, ili povezivanje na server koji se nalazi u istoj zgradi. To je prihvatljiv koncept na manjim rastojanjima, ali je na vedm udaljenostima veoma skup i teško se održava. Zamislite koaksijalni kabl koji povezuje kontrolu leta u NASA-i sa spejs šatlom, ili kabl koji povezuje kontrolne tornjeve na aerodromima u Njujorku i Londonu!

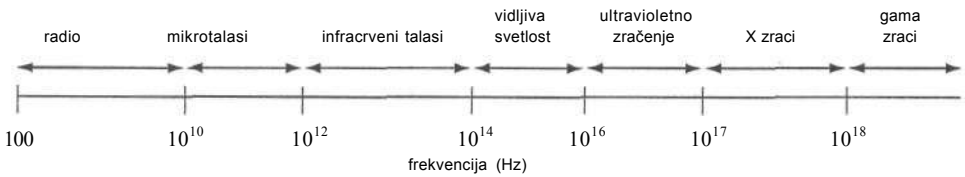
U mnogim situacijama fizička konekcija nije praktična, a često nije ni izvodljiva. Recimo, treba povezati delove mreže u dve različite zgrade izmedu kojih se nalazi autoput sa osam traka. Zavod za urbanizam sigurno ne bi odobrio postavljanje kabla preko autoputa, niti prekid saobraćaja radi postavljanja kablova ispod zemlje. Neophodno je pronaći način za komunikaciju delova mreže bez fizičkih konekcija, tj. za bežičnu komunikaciju.

Bežični prenos uključuje elektromagnetne talase. Pošto je rasprava o elektromagnetnim talasima predmet posebnog kursa, ovde se nećemo upuštati u detaljnija objašnjenja. Za naše potrebe dovoljno je da kažemo da je reč o oscilujudm elektromagnetnim talasima koji se indukuju iz predajne antene. Nakon toga, talasi putuju kroz vazduh, ili slobodan prostor, gde ih prijemna antena može registrovati. Na ovaj način se vrši emitovanje radio i televizijskih signala.

Na slici 2.10 dat je spektar elektromagnetnih talasa. Radio talasi se koriste i za radio prenos i za televizijski prenos. Na primer, televizijski VHF (very-high frequency) opseg nalazi se između 30 i 300 MHz, dok UHF (ultra-high frequency) opseg zauzima područje između 300 MHz i 3 GHz. * Radio talasi se koriste i za AM i FM radio, koriste ih radio amateri, mobilni telefoni i radio na kratkim talasima. Svim tipovima ovih komunikacija frekvenciju dodeljuje Federal Communications Comission (FCC). Neka svojstva elektromagnetnog zračenja su veoma bitna za komunikacije. Jedno od njih je ranije pomenuta relacija izmedu talasne dužine i frekvencije:

talasna dužina = brzina svetlosti/frekvencija

Znači, visokofrekventni talasi imaju kraće talasne dužine i obratno. U tabeli 2.2 možete da vidite neke konkretne vrednosti.



SLIKA 2.10 *Spektar elektromagnetnih talasa*

* MHz je megaherc, ili 10^6 Hz. GHz je gigaherc, ili IO^5 Hz.

Tabela 2.2: Talasne dužine u funkciji frekvencije

Frekvencija (Hz)	Aproksimativna talasna dužina (u metrima)
102	3 x 106
104	3 x 104
106	300
108	3
1010	0.03
1012	0.0003

Niskofrekventni talasi, kada se emituju od zemlje, teže da se reflektuju u višim slojevima atmosfere sa manjim gubitkom. Odskakanje talasa izmedu atmosfere i zemlje, tako da bude moguć prenos signala, prati zakrivljenost Zemljine kugle. Na primer, kratki radio talasi (izmedu 3 i 30 Mhz) poznati su po prijemu signala sa polovine zemaljske kugle. Visokofrekventni signali "nastoje" da se reflektuju sa većirn gubitkom i obično ne prelaze tolika rastojanja (mereno u odnosu na Zemljinu površinu).

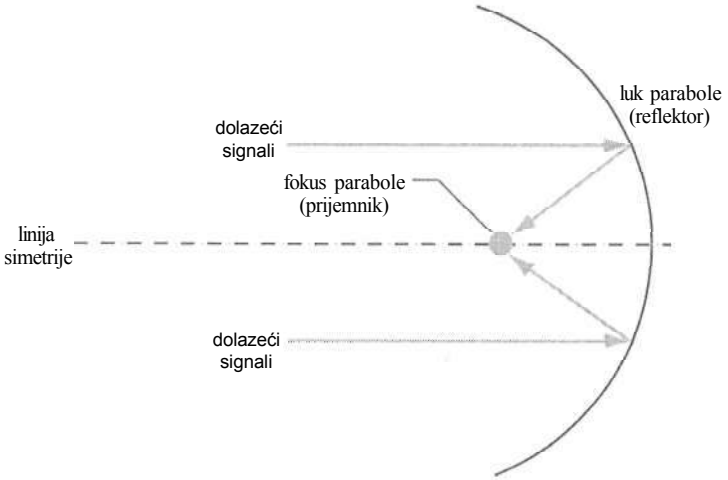
Niskofrekventni talasi zahtevaju veoma dugačke antene. Momarica je 70-ih i 80-ih godina prošlog veka pokušala da instalira veliku antenu (dužine preko 50 milja) u gomjoj polovini poluostrva Mičigen. Trebalo je da se obezbede podmorničke komunikacije sa signalima izuzetno niske frekvencije (ELF signali) u opsegu manjem od 300 Hz. Projekat ELF je bio sporan zbog štetnog uticaja na zdravlje Ijudi koji su bili izloženi elektromagnetnom zračenju. Čak i u današnje vreme postoji rizik po zdravlje ljudi koji su izloženi zračenju izvora napajanja i kompjuterskih terminala.

Posebno su značajna tri tipa bežičnih komunikacija: mikrotalasni, satelitski i infracrveni prenosi.

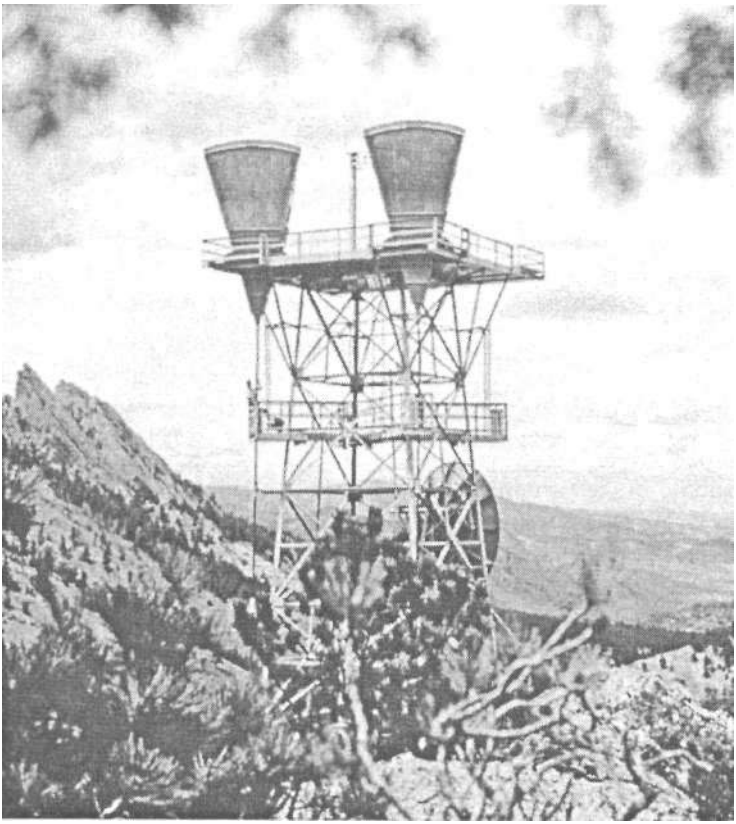
Mikrotalasni prenos

Mikrotalasni prenos se obično koristi izmedu dve stanice na Zemlji. Korišćenje mikrotalasnog prenosa ograničavaju dva svojstva. Prvo, mikrotalasi putuju po pravoj liniji i ne slede zakrivljenost Zemljine kugle, što je moguće kod nekih niskofrekventnih talasa. Drugo, atmosferski uslovi i čvrsti objekti ometaju prenos mikrotalasa, koji, na primer, oni ne mogu da putuju kroz gradevine.

Tipični mehanizam za prenos i prijem mikrotalasnih signala je parabolični tanjir (slika 2.11). Nema sumnje da ste ovakve antene vidali u mnogim dvorištima i na krovovima - za prijem signala satelitskih TV programa, na vrhovirna zgrada, ili na tornjevima u pustim oblastima (slika 2.12). Tornjevi se, uglavnom, koriste za telefonske komunikacije. U skorije vreme veoma su popularne 18-inčne satelitske antene, pomoću kojih se primaju signali brojnih televizijskih stanica i za koje postoji mogućnost plaćanja na osnovu selekcije kanala.



SLIKA 2.11 *Prijem signala na anteni u obliku parabole*

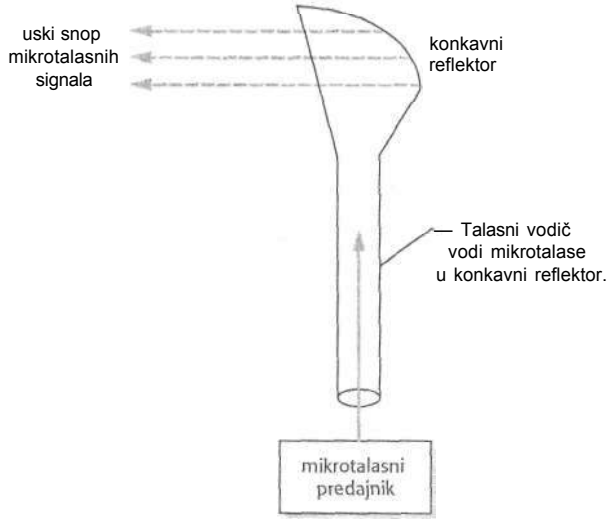


SLIKA 2.12 *Mikrotalasni emisioni toranj*

Parabolične antene koriste dobro poznatu, mada verovatno zaboravljenu činjenicu iz osnovnih studija matematike. Za konkretnu parabolu nacrtajte pravu liniju okomitu na liniju tangente u verteksu (temenoj tački) krive. Ovo je linija simetrije. Sve linije paralelne liniji simetrije reflektuju talase od antene i seku se u zajedničkoj tački koja je poznata kao fokus. Na slici 2.11 prikazano je kako se ovo primenjuje prilikom prijema emitovanih talasa. Stvarna antena nije prijemnik, već samo reflektor. Pošto ima oblik parabole, dolazeći signali se reflektuju i seku u fokusu. Postavljanjem stvarnog prijemnika omogućen je ispravan prijem signala. Napomenimo da se parabolični reflektori koriste za ilustraciju fenomena šapata u muzejima. Dva parabolična tanjira se postavljaju na suprotnim krajevima prostorije. Osoba u fokalnoj tački govori tiho i glas se reflektuje direktno na drugi tanjir. Druga osoba koja sluša u fokalnoj tački čuje glas prve osobe.

Drugi tip antente su antene u obliku roga (horn antenna), kao na slici 2.13. Predajne antene su često ovog tipa (na slici 2.12 prikazane su oba tipa). Antena u obliku roga sastoji od cilindrične cevi pod nazivom talasni vodič (waveguide). Ponaša se kao vodič talasa i reflektuje mikrotalase u uskom snopu. Snop putuje preko neometane oblasti i eventualno ga prima druga antena. Sledeći put kada se budete vozili van grada pogledajte pažljivije unaokolo i možda ćete videti oba tipa antena na tornjevima.

Pošto mora da postoji direktna linija viđenja izmedu predajnika i prijemnika, postoji ograničenje maksimalnog rastojanja na kome mogu da se postave. Granica zavisi od visine tornja, zakrivljenosti terena i tipa terena izmedu njih. Na primer, antene na visokim tornjevima postavljenim u ravnici mogu da "pokriju" velika rastojanja, obično 20 do 30 milja, iako se viši tornjevi konstruišu na vrhovima brda, radi povećanja rastojanja.



SLIKA 2.13 *Antena u obliku roga*

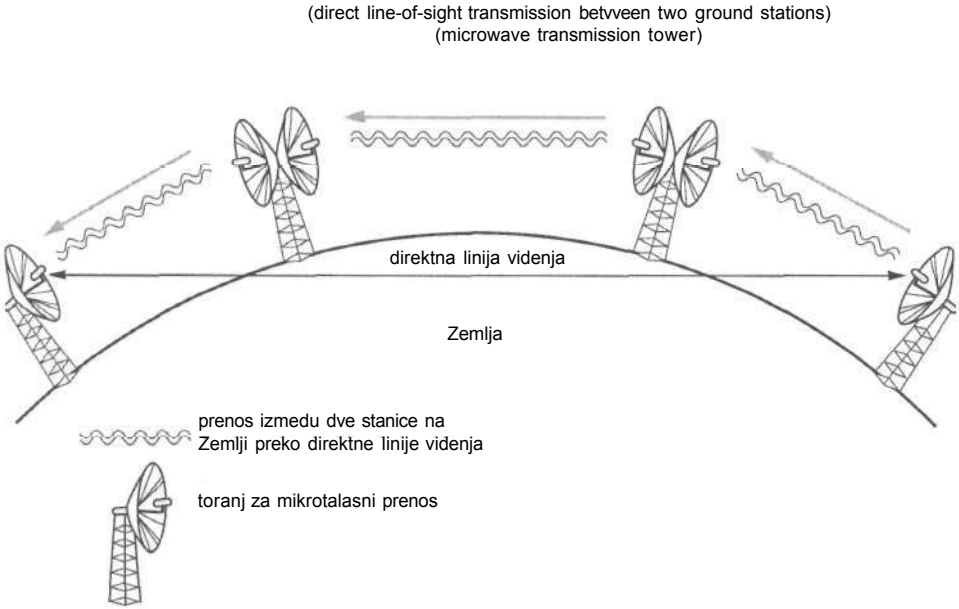
U nekim slučajevima antene se postavljaju na kratkim rastojanjima u okviru grada. Međutim, tu postoji problem ako neko izgradi objekat na direktnoj liniji videnja.

Ako se signali moraju prenositi na velikim rastojanjima, moguće je postaviti nekoliko tornjeva repetitora izmedu antena (slika 2.14). Jedna antena prenosi signale do svog suseda, koji ih dalje prenosi do sledećeg suseda i tako redom. Na taj način je omogućen prenos signala i izmedu tačaka medu kojima ne postoji direktna linija videnja.

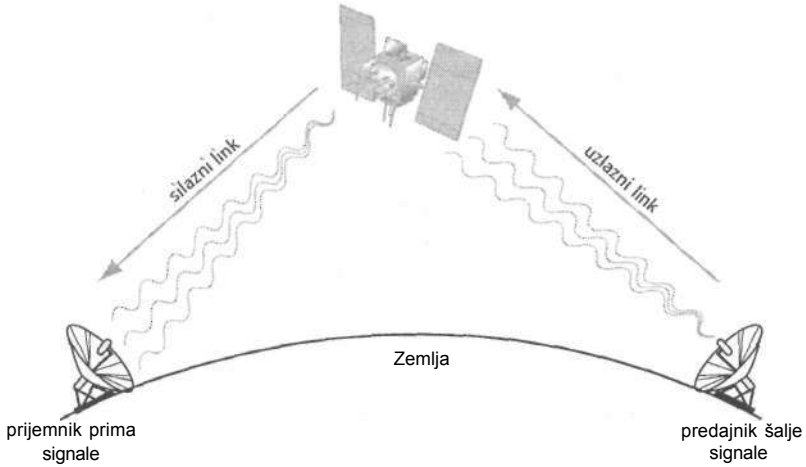
Satelitski prenos

Satelitski prenos je prvenstveno mikrotalasni prenos preko jednog satelita koji se nalazi u Zemljinoj orbiti (slika 2.15). Sigurno je jedno od najčešcih komunikacionih sredstava koja se danas koriste. Koristi se za telefoniju, televiziju, servise za vesti, vremenske prognoze i vojne svrhe, a neki predvidaju da će doći dan kada će to biti sredstvo za povezivanje na Internet (ref. [MeOO]), čime bi se omogućio pristup sa mesta na kojima zemaljski sistemi nisu praktični, ili su, čak, neizvodljivi.

Verovatno ste većčuli za Artura Klarka, autora trilogije "Odiseja u svemiru". Neki možda ne znaju da je Klark bio fizičar i da je 1945. godine pisao o mogućnostima korišćenja satelita za komuniciranje ljudi širom sveta (ref. [C145]). U to vreme takva ideja se smatrala naučnom fantastikom; danas je to nešto sasvim uobičajeno. Istovremeno, Klark nije verovao da bi satelitske komunikacije bile ekonomski isplative, ili tehnički izvodljive pre 21. veka (ref. [Hu90]). Medutim, trebalo je da prode samo 20 godina da se kreiraju tranzistori i satelitske komunikacije postale su stvarnost.



SLIKA 2.14 *Mikrotalasni tornjevi korišćeni kao repetitori*



SLIKA 2.15 *Satelitske komunikacije*

Počelo je 4. oktobra 1957. godine, istorijskim dogadajem koji je šokirao američku naciju i njene političke lidere. Tog dana, naime, Sovjetski Savez je lansirao u svemir satelit "Sputnik". Satelit je ušao u nisku orbitu (na 560 milja) i poslao elektronski "beep" na zemlju. Kako se kretao na nebu, tako je morala da se rotira i stanica na zemlji na kojoj se nalazila antena da bi bilo moguće njegovo praćenje. U poredenju sa današnjim standardima, to nije bilo preterano sofisticirano, ali je bilo zaista neverovatno, jer se pokazalo da je moguća komunikacija izmedu Zemlje i objekata u svemiru.

Pošto se satelit kretao na nebu, komunikacija je bila moguća samo u kratkom vremenskom periodu. Kada je zalazio iza horizonta, komunikacija se prekidala sve do trenutka dok se satelit. ponovo ne pojavi iznad horizonta. Takva situacija bi danas bila neprihvatljiva u brojnim aplikacijama (mada ne svim). Zamislite kablovsku televiziju, ili telefonski razgovor koji se prekida svaki put kada se satelit spusti ispod horizonta (to bi, ipak, bilo zgodno ako bi moglo da se sinhronizuje sa trenucima u kojima počinju da se puštaju reklame)! Sateliti koji zadržavaju fiksnu poziciju omogućavaju kontinuelni prenos, što je defmitivno značajan kriterijum za primenu za masovne medije. Pitanje je kako satelit može da zadrži fiksnu poziciju, a da ne padne.

GEOSTACIONARNI SATELITI Orbite satelita se predviđaju korišćenjem matematičkog modela zasnovanog na Keplerovim zakonima za kretanje planeta. Ideja je sasvim jednostavna. Na određenoj visini, da bi se zadržao u orbiti, objekat mora da ima odredenu brzinu. Veće brzine bi poslale objekte van zemljine orbite u svemir. Manje brzine mogu da budu nedovoljne za savladavanje gravitacije i u tom slučaju objekat pada. Drugim rečima, na odredenoj visini orbitalna brzina je utvrđena. Treći Keplerov zakon povezuje vreme za obilazak planete sa visinom u orbiti.

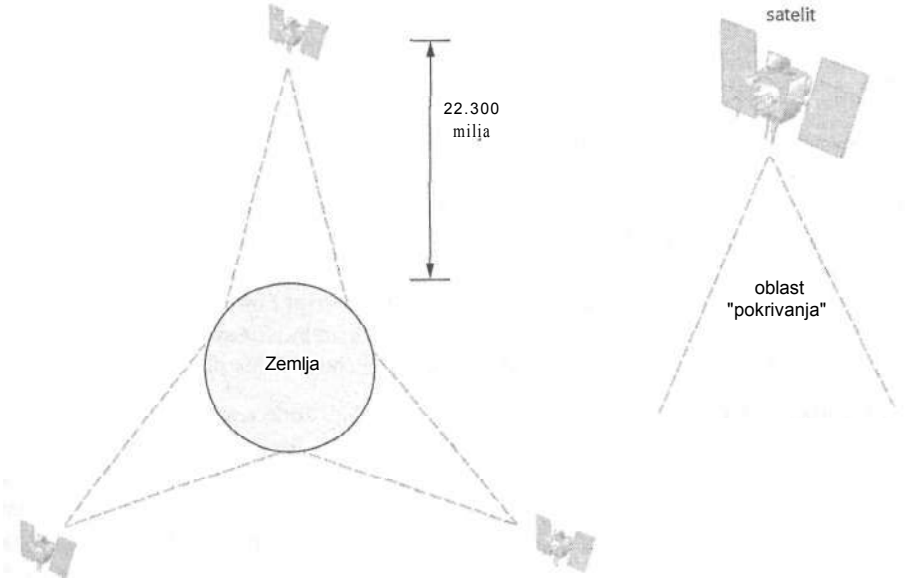
Pretpostavimo da je period P vreme za koje satelit obide oko planete. Treći Keplerov zakon kaže da je P2=KD3, gde je D rastojanje između satelita i centra planete, a K je konstanta koja zavisi od gravitacije. Daigim rečima, više orbite (veće D) znače duži period.

P2=KD3

Ovde se nameće logično pitanje koju orbitalnu visinu ima satelit sa periodom od 24 časa. Prema Keplerovom zakonu, odgovor je 22.300 milja iznad ekvatora, što je znatno veća visina od one na kojoj je leteo *"Sputnik"*. Ovo ima veliki značaj. Pošto se Zemlja za 24 časa rotira oko svoje ose, satelit koji se nalazi u orbiti na toj visini za posmatrača sa Zemlje deluje stacionamo. Ovo se naziva **geostacionarna (geosinhrona) orbita.** Ako je posmatrač predajnik, ili prijemnik, satelit bi u odnosu na posmatrača ostao na fiksnoj poziciji i ne bi dolazilo do prekida komunikacija.

"Sputnik" i mnogi drugi sateliti su leteli na mnogo nižim orbitama. Tehnologija jednostavno nije obezbedivala dovoljno močan raketni pogon koji bi ih podigao više u orbiti. Zato su rotirali oko Zemlje za manje od 24 časa i zato su posmatrači na Zemlji mogli da uoče njihovo kretanje na nebu. Danas komunikacione satelite odlikuje moćan raketni pogon koji ih šalje na veće visine u geostacioname orbite. Tri ravnomerno rasporedena satelita na 22.300 milja iznad ekvatora mogu da "pokriju" skoro celu površinu Zemljine kugle (slika 2.16), osim polarnih oblasti.

Satelitske komunikacije su jednostavne. Svaki satelit ima nekoliko **transpondera,** uredaja koji prihvataju signal sa frekvencijom iz odredenog opsega i reemituju ga sa drugom frekvencijom.



SLIKA 2.16 *Sateliti u geostacionarnoj orbiti iznad ekvatora*

Tabela 2.3: Satelitski frekventni opsezi

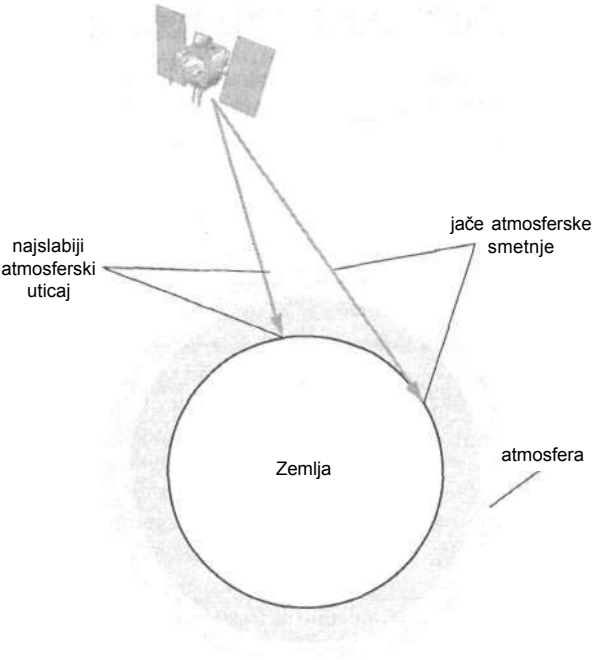
Opseg	Frekventni opseg uzlaznog linka (CHz)	Frekventni opseg silaznog linka (CHz)
L	1.6465-1.66	1.545-1.5585
C	5.925-6.425	3.7-4.2
Ku	14.0-14.5	11.7-12.2
Ka	27.5-30.5	17.7-21.7

Predajnik na Zemlji šalje signal **(uplink)** na satelit, gde jedan od transpondera prenosi signal nazad do Zemlje **(downlink)** na dmgoj lokaciji (slika 2.15). Satelitske komunikacije se sada koriste za prenos telefonskih i televizijskih signala. Mnogi ljudi imaju sopstvene prijemnike za televizijske signale i sve su popularnije 18-inčne satelitske antene, koje se postavljaju na krovove, ili u dvorišta.

Sateliti šalju i primaju satelite u širokom frekventnom opsegu, pri čemu je svaki transponder zadužen za signale iz specifičnog opsega. Obično se frekvencije uzlaznog i silaznog linka razlikuju, tako da ne ometaju jedne druge. U tabeli 2.3 dati su uobičajeni opsezi koji se koriste za komercijalne satelitske komunikacije, zajedno sa frekvencijama za uzlazni i silazni link. Ostali opsezi su rezervisani isključivo za vojsku i vladine potrebe.

Značajna činjenica koju treba zapamtiti u vezi satelitskog prenosa je to da se razli&ti signali iz istog frekventnog opsega ne smeju preklapati. Zato je ograničen broj satelita koji koriste isti frekventni opseg i medusobno rastojanje izmedu satelita (o tome će biti reči nešto kasnije). Najpre je korišćen C opseg; danas se obično koristi za televizijske prenose i VSAT aplikacije (i o njima će kasnije biti nešto više reči). Već je dosta "zatrpan" zbog sve većeg broja medijskih aplikacija i delom zbog toga što se njegove frekvencije koriste i u zemaljskim mikrotalasnim prenosima.

Kako se u svetu sve više koriste digitalne komunikacije, došlo je do povećanja broja digitalnih sistema i prelaska sa emitovanja tradicionalnih televezijskih signala na televizijske signale visoke defmicije (HDTV high-defmition television), šlo je postavilo veće zahteve sa aspekta količine informacija koju je neophodno preneti. Kao što je prikazano u odeljku 3.4, više frekvencije (posebno one iz Ka opsega) podložne su atmosferskim uticajima, kao što su kiša, ili velika vlažnost vazduha. Problem postaje još veći kada je prijemna stanica daleko (u odnosu na celokupnu Zemljinu površinu) i kada signal mora duže da putuje kroz atmosferu (slika 2.17). Ovaj problem može da se reši pojačavanjem snage signala, ili dizajniranjem sofisticiranijih prijernnika za filtriranje šuma. U stvari, jači signali iz Ku opsega dopuštaju korišćenje manjih antena, ili tanjira. Mnogi od vas su već videli 18-inčne tanjire za prijem digitalnih televizijskih signala, koji mogu da se kupe u većini radnji za prodaju elektronske opreme. Sećamo se i vremena kada su korišćene velike satelitske antene koje su zauzimale cela dvorišta. Ku signali omogućavaju uži snop signala, tako da su mogući prenosi na manjim geografskim oblastima.



SLIKA 2.17 *Atmosferske smetnje u funkciji ugla prenosa*

Kako se signali pomeraju ka Ka opsegu, problem interference postaje ozbiljniji. Sateliti koji emituju signale u Ka opsegu su i dalje nova tehnologija, mada već imaju visoki prioritet u polidji i programima vlada SAD-a, evropskih zemalja i Japana. Postoje ambiciozni planovi za postavljanje globalne infrastrukture satelita u orbiti koji komuniciraju u Ka opsegu; predstavićemo ih ukratko.

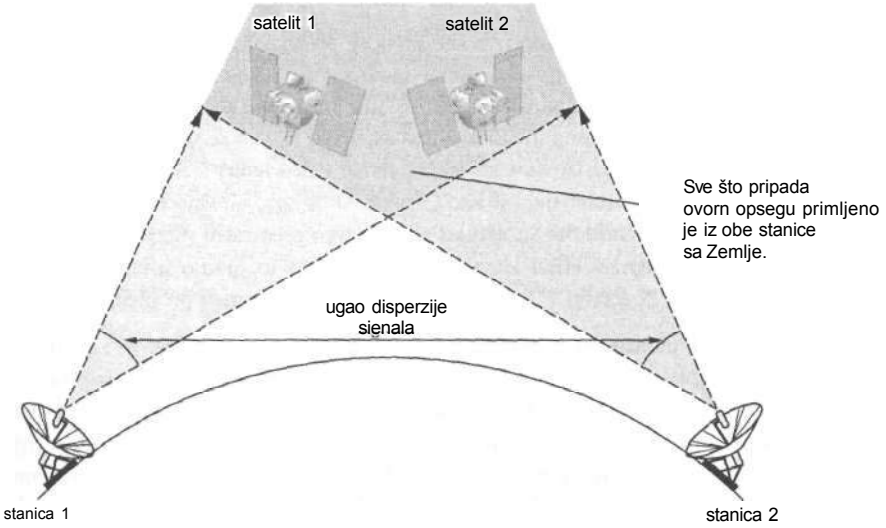
U nižem frekventnom spektru, L opseg se koristi prvenstveno za mobilne satelitske komunikacije. To znači da se komunikacije izmedu automobila, kamiona, brodova, ili bilo kojih drugih pokretnih sredstava oslanjaju na satelitski prenos. Niže frekvencije omogućavaju korišćenje manje, jeftinije opreme, sa manjom snagom. Koriste se u transportu, gde dispečer može da locira kamion sa tačnošću od nekoliko desetina metara (na nivou kontinenta). Mogu da se koriste i u navigacione svrhe: u avionima, na brodovima i u automobilima ovaj sistem može da se koristi za iscrtavanje tekuće lokacije.

Geostacionarni sateliti prenose signal koji se može primiti bilo gde na zemaljskoj kugli, dok god postoji direktna linija videnja. Ovakvi prenosi se koriste za emitovanje televizijskih sadržaja i za filmske servise kod kojih se naplaćivanje vrši po prikazivanju, za koje postoji veliki broj prijemnika i koji ne moraju da imaju pristup kablovskim signalima. Koriste se i za vojne svrhe, mada je tada neophodno ograničiti geografsku oblast u kojoj je moguće primati signale. Specijalne antene koje omogućavaju uobličavanje snopa signala mogu da se koncentrišu na manje oblasti, kao što su gradovi. U budućnosti se očekuje razvoj antena koje omogućavaju usmeravanje snopa signala samo na jedno mesto.

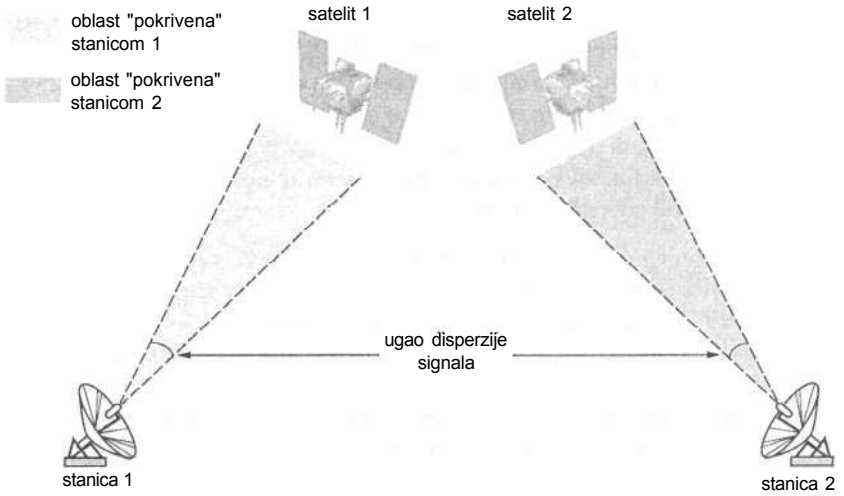
Trenutno postoji nekoliko stotina satelita koji kruže u Zemljinoj orbiti, obezbeđujući komunikacije u različite svrhe korišćenjem signala u C i Ku opsezima. Nameće se pitanje kako satelit izdvaja signale koji su namenjeni za njega? Na primer, pretpostavimo da stanice 1 i 2 šalju signale na satelite 1 i 2, respektivno, koristeći istu frekvenciju (slika 2.18). Oblast "pokrivena" signalom zavisi od ugla disperzije signala. Ako su dva satelita postavljena isuviše blizu, ili ako je ugao disperzije isuviše veliki, oba satelita primaju signale iz obe stanice. Zbog toga, ni jedan satelit ne može da utvrdi koje signale treba da ignoriše.

Ako su uglovi disperzije manji i ako su sateliti dovoljno udaljeni, ni jedan satelit neće primati signale iz tude stanice (slika 2.19). FCC definiše pozicije američkih satelita, sa inkrementom od 2° između *67*° i 143° zapadne geografske dužine za komunikadje koje koriste C opseg. Ovo je bliže nego ranije, kada su korišćena 4°. Sateliti koji prenose signale u Ku opsegu mogu da se postave na manjim udaljenostima, sa inkrementom od 1 °. Pošto postoji sve veći broj zahteva za postavljanje satelita, mora se obezbediti postavljanje na manjim rastojanjima. Zato stanice na Zemlji moraju da imaju manje uglove disperzije prenetih signala.

Satelitske komunikacije su prouzrokovale i odredene probleme. Na primer, kako da se spreči neuatorizovani prijem signala? Kako se definiše neautorizovani prijem signala koji putuje kroz javni vazdušni prostor? Zakonske odredbe nisu uvek "ili crne, ili bele". Na primer, postoje različita gledišta u vezi toga da li je legalno pomoću satelitskih antena primati kablovske televizijske kanale koji se plaćaju, kao što je HBO.



SLIKA 2.18 *Sateliti primaju signale iz više stanica*



SLIKA 2.19 *Sateliti primaju samo sienale iz jedne stanice*

Kompanije za kablovsku televiziju tvrde da su na gubitku zbog ovakvog pristupa. Vlasnici satelitskih antena tvrde da signali putuju kroz javni vazdušni prostor i da zbog toga može svako da ih koristi, kao i sve ostale televizijske signale.

Hi, što je još gore, kako spredti neautorizovane prenose preko satelita? Bilo je slučajeva upada kada su pojedinci slali bezazlene poruke preko satelita. Ali, šta je sa upadima koji ometaju komunikacije? Zamislite koliko se danas "stvari" oslanja na satelitske komunikacije i shvatićete koliko je problem ozbiljan. U mnogim slučajevima satelitski signali se skrembluju, ili šifruju da budu neprepoznatljivi za neautorizovane korisnike. Jedan video haker je 27. aprila 1986. godine izazvao incident, kada je, predstavljajuči se kao Captain Midnight, ometao program HBO kanala puštanjem filma "The Falcon and the Snowman". To je izveo prenosom signala preko satelitskog linka koji je bio jači od signala HBO kanala. Tvrdio je da je to uradio u znak protesta zbog skremblovanja signala. U Poglavlju 7 detaljnije su prikazani šifrovanje i problemi zaštite.

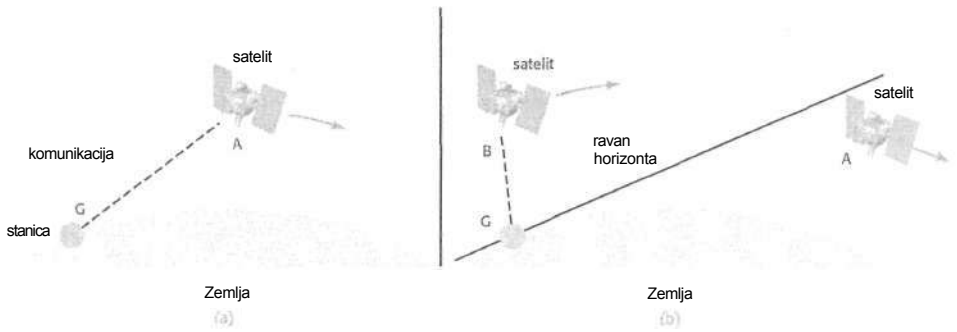
Ne služe svi sateliti za prenos telefonskih signala, za vremenske prognoze, ili za vojne svrhe. Mnoge privatne kompanije vide satelite kao alternativu telefonskom sistemu, posebno ako postoji potreba za prenosom podataka na većim rastojanjima. Korišćenje telefonskog sistema i velikog broja komutatora umanjuje pouzdanost prenosa podataka, a u slučaju potrebe za prenosom veće količine podataka može doći do produžavanja vremena odziva. Rešenje može da bude very **small aperture terminal** (VSAT) sistem, koji je razvijen 80-ih godina prošlog veka. Taj sistem obično povezuje centralnu lokaciju sa više udaljenih lokacija. Na primer, na centralnoj lokaciji može da se nalazi velika baza podataka, kojoj pristupa više korisnika iz udaljenih predstavništava. Komunikacija između dve lokacije se odvija preko satelita i zahteva korišćenje manjih antena koje mogu biti postavljene radi obezbedivanja pristupa centralnoj lokaciji.

VSAT oprema može da se poveže direktno na korisničku opremu, kao što su radne stanice, ili kontroleri. Mnoge aplikacije se oslanjaju na VSAT sisteme, posebno one koje zahtevaju velike brzine prenosa podataka za kratko vreme. Primeri uključuju National Weather Service (američki servis za vremenske prognoze), servise za prenos vesti, verifikaciju kreditnih kartica, sisteme za automatsko obaveštavanje i rent-a-car agencije.

Sateliti koji se nalaze nisko u Zemljinoj orbiti Geostacionami sateliti su veoma korisni za emitovanje različitih sadržaja. Antene mogu da se usmere ka fiksnoj tački na nebu, tako da šalju, ili primaju signale po potrebi. Sateliti koji se nalaze nisko u Zemljinoj orbiti - LEO (low **earth orbit)** sateliti - nude neke prednosti koje geostacionami sateliti ne mogu da ponude. Za vojna osmatranja neophodni su sateliti koji se ne zadržavaju na fiksnoj poziciji. Niža orbita omogućava satelitima da se kreću u odnosu na površinu Zemljine kugle i da skeniraju različite oblasti.

LEO sateliti zahtevaju slabiji pogon i, u sporazumu sa NASA-om, ili vodećim kompanijama za svemirske letove i lansirnim sistemima, kao što je Lockheed Martin, mogu da pošalju satelit u orbitu u spejs šatlu, ili nekoj drugoj letelici, kao što je raketa. Uređaji koji komuniciraju sa LEO satelitima zahtevaju manji pogon, jer je redukovano rastojanje koje signal treba da prede. LEO sateliti nisu mnogo korišćeni za globalne komunikacije, jer se ne nalaze uvek u dometu predajnika i prijemnika na Zemlji. Ipak, i to se menja!

Teorijski, LEO sateliti mogu da se koriste za komunikacije ako postoji dovoljan broj satelita u orbiti. Na slici 2.20 možete da vidite kako je to moguće. Ako se na nižim visinama u orbiti postavi dovoljan broj satelita, svi mogu da se kreću relativno u odnosu na Zemlju. Na primer, na slici 20.2a stanica je uspostavila komunikaciju sa satelitom. Pošto se satelit A nalazi nisko u orbiti, kreće se u odnosu na stanicu. Zbog toga, eventualno može da se spusti ispod horizonta, tako da direktna komunikacija sa stanicom bude nemoguća sve dok on ne obiđe Zemlju i ponovo se ne podigne iznad horizonta. Medutim, umesto da se čeka na satelit A, koristi se drugi satelit (B), koji se, takode, nalazi nisko u orbiti. Dva satelita su pozicionirana tako da, kada A pada ispod horizonta (slika 2.20b), B se podiže iznad horizonta.

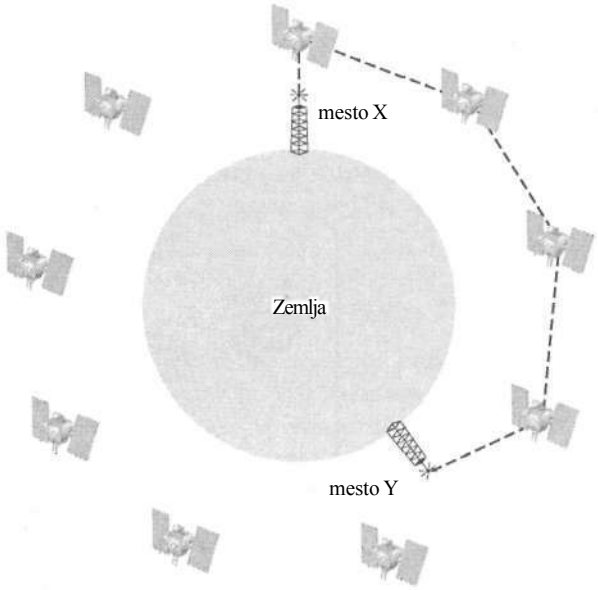


SLIKA 2.20 *Stimica komunicira sa LEO satelitima*

On može da preuzme komunikaciju iz stanice za koju je do tada bio zadužen satelit A. Kada B eventualno side ispod horizonta, postojaće sledeći satelit koji može da preuzme njegovo mesto. Ako postoji dovoljan broj LEO satelita, uvek će postojati satelit koji može da preuzme komunikaciju sa stanicom na Zemlji i sve tačke na našoj planeti, čak i one najizolovanije, naći će se u dometu LEO satelita. Ovo je slično principu na kome se zasniva mobilna telefonija. Vaš mobilni telefon uvek kontaktira sa istim predajnikom, sve dok se ne udaljite toliko daleko da budete u dometu drugog predajnika. Kod LEO satelita je situacija obrnuta - kreću se sateliti, a ne korisnik.

Posledica ovoga je da se bilo koje dve lokacije na Zemlji mogu povezati kao na slici 2.21. Jedna stanica (na mestu X) komunicira sa najbližim LEO satelitom koji u orbiti izvršava protokol, omogućavajud razmenu informacija. Protokol omogućava da se poruka iz mesta X prenese do najbližeg LEO satelita. LEO satelit u orbiti izvršava protokol koji dopušta razmenu informacija. Dakle, protokoli omogućavaju da LEO satelit iznad mesta X proverava koji je LEO satelit najbliži mestu Y, prenosi mu poruku, a zatim satelit iznad mesta Y prenosi poruku do Zemlje.

U vreme kada je ova knjiga pisana nije postojala globalna funkcionalna mreža sa LEO satelitima; bilo je u planu kreiranje jedne takve mreže. Inženjeri u Motorolinom odeljenju za satelitske komunikacije zamislili su 1987. godine globalnu mrežu sa 77 LEO satelita. Mreža je dobila naziv *Iridium,* po 77. elementu periodnog sistema elemenata. To je prvi projekat u kome je predvideno korišćenje LEO satelita. Signali koji bi se koristili za komunikaciju sa zemaljskim stanicama pripadali bi L opsegu, mada se za prenos signala izmedu satelita koristi Ka opseg.



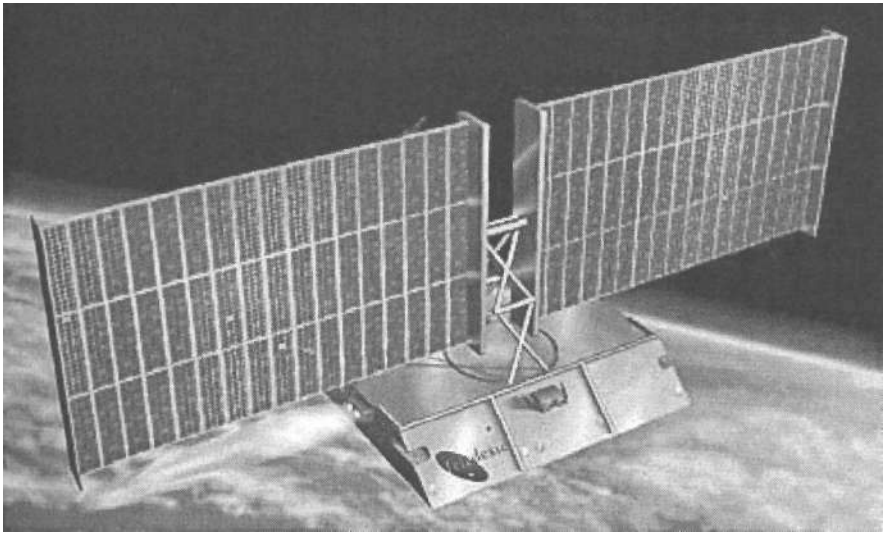
SLIKA 2.21 *Dve proizvoljne stanice komuniciraju preko LEO satelita*

Atmosferski uslovi koji ometaju prenos signala izmedu satelita i zemaljskih stanica predstavljaju manji problem iznad atmosfere. Suočen sa ozbiljnim fmansijskim poteškoćama, Iridium konzordjum je u leto 2000. godine zvanično obustavio realizadju tog projekta.

Sledeći akteru ovoj "priči" je Teledesic Corporation. Ta korporacija je osnovana 1990. godine, a glavni investitori su pionir na polju telekomunikacija Craig McCaW, predsednik i CEO Microsofta William Gates, saudijski princ Alwaleed Bin Talal, Abu Dhabi Investment Company i Boeing. Planirano je da Teledesic mreža sadrži 228 medusobno povezanih LEO satelita. Ranija konfiguracija je imala po 24 satelita u 12 grupa (slika 2.22), gde svaka grupa kruži oko planete u polarnoj orbiti (slika 2.23), na visini aproksimativno od 1.400 kilometara (oko 875 milja i 1/25 visine na kojoj se postavljaju geostacionarni sateliti). Dok se sateliti kreću od severa ka jugu (a zatim od juga ka severu), Zemlja se rotira ispod polarnih orbita. Kolekcija satelita je dizajnirana tako da "pokriva" 95 odsto Zemljine kugle i može da podrži zahteve nekoliko miliona korisnika istovreraeno. U novijim konfiguracijama je redukovan broj satelita na 30; detaljnije informacije nisu bile dostupne u vreme kada je ova knjiga pripremana.

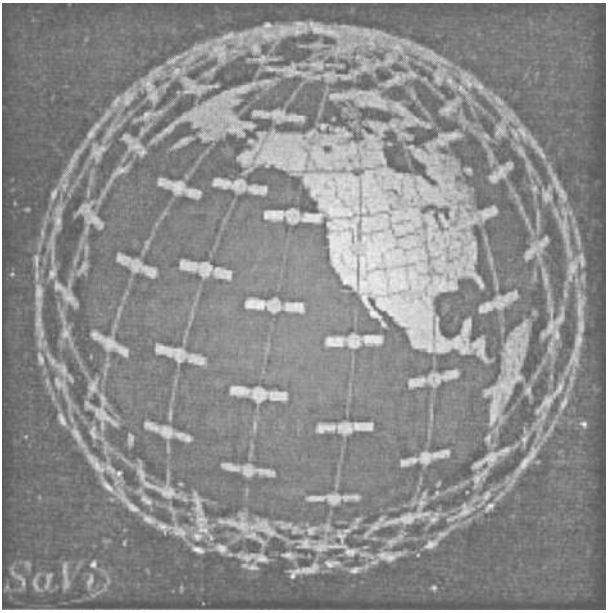
Signali silaznog linka se nalaze u Ka opsegu i funkcionišu izmedu 18,8 i 19,3 GHz; signali silaznog linka se kreću izmedu 28,6 i 29,1 GHz. Bitske brzine se kreću od 100 Mbps za uzlazni, do 720 Mbps za silazni link. Susedni sateliti mogu medusobno da komuniciraju, a zajedno formiraju istinsku globalnu komunikacionu mrežu (slika 2.24).

Prvi probni satelit je lansiran 27. februara 1998. godine, a planirano je da Teledesic postane svetski Internet provajder početkom 2005. godine.

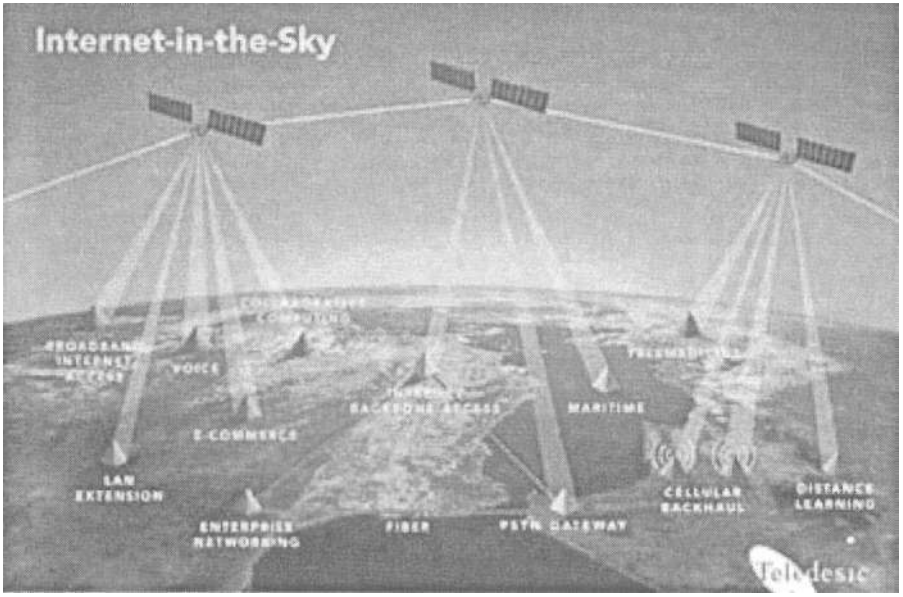


SLIKA 2.22 *Teledesic satelit*

U stvari, Teledesic je dobio licencu od FCC-a još 1997. godine, a ugovor o lansiranju je sklopljen sa Lockheed Martinom dve godine kasnije.



SLIKA 2.23 *Konstelacija Teledesic satelita*



SLIKA 2.24 *Povezivanje tačaka na zemlji*

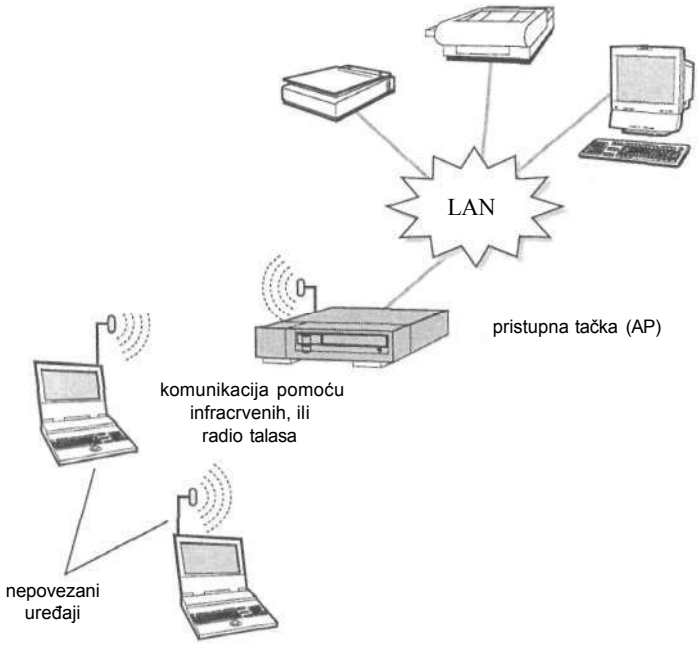
Pre dve godine je potpisan ugovor sa italijanskim proizvodačem satelitske opreme Alenia Spazio Spa za proizvodnju dva satelita. Globalna mreža treba da obezbedi servise onima koji ih pružaju korisnicima, umesto da se obezbeduju direktno za korisnika. Ovo bi uključilo umrežavanje kompjutera, visokokvalitetni prenos glasa, interaktivnu multimediju, širokopojasne (broadband) Internet servise i mnoge druge aplikacije. Ostaje još mnogo štošta da se uradi u ovoj oblasti, posebno u oblasti integracije Interneta sa LEO satelitima i definisanja odgovarajućih algoritama za rutiranje (ref. [Ek02]).

Bežični LAN

Jedna od najinteresantnijih aplikacija bežičnih komunikacija je bežični LAN - sistem koji omogućava komunikaciju personalnih kompjutera i ostalih tipičnih mrežnih uredaja bez fizičke konekcije. "Obećava" potpuno novi svet aplikacija kod kojih kabliranje nije praktično. Ako su ljudi često u pokretu, potrebni su im i mobilni sistemi komunikacija, a ne sistemi koji su ograničeni svojom infrastrukturom. Na primer, medicinsko osoblje može da koristi notebook kompjutere, ili personalne digitalne asistente povezane na bežični LAN radi pristupa medicinskim kartonima pacijenata koje posećuju kod njihovih kuća. Direktori mogu da koriste bežični LAN radi pristupa informacijama za vreme značajnih sastanaka. Zaposleni mogu na putu do posla, ili od posla do svojih kuća da pristupaju Internetu radi preuzimanja značajnog fajla koje su zaboravili pre polaska kući, ili da bi se pripremili za značajan sastanak. Bežični kompjuteri mogu da se koriste i u laboratorijama gde bi kablovi ometali kretanje kroz prostoriju. Kompjuteri pomoću bežičnih konekcija mogu da pristupaju podacima koji su smešteni na serveru, ili mogu da štampaju na "povezanom" štampaču. Bežične konekcije omogućavaju lakše pomeranje opreme, bez potrebe da se vodi računa o isključivanju, ili prekidanju kablova.

Za bežične LAN mreže koriste se dve tehnologije: infracrveni i radio talasi. **Infracrveni talasi** su elektromagnetni talasi čije su frekvencije odmah ispod frekvencija vidljive svetlosti. llređaji su opremljeni LED, ili laserskim diodama, koje emituju infracrvenu svetlost. Ovi talasi mogu da se usmere ili direktno ka prijemniku, ili da se odbijaju od zidova i plafona. Princip je sličan onome na kome funkcioniše daljinski upravljačza promenu televizijskih kanala. Sistemi sa infracrvenim talasima imaju neke prednosti. Na primer, za njih nisu neophodne regulative koje donosi FCC, kao što je slučaj kod radio talasa. Zbog toga, nema licenciranja za korišćenje opreme koja se zasniva na infracrvenim talasima. Sledeća prednost je što infracrveni talasi ne prodiru kroz čvrste objekte, kao što su zidovi. Tako su zaštićeniji od prisluškivanja. Osim toga, omogućeno je postavljanje uredaja u bezbednim prostorima zgrade kada može da se koristi isti infracrveni signal bez interferenci. Inače, infracrvene talase ne mogu da ometaju radio talasi. Ipak, nemogućnost prodiranja kroz čvrste objekte predstavlja i nedoslatak ako je neophodno uspostaviti komunikaciju izmedu zgrada. Sledeći nedostatak je to što infracrveni signali u principu omogućavaju niže bitske brzine u poredenju sa ostalim tehnologijama. Više informacija o infracrvenim talasima možete da pročitate u referenci [We98].

Bežični LAN-ovi mogu da koriste i radio prenose do 2,4 GHz. Na slici 2.25 data je tipična konfiguracija. Uobičajena mreža sa kabliranjem, kao što je Ethernet mreža, formira osnovnu infrastrukturu za povezivanje različitih mrežnih uredaja, kao što su kompjuteri, štampači i skeneri. Medutim, na mrežu je povezana druga komponenta koja se naziva pristupna tačka (AP - access **point);** ona se ponaša kao "most" između bežične mreže i mreže sa kabliranjem.



SLIKA 2.25 *Konfiguracija bežičnog LAN-a*

Pristupna tačka prima informacije iz mreže sa kabliranjem i emituje ih ka ostalim uređajima koji se nalaze u njenom dometu.

Postoje različiti tipovi AP-a, a najvažniji proizvodači su 3Com, Cisco, Nokia, Lucent i drugi (ref. AnOO); razlikuju se po različitim faktorima, kao što su opseg prenosa signala, broj korisnika koji može istovremeno da se podrži, brzina interfejsa i mogućnost kontrole. Razlikuju se **i** po tipu radio signala koje prenose. IEEE je uspostavio standard za bežične mreže, pod oznakom 802.11 (http://grouper.ieee.org/groups/802/11/index.html) i, u nekim slučajevima, koristi se naziv **Wi-Fi** (od **wireless fidelity).** Koriste se različiti protokoli za obezbeđivanje dolaska signala na odgovarajuće uredaje; predstavićemo ih, zajedno sa ostalim aspektima bežičnih LAN mreža, u okviru Poglavlja 9.

Bluetooth

U većem delu prethodnog razmatranja pretpostavka je bila da se koriste tipični mrežni uredaji, kao što su personalni kompjuteri, štampači i skeneri. Medutim, razvija se tehnologija koja bi zauvek mogla da promeni naše videnje mrežnih uređaja. Malo ko bi pomislio da bi ručni sat, tajmer za rernu, ili video rekorder mogli biti mrežni uređaji, mada se kreatori Bluetooth tehnologije nadaju da će to biti moguće. Tehnologija **Bluetooth** je dobila naziv po Haroldu Blatandu (Bluetooth), Vikingu koji je živeo u 10. veku i ujedinio Dansku i Norvešku.

Prikaz vikinške istorije i njene veze sa tekućom tehnologijom prelazi predviđeni obim ove knjige, ali ako želite da saznate zanimljivu perspektivu ove relacije, posetite www.bluetooth.com.

Bluetooth je koncept kod koga se mikročip sa radio predajnikom ugraduje u elektronski uredaj. Naravno, namera je da se omogući komunikacija tih uređaja bez kablova. Osim toga, na taj način je omogućena komunikacija i uredaja izmedu kojih ne postoji direktna linija videnja, koja je bila neophodna u starijim konfiguracijama bežičnih mreža. Ericsson Mobile Communication započinje rad na konceptu Bluetooth 1994. godine, kada su istraživanja omogućila stvaranje jeftinih interfejsa male snage između mobilnih telefona i njihovih pratećih uredaja. IBM, Nokia, Intel i Toshiba 1998. godine formiraju specijalnu interesnu grupu; danas je ona narasla do alijanse od skoro 2.000 kompanija. Iako predstavlja izazov, postoje i sumnjičavi stavovi u vezi Bluetootha. Neki smatraju da je povezivanje na različite korisničke uredaje različitih proizvođača osuđeno na propast. Drugi tvrde da brzine prenosa podataka od 1 Mbps na rastojanjima do IO metara neće omogudti praktičnu realizaciju ove tehnologije u 21. veku.

Koje su moguće primene ove tehnologije? U referenci [DoOO] Dorman kaže: "Zamislite budućnost u kojoj konzerva piva komunicira sa frižiderom, koji može da uspostavi vezu sa Vašim aičnim satom i da Vam saopšti kada je pivo dovoljno hladno da može da se pije. Kada bacite dubre u kantu, kanta opremljena Bluetooth tehnologijom može da kaže službi za odvoz smeća kako da reciklira otpatke. U meduvremenu, frižider može da naruči još piva iz online prodavnice i da naloži Vašem automobilu da se ne pali dok se ne otreznite." Iako u šali, autor ipak ističe da ćemo u budućnosti verovatno vidati uređaje, koji se, inače, ne smatraju tipičnim mrežnim uredajima, kako medusobno komuniciraju. Zbog pojave sve većeg broja novih tehnologija, počinje da se koristi novi termin PAN mreže **(personal area networks).** Ideja je da se obezbedi komunikacija izmedu uredaja koji se smatraju normalnim potrošačkim uredajima. Dan kada ćete paliti svetla u kući pomoću ručnog sata, ili uključivati automat za kafu kada krenete ka kući više nije previše daleka budućnost.

Tehnologija Free Space Optics *s*

Poslednja tehnologija za bežično umrežavanje koju ovde obradujemo je firee **space optics (FSO).** U osnovi, podaci se prenose korišćenjem optičke tehnologije, ali bez fibera. Namenjena je onima koji žele da koriste bitske brzine koje omogućava optička tehnologija, a da, pri tom, izbegnu troškove koji prate instaliranje optičkih fibera. Na primer, prema izveštaju USA Today iz Sijetla, Vašington, u aprilu 2002. godine, "kada je kompanija LifeSpan BioSciences* premestila svoj centar podataka u novu zgradu udaljenu samo tri bloka od stare, njeni čelnici su shvatili da će morati da zatvore svoje istraživačke laboratorije na nekoliko meseci", jer je za povezivanje laboratorija sa kompjuterima u novoj zgradi bilo neophodno postaviti optičke fibere ispod uličruh kolovoza. Umesto da troše vreme i novac, izabrali su drugu opciju - instalirali su laserski predajnik na krovu jedne zgrade i poslali laserski mlaz do prijemnika na krovu druge zgrade.

* Kompanija koja vrši istraživanja u oblasti molekularne biologije

Laser je prenosio signale u opsegu teraherca (10" Hz), što, za razliku od satelitskog i mikrotalasnog prenosa, nije regulisano federalnim zakonom koji propisuje emitovanje signala u opsegu ispod 600 GHz. Naravno, to bi se moglo promeniti u budućnosti ako FSO preuzme primat i iskoristi veći deo spektra.

LifeSpan BioSciences nije jedina kompanija koja koristi FSO tehnologiju. Nakon 11. septembra 2001. godine, kada je izvršen napad na Svetski trgovinski centar u Njujorku, *USA Today* izveštava da je "Merrill Lynch koristio FSO opremu Terabeama iz Sijetla za ponovno uspostavljanje veze iz svoje kancelarije na donjem Menhetnu sa centrima u Njudžerziju i srednjem Menhetnu. Advokatska firma Mayer Brown & Pratt koristila je opremu Light-Point Communications iz San Dijega za otvaranje 400 novih telefonskih linija za klijente koji su izmešteni zbog napada."

Pristalice FSO tehnologije tvrde da Je moguća mnoga niža cena za povezivanje na Internet u odnosu na situaciju u kojoj se koristi optički fiber - postavljanje preko postojećeg optičkog fibera, ili postavljanje novih fibera ne bi bilo neophodno. Osim toga, tvrde da se sistemi mogu inslalirati za nekoliko dana, ili nedelja. Sledeća prednost je bezbednost. Iako bežične tehnologije odlikuje lošija bezbednost, FSO ne potvrduje to pravilo. Za razliku od satelitskih i mikrotalasnih prenosa, koje je lako presresti, FSO mlazevi su uskofokusirani i teško se detektuju. Osim toga, ako se signal presretne, autorizovani primalac će to registrovati, jer dolazi do prekida u prijemu signala. Bitske brzine se porede sa onima koje se postižu kod optičkog fibera.

Ovo ne znači da FSO nema ograničenja. To je tehnologija koja zavisi od direktne linije videnja i zato se moraju izbegavati sve prepreke na putu. To može da bude problem, jer zgrade, posebno one visoke, utiču na prenos i ometaju uskofokusirani laser, tako da on promaši svoju metu ako se ne koriste sofisticirani uređaji za automatsko praćenje, radi usmeravanja smera prenosa. Osim toga, kada laserski zrak prede nekoliko kilometara, mlaz počinje da se širi i na prijemnoj strani ne može ispravno da se interpretira. Neki tvrde da je prenos signala pouzdan na udaljenostima do 2,5 kilometara, mada brojna testiranja pokazuju da je optimalno rastojanje manje od jednog kilometra. Drugi problemi nastaju zbog klimatskih uslova. Ako je grad smešten u oblasti koja je podložna magli, optimalno rastojanje iznosi svega par stotina metara, jer kapljice vazduha rasipaju svetlost iz lasera. Konačno, tu su i socijalni problemi. Većina je gledala brojne filmove u kojima se laseri koriste kao oružje. Mogućnost da ptica naleti na mlaz i padne "pečena" na zemlju nije nimalo prijatna. Proizvodači tvrde da tako nešto nije moguće, jer laseri koriste jačinu koju je odobrio FDA. Mnogi smatraju da je FSO glavna tehnologija budućnosti. Više informacija o ovoj temi možete da pročitate u referenci [AlOl].

Zaključak

U ovom odeljku smo opisali različite medijume za prenos signala. Možda ste se zapitali koji je najbolji. Odgovor nije jednostavan. Kada je reč o razmeni podataka, zanimljivo je da nove tehnologije ne istiskuju stare. Na primer, sa razvojem fiber optičke tehnologije i satelitskih komunikacija nisu napuštene mreže koje koriste kablove sa upredenim paricama, ili koaksijalne kablove. Svaki medijum ima svoje mesto u svetu komunikacija. U tabeli 2.4 možete da vidite poredenje ovde predstavljenih medijuma.

Tabela 2.4: Poredenje medijuma za prenos

	Upredene parice	Koaksijalni kabl	Optički Fiber	Mikrotalasi	Sateliti	Infracrveni talasi	FSO
Bitska brzina	Zavisi od kategorije vodova. Stariji Cat 3 kablovi dugo su podržavali 10 Mbps Ethemet brzine, ali noviji Cat 5 kablovi podržavaju gigabitski Ethemet	Može da se uporedi sa Cat 5 UTP kablovima za LAN okruženja.	Bitske brzine od nekoliko stotina gigabita u sekundi na rastojanjima od više kilometara.	Zavisi od frekvencije signala. Brzine se otprilike kreću izmedu 10 i 300 Mbps.	Kao i kod mikrotalasa, brzina zavisi od frekvencije (10-300 Mbps). * Zbog sve većeg korišćenja Ka opsega, očekuje se povećanje brzine.	Do 10 Mbps, ali ne mogu da prolaze kroz čvrste objekte kao što su zidovi.	Postiže se brzina od gigabita u sekundi sa rastojanjima do 1 kilometra.
Osetljivost na smetnje	Električne smetnje od susednih kablova, ili motora. Upredanjem žica redukuju se neki šumovi koji zavise od frekvencije signala koji se prenosi kroz kabl.	Omotavanje u većoj meri eliminiše električne smetnje.	Imuni su na elelektrične smetnje.	Čvrsti objekti izazivaju smetnje. Potrebna je direktna linija videnja izmedu krajnjih tačaka.	Smetnje nastaju zbog atmosferskih uslova. "Stvari" se pogoršavaju na višim frekvencijama.	Zbog nemogućnosti prolaska kroz čvrste objekte, infracrvene komunikacije su ograničene na zatvorene prostore.	Potrebna je direktna linija viđenja, a vremenske prilike kao što je magla mogu da ometu prenos.
Rastojanje	Zavisi od debljine kabla i bitske brzine. Moguć je prenos signala na udaljenostima od 5-6 kilometara, bez repetitora, mada su kablovi sa upredenim paricama ograničeni na nekoliko stotina	Osim toga, zavisi od bitske brzine. Moguć je prenos 5-6 kilometara, bez repetitora.	Stotine kilometara.	20-30 milja, mada zavisi od visine antene i terena izmedu krajnjih tačaka.	Svetske razmere.	Prilično kratka rastojanja, od nekoliko desetina metara.	Aproksimativno 1 kilometar, mada se u uslovima magle rastojanje svodi na svega par stotina metara.

Tabela 2.4: Poređenje medijuma za prenos

	Upredene parice	Koaksijalni kabl	Optički fiber	Mikrotalasi	Sateliti	Infracrveni talasi	FSO
Tipične primene	Posebno su korisni u LAN okruženjima, gde se konekcije moraju izvesti na malom prostoru iza zidova, ispod poda, ili u kancelarijama gde povezuju radne stanice sa utičnicama na zidu.	Nekada je bio primami medijum za LAN, a sada je zamenjen optičkim fiber i UTP kablovima. 1 dalje se koristi za servise kablovske televizije.	Obično se koristi za telefonske dalekovode. Često se koristi i kao primami komunikacioni medijum ("kičma") u kompjuterskim mrežama.	Obično se koriste kada kablovi nisu praktični su, na primer, za povezivanje telefonskih mreža u retko naseljenim oblastima, ili za razmenu podataka između dva mesta u metropolama. Ponekad se koristi i za LAN konekcije.	Komunikacije svetskih razmera. Ti je uključena telefonija, vojne svrhe, vremenske prognoze i televizija.	Primenjuje se kod bežičnog LAlM-a, posebno kada je neophodno ograničiti oblast prijema zbog bezbednosnih razloga.	Koristi se za prenos na velikim brzinama podataka na kratkom rastojanju, gde je cena pristupa, ili postavljanja optičkih fiber kablova isuviše visoka.
Komentari	Mogu da se omotaju da bi bile omogućene veće brzine prenosa podataka na većim rastojanjima, mada je oblaganje često skuplje i teže za rukovanje.	Mogu i dalje da se koriste u LAN okruženjima za povezivanje uredaja koji se nalaze veoma blizu, tako da se omogud zaštita od električnih smetnji.	Teško se vezuju. Osim toga, otežano je dodavanje novih uredaja. lpak, optički fiber je izuzetno koristan kada se velika količina podataka mora preneti na velikim rastojanjima.	Nove gradevine između krajnjih tačaka mogu da stvore probleme.	Teško se sprečava neautorizovani prijem. Osim toga, nastaju kašnjenja uzrokovana prelaskom signala preko velikih udaljenosti.	Bežični LAN-ovi koji koriste iste frekvencije mogu da se postave u različitim prostorijama.	Ovo je relativno nova tehnologija, i neki mogu da budu uznemireni zbog prisustva lasera (i pored uveravanja da je prenos apso-lutno bezbedan). Kako tehnologija napreduje, možda će se prijem kod korisnika poboljšati.

Napomena: 1 Kbps = 210 bitova u sekundi; 1 Mbps = 220 bitova u sekundi = 1 milion bitova u sekundi; 1 Gbps = 230 bitova u sekundi = 1 bilion bitova u sekundi.

*Ova cifra se odnosi na transpondere. Pošto satelit može da sadrži i dve desetine transpondera, ukupna brzina prenosa podataka predstavlja sumu brzina prenosa za sve njih.

2.5 Kodovi

U prethodnom odeljku smo predstavili različite medijume za prenos podataka i naveli njihove karakteristike. Bez obzira da li se kao medijum koriste svetlost, elektricitet, ili mikrotalasi, nameće se pitanje kako se informacije kodiraju u format koji je prikladan za prenos. U ovom odeljku daćemo odgovor na to pitanje.

Pošto se razmena podataka najvećim delom odvija izmedu kompjutera i perifernih uredaja, polazimo od osnovnih tehnika za skladištenje podataka na kompjuteru. Kompjuteri su digitalni uredaji; funkcionišu zahvaljujući otvaranju i zatvaranju malih električnih prekidača koji su programirani unutar čipa. Ovo je pojednostavljeno objašnjenje, ali ne nameravamo da zalazimo dublje u raspravu o kompjuterima i CPU jedinici. Umesto toga, smatraćemo da svi prekidači, bez obzira na način na koji su implementirani, mogu da imaju jedno od dva stanja: otvoreno, ili zatvoreno. Simbolički, ta stanja možemo da tretiramo kao O i 1; mogu se izraziti bitovima, najmanjim jedinicama informacija koje je moguće smestiti u kompjuteru.

Sami po sebi, bitovi nisu preterano korisni, jer mogu da zabeleže samo po dva moguča dela informacija. Medutim, njihovim grupisanjem moguće je dobiti različite kombinacije nula i jedinica. Na primer, grupisanjem dva bita moguće je dobiti $2' = 4$ jedinstvene kombinacije. Grupa od tri bita omogućava $T = 8$ kombinacija. Formiraju se uzimajući prethodne grupe bitova i dodavanjem 0, ili 1 na kraj. U opštem slučaju, grupa od n bitova dopušta 2 kombinacija (može li to da se dokaže?). Ovakvo grupisanje bitova omogućava pridruživanje odredenih kombinacija specifičnim elementima, kao što su karakteri, ili brojke. Ovo pridruživanje nazivamo kod. Nema ničeg komplikovanog u vezi kodova. Svako može da ih defmiše. Trik je u tome da se napravi kod koji i drugi mogu da koriste. Ako pridobijete dovoljan broj ljudi za korišćenje Vašeg koda, možete da zahtevate od IEEE, ili ITU da ga proglasi standardom.

Postoje razni kodovi. Problem kod razmene podataka je činjenica da je neophodno uspostaviti komunikaciju između uređaja koji prepoznaju različite kodove. To je podjednako frustrirajuće kao kada pokušavate da razgovarate sa nekim ko ne govori Vašim jezikom. Da bi se komunikacija pojednostavila, razvijeni su neki standardni kodovi. Ali, da se ne biste osećali isuviše rasterećeno, recimo da postoje brojni nekompatibilni standardi! Jedna od najmudrijih izjava koja važi u oblasti komunikacija glasi: "Jedini problem sa standardima je to što ih ima toliko mnogo".

U zavisnosti od tipa podataka koji se skladište, vrši se i izbor koda koji će biti korišćen. Kodovi koji se koriste za predstavljanje celobrojnih vrednosti i realnih brojeva imaju značajne razlike i zavise od arhitekture kompjutera. Oni mogu da se sretnu u većini kompjuterskih knjiga, kao što je ona iz reference [St03]. Ovde ćemo se fokusirati isključivo na kodove karaktera.

Rani kodovi

Jedan od najstarijih kodova je Morzeov kod. Razvio ga je Semjuel Morze još 1838. godine, a korišćen je za telegraf. U tabeli 2.5 kod je prikazan kao sekvenca tačaka i crtica. Jedinstveni aspekt ovog sistema je to što su dužine kodova za slova različite dužine; na primer, slovu E odgovara samo jedna tačka, dok se za slovo H koriste četiri tačke. Promenljiva dužina koda omogućava brzo slanje poruka. Kod originalnog telegrafa poruka se šalje lupkanjem prekidača koji otvara i zatvara kolo.

Tabela 2.5: Bodov, Morzeov i BCD kod

Karakter	Bodov kod	Morzeov kod	BCD kod	Karakter	Bodov kod	Morzeov kod	BCD kod
A	00011	.–	110001	S	00101	...	010010
B	11001	–...	110010	T	10000	–	010011
C	01110	–.–.	110011	U	00111	..–	010100
D	01001	–..	110100	V	11110	...–	010101
E	00001	.	110101	W	10011	.––	010110
F	01101	..–.	110110	X	11101	–..–	010111
G	11010	–.	110111	Y	10101	–.––	011000
H	10100	....	111000	Z	10001	––..	011001
I	10110	..	111001	0	10110	–––––	001010
J	01011	.–––	100001	1	10111	.––––	000001
K	01111	–.–	100010	2	10011	..–––	000010
L	10010	.–..	100011	3	00001	...––	000011
M	11100	––	100100	4	01010	....–	000100
N	01100	–.	100101	5	10000	.....	000101
O	11000	–––	100110	6	10101	–....	000110
P	10110	.––.	100111	7	00111	–...	000111
Q	10111	––.–	101000	8	00110	––..	001000
R	01010	.–.	101001	9	11000	––––.	001001

Na primer, pretpostavimo da je dužina koda za svako slovo 5 (korišćenjem koda dužine 4 moguće je postići samo 2^4 = 16 kombinacija). Vreme potrebno za prenos poruke proporcionalno je umnošku broja 5 i broja slova u okviru poruke. Ako neka slova zahtevaju manje udaraca po prekidaču, radnik na telegrafu brže može da pošalje pomku. Da bi se iskoristila prednost koda promenljive dužine, najčešće korišćenim slovima su dodeljene kraće kombinacije. Ovaj metod olakšava redukovanje prosečne dužine koda.* Da bismo to ilustrovali, uzećemo primer alfabeta. Sa kodnom dužinom 5, da bi bila poslata poruka sa svih 26 karaktera, potrebno Je 130 udaraca. Korišćenjem Morzeovog koda ista poruka može da se prenese sa samo 82 udarca.

Sledeći kod, koji je razvio Žan-Mari-Emil Bod, dobio je naziv Bodov kod, a bio je dizajniran za francuski telegraf. Koristi se pet bitova za svaki karakter i slovo (videti tabelu 2.5). Pažljivi čitalac može da zaključi da je sa pet bitova moguće konstruisati 2' = 32 kombinacije, ali postoji 36 slova i cifara (da ne pominjemo ostale simbole koji nisu navedeni u tabeli 2.5). Ako pažljivije proučite tabelu, videćete da postoje duplirani kodovi. Na primer, cifra 1 i slovo Q imaju isti kod. U stvari, svi kodovi za cifre su već iskorišćeni za slova (možete li da ih pronadete?).

Nameće se logično pitanje kako da razlikujemo cifre od slova. Odgovor leži u istom principu koji omogućava jednom tasteru sa tastature da prikaže dva različita karaktera.

* I neki modemi kodovi imaju promcnljivu dužinu, što značajno utiče na cenu prenosa. Poglavlje 5 sadrži detaljnije informacije.

Na tastaturi taster Shift omogućava generisanje koda za jedan od dva moguća karaktera preko istog tastera. Bodov kod dodeljuje 5-bitne kodove 11111 (Shift je pritisnut) i 11011 (Shift je podignut) za utvrđivanje kako treba interpretirati naredne 5-bitne kodove. Nakon detektovanja spuštanja Shifta, prijemni uređaj interpretira sve naredne kodove kao slova. Interpretacija se nastavlja sve dok se ne detektuje podizanje Shifta. Potom se svi naredni kodovi interpretiraju kao cifre i drugi specijalni simboli. Tako bi poaika "ABC123" prevedena na Bodov kod izgledala ovako (čitano sleva udesno):

11111	00011	11001	01110	11011	10111	10011	00001
Shift pritisnut	A	B	C	Shift podignut	I	2	3

Sledeći je binarno-kodirani decimalni (BCD) kod, koji se, obično, koristi u ranim IBM-ovim mainframe kompjuterima. Jedan od razloga njegovog razvoja bilo je olakšavanje unosa i potrebnih izračunavanja numeričkih podataka. Na primer, ako je programer hteo da unese broj 4385, morao je na bušenoj kartici da izbuši cifre 4, 3, 8 i 5 (zapamtite: ovde govorimo o kompjuterskim "dinosaurusima"). *Nakon* toga je čitač kartica *čltao svaku* cifru. *Umesto da se kombinuju kodovi* za svaku cifru i da se kreira jedna reprezentacija za precizni numerički ekvivalent, svaka cifra je zapamćena pomoću BCD koda, koji je prikazan u tabeli 2.5. Ovaj metod se smatrao lakim i efikasnim, posebno ako je bilo potrebno uneti veću količinu podataka. Procesorska jedinica je onda mogla da izvrši aritmetičke operacije nad brojevima smeštenim u tom formatu. Zbog kompatibilnosti, neke arhitekture i dalje podržavaju izračunavanja izmedu brojeva u BCD formatu. Kako se tehnologija razvijala i javljale nove primene, postojala je sve veća potreba za smeštanjem nenumeričkih podataka. Zato je BCD kod proširen da uključuje i ostale karaktere. Tehnički, prošireni kod se zvao BCDIC kod *(binaiy-coded decimal interchange code)*.

ASCll kod

Najšire korišćeni kod je ASCII kod (American Standard Code for Information Interchange). To je 7-bitni kod koji dodeljuje jedinstvenu kombinaciju svakom karakteru sa tastature i nekim specijalnim funkcijama. Najčešće se koristi, ako ne i u potpunosti, na personalnim i nekim drugim kompjuterima. Medu karaktere koje je moguće štampati ubrajaju se slova, cifre i specijalni interpunkcijski znaci, kao što su zarezi, zagrade i upitnici. Pod karakterima koji ne mogu da se štampaju ne podrazumevaju se karakteri koji su istaknuti u novinama, na televiziji, ili na registarskim tablicama, Misli se na kodove koji ukazuju na specijalne funkcije, kao što su početak nove linije (line feed), tabulator, ili prelazak na početak linije (carriage return).

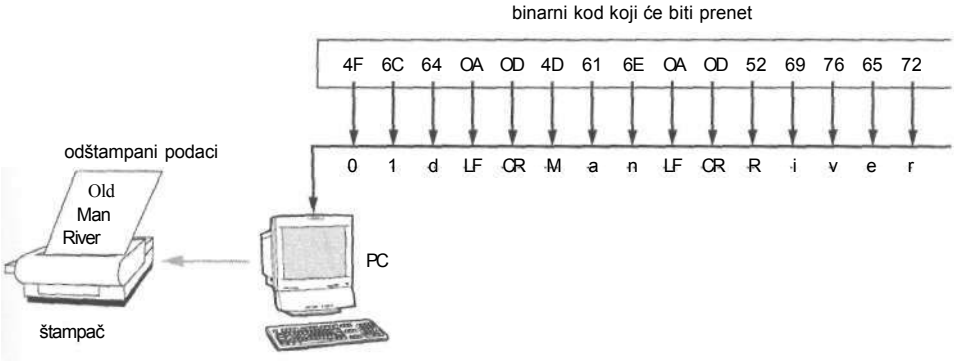
U tabeli 2.6 dati su karakteri i njihovi ASCII kodovi, i u binarnom i u heksadecimalnom formatu. Na primer, slovo M ima ASCII kod 1001101. Korišćenje heksadecimalne notacije omogućava grupisanje bitova u grupe od po četiri bita, 100-1101, što se interpretira kao 4D. Imajte na umu da ovde D nema isto značenje kao i slovo D. To je jednostavno heksadecimalna notacija za 1101.

Da bismo ilustrovali kako prenos može da izgleda, pretpostavimo da kompjuter šalje podatke na slici 2.26 do štampača koji prepoznaje ASCII kodove. Pretpostavimo da se kodovi šalju sleva udesno. Kako štampač bude primao svaki kod, on će ih analizirati i preduzimati neku akciju.

Tabela 2.6: ASCII kodovi

Binarni	Heks-adec	Karakter	Binarni	Heks-adec	Karakter	Binarni	Heks-adec	Karakter	Binarni	Heks-adec	Karakter
0000000	00	NUL	0010000	10	DLE	0100000	20	SP	0110000	30	0
0000001	01	SOH	0010001	11	DCl	0100001	21	!	0110001	31	1
0000010	02	STX	0010010	12	DC2	0100010	22	"	0110010	32	2
0000011	03	ETX	0010011	13	DC3	0100011	23	#	0110011	33	3
0000100	04	EOT	0010100	14	DC4	0100100	24	$	0110100	34	4
0000101	05	ENQ	0010101	15	NAK	0100101	25	%	0110101	35	5
0000110	06	ACK	0010110	16	SYN	0100110	26	&	0110110	36	6
0000111	07	BEL	0010111	17	ETB	0100111	27	'	0110111	37	7
0001000	08	BS	0011000	18	CAN	0101000	28	(	0111000	38	8
0001001	09	HT	0011001	19	EM	0101001	29	)	0111001	39	9
0001010	OA	LF	0011010	1A	SUB	0101010	2A	*	0111010	3A	:
0001011	OB	VT	0011011	1B	ESC	0101011	2B	+	0111011	3B	;
0001100	OC	FF	0011100	1C	FS	0101100	2C	,	0111100	3C	<
0001101	OD	CR	0011101	1D	GS	0101101	2D	-	0111101	3D	=
0001110	OE	SO	0011110	1E	RS	0101110	2E	.	0111110	3E	>
0001111	OF	Sl	0011111	1F	US	0101111	2F	/	0111111	3F	?
1000000	40	Ž	1010000	50	P	1100000	60	'	1110000	70	p
1000001	41	A	1010001	51	Q	1100001	61	a	1110001	71	q
1000010	42	B	1010010	52	R	1100010	62	b	1110010	72	r
1000011	43	C	1010011	53	S	1100011	63	C	1110011	73	S
1000100	44	D	1010100	54	T	1100100	64	d	1110100	74	t
1000101	45	E	1010101	55	U	1100101	65	e	1110101	75	U
1000110	46	F	1010110	56	V	1100110	66	f	1110110	76	v
1000111	47	G	1010111	57	W	1100111	67	g	1110111	77	W
1001000	48	H	1011000	58	X	1101000	68	h	1111000	78	x
1001001	49	I	1011001	59	Y	1101001	69	i	1111001	79	y
1001010	4A	J	1011010	5A	Z	1101010	6A	j	1U1010	7A	Z
1001011	4B	K	1011011	5B		1101011	6B	k	1111011	7B	
1001100	4C	L	1011100	5C	\	UOUOO	6C	I	1111100	7C	d
1001101	4D	M	1011101	5D		1101101	6D	m	1111101	7D	
1001110	4E	N	1011110	5E	^	1101110	6E	n	1111H0	7E	č
1001111	4F	O	1011111	5F	_	1101111	6F	o	1111111	7F	DEL

Dakle, prijem kodova 4F, 6C i 64 izaziva štampanje karaktera 0, 1 i d. Sledeća dva koda OA i OD odgovaraju karakterima koji se ne štampaju. U tabeli 2.6 prikazani su kao LF (line feed) i CR (carriage return), respektivno. Kada štampač primi kod OA, ne štampa ništa, već aktivira mehanizam za prelazak u sledeću liniju. Kod OD izaziva postavljanje mehanizma za štampanje na krajnju levu poziciju. U ovoj tački naredni karakteri koji se mogu odštampati javljaju se u novoj liniji, počevši od krajnje leve kolone (videti sliku 2.26). U tabeli 2.7 predstavljeno je još nekoliko kontrolnih karaktera.



SLIKA 2.26 *Prenos ASCII-kodirane poruke*

Tabela 2.7: ASCII kontrolni karakteri

BEL	Ovaj kontrolni karakter inicira ispuštanje zvuka na prijemnom uređaju (na primer, radnoj stanici), koji se obično čuje kao "bip". Ovaj signal se najčešće koristi za privlačenje korisnikove pažnje kada se pošalje specijalna poruka, ili kada se desi nešto značajno (slika 2.27).
BS	Back Space izaziva pomeranje mehanizma za štampanje, ili kursora za jednu poziciju unazad. Ovaj kontrolni karakter može da se koristi za štampanje dva karaktera na istoj poziciji (korisno kod podvlačenja), ili za štampanje boldiranog karaktera (isti karakter se štampa dva puta na istoj poziciji). Ako je reč o ekranu, prethodno otkucani karakter se menja novim.
CR	Carriage Return izaziva pomeranje mehanizma za štampanje, ili kursora na krajnju levu poziciju za štampanje. Napomena: Ovaj karakter je nezavisan od uvodenja nove linije.
DCI, DC2, DC3, DC4	Device Controls tasteri odgovaraju specijalnim funkcijama, ili karakteristikama koje zavise od uredaja. Na primer, DCI i DC3 ponekad odgovaraju karakterima X-ON i X-Office, koje generišu sekvence control-Q i control-S sa tastature. Ako se baferi uredaja napune, pošiljaocu može da se uputi karakter X-Office, tako da se zaustavi slanje. Slanje karaktera X-ON inicira nastavak prenosa.*
DLE	Data Link Escape se ponaša kao prekidač za promenu načina interpretacije primljenih karaktera.
FF	Form Feed se koristi na specijalnim obrascima, ili ekranima. Ovaj karakter postavlja mehanizam za štampanje, ili kursor ispred sledećeg obrasca, ili ekrana.
HT	Horizontal Tab postavlja kursor, ili mehanizam za štampanje ispred sledeće selektovane pozicije tabulatora.
LF	Line Feed uvodi novu liniju.
NUL	Null se koristi kao filter (za zauzimanje praznog prostora kada nema podataka) u parcijalno popunjenom zapisu.
VT	Vertical Tab postavlja kursor, ili mehanizam za štampanje ispred sledeće unapred definisane linije za štampanje.

* Studenti se često prvi put sasvim slučajno sreću sa karakterima X-ON i X-OFF. Na primer, pretpostavimo da unosite komandu sa radne stanice za štampanje fajla (poput one u mašinskom jeziku) koji sadrži karaktere koji se ne mogu štampali. Sadržaj fajla se pogrešno interpretira kao karakteri ASCII koda. Ako neki od tih kodova odgovara karaktem X-OFF, radna stanica interpretira taj karakter kao znak za zaustavljanje slanja karaktera. Zbog toga se ne prenosi ono što korisnik unosi i radna stanica se "zamrzava".

SLIKA 2.27 *Specifični događaj zahteva korišćenje BEL karaktera*

EBCDIC kod

Sledeći je **EBCDIC (Extended Binary** Coded **Decimal Interchange Code)** kod; koristi se prvenstveno na IBM-ovim mainframe kornpjuterima i periferijama. To je 8-bitni kod koji ornogućava kodiranje najviše 256 karaktera. Poput ASCIl koda, postoje karakteri koji mogu i koji ne mogu da se štampaju; ovde nećemo navoditi celu tabelu. Zainteresovani čitaoci verovatno mogu da pronađu tabelu na Internetu.

Unicode

ASCII i EBCDIC su odavno u upotrebi, ali oni koji ih često koriste termine bajt i *karakter* smatraju sinonimima. Oba koda su prvenstveno korišćena za predstavljanje uobičajenih upravljačkih funkcija, slova i karaktera iz engleskog alfabeta. Medutim, sa internacionalizacijom mrežnih aplikacija, 7-bitni i 8-bitni kodovi su postali nefleksibilni, a razvijen je novi standard **Unicode.** Unicode podržavaju razni skriptovi, ili kolekcije matematičkih simbola i specijalnih karaktera koji postoje u odredenom jeziku. Primeri su skriptovi Arabic, Latin, Greek, Gothic i Cyrillic. I Java koristi Unicode za podršku tipa *char* u Javi.

Unicode definiše jedinstveni 16-bitni broj za svaki karakter, nezavisno od jezika i platforme. Unicode Consortium, neprofitabilna organizacija koja saraduje sa ISO, zadužena je za specifikacije. Članovi su kompanije kao što su Apple, Microsoft, Oracle, IBM, Novell i Netscape. Unicode i dalje uključuje nove skriptove i dodaje ih u svoju definiciju. U vreme kada je ova knjiga nastajala, Unicode 3.2 je imao definisane kodove za više od 90.000 karaktera. Zato nismo naveli tabelu i za Unicode karaktere. Zainteresovani mogu da pronadu više informacija na adresi www.unicode.org.

2.6 Zaključak

Ovo poglavlje je najvećim delom bilo posvećeno komunikacionim medijumima i opremi, aplikacijama i komunikacionim kodovima. U njemu su predstavljeni sledeći značajni koncepti:

- Informacije se kodiraju kao sekvence bitova (predstavljene nulama i jedinicama) i prenose se pomoću električnih signala, ili elektromagnetnih talasa.
- Medu primame komunikacione medijume ubrajaju se kablovi sa upredenim paricama, koaksijalni kablovi, fiber optički, mikrotalasni i satelitski prenosi, infracrveni i radio talasi i tehnologija free space optics.
- Provodni metali koji se koriste u kablovima sa upredenim paricama i koaksijalnim kablovima jeftiniji su od fibera i lakše se slažu. Medutim, imaju manje opsege signala i podložni su električnoj interferenci.
- LAN okruženja često u najvećoj meri koriste kablove sa upredenim paricama za povezivanje opreme u susednim prostorijama, a optički fiber se obično koristi za povezivanje zgrada. Optički fiber se često koristi za dalekovode. Koaksijalni kabl se danas rede koristi nego ranije, mada se obično koristi za kablovske televizijske servise i može da se koristi za povezivanje uredaja na LAN-u na mestima na kojima je neophodno obezbediti zaštitu od električne interference. Ipak, niža cena i veće brzine prenosa čine UTP kablove primarnim izborom u mnogim mrežama.
- Mikrotalasne i satelitske komunikacije se obavljaju kroz slobodan prostor - ne oslanjaju se na fizičke konekcije. Sateliti obično nude komunikaciju širom sveta, a mikrotalasni tornjevi omogućavaju komunikacije na udaljenostima koje bi bilo nepraktično, ili nemoguće "pokriti" klasičnim fizičkim konekcijama.
- Geostacionarni sateliti zadržavaju fiksnu poziciju u odnosu na poziciju posmatrača na Zemljinoj površini i korisni su za emitovanje programa i komunikacione aplikacije. Bitske brzine zavise od frekvencija prenosa. Veće frekvencije obezbeduju veće brzine prenosa podataka, ali su podložnije atmosferskim uticajima.
- LEO (low earth orbit) sateliti neprestano putuju preko neba. Teledesic treba da obezbedi globalnu komunikacionu mrežu.
- Danas se sve više koriste bežične LAN tehnologije. Prenosivi uredaji, kao što su notebook kompjuteri, mogu da prenose infracrvene, ili radio talase do pristupne tačke koja je povezana na LAN. Tako je omogućena komunikacija uredaja sa ostalim uredajima na LAN-u, i pored toga što ne postoji fizička konekcija.
- Free space optics je nova bežična tehnologija koja nudi veće brzine prenosa podataka, U poredenju sa onima koje nudi optički fiber. Za prenos informacija na udaljenosti od kilometra, ili manje koristi se laser (visokofokusirani svetlosni mlaz).
- Za prenos podataka, bez obzira na medijum koji se koristi, moramo da koristimo kodove, mehanizme koji nizovima bitova pridružuju odredene informacije. Najčešće se koriste ASCII (American Standard Code for Information Interchange) i EBCDCI (Extended Binary Coded Decimal Interchange Code). Svakom nizu bitova je dodeljen taster sa tastature, uključujući i razne upravljačke funkcije. Medu češće korišćene kodove ubrajaju se Bodov, Morzeov i BCD (binarno kodirani decimalni kod).

- Unicode je noviji kod koji se koristi u raznim aplikacijama za podršku većoj paleti karaktera i skriptova od one koju mogu da podrže ASCII i EBCDIC kodovi.
- Razmena podataka je polje koje se još uvek menja. Novija istraživanja dovode do povećavanja brzine prenosa podataka na različitim medijumima i redukuje se cena medijuma tako da budu tehnološki i ekonomski pristupačniji nego ikada ranije.

Pitanja i zadaci za proveru

1. Šta je periodični signal?
2. Navedite pet medijuma za prenos i poređajte ih po brzini prenosa podataka.
3. U čemu je razlika između digitalnog i analognog signala?
4. U čemu je razlika izmedu bitske brzine i opseg signala?
5. U kakvoj su vezi perioda i frekvencija signala?
6. U čemu je razlika izmedu prenosa u osnovnom opsegu i širokopojasnog prenosa?
7. U čemu je razlika izmedu kablova sa upredenim paricama Cat 4 i Cat 5?
8. Definišite indeks refrakcije.
9. U čemu je razlika izmedu ThickNeta i ThinNeta?
10. U čemu je razlika između lasera i LED dioda kod optičkih fiber komunikacija?
11. Navedite tri moda za optičke fiber komunikacije i uporedite ih.
12. Da li su sledeće tvrdnje tačne, ili netačne i zašto?
    a. Direktni mikrotalasni prenosi mogu da se dese izmedu bilo koje dve tačke na površini Zemlje.
    b. Satelitski prenos zahteva komunikaciju sa stacionamim satelitom.
    c. Deblji optički fiber dopušta veće brzine prenosa podataka.
    d. Brzine prenosa podataka preko satelita ograničene su samo mogućnostima opreme za slanje i prijem visokofrekventnih signala.
    e. Pošto se LEO sateliti kreću po nebu i silaze ispod horizonta, nisu praktični za komunikacione aplikacije.
    f. Lokalne mreže ne zahtevaju fizičke konekcije izmedu komponenata.
    g. Svetlost može da putuje kroz optički fiber različitim brzinama.
    h. Vidljiva svetlost i elektromagnetni talasi predstavljaju istu "stvar".
    i. Optički fiber ima izdubljeni centar kroz koji svetlost putuje i reflektuje se od reflektujuće površine oko njega.
    j. Free space optics zahteva direktnu liniju viđenja.
13. U čemu je razlika između antena u obliku roga i paraboličnog tanjira?
14. Navedite tri načina za komunikaciju bežičnih uređaja.
15. Gde je korišćena prva bežična mreža?
16. Šta je Bluetooth tehnologija?
17. Koji su prenosi osetljivi na interferencu?

18. Zašto se žice kod kablova sa upredenim paricama upredaju (umesto da se postave paralelno)?
19. Koja je svrha oblaganja optičkih fiber kablova?
20. Štaje"Sputnik"?
21. Navedite neke prednosti i nedostatke LEO satelita u odnosu na geostacionarne satelite.
22. Po čemu se free space optics razlikuje od fiber optičkih komunikacija?
23. Navedite neke prednosti i nedostatke tehnologije free space optics.
24. U čemu je razlika izmedu karaktera koji se mogu štampati i onih koji se ne mogu štampati?
25. U čemu je razlika izmedu ASCII i EBCDIC kodova?
26. Po čemu se Unicode razlikuje od tradicionalnih ASCII i EBCDIC kodova?
27. Šta je bio razlog za defmisanje Unicodea?
28. Bodov kod za karakter 0 (nula) je isti kao i za karakter P. Kako je to moguće?

Vežbe

1. Ako je perioda signala 10 nanosekundi, kolika je frekvencija?
2. Ako je frekvencija signala 500 megaherca, kolika je perioda?
3. Ako je orbitalna visina satelita fiksna, zašto nije moguće menjati vreme potrebno za obilazak oko Zemlje promenom brzine satelita?
4. LAN mreže se često povezuju korišćenjem koaksijalnih kablova, umesto kablova sa upredenim paricama, zbog boljih bitskih brzina i otpornosti na spoljašnji šum. Zašto se koaksijalni kabl ređe koristi u tekućim LAN tehnologijama?
5. Pretpostavimo da kompanija pokušava da uspostavi komunikaciju izmedu nekoliko mesta u različitim delovima grada. Da li su mikrotalasi dobra ideja? Zašto jesu, ili zašto nisu?
6. Kako satelit može da zadrži fiksnu poziciju na nebu? Kako ga gravitacija ne povuče na zemlju?
7. Uz pretpostavku da signali putuju brzinom svetlosti (186.000 milja u sekundi), koliko vremena signal putuje od zemaljske stanice do geostacionarnog satelita? Koliko je vremena potrebno ako se satelit nalazi u niskoj orbiti od 875 milja?
8. Projekat Teledesic je promenio inicijalnu konfiguraciju u odnosu na prvo javno objavljivanje. Proučite ovu temu i napišite nekoliko pasusa o trenutnom stanju projekta Teledesic.
9. Napišite program koji od korisnika traži da unese poruku, ispušta sistemski zvuk za unos broja i štampa unete podatke.
10. Ako imate pristup univerzitetskoj mreži, ili mreži kompanije, pogledajte koji su kablovi korišćeni za povezivanje (UTP, sa upredenim paricama, ili koaksijalni, optički fiber).
11. Kako glasi Bodov kod za niz karaktera SDG564FSDH65?
12. U tabeli 2.6 prikazani su ASCII kodovi za cifre 0 do 9. Zašto nema koda za broj 10?

Reference

[AlOl] Allen, D. "The second Coming of Free Space Optics". *Netivork,* vol. 16, no. 3 (March2001.). 55-63.

[AnOO] Angel, J. "Look Ma, No Cables". *Netivork,* vol. 15, no. Il (November 2000.), 42-52.

[C145] Clarke, A. C. "Extra-Terrestrial Relays: Can Rocket Stations Give World-Wide Radic Coverage?" *Wireless World* (October 1945.).

[DoOO] Dornan, A. "Can Bluetooth Sink Its Teeth into Networking?" *Nettuork,* vol. 15, no 11 (November 2000.), 54-60.

[Ek02] Ekici, E. I. Akyildiz, and M. Bender. "A Multicast Routing Algorithm for LEO Satellite IP Netvvorks". *IEEE/ACM Transactions on Netivorking,* vol. 10, no. 2 (April 2002.), 183-192.

[Hu90] Hudson, H. *Communication Satellites.* New York: Free Press, 1990.

[MeOO] Metz, C. "IP-Over-Satellite: Internet Connectivity Blasts Off'. *IEEE Intemel Computing,* vol. 4, no. 4 (July/August 2000.), 84-89.

[RoOl] Rogers, *A. Understanding Optical Fiber Communications.* Boston: Artech House, 2001.

[Sh90] Sherman, K. *Data Communication:* AUser's Guide, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[StOO] Stallings, W. "Gigabit Ethernet". *Dr. Dobbs Journal,* vol. 25, no. 5 (May 2000.).

[St03] Stallings, W. *Computer Organization and Architecture: Designing for Performance,* 6th ed. Englewood Cliffs, NJ: Prentice-Hall, 2003.

[We98] Wesel, E. *Wireless Kiultimedia Communications.* Reading, MA: Addison-Wesley, 1998.

POGLAVLJE 3

Analogni i digitalni signali

Informacione mreže su se probile u sve aspekte života. Ništa nije izostavljeno. Ali, sa ovolikom količinom informacija, njihovo značenje počinje Aa se rasplinjuje. Nemoguće ih je u potpunosti obuhvatiti!

—**Ginter Gras,** nemački književnik

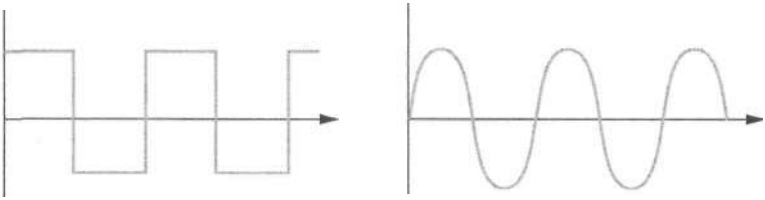
Najniža forma popularne kulture - nedostatak informađja, netačne informacije, dezinformisanost i prezir ka istini, ili realnosti života većine Ijudi - sve je pristunija u novinarstvu. Danas novine prosečnog Amerikanca svakodnevno "zatipavaju" gomilom "đubreta".

—**Karl Bernštajn,** američki novinar

3.1 Uvod

"Pokrili" smo dve primarne oblasti prenosa podataka: medijum i simboličko predstavljanje podataka. Sada je vreme da ih kombinujemo. Drugim rečima, pošto znamo kako se podaci mogu simbolički smeštati, pitanje je kako se uspostavlja reladja sa električnim signalima, mikrotalasima, ili svetlosnim talasima. Kako obični i kablovski modemi prenose podatke? Šta je DSL tehnologija? Sledeći logičan korak je povezivanje fizičkih signala sa simboličkom reprezentacijom podataka. Prostije rečeno, kako O i 1 izgledaju dok "putuju" kroz kabl, optički fiber, ili proslor? Koliko bitova signal može da prenese u jedinici vremena i da li postoje neka ograničenja? Da li na podatke utiče i električna interferenca (šum)? Ako je odgovor potvrdan, kako utiče?

Svi ovi odgovori zavise od toga da li koristimo digitalne, ili analogne signale. Setite se da smo u Poglavlju 2 rekli da se digitalni signal može predstaviti naizmeničnom sekvencom visokih i niskih vrednosti (0 i 1); pogledajte sliku 3.1a. Analogni signal se kontinuelno menja u intervalu izmedu dve vrednosti. Digitalni signal ima konstantnu vrednost za kratko vreme, a zatim se maenja na drugu vrednost. Pošto se analogni signal menja kroz vreme, često se predstavlja karakterističnim sinusoidalnim talasnim oblikom (slika 3.1b).



(a) Digitalni signal (b) Analogni signal

SLIKA 3.1 *Analogni i digitalni signali*

U Poglavlju 2 smo rekli da se digitalni signal može koristiti za predstavljanje podataka pridruživanjem 0 i 1 ili visokom, ili niskom signalu. Medutim, ovaj pristup ima brojne probleme. Zato je neophodna alternativa. U odeljku 3.2 opisani su različiti pristupi. Analogni signali stvaraju nove probleme. Pošto se vrednost signala neprestano menja, kako je moguće defmisati O i 1? Moramo bolje da razumemo karakteristike analognih signala. U odeljcima 3.3 i 3.4 data su detaljnija objašnjenja i povezan je koncept bitske brzine sa karakteristikama signala. Osim toga, predstavljena su dva čuvena rezultata koja definišu granice u količini podataka koju je moguće preneti u jedinici vremena, u skladu sa karakteristikama signala.

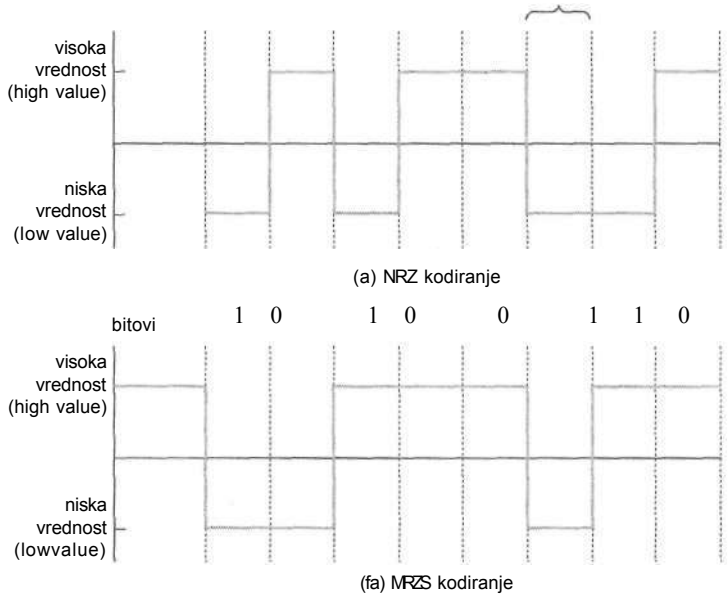
Ovo stvara sledeći problem: digitalni i analogni signali prenose podatke na različite načine. Zbog toga, za kompjuter koji uspostavlja komunikaciju preko telefonske linije moramo da pronademo način za konvertovanje signala iz jednog tipa signala u drugi na odlasku i dolasku. U odeljcima 3.5 i 3.6 opisane su neke tehnike. Odeljak 3.7 "pokriva" modeme (uobičajene uredaje za konvertovanje signala) i standarde za konvertovanje. Konačno, u odeljku 3.8 predstavljamo DSL (Digital Subscribe Line digitalnu pretplatničku liniju), alternativu za konvencionalne i kablovske modeme.

3.2 Šeme za digitalno kodiranje

Izmedu digitalnog signala i digitalno kodiranih podataka postoji prirodna veza. Podaci koji su digitalno zabeleženi predstavljeni su kao nizovi O i 1. Pošto digitalni signali imaju dve moguće konstantne vrednosti, jednostavno se jednoj vrednosti dodeli 0, a drugoj 1. Konkretne vrednosti nisu bitne. Kada se koriste električni signali, često se koriste iste vrednosti, ali sa različitim predznakom. Da bismo raspravu zadržali na opštem nivou, koristićemo termine "visoki napon" i "niski napon".

NRZ kodiranje

NRZ (nonreturn to zero) šema predstavlja verovatno najprostiju šemu kodiranja. 0 se prenosi prelaskom signala sa nižeg na viši nivo, a 1 niskim naponom. Tako se odgovorajućim promenama naponskog nivoa može preneti bilo koja sekvenca nula i jedinica.

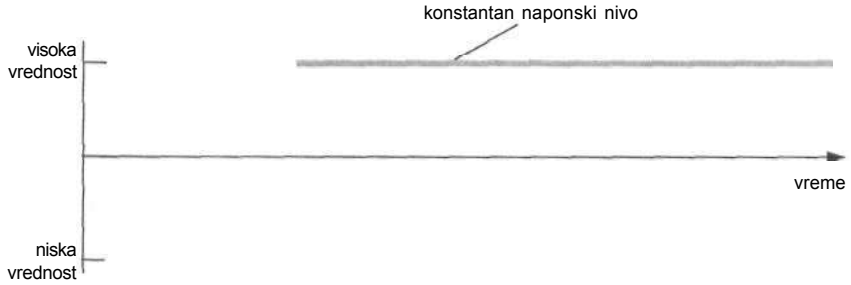


SLIKA 3.2 *NRZ i NRZI kodiranje*

Naziv šeme potiče od činjenice da nivo napona ostaje konstantan (ne vraća se na nulu) dok se bit prenosi. Na slici 3.2 prikazan je NRZ prenos binarnog stringa 10100110.

Alternativa NRZ kodiranju je *NRZI* (NRZ invertovano). Razlika izmedu NRZ i NRZI sastoji u tome što se kod NRZI 1 prenosi promenom vrednosti napona. Ako je signal bio nizak, postaje visok. Ako je bio visok, postaje nizak. Na slici 3.2b prikazan je NRZI signal za isti binarni string sa slike 3.2a.

NRZ i NRZI kodiranja su jednostavna, ali imaju jedan problem. Pogledajte prenos na slici 3.3. Šta se prenosi?



SLIKA 3.3 NRZ *kodiranje sekvence 0*

Vaš odgovor bi trebalo da glasi "sekvenca 0". Dobro, to je možda tačno, ali koliko ima 0? Odgovor na ovo pitanje utvrđujete na osnovu trajanja jednog bita. Pretpostavite sada da smo kod grafičke reprezentacije rekli da jedan bit odgovara liniji dužine 1 milimetar. Sve što treba da uradite je da izmerite liniju i da dužinu pretvorite u milimetre. Tako ćete utvrditi koliko 0 ima u sekvenci. U teoriji ovaj metod funkcioniše, ali ne i u praksi. Pretpostavimo da jedna osoba konstruiše 1.000 jednomilimetarskih segmenata od početka do kraja. Koliko je dugačka rezultujuća linija? Odgovor treba da bude: jedan metar. Medutim, odstupanja u merenju **i** iscrtavanju razlog SU da linija bude dugačka skoro, ali ne i tačno jedan metar. Nakon toga, druga osoba koja meri liniju zaključuje da je ona dugačka malo više, ili manje od 1.000 segmenata. Čak i ukoliko je prva osoba imala sreće i izmerila tačno, nepreciznost u radu druge osobe izaziva neslaganje.

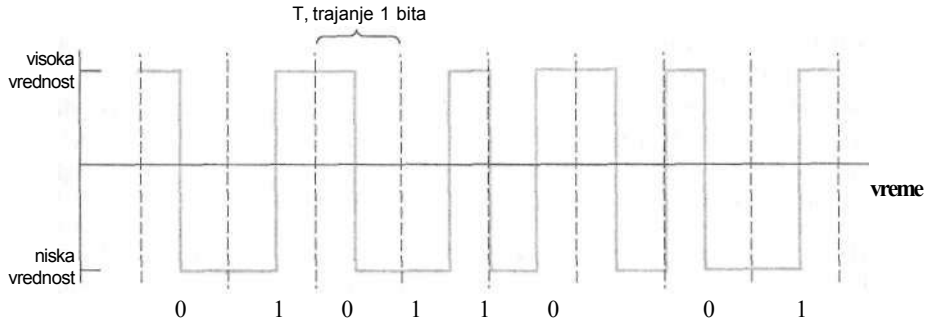
Kakve ovo ima veze sa prenosom podataka? Kada uređaj prenosi digitalni signal za jedan bit, on generiše konstantan signal određenog trajanja, recimo T. Intemi takt definiše tajming. Prijemni uredaj mora da zna trajanje signala, tako da sempluje signa] na svakih T jedinica. Osim toga, i on ima interni takt koji definiše tajming. Dakle, dovoljno je osigurati da oba takta koriste isti T.

Sledeće pitanje je da li svi uredaji u Vašoj kuđ imaju isto vreme odbrojavanja do u sekundu (kod mene to nije slučaj). Nažalost, svaki fizički uredaj je "sklon" ograničenjima dizajna i nesavršenostima, Skoro je sigumo da u taktovima uredaja postoje male razlike koje izazivaju pomeranje semplovanja signala prilikom prenosa sa jednog uredaja na drugi. To je slično sinhronizovanju časovnika u novogodišnjoj noći, kada vidimo da se kod većine javljaju razlike od par sekundi. Slično tome, muzičari u orkestru moraju da počnu sviranje u istom trenutku i istim tempom, ali, ako ne prate dirigenta i ne slušaju jedni druge, doći će do neznatnog razilaženja u tempu (neće nastati veliko kašnjenje koje bi narušilo muziciranje, tako da deluje kao da svi sviraju u istom trenutku).

Komunikacionim uređajima je neophodan neki mehanizam da bi se onemogućila odstupanja u tajmingu, slično dirigentu koji nastoji da održi uskladenost muzičara u orkestru. Kod konstantnog signala ne postoji mehanizam za sinhronizaciju. Međutim, ako se signal menja, promene mogu da se koriste za održavanje sinhronizovanosti uredaja. Zbog toga se u nekim šemama uvodi promena signala.

Mančester kodiranje

Mančester (Manchester) kod koristi promene za očuvanje sinhronizacije izmedu uređaja koji šalju i koji primaju podatke. Neki ga nazivaju i **samosinhronizujući kod.** Da bi se izbegla situacija sa slike 3.3, razlika izmedu 0 i 1 označena je promenom naponskog nivoa. 0 se predstavlja promenom napona sa visoke na nisku vrednost, a 1 promenom napona sa niske na visoku vrednost. Na slici 3.4 prikazan je Mančester-kodirani prenos stringa 01011001. Kao što slika prikazuje, signal nikada neće imati konstantnu vrednost duže od jednog bitskog intervala. Cak i kada je reč o sekvenci 0, ili 1, signal se menja na sredini svakog intervala. Ova promena omogućava taktu prijernnog uredaja da ostane konzistentan sa taktom uređaja koji šalje signale. Nedostatak Mančester kodiranja je to što je neophodan dvostruki opseg signala - signal mora da se menja dva puta češće nego kod NRZ kodiranja.

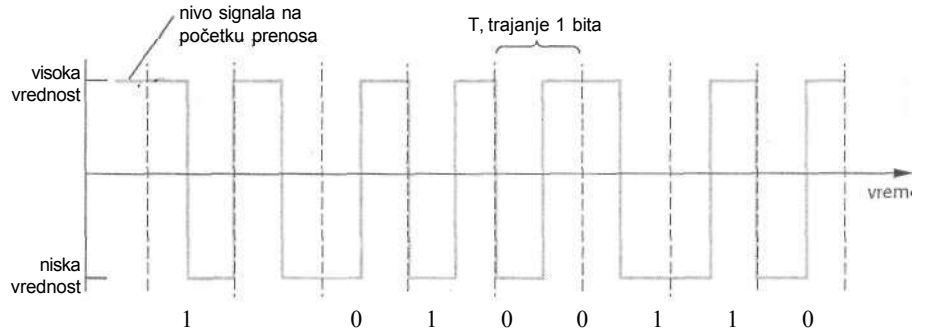


SLIKA 3.4 *Mančester kodiranje*

Varijacija ovog metoda se naziva **difereacijalno Mančester kodiranje.** Slično Mančester kodiranju, uvek dolazi do promene signala na polovini svakog bitskog intervala. Razlika je u onome što se dešava na početku svakog intervala. 1 izaziva zadržavanje signala na istom nivou na kome je bio na kraju prethodnog intervala. Prema tome, 0 može da ide ili sa niskog na visoki, ili sa visokog na niski nivo, u zavisnosti od inicijalne vrednosti signala. Na slici 3.5 prikazano je diferencijalno Mančester kodiranje za string 10100110. U ovom slučaju se 0 i 1 razlikuju po tome da li se promena signala dešava na početku intervala. Detektovanje promena Je često pouzdanije, posebno ako u kanalu postoji veći šum.

3.3 Analogni signali

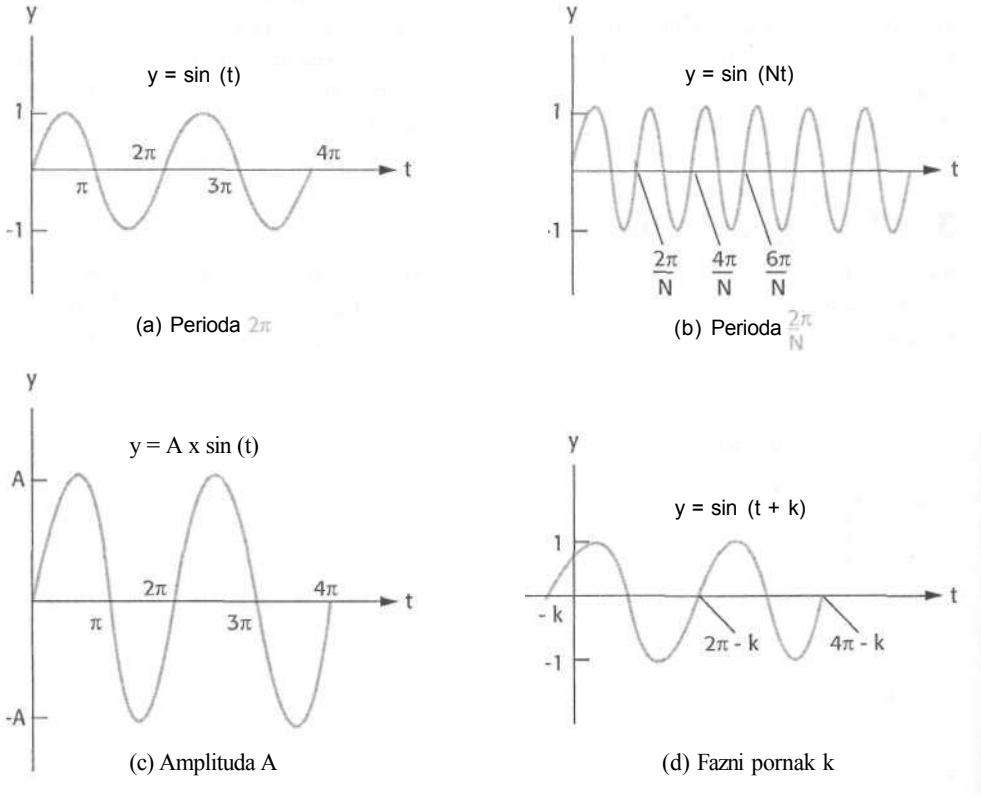
Rad sa analognim signalima uvodi još više složenosti u razmenu podataka. Problem je to što su digitalni kompjuteri nekompatibilni sa analognim medijumima za prenos. Iako je većina telefonskih sistema digitalna, kablovi koji se povezuju direktno sa Vašim telefonom prenose analogne signale.



SLIKA 3.5 *Diferencijalno Mančester kodiranje*

Zbog toga, uredaji u lokalnoj centrali očekuju analogne signale. Ako povežete kompjuter na telefonsku liniju, potreban Vam je uređaj koji konvertuje digitalni signal iz kompjutera u analogni signal **(modulacija)** radi prenosa preko linije. Taj uredaj mora da konvertuje i analogne signale koji se primaju preko telefonske linije u digitalne signale **(demodulacija)** radi prenosa nazad u kompjuter. Modem (skraćeno od modulacija/demodulacija) je uredaj koji vrši obe konverzije. Funkcije i standarde za modeme predstavićemo nešto kasnije u ovom poglavlju, ali najpre moramo da izložimo teorijsku osnovu za rad sa analognim signalima.

Za početak, definisaćemo analogni signal. Ranije smo rekli da se analogni signal kontinuelno menja izmedu dve vrednosti i za njegovu ilustraciju smo koristili dijagram sličan onome na slici 3.6. Ova definicija je sigurno tačna, ali je daleko od kompletnog opisa. Signal sa slike 3.6a može matematički da se predstavi trigonometrijskom funkcijom y = sin (t). Drugim rečima, slika 3.6a je grafik funkcije y = sin (t). Pošto se sinusna funkcija može menjati na razne načine, to znači da možemo da menjamo rezultujući signal. U opštem slučaju, analogni signal karakterišu njegova frekvencija, amplituda i fazni pomak.



SLIKA 3.6 *Analogni signali*

□□□ se signal menja u vremenu i ako se uzorak kontinuelno ponavlja, perioda je vreme koje je potrebno da se kompletira jedan uzorak. Takva funkcija je periodična. Na slici □□□ perioda je 2π. Medutim, promenom funkcije □ = sin (Nt) promenili smo periodu u 2π/N (slika 3.6b). Dakle, kada se t menja od 0 do 2π/N, argument sinusne funkcije (Nt) ide od 0 do 2π. U opštem slučaju, ako je N>l, perioda je manja od 2π. Ako je N<1, perioda je veća od 2π.

Perioda je povezana sa frekvencijom, brojem oscilacija signala u jedinici vremena. Mera za frekvenciju je broj ciklusa u sekundi, tj. herc (Hz). Ako je f frekvencija, a p perioda, onda važi

$$f = \frac{1}{p}$$

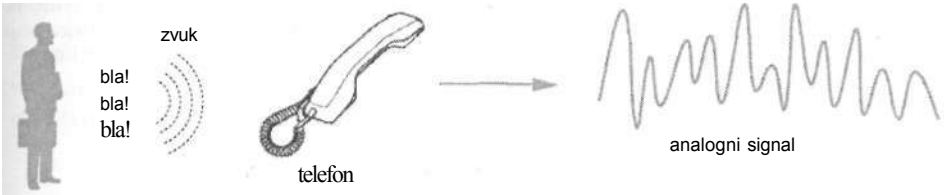
Značni, signa] sa slike 3.6b ima frekvenciju N/27t Hz.

Amplituda definiše vrednosti izmedu kojih signal oscilira. Pošto y = sin (t) oscilira izmedu 1 i -1, y = A * sin (t) oscilira izmedu A i -A (slika 3.6c).

Poslednji način za promenu signala obezbeden je **faznim pomakom.** Grafički, to je horizontalni pomak na grafikonu sinusoidalne funkcije. U opštem slučaju, horizontalni pomak može da se postigne dodavanjem, ili oduzimanjem od argumenta. Na primer, ako je k>O, grafik y = sin (t+k), predstavljen na slici 3.6d, pomeren je za k jedinica ulevo u odnosu na grafikon sa slike 3.6a. Ova promena se lako potvrđuje izračunavanjem obe funkcije sa različitim vrednostima za t.

Furijeovi rezultati

Videli smo da je analogni signal mnogo složeniji od običnog sinusiodalnog talasnog oblika (grafikon sinusne funkcije). U opštem slučaju, amplituda, frekvencija i fazni pomak mogu da se menjaju u vremenu i na taj način je moguće kreirati složene funkcije. Verovatno je najpoznatiji primer analognog signala onaj koji se stvara kada razgovarate preko telefona (slika 3.7). Dok govorite, menjate zvučne talase da bi se proizvodile različite reči. Vaš glas postaje glasniji, ili tiši, u zavisnosti od toga da li se raspravljate sa svojim šefom, ili razgovarate sa verenicom. Izgovaranje glasnijih, ili tiših reči, ili većom, ili manjom snagom, kreira zvuk koji se prevodi u električne analogne signale. Amplituda odslikava jačinu glasa, a frekvencija snagu (za sada, ne postoji jednostavan ekvivalent za fazni pomak). Rezultat je složena kombinacija signala koja predstavlja glas.



SLIKA 3.7 *Zvuk kreira analogni signal*

Sada je problem kako preneti složene signale. Postoji beskonačan broj načina za promenu amplitude, frekvencije i faznog pomaka. Osim toga, praksa pokazuje da su različiti signali podložni različitim stepenima izobličenja. Kako inženjeri dizajniraju hardver kojim se postiže veran prenos signala? Da li dizajniraju različite hardverske uređaje i medijume za prenos različitih tipova signala? Da li su za funkcije koje predstavljaju različite analogne signale neophodne posebne analize?

Odgovor na poslednja dva pitanja je negativan. Čuveni matematičar Žan Batist Furije razvio je teoriju koja pokazuje da svaka periodična funkcija može da se izrazi kao beskonačna suma sinusnih funkcija sa različitim amplitudama, frekvencijom i faznim pomakom. Ta suma se naziva Furijeov red. Ona je značajna zbog toga što pokazuje da se sve funkcije, bez obzira na složenost, sastoje od istih komponenata.

Koristeći matematički zapis, pretpostavićemo da je *s(t)* periodična funkcija sa periodom P. Jedan oblik Furijeovih rezultata ukazuje da je

$$s(t) = \frac{a_0}{2} + \sum_{i=1}^{\infty} \left[a_i \times \cos\left(\frac{2\pi i t}{P}\right) + b_i \times \sin\left(\frac{2\pi i t}{P}\right) \right]$$

(Postoje i drugi oblici, ali ovaj odgovara našim trenutnim potrebama.)

Koeficijenti a, (i=0, 1, 2, ...) i fc, (i=0, 1, 2, ...) odredeni su na osnovu izraza

$$a_i = \frac{2}{P} \int_{-P/2}^{P/2} s(t) \times \cos\left(\frac{2\pi i t}{P}\right) dt \quad \text{for } i = 0, 1, 2, 3, \ldots$$

i

$$b_i = \frac{2}{P} \int_{-P/2}^{P/2} s(t) \times \sin\left(\frac{2\pi i t}{P}\right) dt \quad \text{for } i = 0, 1, 2, 3, \ldots$$

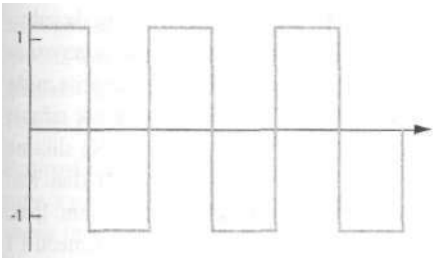
Ovde nećemo pokazivati kako su ove formule izvedene. Jednostavno ih navodimo radi objašnjavanja ograničenja različitih komunikacionih medijuma. Ako ste zainteresovani, detaljniji opis Furijevih redova možete da pronadete u referencama [StOO], [Ge99] i [Ch02]. Ono što je za nas ovde bitno je činjenica da Furijeova analiza pokazuje da je svaki periodični signal suma analognih signala sa različitim frekvencijama i amplitudama. Na osnovu ovog tvrđenja, zaključujemo da mogućnost slanja i analize analognog signala zavisi od opsega frekvencija (opsega signala) koji medijum može da podrži.

Razmotrimo sledeći primer. Neka se *s(t)* definiše kao

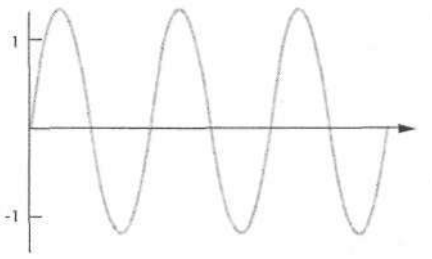
$$s(t) = \begin{cases} 1 \text{ for } 0 \le t < \pi; 2\pi \le t < 3\pi; 4\pi \le t < 5\pi; \text{ etc.} \\ -1 \text{ for } \pi \le t < 2\pi; 3\pi \le t < 4\pi; 5\pi \le t < 6\pi; \text{ etc.} \end{cases}$$

Na slici 3.8a dat je grafik ove funkcije. Pošto je reč o periodičnoj funkciji (sa periodom 2!π), možemo je napisati pomoću Furijeovog reda. U ovom slučaju, sve konstante a_i za i>0, jednake su 0. Konstante bi su definisane sa

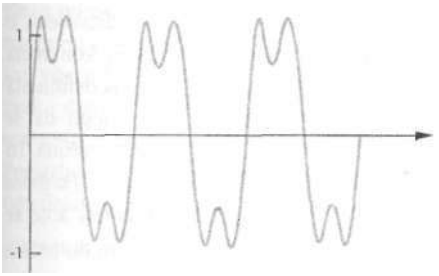
$$b_i = \begin{cases} 0 \text{ if } i \text{ is even} \\ \frac{4}{\pi i} \text{ if } i \text{ is odd} \end{cases}$$



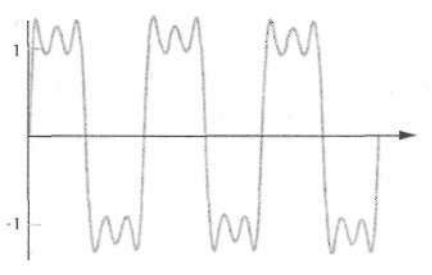
(a) Grafik za s(t)



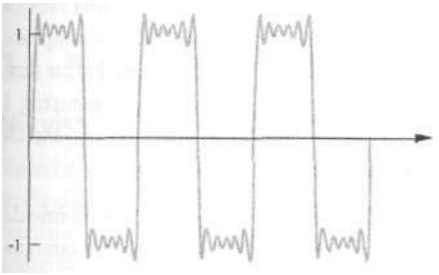
(b) Furijeova aproksimacija funkcije s(t) sa jednim članom



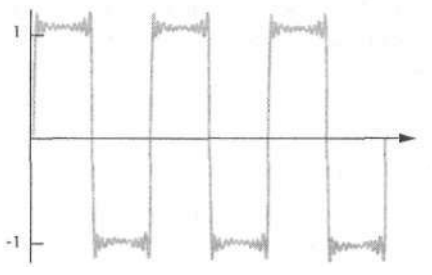
(c) Furijeova aproksimacija funkcije s(t) sa tri člana



(d) Furijeova aproksimacija funkcije s(t) sa pet članova



(e) Furijeova aproksimacija funkcije s(t) sa 11 članova



(f) Furijeova aproksimacija funkcije s(t) sa 21 članom

SLIKA 3.8 *Furijeove aproksimađje*

Koristili smo računicu za utvrđivanje ovih vrednosli i ovde nećemo navoditi postupak za izračunavanje. Ako znate kako funkcioniše integralni račun, možete da proverite naše rezultate. Ako ne znate, obratite pažnju na dole navedenu liniju, u kojoj smo periodičnu funkciju iskazali kao

$$s(t) = \sum_{i=1 \text{ and } i \text{ odd}}^{\infty} \frac{4}{\pi i} \sin(it)$$

Izračunavanje beskonačne sume traje malo duže (verovatno više nego što ste spremni da izdvojite vremena). Najviše čemu možete da se nadate je da će se aproksimacija funkcije izvesti sa dovoljnim brojem članova, ali jasno je da se moraju usvojiti kompromisi. Aproksimacija može brzo da se postigne sa nekoliko članova. Nažalost, ona nije preterano tačna. Naravno, tačnost rezultata se povećava korišćenjem većeg broja članova, mada to zahteva više truda. Na slikama 3,8b do 3.8f prikazani su grafici aproksimacija kod kojih su korišćeni 1, 3, 5, 11 i 21 član. Kao što grafici pokazuju, sa par članova jedva da se dobija sličnost sa originalnom funkcijom. Ipak, kako se broj članova povećava, grafik postaje ravniji u svakom intervalu i "skokovi" izmedu 1 i 1 se dešavaju brže.

Napominjemo da nije uvek neophodno koristiti prethodne jednačine za izračunavanje koeficijenata ai i bi. Ponekad je funkcija za koju pokušavamo da utvrdimo Furijeov red predstavljena, u stvari, nizom tačaka koje se menjaju u skladu sa vremenom, umesto da postoji konkretna formula. Kada se položaj tačaka menja u skladu sa vremenom, kažemo da je funkcija defmisana u vremenskom domenu. U takvim slučajevima, koeficijenti Furijeovog reda mogu da se aproksimiraju (mada često sa manjom greškom) korišćenjem numeričkog integralnog računa. To uključuje izračunavanje *diskretne Furijeove transformacije* (ponekad se naziva *konačna Furijeova transformacija)*. Rezultujuća funkcija može da se interpretira kao funkcija frekvencija koje se javljaju u Furijeovim redovima. Tada kažemo da je funkcija definisana u frekventnom domenu.

Ako funkcija ima veliki broj tačaka, izračunavanje diskretne Furijeove transformacije često uključuje ponavljanje nekih izračunavanja. Druga tehnika, nazvana brza Furijeova transformacija, izvodi prethodni oblik Furijeovih redova u sumu eksponencijalnih đanova sa složenim brojevima i koristi neke dobro poznate matematičke rezultate za efikasnije izračunavanje. Da bi sve bilo još kompliko vanije, analitičari signala zavise i od *inverzne brze Furijeove transformacije.* Ideja je da se koriste brze *Furijeove transformadje* i njihovi inverzni duplikati za transformacije između vremenskog i frekventnog domena za odredene klase funkcija. Ovo je značajno jer su analitičarima signala najbitniji poznavanje frekvencije i razumevanje načina na koji šum utiče na signale.

Detalji su očigledno isuviše složeni i prelaze predviđeni obim ove knjige. Međutim, hteli smo da se ovde osvrnemo i na ovaj aspekat radi kasnije rasprave o DSL-u, kada se ponovo pozivamo na inverzne brze Furijeove transformacije. Detaljna objašnjenja možete da pročitate u referencama [Ge99] i [Ch02].

Primene Furijeovih rezultata

Furijeovi rezultati imaju suštinski značaj za proučavanje komunikacija. Prenos složenog analognog signala preko medijuma sa ograničenim opsegom signala može da se posmatra kao aproksimiranje funkcije korišćenjem nekih članova razvoja Furijeovog reda. Ovaj princip možemo da iskoristimo za objašnjenje zašto se, na primer, kada slušamo nečiji CD preko telefona, drugačije čuje nego kada se direktno sluša.

Kvalitetna oprema može da reprodukuje zvukove u okviru opsega od nekoliko desetina hiljada Hz; naravno, konkretni opseg zavisi od same opreme. Može da proizvodi zvukove od oko 30 Hz (ciklusa u sekundi) do 20.000, ili 30.000 Hz. Sa druge strane, telefon može da prenosi signale samo izmedu 300 i 3.300 Hz. Zbog toga, originalni signal gubi veoma niske i veoma visoke frekventne komponente. Sa aspekta čujnosti, gube se duboki basovi i visoki zvukovi, tako da zvuk gubi na čistoći. Furijeovi rezultati objašnjavaju i zašto glas neke osobe ne zvuči identično preko

telefona i uživo. Sa druge strane, normalni glas nema opseg tonova koje ima muzički instrument - većina frekvencija glasa pripada opsegu signala koji se prenose preko telefona. Iako postoji neki gubitak u kvalitetu tona, prenosi se dovoljno frekvencija koje omogućavaju prepoznavanje glasa i onoga što se govori.

Furijeovi rezultati se koriste i za definisanje hardvera. Na primer, **filter** blokira odredene frekvencije, a ostale propušta. Filteri imaju široki opseg primene. Na primer, ekvilajzer, koji se nalazi u stereo uredajima, može da se podesi da izbacuje odredene muzičke tonove. Ako želimo da istaknemo basove, ili visoke zvukove, kao što su sopran, ili zvuci flaute, pri vrhu muzičke lestvice, možemo da postavimo ekvilajzer da menja frekvencije koje filter blokira.

Drugi primer je kablovska televizija. Neupućeni kupac može da se pita kako je moguće da televizor primi više od 100 različitih kanala. Odgovor leži u mogućnosti da se složeni signal posmatra kao više jednostavnijih. Svakom kanalu je dodeljen odredeni frekventni opseg i signal definiše zvuk i sliku, koristeći frekvencije iz tog opsega. Fizički kabl prenosi jedan signal koji predstavlja sumu signala za sve kanale. Ovaj proces, nazvan **multipleksiranje,** detaljnije ćemo predstaviti u Poglavlju 4. Selektovanjem kanala na televiziji jednostavno omogućavate prolazak frekvencija iz odredenog opsega. Kola u televizoru ih analiziraju i stvaraju zvuk i sliku.

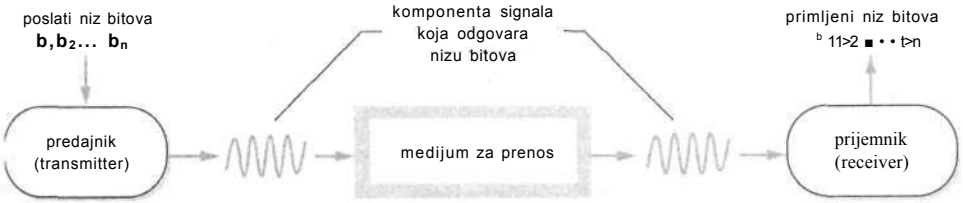
Sledeć primer je DSL tehnologija, koju ćemo opisati nešto kasnije u ovom poglavlju. Videćete da uobičajeni metod povezivanja kompjutera na Internet uključuje generisanje više signala unutar definisanih frekventnih opsega, njihovo kombinovanje i korišćenje brze Furijeove transformacije za generisanje rezultata.

3.4 Bitska brzina

Nikvistova teorema i bešumni kanali

Sledeći korak u raspravi o signalima odnosi se na povezivanje sa transferom bitova. Kompjuterske mreže danas za tu svrhu koriste sve oblike prenosa. Kako se potrebe i kapaciteti mreža neprestano razvijaju, moraju se dati odgovori na osnovna pitanja. Na primer: sa konkretnim medijumom, koliko bitova je moguče preneti u jedinici vremena? Za opisivanje kapaciteta medijuma koristi se bitska brzina **(bit rate),** koja se izražava brojem bitova u sekundi (bps). Značajan rezultat u teoriji komunikacija tiče se veze izmedu bitske brzine i opsega signala. Prosto rečeno, visokopropusni medijum može da podrži veće bitske brzine. Relađja između njih je toliko tesna da mnogi ljudi ove termine smatraju sinonimima.

Pre opisivanja relacije, morate da razumete mehanizam koji omogućava transfer bitova. Na slici 3.9 ilustrovane su glavne komponente. U osnovi, predajnik šalje signal koji predstavlja niz bitova. Prijemnik "osluškuje" medijum i kreira niz bitova na osnovu signala koji prima.



SLIKA 3.9 *Slanje podataka preko signala*

Pogledajmo signal malo pažljivije, Najpre predstavljamo niz bitova sa *b1 b2 ... bn.* Predajnik naizmenično analizira svaki niz, pa prenosi komponentu signala koja je na jedinstven način određena vrednostima bitova. Kada se komponenta pošalje, predajnik uzima sledeći niz bitova i ponavlja proces. Različite komponente signala čine stvarni preneti signal. Frekvencja kojom se komponente menjaju naziva se brzina bauda (baud rate); ona označava broj radnih reči koje se prenesu u jednoj sekundi.

Precizan način na koji predajnik utvrduje svaku komponentu predstavlja temu u narednom odeljku (ovde nije bitan). Ako želite neke konkretnije informacije, samo zamislite da postoji jedinstvena amplituda signala za svaku kombinaciju bitova. Na primer, komponente signala mogu da imaju najviše 2" različitih amplituda, po jednu za svaku kombinaciju vrednosti *bl b2 ... bn.*

Na prijemnom kraju proces je invertovan. Prijemnik naizmenično sempluje dolazeći signal i generiše niz bitova. Niz bitova zavisi od uzorka. Da bi ovaj proces ispravno funkcionisao, prijemnik mora da ima mogućnost semplovanja na frekvenciji koja odgovara brzini bauda (ako se semplovanje vrši ređe nego što se komponenta menja, neke komponente neće biti semplovane i doći će do gubljenja podataka).

Dakle, bitska brzina zavisi od frekvencije sa kojom se komponenta može menjati (brzina bauda) i *n,* broja bitova u stringu. Mnogi ljudi često naizmenično koriste termine brzina bauda i bitska brzina. Na osnovu naše rasprave sada vidite da ova ideja nije tačna. U stvari,

bitska brzina = brzina bauda x n

Ovo može da znači da je uvek moguće povećati bitsku brzinu povećanjem brzine bauda, ili broja bitova u stringu (n). Ovo je tačno, ali samo do odredene tačke. Neki klasični rezultati postavljaju gornju granicu za brzinu prenosa podataka.

Prvi rezultat je začudujuće star - potiče još iz dalekih 20-ih godina prošlog veka, kada je Heri Nikvist (Harry Nyquist) razvio svoju klasičnu teoriju (u referencama [Wa98] i [B199] možeteda pročitate formalniji opis). Ovde se nećemo baviti njegovom teorijom, ali ćemo je navesti i objasniti njen značaj za razmenu podataka. Prvo, Nikvist je pokazao da prijemnik, ako je f maksimalna frekvencija koju medijum može da podrži, može u potpunosti da rekonstruiše signal, semplujući ga 2f puta u sekundi (moramo da napomenemo da je Nikvist pretpostavio da nema šuma i da nikava izobličenja ne menjaju signal, tj. pretpostavio je da se koristi kanal bez ikakvih šumova; ukratko ćemo predstaviti i kanale sa šumovima). Drugim rečima, prijemnik može da rekonstruiše signal semplovanjem u intervalima od 1/(2f) sekundi, ili sa dva puta većom periodom (zapamtite, jedna perioda = 1/f).

Na primer, ako je maksimalna frekvencija 4.000 Hz, prijemnik mora da sempluje signal 8.000 puta u sekundi. Drugim rečima, signal može u potpunosti da se rekonstruiše semplovanjem svakih 1/8000-tih delova sekunde.

Pretpostavimo sada da predajnik menja signale u intervalima 1/(2f). Drugim rečima, brzina bauda je 2f. Zatim, imamo rezultate Nikvistove teoreme koja kaže da je

$$\text{bitska brzina} = \text{brzina bauda} \times n = 2 \times f \times n$$

U nekim knjigama u Nikvistovoj teoremi koristi se broj različitih komponenata signala, umesto broja *n*. Drugim rečima, ako je B broj različitih komponenata, onda je

$$B = 2^n$$

ili, ekvivalentno

$$n = \log_2(B)$$

U takvim shičajevima možemo da zapišemo

$$\text{bitska brzina} = 2 \times / \times \log_2(B)$$

U tabeli 3.1 dati su neki rezultati postignuti uz pretpostavku da je maksimalna frekvencija 3.300 Hz aproksimirana gornja granica za telefonski sistem.

Kanali sa šumovima

Prema do sada predstavljenim informacijama, izgleda kao da ne postoji gornja granica za bitsku brzinu na odredenoj maksimalnoj frekvenciji. Nažalost, to nije tačno. Prvo, veći broj komponenata signalaznači i suptilnije promene izmedu njih. Na primer, pretpostavimo da amplituda signala mora da bude manja, ili jednaka 5 volti i da je svaka komponenta odredena samo amplitudom. Ako koristimo dve komponente defmisane sa 2,5 i 5 volti, signali se razlikuju za 2,5 volti. Medutim, ako se korisli 16 komponenata signala, neophodna je razlika od oko jedne trećine volte izmedu susednih amplituda. Prijemnik mora da bude sofisticiraniji (i skuplji) da bi mogao da detektuje manje razlike.

Tabela 3.1: Rezultati Nikvistove teoreme za maksimalnu frekvenciju od 3.300 Hz

N, broj bitova u okviru komponente signala	B, broj komponenata signala	Maksimalna bitska brzina (bps)
1	2	6.600
2	4	**13.200**
3	8	19.800
4	16	26.400

Ako su razlike isuviše male, uredaj možda neće moći da detektuje sve promene.

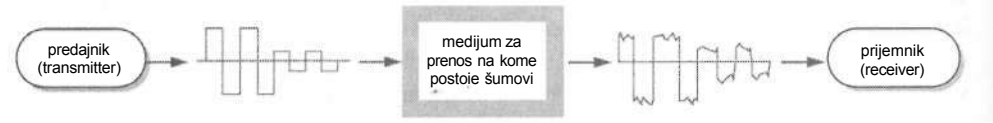
Drugi razlog nastaje zbog toga što su mnogi kanali podložni uticaju šuma, što znači da se preneti signal može izobličiti. Ako je izobličenje veliko, prijemnik ne može da rekonstruiše signal. Na primer, razmotrite digitalni signal sa slike 3.10a (uzimamo primer digitalnog signala, jer je jednostavniji za ilustrovanje; slična diskusija može da se povede i o analognim signalima). Predajnik šalje dva signala - oba osciluju izmedu dva naponska nivoa. Medutim, preneti signal je podložan uticaju šuma i primljeni signal se razlikuje od njega. Izobličenje nije mnogo veliko, tako da primljeni signal i dalje jasno defmiše dva naponska nivoa. Zbog toga, nije teško rekonstruisati primljeni signal.

Na slici 3.10b prikazana je slična situacija, osim što se u ovom slučaju originalni naponski nivoi razlikuju. Kada se šum pojavi, izobličeni signali imaju preklopljene naponske nivoe i zato je teško, ako ne i nemoguće, rekonstruisati originalni signal na osnovu primljenog.

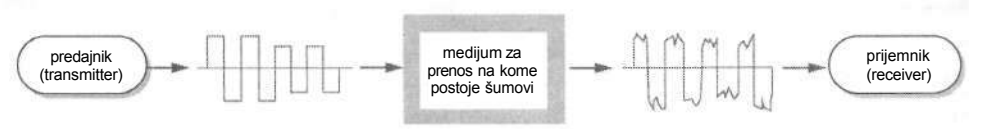
Šenonov rezultat

Naučili smo da šum može da promeni i eventualno izobliči poslate informacije. Mogućnost rekonstruisanja informacija zavisi od jačine šuma. Na primer, mali statički elektricitet neće imati mnogo uticaja na prenos signala sa radio tornja čija snaga iznosi 50.000 vati. Medutim, uticaj munje je daleko ozbiljniji. Naravno, razliku odreduje jačina šuma u odnosu na preneti signal.

Inženjeri koji se bave elektronikom koriste parametar pod nazivom koeficijent signal-šum, na osnovu koga se određuje stepen uticaja šuma na signal. Defmiše se kao S/N, gde je S jačina signala, a N jačina šuma. Ovaj parametar možete da vidite i u specifikacijama audio opreme, gde se koristi kao mera "čistoće" zvuka. Velika vrednost koeficijenta ukazuje na "čisti" signal; mala vrednost ukazuje na izobličeni signal.



(a) Velika razlika u vrednostima napona



(b) Mala razlika u vrednostima napona

SLIKA 3.10 *Efekat šuma na digitalne signale*

Kod visokokvalitetne opreme visoka vrednost koeficijenta signal-šum ukazuje na visokokvalitetni zvuk (iako u nekim slučajevima evidentno povećanje kvaliteta ne može čujno da se registaije). Pošto je S obično veće od N, koeficijent se često skalira logaritamski i izražava se kao

$$B = \log_{10}(S/N) \text{ belova}$$

Bel je jedinica mere. Na primer, ako je *S* 10 puta veće od N (S = 10 □ N), onda je B = $\log_{10}$[(10 □ N)/N]. = $\log_{10}$(10) = 1 bel. Slično tome, S = 100N daje 2 bela, S = 1000 □ N daje tri bela i tako dalje.

Većini je verovatno bliži termin decibel (dB). Defmiše se kao 1 dB = 0.1 bel. Da biste bolje razumeli šta ovo znači sa stanovišta S i N, pogledajmo sledeći primer. Uzmite zvuk jačine 25 dB. To je isto što i 2,5 bela i znači B = $\log_{10}$(S/N) = 2.5. Na osnovu ove jednačine, *S/N* treba da bude jednako 10^{25}, ili ekvivalentno

$$S = 10^{25} \times N = \sqrt{10} \times N \approx 316N$$

Klod Šenon (Claude Shannon) se 40-ih godina prošlog veka nadovezao na Nikvistovu teoremu i proučavao kanale sa šumovima. Pokazao je da maksimalna bitska brzina ne zavisi samo od maksimalne frekvencije, već i od koeficijenta signal-šum. Pokazao je da važi sledeća formula

$$\text{bitska brzina} = \text{opseg signala} \square \log_2(1 + S/N) \text{ belova}$$

Formula pokazuje da veći opseg signala i koeficijent signal-šum omogućavaju veće bitske brzine. Ako se jačina šuma poveća, maksimalna bitska brzina se smanjuje. Ideja ove relacije je da šum, ako je koeficijent signal-šum isuviše mali, može da dovede do identične reprezentacije različitih signala.

Za ilustraciju koristimo primer praktične gornje granice za prenos podataka preko telefonske linije. Telefonski sistem ima opseg signala otprilike od 3.000 Hz, a koeficijent signal-šum od oko 35 dB, ili 3,5 bela. Na osnovu ove relacije, sledi da je 3.5 = $\log_{10}$(S/N), ili S = 10^{35} □ N ≈ 3162N. Na osnovu ovih vrednosti, prema Senonovim rezultatima, važi sledeće:

$$\begin{aligned}&\text{bitska brzina} = \text{opseg signala} \square \log_2(1 + S/N)\\ &= 3000 \square \log_2(1 + 3162) \text{ bps}\\ &\approx 3000 \square 11.63 \text{ bps}\\ &\approx 34{,}880 \text{ bps}\end{aligned}$$

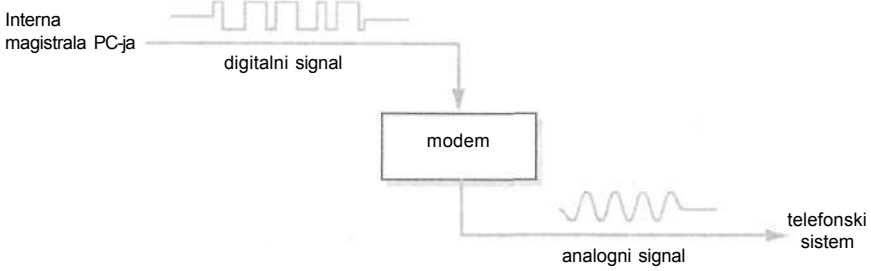
Kao fmalnu napomenu, ističeno da ovo nije "čisto" teorijski rezultat, kojim ne bi trebalo preterano da se bave korisnici mreža i kompjutera. Ovaj rezultat ima stvarne implikacije na korisnike modema. Tokom 80-ih godina prošlog veka 2400 i 9600 bps modemi su postali uobičajeni uredaji. Postojali su i modemi sa veđm bitskim brzinama, ali su bili prilično skupi. Početkom i sredinom sledeće decenije videli smo modeme koji su mogli da prenose podatke sa 28,8 i 33,6 Kbps, a ubrzo su se na sceni pojavili i 56 Kbps modemi. Prema Šenonovim rezultatima, bitska brzina od oko 35.000 bps predstavlja gomju granicu za konvencionalne modeme; proizilazi zaključak da brzina od 56 Kbps namšava verodostojnost Šenonovih rezultata. Ipak, to nije slučaj, jer su Senonovi rezultati i brojke koje smo ovde koristili zasnovani na pretpostavkama koje nisu u skladu sa današnjim prilikama. Pre nego što su Internet provajderi (ISP) postali uobičajeni, ljudi su često koristili modeme za biranje broja i povezivanje na drugi modem u mreži kompanije, ili u univerzitetskoj mreži.

Zbog dodatne konverzije analognih u digitalne signale na udaljenom kraju, nivoi šuma su viši, a granica bitske brzine je blizu 35 Kbps. 56 Kbps modemi mogu da postignu veće bitske brzine kada se "vezujete" direktno na ISP. Oprema provajdera je dizajnirana uz pretpostavku da je većina telefonskih sistema digitalna i da Je, zato, moguća direktna komunikacija; nema konverzije izmedu analognih i digitalnih signala na tom kraju i zato su nivoi šuma niži. U odeljku 3.7 biće detaljnije obrađen ovaj problem.

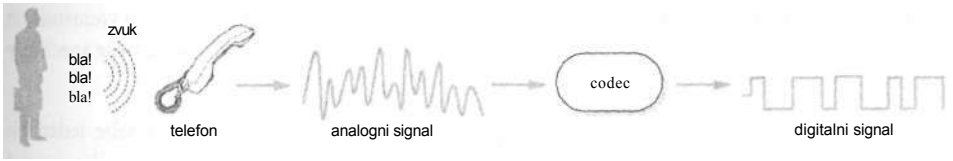
3.5 Konvertovanje digitalnih u analogne signale

U prethodnim odeljcima je opisan prenos podataka pomoću analognih i digitalnih signala. Ako se koriste isključivo analogni, ili isključivo digitalni signali, komunikacija će biti pojednostavljena i sadržaj ovog odeljka Vam neće biti potreban. Naravno, ovo pojednostavljenje je glavni razlog zašto se televizijski prenosi i kablovski signali orijentišu na digitalne signale. Medutim, u praksi postoji veliki stepen mešanja komunikacije analognih i digitalnih uredaja. Osim toga, postoje i dobri razlozi zbog kojih se ne ide na primenu samo analognih, ili samo digitalnih signala.

Do sada ste naučili da su kompjuteri digitalni uređaji. U stvari, većina kompjuterskih komunikacija, kao što je slučaj komunikacije izmedu radne stanice i servera, ili kompjutera sa diskom, koristi digitalne signale. Osim toga, lokalne mreže se obično u potpunosti oslanjaju na digitalne signale. Gde se u celu "priču" uklapaju analogni signali? Odgovor je u komunikacijama na daljinu. Mnogi Ijudi koriste personalne kompjutere u svojim domovima i komunikaciju sa ISP-om uspostavljaju preko telefonskih linija, servisa kablovske televizije, ili, rak, satelitske antene. Kao što smo već istakli, *modem* (skraćeno od modulacija/demodulacija) potreban je za konverziju različitih tipova signala. Standardni modem je uredaj koji se instalira u unutrašnjosti kompjutera i povezuje se na internu magistralu (slika 3.11). Digitalni signali iz memorije, ili fajla na disku "putuju" ka modemu preko inteme magistrale. Modem prihvata te signale, konvertuje ih u analogne i šalje ih telefonskom sistemu preko telefonske linije koja je priključena na modem. Kada se signal nade u telefonskom sistemu, tretira se kao i svi ostali govorni signali. Na odredišnom kompjuteru informacije se podvrgavaju obmutom procesu (recimo prilikom preuzimanja fajla). Analogni signali dolaze preko telefonske linije, stižu do modema i modem ih konvertuje u digitalne signale i preko interne magistrale šalje u memoriju, ili fajl na disku.



SLIKA 3.11 *Podaci iz kompjutera preneti preko telefonskih linija*



SLIKA 3.12 *Govorne informacije prenete u digitalnom obliku*

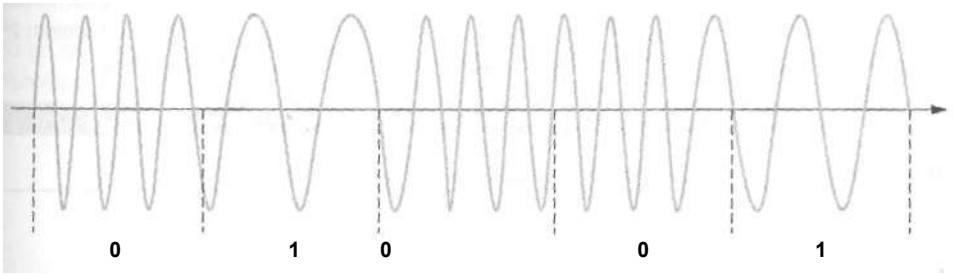
SJedeđ primer analognih uredaja koji komuniciraju koristeći digitalne signale nalazi se u samom (elefonskom sistemu. Znamo da je telefon uređaj koji konvertuje govorne zvuke u analogne signale. U starom telefonskom sistemu analogni signali su prenošeni preko žica, Oi kablova do prijemnog telefona, gde su konvertovani nazad u zvuk. Današnja fiber tehnologija je u potpunosti promenila način na koji se glas prenosi. Pošto optički fiber prenosi digitalne signale, uredaj se naziva codec (skraćeno od coder/decoder) i prevodi analogni govorni signal u digitalni ekvivalent (slika 3.12). Nakon toga se prenosi digitalni signal. U odredenoj tački se konvertuje nazad u analogni signal, tako da može da se konvertuje u zvuk na prijemnom telefonu.

If ovom i narednom odeljku čemo objasniti kako se digitalni signali konvertuju u analogne i obratno. U odeljku 3.7 predstavićemo operacije i standarde modema.

Frekventna modulacija

Konvertovanje digitalnih signala u analogne nije teško. U osnovi, sve što treba da uradimo je da pridružimo grupu vrednosti jednog, ili više bitova određenom analognom signalu. U odeljku 3.3 opisana su tri načina za promenu analognog signala: na osnovu frekvencije, amplitude, ili faznog pomaka.

Jednostavni metod konvertovanja frequency shift keying (FSK), koji se naziva i frekventna modulacija (FM), pridružuje 0 jednoj, a 1 drtigoj analognoj frekvenciji. Na primer, ako 0 odgovara višoj frekvenciji, a 1 nižoj, na slici 3.13 prikazan je analogni signal koji se dobija za string 01001.



SLIKA 3.13 *Pomeranje između frekvencija (dve frekvencije), jedan bit po baudu*

Za svaki bit modem može da prenosi signal odgovarajuće frekvencije u odredenom vremenskom periodu. Period, a samim tim i broj ciklusa, promenljiv je (u odeljku 3.7 date su neke specifične vrednosti za određene modeme).

Korišćenje dve frekvencije podrazumeva da se sa svakom promenom signala šalje jedan bit podataka. To je situacija u kojoj su brzina bauda (koliko se često menjaju karakteristike signala) i bitska brzina slične. Alternativni oblici frekventne modulacije rnogu da koriste više frekvencija. Na primer, pošto dva bita mogu da formiraju jednu od dve moguće kombinacije, svakom paru bitova možemo da pridružimo jednu od četiri frekvencije. Dakle, svaka promena frekvencije predstavlja dva bita podataka - bitska brzina je dva puta veća od brzine bauda.

U opštem slučaju, n bitova može da formira 2n kombinacija i svakoj se može pridružiti jedna od 2n frekvencija. U ovom slučaju je bitska brzina n puta veća od brzine bauda.

Amplitudska modulacija

Metod amplitude shift keying (ASK), poznat i kao **amplitudska modulacija (AM),** liči na FSK metod. Razlika je, kao što ste mogli očekivati, u tome što se svakoj grupi bitova pridružuje analogni signal sa određenom amplitudom. Osim toga, kao i kod FSK metoda, u grupi bitova može da se nadu jedan, dva, ili više bitova, čijim se pridruživanjem definiše relacija izmedu bitske brzine i brzine bauda.

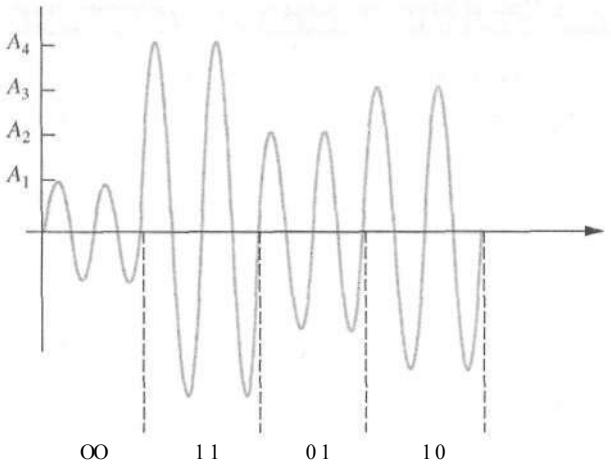
Da bismo ovo ilustrovali, pretpostavićemo da imamo četiri amplitude: A_1, A_2, A_3 i A_4. Koristeći ove oznake, u tabeli 3.2 smo prikazali način pridruživanja bitova svakoj amplitudi. Na slici 3.14 prikazan je analogni signal za string 00110110. U ovom slučaju je bitska brzina dva puta veća od brzine bauda. Oba bita (počevši od levog) definišu signal sa odgovarajućom amplitudom. Kao i kod promene frekvencije, signal se prenosi u okviru fiksnog vremenskog perioda.

Fazna modulacija

Metod phase shift keying (PSK), poznat i kao fazna modulacija (PM), podseća na prethodne tehnike. Signal se razlikuje po faznom pomaku, umesto po frekevenciji, ili amplitudi. Tipično, fazni pomak signala se meri u odnosi na prethodni signal. U takvim slučajevima se često koristi termin **differential phase shift keying (DPSK).** Kao i ranije, sa n bitova moguće je formirati 2^n faznih pomaka, tako da je kod ove tehnike bitska brzina n puta veća od brzine bauda.

Tabela 3.2: Pridruživanje signala kod amplitudske modulacije

Vrednosti bitova	Amplituda generisanog signala
00	A1
01	A2
10	A3
11	A4



SLIKA 3.14 *Amplitudska modulacija (četiri amplitude), dva bita po baudu*

Kvadraturna amplitudska modulacija

Sve prethodno predstavljene tehnike mogu da se koriste za različite signale. Velika raznolikost u karakteristikama signala znači veću bitsku brzinu na određenoj brzini bauda. Problem je što veće bitske brzine zahtevaju legitimnije signale, čime se redukuju razlike između njih. Kao što je prikazano u prethodnom odeljku, ovo stvara poteškoće, jer je neophodna oprema koja može da registruje razlike izmedu signala čije se frekvencije, amplitude, ili fazni pomaci razlikuju samo malo. Osim toga, šum može da izobliči signale, tako da se jedan legitimni signal transformiše u drugačiji, takode legitimni signal, što dovodi do prenosa pogrešnog stringa.

Cesto se koristi kombinacija promene frekvencije, amplitude, ili faznog pomaka, tako da je moguće koristiti veću grupu legitimnih signala dok se izmedu njih stvaraju veće razlike. Najčešće se koristi tehnika **kvadraturne amplitudske modulacije (QAM),** kod koje se grupe bitova dodeljuju signalu definisanim svojom amplitudom i faznim pomakom.*

Na primer, pretpostavimo da koristimo dve različite amplitude i četiri različita fazna pomaka. Njihovom kombinacijom može da se definiše osam različitih signala. U tabeli 3.3 prikazana je relacija izmedu vrednosti stringova sa tri bita i signala.

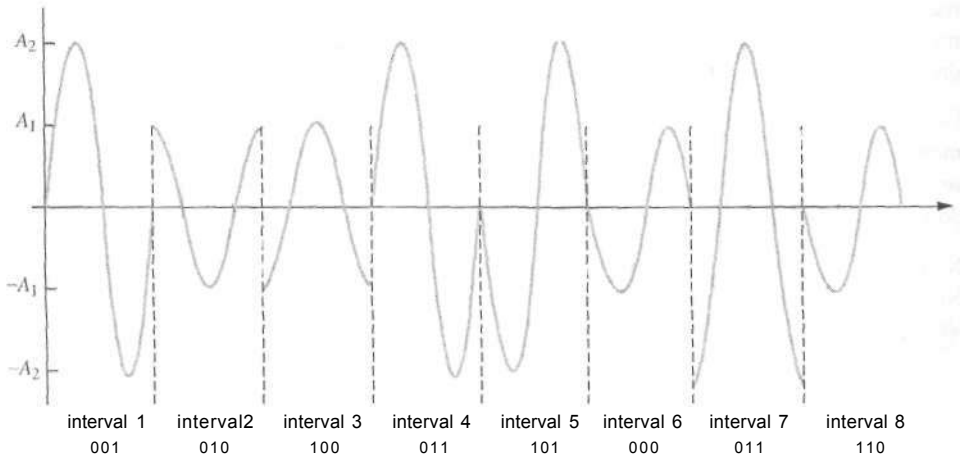
* Inženjeri možda neće prihvatiti ovu definiciju. Kvadraturno amplitudski modulisani signali kreirani su dodavanjem dva analogna signaia iste frekvencije. Jedan signal odgovara sinusnoj funkciji, a drugi kosinusnoj (ove dve funkcije se razlikuju za ugao od 90° i odatle termin kvadraturna). To znaći da je signal oblika C □ sin(x) + D □ cos(x). Promenljiva □ se menja u vremenu u zavisnosti od frekvencije signala, dok C i D zavise od inicijalnog signala. Medutim, trigonometrija pokazuje da C sin(x) + D □ cos(x) može da se zapiše i kao A □ sin(x+P}, gde je A = $\sqrt{C^2+D^2}$ P = arcsin ($C/\sqrt{C2 + D2}$). Zato c'emo, za naše potrebe, smatrati da signal ima promenljivu amplitudu i fazni pomak.

Tabela 3.3: Pridruživanje signala kod kvadraturne amplitudske modulacije

Vrednosti bitova	Amplituda generisanog signala	Fazni pomaci generisanog signala
000	A_1	0
001	A_2	0
010	A_1	$1/(4f)$
OU	A_2	$1/(4f)$
100	A_1	$2/(4f)$
101	A_2	$2/(4f)$
110	A_1	$3/(4f)$
111	A_2	$3/(4f)$

Amplitudu definišemo kao Al i A2, a fazne pomake kao 0, 1/(4/), 2/(4/) i 3/(4/), gde je / frekvencija. Pomaci odgovaraju jednoj četvrtini, dve četvrtine i tri četvrtine periode, respektivno.

Na slici 3.15 prikazana je promena signala do koje dolazi prilikom prenosa niza bitova 001-010-100-011-101-000-011-110 (crtice smo naveli radi bolje preglednosti; one nisu deo prenosa). Da biste razumdi zašto signal izgleda ovako, pažljivo pročitajte ovo što sledi. Prva tri bita, 001, definišu signal sa amplitudom A_2 i faznim pomakom 0. Zbog toga, na osnovu rasprave iz prethodnog odeljka, signal startuje na 0 volti i oscilira izmedu vrednosti A_2 i $-A_2$.



SLIKA 3.15 *Kvadraturna amplitudska modulacija (dve amplitude i četiri faze), tri bita po baudu*

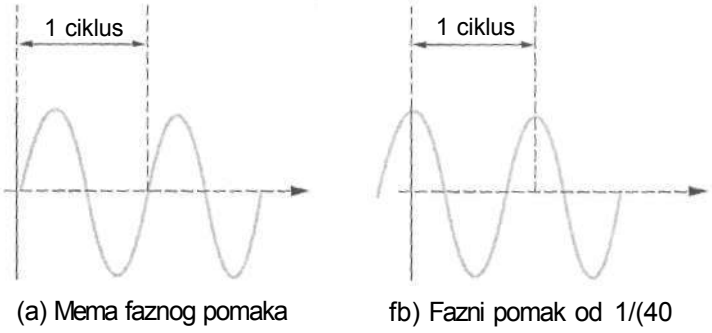
Kao i ranije, broj ciklusa zavisi od frekvencije i dužine ukupnog intervala u kome se signal prenosi. Mi smo, radi bolje preglednosti, nacrtali samo jedan ciklus.

Sledeća tri bita, 010, definišu signal sa amplitudom Al i faznim pomakom 1/(4f). Dakle, kao što može da se vidi na slici 3.15, signal oscilira izmedu A_1 i $-A_1$. Da nema faznog pomaka, signal bi startovao od 0 i rastao bi do A_1. Medutim, kao što je opisano u prethodnom odeljku, pozitivni fazni pomak odgovara levom horizontalnom pomaku na grafiku. Da biste to lakše razumeli, na slici 3.16 smo ilustrovali grafik bez faznog pomaka i (b) sa faznim pomakom od 1/(4f).

Možda će Vam razumevanje grafika sa slike 3.16 biti lakše ako se setite da je p = 1/f, gde je *p* perioda signala. Drugim rečima, 1/(4f) odgovara jednoj četvrtini periode, tako da Je grafik sa slike 3.16b pomeren ulevo u odnosu na sliku 3.16a za jednu četvrtinu periode. Zato signal počinje od maksimuma, opada do minimuma i ponovo raste do maksimuma. Zbog toga izgleda da je prva četvrtina periode, u kojoj signal raste od 0 do maksimuma, odsečena. Ovaj fenomen je upravo ono što možete da vidite na drugom intervalu na slici 3.15.

Treći skup bitova, 100, defmiše signal sa amplitudom A_3 i faznim pomakom 2/(4/). Pre nego što objasnimo šta ovo znači, proučite signal na kraju drugog intervala. Trenutno se nalazi u maksimumu A_1. Da nije bilo faznog pomaka, signal bi jednostavno startovao u A1 i nastavio da opada sve do $-A_1$. Ali, fazni pomak od 2/(4f) znači da je eleminisana polovina periode. Pošto je prethodni signal završen u maksimumu, polovina periode odgovara delu u kome signal opada od A_1 do $-A_1$. Zato signal na početku trećeg intervala startuje od svoje minimalne vrednosti.

Sada ćemo dati opštije objašnjenje načina na koji se signal generiše na osnovu grupe od tri bita. Signal zavisi od toga gde je završen signal prethodne grupe od tri bita. Fazni pomak se defmiše relativno u odnosu na tu tačku. U tabeli 3.4 novi signal se definiše kao funkcija faznog pomaka i pozicije prethodnog signala. Imajte na umu da se maksimum i minimum u prvoj koloni odnose na prethodni signal, doksu maksimum i minimum za tekući signal dati u drugoj koloni.



SLIKA 3.16 *Efekat faznog pomaka na signal*

Tabela 3.4: Pravila za definisanje signala korišćenjem kvadraturne amplitudske modulacije

Pozicija prethodnog signala	Bez faznog pomaka	Fazni pomak od 1/4 periode	Fazni pomak od 2/4 periode	Fazni pomak od 3/4 periode
U 0, rast	Start u 0, rast	Start u maksimumu	Start u 0, opadanje	Start u minimumu
U maksimumu	Start u maksimumu	opadanje Start u 0,	Start u minimumu	Start u 0, rast
U 0, opadanje	Start u 0, opadanje	Start u minimumu	Start u 0, rast	
U minimumu	Start u minimumu	Start u 0, rast	Start u maksimumu	Start u 0, opadanje

Pokazaćemo kako se ova tabela primenjuje kod definisanja signala u četvrtom intervalu sa slike 3.15. Pozicija prethodnog signala (u intervalu 3) odgovara minimumu. Osim toga, bitovi 011 defmišu signal amplitude A_2 i faznog pomaka 1/(4f). Signal je definisan donjim redom treće kolone: startuje u 0 i raste do svog maksimuma A_2, kao što se vidi i na slici.

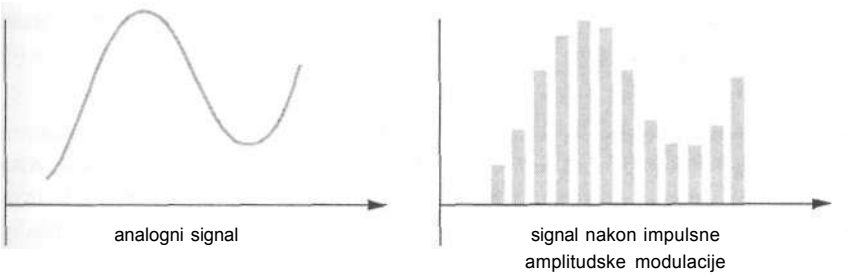
Napomenimo da trobitna vrednost ne definiše uvek isti signal. Na primer, intervali 4 i 7 odgovaraju kombinaciji 011, ali se signali razlikuju. Naravno, oba signala imaju istu amplitudu. Medutim, fazni pomak je relativan u odnosu na kraj prethodnog signala. Zbog toga, i pored činjenice da oba signala imaju isti fazni pomak 1/(4f), startuju u različitim vrednostima. Sledeće što vredi istaći je da isti signal u dva različita intervala može da odgovara različitim kombinacijama bitova. Na primer, signali u intervalima 6 i 8 su identični, ali se kombinacije njihovih bitova razlikuju. Zašto?

Ostali načini modulacije signala pomoću različitih kombinacija amplitude, frekvencije i faznog pomaka predstavljeni su u odeljku 3.7, u okviru predstavljanja standarda za rnodeme.

Kao što je istaknuto u prethodnona odeljku, veće bitske brzine mogu da se postignu pridruživanjem većeg broja bitova po jednom baudu i korišćenjem više defmicija signala. Medutim, imajte na umu da povećanje broja korišćenih signala umanjuje mogućnost detektovanja razlika izmedu njih. Ako se koristi isuviše veliki broj signala, i manji šumovi mogu da učine jedan signal identičnim sa nekim drugim. Ukoliko se to desi, prijemni modem pogrešno interpretira signal i šalje pogrešne bitove ka svom uredaju. Postoji više riačina za otklanjanje grešaka; neki mehanizmi za detektovanje i korekciju grešaka predstavljaju temu Poglavlja 6.

3.6 Konvertovanje analognih u digitalne signale

Neke konverzije analognih u digitalne signale ne predstavljaju ništa drugo nego prethodno objašnjeni proces, ali samo u obrnutom smeru. Modem proverava amplitude, frekvencije i fazne pomake dolazećih signala i u skladu sa detektovanim vrednostima generiše digitalne signale. Ovi analogni signali imaju konstantne karakteristike, bar posmatrano u kratkim intervalima. Medutim, nisu svi analogni signali takvi. Šta je sa analognim signalima koji se menjaju kontinuelno? Najočigledniji primer mogu da budu analogni signali koje stvaraju zvukovi, kao što su glas, ili muzika. Ovi signali su složeniji od onih koje generišu digitalni podaci i zahtevaju alternativne tehnike za konvertovanje.



SLIKA 3.17 *Impulsna amplitudska modulacija*

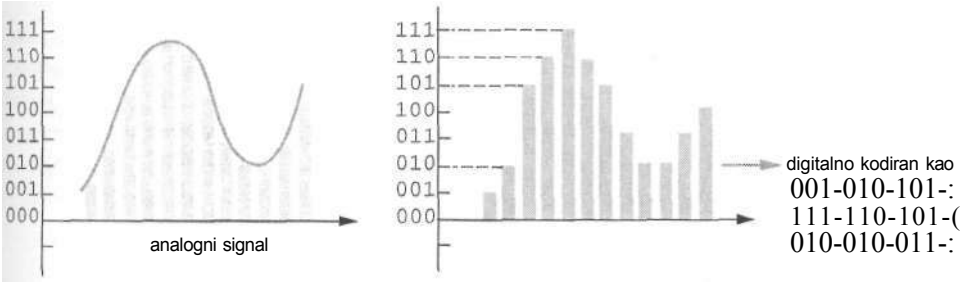
Impulsna amplitudska modulacija

Jedan od metoda za digitalizaciju analognog signala je impulsna amplitudska modulacija (PAM - pulse amplitude modulation). U okvim ovog jednostavnog procesa, analogni signal se sempluje u pravilnim intervalima, a zatim se generiše impuls sa amplitudom semplovanog signala. Na slici 3.17 prikazan je rezultat semplovanja u pravilnim intervalima.

Impulsna kodna modulacija

Signali generisani PAM modulacijom izgledaju digitalno, ali pošto impulsi mogu da imaju proizvoljne amplitude, signal i dalje ima karakteristike analognog. Jedan od načina da se ovi impulsi učine stvarno digitalnim je da se svakom semplovanom signalu dodeli amplituda iz preddefinisanog skupa. Ovaj proces se naziva impulsna kodna modulacija **(PCM - pulse code modulation).** Na primer, pretpostavimo da smo podelili amplitudski opseg u skup od 2^n amplituda **i da** je svakoj amplitudi pridružen n-bitni binarni broj. Na slici 3.18 prikazana je podela na osam vrednosti (n=3).

Kao i ranije, periodično semplujemo analogni signal. Ali, ovoga puta biramo jednu od 2^n amplituda koje najbliže odgovaraju semplovanoj amplitudi. Nakon toga se sekvenca bitova prenosi pomoću digitalnog prenosa koji se koristi u konkretnom uredaju.



SLIKA 3.18 *Impulsna kodna modulacija*

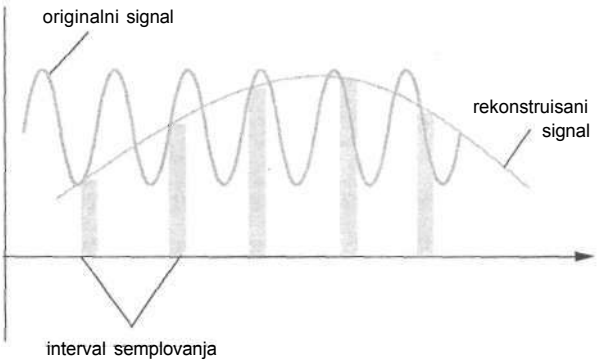
Semplovanjem u pravilnim intervalima, sa učestalošću s u sekundi, postižemo bitsku brzinu od *n x s* bitova u sekunđi. Taj proces je prikazan na slici 3.18. Prvi sempl odgovara kombinaciji 001, drugi 010 i tako redom.

Na prijemnoj strani niz bitova se deli u grupe od po *n* bitova i pristupa se rekonstrukciji analognog signala. Tačnost rekonstruisanog signala zavisi od dva faktora. Prvi je frekvencija semplovanja *s.* Semplovanje sa frekvencijom manjom od frekvencije signala može da uzrokuje propuštanje nekih oscilacija (slika 3.19). Zbog toga, rekonstmisani signal može da predstavlja slabu aproksimaciju originalnog signala. Zato je semplovanje neophodno izvesti sa frekvencijom koja obezbeđuje zadržavanje svih karakteristika originalnog signala. Ovo možda navodi na zaključak da je bolje imati veći broj semplova. Taj zaključak je tačan, ali samo u određenoj meri. Sećate se Nikvistove teoreme iz odeljka 3.4. Tamo je rečeno da je semplovanje signala sa učestalošću koja je dva puta veća od njegove frekvencije sasvim dovoljno da se sačuvaju informacije iz prenetog signala. Ovde sada imamo lepu primenu Nikvistove teoreme. Ako je maksimalna frekvencija signala f, sve što je veće od s = 2f ne obezbeduje bolju aproksimaciju nego kada se koristi s = 2f.

Drugi faktor koji utiče na tačnost jeste broj različitih amplituda koje se dodeljuju impulsima. Na slici 3.18 prikazana je podela na osam vrednosti, radi pojeđnostavljenja ilustracije. Ako postoje velike razlike izmedu semplovanog signala i impulsa, rekonstruisani signal može da bude izobličen. To se naziva *šum kvantizacije.* Redukovanjem razlika između amplituda susednih impulsa olakšava se redukovanje šuma kvantizacije; međutim, bez obzira na broj korišćenih amplituda, određeni stepen šuma je neizbežan.

Još jedna napomena: više frekvencije semplovanja i veći broj amplituda daju visokokvalitetni prenos, ali uz određenu cenu. Da bi bio prenet veći broj bitova u sekundi, potrebna je veća bitska brzina, a to košta više.

PCM ima nekoliko praktičnih primena. Jedna je digitalizacija govornih signala preko medugradskih telefonskih linija. Standard koji je prihvaćen širom sveta izvodi 8.000 semplova u sekundi i koristi osam bitova po semplu. U skladu sa Nikvistovom teoremom, ova frekvencija predstavlja nešto više od dvostruke maksimalne frekvencije glasa koju telefon može da podrži.



SLIKA 3.19 *Semplovanje sa isuviše niskom frekvencijom*

Osim toga, potrebna je bitska brzina od 8 x 8000, ili oko 64 Kbps.*

Sledeća primena je kod tehnologije za snimanje na kompakt diskovima (CD-ovima). Muzika se na CD-u kodira optički u digitalnom formatu korišćenjem PCM modulacije. Ipak, da bi se sačuvao visoki kvalitet zvuka, PCM kodiranje zahteva višu frekvenciju i veći broj bitova po impulsu. Stvarne vrednosti zavise od specifične opreme. Na primer, proverili smo uputstvo proizvodača CD plejera i videli sledeće tehničke specifikacije:

Frekvencija semplovanja: 44.1 kHz

D-A konverzija: 16-bitna linearna

D-A se, kao što ste mogli da pretpostavite, odnosi na konverziju digitalnih u analogne signale. Sesnaest bitova omogućava aproksimativno 64.000 semplovanih amplituda. Frekvencija semplovanja od aproksimativno 44.000 semplova u sekundi nešto malo je veća od dvostruke navedene frekvencije iz opsega od 20 do 20.000 Hz. Termin *linearna* označava da se amplitude impulsa ravnomerno rasporeduju.t

Postoje i druge tehnike modulacije, ali ovde se nećemo baviti proučavanjem tih metoda. Na primer, kod modulacije za promenljivim trajanjem impulsa informacije se kodiraju definisanjem impulsa sa različitim trajanjem. Diferendjalna impulsna modulacija meri razlike u sukcesivnim semplovima. Delta modulacija predstavlja varijaciju diferencijalne impulsne kodne modulacije koja koristi samo jedan bit po semplu. Za više informacija o ovim i drugim metodima modulacije pogledajte referencu [B199].

3.7 Modemi

Pošto smo razmotrili kako se vrši konverzija između analognih i digitalnih signala, možda ste pomislili da smo završili "priču" i da je sada dovoljno samo priključiti modem u kompjuter i povezati se sa Internet provajderom. To i jeste i nije tačno. Početkom 80-ih godina prošlog veka, kada su se pojavili personalni kompjuteri, mnogi ljudi su požurili da budu medu prvima koji će uživati u mogućnostima tih novih i moćnih alatki. Morali su, pri tom, da nauče da koriste softver, što nije bilo jednostavno za početnike (ponekad nije bilo jednostavno ni profesionalcima).

Osim toga, mnogi ljudi koji su uspeli da se "snađu" sa softverom otkrili su da su njihovi prijatelji i kolege kupili drugačije kompjutere i da su naučili da koriste drugačiji softver. Ove razlike su deljenje podataka i komunikaciju učinile skoro nemogućim. Ključne reči koje se primenjuju na modeme su *softver* i *kompatibilnost.* Tehnike modulacije koje smo ranije opisali objašnjavaju kako se digitalni signali menjaju u analogne. Medutim, da bi signali uopšte stigli do modema i da bi se dobili digitalni na osnovu telefonskih signala, potreban je softver.

* U praksi, optički fiber se koristi za povezivanje udaljenih lokacija sa mnogo većim bitskim brzinama, jer može da prenese više telefonskih razgovora istovremeno, zahvaljujući multipleksiranju. U odeljku 4.5 možete da prodtate detaljnija objašnjenja.

t U nekim slučajevima, kao kod telefonskog sistema, amplitude se ne rasporeduju ravnomerno: pre su to amplitude impulsa iz opsega iz koga se vrednosti najverovatnije javljaju. Ovakva neravnomema distribucija naziva se *Uompandovanje,* a može da poboljša kvalitet zvuka bez neophodnog povećavanja broja bitova u okviru sempla.

Srećom, kompjuteri se danas kupuju sa instaliranim modemom, tako da je uz njih uključen i softver za modem.

Pretpostavimo da u svom kompjuteru imate modem i odgovarajući softver za njega. Kada modem primi analogne signale, mora da zna kako su modulisani. Slično tome, kada modulira signale, mora da koristi šemu koju udaljeni sajt može da razume. Ako udaljeni sajt ne prepoznaje metode koje je modem koristio za moduladju digitalnog signala, sajtovi neće moć da komuniciraju. Za to je neophodna kompatibilnost.

Srećom, standardi kojih se pridržavaju proizvodači modema definišu bitsku brzinu, brzinu bauda i šemu modulacije. Najpoznatiji standardi, koje defmiše ITU-T, obično se identifikuju sa V.xx, gde je □□ identifikacioni broj. Postoje i AT&T, ili Bellovi modemi koji koriste metode slične onima koje defmišu ITU-T standardi.

Počećemo opisivanjem nekoliko starijih, mada jednostavnijih standarda i nastaviti sa standardima koji se danas najčešće koriste. ITU-TV.21 modem modulira signale koristeći frekventou modulaciju. Jedan bit defmiše frekvenciju; zato su bitska brzina i brzina bauda jednake (300, što je prilično sporo za današnje standarde).

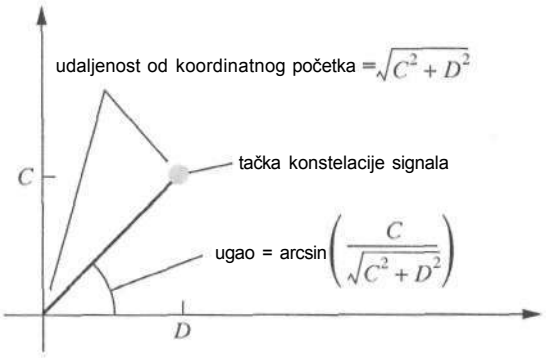
Dodela frekvencije zavisi od toga da li modem inicira *(mod iniciranja),* ili prima *(mod odgovora)* poziv. Ako je modem u modu iniciranja poziva, šalje 0 pomoću frekvencije 980 Hz, a 1 sa 1180 Hz. Ako je u modu odgovora na poziv, 0 odgovara frekvencija od 1.650 Hz, a 1 frekvencija od 1.850 Hz. Korišćenjem dva seta frekvencija omogućena je full-duplex komunikacija, tj. dvosmerna komunikacija.

Ako je brzina bauda 300, trajanje signala je $1/300 \approx 0.0033$ sekunde. U ranim danima kompjuterskih komunikacija relativno dugačko trajanje je umanjivalo uticaj šuma na signal. Ako je i postojalo neko izobličenje, preostajali su dovoljan deo signala koji može da se prepozna i (prema današnjim standardima) nesofisticirani modem. Današnji sofisticirani uredaji koriste kraće signale, na osnovu čega je omogućeno povećanje bitske brzine i brzine bauda.

AT&T modem slično funkcioniše. Koristi 1.070 Hz za 0 i 1.270 Hz za 1 u modu iniciranja poziva, a 2.025 Hz za 0 i 2.225 Hz za 1 u modu odgovora na poziv. Sledeći standard je V.22 modem. Koristi faznu modulaciju, pridružujući dva bita svakom faznom pomaku. Ima brzinu bauda 600 i bitsku brzinu 1.200. Frekvencija i amplituda su konstantne.

Konstelacija signala

Mnogi modemi funkcionišu tako što se menja više komponenata signala, i to obično promenom faznog pomaka i amplitude (QAM modulacija). Ovakva promena omogućava veće razlike izmedu komponenata signala, pa, samim tim, **i** prenos većeg broja bitova u sekundi. QAM metodi mogu da se opišu vizuelno pomoću **konstelacije signala,** dijagrama koji za definisanje svih legitimnih promena signala koristi tačke iscrtane u koordinatnom sistemu. Na slici 3.20 prikazano je kako se interpretira jedna tačka. Kvantifikovana je svojom dužinom (udaljenošću od koordinatnog početka) i uglom koji formira u odnosu na horizontalnu osu. Sećate se iz prethodnog odeljka da se dužina i ugao (fazni pomak) definišu pomoću promenljivih C i D, amplitude sinusne funkcije i kosinusne funkcije, koje kreira QAM signal. Svaka tačka definiše legitimnu promenu signala. Amplituda signala odgovara udaljenosti tačke od koordinatnog početka, a fazni pomak odgovara uglu prema horizontalnoj osi.

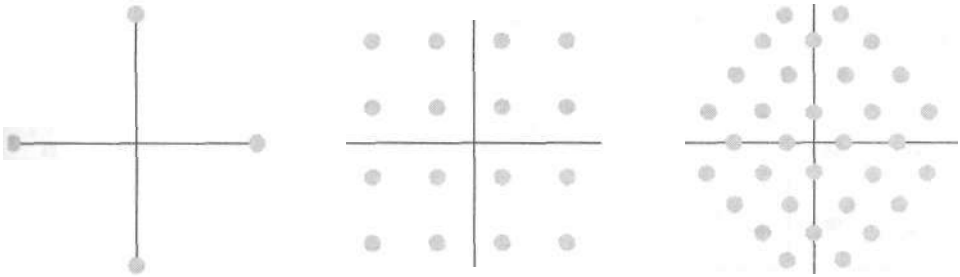


SLIKA 3.20 *Kvantifikovanje tačke na konstelaciji signala*

U opštem slučaju, uglovi kod konstelacije signala nalaze se između 0° i 360°. Ipak, prethodno smo defmisali fazni pomak kao deo periode čija se vrednost nalazi izmedu 0 i jedne periode. Da bi konstelacija signala bila ispravno interpretirana, defmišemo relaciju izmedu uglova u konstelaciji i dela periode. Specijalno, ugao x° odgovara razlomku x/360 periode. Na primer, ugao od 90° odgovara 90/360 = 1/4 periode.

Korišćenjem ovakve interpretacije na slici 3.21 je prikazana konstelacija signala za V.22 modem. Prikazane su četiri tačke sa istom udaljenošću od koordinatnog početka, što znači da se amplituda ne menja sa signalom. Osim toga, ove četiri tačke formiraju uglove od 0°, 90°, 180° i 270° sa horizontalnom osom. Na osnovu toga, legitimni fazni pomaci iznose nula, jednu četvrtinu, jednu polovinu i tri četvrtine periode.

Nešto složeniji je V.22 bis standard. Na slici 3.21 možete da vidite i konstelaciju signala sa 16 tačaka. Standard koristi brzinu bauda od 600 i četiri bita po jednom baudu, tako da je brzina prenosa podataka 2.400 bps.



SLIKA 3.21 *Konstelacije signala*

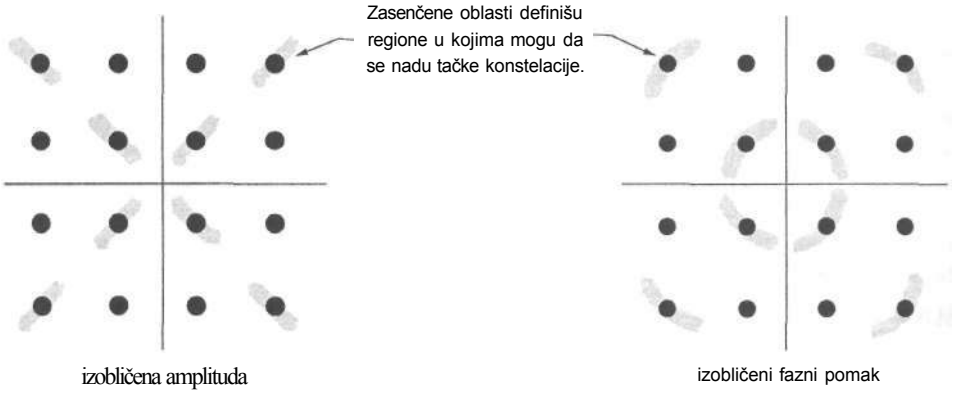
Ako pažljivije pogledate konstelaciju signala, videćete da postoje tri različite amplitude i 12 mogućih faznih pomaka. Ove slike obezbeduju 36 kombinacija, ali se koristi samo 16. Ograničenje postoji zbog mehanizma za detekciju grešaka.

Poslednja konstelacija signala sa slike 3.21 odgovara standardu V.32. To je konstelacija sa 32 tačke, kod koje se koristi brzina bauda od 2.400 i pet bitova po jednom baudu. Medutim, brzina prenosa podataka je 4x9.600 bps. Koristi se dodatni bit po baudu, jer standard koristi rešetkasto kodiranje *(trellis coding),* mehanizam za detekciju grešaka koji kreira bit pamosti definisanjem dodatnih komponenata signala.

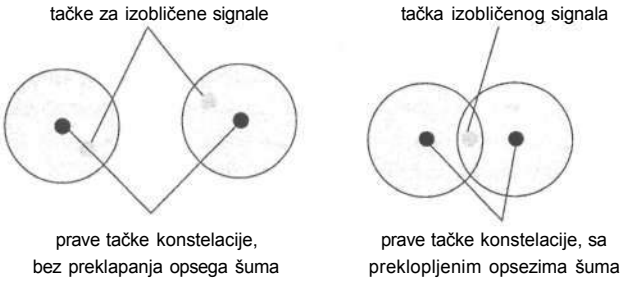
Posmatranjem ovih konstelacija signala (i drugih) možete da primetite da imaju jednu zajedničku osobinu. Izgleda da su sve tačke ravnomemo rasporedene. Ovo nije obezbeđeno radi preglednosti grafičkog prikaza. Modemi, poput nezgrapnih ljudi koji ne cene umetnost, ne vode računa o lepoti slika. Cinjenica je da se prenos signala odvija preko linija na kojima postoje šumovi. Reći da se dva signala razlikuju za 45°, ili da je amplituda jednog signala dva puta veća od amplitude drugog ispravno je samo u slučaju da ne postoje nikakvi šumovi. U stvari, faze mogu da se razlikuju za 45° ± *x°,* gde *x* odgovara šumu. Slično tome, amplituda signala se, u stvari, meri kao A ± *y,* gde y odgovara šumu.

Na slici 3.22 prikazan je efekat šuma na konstelaciju signala. Promena u amplitudi pomera konstelaciju tačke dalje, ili bliže koordinatnom početku. Zbog toga, tačka signala može da se nade bilo gde u zasenčenom regionu. Slično tome, izobličeni fazni pomak može da izazove pomeranje tačke u okviru malih kružnih lukova. Da "stvar" bude još nepovoljnija, šum se ne razaznaje. Različiti tipovi izobličenja mogu da se jave nezavisno jedni od drugih. Rezultat je to da se tačka konstelacije izobličenog signala može naći bilo gde u kružnoj oblasti oko pozicije u kojoj bi trebalo da se nalazi.

Ako su inicijalne tačke dovoljno udaljene i ako je šum dovoljno slab, neće doći do preklapanja oblasti šuma. U tom slučaju modem može da prepozna izobličeni signal. Međutim, ako je šum dovoljno jak da dode do preklapanja regiona, komunikacija će biti oslabljena. Ako se tačka izobličenog signala nalazi na preseku dve zasenčene oblasti, modem ne može da utvrdi kojoj oblasti tačka pripada (slika 3.23).



SLIKA 3.22 *Izobličenje tačaka konstelacije signala*



SLIKA 3.23 *Interpretiranje tačaka konstelacije kod izobličenog signala*

Standardi za modeme

Kao što ste mogli i da očekujete, postoje različiti standardi za modeme. Razlikuju se po brzini bauda, broju bitova po baudu i tehnikama modulacije. Osim toga, noviji standardi definišu i detekciju grešaka, metode za njihovo korigovanje i tehnike kompresije. U tabeli 3.5 rezimirani su neki ITU-T standardi.

Tabela 3.5: Neki ITU-T standardi za modeme iz V serije

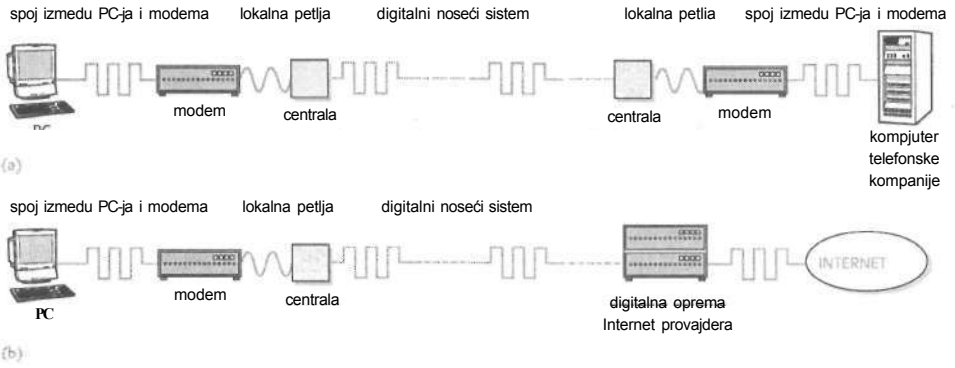
V.21/Bell 103	Bitska brzina od 300 bps, sa FSK modulacijom
V.22/Bell 212	Bitska brzina od 1.200 bps, sa PSK modulacijom
V.22 bis	Bitska brzina od 2.400 bps, sa QAM modulacijom
V.27	Bitska brzina od 4.800 bps, sa PSK modulacijom
V.29	Bitska brzina od 9.600 bps, sa QAM modulacijom (ranije uobičajeni standard za faksimil)
V.33	Bitska brzina od 9.600 bps, sa QAM modulacijom i rešetkastim kodiranjem
V.32 bis	Bitska brzina od 14.400 bps, sa QAM modulacijom i rešetkastim kodiranjem
V.34	Bitska brzina od 33.600 bps, sa QAM modulacijom i rešetkastim kodiranjem (obično se koristi za faks modeme)
V.42	Standard za tehnike za korekciju grešaka
V. 42 bis	Standard koji koristi Lempel-Zivove metode (prikazani su u Poglavlju 5) za kompresiju
V.90	Brzina od 56 Kbps (samo kod preuzimanja) uz korišćenje PCM tehnika (ovo je izvodljivo uz pretpostavku da na udaljenom sajtu nema konverzije analognih u digitalne signale)
V.92	Standard V.90 poboljšan redukovanjem vremena koje je neophodno za povezivanje, tako da je povećana brzina slanja i obezbedena je bolja zaštita od slučajnih prekida veze za one koji koriste poziv na čekanju (u vreme dok je ova knjiga nastajala, većina novih kompjutera je imala V.92 modeme)

Mnogi modemi poštuju nekoliko standarda, što je korisno kada se komunicira sa sajtom na kome je implementirano nekoliko standarda. Obično, modem može da poziva, razmenjuje protokole, pa automatski bira odgovarajući standard. Ovi *autobaud modemi* su pogodni zato što korisnici ne moraju da pamte kojem broju telefona odgovara koji standard. Na ovaj način je korisnicima omogućena i komunikacija pomoću nekoliko standarda preko istog modema i kompjutera. Korisnici mogu da utvrde standarde proučavanjem tehničkih specifikacija u uputstvu za uređaj. Obično postoji sekcija u kojoj je naznačen mehanizam za kodiranje podataka na nekoliko brzina prenosa podataka.

Razvoj modema je nastavljen tokom 90-ih godina prošlog veka, tako da je uspostavljen standard V.90, koji obezbeduje brzinu preuzimanja od 56 Kbps. Na kraju odeljka 3.4 istakli smo da se čini kao da su teorijske granice u brzini prenosa podataka koje defmiše Šenonov rezultat premašene. Ipak, ti modemi su zasnovani na pretpostavkama koje se značajno razlikuju od onih koje su korišćene kod starijih modema.

Na slici 3.14 dato je objašnjenje. Starije konekcije su slične onima koje su opisane na slici 3/24a. Personalni kompjuter šalje signal ka portu modema, koji konvertuje signal u odgovarajući analogni format. Analogni signal "putuje" kroz lokalnu petlju telefonskog sistema do najbliže telefonske centrale, gde se konvertuje nazad u digitalni format, radi kompatibilnosti sa opremom za komutaciju telefonske kompanije. Odatle se digitalni signal rutira preko nosećeg sistema do centrale koja je najbliža udaljenom sajtu. Signal se ponovo konvertuje u analogni oblik i "putuje" kroz drugu lokalnu petlju do udaljenog sajta, gde prijemni modem vrši konvertovanje u digitalni oblik.

Na slici 3.24b ne vide se poslednje dve konverzije. ISP ima digitalnu opremu koja komunicira direktno sa nosačem digitalnih signala koji "putuju" kroz telefonsku mrežu i može da ih rutira preko Interneta, bez ikakve konverzije izmedu analognih i digitalnih signala. Ovo je značajno zbog sledećeg razloga. Pretpostavimo da korisnik preuzima informacije sa udaljenog kompjutera na slici 3.24a.



SLIKA 3.24 *Uspostavljanje konekcija pomoću modema*

Preuzeti signal prolazi kroz konverziju analognog u digitalni oblik na udaljenoj lokaciji i konvertuje se nazad u analogni signal na lokalnom sajtu. Medutim, šum kvantizacije koji se uvodi na udaljenoj lokaciji na lokalnom modemu može da prouzrokuje prijem analognog signala koji se razlikuje od poslatog. Prema Senonovim rezultatima, ako je bitska brzina dovoljno niska, signali se manje izobličavaju i lokalni analogni signal je dovoljno blizak onom na udaljenoj lokaciji, tako da ne dolazi do gubljenja informacija.

Na slici 3.24b nema konverzije analognih u digitalne signale na udaljenoj lokaciji; zato se ne javlja šum kvantizacije i moguće je ostvariti veće bitske brzine. 56 Kbps modemi su dizajnirani tako da se iskoristi ova činjenica. Osim toga, telefonska oprema koristi PCM modulaciju za konverziju izmedu analognih i digitalnih signala; V.90 modemi digitalizuju analogne signale, koristeći PCM tehnike, umesto da se koriste QAM tehnike, kao kod starijih modema.

Slanje informacija je malo drugačije. Pošto postoji konverzija analognih signala u digitalne u izvom na koji se informacije postavljaju, moguća je pojava šuma kvantizacije. Zato se standard V.90 primenjuje samo kod preuzimanja informacija, a tipični 56 Kbps modemi prilikom slanja informacija koriste standard V.34. Zbog toga, mogu brže da preuzimaju informacije nego što ih šalju.

Naravno, sve vreme je korišćena pretpostavka da je linija lokalne petlje na strani modema relativno bez šumova, mada često nema nikavih garancija da će biti tako. Telefonska oprema je dizajnirana da funkcioniše pod odredenim ograničenjima i često ih premašuje. Ipak, nema nikakvih garancija da će sve telefonske linije moći da podrže granicu od 56 Kbps. U stvari, prema nekim procenama, samo 50 odsto konekcija u SAD je dovoljno "čisto" da se podrži prenos od 56 Kbps.

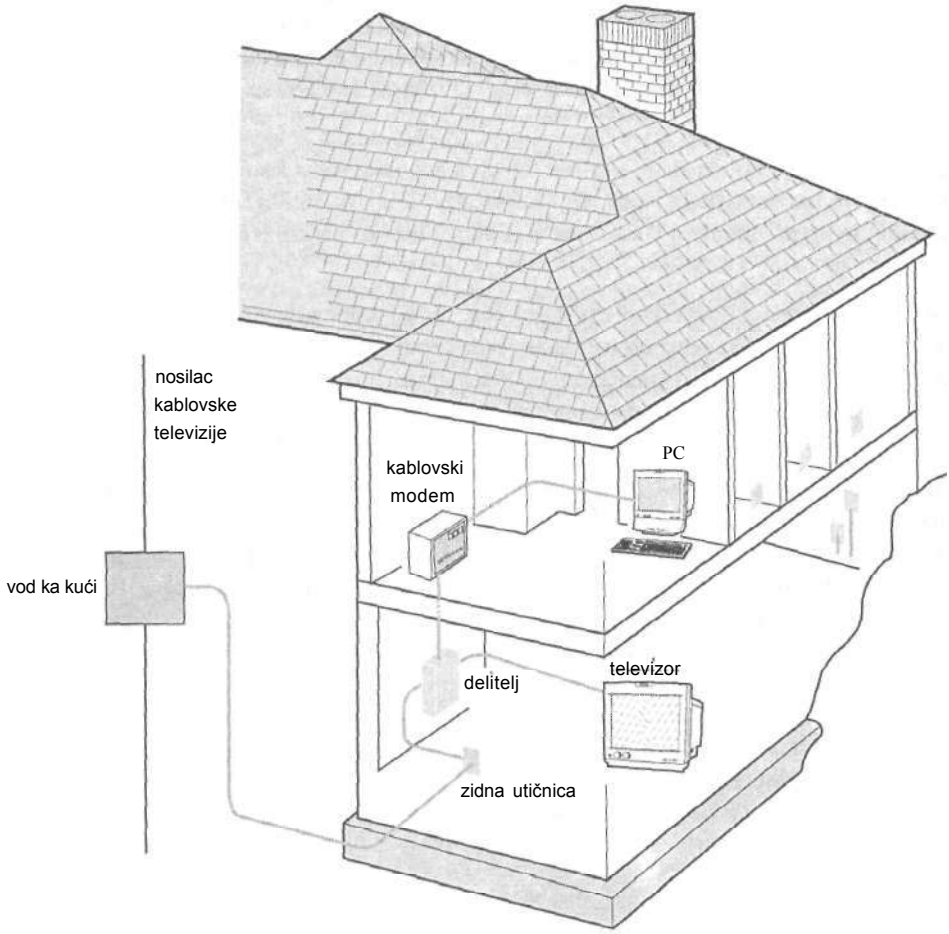
Kako se svet sve više orijentiše ka digitalnim standardima za komunikacije, nameće se logično pitanje kakva je budućnost modema. Zbog faktora šuma i specifikacija telefonske opreme, 56 Kbps modemi se smatraju poslednjim korakom u razvoju modema. Medutim, i pored trenda za uspostavljanje potpuno "digitalnog sveta", konvencionalni telefoni i analogne konekcije do najbliže telefonske centrale verovatno neće biti napušteni u bližoj budućnosti. Tehnologija koja bi sve telefone zamenila onima koji digitalizuju zvuk preko ugradenog čipa definitivno postoji, ali kolika je cena takve zamene na nacionalnoj, ili, čak, globalnoj skali? Za one koji koriste telefon u njegovoj originalnoj nameni (za razgovor) trenutni sistem funkcioniše sasvim zadovoljavajuće i prelazak na digitalni sistem ne bi obezbedio neki vidljiviji napredak za njih. Drugim rečima, nije vredno neophodnih ulaganja. Zato će konvencionalni modemi verovatno biti neophodni i u bližoj budučnosti. Ipak, postoji sve veće "nadmetanje" izmedu kablovskih modema i DSL tehnologije.

Kablovski modemi

Naša sledeća tema je kablovski modem, uredaj koji se poslednjih godina sve više sreće u domoviraa korisnika. Korisnici usluga kablovske televizije zahtevali su brže Internet konekcije, koje nisu mogli da im obezbede konvencionalne telefonske linije i modemi sa granicom od 56 Kbps. Za razliku od konvencionalnog modema, koji je dizajniran za povezivanje sa analognim komponentama telefonskog sistema, **kablovski modem** je dizajniran za povezivanje sa analognim komponentama provajdera kablovske televizije (CATV).* Mnogi pretplatnici kablovske televizije već poznaju kablovsku kutiju koja se nalazi na vrhu televizora.

Ona se priključuje na zidnu utičnicu preko koje se primaju signali provajdera kablovske televizije. Kutija dekodira dolazeće skremblovane signale i šalje ih do televizijskog uredaja. Često ove kutije mogu da primaju komande od daljinskog upravljača i da šalju podatke nazad do kompanije za kablovsku televiziju. Zahvaljujuđ tome, pretplatnik može da naručuje i plaća filmove koje odgleda.

Do određene mere, svi kablovski modemi su slični. Na slici 3.25 prikazana je tipična instalacija. Potrošač se pretplaćuje na CATV servis, a tehničko osoblje sprovodi kabl od spoljašnjeg voda do kuće.



SLIKA 3.25 *Postavka kablovskog modema*

*U budućnosti će svi televizijski signali biti digitalni i tekući tekvizijski sistem više neče biti u upotrebi za prijem takvih signala bez pomoći konvertora. Medutim, nismo stigli do te tačke i još neko vreme ćemo morati da koristimo analogne signale.

Unutar kuće kabl se priključuje na ulazni kraj delitelja (uređaj koji prihvata izvorni signal i aitira ga preko dve, ili više izlaznih konekcija). Sledeći kabl povezuje jedan izlaz delitelja na televizor, ili kablovskn kutiju. Signal iz drugog izlaza se vodi na kablovski modem. Nakon toga, korisnik može da koristi Cat 5 kabl za povezivanje kablovskog modema preko mrežne kartice kompjutera. * Mrežne kartice su jeftine i mogu se kupiti u radnjama sa kompjuterskom opremom. U stvari, one su sastavni deo većine konfiguracija koje danas možete da kupite.

Pomoću mrežne kartice, kao što je Ethernet kartica (opisana u Poglavlju 9), omogućava se kablovskom modemu da konvertuje analogne signale u digitalne i da ih šalje do kompjutera preko postojećih Ethenet protokola. Korišćenje Ethernet konekcije omogućava kablovskom modemu da koristi većpostojeće standarde za povezivanje na kompjuter. Osim toga, omogućeno je povezivanje nekoliko kompjutera preko Ethernet LAN-a deljenjem kablovskog modema preko komutatorat (objašnjen je u Poglavlju 10). U ovom slučaju svaki kompjuter se povezuje na komutator, a između komutatora i kablovskog modema postoji samo jedan kabl.

Kablovski modem je dizajniran da bi omogućio pristup Internetu preko CATV signala, umesto da se poziva Internet provajder preko telefona. Postoji nekoliko prednosti ove postavke. Jedna je da se informacije mogu poslati pomoću visokofrekventnih signala CATV-a, umesto da se koriste mnogo niže frekvencije lokalne petlje telefonskog sistema, tako da su moguće mnogo veće bitske brzine (mere se u Mbps, umesto u Kbps). Sledeća prednost je što ne morate da birate broj da bi konekcija bila uspostavljena; ona je neprestano uspostavljena. Nedostatak je što CATV koristi zajednički kabl (obično optički fiber) za obezbedivanje servisa za više kuća u susedstvu, tj. opseg signala se deli na više korisnika. Zato je moguće da korisnik dobija ove bitske brzine samo ako jedini u svom okruženju preuzima informacije na ovaj način. Medutim, ako i susedi počnu da preuzimaju informacije, to utiče na bitsku brzinu svakog individualnog korisnika.

Na slici 3.26 prikazane su osnovne operacije samog kablovskog modema. Tipično, kablovski signal na ulazu u kućnu instalaciju pripada opsegu od oko 750 MHz. On se deli na više opsega od po 6 MHz, a svaki od tih opsega prenosi signal iz odredene stanice, kao što su Discovery, ESPN, ili CNN. Podešavanje kanala efektivno blokira neželjene frekvencije i omogućava prolazak samo jednog 6-MHz signala, tako da je moguće prikazivanje željene stanice. Radi pristupa Internetu, kompanija za kablovski servis uspostavlja konekciju sa Internetom posredstvom nekog provajdera. Informacije sa Interneta mogu da se preuzimaju u 6-MHz opsegu, negde izmedu 42 i 750 MHz. t

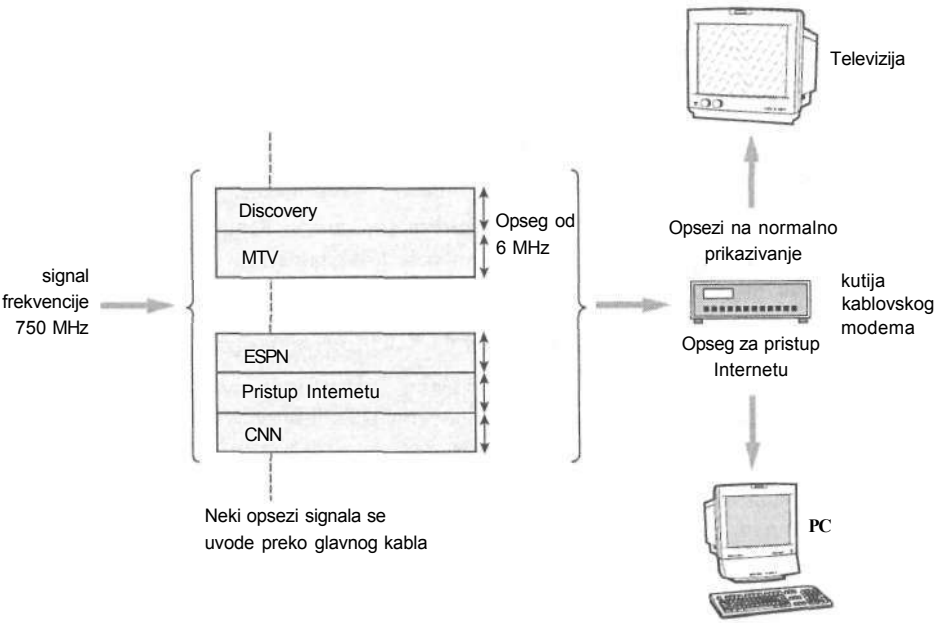
Na strani korisnika kablovski modem može da pristupi preuzetim informacijama podešavanjem u odgovarajućem 6-MHz opsegu i konvertovanjem tih analognih signala u digitalni format. Zatim se ti signali šalju kroz odgovarajući port do povezanog kompjutera.

Za modulaciju i demodulaciju mogu da se koriste razne tehnike; dve najpopularnije su QPSK (quaternary phase shift keying) i varijacija QAM-a poznata kao QAM64.

* Kablovski modem može da se poveže i na USB port kompjulera.

t Može da se koristi i hub, mada se komutatori češće koriste, jer su fleksibilniji i relativno jeftini.

t Preuzeli signali se obično nalaze u opsezima iznad 42 MHz, jer su signali na nižim frekvencijama podložni uticaju interference od kučnih aparata.



SLIKA 3.26 *Kablovski modem i noseći signali*

Oba metoda predstavljaju složenije verzije tehnika sa sličnim nazivima, koje smo opisali ranije u ovom poglavlju. QAM64 je obično namenjen za situacije u kojima se zahteva širokopojasni opsegza preuzimanje informacija. Prema nekim procenama, moguće je preuzimanje podataka sa brzinom od 36 Mbps. Ipak, mnogi personalni kompjuteri ne mogu da prihvate podatke tom brzinom, tako da se realne brzine kreću izmedu 1 i 10 Mbps.

Kablovski modem može da prenosi informacije u drugom smeru (postavljanje podataka). Može da prihvati podatke poslate iz kompjutera i da ih modulira u frekventnom opsegu, obično između 5 i 40 MHz. Ovaj opseg se obično koristi na dvosmernoj kablovskoj mreži prilikom postavljanja podataka (uploading). Problem je što su signali iz ovog opsega više osetljivi na električnu interferencu koja potiče od kućhih aparata. Zato su QPSK tehnike češće korišćene prilikom slanja podataka, jer su robustnije. Nedostatak je što imaju manje bitske brzine. Sa druge strane, prilikom postavljanja podataka često i ne postoje zahtevi za velikim brzinama prenosa. Na primer, slanje emaila, ili komandi obično zahteva manje podataka nego kada se preuzimaju neka slika, ili video zapis.

Kao što smo istakli, ostaje još mnogo štošta da se uradi radi uspostavljanja standarda za korišćenje kablovskih TV mreža za prenos podataka. IEEE 802.14 Working Group radi na kreiranju takvih standarda, a malo je toga do sada postignuto. U referenci [He97] dat je pregled nekih arhitekturalnih opcija koje su razmatrane za obezbedivanje traženih servisa na efikasan, ali fleksibilan način.

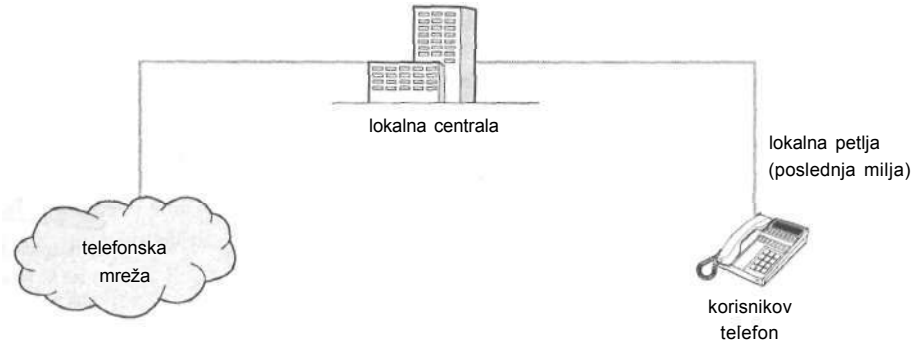
Do sada smo opisali dve primarne opcije za povezivanje korisnika na Internet: modeme i kablovske modeme. Modemi obezbeduju interfejs između kompjutera i telefonskog sistema obezbedivanjem konverzije izmedu digitalnih signala iz kompjutera i analognih signala telefonskog sistema. Kablovski modemi izvršavaju sličan zadatak, osim što koriste liniju kablovskog TV servisa. Konvencionalni modemi imaju nekoliko značajnih nedostataka:

- Spori su (u poredenju sa drugim opcijama) i već su dostigli svoj maksimum.
- Koriste propusni opseg telefonskog sistema, tako da telefon ne može da se koristi istovremeno dok ste povezani na Internet.
- Morate da pozivate Internet provajdera svaki put kada želite da se povežete.

Sa druge strane, konvencionalni modem je dostupan svuda gde postoje telefonske linije i ISP, što je najveća prednost. Za razliku od njih, kablovski modemi su brzi i veza je uvek uspostavljena. Medutim, kablovska televizija nije svuda dostupna.

U ovom odeljku predstavićemo još jednu moguću opciju za korisnike: digitalnu pretplatničku liniju **(DSL - Digital Subscriber Line).** Kao i kablovski modem, DSL postiže velike brzine i omogućava neprekidnu konekciju, ali ne zahteva kablovsku televiziju, već koristi specijalno kabliranje. Koristi postojeće telefonske linije! Na prvi pogled, ovo što smo istakli deluje kontradiktorno sa prethodnim tvrdnjama. Konekcije sa Intemetom koje se uspostavljaju preko telefonskih linija su spore. Kako, onda, DSL postiže veće brzine?

Počnimo sa objašnjenjima proučavajući sliku 3.27. Veđna korisnika telefone povezuje na telefonsku mrežu preko bakarnih žica koje se protežu do lokalne telefonske centrale. Ova postavka bakarnih kablova najčešće se naziva lokalna petlja, ili poslednja milja (odnosi se na činjenicu da je ovo poslednji deo telefonskog sistema **i** da se i dalje koristi stara tehnologija). Lokalna petlja i oprema u lokalnoj centrali obraduju niskopropusne signale, jer nije bilo ekonomično rutirati fiber, ili druge kablove koji podržavaju velike brzine do svih korisnika - cena bi bila ogromna. Prikazana konfiguracija je poznata pod nazivom POST (plain old telephone service - tradicionalni telefonski sistem).



SLIKA 3.27 *Lokalna petlja*

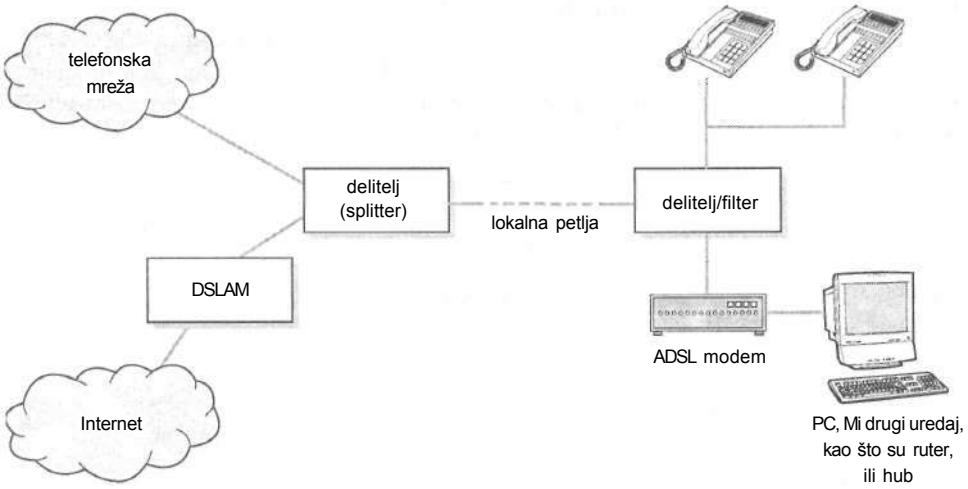
Međutim, mnoge telefonske kompanije su uvidele da je high-speed Internet oblast koju nisu "pokrili", a koja može itekako da se isplati. Suština je bio u tome da se osmisli način na koji će se korisnicima, po razumnoj ceni, obezbediti pristup Internetu sa velikim brzinama prenosa, bez dodatnih troškova zbog zamene postojećih telefonskih kablova. Rešenje leži u činjenici da je ograničenje propusnog opsega postojalo zbog opreme u lokalnoj centrali, a ne zbog bakarnih vodova u lokalnoj petlji. Iako kablovi lokalne petlje nisu izradeni po specifikacijama današnjih veoma brzih kablova sa upredenim paricama i imaju ozbiljna ograničenja, ipak mogu da prenose visokofrekventne signale (do 1 MHz) u odnosu na one koji se koriste za telefoniju. Za sve to je neophodna specijalna oprema u lokalnoj centrali koja može da interpretira visokofrekventne signale. To je osnova DSL-a.

DSL nije jedinstvena tehnologija. U stvari, postoji više njegovih verzija. Mi ćemo ovde opisati ADSL (asimetrični DSL), a zatim ćemo navesti i ostale DSL tehnologije, sa kraćim opisom njihovih razlika. Kompletnu raspravu o DSL-u možete da pročitate u referenci [GoOl].

Kako DSL funkcioniše?

Asimetrični DSL se zasniva na par standarda: ITU-T preporučuje G.992.1 i ANSI standard T1.413 (standard za modulaciju signala). Reč asimetrično se odnosi na činjenicu da je bitska brzina prilikom preuzimanja informacija veća od brzine slanja informacija i ova logička konfiguracija se koristi zato što većina korisnika više vremena provodi u preuzimanju velikih fajlova nego u njihovom slanju. Na slici 3.28 prikazana je osnovna ADSL konfiguracija.

Na strani korisnika je neophodna specijalna oprema: delitelj/filter i ADSL modem. Delitelj funkcioniše korišćenjem dva filtera. Jedan je niskopropusni filter, koji blokira signale iznad 4 kHz i šalje niskofrekventne signale do telefona.



SLIKA 3.28 *ADSL konekcija*

Drugi filter propušta samo visoke frekvencije, i to samo ADSL signale (iznad 4 KHz) do ADSL modema (ponekad se naziva ADSL *Transmission Unit-Remote,* ili ATU-R). Interesantan rezultat ove konfiguracije je što telefon i kompjuter zavise od signala iz različitih opsega. Na osnovu Furijeovih rezultata, koje smo ranije prikazali, ovi signali mogu da se kombinuju u jedan i da se prenesu preko lokalne petlje. Prednost za korisnika je što istovremeno, dok se podaci preuzimaju sa Interneta, može da razgovara sa nekim preko telefona. Time je prevaziđen jedan od glavnih nedostataka konvencionalnih modema.

U lokalnoj centrali mora da postoji odgovarajuća oprema koja može da radi sa različitim tipovima signala koji dolaze od korisnika. Da bi ti signali bili obradeni, u lokalnoj centrali mora da se instalira DSL pristupni multiplekser (DSLAM - **DSL** access multiplexer). Delitelj u lokalnoj centrali prima dolazeći signal i šalje niskofrekventne signale (kojima se prenosi glas) do telefonske mreže, radi kompletiranja telefonskog poziva. Salju se visokofrekventni signali (na primer, iz kompjutera) do DSLAM-a - on interpretira signale koje korisnik DSLAM kreira i rutira podatke ka Internetu, radi kompletiranja Intemet konekcije.

Dakle, po čemu se ovo razlikuje od tehnologije konvencionalnih modema? Umesto da se primenjuje direktna modulacija, tehnologija SDLAM modema je zasnovana na tehnici koja se naziva definisanje diskretnih tonova (discrete multitone) - ANSI standard Tl.413. Osnovna ideja se sastoji u sledećem;

- Frekventni opseg izmedu 0 Hz i 1.104 MHz se deli na 256 zasebnih kanala, sa opsezima od 4,3125 kHz. Ponekad se kanali nazivaju tonovi.
- Za POTS se koristi pet najnižih kanala. Tako POTS dobija opseg od oko 21,5 kHz (više nego što je potrebno), a dodatni propusni opseg obezbeduje dodatno razdvajanje POTS signala od ADSL signala i, samim tim, veću imunost na šumove, kao što je preslušavanje.* Neiskorišćene frekvencije izmedu kanala nazivaju se zaštitni opsezi **(guard bands).**
- Preostali kanali se koriste za prenos informacija u oba smera; veći deo se rezerviše za preuzimanje signala. Na taj način su omogućene veće brzine prenosa prilikom preuzimanja informacija. Sa stanovišta korisnika, ovo je prednost. Jednostavnije je razdvojiti signale koji se koriste za preuzimanje informacija od onih koje se koriste za slanje. Ipak, postoji tehnologija koja omogućava korišćenje nekih kanala za prenos u oba smera. To je malo nezgodno u praksi, jer, ako signali iste frekvencije "putuju" u suprotnim smerovima, dolazi do pojave eha. To znači da uredaj ne prima samo poslati signal, već i eho koje stvaraju sopstveni signali. Medutim, u prijemnim uređajima je moguće ugraditi supresore eha, koji će filtrirati taj eho i obezbediti jasan prijem podataka koji su poslati sa udaljene lokacije. Prednost ove tehnike je proširenje propusnog opsega u oba smera, čak i ako se kanali preklapaju.

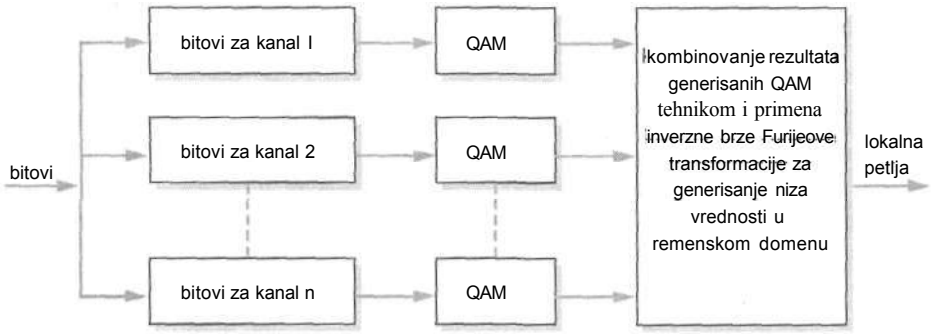
* Preslušavanje može da se pojavi kada signal generiše elektromagnelnu interferencu, koja, sa druge strane, utiče na signal u susednim upredenim paricama. Preslušavanje često može da se redukuje dobro uzemljenim sistemima, ili opremom proizvodenom u skladu sa odredenim specifikacijama. Poš(o, u tom slučaju, signali "putuju" preko postojećih žica, preslušavanje je minimizirano izbegavanjem signala odredene frekvencije.

- Da bi podaci bili preneti, dolazed niz bitova se deli na manje grupe bitova - po jedna grupa za svaki kanal (slika 3.29). Ove grupe bitova se tretiraju konkretno i nezavisno.
- Na bitove u svakom kanalu primenjuje se QAM tehnika (prisetite se odeljka 3.5). Tako se kreiraju noseći signali različitih frekvencija za svaki kanal. Svaki kanal može da koristi sopstvenu QAM tehniku, koju odreduje broj bitova za taj kanal. Uskoro ćete videti da broj bitova može da varira od kanala do kanala.
- Kombinuju se signali generisani QAM tehnikom i podvrgavaju se inverznoj brzoj Furijeovoj transformaciji za modulaciju signala (podsetite se prethodne rasprave o Furijeovim redovima i brzoj Furijeovoj tranformaciji).

Naravno, ovde je dat pojednostavljen pregled, pa je neophodno nekoliko dodatnih komentara. Na primer, jedinstveni aspekt ovog metoda je da se, za razliku od prethodno opisanih metoda, koriste različiti noseći signali za istovremenu obradu grupa bitova. Teorijski, moguće je preneti *60 Kbps za* svaki kanal, a!i *su* ovakvi *rezultati retki* u praksi zbog *prisustva šuma.* U stvari, kao deo DMT prenosa, ADSL modem šalje test signale preko svakog kanala, radi utvrdivanja nivoa šuma u svakom kanalu. Kanali sa boljim koeficijentom signal-šum prenose više podataka, nego oni sa lošijim koeficijentom. Obično su kanali sa višim frekvencijama osetljiviji na šumove, tako da niskofrekventni kanali bolje prenose veće količine podataka. Ovde treba istaći da pri-Iagodljivost DTM-a u utvrdivanju bitske brzine zavisi od uslova na postojećoj liniji.

U opštem slučaju, svaki kana] ima sopstvenu konstelaciju signala koja definiše broj bitova za njega. Bitske brzine kod preuzimanja informacija za ADSL obično se kreću izmedu 1,5 i 6 Mbps, a umnogome zavise od kvaliteta lokalne pedje. Na primer, žice koje su bile upletene, ili popravljane na neki drugi način mogu da dovedu do stvaranja eha i izobličenja, zbog čega dolazi do redukovanja bitske brzine. Čak i veličina (debljina) žice može da ima uticaja.

Problem stvara i dužina petlje, jer se signal degradira prilikom prenosa na većim udaljenostima. Bitske brzine mogu da budu niže kod korisnika koji se nalaze dalje od lokalne centrale. U stvari, većina raspoloživih informacija ukazuje da DSL konekcije nisu dostupne korisnicima koji žive dalje od 3,5 milje od lokalne centrale.



SLIKA 3.29 *Definisanje diskretnih tonova*

Sledeći jedinstveni aspekt je korišćenje inverzne brze Furijeove transformacije. U odeljku 3.3 smo istakli da se nećemo baviti njenim detaljima, jer bi to premašilo predvideni obim ove knjige. Dovoljno je reći da kolekcija QAM signala definiše flinkcije u frekventnom domenu. Propuštanjem tih funkcija kroz inverznu brzu Furijeovu transformaciju dobija se kolekcija vrednosti u vremenskom domenu (tj. diskretna reprezentacija). Eventualno, diskretne vrednosti se konvertuju u analogni oblik i prenose preko lokalne petlje. Proces je suprotan u lokalnoj centrali.

Poput kablovskih modema, ADSL modemi obezbeđuju neprekidnu konekciju. Nema potrebe da pozivate Internet provajdera; ADSL komunikacije mogu da se koriste paralelno sa POTS konverzacijama. Osim toga, ADSL servisi su ponekad dostupni u oblastima u kojima nema kablovske televizije. Sa druge strane, kablovski modemi koriste veći opseg signala nego ADSL rnodemi i obično prosečnom korisniku omogućavaju veće bitske brzine.

Pošto ADSL tehnologija koristi postojeće kabliranje, svaki korisnik ima point-to-point link ka lokalnoj centrali. Ovo je suprotno pristupu kablovskih servisa, gde ljudi iz susedstva dele isti opseg signala. Ako ste jedini korisnik sa kablovskim modemom, brzina Vaših konekcija će verovatno biti maksimalna. Medutim, kako se bude povećavao broj korisnika u Vašem susedstvu, moraćete da delite opseg sa njima i to će imati uticaja na sve pojedinačne konekcije. Naravno, ovo utiče i na brzine prenosa podataka.

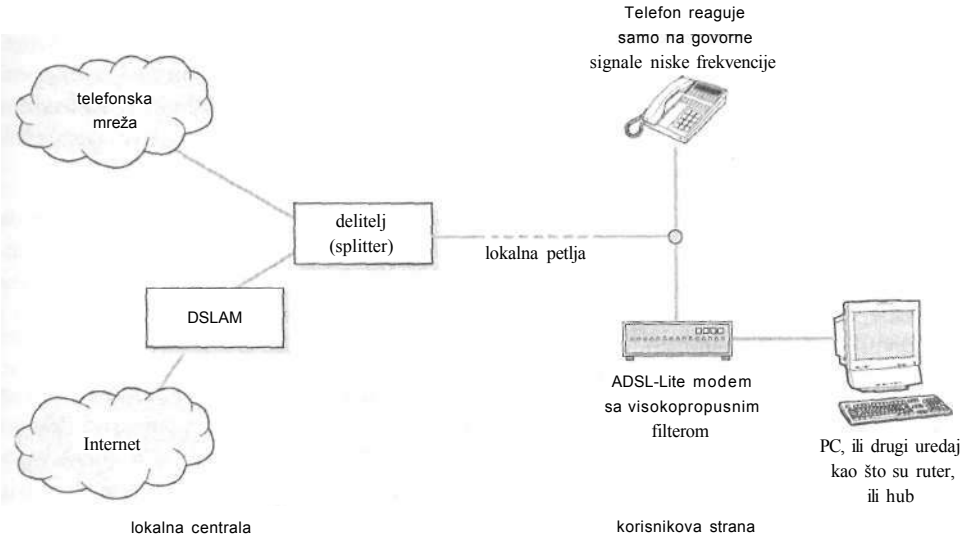
Različite DSL tehnologije

Postoje brojne varijacije DSL tehnologije. Zainteresovani čitaoci mogu da pronadu više informacija na Web sajtovima www.dslforum.org i www.dsllite.com. Za ove tehnologije koristi se opšta notacija xDSL, gde je *x* oznaka varijacije. Slede krad opisi za neke od dostupnih DSL tehnologija.

ADSL LITE (G.LITE, ITU STANDARD ITU G.992.2) Kao što ste videli u prethodnoj raspravi, ADSL zahteva korišćenje delitelja/filtera na strani korisnika. To je obično značilo da ćete morati da angažujete tehničku službu telefonske kompanije da ga instalira. ADSL Lite (ponekad se naziva ADSL bez delitelja) ne postavlja takav zahtev. Njegova prednost je jednostavnost za korisnika; u stvari, ADSL Lite je dizajniran za korisnike u stambenim zgradama. Korisnici običnih i kablovskih modema znaju da je to sve što im je potrebno za povezivanje na telefonsku mrežu, ili liniju kablovske televizije. Ako DSL tehnologija treba da bude konkurentna, mora da se obezbedi laka instalacija.

Deljenje signala se izvodi u lokalnoj centrali, umesto kod korisnika (slika 3.30). Konvencionalni telefoni ne reaguju na signale čije su frekvencije više od 4 kHz i zato ignorišu ADSL signale. ADSL modem uključuje visokopropusni filter, koji blokira signale niske frekvencije i obraduje samo ADSL generisane signale. Korisnik vidi prednost mogućnosti povezivanja postojećeg kabliranja sa bilo koje lokacije u kući. Sa stanovišta korisnika, u odnosu na ADSL, ovo se pre može nazvati plug-and-plav tehnologija.

Sledeća razlika je što se dolazni signali kod ADSL Lite tehnologije nalaze u intervalu od 25 do 500 kHz. Time je bitska brzina kod preuzimanja informacija ograničena na oko 1,5 Mbps. Razlog za ovo ograničenje je činjenica da, bez lokalnog delitelja, ADSL Lite signali "putuju" kroz vodove u kući i visokofrekventni signali mogu da izazovu smetnje na osnovnom telefonskom servisu, posebno na mestima gde je kvalitet izvedenog kabliranja lošiji.



SLIKA 3.30 *ADSL-Lite konekcija*

Ovaj problem se redukuje eliminisanjem signala sa višom frekvencijom. Iako je sporiji od ADSL-a, i dalje omogućava oko 30 puta veće brzine od konvencionalnih modema.

SDSL Za razliku od asimetričnog DSL-a, simetrični DSL (SDSL) obezbeđuje istu bitsku brzinu u oba smera pomocfu jednog para žica u lokalnoj petlji. Osim toga, on se koristi kao zajedniđd termin za druge DSL tehnologije, koje će biti opisane u narednim pododeljcima. Ponekad se SDSL oďnosi na DSL sa jednim kablom, varijantu HDSL-a (videti sledeći pododeljak). SDSL koristi i drugačiji mehanizam za signaliziranje, pod nazivom dvobinarni, jednokvaternarni (2BlQ-tiw *binary, one quaternary).* Ova šema je pomalo nalik NRZ šemi, koju smo opisali u odeljku 3.2, osim što se definišu četiri nivoa signala za svaki par bitova, umesto da se deflnišu dva nivoa signala za jedan bit. Drugim rečima, svaki od parova 00, 01, 10 i 11 ima definisan fiksni signal. Glavna prednost je što je bitska brzina dva puta veća od brzine bauda (svaka promena signala sadrži dva bita).

HDSL I HDSL2 High-bit-rate DSL (HDSL) je simetrični DSL servis koji je razvijen krajem 80-ih godina prošlog veka. Obezbeđuje bitske brzine od oko 1,5, ili 2,3 Mbps (u oba smera), a koriste se dva, ili tri para upredenih bakamih vodova. Kao što smo ranije istakli, postoji varijanta sa jednim parom vodova (SDSL) kod koje se koristi jedan par. HDSL ne obezbeđuje tradicionalni telefonski servis (POTS servis) i bio je razvijen kao ekonomična altemativa za T1, ili El service.* HDSL2 se razlikuje od HDSL-a na dva načina. Prvo, ANSI standard za HDSL2 definiše brzinu od 1,5 Mbps u oba smera. Druga razlika je u činjenici da se ta bitska brzina obezbeđuje preko jednog para žica (HDSL zahteva dva para za tu brzinu). Slično HDSL-u, ne obezbeđuje standardni telefonski servis preko tog para žica.

SHDSL Single-pair high-speed DSL (SHDSL) je nova tehnologija koja je saglasna sa ITlI standardom G.991.2. Kod HDSL i HDSL2 tehnologija postojalo je ograničenje za maksimalnu udaljenost petlje aproksimativno od 12.000 stopa. Kod SHDSL-a je povećana udaljenost petlje i obezbedene su brzine od 2,3 Mbps preko jednog para žica. Standard specificira jedan par žica; medatim, dva para (mod sa četiri žice) mogu da se koriste za povečavanje udaljenosti petlje, ili bitske brzine (mada ne istovremeno). Na primer, mod sa četiri žice može da se koristi za obezbedivanje bitske brzine od 2,3 Mbps na aproksimativno 16.000 stopa, ili 4,6 Mbps na kraćim rastojanjima. Kod moda sa četiri žice bitovi se podjednako raspoređuju preko dva para; sa stanovišta aplikacije, on obezbeduje veće bitske brzine. Uglavnom je namenjen velikim poslovnim organizacijama i korisnicima koji zahtevaju veće brzine prenosa i slanja podataka.

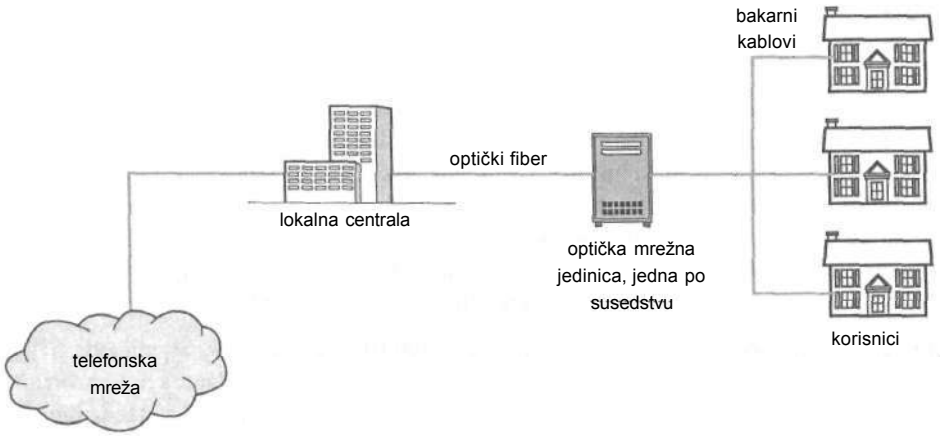
RADSL Asimetrični DSL sa ađaptacijom brzine (RADSL Rate-adaptive asymmetric DSL) ne odgovara specifičnom standardu. Omogućava ADSL modemu da adaptira bitske brzine u zavisnosti od kapaciteta lokalne petlje. Nema mnogo referenci na RADSL, jer standardni ADSL takode omogućava adaptiranje bitske brzine modema.

IDSL Internet DSL (IDSL) je simetrična tehnologija i zasnovana je na ISDN-u.* Obezbeduje brzine prenosa podataka od 128 Kbps na rastojanjima petljedo 18.000 stopa. Obezbeduje servise slične ISDN-u, ali, za razliku od njega, nema govorni kanal.

VDSL Very high data rate DSL (VDSL) je asimetrični servis (sa različitim brzinama prenosa prilikom preuzimanja i slanja podataka), čiji je razvoj još uvek u toku. Trenutno se projektuju brzine preuzimanja podataka od 50 do 55 Mbps na kraćim rastojanjima od 1.000 stopa. Sa povečanjem rastojanja, dolazi do smanjenja bitskih brzina. VDSL se razvija kao odgovor na sve masovnije korišćenje optičkog fibera u komunikadjama. Najveći broj dornova je opremljen bakarnim kablovima i mala je verovatnoća da će u skorijoj budućnosti doći do zamene postojećih instalacija fiberom. Ipak, to ne znači da će cela lokalna petlja koristiti bakame kablove.

Pod uticajem konkurencije, postalo je ekonomično i tehnološki praktično uključiti i fiber i bakarne kablove u lokalnim petljama. Fiber može da se postavi od lokalne centrale do optičke mrežne jedinice (ONU - *optical netivork unit)* u susedstvu (slika 3.31); odade se ka individualnim kućama postavljaju bakarni kablovi. Tako se smanjuje dužina lokalne petlje, koja zavisi od bakarnih kablova, i omogućene su veće bitske brzine. Sledeća karakteristika VDSL-a je da se razvija bez korišćenja nižeg spektra signala. Time je obezbedena kompatibilnost sa običnim telefonskim servisom, što predstavlja prednost za mnoge korisnike. Kombinovane bakarne/fiber konekcije obezbeduju brojne prednosti za okruženja kao što su univerzitetske mreže, poslovni centri, ili industrijski parkovi, gde postoji velika koncentracija korisnika sa potrebama za prenosom podataka na velikim brzinama.

* ISDN (predstavičemo ga u Poglavlju 13) prvobitno je bio razvijen kao standard CCITT-a (sada ITU) još 1984. godine. Dizajniran je tako da obezbedi potpuno digitalizovanu komunikarionu mrežu (u vreme kada je ova ideja još bila u povoju) i mnogi su predvidali da će biti eventualni naskdnik telefonske mreže.



SLIKA 3.31 *Lokalna petlja: Fiber/bakarni hibrid*

3.9 Zaključak

U ovom poglavlju su predstavljeni analogni i digitalni signali, teorijski rezultati o bitskim brzinama, opseg signala, koeficijent signal-šum, tehnike modulacije i načini za povezivanje na Internet. Predstavljeni su sledeći važni koncepti:

- Šeme za digitalno kodiranje uključuju NRZ (nonreturn to zero), Mančester i diferencijalnu Mančester šemu. NRZ dodeljuje jedan fiksni naponski nivo za 0 i drugi za 1. Obe Mančester šeme (nazivaju se i autotaktni kodovi) prepoznaju 0 i 1 na osnovu prelaska signala sa visokog na niski, ili sa niskog na visoki naponski nivo. Mančester šeme sprečavaju dugačke signale, koji mogu da stvore probleme u tajmingu.
- Analogni signali prenose informacije promenom amplituda (amplitudska modulacija), frekvencije (frekventna modulacija), ili faznih pomaka (fazna modulacija). U opštem slučaju, broj bitova po promeni zavisi od broja raspoloživih promena.
- Furijeova teorija pokazuje da su složeni periodični signali sastavljeni od većeg broja signala sa fiksnim frekvencijama. Ovo se primenjuje prilikom dizajniranja filtera za uklanjanje neželjenih frekvencija i prilikom dizajniranja DSL tehnologija.
- Bitska brzina zavisi od brzine bauda, frekvencije i šuma. Nikvistova teorema i teorema o semplovanju pokazuju da se bitska brzina preko bešumnog kanala može izračunati na osnovu izraza $2/ \times \log_2 (B)$, gde je / maksimalna frekvencija, a *B* je broj različitih signala.
- Klod Šenon je proširio Nikvistovu teoremu, uključujući kanale sa šumovima. Njegov čuveni rezultat pokazuje da je bitska brzina = opseg signala x $\log_2 (1+S/N)$, gde su S i N jačine signala i šuma, respektivno. Ovaj rezultat postavlja praktično ograničenje bitske brzine na kanalu sa šumovima.
- Značajna količina komunikacija uključuje prenos digitalnih podataka pomoću analognih signala i analognih podataka pomoću digitalnih signala. Zato je neophodno proudti tehnike modulacije i demodulacije.

Konverzija digitalnih u analogne signale često zahteva promenu analognog signala u skladu sa grupama bitova. Tipične promene utiču na amplitudu (amplitudska modulacija), frekvenciju (frekventna modulacija), ili fazni pomak (fazna modulacija). Sledeća tehnika, poznata kao kvadraturna amplitudska modulacija, koristi kombinaciju ovih promena, a koristi se kod nekih konvencionalnih modema.

- Jedan od načina za konvertovanje analognih signala nazad u digitalne je jednostavno obrtanje ovih procesa. Međutim, ako je originalni signal bio složeni analogni signal, kao što je glas, potreban je drugačiji mehanizam. Jedan pristup se naziva PCM modulacija (impulsna kodna modulacija), a vrši semplovanje analognih signala u pravilnim intervalima. Zatim, svakom semplu dodeljuje niz bitova i prenosi ih. Na prijemnoj strani bitovi se prihvataju i vrši se rekonstrukcija signala.

- Modemi su najrasprostranjeniji uredaji za modulaciju/demodulaciju. Koriste se za povezivanje digitalnih uredaja, kao što su kompjuteri, preko telefonskog sistema. Modemi koriste različite tehnike modulacije, koje su defmisane standardima. Dva uredaja mogu da komuniciraju samo ako njihove opreme prepoznaju isti standard. Osim toga, većina modema predstavlja svojevrsne male specijalizovane kompjutere - mogu pomoću odgovarajućeg softvera da reaguju na komande korisnika, koje se unose preko personalnog kompjutera.

- Kablovski modemi su dizajnirani da bi se obezbedio interfejs izmedu kompjutera i kablovske televizije, koja je sve rasprostranjenija. Teorijski, obezbeden je pristup signalima sa mnogo višim frekvencijama CATV servisa, što omogućava mnogo veće brzine prenosa podataka u odnosu na klasične telefonske modeme.

- DSL tehnologija je sledeća opcija za korisnike koji žele bolje konekcije sa Internetom. Poput kablovskih modema, DSL obezbeduje neprekidnu konekciju. Za razliku od kablovskih modema, DSL se povezuje preko običnih telefonskih linija. Ova tehnologija je zasnovana na prenosu visokofrekventnih signala, koji ne ometaju prenos signala za obične govorne konverzacije. Specijalni DSL modemi implementiraju složenu tehniku koja koristi više kanala (tonova) za prenos grupa bitova i složene matematičke funkcije (inverznu brzu Furijeovu transformaciju) za njihovo kombinovanje.

Pitanja i zadaci za proveru

1. Koje tri komponente u potpunosti opisuju analogni signal?
2. Da li su sledeće konstatacije tačne, ili netačne (zašto)?
   a. Proizvoljan broj bitova može da se pridruži analognom signalu, jer je broj karakteristika signala beskonačan.
   b. Kod digitalnih signala uobičajeno je korićenje visokog signala za predstavljanje 0 i niskog signala za predstavljanje 1, zbog jednostavnosti.
   c. I pored činjenice da je najveći deo opreme, uključujući telefonsku, digitalan, modemi su najznačajniji uredaji u poslednjih nekoliko godina.
   d. DSL modemi izvršavaju iste funkcije kao i konvencionalni; samo koriste više frekvencije.
   e. NRZ kodiranje zahteva dva puta veću brzinu bauda od bitske brzine.

f. Zbog zahteva korisnika za većim bitskim brzinama, telefonske kompanije će verovatno u skorijoj budućnosti zameniti sve bakarne kablove optičkim fiberima.

g. PCM tehnike, iako su dizajnirane za analogne podatke, mogu da se koriste za digitalne podatke.

h. Bitska brzina Internet konekcije preko kablovskog modema zavisi od toga da li i Vaši susedi koriste kablovske modeme.

i. Korisnici mogu da koriste telefonske linije za povezivanje na Intemet, ali, ako žele da razgovaraju sa nekim preko telefona dok neko koristi njihovu konekciju, moraju da imaju drugu telefonsku liniju.

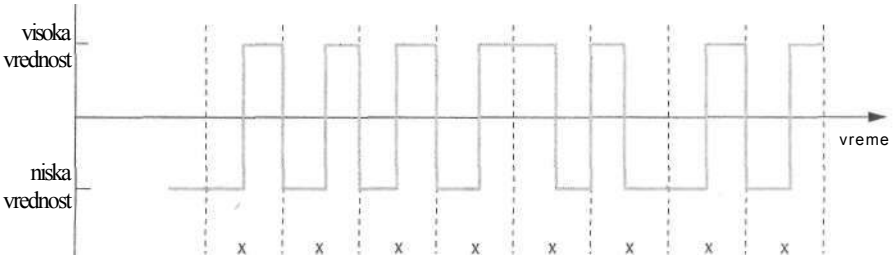
j. Korisnici mogu da koriste servis kablovske televizije za povezivanje na Internet i da istovremeno gledaju kablovsku televiziju dok neko od ukućana koristi konekciju.

3. Navedite razlike izmedu NRZ, Mančester i diferencijalnog Mančester digitalnog kodiranja.
4. Uz svu raspoloživu preciznu opremu, zašto dugački nizovi nula, ili jedinica predstavljaju problem kada se koriste šeme kodiranja, kao što je NRZ?
5. Kolika je brzina bauda neophodna za realizovanje prenosa podataka na brzini od IO Mbps korišćenjem NRZ kodiranja, a kolika u slučaju da se koristi Mančester kodiranje?
6. Navedite razliku izmedu Nikvistovih i Senonovih rezultata.
7. Defmišite koeficijent signal-šum.
8. Pretpostavimo da je prenos mogućbez ikakvih šumova. Da li to znači da nema granice za maksimalnu brzinu prenosa podataka koju je moguće postići pomoću tekuće opreme?
9. Uzimajući u obzir sve veće interesovanje za kompjutersku opremu koja koristi telefonske linije, zašto je opseg signala telefonskog servisa ograničen aproksimativno samo na 3.000 Hz?
10. Šenonovi rezultati povezuju brzine prenosa podataka sa opsegom signala u prisustvu šuma. Međutim, količina šuma zavisi od medijuma i izvora. Kako je to uzeto u obzir u Senonovim rezultatima?
11. Šta modem omogućava da uradite pomoću personalnog kompjutera?
12. U čemu je razlika između modulacije i demodulacije?
13. Navedite razlike izmedu frekventne, amplitudske i fazne modulacije.
14. Sta je kvadratuma amplitudska modulacija?
15. Navedite razlike izmedu impulsne kodne i impulsne amplitudske modulacije.
16. Zašto postoji mnogo različitih standarda za modeme?
17. Šta je konstelacija signala?
18. Šta je "inteligentni" modem?
19. Sta je šum kvantizacije?
20. Ako modem podržava nekoliko različitih standarda, kako zna koji treba da koristi kada se povezuje na drugi kompjuter preko telefonske linije?
21. Po čemu se DSL razlikuje od konvencionalnih i kablovskih modema sa stanovišta mogućnosti za povezivanje sa Internet provajderom od kuće?

22. Navedite razlike između konvencionalnog, kablovskog i DSL modema.
23. Šta je defmisanje diskretnih tonova?
24. U čemu je razlika između asimetričnog i simetričnog DSL servisa? Zašto ona posloji?

Vežbe

1. Digitalni signali mogu da se prevedu u analogne signale korišćenjem relativno jednostavnih tehnika, kao što je promena amplitude, ili frekvencije izmedu dve specifične vrednosti. U čemu je prednost korišćenja složenijih šema kao što je QAM?
2. Nacrtajte digitalne signale za niz bitova 0010100010, koristeći NRZ, Mančester i diferencijalno Mančester kodiranje. Pretpostavite da je pre prijema prvog bita signal bio na visokom nivou.
3. Koji je niz bitova pridružen sledećem signalu koji je kodiran Mančester šemom? Kako bi niz bitova izgledao da je korišćeno diferencijalno Mančester kodiranje?

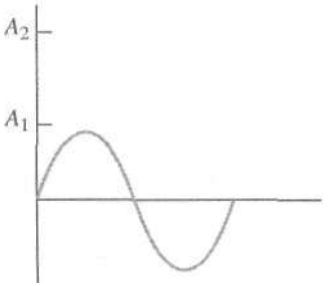


4. Nacrtajte analogne signale koji odgovaraju sledećim funkcijama:
   a. $y = \sin(t)$
   b. $y = \sin(2t)$
   c. $y = 4\sin(2t)$
   d. $y = 2\sin(2t + \pi/2)$
   e. $y = 3\sin(t)$
   f. $y = \sin(t + \pi/4)$
   g- $y = \sin(2t - \pi/2)$
5. Pretpostavimo da maksimalna frekvencija analognog signala na medijumu iznosi 6.000 Hz. Prema Nikvistovoj teoremi, kolike su bitske brzine moguće kod šema koje koriste jedan, dva, tri, ili četiri bita po komponenti signala?
6. Prema Nikvistovoj teoremi, kolika je frekvencija neophodna za podršku prenosa sa brzinom od 30.000 bps kada se koristi samo jedan bit po komponenti signala i kada se koriste tri bita po komponenti signala?
7. Sopstvenim rečima opišite značaj Senonovih rezultata.
8. Kolika je stvarna jačina signala (u odnosu na jačinu šuma) ako koeficijent signal-šum iznosi 60 decibela?
9. Koliko decibela iznosi koeficijent ako Je jačina signala dva puta veća od jačine šuma?

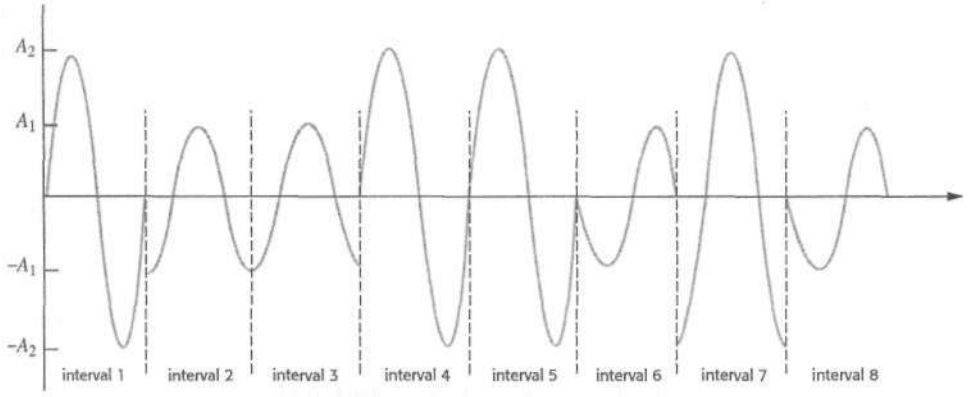
10. Pretpostavite da je maksimalni opseg signala na analognom medijumu 6.000 Hz. Prema Šenonovim rezultatima, kolika je maksimalna bitska brzina ako koeficijent signal-šum iznosi 40 decibela, a kolika u slučaju 60 decibela?
11. Prema Šenonu, koliki je opseg signala neophodan za podršku bitske brzine od 30.000 bps uz pretpostavku da koeficijent signal-šum iznosi 40 decibela? Koliki je opseg signala neophodan ako je broj decibela udvostručen?
12. Pretpostavimo da želimo da postavimo bitsku brzinu od 64.000 bps korišćenjem opsega signala od 10.000 Hz. Koliki je maksimalno dopušteni koeficijent signal-šum?
13. Da li se za razlikovanje signala može koristiti fazni pomak od jedne periode?
14. Da li je u slučaju kada se koristi QAM tehnika moguće imati isti signal sa dva različita intervala koji odgovara različitim vrednostima bitova?
15. Da li, u slučaju kada se koristi QAM tehnika, isti bitovi uvek odgovaraju istim analognim signalima?
16. Nacrtajte QAM analogni signal (pažljivo) koji prenosi sledeći niz bitova:

    0010110101011010 10110

    Pretpostavite da je tekući analogni signal uspostavljen kao ovde prikazani signal. Morate da nacrtate samo jedan kompletan ciklus za svaku modulacionu promenu.



17. Pretpostavimo da modem koristi kvadraturnu amplitudsku modulaciju, kao što je opisano u tabeli 3.3. Koja sekvenca bitova odgovara sledećem signalu (počevši od drugog vremenskog intervala)?



18. Dizajnirajte QAM tehniku koja koristi najviše osam faznih pomaka i dve amplitude. Koliko bitova postoji u okviru jednog bauda?
19. Pretpostavite da QAM tehnika ima najviše m faznih pomaka i n amplituda. Koliki je broj bitova po jednom baudu?
20. Ako imate CD plejer, proučite njegovu tehničku specifikaciju i povežite ono što pročitate sa raspravom o PCM-u.
21. Zašto nije moguće dizajnirati modeme sa proizvoljno velikim brzinama bauda tako da se realizuju neograničene bitske brzine?
22. Ako imate modem, napišite kraći pregled standarda koje on podržava.
23. Nacrtajte konstelaciju signala za modem koji koristi QAM definisan tabelom 3.3. Nacrtajte još jednu konstelaciju za QAM tehniku opisanu u vežbi 18.
24. Opišite promene signala (tj. defmišite promene amplitude i faze) pridružene konstelaciji signala sa slike 3.21.
25. Koliko bitova odgovara jednoj komponenti signala kada se koristi standard V.21? Koliko je trajanje jedne komponente signala?
26. Ponovite vežbu 25 za standard V.32.
27. Kao korisnik, objasnite zašto biste se odlučili za DSL konekcije ka Internetu, umesto za konekcije sa kablovskim modemom.
28. Kao korisnik, objasnite zašto biste se odlučili za konekciju ka Internetu sa kablovskim modemom, umesto za DSL konekciju.

Reference

[B199] Black, U. *ATM: Foundation for Broadband Netivorks,* 2nd ed. Englevvood Cliffs, NJ: Prentice HaIL 1990.

f Ch02] Chapra, S. i R. Canale. *Numerical Methods for Engineers.* New York: McGraw-Hill, 2002.

[Ge99] Gerald, C. i P. Wheatly. *Applied Numerical Analysis,* 6th ed. Reading, MA: Addison-Wesley, 1999.

[GoOl] Goralski, W. *ADSL and DSL Technologies.* New York: McGraw-Hill, 2001.

[He97] Hernandez-Valencia, E. J. "Architectures for Broadband Residential IP Services over CATV Netvvorks". *IEEE Network,* vol. 11, no. 1 (January 1997).

[StOO] Stallings, W. *DaIa and Computer Communications.* Englewood Cliffs, NJ: Prentice HaIl, 2000.

[Wa98] Walrand, J. *Communications Nettvorks: A First Course,* 2nd ed. New York: McGraw-Hlll, 1998.

POGLAVLJE

4 Uspostavljanje konekcija

Ideja da se informacije mogu sačuvati u svetu koji se neprestano menja bez gubljenja njihove vrednosti je pogrešna. To je skoro podjednako netačno kao da kažemo da posle rata možemo da uzmemo svo raspoloživo oružje, namaiemo ga uljem, obložimo gumenim omotačima i ostavimo da čeka sledeću ratnu uzbunu.

—Norbert Wiener (1894-1964), američki matematičar

4.1 Uvod

U poglavljima 2 i 3 predstavljeni su osnove prenosa podataka i specifični mehanizmi koji su neophodni za prenos informacija. U ovom poglavlju idemo korak dalje i predstavićemo komunikaciju. Možda ćete se zapitati u čemu je razlika između prenosa i komunikacije.

Da biste dobili odgovor, napravite analogiju sa ljudskim govorom. Možemo da analiziramo "mehanizme" govora - kako se glasne žice savijaju i proširuju da bi propustile vazduh koji dolazi iz pluća i formirale zvuk i kako se ti zvuci artikulišu u ustima u oblik koji smatramo govorom. Osim toga, mogli bismo da raspravljamo o jeziku i poreklu raznih reči koje danas koristimo. Ipak, sve to je daleko od komuniciranja. Reči koje izlaze iz usta moraju da budu organizovane, tako da imaju smisla. Ako se izgovaraju suviše brzo, ili presporo, nećemo razumeti onog ko govori. Ako više ljudi govori istovremeno, velika je verovatnoća da nikoga nećemo razumeti. Ako Vas niko ne sluša, ili ako neko govori jezikom koji ne razumete, komunikacija je izgubljena. Ako u rečenici nedostaju reči, ili fraze (što može da se desi kada neko ne govori maternjim jezikom), može doći do gubitka značenja.

Slični problemi postoje i kod elektronskih komunikacija. Prijemnik mora da zna kako su organizovani bitovi u poruci da bi mogao da je razume. Osim toga, on mora da zna i kojom brzinom bitovi stižu da bi mogao da interpretira poruku. Šta se dešava ako više ljudi pokušava istovremeno da koristi zajednički medijum, što je čest slučaj u lokalnim mrežama (LAN)?

Ako svi uređaji istovremeno na isti način pokušavaju da pošalju podatke, može doći do kolizije signala i komunikacija neće biti uspešna. Mora se implementirati neki redosled, tako da se omogući neometana komunikacija između uredaja.

U ovom poglavlju se bavimo tim temama. Odeljak *4.2* počinje predstavljanjem nekih nosača 1 popularnih uredaja koje ljudi često koriste. U odeljku 4,3 obradićemo nadne za komuniciranje korišćenjem serijskog, paraielnog, sinhronog i asinhronog prenosa. Osim toga, predstavljene su razlike izmedu jednosmernih i dvosmernih komunikacija. Najbolji način da se osiguraju kompatibilno slanje i prijem podataka između uređaja je poštovanje standarda. LJ odeljku 4.4 obradićemo dobro uspostavljene standarde koji se najčešće koriste za povezivanje uredaja, kao što su EIA-232 i EIA-449, i neke novije standarde, kao što su USB (universal serial bus univerzalna serijska magistrala) i FireWire, koji se sve češce sreću na novijim personalnim kompjuterima. LI odeljku 4.5 predstavićemo kako više uredaja može istovremeno da koristi zajednički medijum pomoću tehnika multipleksiranja, a u odeljku 4.6 upoznaćete dva uobičajena digitalna noseća sisteraa: Tl i SONET. Međutim, postoji veliki broj slučajeva u kojima multipleksiranje nije opcija (nesumnjivo u LAN mrežama); tada je uredaju neophodna ekskluzivna kontrola nad medijumom da bi komunikacija bila moguća. Dakle, uredaji se "nadmeću" za dobijanje ekskluzivne kontrole nad medijumom; u odeljku *4.7* predstavićemo nekoliko pristupa koji se koriste u LAN standardima.

Glavna činjenica koju treba imati na umu dok se čita ovo poglavlje je da obično postoji veći broj uređaja koji žele medusobno da komuniciraju i da često moraju da dele zajednički medijum. Način na koji se deljenje medijuma organizuje umnogome odreduje uspostavljanje smisaonih komunikacija.

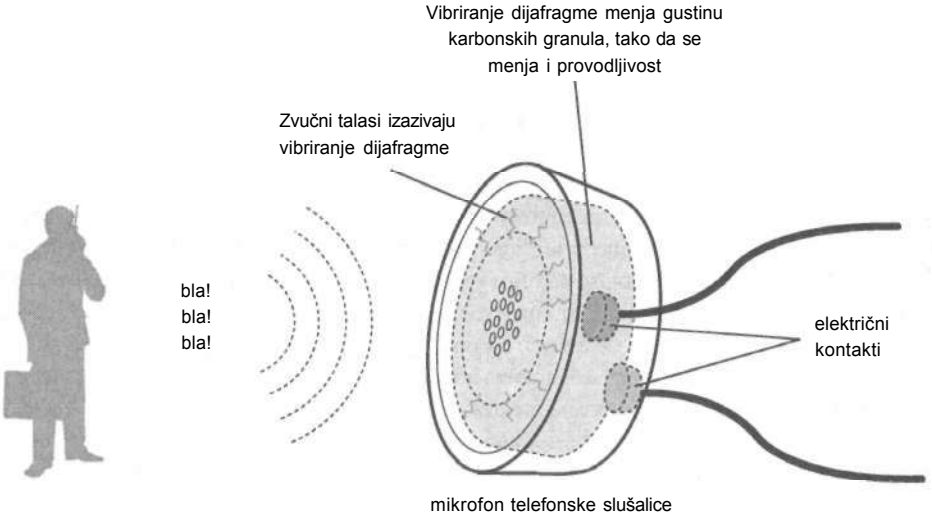
4.2 Nosioci i uređaji za uspostavljanje komunikacije

Telefonski sistem

Od svih izuma u prošlom veku telefon je sigurno imao najvećeg uticaja na naše živote. Mogućnost pozivanja nekoga ko se nalazi na bilo kom kraju sveta biranjem nekoliko brojeva bila je apsolutno neverovatna. Ali, telefon je omogućio i više od obične glasovne konverzacije između prijatelja i rodaka; postao je neprocenjiv u obavljanju poslova kada je počeo da se koristi za kompjuterske komunikacije. Mogućnost prenosa informacija u okviru kompjuterske mreže, ili faksom sada predstavlja nešto sasvim normalno.

Telefon funkcioniše tako što konvertuje zvuk u električnu energiju. Ono što mi registrujemo kao zvuk izazvano je malim fluktuacijama u vazdušnom pritisku (zvučni talasi). Talasi "putuju" kroz vazduh, izazivajuđ vibnranje nekih objekata. Isti princip izaziva podrhtavanje starih prozora, ili lustera za vreme oluje. Osim toga, omogućava nam da čujemo - talasi izazivaju vibriranje bubne opne U našim ušima, tako da se signali šalju do mozga.

Telefon upotrebljava nekoliko metoda za konvertovanje zvuka u električnu energiju. Prvi metod koristi mikrofon u telefonskoj slušalici, koji se sastoji od komore ispunjene karbonskim granulama (slika 4.1). Sa komorom su povezana dva električna kontakta, Dok govorite u slušalicu, zvučni talasi izazivaju vibriranje dijafragme koja prekriva komoru. Zbog vibracija, dolazi do promena pritiska na karbonskim granulama. Veći pritisak izaziva veće zbijanje granula, tako da postaju bolji provodnici elektriciteta.



SLIKA 4.1 *Konvertovanje zvučnih talasa u električne signale*

Manji pntisak ima suprotan efekat. Krajnji rezultat je promenljiva kohcina elektnciteta koju rzaziva zvuk koji se vodi u slušalicu. Na prijemnoj strani elektricitet aktivira zvučni namotaj, tako da pridruženi zvučnik vibrira. Vibrirajući zvučnik izaziva promene u vazdušnom pritisku, koje se interpretiraju kao zvuk.

Drugi metod koristi mikrofon sa provodnom metalnom folijom (*foil-electret condenser microphone).* Takođe sadrži dijafragmu, koja vibrira kada je "pogode" zvučni talasi. Razlika je u tome što dijafragma prekriva šuplji disk i razdvojena je od perzistentne elektrode. Dijafragma je naelektrisana i ima metalni omotač (elektret) na jednoj strani, iako se u nekim slučajevima takav omotać može sresti i na elektrodi. Kada dijafragma vibrira, menja se električno polje izmedu nje i elektrode i dolazi do promene kapacitivnosti izmedu njih. Tako se indukuje struja koja se generiše iz elektrode. Ova tehnologija se koristi i kod slušnih aparata i mikrofona-bubica.

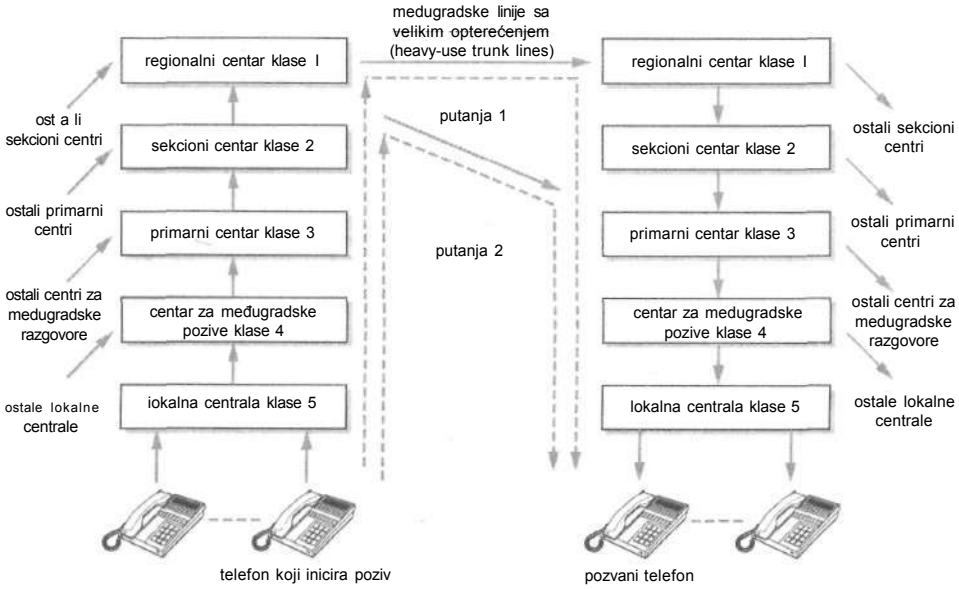
Da bi se postavio poziv, unosi se sekvenca brojeva okretanjem brojčanika, ili pritiskanjem tastera. Svaka cifra šalje kod do telefonske centrale koja interpretira kodnu sekvencu i utvrduje odredište. Ako postoji raspoloživa ruta do odredišta i ako linija nije zauzeta, šalju se dva signala. Prvi ide do odredišta i izaziva zvonjenje telefona. Drugi ide do izvora i obaveštava osobu koja inicira poziv da telefon na drugom kraju zvoni.

Pošto su signali razdvojeni, osoba koja poziva ne čuje da telefon zvoni. U stvari, zbog kašnjenja u kolima, možda nećete ni stići da čujete zvono. Možda se dešavalo da Vam se neko javi pre nego što čujete da je telefon zvonio. Često se misli da su za to odgovorni gremlini u liniji, ili neke mistične moći telefonskog sistema. Sada znate da je to samo zato što je neko odgovorio pre nego što se signal vratio do Vas.

Kod svake cifre zavisi od toga da li imate tonsko, ili pulsno biranje. Kod tonskog biranja svaka cifra šalje jedan ton, koji se sastoji od jedinstvenog para frekvencija. Kod pulsnog biranja svaka cifra generiše 1 do 10 impulsa. Svaki impuls odgovara otvaranju, ili zatvaranju kola, slično pritiskanju tastera za prekid veze. U stvari, ako imate dovoljno brze prste, možete da birate broj brzim pritiskanjem tastera odgovarajući broj puta.

RUTIRANJE POZIVA Način na koji se telefonski pozivi rutiraju predstavlja neverovatni podvig inženjeringa. Ovde govorimo o mreži koja povezuje milione korisnika. Prvi njen deo je lokalna petlja. Čine je telefoni priključeni na bakarne vodove koji su postavljeni od telefonskog stuba, ili na podzemne kablove koji vode do lokalne telefonske centrale.* Lokalna telefonska centrala sadrži prekidačku logiku i utvrduje gde treba rutirati poziv. Ako se poziva broj koji pripada istoj centrali (prve tri cifre u broju telefona), konekcija može da se uspostavi direktno do destinacije. O suprotnom, strategija rutiranja zavisi od toga gde treba rutirati poziv.

Na slici 4.2 prikazane su glavne komponente (centri) telefonske mreže. Regionalni centri klase 1 su malobrojniji, a "pokrivaju" veće oblasti (obično više država).



SLIKA 4.2 *Telefonska mreža*

* Kao sinonimi za lokalnu centralu ovde su često korišceni termini krajnja centrala, telefonska centrala, ili centrala klase 5.

Tabela 4.1: Glavni centri u telefonskoj mreži

Centar	Vlasnik	Pokrivena oblast
regionalni centar klase 1	noseći sistemi na velikim rastojanjima	više država
sekcioni centar klase 2	noseći sistemi na velikim rastojanjima	jedna država
primarni centar klase 3	lokalna kompanija, ili noseći sistemi na velikim rastojanjima	metropola
centar za medugradske pozive klase 4	lokalna kompanija	Jedan, ili više gradova
lokalna centrala klase 5	lokalna kompanija	susedstvo

Klase 2, 3, 4 i 5 su sve brojnije i "pokrivaju" manje oblasti. Noseći sistemi na velikim rastojanjima poseduju centre koji "pokrivaju" velike oblasti, dok lokalne kompanije poseduju ostale centre. U tabeli 4.1 dat je pregled ovih informacija. Ukratko ćemo predstaviti ulogikovih centara u složenom sistemuza rutiranje telefonskih poziva. Ako želite detaljnije informacije, pogledajte reference [LeOO] i [Sh90],

Centri klase 1 povezuju više centara klase 2. Slično tome, centri klase 2 povezuju više centara klase 3 i tako redom. Na vrhu hijerarhije centri klase 1 su povezani dalekovodima visokog kapaciteta, koji mogu da prenesu više telefonskih konverzacija istovremeno. U opštem slučaju, dva telefona mogu da se povežu praćenjem hijerarhije od lokalne centrale do regionalnog centra, preko međugradskih linija do drugog regionalnog centra i nazad do odgovarajuće lokalne centrale (putanja 1 na slici 4.2). Ipak, ovo nije uvek najbolja ruta. Idelano bi bilo da poziv prolazi kroz minimalan broj centara. Ekstremna situacija podrazumeva direktno povezivanje svih parova telefona u mreži, ali, naravno, to nije realno. U nekim slučajevima, mnogi telefonski pozivi mogu da se dese izmedu odredenog sekcionog centra i primarnog centra. Tada je korisno postaviti još jednu medugradsku liniju visokog kapaciteta, koja će omogućiti spajanje tih centara i tako obezbediti alternativne rute (putanja 2 na slici 4.2).

Postoji više međugradskih linija koje povezuju različite klase centara. Ove konekcije, koje, uglavnom, odreduje saobraćaj između dve oblasti, obezbeduju veći broj alternativnih ruta. U manje verovatnom slučaju da sve linije funkcionišu na maksimalnom kapacitetu neće biti moguče proslediti poziv i tada se inicijatoru poziva šalje signal zauzeća. Ipak, sa današnjim hardverskim i softverskim resursima, ovakve situacije su malo verovatne.

Privatne centrale

Osim javnog telefonskog sistema, postoje i privatni telefonski sistemi, poznati kao privatne telefonske centrale (PBX - private branch exchange). Cesto se koriste i nazivi *privatne automatske centrale* (PABX - *private automatic branch exchange)* i kompjuterske centrale (CBX - computer branch exchange). PBX je kompjuter dizajniran za rutiranje telefonskih poziva unutar kompanije, ili organizacije. Ovaj sistem je veoma koristan većim organizacijama, kod kojih je neophodno obezbediti kontakte između brojnog osoblja.

PBX obezbeduje kompletnu kontrolu nad govornim komunikacijama i razrnenom podataka, umesto da se oslanja na podršku telefonske kompanije. Ovaj sistem ima svoje prednosti i nedostatke. Organizacija mora da plati specijalizovani hardver, softver i osoblje koje će održavati sistem. Sa druge strane, ako je kompanija velika i ima velike zahteve, ovo može da bude isplativa varijanta za uspostavljanje komunikacija.

Medutim, PBX je nešto više od telefonskog sistema. Kada se on instalira, kablovi povezuju sve kancelarije, konferencijsku salu, ili lokacije na kojima se mora koristiti telefon. Zato se dizajneri često odlučuju na postavljanje dodatnih vodova sa upredenim paricama, koaksijalnih kablova, ili optičkih fibera. Oni treba da budu na raspolaganju za prenos podataka izmedu kompjutera, ili periferija. U mnogim slučajevima se dodatni vodovi instaliraju čak i kada nema trenutne potrebe za prenos podataka. Dodatna cena instalacije redundantnih vodova je daleko manja od novih instalacija u budućnosti.

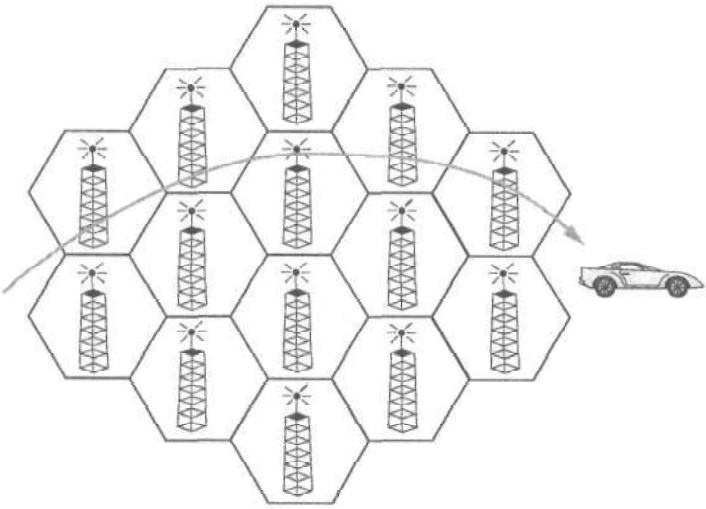
Brojne funkcije PBX-a odgovargju funkcijama lokalne mreže, ali između njih postoje značajne razlike. Na primer, kod lokalnih mreža obično postoji mogućnost radio difuzije. To znači da jedan uredaj može da šalje poruku svim ostalim uredajima, ili odredenoj grupi uredaja na mreži. PBX se obično koristi za point-to-point komunikacije. Sa druge strane, PBX može da definiše kolo izmedu komunikacionih uređaja, nešto što nije tipično za LAN. Ova karakteristika je prednost ako je neophodno brzo preneti veću količinu podataka između dva uređaja. Detaljnije predstavljanje PBX-a i poredenje sa LAN mrežom nisu tema ove knjige. Ako ste zainteresovani, pročitajte više detalja u referenci [Sh90].

Mobilni telefoni

Sigurno već poznajete tehnologiju koja je uveliko sastavni deo života mnogih ljudi - mobilne (celularne) telefone. Pre svega, ova tehnologija korisniku omogućava komuniciranje preko telefonskog sistema i kada je daleko od tradicionalnih telefonskih uredaja. Za razliku od nekih mišljenja da se termin celularni (ćelijski) tiče biološkog fenomena kod koga bi na nečijoj glavi narastao neki komunikacioni uređaj, reč je o deljenju geografske oblasti na ćelije. Oblast je podeljena na više regiona, ili ćelija (slika 4.1), gde svaka ćelija ima prijemni i predajni toranj. Centrala mobilne telefonske mreže (MTSO - mobile telephone switching office) ima kompjuter koji kontroliše sve ćelije i povezuje ih sa telefonskim sistemom.

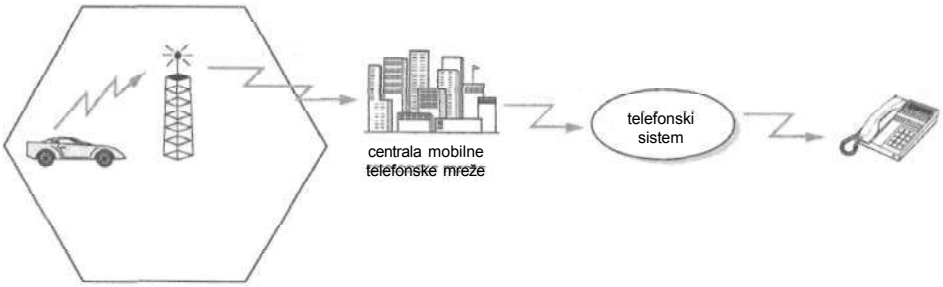
Mobilni telefon je, u stvari, dvosmerni radio koji može da komunicira sa tornjern. Na granicama izmedu ćelija mobilni telefon može eventualno da bude izvan dometa nekoliko tornjeva. Svaki toranj neprestano emituje signale, tako da telefon može na osnovu jačine signala da utvrdi koji je toranj najbliži. Kada se inicira poziv sa mobilnog telefona, uspostavlja se komunikacija sa najbližim tornjem (slika 4.4). Sa druge strane, toranj komunicira sa MTSO, preko koje može da se poveže na tradicionalni telefonski sistem.

Primanje poziva je nešto složenije, jer ne postoji način da se zna u kojoj se ćeliji konkretni telefon trenutno nalazi. Ipak, svaki mobilni telefon ima svoj jedinstveni identifikacioni broj. Kada se taj broj pozove, MTSO ga prenosi do svih tornjeva pod svojom kontrolom. Nakon toga, svi tornjevi emituju primljeni broj. Pošto mobilni telefon neprestano nadgleda emitovanje, on registruje svoj ID i odgovara. Kada toranj detektuje odgovor, prenosi ga ka MTSO, koja kompletira konekciju.



SLIKA 4.3 *Mobilna mreža*

Potencijalni problem se javlja kada se korisnik premešta u susednu ćeliju. U tom slučaju, toranj sa kojim korisnik komunicira može da bude izvan dometa. MTSO nadgleda signale koje prima od tornjeva iz ćelija; ako detektuje da signal postaje sve slabiji, može da prebaci komunikaciju na drugu ćeliju. Ovo se naziva prenošenje *[handoff),* koje se vrši veoma brzo, tako da nema primetnog prekida u razgovoru. Medutim, ako se telefon koristi za prenos podataka, može doći do gubljenja odredenih podataka.



SLIKA 4.4 *Komunikacija mobilnim telefonom*

Naravno, nameće se logično pitanje šta se dešava ako mobilni korisnik prelazi u ćeliju koja je pod kontrolom druge MTSO centrale, odnosno, ako prede u oblast koja je "pokrivena" drugom mobilnom mrežom. U tom slučaju se koristi roaming. Pretpostavimo da putnik prelazi u oblast drugog operatera mobilne telefonije. Ako je ta nova oblast kompatibilna sa matičnom mrežom putnika, pozivi koje su mu upućeni biće rutirani u oblast u kojoj se trenutno nalazi. Naravno, takva usluga se pruža po specijalnoj ceni. To je pomalo slično lutanju kroz šumu kada se neprestano bacaju kamenčići. Praćenjem bačenih kamenčića svako može da sledi Vaš trag i tako da Vas nade. Glavna razlika izmedu roarninga i prepuštanja je to što roaming uključuje usluge drugog operatera; to nije slučaj kod prepuštanja.

Korišćenje mobilnog telefona za razmenu podataka nameće sliku nekoga ko se utrkuje na autoputu sa kompjuterom, ili faks mašinom u krilu. Ako to i sami radite, ili znate nekoga ko to radi, kažite mi da ne idem nikada tim putevima. Medutim, poneki transferi podataka nisu toliko bizarni kao što možda u početku izgledaju. Na primer, ambulantna kola imaju legitimnu potrebu za prenosom podataka kada se bolnica obaveštava da stižu sa osobama povredenim u nekoj nesreći. Medutim, da bi se obeshrabrilo korišćenje mobilnih telefona u toku vožnje, u nekim državama postoje zakonske regulative kojima je ono zabranjeno, osim u hitnim slučajevima.

Faks mašine

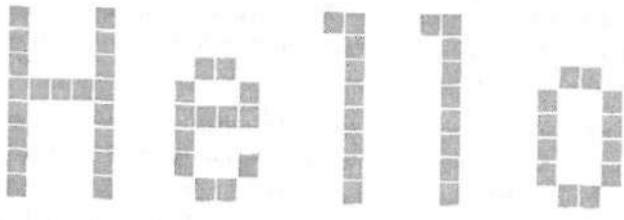
Sledeći uredaj koji je postao popularan krajem 80-ih godina prošlog veka je **faksimil (faks)** mašina (slika 4.5). Može da prenosi crteže, pisma, ili dijagrame preko telefona za samo nekoliko sekundi i zato je za mnoge postao uredaj od neprocenjivog značaja.

Faks mašine su zasnovane na principu sličnom onome koji se koristi za prikazivanje slike na video ekranu PC-ja, na televizoru, ili na fotografiji. Slika koja izgleda kao da je sastavljena od linija i boja u stvari je kolekcija tačaka, ali su one toliko male da ne mogu da se razaznaju, ako ne primaknete nos sasvim uz ekran.

Faks mašina funkcioniše tako što uzima sliku koja se šalje i konvertuje je u binarni format. Ubacujete list papira u faks mašinu kao da je reč o kopir aparatu. Strana se deli na veliki broj tačaka (još se nazivaju i **elementi slike,** ili **pikseli),** koje predstavljaju delove papira. Ovo nazivamo *bitmapirana reprezentacija,* jer se svaka tačka pamti kao jedan bit podataka. Svaka tačka je cma, ili bela (binarno 0, ili 1), u zavisnosti od toga šta se nalazi na papiru. Nakon toga, tačke se prenose kao binarni podaci i ponovo se spajaju na prijemnoj strani. Na primer, na slici 4.6 prikazana su slova reči "Hello" u bitmapiranoj reprezentaciji. Radi jednostavnosti, prikazali smo sarao par velikih tačaka; tipična bit mapa može da koristi 200 tačaka po inču. Veći broj tačaka obezbeduje bolji kvalitet prenosa. Manje tačaka daje granularniju primljenu sliku.

Većina faks mašina ne uzima jednostavno sliku i prenosi rezultujuće tačke. Ako bi se nastavilo na ovakav način, bilo bi neophodno ogromno vreme prenosa za jednostavnije dokumente. Na primer, pretpostavimo da faks prepoznaje 200 tačaka po inču. Malo računice pokazuje da postoji 200 200, ili 40.000 tačaka po jednom kvadratnom inču. Medutim, tipična strana papira iznosi 8,5x11 inča, ili 93,5 kvadratnih inča. Sa 40.000 tačaka po kvadratnom inču, tipična stranica papira zahteva oko 40.000x93,5, ili 3.740.000 tačaka.* Sa uobičajenom brzinom prenosa kod slanja od 33,6 Kbps, potrebno je 3.740.000/33,6, ili skoro dva minuta, da bi slika bila poslata sa jednog lista papira. Vaša prva reakcija može da bude: "Koristio sam faks mašinu i nisam primetio da traje toliko dugo." U pravu ste.

SLIKA 4.5 *Faks mašina*



SLIKA 4.6 *Bitmapirana reprezentacija reči "Hello"*

* Ovo su samo aproksimativne brojke. Više specifičnih informacija možete da pročitale u odeljku 5.3.

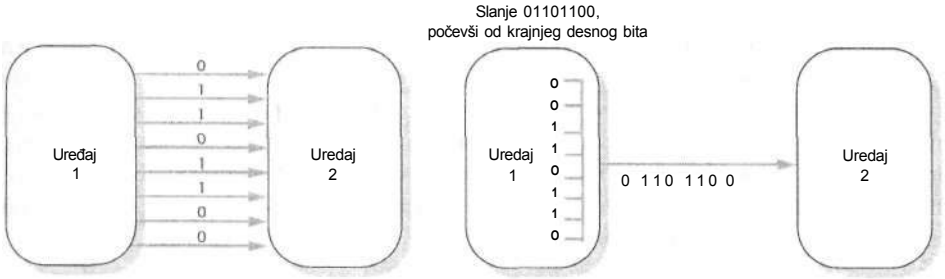
Većini faks mašina ne treba toliko vremena, jer koriste kompresiju podataka. Umesto da se šalje svaka tačka individualno, faks grupiše tačke i definiše ekvivalentnu binarnu reprezentaciju za grupu, koristeći manje bitova. Na primer, pretpostavimo da deo slike ima 800 crnih tačaka u nizu. Umesto da se šalje 800 crnih tačaka, možete da pošaljete 1 crnu tačku ispred koje se navodi broj 800. Pošto je za 800 binarna reprezentacija 1100100000 (10 bitova), za **prenos je** neophodno 11 bitova (ne zaboravite jedan bit za tačku). Jasno, ovo je značajna redukcija. Postoje i drugi načini za kompresovanje podataka; u Poglavlju 5 detaljnije ćemo predstaviti kompresiju podataka.

4.3 Modovi prenosa

Modovi prenosa definišu način na koji se grupa bitova prenosi od jednog uredaja do drugog. Osirn toga, definišu da li bitovi mogu da "putuju" u oba smera istovremeno, ili uredaj mora naizmenično da šalje i prima podatke.

Serijski i paralelni prenos

Najpre ćemo istaći razliku izmedu serijskog i paralelnog prenosa. **Paralelni prenos** znači da se grupa bitova prenosi istovremeno korišćenjem zasebnih linija (žica) za svaki bit (slika 4.7a). Linije su, obično, upletene u kabl. Paralelni prenosi su česti, posebno ako je rastojanje izmedu uredaja kratko. Na **primer,** konekcija izmedu PC-ja i štampača koji se ne nalaze na rastojanju većem od 25 stopa smatra se bezbednim rastojanjem. Najčešći primer je komunikacija izmedu kompjutera i perifernih uredaja. Medu druge primere ubrajaju se komunikacije izmedu CPU-a, memorijskih modula i kontrolera uredaja.



SLIKA 4.7 *Paralelni i serijski prenos*

Paralelni prenos gubi prednosti na većim rastojanjima. Prvo, konšćenje veceg broja hmja na većim rastojanjima je skuplje od korišćenja samo jedne linije. Drugo, prenos na većim udaljenostima zahteva deblje žice da bi bilo redukovano degradiranje signala. Uplitanje takvih žica daje "nezgrapan" kabl. Trecl problem je vreme potrebno za prenos bitova. Na kraćim rastojanjima bitovi koji se šalju istovremeno mogu da se primaju takode skoro istovremeno. Medutim, na većim rastojanjima otpornost žica može da izazove manja pomeranja bitova, tako da stižu u različito vreme, što može da prouzrokuje probleme na prijemnoj strani.

Serijski prenos obezbeduje alternativu za paralelni prenos (slika 4.7b). Koristi se samo jedna linija, preko koje se bitovi prenose jedan za drugim. To je jeftinije i pouzdanije od paralelnog prenosa na većim rastojanjima. Osim toga, prenos je sporiji, jer se prenosi samo po jedan bit u jednom trenutku.

Uredaje za slanje i prijem odlikuje dodatna složenost. Pošiljalac mora da utvrdi redosled kojim se bitovi šalju. Na primer, kada se šalje osam bitova iz jednog bajta, pošiljalac mora da utvrdi da li se najpre šalje bit najveće, ili najmanje težine. Slično tome, prijemnik mora da zna gde da postavi bit koji prvi primi u okviru odredišnog bajta. Ovo možda izgleda kao trivijalan problem, ali različite arhitekture mogu drugačije da numerišu bitove u bajtu; ako ne postoji saglasnost protokola o redosledu bitova, informacije neće biti ispravno prihvaćene.

Asinhroni, sinhroni i izohroni prenos

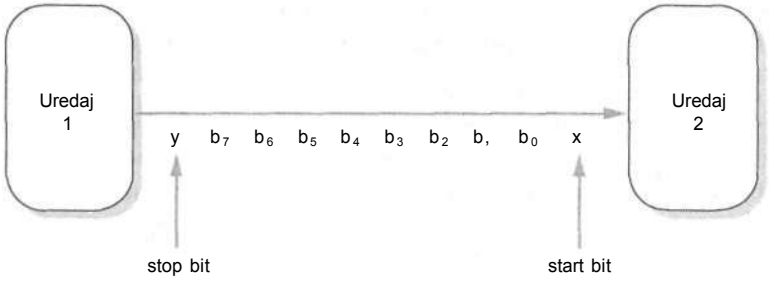
Postoje tri načina za obezbedivanje serijskih komunikacija: asinhroni, sinhroni i izohroni prenos. Pod asinhronim prenosom se podrazumeva nezavisni prenos bitova podeljenih u manje grupe (obično bajtove). Pošiljalac može da pošalje grupe u bilo koje vreme i prijemnik nikada ne zna kada stižu (pomalo slično poseti davno zaboravljenog rodaka).

Jedan primer (nekada uobičajen) koristi terminal za komunikaciju sa kompjuterom. Pritiskom na taster koji predstavlja slovo, broj, ili specijalni karakter šalje se 8-bitni ASCII kod.* Terminal šalje kodove u bilo koje vreme, u zavisnosti od toga koliko dobro, ili brzo kucate. Interno, hardver mora da ima mogućnost prihvatanja unetih karaktera u bilo kom trenutku. Treba da istaknemo da se za sve tastere sa tastature koristi asinhroni prenos. Mnogi PC-ji koriste softver za emulaciju terminala prilikom povezivanja na udaljene kompjutere i mogu da baferuju unete vrednosti i prenesu celu liniju, ili ekran do kompjutera. Ovo je sinhroni prenos, koji ćemo ukratko predstaviti.

Ulaz preko terminala nije jedini primer asinhronog prenosa. U nekim slučajevima se podaci šalju na linijski štampač po bajtovima. Asinhroni prenos je tipični *bajtu-orijentisani ulaz-izlaz* (I/O), termin koji ukazuje da se podaci prenose po jedan bajt u jednom trenutku.

Kod asinhronih prenosa postoji jedan potencijalni problem. Zapamtite da prijemnik ne zna kada će podaci stići do njega. Kada on detektuje primljene podatke i bude spreman da reaguje, prvi bit je već došao i otišao.

*Nismo zaboravili da smo defmisali ASCII kod kao 7-bitni kod. Međutim, 7-bitni kod može da se posmatra kao 8-bitni kod, gde Je vodeći bit uvek O. To je slično nekome ko iznenada staje iza Vas i počinje da govori. Kada reagujete i počnete da slušate, verovatno ste već.



SLIKA 4.8 *Asinhrona komunikacija*

propustih nekoliko reči. Zato svaki asinhroni prenos započinje start bitom (slika 4.8), koji upozorava prijemnika da podaci stižu. Tako prijemnik ima vremena da reaguje, da prihvati i da baferuje bitove podataka. Na prijemnoj strani stop bit ukazuje na kraj prenosa. Po konvenciji, neaktivna linija (preko koje nema prenosa) prenosi, u stvari, signal koji definiše binarnu jedinicu. Nakon toga, start bit dovodi do promene signala, što odgovara O. Preostali bitovi mogu da izazovu promenu signala u zavisnosti od vrednosti bitova. Konačno, stop bit vraća signal nazad na ekvivalent jedinice i zadržava to stanje sve dok ne stignu naredni bitovi.

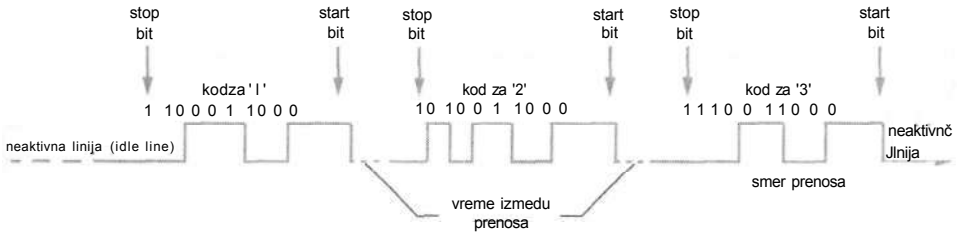
Na primer, pretpostavimo da ste uneli cifre 321 preko terminala. Korišćenjem 8-bitnog proširenog ASCII koda (sa vodećom 0) definišu se sledeći bitovi za slanje:

00110001 zacifru 1

00110010 zacifru 2

00110011 zacifru3

Pretpostavimo da šaljemo svaku cifru (prvo krajnji levi bit), nezavisno koristeći NRZ kodiranje Na slici 4.9 prikazan je preneti signal. U svakom slučaju start bit podiže signal, upozoravajući na dolazak bitova.



SLIKA 4.9 *Asinhroni prenos cifara 1,2x3 pomoću NRZ kodiranja*

Kada stignu svi bitovi za konkretnu cifru, stop **bit** spušta signal. Ostaje na niskom nivou sve do pojave sledećeg start bita, koji ponovo podiže signal.

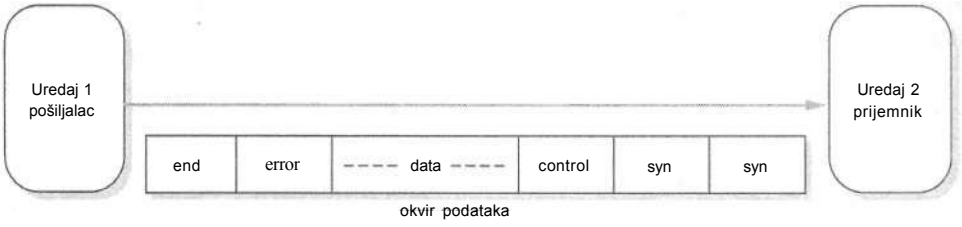
Asinhroni prenos je dizajniran za korišćenje na sporijim uređajima, kao što su tastature, ili neki štampači. Ispoljava velika prekoračenja. U primeru koji smo prethodno dali za svakih osam bitova prenose se dva dodatna bita. Ovo predstavlja povećanje od 25 odsto u ukupnom prenosu (kod sporijih uredaja koji prenose dosta podataka to predstavlja značajno povećanje).

Kod **sinhronog prenosa** šalju se mnogo veće grupe bitova. Umesto da se šalje više karaktera nezavisno, svaki sa svojim start i stop bitom, karakteri se grupišu i prenose u celini. Ovu grupu možemo da nazovemo okvir podataka **(data frame),** ili, jednostavno, okvir. Precizna organizacija okvira zavisi od protokola koji se koristi. Međutim, okviri nemaju mnogo zajedničkih karakteristika. Na slici 4.10 prikazana je organizarija generičkog okvira podataka. Orijentacija je postavljena u odnosu na krajnji desni bit.

Prvi deo okvira sadrži **SYN karaktere,** jedinstveni uzorak bitova koji upozorava prijemnik da okvir dolazi. SYN karakter je sličan prethodno predstavljenom start bitu, osim što definisani uzorak obezbeduje da učestalost semplovanja bude konzistentna sa brzinom kojom bitovi dolaze.

Zatim, slede kontrolni bitovi, koji mogu da uključe sledeće elemente:

- Izvorna adresa: Definiše odakle okvir potiće.
- Odredišna adresa: Definiše gde okvir treba da ide. Ovo je značajno u mrežama u kojima okvir može da prođe kroz više čvorova na putu do svog odredišta. Svaki posrednički čvor koristi odredišnu adresu da bi utvrdio kuda treba da rutira okvir. Rutiranje je detaljnije obrađeno u Poglavlju 10.
- Stvarni broj bajtova



syn = bltovi sinhronizacije
control ~ kontrolni bitovi
data = bltovi podataka
error = bitovi za proveru grešaka
end = bitovi za označavanje kraja okvira

SLIKA 4.10 *Sinhroni prenos*

- Broj sekvence: Ovo je korisno kada se šalje veći broj bitova i ako oni zbog nečega ne stižu na odredište u ispravnom redosledu. Prijemnik tada koristi brojeve sekvenci za ponovno sastavljanje podataka. O ovome će biti više reči u Poglavlju 8.
- Tip okvira: Razlikuje se od protokola do protokola. U Poglavlju 8 predstavićemo neke tipove.

Bitovi podataka defmišu informacije koje se šalju. Nema start i stop bitova izmedu karaktera. Bitovi za proveru grešaka se koriste za detektovanje i korekciju grešaka koje nastaju u toku prenosa. Kao što ste saznali u poglavljima 2 i 3, električna interferenca može da izobliči signale. Kako prijemnik zna kada se to desilo? Tipično, pošiljalac prenosi ekstra bitove koji zavise od podataka. Ako su podaci promenjem, ekstra bitovi nisu konzistentni sa podacima. U Poglavlju 6 predstavićemo detekciju grešaka i tehnike za njihovo korigovanje.

Poslednji deo okvira je marker kraja okvira. Poput SYN karaktera, to je jedinstveni niz bitova koji ukazuje da neće stići novi bitovi (bar ne do početka nekog novog okvira).

Sinhroni prenos je u opštem slučaju mnogo brži od asinhronog. Prijemnik ne prima start i stop bitove za svaki karakter. Kada detektuje SYN karaktere, sve ostale bitove prima odmah čim stignu. Osirn toga, manje je opterećenje. Na primer, tipični okvir može da ima 500 bajtova (4.000 bitova), sa još 100 dodatnih bitova (ovaj broj varira). U ovom slučaju dodatni bitovi znače povećanje od 2,5 odsto u ukupnom broju prenetih bitova. Uporedite to sa povećanjem od 25 odsto kod asinhronog prenosa.

Kako se broj bitova povećava, tako se procenat prekoračenja smanjuje. Sa druge strane, više polja podataka zahteva veće bafere za njihovo smeštanje, što stvara ograničenje za veličinu okvira. Osim toga, veći okviri zauzimaju medijum u dužem neprekidnom intervalu. U ekstremnim slučajevima ovo može da prouzrokuje preterano kašnjenje za druge korisnike. I pored sinhrone prirode bitova unutar okvira, postoji i asinhrona karakteristika prilikom slanja okvira.

Treči mod prenosa koji se sve češće koristi je izohroni prenos. I asinhroni i sinhroni prenos imaju jednu zajedničku karakteristiku: posmatrano u dužim vremenskim periodima, podaci ne moraju da stižu fiksnom brzinom. Čak i ako podaci iz jednog okvira stižu fiksnom brzinom, može da postoji proizvoljan razmak između okvira, što daje asinhronu crtu prenosu okvira. Zato se može desiti da u odredenim trenucima dolazi do iniciranja slanja velikog broja okvira. Osim toga, ako metodi za detektovanje grešaka otkriju grešku, obično se traži ponovno slanje okvira, što utiče na ukupnu brzinu transfera. U mnogim slučajevima, kao što je transfer fajlova i Web aplikacije, to je u redu. Mnogo je značajnije ispravno preneti informacije, nego voditi računa o manjim kašnjenjima.

Medutim, real-time aplikacije zahtevaju različit kvalitet servisa (QoS). Primeri uključuju zavisnost prikazivanja video slika u realnom vremenu, ili slušanja radio stanice (tj. Internet radija) od brzine preuzimanja bitova. Na primer, standardi za prikazivanje televizije zahtevaju da se TV slika javlja brzinom od 30 slika u sekundi - ni manje, ni više. Signali moraju da stižu fiksnom brzinom; nema drugih opcija. Sledeći primer je Web kamera. Video kamera je postavljena na određenoj lokaciji, digitalizuje slike u zapise i podaci se prenose preko Interneta; korisnik vidi slike odmah čim se slika pojavi. Slike moraju da stižu brzinom koja je prikladna za prikazivanje; kašnjenje nije dopušteno.

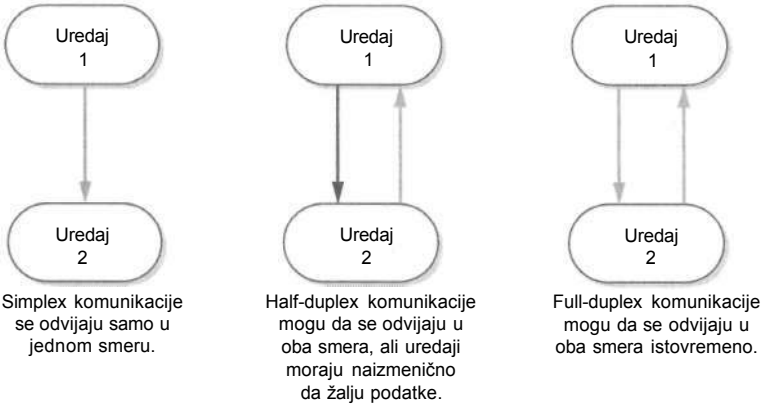
Izohroni prenos garantuje da će podaci stići na odredište fiksnom brzinom, tako da korisnik vidi kvalitetnu sliku, ili čuje kvalitetan zvuk bez praznina koje odvlače pažnju. Obično ne postoji nikakva detekcija grešaka. Ako dode do greške u toku prenosa, ona se ignoriše i korisnik može da primeti manje podrhtavanje slike, ili zvuka. Ovakve smetnje obično nisu problem i, u stvari, nisu gore od efekta koji stvaraju udar munje, električni motor koji radi u blizini, ili, čak, paljenje svetla. Smetnje se javljaju i nestaju.

U literaturi često možete da naidete na termin bysinc, akronim za binarne sinhrone komunikacije (binary synchronous communications), mada se ponekad koristi i skraćenica BSC. To nije mod prenosa u istom smislu kao što su asinhroni i sinhroni modovi. Reč je o protokolu koji je IBM uveo 60-ih godina prošlog veka za sinhrone komunikacije izmedu kompjutera i terminala. Predstavićemo ga u Poglavlju 9.

Simplex, half-duplex i full-duplex komunikacije

Do sada smo se u ovom poglavlju bavili načinima za prenos informacija sa jednog uredaja na drugi, sa jasno definisanom razlikom izmedu pošiljaoca i primaoca. To je bio primer simplex komunikacija (slika4.11) - komunikacija se obavlja samo u jednom smeru. Medu brojne primere ovakve vrste komunikacije ubrajaju se monitori na aerodromima, štampači, televizijski uredaji, ili razgovor sa nesimpatičnim profesorom u vezi rasporeda ispita.

Drugi oblici komunikacija zahtevaju veću fleksibilnost, tako da uređaj može i da šalje i da prima podatke. Metodi kojima se to postiže su različiti. Neki koriste half-duplex komunikacije, kod kojih oba uredaja mogu da šalju i primaju podatke, ali to mora da se izvršava naizmenično. Ovaj metod se koristi kod nekih modema, LAN standarda i perifernih uredaja. Na primer, prethodno pomenuti bisync protokol koristi half-duplex komunikacije.



SLIKA 4.11 *Simplex, half-duplex i full duplex komunikacije*

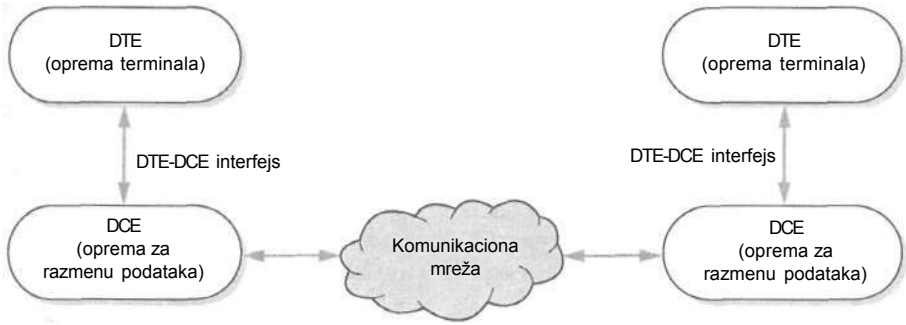
Najfleksibilniji metod su **full-duplex komunikacije,** kod kojih uredaj istovremeno može da šalje i da prima podatke. Kada uredaj šalje podatke preko jedne linije, može da prima podatke preko druge. Mnoge konekcije PC-ja sa udaljenim kompjuterima koriste full-duplex komunikacije. To može da se vidi na primeru kontinuelnog kucanja na tastaturi, istovremeno sa prikazivanjem informacija na ekranu. Mnogi modemi su full-duplex uredaji.

Dvosmema komunikacija postaje složena, posebno u mreži. Moraju da se koriste razni protokoli da bi se obezbedio ispravan prijem informacija na ureden način, tako da komunikacija izmedu uredaja bude efikasna. O ovome će biti reči u haredna četiri poglavlja.

4.4 Standardi za interfejse

U Poglavlju 3 i prethodnom odeljku opisano je nekoliko načina za prenos informacija. Možda ste pomislili da je za komunikaciju dva uređaja dovoljno da koriste isti mehanizam sa slanje i prijem podataka. To ipak ne mora da znači da će doći do komunikacije. Ako dve osobe govore istovremeno, one ne komuniciraju. Zdrav razum nalaže da se u okviru komunikacije naizmenično sluša i govori. Uređene diskusije zahtevaju poštovanje odredenih pravila (protoko-Ia). Slično važi i u slučaju komunikacija izmedu uredaja. Nema nikakvog smisla slati modulisane signale do uredaja koji nije spreman za osluškivanje i interpretiranje signala.

Na slici 4.12 prikazano je tipično uredenje povezanih uredaja. Akronim DCE podrazumeva opremu za razmenu podataka **(data circuit-terminating equipment),** dok DTE podrazumeva opremu terminala **(data terminal equipment). DTE** (na primer, personalni kompjuter) ne povezuje se direktno na mrežu. On komunicira preko DCE (na primer, modem) opreme. Konekdju izmedu DTE i DCE nazivamo **DTE-DCE interfejs.** U ovom odeljku predstavljamo neke standarde za DTE-DCE interfejse, a zatim nekoliko novijih protokola za uspostavljanje veze, koji se često sreću kod personalnih kompjutera.



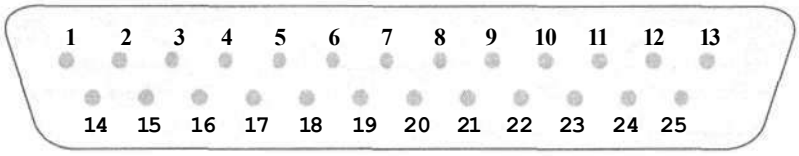
SLIKA 4.12 *DTE-DCE interfejs*

EIA-232 interfejs

Opštepoznati standard je EIA-232.* Razvila ga je EIA (Electronic Industries Association) početkom 60-ih prošlog veka, a doživeo je nekoliko revizija, koje su označene dodavanjem slova na kraj oznake standarda. Verzija koja se verovatno najduže zadržala je EIA-232-C. ITU ekvivalent V.24 definiše funkcionalne aspekte operacija i referencira se na naredni standard (V.28), koji definiše električne specifikacije. Osim toga, EIA je kreirala nekoliko novih varijacija, a 1997. godine je uspostavljen standard EIA-232-F. Nemamo nameru da sada navodimo sve razlike izmedu varijacija; smatrali smo da je dovoljno da predstavimo originalnu EIA-232 specifikaciju. Gde bude potrebno ukazaćemo na promene koje su izvršene u narednim verzijama.

Najočigledniji (i najvidljiviji) aspekt standarda je broj linija (25) između DTE i DCE. Ako se standard u potpunosti implementira, DTE i DCE su povezani 25-žilnim kablom (ponekad se naziva DB-25 kabl) koji spaja svaki uređaj preko 25-pinskog konektora (slika 4.13). Svaka linija ima specifičnu funkciju u uspostavljanju komunikacije izmedu uređaja. U tabeli 4.2 dat je pregled nekoliko linija sa naznačenim smerom signala (da li se linija koristi za prenos podataka od DCE-a ka DTE-u, ili obratno). Osim toga, u tabeli je data i EIA oznaka (kod kola) za svaku liniju. Ovde nećemo detaljnije predstaviti sve konekcije, ali ćemo opisati ulogu koju neka kola imaju u okviru tipične DTTi-DCE konekcije. Ako želite detaljniji opis, pronaći ćete ga u referenci [StOO].

Pretpostavimo da je DTE personalni kompjuter, a da je DCE modem. Ovo je uobičajena konflguracija kada se koristi eksterni modem. U Poglavlju 3 smo predstavili kako modem komunicira sa analognim svetom, a sada ćemo se fokusirati na razmenu podataka sa kompjuterom. Prvih šest kola se prvenstveno koristi za uspostavljanje razmene koja osigurava da ni jedan uredaj neće slati podatke ako ih drugi uredaj ne očekuje. Na slici 4.14 pokazano je da se razmena odvija u odredenom vremenskom intervalu.



SLIKA 4.13 *EIA-232 konektor*

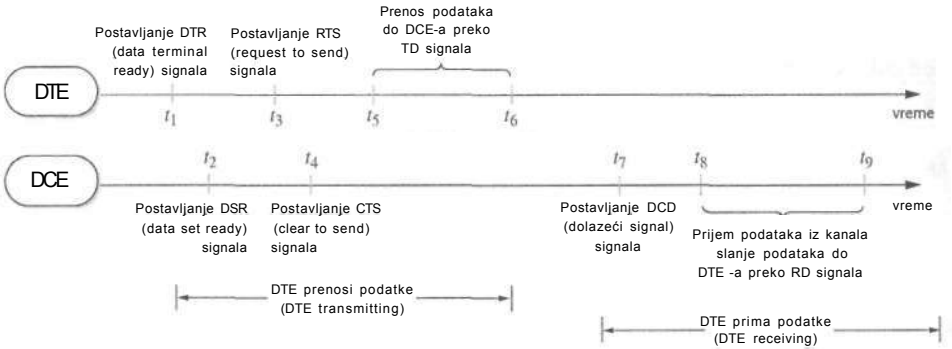
* Godinama je ovaj standard bio poznat i kao RS-232 serijski port na PC-jima. Ponegde se označava i kao EIA/TIA-232. Ovo ukazuje da je EIA radila zajedno sa TIA (Telecommunications Industry Association); 1991. godine su objavili EIA/TIA-232-E.

Tabela 4.2: Definicije kola za EIA-232-C

Kod kola	Broj linije	Smer signala	Funkcija
AA	1		Zaštitno uzemljenje - Povezuje se na okvir opreme, a ponekad na eksterno uzemljenje.
AB	7		Električno uzmeljenje - Svi naponski signali se mere u odnosu na ovo uzemljenje.
BA	2	DTE ka DCE-u	Prenos podataka (TD - transmit data) - Preko ovog kola DTE prenosi podatke ka DCE-u.
BB	3	DCE ka DTE-u	Prijem podataka (RD - receive data) - Preko ovog kola DTE prima podatke od DCE-a.
CA	4	DTE ka DCE-u	Zahtev za slanje (RTS - request to send) - DTE koristi ovo kolo za traženje dozvole od DCE-a pre nego što počne prenos podataka.
CB	5	DCE ka DTE-u	Dozvola za slanje (CTS - clear to send) - Ovo kolo DCE koristi da bi dao dozvolu DTE-u za slanje podataka.
CC	6	DCE ka DTE-u	Podaci spremni za slanje (DSR - data set ready) - Signal na ovoj liniji ukazuje da je DCE povezan na komunikacioni medijum i da je spreman za izvršavanje operacija. Na primer, ako je DCE modem, ovo kolo ukazuje da je veza uspostavljena.
CD	20	DTE ka DCE-u	Terminal spreman za prenos (DTR - data terminal ready) - Signal na ovoj liniji ukazuje da je DTE spreman za slanje, ili prijem podataka. Može da se koristi kao signal modema kada se poveže na komunikacioni kanal.
CE	22	DCE ka DTE-u	Indikator zvona - Ukazuje da DCE prima signal zvona (tj. kada modem primi poziv) od komunikacionog kanala.
CF	8	DCE ka DTE-u	Detekcija nosećeg signala (DCD - data carrier detected) - Ukazuje da DCE prima noseći signal koji zadovoljava uspostavljene kriterijume komunikacione mreže. U suštini, ovo znači da DCE "razume" dolazeći signal.

DTE ukazuje na svoju spremnost umetanjem (slanjem signala) DTR kola broj 20 (na slici 4.14 trenutak t,). DCE registruje signal i reaguje, povezujuđ se na mrežu (ako većnije povezan). Kada se DCE poveže i kada je spreman, on umeće DSR kolo broj 6 (trenutak t_2). U suštini, DCE potvrduje spremnost DTE-a; i on se deklariše kao spreman.

Kada su oba uredaja spremna, DTE traži dozvolu za prenos podataka do DCE-a umetanjem RTS kola broj 4 (trenutak t_3). Ovo kolo kontroliše i smer prenosa u half-duplex komunikacijama. Nakon detektovanja RTS signala, DCE ulazi u mod za prenos, što znači da je spreman za slanje podataka preko mreže. Zatirn, reaguje umetanjem CTS kola broj 5 (trenutak t_4). Konačno, DTE šalje podatke preko TD kola broj 2 (između trenutaka C_5 i t_s).



SLIKA 4.14 *Slanje i prijem signala preko ElS-232 konekcije*

Kada DCE detektuje dolazeći signal sa mreže koju prepoznaje, on potvrđuje DCD kolo broj 8 (trenutak t2). Kada signal stigne, DCE šalje podatke ka DTE-u preko RD kola broj 3. Neki stariji modemi su imali lampice na prednjoj strani - one su ukazivale koje su linije potvrđene. Ovaj signal je korisniku davao šansu da vidi šta se, u stvari, dešava. Ipak, u većini slučajeva, lampice su treperele toliko brzo da se nije znalo da li su uključene, ili isključene.

EIA-232 PODSKUPOVI Interesantno je da mnogi konektori EIA-232 portova na kompjuterima nisu imali 25 pinova. Sećate se da smo predstavili EIA-232 standard. Da li je proizvođač u potpunosti implementirao *standard* to je već bilo drugo pitanje. Činjenica je da su mnogi uključili samo podskup EIA-232 defmicija.

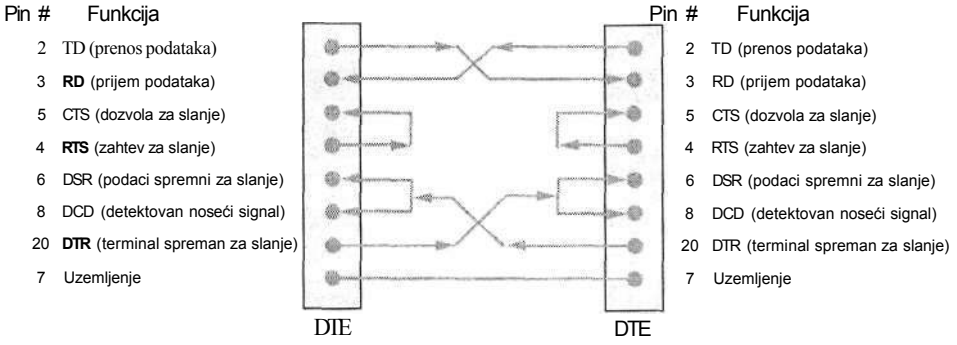
Ilustracije radi, pre nego što su modemi počeli da se instaliraju u kompjuterima, korisnik je morao da kupi eksterni modem i da ga poveže sa kompjuterom pomoću kabla. Tipični kabl je imao 25-pinski konektor na jednom kraju, koji se priključivao u modem, a na drugom kraju je bio 9-pinski konektor, koji se priključivao u kompjuter. Ovo je pomalo nalik priključivanju utikača sa tri zupca u utičnicu sa dva otvora, ali postoji razlog za ove razlike. Mnogi modemi su bili saglasni sa kompletnim standardom. Medutim, mnogi korisnici nisu koristili puni opseg EIA-232 mogućnosti. Prvenstveno, bila im je potrebna mogućnost komunikacije na način koji smo opisali na slici 4.14. Zato su serijski portovi u opštem slučaju zahtevali 9-pinski konektor koji je koristio sedam kola opisanih u primeru i jedno, ili dva za uzemljenje. Odluka o načinu implementacije standarda uglavnom je zavisila od ekonomske računice: zašto implementirati (i plaćati) puni opseg karakteristika kada je mala verovatnoća da ćete ih ikada koristiti! Kablovi sa različitim konektorima na krajevima povezuju samo potrebna kola. Ekstra linije na strani modema nisu povezane na personalni kompjuter.

Jedan nedostatak EIA-232 standarda je što ima ograničen opseg signala i ogranično rastojanje na kome može da funkcioniše. Obično se koristi za prenos do 20.000 bitova u sekundi (bps) na rastojanjima do 50 stopa. U nekim slučajevima, kao što su situacije sa manjom interferencom, moguća su i veća rastojanaja, ali tada se koriste drugi standardi, koje ćemo uskoro ukratko predstaviti.

Kao finalnu napomenu, ponovićemo da je bilo nekoliko revizija EIA-232 standarda. Iako ovde nismo detaljnije obuhvatili razlike između verzija, istaď ćemo jednu promenu. Verzija EIA-232-E, izdata 1991. godine, promenila je način interpretacije signala RTS i CTS. Prethodni primer razmene podataka demonstriran je u half-duplex modu; medutim, vedna današnjih modema je full-duplex i često ima neki oblik hardverske kontrole toka.* DTE može da koristi RTS da bi ukazao da može da primi podatke od DCE-a, a DCE može da koristi CTS u analogne svrhe. Za većinu operacija ova kola su stalno potvrdena.

-)ULL MODEMI Ponekad ćete možda hteti da dopustite da dva uređaja (kao što su personalni kompjuteri) komuniciraju direktno, tj. bez mreže, ili DCE uređaja između njih. U takvim slučajevima Vaša prva reakcija može da bude povezivanje EIA-232 portova preko kabla, a pre-ostali deo posla bi se prepustio protokolima. Na kraju krajeva, oba kompjutera šalju i primaju podatke preko svojih EIA-232 portova. Medutim, povezivanjem pomoću prostog kabla uspostavljate kontakt između istih pinova na oba kraja. Na primer, kabl bi povezao pin 2 oba DTE uredaja. Problem je što bi oba pina pokušala da šalju podatke preko iste linije. Prvi DTE šalje podatke, a drugi ih prima preko linije 2. Pošto drugi DTE očekuje da primi podatke preko linije 3, direktna veza neće funkcionisati. Slično tome, pošto kabl povezuje pin 3 na oba kraja, oba uredaja očekuju prijem podataka preko istog kola, a ni jedan ih ne šalje preko te linije.

Jedno rešenje ovog problema je povezivanje DTE uredaja ukrštanjem nekih kola. Na slici 4.15 prikazan je jedan mogući način povezivanja preko null modema. **Nill modem** može da bude ili kabl koji povezuje različite pinove na svakom konektoru, ili uredaj koji jednostavno



SLIKA 4.15 *Nill modem*

* Kontrolu toka detaljnije ćemo predstaviti u Poglavlju 8. Za sada možete da je smatrate mehanizmom za kontrolu brzine kojom podaci stižu, uz održavanje mogucnosti za stopiranje i nastavak dolazećeg toka podataka. Može da se uporedi sa baterijom na slavini. Okrenete je u jednom smeru i teće veća količina vode; okrenete je u drugom smeru i voda tece slabije.

ukršta konekcije, koristeći postojeće kablove. U svakom slučaju, rezultat je isti. Null modem sa slike 4.15 povezuje pin 2 na jednom kraju sa pinom 3 na drugom kraju. Zbog toga, kada DTE šalje podatke preko pina 2, oni se rutiraju na pin 3 na drugom kraju, gde je omogućen ispravan prijem podataka.

Nill modem sa slike 4.15 povezuje pinove 4 i 5 istog DTE uredaja. Razlog za to je prethodno opisani primer. Kada DTE želi da prenese podatke, on mora da zatraži dozvolu i da čeka CTS signal (signal dozvole) od DCE uređaja. Pošto ovde nema DCE uredaja, null modem rutira RTS signal (pin 4) nazad do pina 5. DTE šalje sopstveni signal sa pina 5 i tako "misli" da je DCE odgovorio CTS porukom na njegov zahtev.

Ostale ukrštene konekcije osiguravaju da svaki DTE bude spreman pre nego što se podaci pošalju. Kao što smo prethodno opisali, DTE potvrduje DTR liniju broj 20 kada bude spreman i očekuje da DCE reaguje potvrdom DSR linije broj 6. Ovde se linija 20 jednog DTE-a rutira na liniju 6 drugog DTE-a; kada jedan signalizira da je spreman, drugi odmah prima signal. Ovim se DTE navodi da "pomisli" da je DCE povezan na mrežu i da je takode spreman za prenos.

Na slici 4.15 prikazan je još jedan primer brojnih varijacija null modema. Razlike se javljaju u zavisnosti od zahteva uređaja i u kojoj se meri koristi EIA-232 protokol. Informacije o drugim varijacijama možete da pronadete u referencama [StOO] i [Ru89].

X.21 interfejs

Standard X.21 interfejsa je defmisao ITLI-T. Koristi 15-pinski konektor i omogućava balansirana (električni standard X.27) i nebalansirana (X.26 standard) kola. Postoji nekoliko značajnih razlika izmedu X.21 i EIA standarda. Prva je da je X.21 definisan kao interfejs za digitalno signaliziranje. Druga razlika uključuje način na koji se razmenjuju kontrolne informacije. EIA standard definiše specifična kola za kontrolne funkcije. Za veću kontrolu je neophodan veći broj kola, što otežava uspostavljanje konekcija. Princip na kome je zasnovan X.21 podrazumeva više logičkih kola (inteligencije) u DTE-u i DCE-u koja mogu da interpretiraju kontrolne sekvence, tako da se redukuje broj kola koja se moraju povezati.

Tabela 4.3: X.21 standard za balansirano kolo

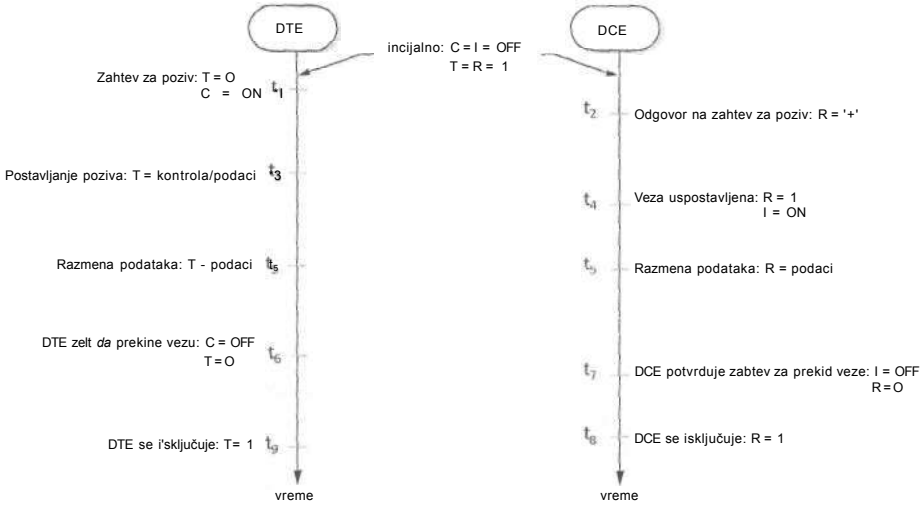
Kod kola	Broj pina	Smer signala	Funkcija
	1		Omotač
G	8		Uzemljenje signala
T	2, 9	DTE ka DCE	Prenos podataka, ili kontrolnih informacija
R	4, 11	DCE ka DTE	Prijem podataka, ili kontrolnih informacija
C	3, 10	DTE ka DCE	Kontrola
I	5, 12	DCE ka DTE	Indikacija
S	6, 13	DCE ka DTE	Tajming elementa signala
B	7, 14	DCE ka DTE	Tajming bajta

U tabeli 4.3 prikazane su X.21 definicije kola za balansirano kolo. DTE koristi samo dva kola (T i C) za prenos do DCE. Slično tome, DCE koristi dva kola (R i I). Druga dva kola su korišćena za tajming signala kod sinhronih komunikacija. Sa manjim brojem kola koja služe za prenos, potrebna je bolja logika za interpretiranje signala koji se prenose. Tipično, T i R se koriste za prenos niza bitova, a C i I su postavljeni na ON (binarno O), ili OFF (binarno 1) stanje. Dakle, T i R se koriste za signaliziranje i slanje podataka i kontrolnih informacija.

Signali na T, G, R i I linijama definišu stanja (status) DTE i DCE uredaja. ITU-T definiše različita stanja za X.21, ali ovde ih nečemo detaljnije proučavati. Ipak, ilustrovaćemo kako protokol funkcioniše na primeru proste konekcije.

Na slici 4.16 ilustrovana je sekvenca razmene signala dok DTE i DCE razmenjuju informacije. U početku, kada su oba uredaja neaktivna, kola C i I su postavljena na OFF, a kola T i R prenose binarne jedinice. Kada DlE želi da se poveže na udaljeni DTE, počinje da šalje nule preko T kola i postavlja C na ON (trenutak *t,* na slici 4.16). DCE registruje promenu stanja i reaguje slanjem sekvence + karaktera preko R (trenutak I_2). Ovo je analogno podizanju slušalice na telefonu i reakciji lokalne centrale koja šalje ton za dozvolu iniciranja poziva.

Ako ste inicirali telefonski poziv, Vaš sledeći korak treba da bude pozivanje broja. DTE reaguje slično prenošenjem kontrole i inforrnacija o podacima preko T (trenutak t_3).



SLIKA 4,16 *Slanje i prijetn podataka preko X.21 konekcije*

Na ovaj način, DCE dobija neophodne informacije, kao što je adresa, tako da može da uspostavi komunikaciju sa udaljenim DTE uređajem preko mreže. Dok DCE pokušava da uspostavi konekciju, šalje niz SYN karaktera preko R linije. Kada se konekcija uspostavi, on obaveštava DTE slanjem jedinice preko R linije i postavljanjem I na ON (trenutak t_4).

U ovoj tački (trenutak t_5) DTE i DCE mogu da razmenjuju podatke, pri čemu DTE koristi T kolo, a DCE R kolo. Eventualno, DTE može da odluči da okonča svoje aktivnosti. On ukazuje na svoju nameru slanjem O preko T kola do DCE-a i postavljanjem C na OFF (trenutak C_6). DCE potvrđuje nameru prenošenjem O preko R i postavljanjem I na OFF (trenutak t_7). Konačno, DCE prekida vezu prenošenjem jedinica preko R (trenutak C_8) i DTE se isključuje prenošenjem jedinica preko T (trenutak t_9). Tako se oba uredaja vraćaju u neaktivno stanje, kojim smo i započeli primer.

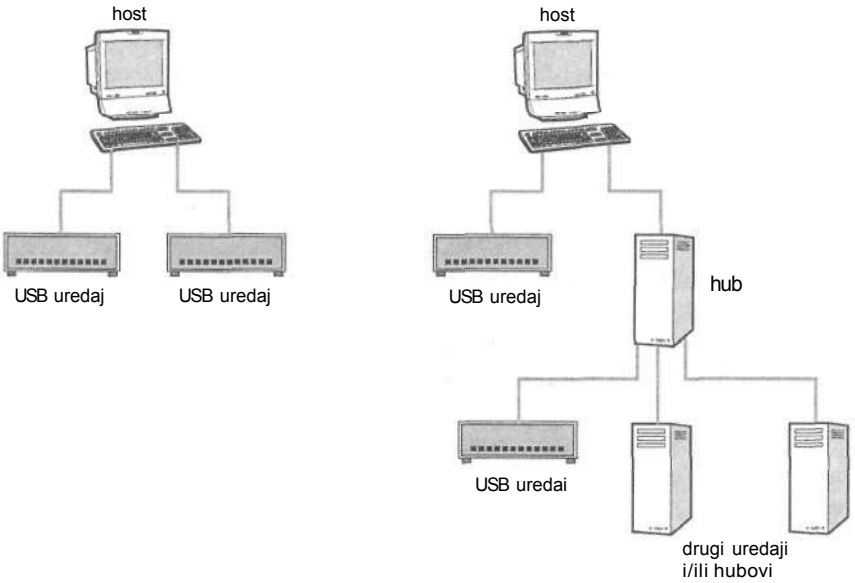
USB

Jedna od najčešćih žalbi korisnika personalnih kompjutera do nedavno se ticala složenosti povezivanja perifernih uredaja. Korisnici su morali da se "bore" sa serijskim i paralelnim portovima i specijalnim konekcijama namenjenim za kontrolere igara, tastaturu, miša i tako dalje. Naravno, nakon toga su morali da instaliraju odgovarajuće drajvere da bi konekcija pro-funkcionisala. Sve to je bilo prilično konfuzno. Tu se nameće logično pitanje zašto proizvodači ne bi standardizovali konekcije i čak se i složili za standard izmedu uređaja. Na kraju krajeva, transfer podataka se uvek svodi na slanje i prijem bitova. Srećom, ovo pitanje su postavili pravi ljudi. I, da "stvar" bude još bolja, dali su i odgovor, koji se zove univerzalna serijska magistrala (USB - **universal serial bus).**

U početku je na definisanju USB-a zajedno radilo sedam kompanija (Compaq, DEC, IBM, Intel, Microsoft, NEC i Northern Telecom). Primarni motiv je bilo pojednostavljenje povezivanja, a da se istovremeno obezbede i veće brzine prenosa za novije uređaje koji su se pojavili na tržištu. Rezultat je bio USB verzija 1.0, a ubrzo je izašla i verzija 1.1, dok je verzija 2.0 izašla u skorije vreme. Nećemo se baviti detaljima razlika izmedu ovih verzija, ali ćemo dati opšti pregled kako USB funkcioniše - videćemo da on obezbeduje veoma fleksibilno uredenje za povezivanje više uredaja, tako da korisnik ima na raspolaganju više opcija.

USB KONEKCIJE Većina kućnih kompjutera verovatno koristi uredenje sa slike 4.17a. Kompjuter (host na slici 4.17a) ima par USB soketa, a korisnik jednostavno povezuje jedan, ili dva USB-kompatibilna uredaja na njih, kao što su skener, ili digitalna kamera. U stvari, USB korisniku omogućava da poveže do 127 različitih uredaja, mnogo više nego što će korisniku PC-ja ikada biti potrebno. Naravno, ne postoji 127 zasebnih soketa na zadnjoj strani kućišta hosta. Umesto toga, ako korisnik želi da poveže više uredaja, mora da definiše uredenje slično onom na slici 4.17b.

Jedan USB uredaj može da se poveže direktno na host. Medutim, da bi se povezao veći broj uredaja, potreban je hub, jednostavni uredaj sloja 1, koji regeneriše i ponavlja signale primljene preko jedne konekcije do svih ostalih konekcija. Korisnik može da poveže **hub** na host, pa da poveže nekoliko drugih USB uredaja na hub. U stvari, korisnik može da poveže i druge hubove na prvi, a uredaje na drugi hub.



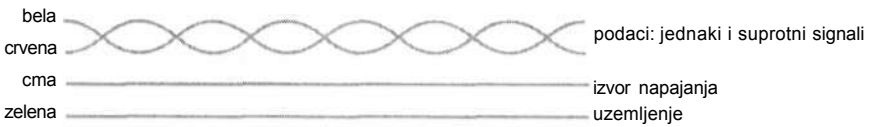
(a) Jednostavna konekcija koja je dovoljna kod većine kućnih PC-ja

(b) Složenija hijerarhijska konekcija

SLIKA 4.17 *Konektovanje USB uređaja*

Korisnik, dakle, kreira hijerarhijsku topologiju kod koje je host (personalni kompjuter) koren, USB uredaji su čvorovi listovi (čvorovi na kraju hijerarhijske putanje), a hubovi su međučvorovi (čvorovi koji povezuju nešto iznad njih sa nečim ispod njih). U suštini, sve što host pošalje preko USB konekcije "putuje" do svih čvorova u hijerarhiji. Teorijski, ne postoji ograničenje za broj uredaja koje je moguće povezati na ovakav način. Međutim, USB koristi 7-bitnu šemu adresiranja za referenciranje uređaja, tako da je ukupan broj USB uređaja ograničen na 127 i sam host.

USB kabl povezuje uređaje. Kabl sadrži četiri žice (slika 4.18). Dve žice (obično zelene i bele) služeza prenos podataka korišćenjern električnih signala koji su isti po jačini, ali imaju suprotan polaritet (tj. balansirani signali). Žice su upredene jedna oko druge i zaštićene omotačem da bi se redukovao efekat šuma. Mehanizam signalizacije predstavlja jednostavnu varijaciju NRZ šeme kodiranja.



SLIKA 4.18 *USB žice*

USB definiše 0 promenom signala na početku intervala, a 1 zadržavanjem konstantnog nivoa signala. U svakom slučaju, signal ostaje fiksan za vreme intervala bita (za razliku od Mančester kodiranja, gde se prelaz izvršavao na sredini intervala). Ovde se niz jedinica predstavlja konstantnim signalom. USB 1.1 definiše maksimalnu bitsku brzinu od 12 Mbps; novija verzija USB 2.0 obećava brzine do 480 Mbps. Osim toga, u kablu se nalazi i crna žica, koja se koristi za uzemljenje, a crvena prenosi napajanje niske amplitude do USB uređaja. Ovim je poboljšana fleksibilnost, jer neki USB uredaji (na primer, miš i tastatura) zahtevaju veoma slabo napajanje. Ova linija obezbeduje napajanje koje im je potrebno, a proizvođači ne prave izvor napajanja na uredaju.

Sam USB kabl ima dva tipa priključaka, tako da korisnici ne mogu da priključe pogrešan kabl na uređaj. Mnogi korisnici personalnih kompjutera prepoznaju mali ravni priključak (slika 4.19) koji se povezuje sa kompjuterom, a naziva se standardni A priključak. Standardni B priključak (slika 4.19) povezuje se na USB uređaj i može da se prepozna po nešto više kvadratnom obliku. Zbog različite geometrije, skoro je nemoguće greškom priključiti B priključak u A soket, ili obratno. Svako ko je nekada slučajno povezao pogrešne krajeve kabla u sokete i pitao se zašto ništa ne funkcioniše zna prednosti ove šeme. Autor ove knjige može to da potvrdi iz sopstvenog iskustva (to je stvarno nešto čega se stidim). Dužina kabla je ograničena na 4,5 metara. Veća dužina ne daje nikakve garancije za integritet električnih signala.

Transfer podataka USB komunikacija se odvija u master/slave modu. Prosto rečeno, host kontroliše sav transfer, a uređaj preuzima akciju samo kada dobije nalog od hosta. Način na koji ovo sve funkcioniše je prilično interesantan. Osnovni koncept za transfer podataka pomoću USB konekcije je koncept okvira (frame). Imajte na umu da se reč okvir koristi na specifičan način kada se govori o USB-u. Ranije smo u ovom poglavlju definisali okvir kao organizovanu grupu bitova. U kontekstu USB-a, on predstavlja vremenski interval od 1 milisekunde. Za vreme te milisekunde informacije mogu da se prenose organizovane po paketima (USB termin za organizovanu grupu bitova). Šteta je što se neki termini koriste za različite "stvari", u zavisnosti od konteksta. Medutim, na polju koje se menja veoma brzo uspostavljanje univerzalnih termina za sve oblasti nije nimalo jednostavno.

Interesantna karakteristika USB-a je da su svi uredaji sinhronizovani u odnosu na okvir. Ovo se ne postiže zajedničkim taktom za sve uredaje, već preko hosta. Na početku svakog okvira host šalje specijalni paket koji "putuje" do svakog USB uredaja i obaveštava da započinje novi okvir. Šta se dešava nakon toga zavisi od hosta koji kontroliše transfer informacija, od tipova uredaja i od podataka raspoloživih za transfer. Pogledajmo kako to funkcioniše.

SLIKA 4.19 *USB kabl i priključci*

Za početak, LISB definiše četiri različita tipa prenosa (ponekad se nazivaju tipovi okvira): kontrolni, teretni (bulk), izohroni i prekidni. Svaki odgovara različitom tipu razmene i implementira se razmenom paketa. Ukratko ćemo opisati kako se sve to dešava u kontekstu okvira, ali najpre ćemc defmisati različite tipove transfera:

- Kontrolni transfer: USB uredaji se mogu priključivati (ili isključivati) bez isključivanja sistema sa napajanja, ili učitavanja novog softvera da bi funkcionisali. Zbog toga, operativni sistem na hostu mora da detektuje kada se uredaj doda, ili ukloni. U toj tački ulazi u fazu inicijalizacije, u kojoj host traži od novog uređaja da potvrdi svoj tip i odredi kojim brzinama može da prenosi podatke. Uređaj odgovara na ove upite i host eventualno pridružuje adresu preko koje može da komunicira sa uredajem. Tako host može da razlikuje uredaje koji su povezani na njega. Nakon što se uređaji povežu, host može da pošalje komande do njih, da zahteva izveštavanje o njihovom statusu, ili da inicira razmenu podataka.

- **Teretni (bulk) transfer:** Neki USB uredaji su dizajnirani za prenos velikih količina informacija. To su, na primer, skeneri, ili digitalne kamere. Podaci se obično smeštaju i prenose u vidu paketa, a na prijemnoj strani se koristi mehanizam za detekciju grešaka. U Poglavlju 6 ćemo predstaviti detekciju grešaka i njene detalje; ideja je jednostavna: proverava se da li u dolazećem paketu ima grešaka koje su eventualno mogle da se jave u toku prenosa. Ako se greška detektuje, inicira se ponovno slanje paketa. Pomoću tehnika za detekciju grešaka osiguran je pouzdan prenos podataka. Zapamtite da u teretnom transferu može da učestvuje nekoliko uredaja i da nema garancija za vreme pristizanja podataka. Ovde se garantuje pouzdan, ali ne i pravovremeni transfer. Host koordiniše transfer.

- **Prekidni transver:** Neki periferni uredaji kompjutera (na primer, kontroler diska) funkcionišu u sistemu prekida. Kada su podaci spremni za transfer, uredaj šalje signal do centralne procesorske jedinice. Time se generiše prekid u operaciji koju procesor trenutno obavlja. U toj tački operativni sistem preuzima kontrolu, utvrduje razlog prekida i eventualno poziva upravljačku rutinu (handler routine) da reaguje. Sve ovo zahteva složene protokole koji omogućavaju razmenu signala prekida i odziva.

  USB ne funkcioniše na ovaj način. Kada uređaj ima podatke spremne za transfer, jednostavno ih zadržava i čeka da dobije upit od hosta. Ovo pomalo liči na pokušaj saznavanja pravih informacija od dva mala deteta koja su počela da se prepiru. Znate **ko** je prvi počeo, ali niko neće ništa da prizna - morate da budete dovoljno lukavi da to izvučete od njih. Prekidni transferi obično prate uredaje koji prenose manje informacija. Na primer, kada kucate na tastaturi, kodovi se pamte sve dok ih host ne zatraži. Po prijemu zahteva od hosta, tastatura prenosi karaktere koje je korisnik uneo. Proces u kome host pita uredaj da li ima podatke za slanje naziva se prozivka **(polling).** Pošto host "proziva" uređaje sa fiksnom frekvencijom, može se garantovati "glatka" isporuka informacija. Na primer, ako host proziva tastaturu svakih 50 okvira (50 milisekundi), može da uzme do 20 karaktera u svakoj sekundi. To je mnogo brže nego što većina nas može da se nada da može da otkuca. Jedino bi Supermen mogao brže da kuca.

- **Izohroni transfer**: Neki USB uređaji (na primer, mikrofoni i slušalice) su real-time uredaji koji moraju da šalju podatke zagarantovanom brzinom. Host može da rezerviše deo svakog okvira za taj uređaj i tako može da garantuje da će se deo okvira potrošiti na prenos podataka ka tom uređaju, ili od tog uredaja. Prenošenjem određenog broja bitova u svakom okviru host može da garantuje određenu bitsku brzinu.

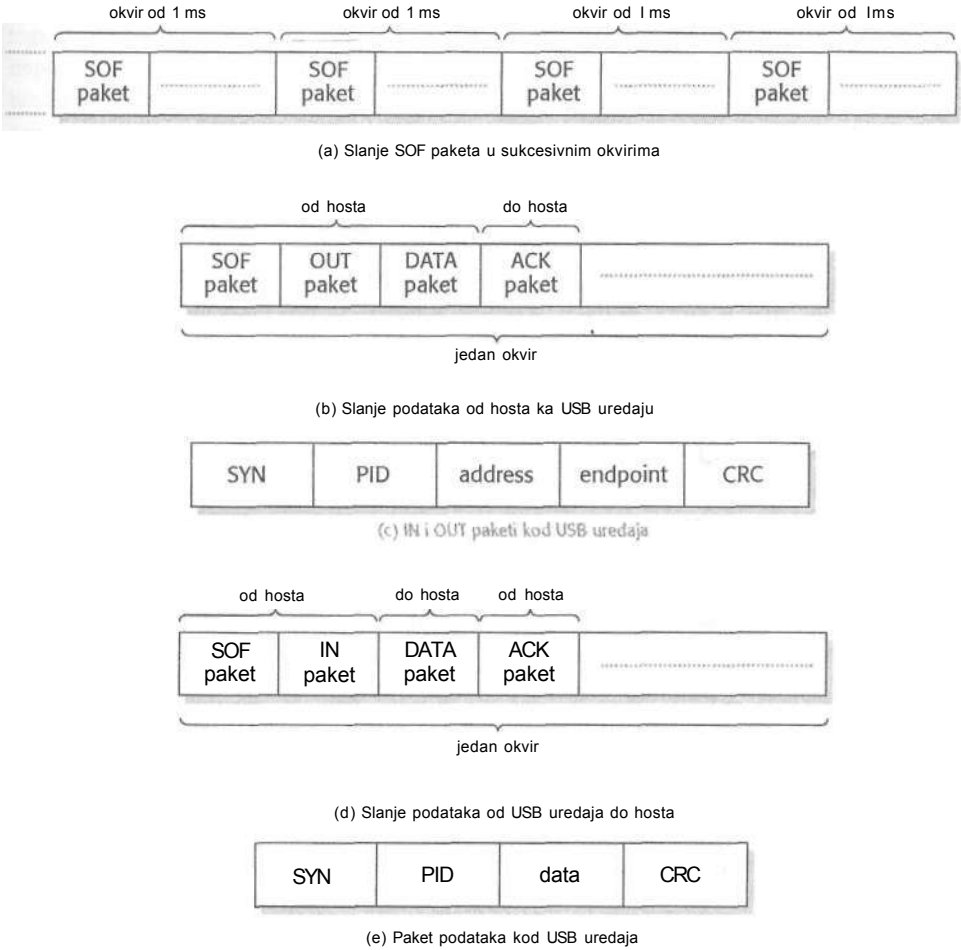
  Za razliku od teretnog transfera, nema detekcije grešaka. Ovde se nastoji obezbediti pravovremena isporuka podataka. Greške koje nastaju u prenosu se tolerišu. Ako se prenose audio zapisi, takve greške (čak i ako su primetne) dovode do manjih devijacija (škriputanje, klik, statičan zvuk, periodi tišine, itd) od originalnog zapisa, ali se isporuka zapisa nastavlja. Uredaj neće zahtevati ponovno slanje podataka dok se ne postigne savršen zvuk. Vlasnici starih gramofonskih ploča (posebno onih sa ogrebotinama) dobro znaju kako izgleda kada se javljaju devijacije u poređenju sa originalnim zapisom. Čak i izgrebani CD-ovi mogu slično da se ponašaju.

USB PAKETI Sledeći logičan korak je opisivanje relacije izmedu okvira i paketa podataka. U opštem slučaju, može da postoji nekoliko razmena paketa u toku jednog okvira. Ono što se pomoću njih postiže zavisi od tipa paketa koji se razmenjuju. Ovde ćemo predstaviti tri tipa paketa: tokene, podatke i pakete za usaglašavanje. Iako se sadržaj ovih paketa razlikuje, postoje dve zajedničke "stvari" za sve vrste paketa: polja SYN i ID paketa (PID). Polje SYN sadrži specijalni uzorak bitova koji izaziva promenu električnih signala u skladu sa taktom pošiljaoca. Tako je omogućena sinhronizacija internog takta prijemnog uređaja sa brzinom kojom bitovi stižu, čime je obezbeden ispravan prijem bitova. Polje PID sadrži bitove koji identifikuju tip paketa. Zapamtite da uređaj može da primi paket takoreći u bilo kom trenutku i mora da postoji neki način na koji će se identifikovati tip primljenog paketa.

Počećemo sa tokenom. Host koristi token pakete za slanje informacija, ili zahteva do USB uređaja. Postoji nekoliko vrsta tokena. Jedan je SOF (Start of Frame) paket (slika 4.20a). Prethodno smo istakli da su svi uredaji sinhronizovani za početak sledećeg okvira. Drugim rečima, svaki uredaj "zna" kada počinje okvir: Medutim, umesto da se svi uredaji vezuju na zajednički takt (ovo je teško izvodljivo kada se nasumice uključuju i isključuju), taj posao obavlja host. Na početku svakog okvira (jednom svake milisekunde) host šalje SOF paket do svih USB uređaja. SOF paket je posebno važan za real-time uredaje koji moraju da prenesu minimalni broj bitova u svakom okviru.

Dva primera tokena su IN i OUT paketi. Oni predstavljaju zahteve hosta za iniciranje prenosa podataka. Razlika je u smeru transfera. Da bismo to ilustrovali, pretpostavićemo da host želi da pošalje podatke do USB uredaja (slika 4.20b). Dešava se sledeće:

1. Nakon slanja SOF paketa, host šalje OUT paket. OUT paket (slika 4.20c) sadrži 7-bitnu adresu - ona identifikuje USB uredaj koji treba da primi podatke. Odgovarajući uredaj "vidi" OUT paket, prepoznaje svoju adresu u njemu i priprema se za prijem podataka. Osim SYN i PID polja, OUT paket sadrži CRC (Cyclic Redundance Check) polje, koje se koristi za detekciju grešaka koje se eventualno javljaju u toku prenosa.



SLIKA 4.20 *USB okviri i paketi*

Ovo polje ćemo sretati i kod drugih paketa, ali pošto još uvek ne govorimo o detaljima detekcije grešaka, odložićemo predstavljanje CRC polja za Poglavlje 6. OUT paket ima i polje Endpoint, koje ćemo uskoro predstaviti.

2. Zatim, host šalje DATA paket (slika 4.2Oe), koji sadrži podatke koji se prenose.

3. Konačno, ako nema grešaka u toku prenosa, USB uređaj šalje ACK paket (u stvari, paket usaglašavanja) nazad do hosta. Ako je detektovana greška, uredaj će do hosta poslati drugu vrstu paketa usaglašavanja (NACK paket). U toj tački host ponavlja proceduru i pokušava ponovo da pošalje podatke.

Host može da inicira i druge transfere u konkretnom okviru (isprekidana linija na slici 4.20b), ali tekući transfer mora da se kompletira.

Ako host želi da primi podatke od uređaja, razmena bi se odvijala kao na slici 4.2Od:

1. Host šalje IN paket.
2. Nakon prijema IN paketa, uredaj šalje DATA paket.
3. Ako ne dode do grešaka u toku prenosa, host vraća ACK paket.

Ako uredaj nema podatke za slanje, on će odgovoriti NACK paketom. Host će to prepoznati kao indikator da uredaj nema ništa za slanje.

Pažljiviji čitaoci su možda primetili nešto u prethodnom opisu: DATA paketi ne sadrže adresu. Medutim, DATA paketu prethodi IN, ili OUT paket, koji identifikuje uređaj koji učestvuje u transferu. Tako uredaj može da se pripremi za transfer. Osim toga, pažljiviji čitaoci su mogli da primete da smo "prešli" preko Endpoint polja u IN i OUT paketima. Razlog je činjenica da se to ne tiče prethodne rasprave. Međutim, detaljnija analiza transfera podataka bi pokazala da naznačavanje adrese uredaja nije dovoljno, jer adresa možda ne definiše u potpunosti odredište podataka. Neki uredaji imaju više pridruženih izvora, ili odredišta. Primer je kontroler igara sa višestrukom dugmadi, gde svako od njih može da uključuje transfer različitih informacija. U takvim slučajevima identifikator Endpoint se koristi radi daljeg defmisanja tačnog izvora, ili odredišta podataka.

O USB-u može da se kaže još mnogo štošta, ali moramo da predemo i na daige teme. Čitaoci koji su zainteresovani za više detalja mogu da pogledaju referencu [AxOl], ili mogu da posete Web sajt www.usb.org.

FireWire

Krajem 80-ih godina prošlog veka Apple je razvio FireWire, tehnologiju dizajniranu za povezivanje elektronskih uredaja. Na osnovu tog dizajna, IEEE usvaja 1995. godine standard IEEE 1394. Zaštićeni naziv (trademark) je Apple FireWire. I Sony proizvodi verziju IEEE 1394 standarda i plasira je pod oznakom i.Link. Kada kupite novi kompjuter, možete da vidite specifikaciju "IEEE 1394" u listi "External Ports", zajedno sa specifikacijama za jedan, ili dva USB porta.

FireWire ima dosta zajedničkog sa USB-om. Za oba standarda važi sledeće:

- Mogu se priključivati na sistem bez potrebe za isključrvanjem sistema sa izvora napajanja
- Reč je o plug-and-play uredajima.
- Koriste serijske konekcije.
- Obezbeduju standardizovani način za priključivanje velikog broja različitih uredaja.
- Relativno su jeftini za implementiranje.

Ipak, postoje i neke velike razlike. U vreme kada je ova knjiga nastajala najvažnija razlika je bila u brzini. FireWire specifikacije definišu bitsku brzinu od 400 Mbps, dok USB ima samo 12 Mbps (ovo poredenje verovatno neće biti validno u vreme kada ova knjiga izade iz štampe).

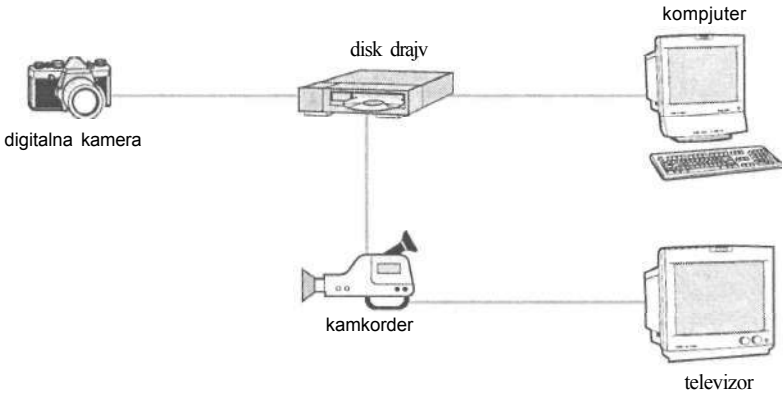
USB 2.0 specifikacija nagoveštava prenos od 480 Mbps, što može da se uporedi sa FireWire. Osim toga, priprema se povećanje brzine za FireWire do 800 Mbps. Postoje i dugoročni planovi za implementiranje FireWire standarda kod konekcija sa optičkim fiberom, tako da se dostignu brzine izražene u Gbps. Nećemo se zadržavati na konkrelnim bitskim brzinama, jer će se one najverovatnije menjati u skorijoj budućnosti.

KONEKCIJE Kao i USB, FireWire je dizajniran za povezivanje različitih uređaja, ali ovde su se dizajneri fokusirali na podršku multimedijalnim uredajima (posebno onima koji podržavaju digitalne video aplikacije), kao što su digitalni kamkorderi, ili digitalne kamere. Mnogi FireWire vide kao bolju alternativu za SCSI (Small Computer Systems Interface), standardni high-speed interfejs za povezivanje uredaja. Međutim, SCSI koristi paralelne komunikacije, što podrazumeva uplitanje kablova i veću cenu.

FireWire povezuje više uredaja koristeći daisy chain pristup (slika 4.21). To znači da možete da povežete nekoliko uredaja u nizu pomoću FireWire kabla (čija je maksimalna dužina oko 4,5 metara) između susednih parova. Osim toga, moguće je da jedan uredaj označava početak dva daisy chaina (primećujete dva daisy chaina levo i ispod disk drajva na slici 4.21). U stvari, moguće je vizuelizovati uredaje u okviru hijerarhijskog uređenja. Jedino nije dopušteno povezivati ih tako da formiraju petlju. Na primer, na slici 4.21 ne bi bilo dopušteno povezivanje televizora sa kompjuterom.

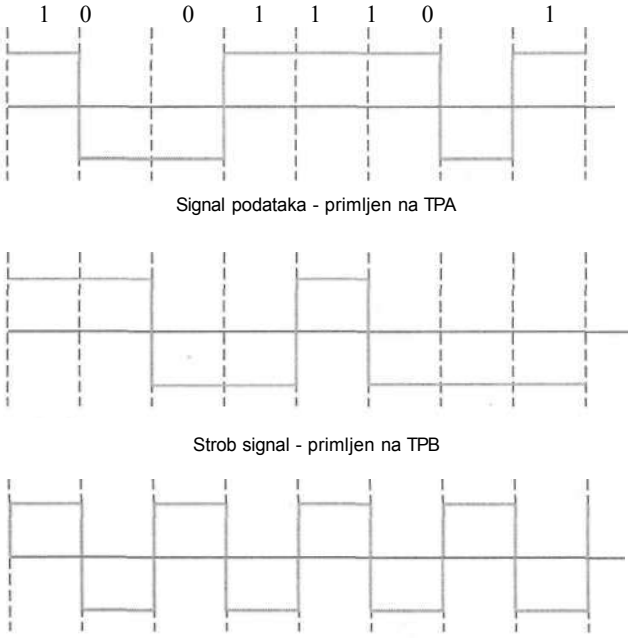
Kada se koristi daisy chain, nema potrebe za hubom. Svaki uredaj ima jedan, ili više FireWire portova, koji se, takode, ponašaju kao ponavljači. Svaki signal koji stigne na takav port biva regenerisan, a zatim se šalje do drugih portova. U stvari, kao što ćemo ukratko opisati, FireWire tehnologija ne zahteva host kompjuter.

Ono što će korisnik sigurno primetiti je da je FireWire kabl šestožilni (za razliku od četvorožilnog USB kabla). Postoje dva para upredenih parica, koje su poznate i kao TPA i TPB, i dve linije za napajanje. Kao i kod USB-a, linije napajanja mogu da obezbede napajanje za FireWire uredaje, tako da nije neophodan poseban izvor napajanja.



SLIKA 4.21 *Povezivanje FireWire uređaja*

Korišćenje dva para upredenih parica pomalo se razlikuje od onoga što smo prethodno videli. FireWire koristi metod kodiranja poznat kao kodiranje podataka sa strob signalom **(data strob encoding).** Na slici 4.22 prikazano je kako to funkcioniše. Podaci koji će biti preneti najpre se kodiraju jednim oblikom NRZ kodiranja. U ovom slučaju 1 je visoki signal, a O niski. Prijemnik dobija podatke preko TPA. Medutim, kao što smo ranije istakli, dugački konstantni signali (dugački nizovi jedinica, ili nula) mogu da izazovu odstupanja izmedu pošiljaoca i primaoca. Kod FireWire standarda drugi par upredenih parica TPB koristi se za prenos **strob signala.** Pošiljalac generiše strob signal, koji ostaje konstantan svaki put kada dolazi do promene u podacima sa 1 na 0, ili obratno. Ako nema promene u signalu podataka, dolazi do promene u strob signalu. Ovde je važno istaći da sa svakim ciklusom takta u pošiljaocu postoji promena ili u signalu podataka, ili u strob signalu. Prijemnik može da detektuje te promene korišćenjem logike isključivog ILI (exclusive OR), tako da se ponovo uspostavlja signal takta pošiljaoca. Prijernnik može da sinhronizuje svoj takt sa dolazećim bitovima; tako je osiguran ispravan prijem podataka. Ovo pomalo podseća na Mančester kodiranje, jer postoji mehanizam za samostalno podešavanje takta. Najveća razlika je u tome što Mančester kodiranje zahteva brzinu bauda dva puta veću od bitske brzine. Ovde su brzina bauda i bitska brzina jednake. Medutim, ova tehnika zahteva dodatni par upredenih parica.



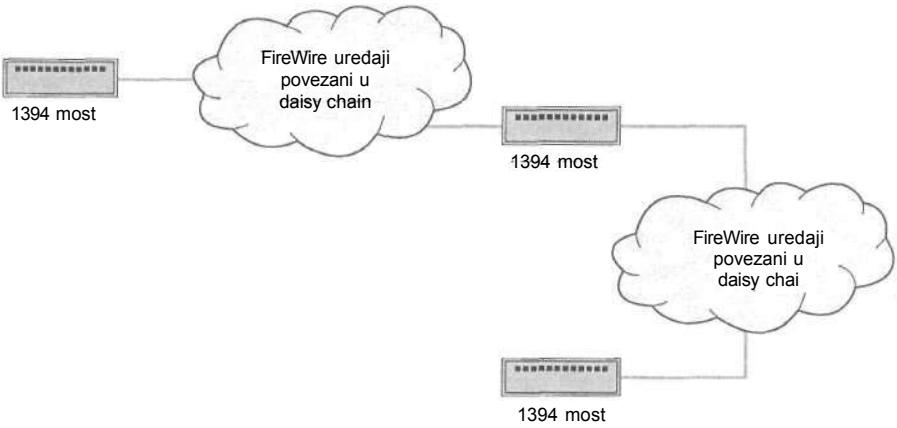
SLIKA 4.22 *Kodiranje strob signalom*

Suštinska razlika izmedu FireWire i USB-a je verovatno tip protokola. Kao što smo videli, USB funkcioniše u master/slave modu, a FireWire je peer-to-peer protokol. U našem slučaju to znači da komunikacija ne zavisi od jednog hosta, kao što je personalni kompjuter. Ako postoji FireWire konekcija izmedu kamere i eksternog disk drajva, podaci se mogu preneti direktno sa kamere do skladišta. Host kompjuter nije neophodan. Kompletni detalji su složeni i mi ćemo ovde dati samo opšti pregled protokola. Čitaoci koji su zainteresovani za detaljnija objašnjenja mogu da pogledaju reference [An99] i [St03], ili mogu da posete sajt www.apple.com/firewire.

Peer-to-peer dizajn omogućava širi opseg topologija za povezivanje. Na slici 4.22 već je prikazano nekoliko uređaja koje je moguće povezati pomoću daisy chaina. Na slici 4.23 data je opštija konfiguracija koja koristi 1394 bus mostove. Uredaji povezani metodom daisy chain mogu da formiraju bus grupu. Te grupe mogu da se povežu pomoću bus "mostova" - oni izoluju grupe jedne od drugih, tako da uređaji u jednoj grupi mogu da komunidraju nezavisno od uredaja druge grupe (paketi u opštem slučaju ostaju unutar grupe). "Mostovi" formiraju konekcije sloja 2 izmedu uredaja. U Poglavlju 10 detaljnije ćemo predstaviti tipove "mostova" i njihove funkcije. Za sada, "most" možete da smatrate uredajem koji omogućava nezavisno funkcionisanje uređaja u grupi, a zadužen je za prenos paketa adresiranih na uredaje iz druge grupe. Moguće je postaviti do 63 uredaja (koristi se 16-bitni ID) u okviru jedne grupe i do 123 različite bus grupe (sa 10-bitnim ID-om).

KOMUNIKACIJE FireWire podržava komuniciranje u dva razlidta moda: asinhronom i izohronom. FireWire definiše asinhrone komunikacije kao komunikacije koje uključuju razmenu paketa i potvrda. Opšti princip se sastoji u sledećem:

1. Slanje paketa
2. Čekanje na odgovor



SLIKA 4.23 *Više FireWire magistrala*

3. Ako se dobije potvrdan odgovor, smatra se da je paket primljen.

4. Ako je odgovor negativan (drugi tip paketa), pretpostavlja se da se nešto desilo sa paketom i inicira se njegovo ponovno slanje.

Detalji implementiranja ovakvih protokola mogu da budu složeni; o njima će biti više reči u Poglavlju 8. Ovde treba istaći da paketi stižu u proizvoljnim trenucima; ako neki moraju ponovo da se šalju, moguće je da će doći do fluktuacije u brzini prenosa.

Izohroni transfer smo već opisali. FireWire garantuje da se paketi šalju u pravilnim intervalima, tako da je zagarantovana određena bitska brzina. Nema ćekanja na potvrdu, niti se inicira ponovno slanje paketa.

Asinhroni paket ima zaglavlje koje sadrži 64-bitnu adresu. Šesnaest bitova identifikuje odredeni uređaj preko 10-bitnog ID-a magistrale, a šest bitova predstavlja ID čvora. Preostalih 48 bitova se koristi za referenciranje memorije, tako da je obezbeđen 256-terabajtni (2^{18}) kapacitet memorije. Izohroni paketi nemaju adresu. Umesto toga, svaki ima polje sa brojem kanala koji identifikuje ranije uspostavljeni tok između dva uređaja.

Jedan od razloga za definisanje brojeva kanala je osiguravanje traženog kvaliteta servisa za izohroni kanal. Kada dva uredaja zahtevaju izohronu komunikaciju, menadžer resursa dodeljuje broj kanala. Ako neka druga dva uređaja zahtevaju izohronu komunikaciju, dodeljuje im se drugi broj kanala. Medutim, šta se dešava ako menadžer resursa dodeli isuviše kanala za izohrone komunikacije? U tom slučaju, fizička bitska brzina neće biti dovoljna za podršku svih zahteva za prenos. Sve konekcije imaju konačan kapacitet za prenos bitova i zato se mora obezbediti da se ne nude obećanja koja se ne mogu ispuniti. Zbog toga, postoji ograničenje za broj dodeljenih izohronih kanala.

ARBITRAŽA Kao što smo ranije istakli, FireWire je peer-to-peer protokol, koji eliminiše zavisnost od hosta prilikom svih transfera podataka. Tako uredaji mogu sami da iniciraju prenos svojih podataka. Zvuči dovoljno jednostavno sve dok ne shvatite da se javljaju problemi ako više uredaja pokušava da inicira transfer u isto vreme. Kada se to desi, neophodna je arbitraža koja će utvrditi koji uređaj "pobeduje". Kako arbitraža funkcioniše?

Da bismo dali odgovor na to pitanje, vratićemo se malo unazad. Kada se novi uredaj poveže na postojeći daisy chain, svi uredaji sarađuju tako da se formira hijerarhijsko uredenje čvorova. Napomenimo da se te hijerarhije nalaze u grupama koje su medusobno razdvojene mostovima (uređaji svake grupe su nezavisni od uredaja iz drugih grupa).

Tačan način saradnje izmedu uredaja koji formiraju ovu konfiguraciju je prilično složen; u Poglavlju 10 detaljnije proučavamo tu temu. Za sada je dovoljno da znate da uredaji mogu da komuniciraju i da se ureduju u strukturi stabla, kod koje se jedan čvor ponaša kao koren. Napominjemo da uređenje mora da bude u vidu stabla i da nisu dozvoljene petlje. Nakon što se uređenje definiše, svaki čvor selektuje svoj ID broj na osnovu pozicije u hijerarhiji. Ovde nećemo detaljno objašnjavati kako se to izvodi.

Uređaj koji je postavljen kao koren stabla igra ulogu arbitra. Kada se traži pristup magistrali, šalje se zahtev arbitru radi izdavanja potvrde za pristup. Ako dva uređaja istovremeno traže pristup, arbitar donosi odluku na osnovu njihovog prioriteta. Prioritet se dodeljuje na osnovu udaljenosti uređaja od korena stabla (oni koji se nalaze bliže korenu stabla imaju veći prioritet od onih

koji se nalaze dalje). Ovde treba istaći da arbitar uvek može da donese odluku koji uredaj dobija pristup magistrali.

Pažljiviji čitalac će možda postaviti najznačajnije pitanje - kako je pomoću ovog metoda moguće podržati izohrone komunikacije ako uredaj koji želi da pošalje podatke ima nizak prioritet. Na kraju krajeva, i sama reč izohrono ukazuje na zagarantovani broj bitova po jedinici vremena. Uredaj sa visokim prioritetom (bliži korenu stabla) može da preuzme monopol nad magistralom, sprečavajuć uređaje nižeg prioriteta da ikada dobiju pristup. Odgovorleži u činjenici da je arbitraža koju smo ovde predstavili samo deo procesa i da funkcioniše zajedno sa još dva metoda arbitraže: nepristrasnom i urgentnom arbitražom.

Nepristrasna arbitraža koristi koncept fer intervala (količine vremena). Sledi tipična sekvenca dogadaja:

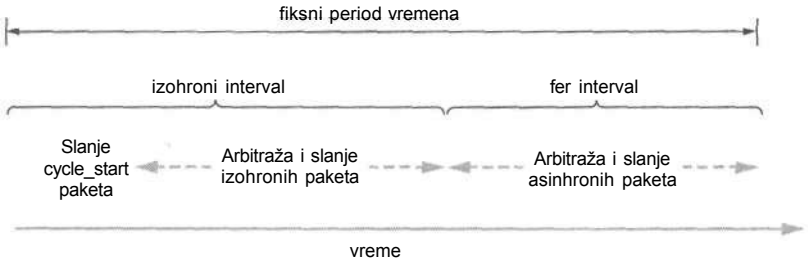
1. U početku fer intervala svi uredaji koji imaju pakete za slanje postavljaju flegove.
2. Svi uredaji koji se "nadmeću" za pristup magistrali šalju zahteve do arbitra.
3. Uređaj koji dobija magistralu šalje svoje pakete i dobija potvrdu o prijemu od primaoca (uredaj koji šalje potvrdu ne mora da se "nadmeće" za magistralu, jer je ona i dalje pod kontrolom uređaja koji je poslao pakete). Zatim, uredaj uklanja svoj fleg. Izbrisani fleg više nema pravo da postavlja zahteve za magistralom dok ne istekne zadati interval.
4. Eventualno, moguće je da svi uredaji dobiju po jednom pristup magistrali u toku zadatog intervala.
5. Ako nema uredaja koji i dalje čekaju na magistralu, ona je neaktivna za odredeno vreme. Na taj način se definiše kraj fer intervala.
6. Počinje novi interval i ceo proces se ponavlja.

Ova procedura garantuje da ni jedan uređaj neće imati monopol nad magistralom, niti da će u arbitraži neprestano "pobedivati" isti uredaj.

Da bi se, ipak, zadržali prioriteti uredaja, metod *urgentne avbitraže* omogućava odredenim uredajima da se konfigurišu kao urgentni. Svaki urgentni uredaj takode mora da postavi svoj fleg na početku fer intervala, ali se postavlja i vrednost brojača. Uredaj počinje "nadmetanje" za pristup magistrali. Razlika je u tome što kada dobije pristup magistrali, njegov se brojač umanjuje za 1. Ako je brojač i dalje pozitivan nakon što se paket pošalje i stigne potvrda o prijemu, uredaj može ponovo da se "nadmeće" za magistralu u okviru istog fer intervala.

Napomenimo da urgentno označavanje nije isto što i izohroni prenos. Ovim metodom je jednostavno omogućeno definisanje prioriteta za asinhrone pakete; i dalje nema garancija za specifičnu bitsku brzinu. Kod izohronih prenosa uredaj u korenu stabla ima ulogu i *mastera ciklusa.* Kao master ciklusa, on redovno šalje paket cycle_start (slika 4.24). Zapamtite da je uredaj u korenu stabla ujedno i arbitar i ima najviši prioritet, jer se svi ostali nalaze dalje od korena. Zato može sebi da potvrdi pristup magistrali svaki put kada mu je potrebna za slanje cycle_start paketa. Cycle_start paket označava start izohronog ciklusa. U vreme tog ciklusa uredaji sa izohronim paketom dobijaju šansu da pošalju po jedan paket. Pošto master ciklusa redovno stratuje izohorni ciklus, time se ispunjava zahtev za obezbedivanje izohronog prenosa.

Kada se izohroni paketi pošalju preko svih kanala, javlja se praznina, nakon koje počinje fer interval. U toj tački uređaji mogu da šalju *asinhrone pakete,* kao što smo već opisali, sve dok svi ne budu poslati, ili dok master ciklusa ponovo ne pošalje cycle-start paket.



SLIKA 4.24 *FireWire arbitraža*

Sigurno smo ovde izostavili brojne detalje; zainteresovani čitaoci mogu da saznaju nešto više o svemu ovome u referencama [An99] i [St03] i na Web sajtu www.apple.com/firewire.

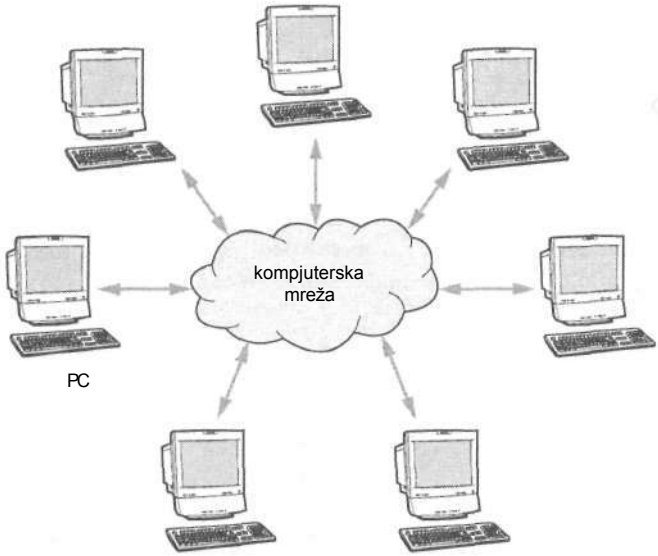
Postoji još veliki broj standarda, ali njihovim opisivanjem bismo premašili predvideni obim ove knjige; u referenci [St03] možete da pronađete listu od oko 100 ovakvih standarda. Opisali smo neke od najčešće korišćenih i najpoznatijih standarda, a opisaćemo još nekoliko nešto kasnije (na primer, u Poglavlju 13 X.25 mrežni interfejs i ISDN).

4.5 Multipleksiranje

Kada je rec o prenosu podataka, sigurno nece biti mkakvih problema da Vas ubedimo da je bolje imati što veće bitske brzine. Ako ste nedavno poboljšavali svoju konfiguraciju tako da imate brži procesor, veći disk, ili bolji modem, nema sumnje da Vam je kraće vreme pomoglo da budete efikasniji u radu. Neki od nas koji smo zaista dugo u svetu kompjutera sećaju se dana kada su se pojavili 3.5-inčni drajvovi. Mogućnost prebacivanja fajla za svega par sekundi bilo je neverovatno poboljšanje u odnosu na dotadašnje 5.25-inčne drajvove. Sa današnjim hard diskovima, ne postoji način da se zabavimo dok čekamo na snimanje bilo čega na 3.5-inčni disk.

Isto važi i za mreže i komunikacije. U opštem slučaju, važi izreka "što brže, to bolje". Ipak, brzina ima i svojih nedostataka. Prvo, izaziva veće finansijske izdatke i, daigo, postoji određena tačka slabljenja (nakon te tačke, korisnici ne mogu da koriste povećanu brzinu). Na primer, LAN mreže danas podržavaju gigabitske brzine, ali većina PC aplikacija nema kolidnu podataka koja bi imala koristi od tih brzina - čak i kada bi postojala, većina mrežnih kartica na PC-jima ne podržava tolike brzine.

Postoji mišljenje da ne treba mnogo brinuti zbog razvoja high-speed mreža, jer većina korisnika ne može da iskoristi njihov potencijal. Ovo rešenje ima ozbiljnu manu. Pretpostavimo da mreža sa slike 4.25 podržava bitsku brzinu od IO Mbps. Ako jedina aktivnost uključuje dva PC-ja koja komuniciraju na toj brzini, mreža je opravdala svoju svrhu.



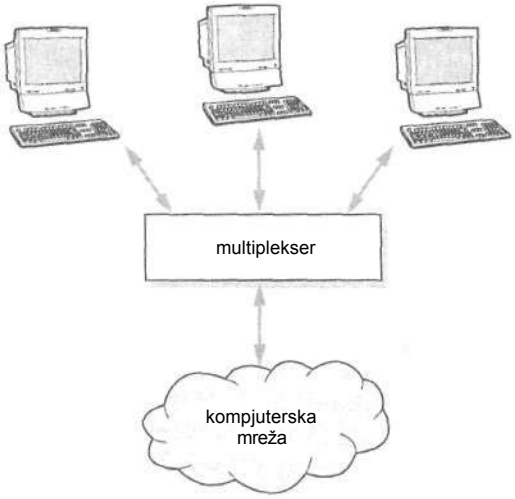
SLIKA 4.25 *Više korisnika komunicira preko mreže*

Međutim, ako nekoliko stotina PC-ja treba međusobno da komunicira, granica od 10 Mbps ce stvoriti "uska grla", koja se redukuju povećanjem bitske brzine. Ovde može da se povuče analogija sa velikim saobraćajnicama u velikim gradovima u vreme saobraćajnih špiceva. Ako se saobraćaj odvija brzinom od 4 km/h, red ispred naplatne rampe će biti veoma dugačak. Ako se odvija norrnalnom brzinom od 15 km/h, automobili neće morati dugo da čekaju na ulazak na autoput.

Drugi problem kod razvoja velikih bitskih brzina je što je neophodno razviti high-speed mreže, ali, pri tom, na neki način redukovati cenu njihovog povezivanja. Na slici 4.25 data je cena povezivanja svaka dva PC-ja. Na slici 4.26 prikazana je uobičajena alternativa kod koje je korišćen **multiplekser** (ponekad se, radi lakšeg izgovora, označava kao mux). To je uredaj koji rutira prenose od više izvora ka jednom odredištu. Na slici 4.26 izvori su PC-ji, a odredište je mreža. Osim toga, multiplekser rutira i prenose u suprotnom smem, od mreže do bilo kog PC-ja.

U opštem slučaju, izlazna linija multipleksera ka mreži obično može da podrži mnogo veće bitske brzine nego ulazne linije od PC-ja. Na taj način je moguće iskoristiti visoki kapacitet mreže obezbedivanjem jedne konekcije za više korisnika, sa manjom cenom po konekciji.

Ovaj opis multipleksiranja je samo jedan od nekoliko mogućih koji zavise od tipa signala i aktivnosti jedinica koje se vezuju za multiplekser. Opisaćemo specifične metode multipleksiranja i navesti primer telefonskih komunikacija na većim udaljenostima.



SLIKA 4.26 *Multipleksiranje uređaja sa manjom brzinom*

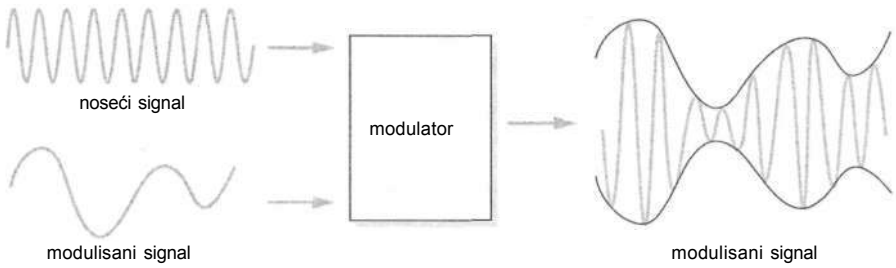
Multipleksiranje sa podelom frekvencije

Multipleksiranje sa podelom frekvencije (FDM - frequency-division multiplexing) koristi se sa analognim signalima. Verovatno se najčešće koristi kod televizijskih i radio prenosa. Multiplekser pnbvata analogne signale iz više izvora, gde svaki signal pripada odgovarajućem ospegu. Nakon toga se signali kombinuju u jedan složeniji signal sa mnogo većim opsegom. Rezultujući signal se prenosi preko nekog medijuma do svog odredišta, gde ga drugi multiplekser izvlači i rastavlja na individualne komponente.

Metod multipleksiranja uključuje nekoliko koraka. Prvo, raspoloživi opseg signala medijuma deli se na zasebne opsege, ili kanale. Na primer, opseg signala za televizijski prenos (54 do 806 MHz) deli se na 68 kanala od po 6 MHz. VHF kanali 2 do 13 odgovaraju opsezima od po 6 MHz izmedu 54 i 215 MHz. UHF kanali 14 do 69 odgovaraju opsezima od po 6 MHz izmedu 470 i 806 MHz. Svaki kanal odgovara jednom ulaznom signalu multipleksera.

Zatim, za **svaki kanal** se definiše noseći signal. Menja ga (moduliše) odgovarajući ulazni signal da bi bio kreiran drugi signal (modulisani signal). Postoji nekoliko načina da ovo izvedete. Na primer, na slici 4.27 ilustrovana je amplitudska modulacija. Noseći signal ima defmisanu frekvenciju, obično centriranu u opsegu kanala. Amplituda signala se menja naizmenično između vrednosti koje zavise od maksimalnih i minimalnih vrednosti originalnog signala.

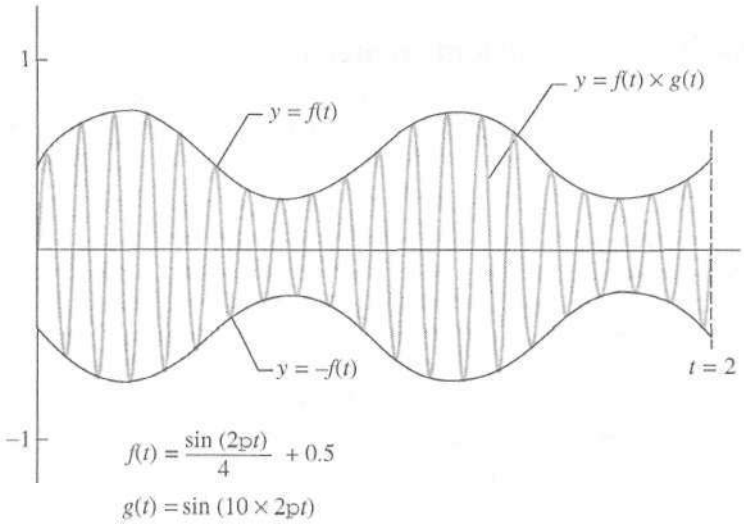
Potpuno razumevanje amplitudske modulacije zahteva bolje poznavanje matematičke reprezentacije talasnih oblika i Furijeovih redova. Pošto detaljnije objašnjenje prelazi predviđeni obim ove knjige, ako želite da studioznije proučite ovu temu, pogledajte reference [StOO], [Wa98] i [St96]. Mi ovde možemo da ilustrujemo proces jednim jednostavnim primerom.



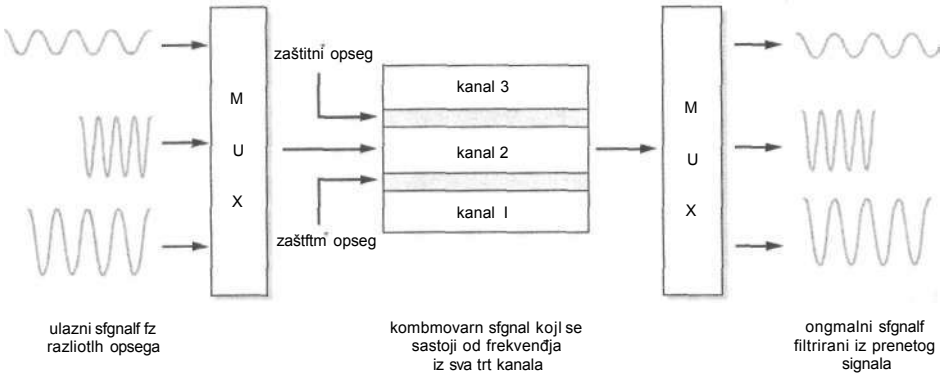
SLIKA 4.27 *Amplitudska modulacija*

Razmotrimo signal koji se predstavlja formulom /(() = [sin(2Ttt)/4] + 0.5. Na slici 4.28 prikazan je grafik u intervalu od $t = 0$ do $t = 2$. Pretpostavimo da se noseći signal može predstaviti formulom $g(t) = \sin(]0 \times 2\text{-}Kt)$. Na istoj slici nije dat i grafik za $g(t)$, ali, kada bi se iscrtao, oscilovao bi 20 pula izmedu l i l, dok se t kreće izmedu 0 i 2. Množenjem $f(t)$ i $g(t)$ generiše se modulisani signal prikazan na slici 4.28.

U opštem slučaju, signali imaju mnogo više frekvencije i odgovaraju složenijim sumama sinusnih funkcija. I pored toga, proces amplitudske modulacije ostaje, u suštini, isti kao i u ovom našem jednostavnom primeru. Medu ostalim tehnikama modulacije su frekventna i fazna modulacija. Kao šlo možete i da pretpostavite, frekventna menja frekvenciju nosećeg signala, a fazna fazni pomak nosećeg signala. Više detalja o ovoj temi možete nađ u referencama [StOO] i [Wa98].



SLIKA 4.28 *Grafik modulisanog signala*



SLIKA 4.29 *Multipleksiranje sa podelom frekvencije*

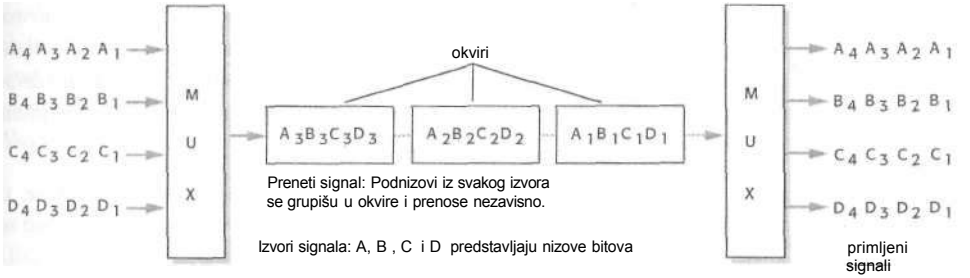
U poslednjem koraku multipleksiranja sa podelom frekvencije modulisani signal iz svih ulaza se kombinuje u jedan, složeniji analogni signal (slika 4.29). Njegove frekvencije leže u opsezima svih kanala. Sami kanali su razdvojeni zaštitnim opsezima **(guard bands),** neiskorišćenim delovima frekventnih opsega, da bi bila sprečena interferenca izmedu susednih kanala. Nakon toga se rezultujući signal prenosi tamo gde ga prima drugi multiplekser. Zatim se koriste propusni filteri koji izvlače modulisane signale. Konačno, signali se demodulišu i obnavlja se originalni signal. U slučaju televizije i radija kanali, ili selektori frekvencija određuju koji se originalni signali konvertuju u zvuk i sliku.

Multipleksiranje sa podelom vremena

Multipleksiranje sa podelom vremena (TDM - time-division multiplexing) kombinuje i zajednički prenosi više ulaznih signala, kao kod FDM-a. Međutim, TDM se koristi sa digitalnim signalima. TDM održava fizički različite signale, ali ih logički pakuje zajedno, za razliku od FDM-a, koji ih kombinuje u jedan, složeniji signal.

Na slici 4.30 ilustrovan je TDM. Pretpostavimo da Aj, B_1 Q i D_1 *[i* = 1, 2, 3, ...) predstavljaju nizove bitova iz različitih izvora. U multiplekseru se privremeno baferuje nekoliko bitova iz svakog izvora. Multiplekser skenira svaki bafer, smešta bitove iz svakog bafera u okvir, a, zatim, šalje okvir. Kada to završi, započinje formiranje novog okvira ponovnim skeniranjem ulaznih bafera da bi se proverilo da li su stigli novi podaci. Ako je tajming dobar, biće konstruisan novi okvir tačno na vreme da se prenese odmah nakon prethodnog okvira. Ovaj proces održava izlaznu liniju aktivnu i omogućava potpuno korišćenje njenog kapaciteta.

Na slici 4.30 nizovi bitova A_1, B_1, C_1 i Dj se baferuju zasebno. Multiplekser ih pakuje u jedan okvir i prenosi ga. Nakon toga, sledi prikupljanje novih nizova A_2, B_2, C_2 i D_2 i slanje novog okvira. Proces se nastavlja sve dok stižu novi nizovi bitova.



SLIKA 4.30 *Multipleksiranje sa podelom vremena*

Dizajn multipleksera zavisi delom od brzine prenosa ulaznih i izlaznih signala. Na primer, ako izvorni bitovi iz kombinovanih ulaza pristižu brže nego što prethodni okvir može da bude poslat, okviri se generišu mnogo brže nego što se mogu proslediti. Ako multiplekser nema dovoljan kapadtet za smeštanje dodatnih okvira, oni će biti izgubljeni. Multiplekser se ne sme "obasipati' informacijama brže nego što on može da ih otpusti. Sa druge strane, ako izvorni bitovi dolaze isuviše sporo, prethodni okviri će biti poslati i multiplekser će ili čekati da stigne dovoljan broj bitova, ili će biti poslat parcijalno kompletan okvir. U svakom slučaju, izlazna linija se ne koristi sa kompletnim kapacitetom.

U optimalnoj situaciji kombinovana ulazna brzina (suma brzina od svih izvora) jednaka je izlaznoj brzini. Pretpostavimo da je T_1 rulazna brzina iz i-tog izvora, a da je r_{output} brzina kojom multiplekser prenosi okvire. Matematički, to se izražava formulom

$$\sum_{i=1}^{n} r_i = r_{output}$$

Na primer, ako podaci iz 10 izvora stižu brzinom od 10 Mbps, trebalo bi da ih multiplekser šalje brzinom od 100 Mbps (ukratko ćemo opisati alternativu).

Drugi faktor koji utiče na dizajn multipleksera je veličina komponenata okvira. U jednoj varijanti se A_n B_l, Q i D, definišu sa osam bitova, odnosno jednim bajtom. U tom slučaju se multiplekser naziva bajtni multiplekser **(byte multiplexer).** U drugim slučajevima su komponente A;, Bj, C_1 i D_1 veće, raspolažući većim brojem bajtova (blok). Zato se tada koristi naziv blokovski multiplekser *[block multipleier).*

Statistički multiplekseri

Prethodno smo istakli da optimalni dizajn zahteva da suma ulaznih brzina bude jednaka izlaznoj brzini. Medutim, to ponekad nije praktično. U prethodnom primeru ste primetili da bitovi stižu iz svakog izvora kontinuelno, ali u mnogim slučajevima dešava se da naglo pristižu u odredenim trenucima, uz povremene periode neaktivnosti.

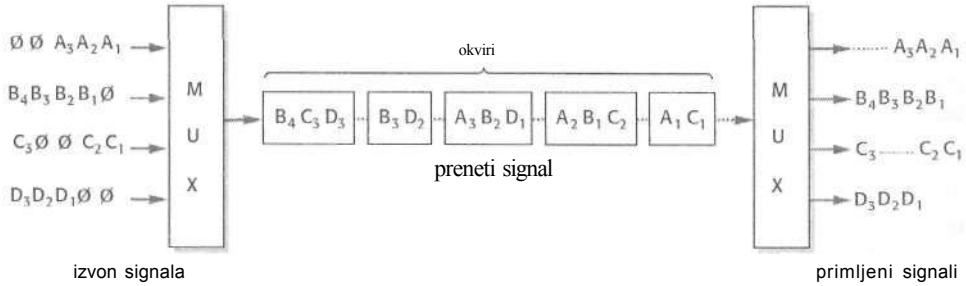
U takvim slučajevima postoje dva pristupa za multipleksiranje podataka. Prvi pristup nalaže dizajniranje multipleksera tako da se izbegnu prazni baferi i da se deo okvira ostavi prazan.

Na primer, na slici 4.30 možemo da pretpostavimo da je treći izvor bio neaktivan, tj. da nema podataka sa linija C_1, C_2, C_3, i C_4. U svakom okviru postoji prostor rezervisan za te bitove, ali ne sadrži nikakve korisne informacije. Ovo omogućava očuvanje fiksne veličine okvira i uprošćavanje protokola. Očigledni nedostatak je što beskorisne informacije zauzimaju prenosni medijum i tako se "traći" propusni opseg.

U sklopu drugog pristupa multiplekser skenira bafere i kreira okvire promenljive veličine, u zavisnosti od toga koliko bafera sadrži podatke. On se naziva statistički multiplekser (koristi se i termin *koncentrator).**

Na slici 4.31 ilustrovano je kako ovo funkcioniše. Ovde su svi izvori aktivni, ali ne istovremeno. Simbol ∅ ukazuje da iz izvora ne stižu nikakve informacije. Inicijalno, nizovi bitova Aj i C_1 se baferuju, ali su drugi prazni zbog neaktivnosti izvora. Zato multiplekser postavlja A_1 i C_1 u okvir i šalje ga. U meduvremenu, stižu A_2 i C_2, zajedno sa B_1. Multiplekser ih sastavlja u jedan veći okvir i šalje ga. U ovom trenutku izvor C je neaktivan, ali izvor D postaje aktivan. Sada stižu A_3, B_2 i D_1. Kao i ranije, multiplekser ih postavlja u okvir i šalje ga. Proces se nastavlja sve dok postoje aktivni izvori.

Komplikacija u ovom pristupu nastupa zbog toga što izvori nemaju fiksnu poziciju u okviru. Na primer, na slici 4.30 bitovi iz svakog izvora uvek zauzimaju istu poziciju u okviru. Na slici 4.31 to nije tako. Na primer, bitovi iz izvora B ponekad zauzimaju drugu, ili treću poziciju (gledano sa desne strane). U takvim slučajevima, format okvira je složeniji i zahteva dodatne informacije, kao što je odredišna adresa. Prijemni multiplekser mora da ima dodatnu logiku za traženje adrese i rutiranje informacija u odgovarajućem smeru.



SLIKA 4.31 *Statističlio multipleksiranje sa podelom vremena*

* Strogo govoreći, ta dva termina se razlikuju. Koncentrator je "inteligentniji" slatistički multiplekser, koji može da obavlja i druge "poslove", kao što su verifikacija, potvrda i kompresovanje podataka. Ove teme čemo obraditi u poglavljima 5, 6, 7 i 8. Konkretna definicija često zavisi od toga sa kim se razgovara. Mi ovde nećemo isticati razliku. U stvari, ponekad se koristi termin asinhroni multiplekser sa podelom vremena.

Po defmiciji, statistički multiplekser ne mora u potpunosti da iskoristi svoj izlazni kapacitet. U ekstremnom slučaju, ako ni jedan izvor nije aktivan, nema prenosa. Verovatnoća da ni jedan izvornije aktivan zavisi od ukupnog broja izvora. Proniciljiviji dtalac može da primeti da bismo mogli povezati dodatne izvore da bi ova verovatnoća bila umanjena i da bi izlaz bio zaposleniji. Ako se to desi, onda je

$$\sum_{i=1}^{n} r_i > r_{output}$$

Ulazni kapacitet multipleksera je sada veći od izlaznog.

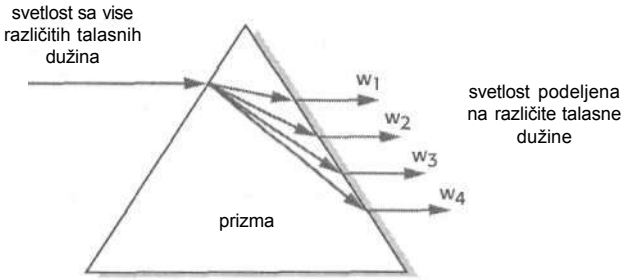
Veća brzina na ulazu ne mora da predstavlja problem. Zapamtite da r_i predstavlja kapacitet i-tog izvora, a ne stvarnu brzinu. Ako Je izvor neaktivan, stvarna bitska brzina je 0. Prilikom dizajniranja je pretpostavljeno da nisu svi izvori aktivni istovremeno ako je suma ulaznih brzina veća od r_{output}. Idealno bi bilo da je kombinovana ulazna brzina od aktivnih izvora jednaka r_{output}. Pošto aktivnost zavisi od korisnika, teško je predvidljiva (idealna aktivnost se teško postiže). U nekim slučajevima će kombinovana ulazna brzina od aktivnih izvora biti manja, ili, čak, veća od r_{output}. U drugom slučaju se moraju dizajnirati dodatna logika i baferi u kojima će se podaci privremeno baferovati. Zbog ovoga se statistički multiplekseri ponekad nazivaju koncentratori: u njima se koncentriše dodatna količina podataka u odredenim kratkim periodima.

Analizu statističkih multipleksera otežava činjenica da izvori podatke šalju nasumice. Moguće je postaviti brojna pitanja. Koliko često kombinovane ulazne brzine premašuju izlaznu brzinu? Koliko često se dešava da su svi izvori zauzeti? Koliko baferi moraju da se koriste za privremeno smeštanje podataka? Kolika su kašnjenja kada dode do naglog pristizanja veće količine podataka? Jedan pristup analizama podrazumeva korišćenje teorije čekanja *(queueing iheory);* to je matematička grana koja definiše modele za proučavanje događaja, kao što je čekanje na linijama *(redovi čekanja)* dok se ne desi odredeni dogadaj. Može da se primeni u raznim oblastima, uključujući komunikacione sisteme kod kojih ulazni nizovi podataka mogu da pristižu u nasumičnim uzorcima. U takvim slučajevima se događaji na koje se čeka prenose preko izlaznih linija. U referencama [StOO], [WaOO] i [Ma72] možete da pronadete uvodnu raspravu o teoriji čekanja.

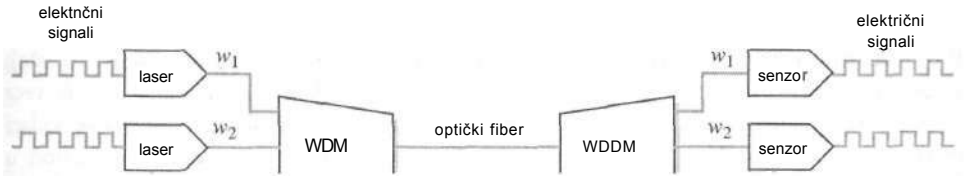
Multipleksiranje sa podelom talasnih dužina

Sledeći tip multipleksiranja je zasnovan na zakonima optike, ali ima sličnosti sa multipleksiranjem sa podelom frekvencije. Naziva se multipleksiranje sa podelom talasnih dužina i zasniva se na nekoliko svojstava vidljive svetlosti. U Poglavlju 2 predstavili smo prelamanje svetlosti - promenu smera do koje dolazi kada svetlost prelazi iz jednog medijuma u drugi. Ugao prelamanja zavisi ne samo od optičke gustine dva medijuma, već i od talasne dužine svetlosti. Većina svetlosnih zraka ima više različitih talasnih dužina. Ova dva svojstva opisuju šta se dešava kada svetlost prolazi kroz prizmu (slika 4.32), nakon čega se deli na različite boje. Različlte talasne dtržine se prelamaju pod različitim uglovima, tako da pod različitim uglovima i napuštaju prizmu. Na istom principu se formira i duga kada sunce sija za vreme kiše.

Na slici 4.33 ilustrovano je kako funkcioniše multipleksiranje sa podelom talasnih dužina. Postoji nekoliko izvora električnih signala, koji se vode na ulaz lasera (ili LED diode). Kao što je rečeno u Poglavlju 2, laser reaguje na električne signale i proizvodi svetlosne impulse.



SLIKA 4.32 *Prelamanje svetlosti kroz prizmu*



SLIKA 4.33 *Multipleksiranje sa podelom talasnih dužina*

Ovde je razlika u tome što laser stvara svetlosne impulse različitih talasnih dužina W_1, w_2, ■ ■■, W_n, Svetlost različitih talasnih dužina dovodi se na ulaze multipleksera sa podelom talasnih dužina (WDM-wave-division multiplexer), koji kombinuje različite svetlosne izvore u jedan. Ta svetlost (koja se sastoji od nekoliko talasnih dužina) prenosi se preko optičkog fibera, kao što je opisano u Poglavlju 2.

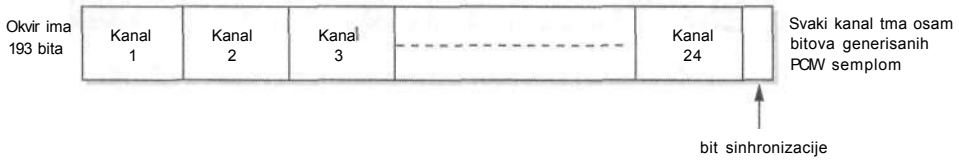
Proces je obmut na drugoj strani. Svetlost iz optičkog fibera ulazi u demultiplekser sa podelom talasnih dužina (WDDM - wave-division demultiplexer), koji razdvaja talasne dužine slično kao u slučaju prizme (iako je WDDM mnogo složeniji). Nakon toga, svetlost svih talasnih dužina nastavlja da se prenosi ka željenom odredištu.

Bitske brzine koje se postižu kod ove tehnologije imaju potencijala da postanu enormne. Standardni optički fiber već ima kapacitet za prenos desetina gigabita u sekundi. Dok je ova tehnologija još u razvoju, postoji potencijal za stotine gigabita u sekundi, a možda ćemo jednog dana govoriti o terabitima (1000 Gbps) u sekundi.

4.6 Digitalni nosioci

TI

U ovom odeljku opisaćemo nekoliko standarda koji se koriste u komunikacijama na velikim udaljenostima. Mnogi od nas misle da je telefonski sistem dizajniran za prenos digitalizovanih govornih signala preko high-speed medijuma, kao što su optički fiber, ili mikrotalasni prenosi.

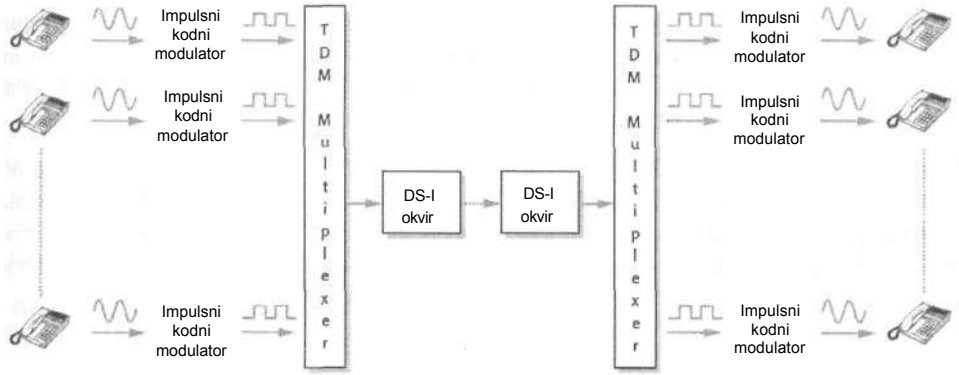


SLIKA 4.34 *DSl okvir*

U stvari, AT&T je razvio složene hijerarhijske komunikacione sisteme koji se koriste za multipleksiranje govornih signala i njihov prenos preko cele teritorije SAD. Sistem se koristi i u drugim zemljama, kao što su Kanada, ili Japan. Ipak, druge zemlje i dalje koriste sličan, ali drugačiji sistem definisan ITU-T standardima.

Sistem koristi multipleksiranje sa podelom vremena za kombinovanje većeg broja govornih kanala u jedan okvir. Medu brojnim načinima da se to uradi jedan pristup koristi Tl prenos i DSl signaliziranje. Oznake *Tl* i *DSl* tiču se kola i signala, respektivno. Na primer, na slici 4.34 dat je DSl okvir u kome se govorni podaci digitalizuju impulsnom kodnom modulacijom (predstavljena je u Poglavlju 3). Sadrži 193 bita podeljena u 24 slota (po jedan za svaki kanal) od po osam bitova. Tako ostaje jedan *dodatni bit* koji se koristi za sinhronizaciju.

Na slici 4.35 prikazano je kako funkcioniše Tl noseći sistem. Osmobitni govorni semplovi su uzeti iz sva 24 kanala brzinom od 8.000 u sekundi. Svaki sempl zauzima jedan slot u DSl okviru. Prema Nikvistovoj teoremi, predstavljenoj u Poglavlju 3, ovo je dovoljno za održavanje svih informacija u originalnom govornom analognom signalu.



SLIKA 4.35 *Tl noseći sistem*

Tabela 4.4: Severnoamerički komunikacioni nosioci

Nosilac	Format okvira	Broj kanala	Brzina prenosa podataka (Mbps)
TI	DSI	24	1.544
TIc	DSIC	48	3.152
T2	DS2	96	6.312
T3	DS3	672	44.376
T4	DS4	4032	274.176

Sukcesivni semplovi su smeštani u različitim DSl okvirima. Tako se govorne poruke prenose pomoću više DSl okvira do drugog multipleksera. Ovaj multiplekser izvlači bitove iz svakog slota i rutira ih do odgovarajućeg odredišta, gde se eventualno konvertuju nazad u analogne signale. Rezultat se konvertuje u originalni zvuk koji odgovara glasu osobe.

Kolika je bitska brzina Tl nosećeg sistema? Svaki osmobitni slot je generisan brzinom od 8.000 u sekundi, za brzinu od 64 Kbps. Da bi se podržala ta brzina, Tl mora da prenosi DSl okvir svakih 1/8.000 delova sekunde, ili 8.000 okvira u sekundi. Drugim rečima, mora da prenese 8.000 x 193 bita svake sekunde za brzinu prenosa podataka od 1,544 ivlbps. Sa većim brojem kanala i većim brzinama prenosa postoje i drugi nosioci i oznake signala. U tabeli 4.4 dat je pregled nekih od njih. Obično se multipleksiraju signali od sporijih nosilaca u one na većim brzinama. Na primer, T3 nosilac može da multipleksira 7 DS2 okvira, 14 DSlC okvira, ili 28 DSl okvira, što daje mogućnost za prenos 672 kanala u svakom okviru.

Govomi podaci nisu jedina vrsta podataka koju je moguće preneti. Mnoge kompanije iznajmljuju telefonske linije za prenos digitalnih informacija između kompjutera. U stvari, princip na kome se zasniva faks mašina konvertuje slike sa papira u digitalne signale i prenosi ih preko telefonske linije.

Finalna napomena je da broj kanala u svakom sistemu može da se poveća korišćenjem različitih tehnika modulacije. Impulsna kodna modulacija digitalizuje govorne informacije na brzinama od 64 Kbps. Adaptivna diferencijalna impulsna kodna modulacija digitalizuje govorne informacije na 32 Kbps, tako da svaki noseći sistem može da podrži dva puta više kanala nego što je navedeno u tabeli 4.4.

SONET

Verovatno jedan od najznačajnijih i najpoznatijih digitalnih nosećih sistema je SONET (Synchronous Optical Netvvork - sinhrona optička mreža). Razvio ga je Bellcore (Bell Communications Research) i predstavlja ANSI standard tehnologije sa komutacijom kola koja se koristi u komunikacionim sistemima za prenos podataka na velikim udaljenostima.

Može da poveže high-speed radne stanice, Internet rutere i telefonske i ATM (Asynchronous Transfer Mode) komutatore.* U stvari, mnogi Internet upiti i telefonski pozivi putuju preko SONET konekcije.

Sličan sistem **SDH (Synchronous Digital Hierarchy)** je razvijen ubrzo nakon SONET-a, a predstavlja ITU-T standard. Verovatno je najpoznatiji u evropskim zemljama. Postoje razlike izmedu ovih sistema, ali na ovom nivou rasprave nisu mnogo bitne. U stvari, često se koristi oznaka SONET/SDH. Mi ćemo se fokusirati na opšti pregled SONET-a, ali zainteresovani čitaoci mogu da pronađu više detalja u referenci [Go02].

Kao što može da se nasluti iz samog naziva, SONET je dizajniran kao nosilac koji koristi optičke fiber komunikacije. Osim toga, svi linkovi u SONET mreži funkcionišu sa istim taktom, čime je obezbedena sinhronizacija, tj. svi SONET predajnici šalju i primaju podatke u skladu sa zajedničkom frekvencijom takta. SONET ima neke karakteristike koje još uvek nisu proučavane u ovoj knjizi. Medutim, pre nego što predemo na detaljniju raspravu o komunikacijama i strukturi paketa i okvira, moramo da damo opšti pregled.

Postoji više nivoa SONET signalizacije: primarni mod za signalizaciju je STS-I (sinhroni transportni signal 1 nivoa). SONET definiše STS-I okvirkao okvir sa 810 bajtova (ukratko ćemo predstaviti i format). Zbog sinhrone prirode tehnologije, SONET predajnik šalje jedan okvir svakih 125 Lisec (10 * sekundi). To znači da SONETšalje jedan okvir sa 8 x 810 bitova 8.000 puta u sekundi. Množenjem ovih brojki dobija se bitska brzina od 51,84 Mbps, što je bazna brzina za STS-I. Ostale oznake signala su STS-3, STS-9, STS-12, STS-18, STS-24, STS-36, STS-48, STS-92 i STS-192. U opštem slučaju, STS-n signal ima bitsku brzinu n puta veću od bazne brzine STS-I signala, ili n x 51,84 Mbps. Na primer, STS-3 signal ima bitsku brzinu od 155,52 Mbps. Najviši nivo, STS-192 obezbeđuje bitsku brzinu od oko 9,953 Gbps. Nivoi signalizacije će se sigurno povećavati kako tehnologija bude napredovala. Za signale viših nivoa je tipično da enkapsuliraju nekoliko nižih signala. Na primer, STS-3 okvir često sadrži tri STS-I okvira.

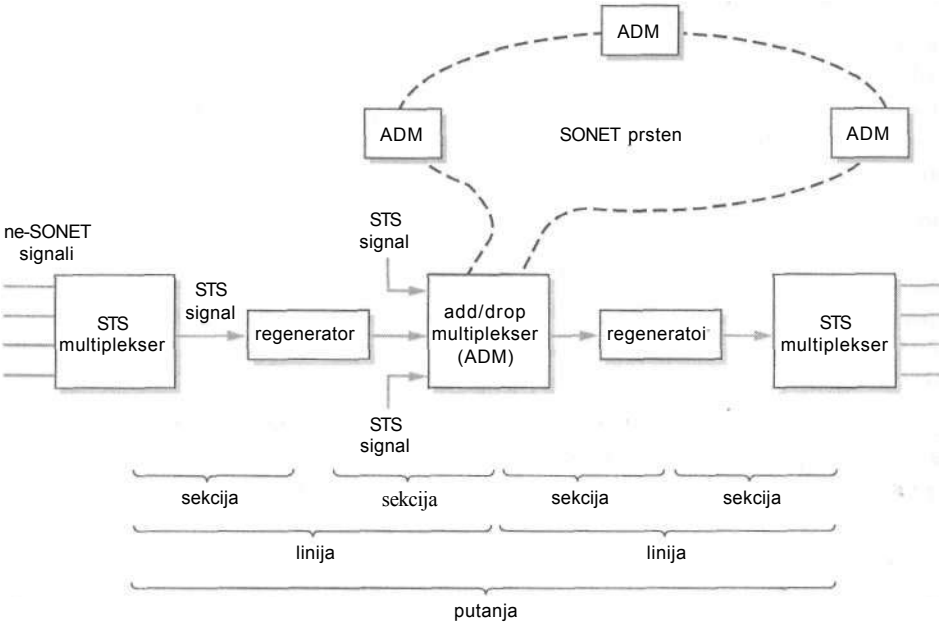
U zavisnosti od konteksta rasprave, može da se koristi oznaka OC-n (optički nosilac n). Obično STS oznake ukazuju na električne signale uređaja koji imaju optičku konekciju, a OC oznake predstavljaju stvarni optički signal. Zato nije neuobičajeno da se termini STS-n i OC-n koriste nazimenično, tj. STS-3 električni signal odgovara OC-3 optičkom signalu; u svakom slučaju, bitska brzina je 155,52 Mbps.

TIPOVI UREDA)A Pre nego što damo opšti pregled SONET tehnologije, najpre ćemo opisati tri primarna tipa uredaja u SONET mreži i kako oni zajedno funkcionišu (slika 4.36). Ovi uredaji definišu i slojeve na kojima SONET funkcioniše. Koriste se sledeća tri tipa uredaja.

- Regenerator Regenerator je uredaj koji regeneriše optičke signale. On je potreban zato što se optički signali, poput električnih signala, degradiraju dok "putuju" kroz optički fiber. Ako je fiber isuviše dugačak, signal se degradira do tačke u kojoj postaje neprepoznatljiv.

* U Poglavlju 13 predslavićemo ATM.

Kod dugačkih fibera (dužih od nekoliko desetina milja) regenerator mora da se koristi za regenerisanje bitova u SONET okviru.



SLIKA 4.36 *SONET konekcije i slojevi*

U jednoj sekvenci može da se koristi više regeneratora (na slici 4.36 prikazan je samo jedan regenerator izmedu dva uredaja). Prema SONET terminologiji, sekcija je deo mreže izmedu bilo koja dva susedna uredaja koja mogu da regenerišu signal. Moramo da napomenemo da regenerator nije u potpunosti uređaj sloja 1, jer modifikuje nekoliko bitova u svakom okviru koji regeneriše. Ukratko ćemo opisati detalje.

- **Add/drop multiplekser (ADM)** SONET ADM-i mogu da formiraju prsten, često povezujući tačke u oblastima velikih metropola, u državama, ili, čak, na područjima više država. SONET koristi ADM za izvlačenje saobraćaja izvan prstena i mešanje sa postojećim saobraćajem u prstenu. ADM funkcioniše i u suprotnom smeru, obezbedujući izlaz za saobraćaj u prstenu. Iako se koristi termin multiplekser, uredaj se malo razlikuje od *multipleksera* sa podelom vremena koji smo ranije opisali. U prethodnom slučaju multiplekser kombinuje signale iz različitih izvora u jedan zajednički okvir, a demultipleksiranjem se izvlače sve komponente signala i rutiraju se ka zasebnim odredištima. U ovom slučaju ADM ne mora da multipleksira i demultipleksira sve signale u zajednički okvir. Okvir koji većsadrži multipleksirane podatke može da "putuje" preko prstena i da stigne do ADM-a. ADM može da doda neke podatke u taj okvir. Drugim rečima, ne multipleksira se ceo okvir, većsamo njegov deo. Demultipleksiranje funkcioniše analogno. Moguće je da se iz okvira izvlače samo određeni podaci; nije neophodno demultipleksirati sadržaj celog okvira.

Ovo je analogno javnom gradskom saobraćaju. Na početku svi putnici ulaze u autobus i pronalaze mesta za sedenje (potpuno multipleksiranje), a na zadnjoj stanici svi izlaze (potpuno demultipleksiranje). Medutim, duž linije putnici ulaze, ili izlaze na usputnim stanicama (add/drop multipleksiranje). Vozač autobusa ne zahteva od svih putnika da izadu i da se ponovo vrate u autobus.

- **STS multiplekser** STS multiplekser uzima signal iz više različitih izvora i generiše STS okvir kao izlaz. Deo mreže koji povezuje dva STS multipleksera naziva se putanja, a takvi uređaji se ponekad nazivaju oprema za priključivanje na putanju *(path terminating equipment),* ili SONET terminali. Signal iz svakog eksternog izvora se multipleksira u STS okvir pod nazivom tributary. Tributary može da predstavlja eksterne izvore, kao što su ATM niz, DSl servis, ili bilo koja vrsta različitih protokola.

Ovi tipovi uredaja definišu tri sloja SONET operacija, koji mogu da se uporede sa funkcijama protokola na drugom sloju veze podataka. *Sloj sekcije* je implementiran na svim uređajima i najniži je od ova tri sloja, a zadužen je za upravljanje saobraćajem duž fizičke sekcije mreže (direktni *optički link).* Obezbeduje definisanje okvira (generisanje specijalnih uzoraka bitova koji označavaju početak okvira) i neke provere parnosti da bi bile otkrivene eventualne greške koje nastaju u toku prenosa. U Poglavlju 6 detaljnije analiziramo proveru parnosti. Sloj linije je implementiran u ADM, ili STS multiplekseru, a zadužen je za STS prenos. On locira korisne informacije u okviru i bavi se prenosom između krajnjih tačaka; odgovoran je za multipleksiranje, definisanje tipova i strukture korisnih informacija i njihovo smeštanje u okviru. Osim toga, obezbeduje neke funkcije održavanja i praćenja performansi.

Korisne inforrnacije (payloads) i okviri Sledeći korak je analiza SONET okvira i korisnih informacija u njima. Kao što smo ranije istakli, SONET STS-I okvir sadrži 810 bajtova, a okviri se prenose u intervalima od 125 irsec. Drugim rečima, SONET predajnik stvara konstantni niz STS-I okvira brzinom od 8.000 u sekundi. Ovo izaziva malu dilemu, zato što informacije ne stižu uvek iz eksternih izvora na sinhroni način. Zato se nameće logično pitanje kako da se podaci koji stižu asinhrono enkapsuliraju u okvirima koji se prenose sinhrono.

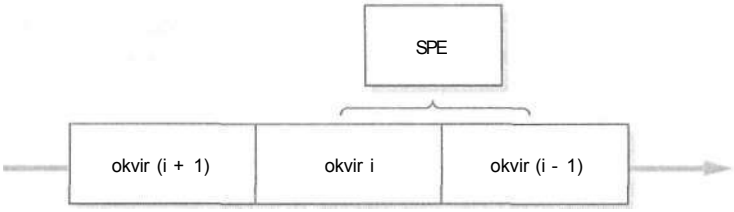
Jedna opcija može da bude pokušaj sinhronizacije svih uređaja, ali opseg i domet takvih uredaja taj zahvat čini nepraktičnim. Druga opcija je da se koristi prijemni bafer koji će ispuštati podatke u okvire dok se budu prenosili. Medutim, ako podaci stignu ubrzo nakon što predajnik počne da šalje okvir, podaci moraju da čekaju na slanje sledećeg okvira, a odlazeći okvir može da bude prazan. Tako nastaju kašnjenja u prenosu podataka.

Na slici 4.37 prikazan je pristup koji SONET koristi. Podaci koji se smeštaju u okvir označavaju se kao SPE **(synchronous** payload envelope) i definišu se na sloju putanje. Pošto SPE i odlazeći okvir možda nisu sinhronizovani, SPE može da bude smešten u dva sukcesivna okvira. Način na koji se ovo izvodi zavisi od SPE i formata okvira. Na slici 4.38a prikazana je struktura STS-I okvira: 810 bajtova se organizuje u 90 kolona i devet redova. Prva tri bajta u prva tri reda predstavljaju dodatak sekcije i koriste se za aktivnosti sloja sekcije. Prva tri bajta u svakom od poslednjih šest redova predstavljaju dodatak linije i koriste se za funkcije sloja Hnije.

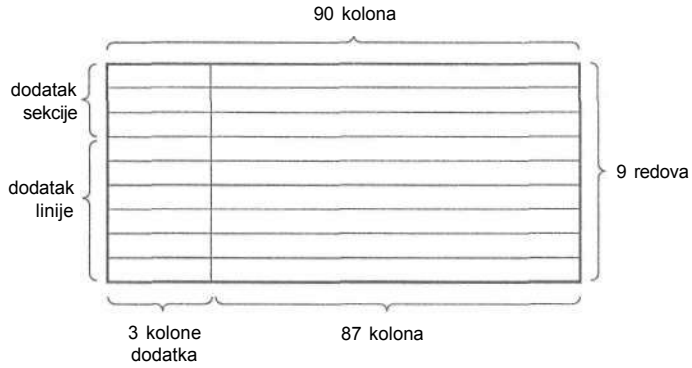
Na slici 4.38b prikazana je relacija izmedu SPE i dva sukcesivna STS-I okvira. U opštem slučaju, SPE bajtovi (zasenčena oblast) su organizovani u devet redova sa po 87 kolona i mogu u potpunosti da se uklope u okvir. Medutim, SPE može da se podeli izmedu dva sukcesivna okvira, gde se SPE red deli preko dva reda u okviru (ili prvog i poslednjeg reda u sukcesivnim okvirima). Pošto se okviri šalju u pravilnim intervalima, SPE bajtovi se i dalje prenose u konzistentnom nizu. Bitno je to da ih SONET slojevi prepoznaju kao bajtove koji su eventualno distribuirani preko dva sukcesivna okvira. Napomenimo i da svaki SPE ima dodatke za operacije sloja putanje. Ovaj proces pomalo podseća na čekanje autobusa na naznačenoj lokaciji. Medutim, umesto da autobus stane da prihvati putnike, kolona autobusa putuje konstantnom brzinom, a ljudi koji čekaju na liniji moraju da uskaču u autobuse u vožnji. Ako grupa putnika koja čeka na autobus treba da ide na rekreativnu utakmicu, neki od njih će uspeti prvi da dotrče do autobusa, a ostali neće uspeti da se popnu i moraće da sačekaju sledeći autobus. Kada svi stignu do igrališta, ekipa će ponovo biti na okupu.

Naši fmalni komentari se tiču dodatnih bajtova u SONET okvirima u SPE-ima. Svaki dodatni bajt je zadužen za neku akciju na jednom od tri sloja. Na primer, u dodatku odeljka nalaze se:

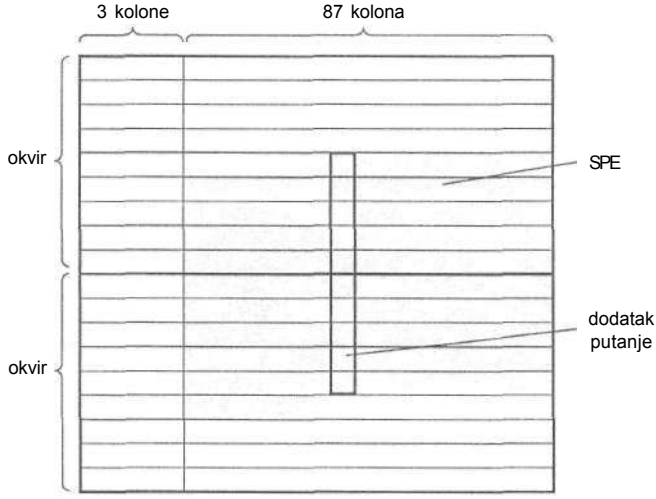
- ID kanala Ovo je posebno važna informacija ako se nekoliko STS-I okvira kombinuje u STS-n okvir. Bajtovi se razlikuju po ID-u kanala.
- Uzorak sinhronizacije Dva bajta iz dodatka sadrže uzorak za sinhronizaciju. Ovaj uzorak ukazuje na start okvira i omogućava prijemniku da se sinhronizuje sa dolazećim bitovima.



SLIKA 4.37 *Relacija između okvira i SPE bajtova*



(a) Okvir



(b) SPE koji se prostire preko granica okvira

SLIKA 4.38 *STS-I okviri i SPE*

Pošto se svaki okvir prenosi u pravilnim intervalima, prijemnik zna da bitovi sinhronizacije stižu u redovnim intervalima. Osim toga, ovo može da se koristi za detektovanje problema u sekciji. Ako nekoliko intervala prođe bez dolaska bitova sinhronizacije, onda sloj sekcije "izjavljuje" da je došlo do problema.

- **Bajt parnosti** U Poglavlju 6 detaljnije ćemo predstaviti proveru parnosti. U suštini, bajt pamosti utvrduje da li su neki bitovi oštećeni u toku prenosa duž sekcije. U stvari, on proverava da li je bilo grešaka u prethodnom okviru.
- **Orderwire bajt** Jedan bajt u svakom okviru (ili jedan bajt na svakih 125 *\xsec*) koristi se da bi 64 Kbps govorni komunikacioni kanal bio korišćen za održavanje aktivnosti. Naziva se i servisni kanal; tehničari i inženjeri mogu da ga koriste za

otklanjanje grešaka, ili za održavanje izmedu susednih uredaja na mreži, bez prekidanja ostalog saobraćaja.

- **Korisnički kanal** Obezbeđuje 64 Kbps kanal koji mogu da koriste lokalne aplikacije.
- **OAM komunikacija** OAM je skraćenica za operaciju, administraciju i održavanje (operation, administration, maintenance). Tri bajta po okviru obezbeduju 192 Kbps kanal koji se koristi za održavanje, praćenje i otklanjanje grešaka u komunikacijama na nivou sekcije, ponovo bez prekidanja ostalog saobraćaja. Pojednostavljeno gledano, ova tri bajta mogu da obezbede "glatko" funkcionisanje.

Dodatak linije sadrži sledeće:

- **Lokator SPE-a** Pošto se SPE može nalaziti bilo gde u okviru, postoje tri dodatna bajta koja lociraju bajt okvira u kome SPE počinje.
- **Bajt parnosti** Ova provera parnosti služi za za otkrivanje grešaka koje se mogu javiti na liniji.
- **Automatsko zaštitno komutiranje** Dva bajta se koriste za detektovanje eventualnih problema koji se mogu javiti u multiplekseru, a dizajnirani su za zaštitu svih konekcija unutar linije. Detalji su složeni, ali automatsko zaštitno komutiranje je dizajnirano da bi bila obezbedena tolerancija na otkaze; u slučaju otkaza linije, postoji rezervna linija koja može da preuzme njeno mesto.
- **OAM komunikacija** Ovo je u suštini isto kao i prethodno pomenuti OAM, ali na drugom sloju.
- **Ordervvire bajt** Ovo je, takođe, kao i prethodno pomenuti OAM, mada na drugom sloju.

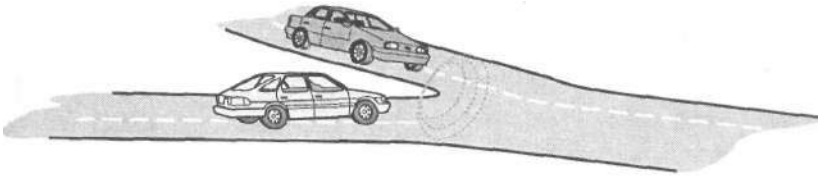
Dodatak putanje sadrži sledeće:

- **Trag STS putanje** Pošiljalac koristi jedan bajt u svakom okviru za neprestano slanje unapred definisane sekvence bajtova. Prijemnik poredi dolazeće bajtove sa poznatim uzorkom; ako dode do bilo kakve devijacije, zna da postoji problem sa konekcijom.
- **Bajt parnosti** Ponovo provera parnosti, ali za greške koje mogu da se jave duž putanje
- **Labela STS putanje signala** Sećate se da SONET može da posluži kao nosilac za mnoge druge tehnologije. Ovaj bajt identifikuje protokol na višem nivou i ukazuje na sadržaj SPE-a.
- **Status putanje** Omogućava enititetu na prijemnoj strani da označi status okončanja putanje za pošiljaoca (inicijator putanje).
- **Korisnički kanal** Ima istu funkciju kao i prethodni korisnički kanal, mada funkcioniše između elemenata putanje.
- **Indikator virtuelnog tributaryja** Tributary predstavlja signal iz više eksternih izvora (obično imaju brzinu manju od STS-I), a SONET omogućava uključivanje podataka iz više izvora u jedan SPE. Dakle, SPE sadrži informacije iz više izvora; svaki od njih je virtuelni tributary. Postoje različiti načini za struktuiranje SPE-a, koji zavise od izvora podataka. Ovaj indikator ukazuje na način koji je bio korišćen.

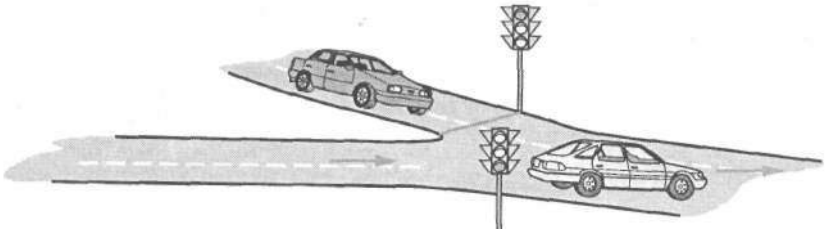
4.7 Protokoli nadmetanja za pristup zajedničkom medijumu

Multipleksiranje (posebno TDM) umnogome je omogudlo da se medijum stavi na raspolaganje većem broju korisnika, mada ne dovoljno. Rutiranje svih korisnika preko multipleksera na jedan medijum nije realno zbog dva razloga. Prvo, može da postoji isuviše veliki broj korisnika za jedan multiplekser. Drugo, logistika može da zabrani korisniku da koristi određeni multiplekser.

Lepa paralela može da se povuče sa primerom autoputa. Multiplekser obezbeduje pristup za više korisnika jednoj pristupnoj tački, ali u velikim mrežama je neophodno imati više pristupnih lačaka, kao što postoji više uključenja na autoput. Pristup medijumu iz više ulaznih tačaka naziva se nadmetanje (contention). Kontroliše se protokolom nadmetanja. Na slici 4.39 prikazano je šta se dešava kada nema protokola nadmetanja na autoputu. Vozila nasumice ulaze i povremeno dolazi do sudara. U najboljem slučaju dolazi do neprijatnih situacija; u najgorem su posledice fatalne. Srećom, autoputevi se ne postavljaju na ovakav način (mada sa nekim vozačima nikada ne možete da budete sigurni). Na slici 4.40 prikazana je uobičajena i jednostavna strategija nadmetanja u saobraćaju. Ovaj stop-and-go protokol koristi semafore za kontrolu saobraćaja. Ukoliko se vozači pridržavaju protokola, neće dolaziti do situacija poput one na slici 4.39. Pretpostavljamo da su Vam već poznati detalji ovog protokola, osim ako dolazite sa Himalaja.



SLIKA 4.39 *Nema protokola nadmetanja*

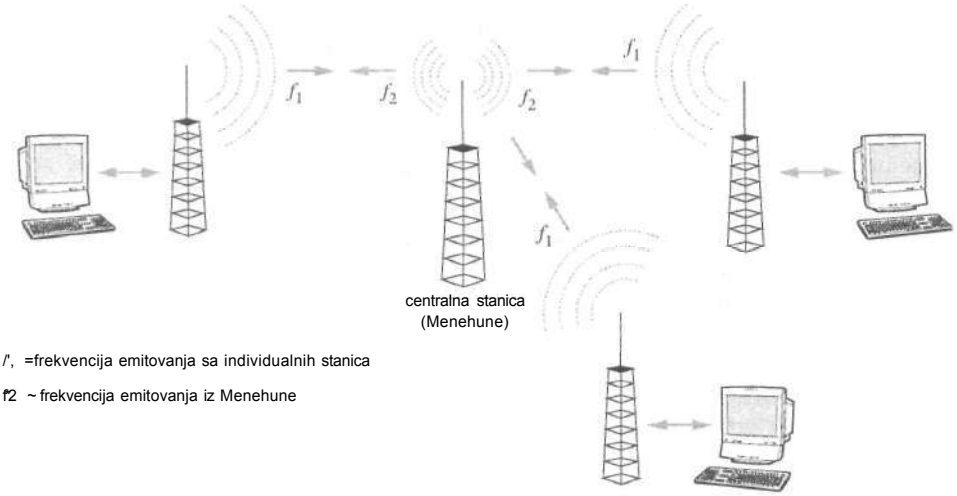


SLIKA 4.40 *Stop-and-Go protokol pristupa*

1 u komunikacijama su neophodni neki protokoli koji će obezbediti da poslati podaci stignu do svojih odredišta. U ovom odeljku izučavamo neke protokole koji se danas koriste. Verovatno Vas neće iznenaditi činjenica da postoji veliki broj različitih opcija, koje zavise od konkretnog medijuma, količine saobraćaja i sofisticiranosti korisničkih potreba (koji su u direktnoj vezi sa cenom).

Aloha protokol

Jedan od najstarijih protokola nadmetanja bio je razvijen u oblasti koja se veoma razlikuje od udaljenih područja Himalaja. Aloha protokol je razvijen početkom 70-ih prošlog veka na Universjty of Hawaji. Naziva se još i "čista" Aloha **(pure Aloha),** za razliku od drugog protokola koji ćemo uskoro predstaviti. Aloha sistem je bio dizajniran radi uspostavljanja komunikacije između ostrva korišćenjem paketnog radio sistema. Reči paket, ili okvir* tiču se informacija koje se prenose u toku jednog prenosa. Terminali (slika 4.41) su bili povezani na radio kanal, koji je emitovao informacije od lerminala do centralne stanice pod nazivom Menehune. Uredaji ernitu-ju okvire na istoj frekvenciji. Tako je medijum (vazdušni prostor) stvarno deljiv. Svaki pokušaj emitovanja dva različita okvira istovremeno korišćenjem iste frekvencije ometa oba signala.



SLIKA 4.41 *Aloha sistem*

* U zavisnosti od korišćenog protokola, za sadržaj jednog prenosa koriste se termini paket, ili okvir. Za sada ćemo nastaviti da koristimo lermin okvir.

Naravno, krajnji rezultat je da ni jedan prenos neće biti uspešan.

Aloha protokol je zasnovan na veoma jednostavnom principu. U suštini, svaki uredaj je mogao da inicira prenos u bilo kom trenutku. Ako bi došlo do kolizije dva signala, tu ništa nije moglo da se uradi. Uredaji su, jednostavno, čekali da istekne nasumice izabrani period i nakon toga su ponovo pokušavali da pošalju informacije. Ako se vratimo na naš primer autoputa, to bi bilo isto kao kada bi se vozači uključivali u saobraćaj zatvorenih očiju. Ako dode do sudara, to bi bilo isto kao da odete da kupite nova kola i nastavite putovanje kao da se ništa nije desilo. Iako bi ovo bio skup protokol za kontrolu saobraćaja, ipak je dobro funkcionisao kod Aloha sistema.

Kolizija se lako detektuje. Kada Menehune primi okvir, šalje potvrdu o prijemu. Polvrda se šalje na drugoj frekvenciji, tako da ne ometa dolazeće signale. Ako uređaj primi potvrdu, "zaključuje" da je okvir uspešno prenet. Ako ne primi potvrdu, "pretpostavlja" da se nešto desilo sa okvirom i čeka da ponovo pošalje okvir. Pošto svaki uredaj čeka nasumice izabrani period, šanse da dva, ili više uredaja čekaju da protekne isto vreme značajno su smanjene. Na taj način je umanjena i šansa da dode do druge kolizije. Ukoliko, ipak, dode do kolizije (možda sa nekim drugim uređajem), primenjuju se ista pravila: čeka se nasumice izabrani period i ponovo se pokušava slanje okvira.

U ovakvim situacijama do kolizije dolazi ne samo kada dva uredaja simultano šalju okvire, veći kada se dva prenosa preklapaju makar i za minimalnu količinu vremena. Uopšte nije bitno da li je uništen ceo okvir, ili samo jedan njegov deo. To bi moglo da se uporedi sa telefonskim razgovorom u kome čujete; "Dobio sam vesti za tebe. Imaš samo $%"#$%". Ako $%^{A}#$% predstavlja statičke smetnje, ne možete da znate da li ste dobili na lotou, ili ste dobili nasledstvo, ili ste kandidovani na izborima u stranci (dobra vest?). Ono što je izgubljeno ne može da se povrati, pa zdrav razum nalaže da se ceo okvir ponovo pošalje.

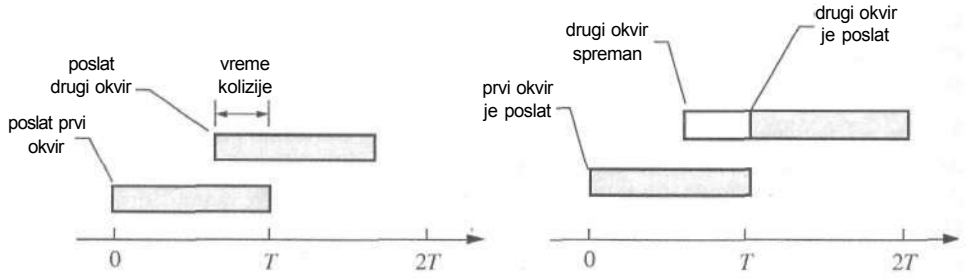
Prednost Aloha protokola je njegova jednostavnost. Funkcioniše veoma dobro ako ne postoji veliki broj prenosa, ali ako uredaj često emituje informacije, ili ako postoji veći broj uredaja, on je manje efikasan. U svakom slučaju, javiče se više kolizija, baš kao što bi se desilo na opterećenim putevima.

Kada se suočimo sa većim brojem prenosa, šta možemo da uradimo da se smanji broj kolizija? Najpre ćemo pažljivije analizirati zašto one naslaju. Kao što smo ranije istakli, do njih dolazi kada se delovi dva prenosa preklapaju. Pretpostavimo da je T vreme neophodno za jedan prenos i da dva uređaja moraju da prenose informacije. Ukupno vreme potrebno da oba uredaja uspešno obave prenos je *TT*.

Zatim, razmotrimo proizvoljni interval dužine *27*. Ukoliko jedan uredaj ne počne svoj prenos na samom početku ovog intervala, nemoguće je obaviti oba prenosa do isteka intervala. Zato se u slučaju da se uređajima omogući startovanje prenosa u proizvoljnom trenutku najviše može "protraćiti" *TT* vremena.

Eventualno, možemo da pretpostavimo da se vreme deli na intervale (slotove) od po T jedinica za svaki uredaj i da je neophodno da uređaji započnu prenos na početku slota. Drugim rečima, ako je uredaj spreman za slanje na sredini slota, mora da čeka dok ne počne sledeći slot (slika 4.42b). Na ovaj način je moguće da do kolizije dode samo ako su dva uredaja spremna za slanje na početku slota. Ovo se razlikuje od prethodnog scenarija (slika 4.42a), kod koga se kolizija javlja ako drugi uredaj inicira prenos kada je okvir spreman.

Varijanta kod koje se uređaju dopušta iniciranje prenosa samo na početku vremenskog slota naziva se slotovani Aloha protokol. Ovo deluje kao bolja varijanta od "čistog" Aloha protokola. U stvari, rigorozna analiza oba protokola pokazuje da se slotovani Aloha protokol bolje izvršava.



(a) Prenos korišćenjem "čistog" Aloha protokola (b) Prenos korišćenjem slotovanog Aloha protokola

SLIKA 4.42 *Prenos okvira kada se koriste "čisti" Aloha protokol i slotovani Aloha protokol*

Ovde nećemo navoditi detalje analize, ali ćemo dati pregled rezultata i interpretirati ih. Detaljniju diskusiju možete da pročitate u referencama [Wa98], [StOO], [Ro75], [Ru89] i [Mi87],

Intuitivno, znamo da postoji relacija izmedu broja poslatih okvira i broja uspešno poslatih okvira. Moguće je kreirati matematički modem, koji, uz određene pretpostavke, definiše relacije izmedu njih.

Neka G predstavlja saobraćaj izražen kao prosečni broj okvira koji se generišu po slotu, a neka S bude učestalost uspešnosti, predstavljena prosečnim brojem uspešno poslatih okvira po slotu *(e* je matematička konstanta 2.718...). Relacija izmedu G i S i za "čisti" i za slotovani Aloha protokol je

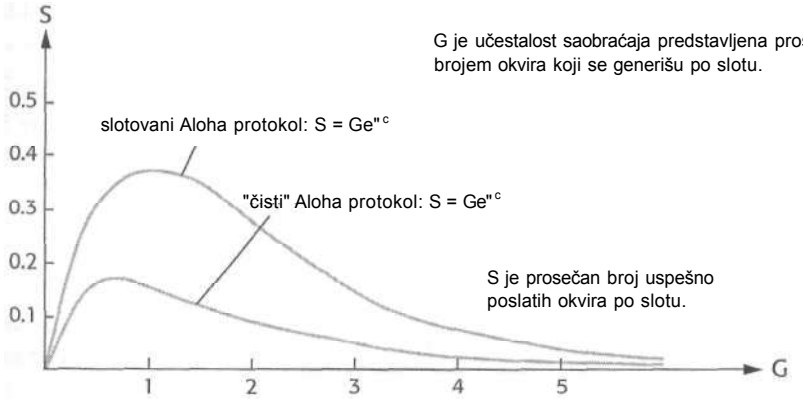
$S = Ge^{-2G}$ ("čisti" Aloha protokol)

$S = Ce^{-G}$ (slotovani Aloha protokol)

Vaše sledeće logično pitanje je verovatno šta to, u stvari, znači. Da biste dobili odgovor, pogledajte grafike na slici 4.43. Vertikalna osa predstavlja S_I a horizontalna G. Vrednosti za *S* se nalaze u opsegu izmedu O i 1. Pošto smo izabrali da je vremenski slot jednak vremenu potrebnom za slanje jednog okvira, S ne može da bude veće od toga.

Najpre napomenimo da oba grafika imaju isti osnovni oblik. Ako je G malo, malo je i S. Ovo ima smisla, zato što će uspešno biti poslat manji broj okvira ako se generiše samo nekoliko okvira. Kako se G povećava, tako se povećava i S, ali do određene tačke. Veći broj prenosa znači veći broj uspešnih okvira, sve do trenutka kada počnu kolizije. U toj tački, koja odgovara najvišoj tački na oba grafika, učestalost uspešnosti počinje da opada. Dok G nastavi da raste, S teži 0. Ovo odgovara situaciji u kojoj postoji toliko okvira da su kolizije skoro neizbežne. Okvir retko uspeva da stigne do svog odredišta bez kolizija.

Model pokazuje poredenje izmedu "čistog" i slotovanog Aloha protokola. Diferencijalni račun pokazuje kako se izračunava maksimalna vrednost za svaki slučaj. Ako su Vam poznati detalji, jednostavno nadite izvod svake funkcije po G i izjednačite ga sa nulom. U svakom slučaju, maksimum za slotovani Aloha protokol dešava se u G = 1, za koje je S = 1/e = 0.368.



SLIKA 4.43 *Učestalost uspešnosti za slotovani i "čisti" Aloha protokol*

Drugim rečima, najbolja vrednost uspešnog slanja je aproksimativno 0,368 okvira po slotu. Drugi način da se ovo kaže je da se u 37 odsto vremena okviri uspešno prenose. U ostalo vreme dolazi do kolizija, ili uopšte nema prenosa.

Kod "čistog" Aloha protokola maksimum se javlja u G = 0.5, za koje je S = l/2e s 0.184. Ovo znači da se oko 18 odsto vremena utroši na uspešan prenos okvira.

Na prvi pogled možda deluje čudno što slotovani Aloha protokol daje bolje rezultate. Dopuštanje prenosa okvira na svaka tri slota deluje kao nedovoljno iskorišćenje medijuma. Medutim, kritične pretpostavke modela omogućavaju spremnost okvira u nasumično definisanim trenucima. Nema nikakve namere da se koordinira prenos pored toga što se mora sačekati početak narednog slota. Kao što model pokazuje, ova značajna pretpostavka degradira performanse protokola.

Sledeće pitanje može da bude da li treba pokušati koordinaciju prenosa da bi se postigla bolja efikasnost. U opštem slučaju, odgovor je negativan. Ako se nalazite za personalnim kompjuterom koji je povezan na mrežu, verovatno ćete otkriti da je nezgodno koordinisati pristup mreži sa ostalim korisnicima. Većina korisnika želi pristup mreži odmah po izdavanju zahteva i očekuje dobar odziv od protokola i hardvera.

Protokol Carrier Sense Multiple Access (CSMA)

Pronicljiviji čitalac može da postavi pitanje zar ne bi mogla da se poboljša učestalost uspešnosti ako bi uređaji "osluškivali" da li je na medijumu* u toku prenos pre nego što iniciraju slanje svog okvira. Uređaji sigurno imaju mogućnost utvrdivanja da li je u toku prenos okvira.

* Iako je Aloha sistem bio razvijen za paketni radio, mi koristimo termin medijum u opštijem smislu. Ukoliko nije drugačije naglašeno, to mogu da budu vazdušni prostor, fiber, kabl, ili upredene parice.

Zašto se okvir ne bi zadržavao sve dok se ne završi postojeći prenos? Tako uređaji ne bi uništavali okvir koji se trenutno prenosi i bila bi poboljšana učestalost uspešnog prenosa.

Ideja "osluškivanja" ima smisla i danas se koristi u mnogim mrežama, kao što je Ethernet (on koristi varijaciju koju ćemo opisati nešto kasnije). Ovaj pristup je označen kao **Carrier Sense Multiple** Access (CSMA). U opštem slučaju, protokol se jednostavno opisuje. Ako uređaj ima okvir za slanje, sledi naredne korake:

1. "Osluškuje" da li na medijumu postoji neka aktivnost.
2. Alra nema aktivnosti, inicira prenos; u suprotnom, čeka.

Da li ovakav sistem eliminiše kolizije? Eliminisaće ih u odredenoj meri, ali ne u potpunosti. Kolizije i dalje mogu da se jave ako dva (ili više) uredaja "žele" da iniciraju prenos skoro istovremeno. Ako trenutno nema aktivnosti, oba "zaključuju" da je bezbedno inicirati prenos i to i rade. Naravno, tada dolazi do kolizije. Međutim, takve kolizije su u opštem slučaju mnogo ređe, jer uvek postoji manje kašnjenje između trenutka kada uredaj detektuje da nema aktivnosti i trenutka kada okvir koji on prenosi stigne do drugih uređaja. Da bi došlo do kolizije, drugi uređaj mora da odluči da inicira prenos u tom periodu. Pošto je taj interval izuzetno kratak, mala je verovatnoća da će drugi uredaj inicirati prenos okvira u tom intervalu.

Medutim, kolizije su i dalje moguće i postoje varijacije CSMA koje pokušavaju da ostvare bolju efikasnost redukovanjem broja kolizija. Sa jednim tipom, **p-perzistentnim CSMA,** uredaj nastavlja da "motri" aktivni medijum. Kada se utiša, uredaj inicira prenos sa verovatnoćom pfO $<p < I)$. U suprotnom, čeka da protekne jedan vremenski slot (verovatnoća od 1 - $p)$. Ako je p = 1, uredaj uvek inicira prenos kada na medijumu prestane aktivnost. Ako je $p = 0$, uvek čeka. Kod **neperzistentnog CSMA** uredaj ne nastavlja da prati medijum. Jednostavno, čeka jedan vremenski slot i ponovo proverava postojanje aktivnosti. U ovoj tački inicira prenos ako na medijumu nema aktivnosti; ako ih ima, čeka na još jedan vremenski slot.

Kolizije i dalje mogu da predstavljaju problem, posebno kod p-perzistentnog CSMA. Ako dva uredaja "žele" da iniciraju prenos skoro istovremeno, a medijum je neaktivan, dolazi do kolizije. Ovako izazvane kolizije možda nisu česte, a moguće su kolizije i zbog drugih razloga. Na primer, razmotrimo slučaj u kome je $p = 1$. Ako dva (ili više) uredaja postaju spremna dok neki treći uređaj prenosi okvir, oni čekaju. Ali, pošto je $p = I_1$ oba uređaja iniciraju prenos čim se medijum oslobodi, tako da dolazi do kolizije njihovih okvira. Cim postoji veći broj uređaja, ovaj scenario se češće ponavlja i kolizije postaju ozbiljniji problem.

Jedan od načina za smanjivanje učestalosti kolizija je smanjivanje verovatnoće za iniciranje prenosa odmah nakon što prethodni uređaj završi prenos. Na primer, pretpostavimo da na prenos čekaju dva uredaja koja koriste 0.5-perzistentni protokol. Kada je medijum neaktivan, svaki inicira prenos sa verovatnoćom 0.5. Tako se četiri događaja dešavaju sa jednakom verovatnoćom:

- Oba uredaja odmah iniciraju prenos.
- Oba uredaja čekaju.
- Prvi uredaj šalje okvir, a drugi čeka.
- Drugi uredaj šalje okvir, a prvi čeka.

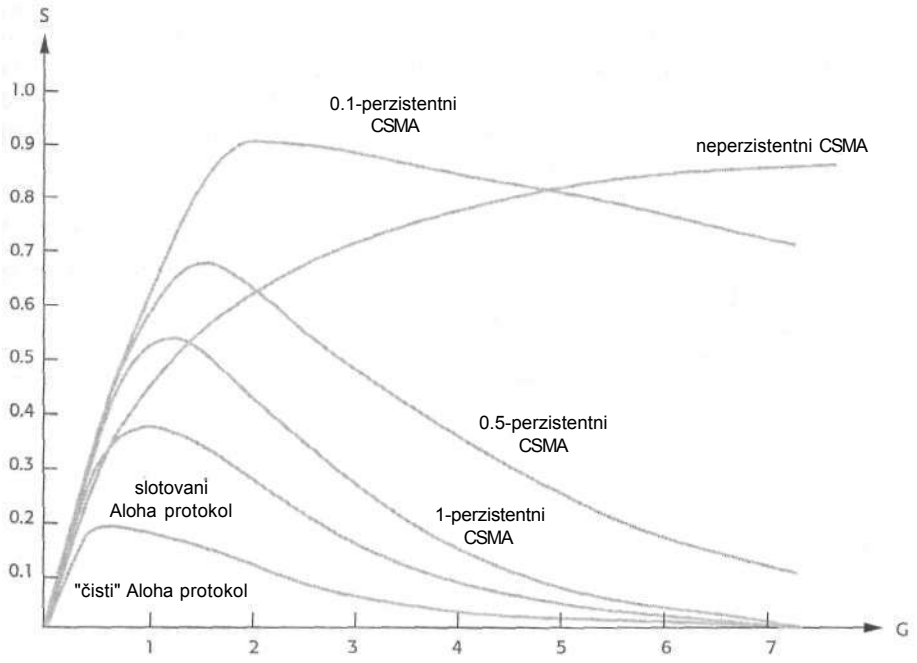
Umesto izvesne kolizije, postoji verovatnoća 0.5 da ni jedan uredaj neće inicirati prenos na početku sledećeg slota. Ipak, napomenimo da postoji i verovatnoća 0.25 da ni jedan uredaj neće inicirati prenos. Ovo je sledeći tip neefikasnosti, koji ćemo uskoro predstaviti.

Na slici 4.44 prikazana je učestalost uspešnosti različitih p-perzistentnih protokola. Ovde nećemo analizirati jednačine pomoću kojih su ove krive generisane, jer su prilično složene. Ipak, razmotrićemo rezultate i objasniti njihovo značenje. Ako Vas interesuju konkretne jednačine, ili njihovi izvodi, pogledajte referencu [K175].

U opštem slučaju, manje vrednosti za *p* daju manje kolizija. Medutim, tako se povećava vreme čekanja uređaja, pa, samim tim, dolazi do proporcionalnog povećanja verovatnoće da niko neće inicirati prenos. I pored toga, kako se broj uredaja povećava, saobraćaj raste. Povećava se verovatnoća da če bar dva uređaja istovremeno pokušati da iniciraju prenos i kolizije mogu ponovo da predstavljaju problem.

Kod neperzistentnog CSMA ni jedan uredaj ne čeka da medijum bude neaktivan. Umesto toga, periodično proverava stanje medijuma i čeka jedan vremenski slot ako je medijum zauzet. Tako će doći do kolizija samo ako dva uređaja detektuju prestanak aktivnosti i skoro istovremeno iniciraju prenos. Kao što je prikazano na slici 4.44, učestalost uspešnosti se povećava sa G; kod većih vrednosti za G dobijaju se mnogo bolji rezultati, nego kod perzistentnog, ili Aloha protokola.

Učestalost uspešnosti je samo statistički pokazatelj i često može pogrešno da se protumači. Onako kako smo je definisali ona je impresivna, ali ne može da se postigne sve dok ne postoji ogroman broj okvira koje uređaji treba da pošalju. Na primer, jednačine modela pokazuju da se učestalost uspešnosti od 0.9 postiže kada je G = 9. Drugim rečima, 90 odsto vremena se troši na slanje okvira samo ako se generišu brzinom od devet okvira po slotu. Ovo je brzina koja premašuje kapacitet medijuma sa faktorom 9.



SLIKA 4.44 *Učestalost uspešnosti za CSMA i Aloha protokole*

U ovom slučaju imamo druge probleme koji premašuju one koji nastaju zbog protokola. Uređaji su zatrpali medijum i tako izazvali kašnjenja.

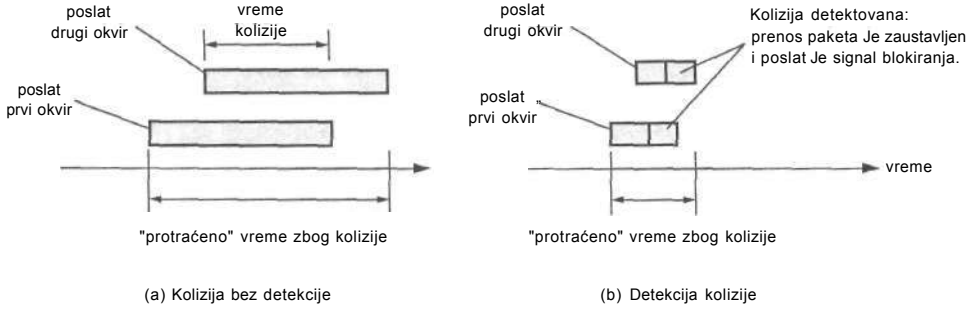
Ovde može da se povuče dobra paralela sa primerom banke koja angažuje samo jednu osobu na šalteru u danima sa najvećim brojem stranaka. Stranke čekaju i po 45 minuta da bi položile depozit i žale se menadžeru, koji odgovara da je šalterski službenik sve vreme zauzet i da zato nema nikavih problema. Tada shvatate da je najbolje da pokupite svoj novac i sakrijete ga u dušek.

Kod manjih vrednosti za G perzistentni protokol ima veću učestalost uspešnosti, jer je period neaktivnosti veći problem nego kod neperzistentnih protokola. Ako uredaj detektuje da je prenos u toku, čeka da istekne kompletan slot. Ako se prenos dobro završi pre tog vremena, medijum je suviše neaktivan. U slučajevima slabijeg saobraćaja, neperzistentni protokol nije dovoljno agresivan.

Detekcija kolizije

Sledeći način da se poboljša učestalost uspešnosti prenosa je da se redukuje vreme u kome može da dode do kolizije. Ranije, kada je uređaj imao okvir, slao ga je u celini i dolazio do "zaključka" da se desila kolizija ako nije dobio potvrdu o prijemu. Problem je to što drugi uredaji nisu mogli da koriste medijum u vreme kada je došlo do kolizije okvira. Da li postoji neki način da uredaj nadgleda medijum da bi osluškivao kolizije? Ako postoji, da li može odmah da prekine prenos i tako smanji vreme u kome traje kolizija?

Na slici 4.45 je data ilustracija. Na slici 4.45a vreme kolizije se proteže od prenosa prvog do kraja prenosa drugog okvira. "Protraćeno" vreme može da se produži najviše na dva vremenska slota. Na slici 4.45b oba uredaja prekidaju prenos čim dođe do kolizije. Tipično, svaki uredaj šalje signal blokiranja (jamming signal - vrsta električnog upozorenja) da bi se osiguralo da svi uređaji budu obavešteni da je došlo do kolizije. U ovorn slučaju "protraćeno" vreme iznosi samo jedan deo slota i plus vreme koje je bilo potrebno za slanje signala blokiranja.



SLIKA 4.45 *Kolizija sa i bez detekcije*

Postoji ovakav protokol i obično se koristi zajedno sa CSMA. Nosi naziv **Carrier Sense Multiple Access with Collision Detection (CSMA/CD). On** se, obično, koristi sa nekim perzistentnim algoritmima. Opšti pregled bi izgledao ovako:

- Ako je medijum zauzet, uredaj čeka u skladu sa perzistentnim algoritmom.
- Ako nema aktivnosti na medijurnu, uredaj prenosi okvir i nastavlja "osluškivanje".
- Ako uredaj detektuje koliziju, odmah prekida slanje i šalje kratki signal blokiranja.
- Nakon kolizije, uredaj čeka da istekne nasumično izabrani period pre nego što ponovo pokuša da pošalje okvir.

Poslednji korak je bitan radi redukovanja šansi da i drugi put dođe do kolizije dva okvira. Ne postoji nikakva izvesnost da će dva uredaja čekati da protekne identičan vremenski interval dok ne iniciraju ponovni prenos.

Dva problema koja izbijaju na površinu kod rasprave o detekciji kolizije su veličina okvira i udaljenost koju okviri prelaze. Ako je okvir isuviše veliki, jedan uredaj može da uzme monopol nad medijumom. Sa druge strane, detekcija kolizije zahteva da okviri imaju minimalnu veličinu da bi kolizija bila detektovana pre nego što se završi slanje okvira. Ako se kolizija detektuje nakon što se okvir pošalje, ne zna se da li je okvir bio uključen u koliziju - on je mogao da stigne do svog odredišta, a da kolizija nastupi izmedu neka druga dva okvira. Možda ste pomislili da bismo mogli iskoristiti prethodno opisani metod koji traži potvrdu o prijemu. Mogli bismo, ali **u** tom slučaju bi bio eliminisan razlog zbog koga se detekcija kolizije koristi - da bi se izbeglo slanje celog okvira pre nego što dode do kolizije.

Do koje mere treba smanjiti okvir? Odgovor zavisi od maksimalnog vremena koje je potrebno za detektovanje kolizije. Ponekad se kolizija detektuje skoro trenutno. U drugim slučajevima signal može da "putuje" preko velikih udaljenosti samo da bi se sreo sa nekim drugim signalom. Čak i tada šum koji nastaje zbog kolizije može da se vrati do uredaja koji je poslao okvir. U najgorem slučaju, vreme potrebno za detekciju je dva puta duže od vremena koje je signalu potrebno da prede veliku udaljenost na kojoj se medijum prostire.

Primera radi, pretpostavimo sledeće:

- Uredaj šalje okvire preko koaksijalnog kabla brzinom od IO Mbps.
- Najveće rastojanje izmedu dva uredaja na kablu iznosi 2 km.
- Signal se prostire duž kabla brzinom od 200 m/usec (metara u mikrosekundi).

U najgorem slučaju, okvir "putuje" 2 km (što traje 10 usec) pre nego što dode do kolizije sa nekim drugim okvirom. Nakon toga, oštećeni signal "putuje" 2 km nazad do pošiljaoca. Ukupan put traje 20 usec. Tako bi okviru trebalo najviše 20 usec za slanje. Brzina prenosa podataka od 10 Mbps je ista kao da se šalje 10 bitova po usec; zbog toga, uredaj može da postavi 200 bitova za 20 trsec. Ovo bi značilo da okvir treba da ima najmanje 200 bitova, ili 200/8 = 25 bajtova. U Poglavlju 9 predstavićemo specifične mreže i protokole i tada ćete videti da oni zaista odreduju minimalnu veličinu okvira.

Drugi problem kod detekcije kolizije je rastojanje. Na primer, "osluškivanje" i detekcija kolizije ne funkcionišu dobro kod satelitskih mreža, kod kojih vreme potrebno za prenos okvira od

zemaljske stanice do satelita i vraćanje nazad iznosi, aproksimativno, jednu četvrtinu sekunde. U poređenju sa svetom u kome se sve meri mikrosekundama, reč je o dugačkom periodu. Ako zemaljska stanica osluškuje satelitski prenos, može da detektuje samo ono što je poslato pre jedne četvrtine sekunde. Tako Je moguće da ne čuje ništa kada nečiji okvir ide ka satelitu na kolizionom kursu sa nečim što je većposlato.

Velika prednost do sada predstavljenih protokola je što protokol ne mora da se menja kada se doda novi uredaj. Uredaji ne moraju da imaju posebna znanja o ostalim uređajima na mreži. Ova mogućnost dodavanja novih uredaja bez promene protokola olakšava proširenje mreže. Medutim, značajan nedostatak je što dolazi do kolizija. Raspravljali smo o tome da li protokol može sasvim dobro da funkcioniše, iako ne postoji teorijsko ograničenje u broju kolizija do kojih može doći. Uvek postoji mala šansa da će doći do neuobičajenih kašnjenja. Kod većine aplikacija periodična kašnjenja nisu mnogo ozbiljna, ali mogu da budu katastrofalna u real-time aplikacijama, kao što su hemijske i nuklearne elektrane, sistemi za kontrolu vazdušnog saobraćaja, ili procesi za automatizaciju proizvodnje.

U nekim slučajevima vreme čekanja nakon kolizije nije u potpunosti nasumično izabrano. Jedna uobičajena tehnika defmiše vreme čekanja kao integral umnoška vremenskog slota. Broj vremenskih slotova mora da bude ograničen, tako da uredaj ne čeka preterano dugo.

Definisanje granice nije jednostavno. Na primer, ako je velika, nasumično izabrani periodi čekanja mogu da budu dugački i da, zbog toga, postoje preterano dugački periodi neaktivnosti medijuma. Sa druge strane, veći broj vremenskih slotova iz kojih se bira smanjuje šansu da dva, ili više uredaja izaberu isti slot i da ponovo dode do kolizije. Ako je granica relativno mala, uredaji između čijih okvira dolazi do kolizije ne čekaju isuviše dugo; ipak, sa raanjim brojem raspoloživih slotova, povećava se šansa da će ponovo doći do kolizije.

Tehnika poznata pod nazivom binarni eksponencijalni backoff algoritam utvrduje različite granice. Funkcioniše na sledeći način:

- Ako dode do kolizije okvira nekog uredaja prvi put, on čeka 0, ili I vremenski slot (izabrano nasumično) pre nego što ponovo pokuša slanje.
- Ako dode do kolizije drugi put, čeka 0, 1,2, ili 3 slota (ponovo izabrano nasumično).
- Nakon treće kolizije, čeka između 0 i 7 Vremenskih slotova.
- U opštem slučaju, nakon n kolizija, čeka da protekne između 0 i 2" - 1 slot ako je $n < 10$. Ako Je $n > 10$, čeka između 0 i 1024 (2^{10}) slotova.
- Nakon 16 kolizija, odustaje. Verovatno negde postoji neka greška i do administratora mreže se šalje obaveštenje o nemogućnosti prenosa. U ovom slučaju drugi softver, ili menadžer mreže moraju da ispitaju u čemu je problem.

Ovaj pristup jasno pokušava da minimizuje preterana čekanja održavanjem broja mogućih vremenskih slotova na nižoj vrednosti. Na kraju krajeva, ako dolazi do kolizije izmedu okvira dva uredaja, postoji verovatnoća od 50 odsto da će oni uspeti da pošalju okvire u narednom pokušaju (uz pretpostavku da drugi uredaji ne pokušavaju da pošalju svoje okvire).

Ipak, u mnogim slučajevima kolizije veoma su male šanse da će bilo koji uredaj uspeti da pošalje okvir u sledećem pokušaju. Da bi uredaj bio uspešan, on treba da izabere ili 0, ili I vremenski slot, a svi ostali uredaji moraju da izvrše drugačiji izbor. Povećavanjem broja vremenskih slotova

nakon svake kolizije šanse za ponovnu koliziju se ponovo smanjuju eksponencijalno. Moguće je imati veliki broj vremenskih slotova samo ako su svi prethodni pokušaji bili neuspešni. U takvim situacijama dugo čekanje može da bude jedino rešenje.

Izbegavanje kolizije

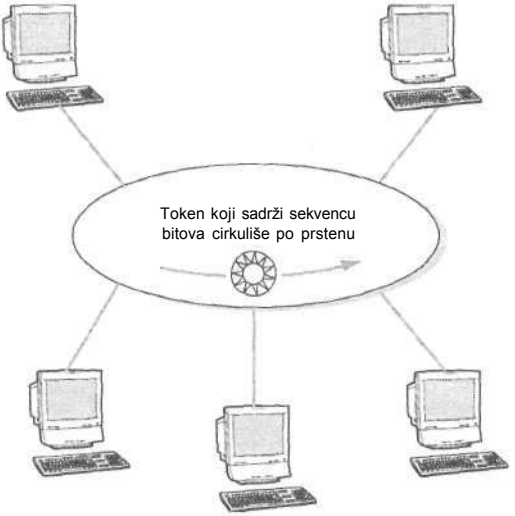
U bežičnim LAN okruženjima detektovanje kolizije nije uvek raspoloživa opcija. Na primer, pretpostavimo da uredaji A i B istovremeno šalju okvir do uredaja C. Moguče je da ni A, ni B ne vide signal onog drugog. Ako se prenos vrši preko infracrvenih talasa, čvrsta prepreka može da omete medusobno registrovanje signala. Ako se prenos vrši preko radio talasa male snage, oni mogu da budu veoma udaljeni (iako je C u dometu oba uredaja). U takvim situacijama CSMA/CD nije opcija.

Protokol 802.11 za bežične LAN mreže usvaja šemu **izbegavanja kolizije.** Bez obzira na naziv, protokol ne izbegava sve kolizije; ipak, značajno se redukuje njihov broj. Reč je komplikovanom protokolu, tako da ćemo njegovo detaljnije predstavljanje odložiti do Poglavlja 9.

Prosledivanje tokena

Prethodno opisani protokoli su imali pomalo "anarhistički" pristup slanju signala, jer su dopuštali Liredajima da pošalju ono što su hteli. Logično pitanje koje se ovde nameće je da li je moguće da se neki uredaji unapred "dogovore" koji kada šalje svoje podalke. Drugim rečima, da li postoji mogućnost za kruženja?

Poleškoća koja se javlja kod mnogih mreža je što nema centralne kontrole, niti autoriteta koji bi donosio takve odluke. Ipak, postoji način da svi uređaji učestvuju u donošenju odluka na nivou protokola za kruženje: prosledivanje tokena je uobičajeni protokol koji se koristi u token ring mrežama. U token ring mreži (slika 4.46) uredaji su obično personalni kompjuteri, povezani kružno pomoću običnog kabla, ili fibera. Mrežna interfejs kartica **(NIC - netvvork interface card)** povezuje svaki uredaj na mrežu i sadrži hardver i logiku koja personalnom kompjuteru omogućava komuniciranje sa mrežom.



SLIKA 4.46 *Token ring mreža*

Ovde korišćeni protokol nadmetanja za pristup medijumu može da se uporedi sa prethodno predstavljenim protokolom "šalji kad možeš". Uređaji u token ringu izvršavaju protokol, omogućavajud nazimenično slanje okvira. Proces uključuje spedjalno formatirani okvir, nazvan token. On sadrži kodove koje NIC prepoznaje i kruži kroz prsten, posedijući svaku NIC karticu. U skladu sa protokolom, svaki PC može da šalje svoje podatke kada njegova NIC kartica ima token.*

U Poglavlju 9 ćemo objasniti kako tačno uredaji razmenjuju okvire. U opštem slučaju koncept je sledeći. Kada uredaj primi token, moguće su dve solucije. Ako nema ništa za slanje, jednostavno prosleduje token do svog suseda. Ako ima nešto za slanje, umeće svoje podatke i adresu odredišta u token. Osim toga, menja neke kontrolne bitove da bi token bio identifikovan kao okvir sa podacima (okvir u kome se prenose informacije). Nastavljajući na ovakav način, token može da poseti svaki uredaj i da pokupi podatke kada budu raspoloživi. Ovo je slično elektronskom Federal Expressu koji putuje unaokolo, sakupljajud i ispuštajući okvire.

Kada uredaj primi token, on najpre ispituje kontrolne bitove. Ako oni ukazuju da nema podataka u okviru, uredaj postupa u skladu sa tim da li ima nešto za slanje. Ako u okviru već postoje podaci, uredaj ispituje odredišnu adresu. Ako odredišna adresa ukazuje na neki drugi uredaj, okvir se, jednostavno, rutira do susednog uređaja. U suprotnom, uredaj kopira informadje iz okvira. Nakon toga, ponovo postavlja okvir u prsten. Eventualno, okvir se vraća do pošiljaoca, koji uklanja podatke i postavlja token, ili drugi okvir nazad na prsten. Nastavljajud na ovaj način, podaci eventualno stižu do svog odredišta.

Token ring operadje mogu da se prošire da uključuju prioritete i sistem rezervadja koji će uredaju omogudti rezervisanje tokena koji već ima neaje podatke radi budućeg korišćenja. Ove operadje predstavićemo u odeljku 9.6.

Prsteni imaju još jednu prednost slično protokolima koje smo ranije predstavili: novi uredaji se lako dodaju. Kada uredaj šalje okvir do svog suseda, on ne zna da li je taj sused dodat u skorije vreme, ili je tu oduvek. Zbog toga, uredaji ne moraju da budu obavešteni o promenama kod suseda.

Nedostatak je što uredaj može da "zarobi" token. Osim toga, prekid linka između dva sukcesivna uredaja može da obori mrežu, jer token ne može da kruži. Drugi problem se javlja ukoliko uredaj otkaže dok šalje token na prsten. Rezultat je nekompletan i neispravan token koji nastavlja da kruži po prstenu. Uredaji koji čekaju na prijem ispravnog tokena nastavljaju da čekaju na nešto čega nema u prstenu. Sledeđ problem se javlja ako, zbog nekog razloga, otkaže uredaj koji je odgovoran za oslobadanje tokena. Svaki uredaj prosleduje token do svog suseda i okvir beskonačno kruži. I ovi problemi imaju rešenja; neka od njih predstavićemo u Poglavlju 9.

* U nekim situacijama ovo nije tačno. U Poglavlju 9 možete da pročitate detaljniju analizu o token ring protokolu.

Tabela 4.5: Pregled protokola nadmetanja za pristup zajedničkom medijumu

Protokol	Značai	Prednosti	Nedostaci
Aloha	vanjacija je konsna u satelitskim komunikacijama, gde "osluškivanje" nije praktično zbog velikog kašnjenja.	Jednostavan pnstup	Potencijalna kasnjenja, jer uredaj možda ne zna da je došlo do kolizije okvira sve dok ga ne pošalje celog.
p-perzistentni	U kombinaciji sa CSMA/CD predstavlja popularan izbor kod Ethemeta.	Nastoji da smanji šanse za neaktivnost medijuma.	Moguć je preveliki broj kolizija u slučaju većeg saobraćaja, tako da se "traći" opseg medijuma.
Neperzistentni	Prema modelu, učestalost uspešnosti se ne smanjuje sa većim opterećenjima.	Redukuje se broj kolizija, posebno pod većim opterećenjem.	Prenosni medijum može da bude neaktivan kada postoje uredaji koji žele da pošalju neke podatke.
CSMA/CD	U kombinaciji sa p-perzistentnim protokolom, predstavlja čestu varijantu za Ethernet.	Redukuje vreme kolizije.	Nema teorijske gornje granice za vreme koje je potrebno za uspešno.
Token ring	Nekada populami protokol u kancelarijskim i poslovnim okruženjima	Gornja vremenska granica za uredaj odredena je čekanjem na dobijanje tokena.	Ukoliko u protokol nisu ugradeni mehanizmi oporavka, otkazi na jednom uređaju mogu da unište token, ili da raskinu prsten, tako da se uticaj prenosi na celu mrežu.

Rezime protokola

U ovom odeljku smo predstavili najčešće korišćene protokole nadmetanja za pristup zajedničkom medijumu. Njihov pregled je dat u tabeli 4.5. Postoje i mnogi drugi protokoli. Ako ste zainteresovani, pogledajte reference [StOO] i [Ta03].

4.8 Zaključak

U ovom poglavlju smo se najviše bavili uredajima, sistemima, protokolima i tehnologijama koje imaju kritični značaj za uspostavljanje konekcija. Prosta mogućnost prenosa informacija u obliku niza bitova nije dovoljna ako ne postoji način da više uredaja medusobno saraduje, pristupa i deli medijum za prenos. U suprotnom, uredaj prenosi bitove koji završavaju u nedogled. Drugim rečima, prenos je ostvaren, ali ne i komunikacija. Moramo da obezbedimo da preneti bitovi stižu do svojih odredišta i da sistemi koji prenose te signale funkcionišu u skladu sa specifičnim protokolima. U ovom poglavlju su predstavljeni sledeći važni koncepti:

- Među komunikacione nosioce i uredaje uključuju se telefonski sistem, PBX (privatne telefonske centrale), mobilni telefoni i faks mašine. Telefonski sistem povezuje najveći broj ljudi. Godinama je bio korišćen prvenstveno za govorne komunikacije, ali iz godine u godinu povećavao se broj ljudi koji su ga koristili za razmenu podataka. U stvari, neke organizacije su instalirale svoj privatni sistem pod nazivom PBX (privatne centrale) za prenos i glasa i podataka. Za interne komunikacije u okvim jedne kompanije pomoću više privatnih centrala zaobilazi se korišćenje usluga javnih telefonskih centrala, ali su, i pored toga, neophodne međugradske linije.

- Mobilni telefoni su korisnicima omogućili slobodu fizičkog kretanja prilikom uspostavljanja konekcija. Oblast se deli na regione, ili ćelije, koje imaju sopstvene predajnike, koji mogu da komuniciraju sa telefonskim sistemom. Nakon toga, mobilni telefon komunicira sa predajnikom ćelije u kojoj se trenutno nalazi.

- Faks mašina kombinuje tehnologije kopiranja i prenosa informacija. Poput kopir mašine, reprodukuje sliku na papiru. Medutim, slika se reprodukuje u elektronskom obliku i prenosi preko telefonskog sistema. Faks na drugom kraju prima signal i ponovo kreira originalnu sliku.

- Postoje različiti modovi komunikacije. Kod paralelnog prenosa bitovi se šalju istovremeno korišćenjem nekoliko linija, a kod serijskog prenosa šalju se u sekvenci preko samo jedne linije. Simplex komunikacije su isključivo jednosmerne. Half-duplex komunikacije omogućavaju dvosmerne komunikacije, ali se slanje i prijem podataka vtše naiz,meničao. Full-duplex komunikacije omogućavaju istovremeno slanje informacija iz oba uredaja.

- Siedeći bitan faktor u koimmikatijama 'je te'jm'vng. Ksvrv Viram pienos %a\\e svak\ Y>a\t zasebno, sa start i stop bitom pre i nakon bajta. Tako bajtovi stižu sa nasumičnim razmacima između njih. Kod sinhronog prenosa bajtovi se grupišu u format okvira, a zatim se šalje ceo okvir. I pored toga, podaci u sukcesivnim okvirima mogu da stižu naglo jedni za drugim, sa velikom količinom podataka, a zatim da slede dugački periodi čekanja do sledećeg niza podataka. Izohroni prenos garantuje da će podaci stizati konzistentnom brzinom sve vreme. Ovo je najprikladniji prenos za real-time aplikacije, kao što je prikazivanje slika odmah čim se generišu, ili slušanje muzike dok se izvodi.

- EIA-232 protokol povezuje DTE i DCE; verovatno je najpoznatiji protokol interfejsa. Određuje defmicije kola i pravila koja oba uredaja moraju da poštuju da bi uspešno komunicirali. X.21 interfejs je dizajniran za digitalne interfejse. U poredenju sa EIA-232 standardima, koristi manje linija između DTE-a i DCE-a, ali zahteva bolju logiku za interpretiranje razmenjenih signala.

- Univerzalna serijska magistrala je dizajnirana da bi se pojednostavilo povezivanje perifernih uređaja na personalni kompjuter. Uređaji mogu da se priključuju bez prethodnog isključivanja sistema sa izvora napajanja. Korišćenjem hubova korisnik može da poveže do 127 uređaja na personalni kompjuter. USB uredaji komuniciraju korišćenjem serijskih komunikacija i funkcionišu u master/slave modu, koji diktira da komunikaciju koordinira host (obično PC). Kod tipičnog pristupa host "proziva" uređaje, nalažući im da iniciraju prenos ako imaju podatke za slanje, ili upozorava

uređaj da šalje podatke. USB podržava teretni (bulk), kontrolni, prekidni i izohroni transfer radi povezivanja različitih tipova uredaja.

- Kao i USB, FireWire je dizajniran radi uprošćavanja povezivanja perifernih uredaja. Za razliku od USB-a, dizajniran je pre za uredaje koji podržavaju multimedijalne aplikacije i zahtevaju veće bitske brzine. Postoji nekoliko značajnih razlika izmedu USB-a i FireWirea. FireWire koristi šestožilni kabl i strob signal koji prijemnik koristi za sinhronizaciju sa dolazećim bitovima. Osim toga, FireWire je peer-to-peer protokol. Uredaji mogu da komuniciraju nezavisno od hosta pomoću mehanizma arbitraže koji je ugraden u FireWire protokol.

- Mnogim aplikacijama nije neophodna puna snaga high-speed rnreže, ili velika cena predstavlja ograničenje za kreiranje zasebnih konekcija. Multiplekser može da posluži kao interfejs izmedu nekoliko uredaja i jedne mrežne konekcije. Multiplekseri sa podelom frekvencije kombinuju analogne signale iz različitih kanala u jedan analogni signal. Multiplekseri sa podelom vremena postavljaju nizove bitova iz različitih izvora u jedan okvir fiksne dužine. Statistički multiplekseri mogu da kombinuju nizove bitova Li jedan okvir, ali mogu da koriste okvire promenljive veličine, u zavisnosti od izvora koji šalje podatke. Multipleksiranje sa podelom talasnih dužina kombinuje optičke signale različitib frekvencija u jednom zajedničkom fiberu, a zatim se na drugom kraju signal rastavlja na komponente, slično načinu na koji prizma prelama zrak na svetlost razlićitih boja.

- Dva uobičajena digitalna nosioca su Tl i SONET. Tl koristi multipleksiranje sa podelom vremena za kombinovanje 8-bitnih grupa iz razlidtih izvora u jedan zajednički DS-I okvir. Obično impulsni kodni modulator u svakom izvoru generiše 8.000 8-bitnih grupa u sekundi da bi bila postignuta bitska brzina koja odgovara telefonskim konverzacijama. SONET je složeniji noseći servis koji se koristi u komunikacijama na velikim razdaljinama. SONET prenosi 810-bajtne STS-I okvire na svakih 125 Lisec, ili 8.000 okvira u sekundi. Ovim se definiše bazna signalna brzina od 54,84 Mbps. Veće brzine postiže STS-192m, što odgovara brzini od 9,953 Gbps. Tada je u jednom, ili više okvira moguće prenositi SPE (synchronous payload envelope) format podataka. SONET koristi regeneratore za regeneraciju većine bitova u okviru, add/drop multipleksere za spajanje eksternih podataka sa SONET okvirima, bez multipleksiranja celog sadržaja okvira, i STS multipleksere za izvršavanje kompletnog multipleksiranja eksternih signala u STS okvire.

- Pošto mreža treba da opslužuje veći broj korisnika, mora da im omogući međusobnu komunikaciju. To znači da je neophodno donošenje odluke o tome kada dva, ili više uredaja "nameravaju" da istovremeno pošalju svoje podatke. Jedan pristup (Aloha protokol) dizajniran je za paketne radio komunikacije na Havajskim ostrvima. Ako se dva okvira preklapaju, dolazi do kolizije. Kada pošiljalac ne dobije potvrdu o prijemu okvira, ponovo inicira slanje istog okvira. Slotovani Aloha protokol je sličan, ali zahteva slanje podataka samo na početku unapred definisanih vremenskih slotova.

- CSMA ide korak dalje od Aloha protokola, tako što uredaji "osluškuju" medijum pre nego što iniciraju slanje okvira.

Okviri se šalju samo ako nema saobraćaja na medijumu. Ako saobraćaj postoji, sledeđ korak zavisi od varijadje protokola koja se koristi. Kod p-perzistentnog CSMA protokola uredaj nastavlja da nadgleda medijum - kada na medijumu prestanu aktivnosti, on inicira slanje okvira sa verovatnoćom p. Kod neperzistentnog CSMA protokola uređaj ne prati stanje medijuma. Jednostavno, čeka nasumično izabrani broj vremenskih slotova i ponovo pokušava da pošalje okvir. Poslednja varijacija CSMA/CD koristi tehnike na osnovu kojih se prekida slanje ako uredaj detektuje koliziju. Namera je da se smanji količina vremena u kome može doći do kolizije okvira.

- Sledeći protokol je prosledivanje tokena, koji se koristi u token ring mrežama. Specijalni okvir, nazvan token, kruži kroz mrežu između uređaja. Uređaj može da šalje podatke sarno ako ima token. Token ring uredaji su organizovani u fizički prsten.

Pitanja i zadaci za proveru

1. Navedite pet glavnih komponenata telefonskog sistema.
2. Opišite način funkcionisanja mobilne telefonije.
3. Sta su privatne centrale?
4. U čemu je razlika izmedu roaminga i prenošenja (handoff) u kontekstu mobilne telefonije?
5. Navedite razlike između serijskih i paralelnih komunikacija.
6. Navedite razlike izmedu sinhronih, asinhronih i izohronih komunikacija.
7. Navedite tipična polja u okviru podataka i šta ona sadrže.
8. Navedite razlike izmedu simplex, half-duplex i full-duplex komunikacija.
9. Da li su sledeća tvrdenja tačna, ili netačna (i zašto)?
   a. Paralelne i serijske komunikacije zahtevaju različite vrste kablova.
   b. EIA-232 interfejs zahteva 25-pinski konektor.
   c. Dva kompatibilna PC-ja mogu da komuniciraju ako se instalira kabl između njihovih EIA-232 portova.
   d. Uređaji koji koriste EIA-232 interfejs mogu automatski da komuniciraju.
   e. USB uredaji ne mogu da komuniciraju nezavisno jedni od drugih.
   f. Iako većina personalnih kompjutera ima svega par USB portova, moguće je povezati više od 100 USB-kompatibilnih uređaja.
   g. Pojedine aplikacije mogu da tolerišu gubitak nekih podataka zbog grešaka koje nastaju u toku prenosa.
   h. FireWire kabl ima dva para upredenih parica za paralelni prenos dva niza bitova.
   i. FireWire-kompatibilni uređaji ne zahtevaju host PC za koordinaciju komunikacije.
   j. Multipleksiranje sa podelom frekvencije predstavlja oblik paralelnih komunikacija.
   k. Multipleksiranje sa podelom vremena je rezervisano samo za digitalne komunikacije.

I. SONET okviri se prenose samo kada ima podataka.

m. Multiplekser sa podelom vremena omogućava kombinovanje ulaznih kapaciteta tako da se premaši izlazni kapacitet.

n. Neperzistentni protokoli nadmetanja za pristup zajedničkom medijumu bolji su od perzistentnih protokola u svim slučajevima.

o. 1-perzistentni protoko] je optimalan u grupi p-perzistentnih protokola, jer uredaj nikada ne čeka dobrovoljno, tako da se ne "traći" vreme.

10. Navedite razliku između DTE-a i DCE-a.
11. Sta je null modem?
12. Zašto je na slici 4.15 pin 20 svakog DTE-a priključen na pin 8 drugog DTE-a?
13. Zašto je na slici 4.15 pin 4 povezan na pin 5 istog DTE-a?
14. Navedite razliku izmedu peer-to-peer i master/slave protokola.
15. Navedite različite tipove transfera koje podržava USB i opišite svaki od njih.
16. Navedite razliku između USB okvira i paketa.
17. Zašto USB paketi sa podacima ne sadrže adresu?
18. Sta je prozivka i zašto je neophodna?
19. Sta je daisy chain? U čemu se razlikuje od povezivanja uredaja na zajedničku magistralu?
20. Sta je strob signal? Koje su njegove prednosti, a koji su njegovi nedostaci?
21. Koja dva tipa komunikacija podržava FireWire?
22. Navedite razliku izmedu multipleksiranja sa podelom frekvencije i podelom vremena.
23. Šta je multiplekser?
24. Šta su zaštitni opsezi?
25. Šta je kanal u kontekstu multipleksiranja sa podelom vremena?
26. Navedite razliku između nosećeg, modulišućeg i modulisanog signala.
27. Sta je primarna motivacija za korišćenje multipleksera?
28. Zašto polje DS-I okvira ima osam bitova za svaki kanal?
29. Navedite razliku između add/drop multipleksera i STS multipleksera.
30. Zašto SONET ne garantuje da će se sve korisne informacije naći u okviru?
31. Navedite razliku između sekcije, linije i putanje u SONET nosećem sistemu.
32. Sta Je protokol nadmetanja za pristup zajedničkom medijumu?
33. Opišite Aloha protokol, zajedno sa listom njegovih prednosti i nedostataka.
34. Navedite razliku između slotovanog i "čistog" Aloha protokola.
35. U čemu je razlika između O-perzistentnog CSMA i neperzistentnog CSMA protokola?
36. Zašto detekcija kolizije poboljšava performanse CSMA?
37. Zašto se performanse perzistentnog protokola nadmetanja degradiraju kako se G povećava, dok kod neperzistentnog protokola važi suprotno pravilo?
38. Šta je binarni eksponencijalni backoff algoritam?
39. Šta je token?

Vežbe

1. Zašto asinhrone komunikacije zahtevaju dodatne start i stop bilove? U čemu bi bila pogreška kada bi se uzelo da je prvi bit podataka, ujedno, i start bit, a poslednji stop bit?
2. Pošto se kod paralelnih komunikacija bitovi prenose istovremeno, zašto se paralelne komunikacije ne dizajniraju sa proizvoljno velikim brojem paralelnih linija radi skraćivanja vremena prenosa?
3. Vode se rasprave da li čak i sinhrone komunikacije imaju asinhronu komponentu. Kako je to moguće?
4. Koliki je minimalni broj kola neophodan za uspostavljanje full-duplex komunikacija preko EIA-232 interfejsa?
5. Zašto DTE mora da potvrdi RTS (Recjuest to Send) kolo pre slanja do DCE, a DCE ne mora da potvrđuje ni jednu RTS liniju pre slanja do DTE-a?
6. Neki null modemi povezuju pin 20 sa pinom 5, ili 6 na DTE-u. Koja je svrha takvog povezivanja?
7. Neki null moderni povezuju pin 4 DTE-a sa sopstvenim pinom 5 i pinom 8 drugog DTE-a. Čemu to služi?
8. Ako imate personalni kompjuter, ili pristup uređaju koji ima EIA-232 interfejs, proverite u uputstvu koja su kola korišćena.
9. Napišite program za generisanje grafika modulišućeg i modulisanog signala, sličnih onima sa slike 4.28.
10. Pretpostavimo da možete da kucate 100 reči u minutu i da je prosečna dužina reči šest karaktera (uključujući blanko znake). Koji je minimalni broj USB okvira u sekundi za koje možete da očekujete da će ih USB-kompatibilna tastatura "pozvati"?
11. Pretpostavimo da FireWire šalje bitove 01100011000 preko FireWire kabla. Skicirajte oba signala koja odgovaraju i bitovima podataka i strob signalu.
12. Pretpostavimo da je pet uređaja povezano na statistički multiplekser sa podelom Vremena (slično situaciji predslavljenoj na slici 4.31) i da svaki od njih stvara izlaz koji je ovde prikazan. Konstruišite okvir koji multiplekser šalje.

Device 1 : 0	A3	0	A2	A1
Device 2: B4	B3	0	B2	B1
Device 3: 0	C2	0	0	C1
Device 4: D5	D4	D3	D2	D1
Device 5: 0	0	E2	0	E1

13. Koja je svrha dodavanja 0.5 na sinusnu funkciju za formiranje f(t) sa slike 4.28? Odnosno, šta se dešava ako se taj član eliminiše?
14. Analizirajte značenje grafika sa slike 4.43.
15. Komentarišite sledeću konstataciju:

Sa 1-perzistentnim CSMA protokolom uredaj koji čeka uvek prenosi okvir ako nema aktivnosti na medijumu. Zašto nc promeniti protokol tako da, kada je medijtim slobodan, uredaj čeka odredeni period u toku koga može da se kompletira prenos nekog drugog uredaja? Ako Je medijum i dalje slobodan, inicira prenos. Ovo bi trebalo da umanji verovatnoću da dode do kolizije dva uredaja koji čekaju na prenos.

16. Komentarišite korisnost O-perzistentnog CSMA protokola.
17. Pretpostavimo da tri uredaja koja koriste 0.5-perzistentni protokol čekaju na neaktivan medijum.
    a. Kolika je verovatnoća da će doći do kolizije kada se medijum oslobodi?
    b. Kolika Je verovatnoća uspešnog prenosa kada se medijum oslobodi?
    c. Kolika je verovatnoća da ni jedan uređaj neće slati ništa kada se medijum oslobodi?
18. Ponovite vežbu 17, ali pretpostavite da uredaji koriste 0.25-perzistentni protokol.
19. Pretpostavite da su dva uređaja koja koriste CSMA/CD i binarni eksponencijalni algoritam poslali okvire izmedu kojih je došlo do kolizije.
    a. Kolika je verovatnoća da će ponovo doći do kolizije za vreme sledećeg slota?
    b. Kolika Je verovatnoća da će oba uređaja uspešno preneti okvire za vreme sledeća dva slota?
    c. Kolika Je verovatnoća da će doći do kolizije: još dva, ili još tri puta?
20. Pretpostavite da su tri uredaja koja koriste CSMA/CD i binarni eksponencijalni algoritam poslala okvire izmedu kojih je došlo do kolizije.
    a. Kolika je verovatnoća da će ponovo doći do kolizije za vreme sledećeg slota?
    b. Kolika je verovatnoća da će sva tri uredaja uspešno preneti okvire za vreme sledeća tri slota?
    c. Kolika je verovatnoća da će doći do kolizije izmedu bilo koja dva uredaja za vreme sledećeg slota?
21. Pretpostavite da je binarni eksponencijalni backoff algoritam promenjen tako da uređaj uvek čeka 0, ili 1 vremenski slot, bez obzira na broj nastalih kolizija. Kako ovo menja efikasnost?
22. Pretpostavite da Je n neki pozitivan ceo broj, a da je binarni eksponencijalni backoff algoritam promenjen tako da uredaj uvek čeka izmedu 0 i 2n - 1 vremenski slot, bez obzira na broj nastalih kolizija, Kako to utiče na efikasnost?
23. Za svaku od narednih primena navedite najprikladniji mod prenosa (asinhroni, sinhroni, ili izohroni).
    a. Preuzimanje fajla
    b. Povezivanje štampača
    c. Prikazivanje slika sa Web kamere
    d. Video konferencija
    e. Preuzimanje multimedijalnog fajla
    f. Slušanje Web radija
    g. Povezivanje tastature
    h. Korišćenje softvera za iniciranje telefonskog poziva
    i. Praćenje predsedničkih debata uživo

Reference

[An99] Anderson, D. *FireWire System Architecture:* IEEE 1394A, 2nd ed. Reading, MA: Addison-Wesley, 1999.

[AxOl] Axelson, J. *USB Complete,* 2nd ed. Madison, WI: Lakeview Research, 2001.

[Ga02] Goralski, W. *SONET/SDH,* 3rd ed. New York: McGraw-Hill, 2002.

[K175] Kleinrock, L., and F. Tobagi. "Random Access Techniques for Data Transmission over Packet-Switched Radio Channels". *AFIPS Conference Proceedings,* vol. 44 (1975), 187.

[LeOO] Leon-Garcia, A. and I. Widjaja. *Communication Networks.* New York: McGraw-Hill, 2000.

[Ma72] Martin, J. *Systems Analysis for Data Transmission.* Englewood Cliffs, NJ: Prentice-Hall, 1972.

[Mi87] Mitrani, I. *Modeling of Computer and Communications Systems.* London: Cambridge University Press, 1987.

[Ro75] Roberts, L. "ALOHA Packet System with and Without Slots and Capture". *Computer Communications Review,* vol. 5 (April 1975), 28-42.

[Ru89] Russel, D. *The Principles of Computer Networking.* New York: Cambridge University Press, 1989.

[Sh90] Sherman, K. *Data Communications:* AUser Guide. 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1990.

[StOO] Stallings, W. Data and Computer Communications. 6th ed. Englewood Cliffs, NJ: Prentice-Hall, 2000.

[St03] Stallings, W. *Computer Organization and Architecture.* 6th ed. Englewood Cliffs, NJ: Prentice-Hall, 2003.

[St96] Stanley, W. *Network Analysis ivith Applications.* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 1996.

[Ta03] Tanenbaum, A. S. Computer Networks. 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2003.

[Wa98] Walrand, J. *Communications Networks: A* First Course, 2nd ed. Boston: Richard D. Irwin, 1991.

[WaOO] Walrand, J. and P. Varaiya. *High-Performance Communication Networks,* 2nd ed. San Francisco: Morgan Kaufmann, 1996.

POGLAVLJE 5

Kompresija podataka

Kompjuter ne misli, ne oseća ništa. Nema ni memoriju, samo prostor za skladištenje. Kad god mu se obratite, ili dobijate kompletan odgovor, ili ništa. Neodređenost, inteligentna konfuzija, originalno baratanje rečima, ili idejama se nikada ne dešava, a inteme veze su uvek nepromenljive; uspostavljene su jednom od istinskih umova koji su kreirali mašinu i program. Kada se uključi, i najjednostamiji kompjuter sigumo može da obavi posao u skladu sa svojim maksimalnim mogućnostima; najveam umovima ne mogu da se povere takvi poslovi zbog jednog jednostavnog razloga - zbog njihove promenljivosti, koja ih čini toliko superiarnim.

—**Jacques Barzun** (1907-), američki predavač i autor

5.1 Uvod

Da li ste nekada slušali MP3 audio fajl? Da li ste nekada preuzeli filmski triler snimljen u formi MPG, ili MPEG fajla? Da li ste nekada koristili faks (faksimil) mašinu? Sasvim sigurno, ako čitate ovu knjigu, imate neka tehnička predznanja i izveli ste neke od ovih akcija. Medutim, ništa od ovoga ne bi bilo praktično bez sofisticirane matematike i rutina za kompresovanje. Sa tolikim brojem novih aplikacija koje zahtevaju elektronske komunikacije očigledan trend je kreiranje bržih i jeftinijih načina za slanje podataka. U Poglavlju 2 smo pomenuli neke od tekućih razvojnih oblasti, kao što su optički fiber, visokofrekventni mikrotalasi i UTP višeg stepena.

Neke aplikacije ipak ne mogu da čekaju na nov razvoj. Njihovi zahtevi su primorali ljude da potraže nove načine za postizanje brzih i jeftinih komunikacija. Na primer, razmotrite faks mašinu, jedan od najznačajnijih izuma iz 80-ih prošlog veka. U Poglavlju 4 opisano je kako ona deli sadržaj parčeta papira u tačke, u zavisnosti od slike na papiru. Tipična faks mašina koristi 40.000 tačaka po kvadratnom inču, što daje skoro četiri miliona tačaka po stranici. Ako se koristi 56 Kbps modem, potrebno je više od jednog minuta da se prenesu ove informacije. Ako ste ikada koristili faks mašinu, znate da to, ipak, ne traje toliko dugo.

Sledeći primer je video. Ono što mi vidimo kao kretanje je standardni TV ekran, koji, u stvari, prikazuje 30 slika (kadrova) u sekundi (isti princip na kome se zasnivaju pokretne slike).

Osim toga, svaka slika sadrži aproksimativno 20.000 tačaka, ili **piksela** - svaki piksel ima različitu jačinu osnovnih boja plave, zelene i crvene. Različite kombinacije omogućavaju generisanje različitih boja spektra. Ako se svaka osnovna boja jednog piksela predstavlja sa osam bitova, onda je za svaku sliku potrebno 20.000 24 = 4.800.000 bitova. Dvočasovni film bi zahtevao oko 216.000 zasebnih slika, ili 216.000 4.800.000 = 1.0368 1012 bitova, mnogo više od kapaciteta jednog DVD-a. Ipak, na DVD staje dvočasovni film.

Oba primera pokazuju da mogu da se zaobidu fizička ograničenja medijuma. Kako? Odgovor je pomoću **kompresije podataka,** načina da se obavi redukovanje broja bitova u okviru, a da se, pri tom, zadrži značenje. Na taj način se smanjuje cena i skraćuje vreme prenosa. Kompresija podataka se koristi u različitim oblastima, kao što su faks mašine, DVD tehnologija i V.42 modemi. Koristi se i prilikom smeštanja na disk, a mnogi proizvođači softvera kompresuju svoje programe na diskovima i CD-ima radi uštede prostora.

Sledeće logično pitanje je kako eliminisati bitove, a, pri tom, zadržati neophodne informacije. Na primer, pretpostavite da imate podatke u fajlu koji se u potpunosti sastoji od niza velikih slova. Ako pošaljete fajl e-mailom, koliko će bitova biti preneto? Ako se karakteri smeštaju kao 8-bitni ASCII kodovi, broj prenetih bitova je *8n,* gde je n ukupan broj karaktera. Medutim, informacije koje se šalju sadrže samo velika slova, tako da nije potreban kompletan 8-bitni ASCII kod. Možete li da izvedete kod koji predstavlja samo velika slova? Da! U tabeli 5.1 prikazan je 5-bitni kod nastao pomoću brojeva izmedu 0 i 25 (u binarnom obliku). Koristeđ ovu tabelu, aplikacija može da zameni svaki 5-bitni kod originalnim 8-bitnim kodom. Prijemna aplikacija može da ih konvertuje nazad. Rezultat je to da se informacije šalju i da je broj prenetih bitova 5n - redukcija od 37,5 odsto.

Nameću se brojna pitanja. Šta učiniti ako postoje kontrolni karakteri? Šta je sa malim slovima? Šta učiniti ako se u podacima ne nalaze samo slova? Ovo su validna pitanja i na njih moramo da nademo odgovore. U tabeli 5.1 predstavljen je veoma jednostavan metod kompresije, koji nije prikladan za mnoge aplikacije. Njegova glavna svrha je da pokaže šta je kompresija i šta može da se postigne pomoću nje.

Tabela 5.1: Alternativni kodovi za velika slova

Slovo	Kod
A	00000
B	00001
C	00010
D	00011
.	.
.	.
.	.
X	10111
Y	11000
Z	11001

Postoje brojni načini za kompresovanje podataka. Medutim, bez obzira na korišćeni metod, bitno je da se utvrdi da li postoji redundansa u originalnim podacima; ako postoji, treba je eliminisati. Na primer, u primeru ASCII koda za slova od A do Z redundantnost postoji, jer su prva tri bita svake kodne kombinacije ista. Redundansa nalaže eliminisanje tih bitova i korišćenje kodova iz tabele 5.1. Medutim, kao što ćete videti, postoji više tipova redundanse.

5.2 Frekventno zavisni kodovi

ASCII kod i kod iz tabele 5.1 imaju nešto zajedničko. Svi karakteri koriste isti broj bitova. Medutirn, šta ako analiza teksta pokaže da se odredeni karakteri češće javljaju? Ovo nije ništa neuobičajeno, jer se neka slova stvarno koriste češće od drugih (faktor učestalosti je razlog zašto ona manje vrede u igrama kao što je Skrebl). Pregledajte nekoliko prethodnih pasusa i primetićete da se slova e, s, ili t javljaju češće nego *z, x, i\[j.* Da li bi imalo smisla menjati dužinu koda tako da češće korišćeni karakteri odgovaraju kraćim kodovima - da imaju manje bitova? Ovakav kod se naziva i frekventno zavisni kod; za njegov prenos je potreban manji broj bitova.

Hafmanov kod

Primer frekventno zavisnog koda je Hafmanov (Huffman) kod (ref. [hu52]). Na primer, pretpostavimo da Je u tabeli 5.2 prikazana učestalost karaktera (procenat vremena koliko se javljaju) u fajlu. Da biste lakše kontrolisali primer, pretpostavite da imate samo pet karaktera. Ako želite, možete da izvedete sličan primer sa svih 26 slova.

U tabeli 5.3 prikazan je Hafmanov kod za te karaktere. Primećujete da kažemo Hafmanov kod, jer, kao što ćete pokazati, nije jedinstven. Uskoro ćemo pokazati kako se razvija.

Zatim, pretpostavimo da je niz bitova 01110001110110110111 bio kodiran Hafmanovim kodom. Ako Je najpre prenet krajnji levi bit, kako se interpretira? Kodovi fiksne dužine imaju prednost. U okviru prenosa uvek znamo gde se jedan karakter završava i gde sledeći počinje.

Tabela 5.2: Učestalost pojavljivanja slova od A do E

Slovo	Učestalost (%)
A	25
B	15
C	10
D	20
E	30

Tabela 5.3: Hafmanov kod za slova od A do E

Sovo	Učestalost (%)
A	01
B	110
C	111
D	10
E	00

Na primer, kod prenosa ASCII-kodiranih karaktera svaki skup od po osam bitova definiše novi karakter. Ovo nije slučaj kod Hafmanovog koda, pa kako onda da se interpretira niz bitova kodiran Hafmanovim kodom? Kako znamo gde se jedno slovo završava, a gde sledeće počinje?

Odgovor leži u svojstvu Hafmanovih kodova koje je poznato kao no-prefix svojstvo - kod jednog karaktera nikada ne može da se koristi kao prefiks drugog koda. Na primer, Hafmanov kod za A je 01, tako da ni jedan kod ne počinje sa 01.

Na slici 5.1 prikazano je kako se interpretira string kodiran Hafmanovim kodom. Dok se bitovi primaju, uređaj kreira podstring nadovezivanjem. Zaustavlja se kada podstring odgovara podstringu 01, što znači da je A prvi karakter koji je poslat. Da bi bio naden sledeći karakter, odbacuje se tekući string i počinje kreiranje novog sa prijemom svakog se kada podstring odgovara kodiranom karakteru. sledećeg bita. U ovom slučaju sledeća tri bita (110) odgovaraju karakteru *B.* Napomenimo da string ne odgovara ni jednom Hafmanovom kodu sve dok se ne prime tri bita. Ovo je posledica no-prefix svojstva. Uređaj nastavlja u ovom "maniru" sve dok se ne prime svi bitovi. Podaci sa slike 5.1 sadrže string ABECADBC.

" Sledeći koraci pokazuju kako se kreira Hafmanov kod.

1. Svakom karakteru se dodeljuje binarno stablo koje se sastoji samo iz jednog čvora. Svakom stablu je dodeljena učestalost pojavljivanja karaktera, koju nazivamo težina stabla (tree's *weight).*

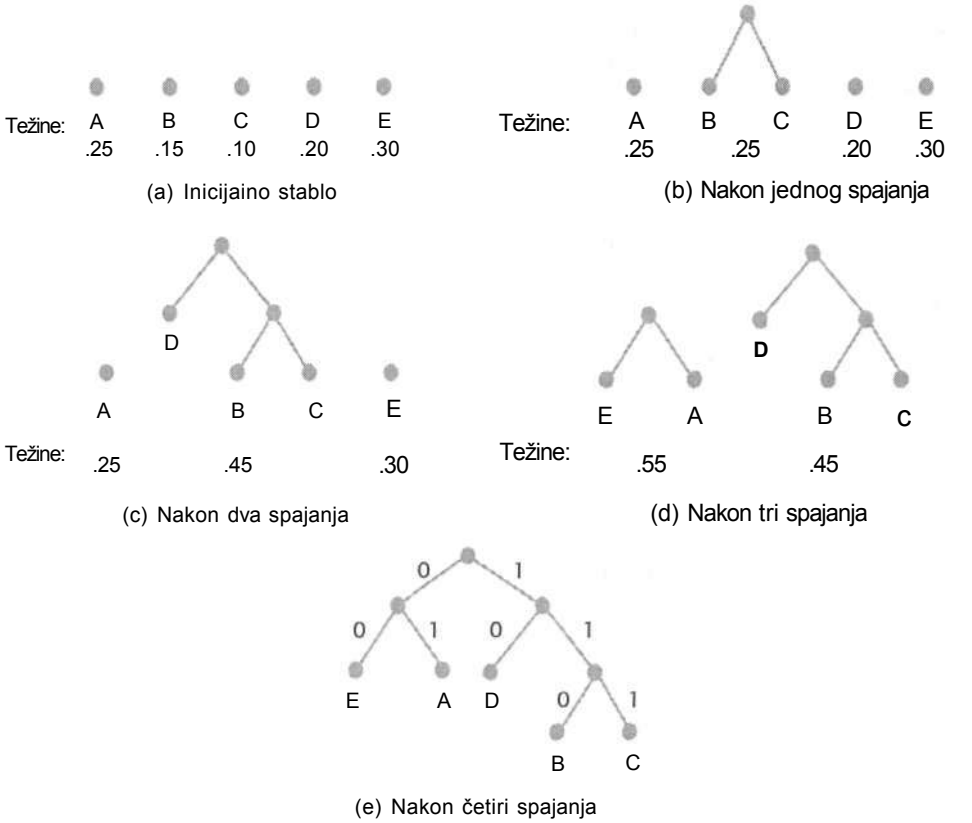
2. Potražite dva najlakša stabla. Ako ih ima više od dva, izaberite bilo koja dva. Spojite ih u jedno sa novim korenom, čije levo i desno podstablo odgovara ranijim stablima. Dodelite sumu težina spojenih stabala kao sumu novog stabla.

3. Ponovite sledeći korak sve dok ne budete imali samo jedno stablo.

SLIKA 5.1 *Prijem i interpretiranje poruke kodirane Hafmanovim kodom*

Kada se stablo kompletira, svi originalni čvorovi predstavljaju listove u fmalnom binarnom stablu. Kao i kod bilo kog binarnog stabla, postoji jedinstvena putanja od korena do lista. Za svaki list putanja do njega definiše Hafmanov kod. Putanja se određuje tako što se dodeljuje 0 svaki put kada se sledi levi pokazivač i 1 za svaki desni pokazivač.

Na slici 5.2 (delovi od a do e) prikazana je konstrukcija Hafmanovog koda u tabeli 5.3, a na slici 5.2a prikazano je jedno stablo sa pet samostalnih čvorova sa njihovim težinama. Stabla za slova B i C imaju manje težine, tako da se njihovim spajanjem dobija rezultat sa slike 5.2b. Kod dmgog spajanja postoje dve mogudhosti: spajanje novog stabla sa *D,* ili spajanje A sa D. U ovom slučaju proizvoljno biramo prvo spajanje; rezultat je prikazan na slici 5.2c. Nastavljajući na ovaj način, eventualno dobijate stablo kao na slici 5.2e. Kod njega možete da vidite da su svakom levom i desnom pokazivaću dodeljene O i 1 . Praćenjem pokazivača do čvora lista dobija se Hafmanov kod za pridruženi karakter. Na primer, praćenjem levog (0) pokazivača, pa desnog (1), dolazimo do lista čvora A. Ovo je konzistentno sa Hafmanovim kodom 01 za slovo A.



SLIKA 5.2 *Spajanje Hafmanovih stabala*

Aritmetička kompresija

Sledeći primer metoda za kompresovanje podataka koji generiše frekventno zavisni kod je **aritmetička kompresija,** koja je zasnovana na interpretiranju stringa kao jednog realnog broja. Ovo nije neuobičajen koncept; na primer, mogli bismo da interpretiramo Hafmanovim kodom kodirane nule i jedinice u stringu kao bitsku reprezentaciju za veoma veliki ceo broj. Više karaktera u stringu znači više bitova i, samim, tim veći broj.

Aritmetička kompresija funkcioniše tako što se defmiše asocijacija između stringa karaktera i realnog broja između 0 i 1. Pošto postoji beskonaćno mnogo brojeva izmedu 0 i 1 i beskonačno mnogo stringova,* možemo da definišemo algoritam koji pridružuje realni broj bilo kom stringu.

Ilustracije radi, koristićemo ista slova i učestalost koji su bili korišćeni u prethodnom primeru za Hafmanov kod. Razlika je u tome što dodeljujemo opseg brojeva (podinterval intervala [0, 1]) na osnovu učestalosti pojavljivanja svakog karaktera. Dužina podintervala odgovara ućestalosti simbola. U tabeli 5.4 prikazano je kako ovo funkcioniše. Slovo A ima učestalost pojavljivanja 25 odsto i dodeljujemo mu interval [0, 0.25]. Ovo predstavlja 25 odsto intervala i dužine je 0.25. Zatim, pošto B ima učestalost 15 odsto, dodeljujemo mu sledećih 15 odsto intervala (nakon podintervala [0, 0.25]). Dakle, podinterval [0.25, 0.40], dužine0.15, odgovara slovu B. Pošto C ima učestalost pojavljivanja 10 odsto, dodeljuje mu se sledećih 10 odsto, što odgovara podintervalu [0.40, 0.50], dužine 0.10. Prateći ovaj šablon, dodeljujemo sledećih 20 odsto intervala slovu *D* i preostalih 30 odsto slovu *E.* Pošto je suma učestalosti 100 odsto, ukupna suma podintervala je 1. Osim toga, kolekcija podintervala kompletao "pokriva" originalni interval [0, 1].

Kako ovo može da pomogne u našem slučaju? Pretpostavite da imate string karaktera. Osnovna ideja je u sledećem.

1. Startujte od intervala [x, y] = [0, 1].
2. Pogledajte prvi karakter i utvrdite odgovarajući podinterval od [x, y], koji se zasniva na učestalosti karaktera.

Tabela 5.4: Pridruživanje opsega individualnim slovima na osnovu učestalosti pojavljivanja

Slovo	Učestalost (%)	Podinterval [p, q]
A	25	[0, 0.25]
B	15	[0.25,0.40]
C	10	[0.4, 0.5]
D	20	[0.5, 0.7]
E	30	[0.7, 1.0]

* Oni koji bolje poznaju matematiku možda znaju da postoje različiti nivoi beskonačnosti. U stvari, prema nekim merenjima, postoji mnogo više vrednosti izmedu 0 i 1 nego Što ima stringova karaktera.

3. Redefinišite interval [x, y] koji će predstavljati taj podinterval, tj. smanjite [x, y],

4. Ispitajte sledeći karakter i ponovo utvrdite odgovarajući podinterval [x, y], u zavisnosti od učestalosti karaktera. Ovo je identično onome što ste uradili u koraku 2, osim što sada radite sa manjim intervalom [x, y], umesto sa [0, I].

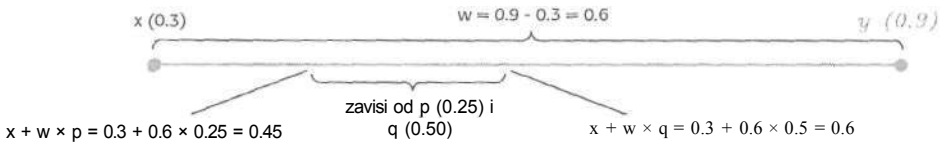
5. Ponovljamo korake 3 i 4 za svaki karakter u stringu.

Verovatno je najbitniji korak izračunavanje novog intervala na osnovu starog, uzimajući vrednosti za *p* i *q* (videti tabelu 5.4). Osnovna ideja je predstavljena na slici 5.3. Počinjemo od intervala [x, y] širine $w = y - x$. Zatim, postoje podintervali zasnovani na *p* i *q* za svaki karakter. Vrednosti p i q predstavljaju procenat rastojanja od *x* do *y*. Nakon toga se za novu vrednost za *x* postavjja vrednost *p,* a za *y* vrednost *cj*. Dalde, pretpostavimo da je [x, y] = [0.3, 0.9], w = 0.9 - 0.3 = 0.6 i da je [p, q] = [0.25, 0.50]. Nova vrednost za *x* je 25 odsto rastojanja od tekuće vrednosti *x* ka *y*. Drugim rečima, sledeća vrednost x = tekuća vrednost za $x + w \times p = 0.3 + 0.6 \times 0.25 = 0.45$. Nova vrednost za y = tekuca vrednost za $x + w \times q = 0.3 + 0.6 \times 0.5 = 0.6$. Sledeći način na koji ovo može da se zamisli je da se pretpostavi da 0.45 iznosi 25 odsto rastojanja od 0.3 do 0.9, a da 0.6 predstavlja 50 odsto rastojanja od 0.3 do 0.9.

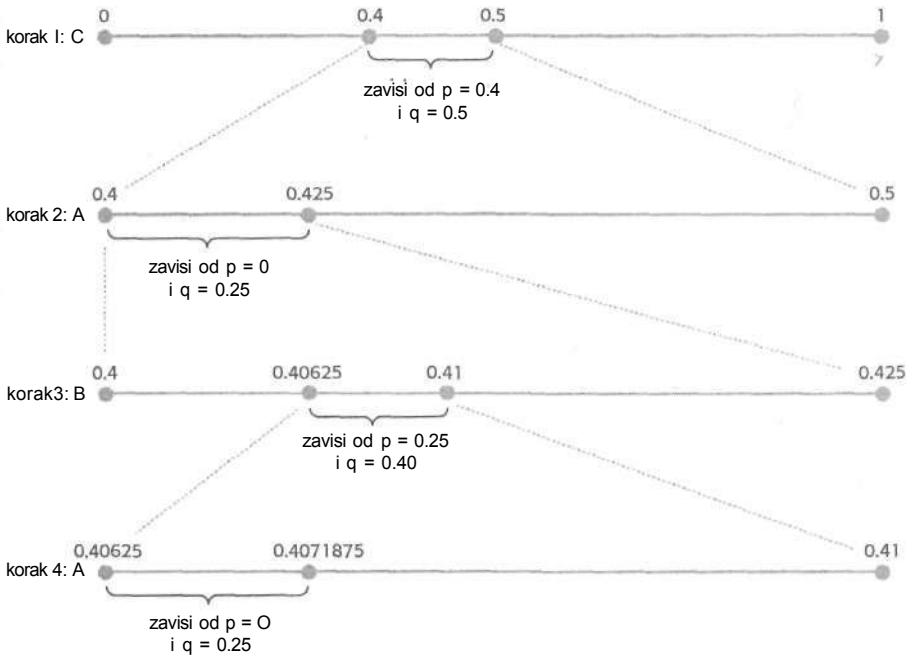
U osnovi, ovaj proces opisuje generisanje podintervala sa manjom dužinom. Osim toga, podintervali na jedinstven način zavise od karaktera u stringu i učestalosti njihovog pojavljivanja. Što je veći broj karaktera u stringu, to su podintervali manji (oni se smanjuju sa svakim novim karakterom u stringu). Kada se obradi poslednji karakter u stringu, birajte realni broj u finalnom intervalu (na primer, središnju tačku). Birajte bitsku reprezentaciju tog broja, a to je kompresovani kod.

Vreme je za primer. Pretpostavimo da važe učestalosti iz tabele 5.4 i da se koristi string CABACADA. Na slici 5.4 i u tabeli 5.5 prikazano je prvih nekoliko koraka procesa. Koristeći izračunavanja koja smo prethodno naveli, generišete sledeću sekvencu intervala: [0, 1], [0.4, 0.5], [0.4, 0.425], [0.40625, 0.41], [0.40625, 0.4071875] i [0.406625, 04067187], Trebalo bi da uklopite vrednosti iz svakog koraka na slici sa odgovarajućim koracima u tabeli i da potvrdite da su tačne. Još uvek nismo prošli kroz ceo string, tako da možete sami da izvežbate ostatak. Pretpostavite da ste se zaustavili nakon petog koraka. Onda možete da izaberete bilo koji broj iz intervala [0.406625, 04067187], kao, na primer, 0.4067, za predstavljanje stringa CABAC.

Kako sada obrnuti proces? Osnovna ideja je da se najpre utvrdi u kom delu intervala [0, 1] se nalazi broj. Tako se utvrduje prvi karakter u stringu. Na primer, pretpostavite da ste predstavili broj N = 0.4067 i da znate samo sadržaj tabele 5.4.



SLIKA 5.3 *Utvrđivdnje novog intervala na omovu starog*



SLIKA 5.4 *Intervali kod aritmetičkog kodiranja*

Uočavamo da se nalazi u intervalu [0.4, 0.5]. To znači da je prvi korak u procesu kompresije morao da počne intervalom [0.4, 0.5]. Dakle, prvi karakter je bio C, a svi ostali su morali da imaju broj iz nekog drugog podintervala. Kuda idemo dalje?

U tabeli 5.6 dati su koraci procesa dekompresije. Sada treba da utvrdite u kom delu intervala [0.4, 0.5] se nalazi vrednost N. Tako ćete utvrditi sleded karakter. Primećujete da je N mnogo bliže 0.4 nego 0.5. Koliko bliže? Da biste to izračunali, oduzmite *p* od *N,* čime se dobija 0.0067 (korak 1 u tabeli 5.6), i podelite to širinom intervala, tako da se dobija 0.067. To ukazuje da N iznosi samo 6,7 odsto rastojanja od *p* do *cj.* Ako ovo uporedite sa mogućim *[p, cj*] intervalima, videćete da se vrednost nalazi u prvoj četvrtini intervala [0.4, 0.5]. Drugim rečima, drugi korak procesa kompresije je morao da koristi [p, q] = [0, 0.25], što znači da Je drugi karakter morao da bude A.

U kom delu intervala [0, 0.25] se nalazi 0.067? Izgleda da je na četvrtini puta od 0 do 0.25, ali treba da budete malo precizniji. Izvodeći ista izračunavanja kao i malopre, oduzimajući p = 0 od N = 0.067 da bi se dobilo 0.067 i deleći sa širinom intervala 0.25, dobija se 0.268, što znači da se N = 0.067 nalazi na oko 26,8 odsto rastojanja od p = 0 do q = 0.25. Na taj način, treći korak procesa kompresije pokazuje da se mora koristiti interval [0.25, 0.40], odnosno da je treći karakter bio B.

Tabela 5.5: Koraci u procesu aritmetičkog kodiranja

Korak	String	Sledeći karakter	Tekući interval [x, y]	[p,q]	Širina intervala (w = y - x)	Izračunavanje za novo x (x = x + w × p)	Izračunavanje za novo y (y = x + w × q)
1	`	C	[0, 1]	[0.4, 0.5]	1.0	0 + 1 x 0.4 = 0.4	0 + 1 × 0.5 = 0.5
2	C	A	[0.4, 0.5]	[0, 0.25]	0.1	0.4 + 0.1 × 0 = 0.4	0.4 + 0.1 × 0.25 =0.425
3	CA	B	[0.4,0.425]	[0.25,0.40]	0.025	0.4 +0.025 × 0.25 = 0.40625	0.4 + 0.025 × 0.4 = 0.41
4	CAB	A	[0.40625,0.41]	[0,0.25]	0.00375	0.40625 +0.00375 × 0 = 0.40625	0.40625 +0.00375 × 0.25 = 0.4071875
5	CABA	C	[0.40625,0.4071875]	[0.4,0.5]	0.0009375	0.40625 + 0.0009375 × 0.40 = 0.406625	0.40625 + 0.0009375 × 0.5 = 0.4067187

Tabela 5.6: Izvlačenje karaktera na osnovu aritmetički kodiranog broja

Korak	N	Interval [p, q]	Širina	Karakter	N - p	Podeljeno sa širinom
1	0.4067	[0.4, 0.5]	0.10	C	0.0067	0.067
2	0.067	[0,0.25]	0.25	A	0.067	0.268
3	0.268	[0.25, 0.40]	0.15	B	0.018	0.12
4	0.12	[0,0.25]	0.25	A	0.12	0.48
5	0.48	[0.4, 0.5]	0.10	C	0.08	0.8

Ako nastavite na ovaj način, dobijate da je četvrti karakter bio A, a peti C (videti tabelu 5.6). Zajedničkim postavljanjem karaktera koje ste generisali dobijate string CABAC, tačno ono što je generisala i tabela 5.5.

Postoji jedan manji problem kod ovog metoda - kada treba da se zaustavite. Kada se vrši dekompresija, sve što predstavljate je broj i slično onome što se nalazi u tabeli 5.4. Da ste izveli algoritam kompresije još nekoliko koraka dalje, generisali biste manji interval u kome bi se našao broj 0.4067. Ne postoji ništa što bi ukazalo da se treba zaustaviti u odredenoj tački.

Algoritmi aritmetičke kompresije obično rešavaju ovaj problem dodavanjem jednog karaktera, terminalnog karaktera, na kraj originalnog stringa. Tretira se kao i ostali karakteri. Međutim, kada se generiše za vreme kompresije, proces se zaustavlja. Ovo je slično kompresovanju rečenica promenljive dužine, jedne po jedne, kada se zaustavljamo u slučaju da se generiše tačka.

Postoji još nešto što treba da istaknemo. Jednostavno smo opisali kako da pridružimo realan broj stringu. Pošto smo pretpostavili da će string biti kompresovan, moramo da se zapitamo kako je ovo povezano sa uzorcima bitova. Odgovor leži u metodima koji se koriste za smeštanje realnih brojeva. Ovde nećemo navoditi raspravu o uzorcima bitova za brojeve u pokretnom zarezu; ako Vas interesuju detalji, pogledajte referencu [St03], Glavna ideja je da se bilo koji realni broj izmedu 0 i 1 može predstaviti sa preciznošću od sedam cifara pomoću 32-bitne reprezentacije. Na osnovu tabele 5.5, svaki broj iz intervala [0.406625, 0.4067187] lako može da se predstavi sa željenom tačnošću pomoću 32 bita. U stvari, mogli bismo da obradimo još nekoliko karaktera, a da se i dalje koristi 32-bitna reprezentacija.

Medutim, da je string bio duži, subintervali bi bili manji nego što je potrebno za tačnost od sedam cifara. U ovom slučaju je potrebna proširena preciznost. Detalji bi bili prildadniji za knjigu o organizaciji kompjutera, ali moguće je pokazati da bilo koji broj između 0 i 1 može da se predstavi uz preciznost od 16 decimalnih mesta pomoću 64 bita. Ako 16 cifara nije dovoljno, onda bismo mogli da predemo na sledeći nivo.

U opštem slučaju, interval [x, y] postaje manji dok se broj karaktera povećava. Osim toga, postoje procedure koje mogu da aproksirniraju bilo koji realni broj do odredene tačnosti ako se koristi dovoljan broj bitova.

5.3 Run-Length kodiranje

Hafmanovi kodovi redukuju broj bitova koji se šalju, ali je kod njih neophodno znati vrednost učestalosti pojavljivanja. Kao što smo opisali, kod njih je pretpostavljeno da su bitovi grupisani u karaktere, ili neke druge jedinice koje se ponavljaju. Mnogi elementi koji putuju preko komunikacionog medijuma, uključujud'binarne (mašinski kodirane) fajlove, podatke koji se šalju faksom i video signale, ne ubrajaju se u tu kategoriju.

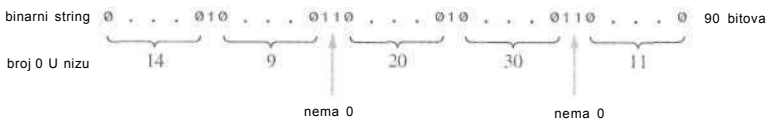
Na primer, bitovi koji se prenose preko faksa odgovaraju rasporedu svetlih i tamnih oblasti na parčetu papira. Nema direktnog prenosa karaktera. Zbog toga, postoji potreba za opštijom tehnikom koja može da kompresuje proizvoljne nizove bitova. Jedan pristup, poznat pod nazivom **run-length kodiranje,** koristi jednostavan i očigledan koncept: niz bitova se analizira da bi se pronašli dugački nizovi nula ili, jedinica. Umesto svi bitovi, šalje se samo njihov broj u nizu.

Ova tehnika je posebno korisna prilikom prenosa podataka pornoću faksa. Ako pažljivije proučite prostor na kome je otkucan jedan karakter, videćete da je 70 do 80 odsto prostora bele boje. Naravno, tačna količina zavisi od fonta i samog karaktera. Konkretna tamna mesta otkucanog karaktera učestvuju sa veoma malim procentom u prenosu pomoću faksa. Na primer, primetićete količinu belog prostora kod uvećane reprezentacije malog slova / na jednoj štampanoj poziciji.

Nizovi istog bita

Postoji nekoliko načina za implementaciju run-length kodiranja. Prvi je posebno koristan kod binamih stringova, kod kojih se u večini nizova javlja isti bit. U primeru stringa koji se prenosi preko faksa imamo string koji je sastavljen prvenstveno od karaktera i zato će postojati dugački nizovi 0 (uz pretpostavku da 0 odgovara svetlim mestima). Ovaj pristup podrazumeva slanje dužine svakog niza u vidu binarno izraženog celog broja fiksne dužine. Prijemni uredaj prihvata dužinu svakog niza i generiše oďgovarajući broj bitova u nizu, umećući drugi bil izmedu tih nizova.

Na primer, pretpostavimo da se za predstavljanje dužine niza koriste grupe od po četiri bita. Razmotrimo niz sa slike 5.5a. Na slici 5.5b prikazan je poslati kompresovani string.



(a) String pre kompresovanja

binami nizovi fiksne dužine	1110	1001	0000	1111	0101	1111	1111	0000	0000	1011
decimalniniz	14	9	0	15	5	15	15	0	0	11

(b) Run-length kodirani string

SLIKA 5.5 *String pre kompresije i nakon run-length kodiranja*

Originalni string počinje sa 14 nula, tako da su prva četiri bita kompresovanog stringa 1110 (binarno 14). Sledeća četiri bita u kompresovanom stringu su 1001 (binarno 9 za sledeći niz od devet 0). Nakon drugog niza postoje dve uzastopne jedinice. Međutim, kod ovog pristupa smatra se da izmedu uzastopnih jedinica postoji niz bez nula. Zato se u trećoj grupi od četiri bita koristi kombinacija 0000.

U četvrtom nizu postoji 20 nula. Medutim, broj 20 ne može binarno da se predstavi sa četiri bita. Il tom slučaju, koristi se još jedna grupa od četiri bita. Dva 4-bitna broja se nakon toga sabiraju i dobija se ukupna dužina niza. Na slici 5.5b 1111 (binarno 15) i 0101 (binarno 5) određuju niz dužine 20.

Ako je dužina niza suviše velika da bi se izrazila kao suma dve 4-bitne grupe, koristi se onoliko grupa koliko je potrebno za definisanje te sume. Prijemni uredaj mora da zna da grupa sa svim jedinicama ukazuje da iza nje sledi još jedna grupa koja defmiše dužinu niza. Tako se nastavlja sumiranje sve dok se ne dođe do grupe u kojoj se ne nalaze sve jedinice. Na primer, niz dužine 30 je predstavljen sa 1111, 1111 i 0000. U ovom slučaju je neophodna grupa sa svim nulama da bi se uređaju "saopštilo" da se niz završava nakon 30 nula.

Kako bi izgledao kompresovani string da je originalni string sa slike 5.5a počeo jedinicom? Slično slučaju sa dve uzastopne jedinice, metod "smatra" da string počinje nizom nula dužine 0. Dakle, prva četiri bita su 0000.

Ova tehnika je najefikasnija kada postoji veliki broj dugačkih nizova 0. Kako se povećava učestalost pojavljivanja jedinica, tako se smanjuje efikasnost ovog metoda. U stvari, moguće je iskoristiti ovaj princip za kodiranje dugačkih nizova nekog drugog karaktera.

Nizovi sa različitim karakterima

Ako znate da se u stringu često javlja isti bit, okolnosti se pojednostavljuju, jer je potrebno slati samo dužinu niza. Sta je sa slučajevima kada postoje nizovi različitih bitova, ili, čak, karaktera? Verovatno pretpostavljate rešenje za te slučajeve: šalje se konkretni karakter, zajedno sa dužinom njegovog niza. Na primer, string

HHHHHHHUFFFFFFFFFFFFFFYYYYYYYYYYYYYYYYYYYYDGGGGGGGGGGG

može da se prenese naizmeničnim slanjem broja pojavljivanja u nizu i karaktera 7, H, 1, U, 14, F, 20, Y, 1,Di 11, G.

Faksimil kompresija

Duži niz godina (pre pojave WinZipa i Interneta) jedan od najčešće korišćenih primera korišćenja kompresije bio je prenos podataka pomoću faksa. U odeljku 4.2 opisan je osnovni princip rada faks mašine koja skenira stranicu i kreira bit mapiranu reprezentaciju slike sa papira. U opštem slučaju, crno-bela slika je sastavljena od velikog broja piksela, koji predstavljaju bele i crne tačke na stranici. U stvari, ITU je defmisao operacione standarde za različite grupe mašina. Ovde se nećemo baviti razlikama izmedu tih grupa, već ćemo se fokusirati na šeme kompresije da bismo se nadovezali na prethodnu raspravu. Oni koji su zainteresovani za detaljniju diskusiju mogu da pogledaju reference [SaOO], [HaOl], [Ho97] i [He96].

Za našu diskusiju relevantna su dva ITU standarda - T.4 i T.6, koji defmišu kompresiju za ono što nazivamo mašinama Grupe 3 i Grupe 4. U osnovi, to su mašine koje koriste digitalne metode za prenos slika preko telefonske mreže. Iako su moguće različite veličine stranica, mi ćemo se fokusirati samo na A4 dokumente, tj. stranice veličine 210x297 mm.

A4 dokument sadrži 1.728 piksela u svakoj liniji. Ako bismo poslali po jedan bit za svaki piksel, morali bismo da prenesemo više od tri miliona bitova za svaku stranicu - to je ogromna količina informacija za veoma jednostavne slike. Standardi T.4 i T.6 koriste činjenicu da tipična slika sadrži mnogo uzastopnih belih, ili crnih piksela, tako da postoji veliki broj nizova sa uzastopnim nulama, ili jedinicama. Ovo je defmitivno dobra osnova za korišćenje run-length kompresije. Ipak, postoje dve "stvari" koje treba istađ. Prvo, dužina niza može da varira izmedu 0 i 1.728, kreirajući razne moguće kombinacije. Međutim, ovaj metod ispoljava odredenu dozu neefikasnosti prilikom postavljanja formata za predstavljanje bilo kog broja iz ovog opsega. Drugo, neki nizovi mogu da se jave sa veoma visokom učestalošću pojavljivanja. Na primer, veclna otkucanih stranica sadrži uglavnom bele piksele, rnožda čak 80 odsto, ako ne i više. Razmak izmedu susednih slova, ili pre prvogslova u liniji obično je konstantan. I između linija možete da očekujete da nema crnih piksela - drugim rečima, postoje samo dugački nizovi belih piksela. Posle svake linije teksta možete da očekujete nekoliko linija (u zavisnosti od proreda) od po 1.728 belih piksela.

Krajnji zaključak je da se kod mnogih faks slika može predvideti, prilično pouzdano, verovatnoća da ce se odredeni nizovi pojaviti. Ovo nalaže korišćenje nekog tipa frekventno zavisnog kodiranja koje se zasniva na dužini niza. Standardi T.4 i T.6 za faksimil kompresiju koriste, u stvari, kombinaciju nizova sa belim i cmim pikselima, poštujući frekventno zavisni kod koji je definisan učestalošdi pojavljivanja tih nizova - naziva se *modifikovani Hafmanov kod.* Pretpostavke i proces se sastoje u sledećem:

1. Svaka linija sadrži naizmenične nizove belih i crnih piksela.
2. Svaka linija počinje nizom belih piksela. Cak i kada se prenosi slika koja ima crne ivice, proces poznat pod nazivom *overscanning* dodaje po jedan beli piksel na početak i na kraj svake linije.
3. Izračunavaju se kodovi za naizmenične nizove belih i crnih piksela, a zatim se prenose kodirani bitovi.

U tabeli 5.7 prikazan je podskup kodova koji definišu dužine nizova belih i crnih piksela. Kompletna tabela defmiše kod za nizove dužine izmedu 0 i 63, zaključno, i za dužine 64, 128, 192, 256 i tako redom. Nakon 64, ITU definiše samo kodove za dužine koje predstavljaju umnožak broja 64. Tako se smanjuje ukupan broj potrebnih kodova. Kodovi za nizove čija je dužina manja od 64 nazivaju se konačni kodovi *[terminating codes),* a oni čija je dužina umnožak broja 64 nazivaju se kodovi "doterivanja" *(makeup codes).* Svaki niz čija je dužina manja od 64 piksela kodira se pomoću konačnog koda. Ako je dužina veća od 64 piskela, metod koristi kod "doterivanja" za najduži niz koji se u potpunosti može smestiti u originalni niz i konačni kod za preostale bitove.

Na primer, niz koji se sastoji od 50 belih piksela može da se kodira kao 01010011. Zatim, uzmimo primer niza koji ima 572 bela piksela. On se interpretira kao niz dužine 512 piksela, iza koga slediniz od još 60 piksela. Pridruženi kod je 01100101-01001011. U ovoj instanci 512 bitova je kompresovano na 16 bitova, što predstavlja redukciju od skoro 97 odsto.

Tabela 5.7: Neki kodovi faksimil kompresije

	Broi piksela u nizu	Kod: Niz belih piksela	Kod: Niz crnih piksela
Konačni kodovi	0	00110101	0000110111
	1	000111	010
	2	0111	11
	3	1000	10
	10	00111	0000100
	20	0001000	00001101000
	30	00000011	000001101000
	40	00101001	000001101100
	50	01010011	000001010010
	60	01001011	000000101100
Kodovi "doterivanja"	64	11011	0000001111
	128	10010	000011001000
	256	0110111	000001011011
	512	01100101	0000001101100
	768	011001101	000000101100
	1024	011010101	0000001110100
	1280	011011001	0000001010010
	1536	010011001	0000001011010

Naravno, veći broj manjih nizova neće imati toliko veliki procenat kompresije, ali nije neuobičajeno postići kompresiju od 90 odsto.

Sledi nekoliko opažanja na osnovn tabele 5.7.

- Kodovi za nizove belih piksela su kraći od nizova crnih piksela, jer su beli nizovi češći.
- I za bele i za crne nizove koriste se run-length kodovi sa no-prefix svojstvom.
- Kompresija je bolja kod slika sa dužim nizovima.

Iako ovo važi i za T.4 i za T.6 standard, postoje razlike izmedu njih. Na primer, šema koju smo opisali dobro funkcioniše za tipične kucane stranice. Obično postoji dovoljno nizova u svakoj liniji da bi se efikasno iskoristili kodovi za kompresiju. Međutim, ako slika sadrži složene uzorke, ili je, možda, reč o fotografiji, nekoliko dugačkih nizova može da se nađe bilo gde na slici, a ostatak obično čine veoma kratki nizovi. Pošto se kratki nizovi kompresuju sa manjim procentom redukcije, ovaj pristup (bar ovaj koji smo ovde opisali) ne funkcioniše baš najbolje.

Alternativa koju je usvojio standard T.6 koristi činjenicu da se dve uzastopne linije verovatno ne razlikuju isuviše. Zbog toga, umesto da se kompresuje svaka linija nezavisno, standard T.6 uspostavlja osnovnu liniju, pa utvrduje razliku izmedu nje i sledeće linije. Ako je razlika veoma mala, sledeća linija može da se kodira sa manjom količinom dodatnih informacija; tako može da se postigne zadovoljavajuća kompresija. Pošto u sledećem odeljku objašnjavamo kompresiju

koja se zasniva na utvrdivanju različitosti, ovde se nećemo baviti detaljima. Detaljnije predstavljanje standarda T.6 možete da pronadete u referenci [SaOO].

5.4 Relativno kodiranje

Do sada predstavljene tehnike kompresovanja imaju svoju primenu, ali u odredenim slučajevima nisu mnogo korisne. Uobičajeni primer je prenos video signala, kod kojih slike rnogu da budu veoma složene, za razliku od crno-belih slika koje se prenose faksom, ili kada se prenosi tekstualni fajl. Osim možda manjeg broja probnih uzoraka koji se šalju pre nego što se stanica zvanično pusti u rad, video slike se retko ponavljaju. Zbog toga, ni jedan prethodno opisani metod "ne obećava" mnogo.

Tipični televizijski signal šalje 30 slika u sekundi. Osim toga, svaka slika u opštem slučaju ima samo manje varijacije u poređenju sa prethodnom. U toku jednog delića sekunde ne može da se desi mnogo toga. Llmesto da se svaki kadar tretira kao zaseban entitet koji se zasebno kompresuje, možemo da razmišljamo u pravcu utvrdivanja razlika u svakom sledećem kadru u odnosu na prethodni. Kodiranje tih informacija i njihovo slanje ima potencijala kada su razlike male. Ovaj metod se naziva relativno, ili diferencijalno kodiranje.

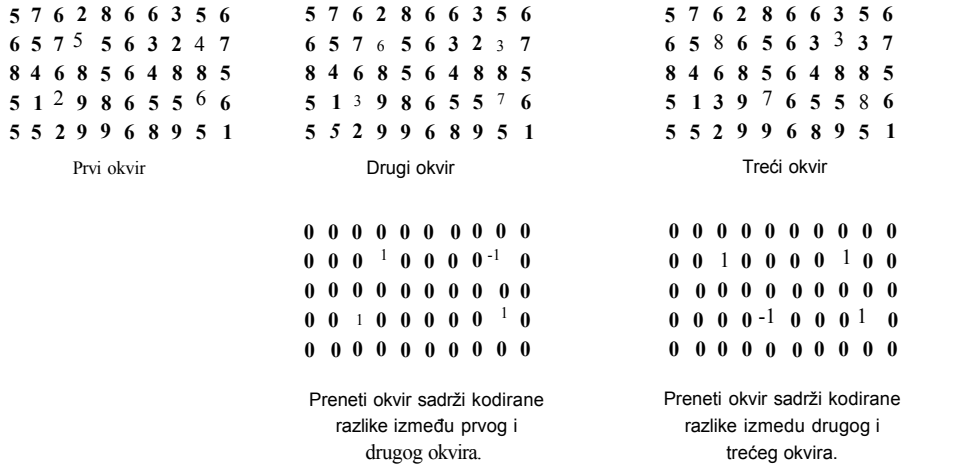
Prindp je prilično jednostavan. Šalje se prvi kadar i smešta se u bafer prijemnika. Nakon toga, pošiljalac poredi drugi kadar sa prvim, kodira razlike i šalje ih u formatu okvira. Prijemnik dobija okvir i primenjuje razlike na kadar koji ima i tako se kreira sledeći kadar. Drugi kadar se smešta u bafer, pa se proces nastavlja za svaki sledeći kadar.

Na slici 5.6 pokazano je kako ovo funkcioniše. Ovde smo okvir predstavili kao dvodimenzionalni niz celih brojeva. Ne interpretiramo njihovo značenje; ovo je jednostavnija reprezentacija Li poredenju sa crtanjem video signala. Prvi okvir sadrži skup celih brojeva, a drugi se malo razlikuje od prethodnog (obojeni brojevi ukazuju na razlike).

Na slici je prikazan sledeći dvodimenzionalni niz ispod drugog okvira, sa nulama (O), jedinicama (1) i -jedinicama (-1). 0 na bilo kojoj poziciji znači da nema promene u odnosu na prethodni kadar. Nenulta vrednost ukazuje na promenu. 1 znači da je element na toj poziciji za 1 veći od elementa na istoj poziciji u prethodnom kadru, a -1 znači da je element manji za 1. Naravno, mogu se koristiti i druge vrednosti osim I i -1. Poenta je u tome da poslati okvir sadrži dugačke nizove 0, tako da predstavlja dobrog "kandidata" za primenu run-length kodiranja.

5.5 Lempel-Ziv kompresija

Kod run-length kodiranja kompresija se postiže traženjem dugaćkih nizova istog karaktera, ili bita. Ideja je da se redukuje broj ponavljanja, ili redundantnih prenosa. Ali, sve redundantnosti se ne javljaju u formi ponavljanja jednog bita, ili jednog karaktera. Ovo je čest slučaj kod velikih tekstualnih fajlova, kao što su rukopisi.



SLIKA 5.6 *Relativno kodiranje*

Autorov stil pisanja ćesto odreduju reči, ili fraze koje najčešće upotrebljava. Izvorni kod programa obično ima naziv jedne promenljive (string) koji se često ponavlja.

Metod **Lempel-Ziv kompresije** traži stringove koji se često ponavljaju i smešta ih samo jednom.* Zatim se sve pojave tog stringa menjaju kodom koji odgovara originalnom stringu. Ovo je jedan od osnovnih principa za upravljanje bazama podataka: jedan deo informacija se smešta samo na jednom mestu i referencira se kroz specijalne kodove. Ova tehnika se koristi u UNIX-ovim komandama za kompresovanje i u V.24bis standardu za kompresovanje za modeme. Osim toga, GIF (Graphics Interchange Format) fajlovi koriste varijantu Lempel-Ziv kodiranja.

Na primer, razmotrite sledeći pisani primer (ref. [Cr90]):

The tropical rain fell in drenching sheets, hammering the corrugated roof of the clinic building, raoring down the metal gutters, splashing on the ground in a torrent.

Nekoliko slovnih sekvenci se ponavlja. Ignorišući veličinu slova, neka od njih su *the, ro* i *ing.* Zamenimo svaki od ovih stringova specijalnim karakterima ⊗,⊕ i ∅, respektivno. Kompresovani string u tom slučaju postaje

⊗t⊕pical rain fell in drench∅ sheets, hammer∅ ⊗corrugated ⊕of ⊗clinic build∅,
⊕ar∅ down ⊗metal gutters, splash∅ on ⊗g⊕und in a torrent.

Ovaj tekst se ne kompresuje u velikoj meri u poredenju sa originalnim, ali ne možete da očekujete mnogo ponavljanja u jednoj rečenici. Kompresija u opštem slučaju nije veoma efikasna (ili korisna, u tu svrhu) kada se radi sa kraćim primercima.

*U stvari, postoji nekoliko drugaČijih Lempel-Ziv šema kompresija. Ove varijacije mogu da se označe kao Lempel-Ziv, Lempel-Ziv-Welch, LZW, *IZ77,* LZ78, ili, čak, Ziv-Lempel. Nećemo navoditi razlike izmedu varijacija; ako želite da saznate više detalja, možete da konsultujete referencu [Ho97].

Kod dužeg teksta i većeg broja ponavljanja kompresija se poboljšava, jer tipični tekst na engleskom jeziku ima brojna ponavljanja reči, kao što su *the, then, them, their, there* i *these*. Svaki fajl može da se smatra sekvencom ASCII definisanih karaktera. Ako fajl sadrži brojne sekvence karaktera koje se često ponavljaju, onda je "kandidat" za Lempel-Ziv kompresiju.

Značajna karakteristika ovog metoda je da ne postoje nikakve pretpostavke kako izgleda string koji se ponavlja, što algoritam čini robustnim i dinamičnim. Medutim, na prvi pogled može da izgleda kao da ne postoji efikasan način za implementiranje algoritma, jer traženje sekvenci koje se ponavljaju izgleda prilično naporno. Zatim, tu je problem dekompresije. Algoritam kompresije nema velike koristi ako proces ne može da se izvede u suprotnom smeru. Kako se vrši dekompresija? Ako smo primili prethodno prikazani string, kako možemo da znamo koje su sekvence slova zamenjene specijalnim simbolima? Jedna opcija može da bude slanje tabele simbola u kojoj su prikazani stringovi koji se ponavljaju zajedno sa kompresovanim tekstom. Naravno, i tim se postupkom delimično umanjuje prvobitna vrednost kompresije.

Ispostavlja se da postoji efikasan način za identifikovanje stringova koji se često ponavljaju. Osim toga, iako možda deluje čudno, moguće je utvrditi stringove pridružene specijalnim simbolima bez njihovog prenosa zajedno sa kompresovanim tekstom. Pokazaćemo kako ovo funkcioniše opisivanjem algoritma i kompresije i dekompresije. Ipak, skratićemo primer kompresovanjem teksta koji se sastoji samo od tri karaktera A, B i C. Da smo radili sa celim alfabetom, bile bi neophodne stotine koraka da se potpuno razume vrednost kompresije. Ovaj ograničeni primer zadržava osnovnu logiku na kojoj se algoritam zasniva i izvršava taj zadatak u svega nekoliko koraka.

Na slici 5.7 prikazani su algoritmi kompresije i dekompresije - reference IWe84] i [DrOl]. Oni zadržavaju osnovnu logiku, ali ne obezbeđuju sve detalje i deklaracije koje su specifične za programski jezik C. Algoritam kompresije je zasnovan na sledećim centralnim idejama:

1. Dodeljujete kod za svako slovo, ili karakter koji je deo inicijalnog tekstualnog fajla (Iinija 3 algoritma kompresije) i smeštate ih u kodnu tabelu.

2. Postavlja se petlja i uzima jedan po jedan karakter iz fajla. Koristićemo baferovani string (inicijalno prvi karakter, linija 4) koji se formira nadovezivanjem karaktera iz fajla.

3. U svakom prolasku kroz petlju čita se jedan karakter i dodaje se na baferovani string, tako da se formira novi privremeni string (tempstring, linija 7). Ako je taj privremeni string već ranije pronaden (ako se već nalazi u kodnoj tabeli), privremeni string se premešta u bafer (linija 10).

4. Ako privremeni string nije pronaden u kodnoj tabeli, njemu se dodeljuje kod, oba se smeštaju u kodnu tabelu (linija 14) i šalje se kod pridružen baferovanom stringu (linija 13). Ovaj kod predstavlja kompresovani ekvivalent stringa. Na kraju, reinicijalizuje se baferovani string za jedan karakter koji je upravo pročitan (linija 15).

Kompresija

```
1   void compress (FILE * fileid)
2   {
3     initialize(code table);
4     buffer=string oonsisting
        of first character from the file
5    while ( (c=getc(fileid)) != EOF)
6    {
7   tempstring=concat(buffer, c);
8   search for tempstring in the
    code table;
9   if found
10  buffer = tempstring;
11  else
12  {
13  send the code associated
    with buffer;
14  assign a code to tempstring;
      store both in the code table;
15  buffer=string consisting of
      one character c;
16  }
17  } //while loop
18  send the code associated
      with buffer;
19  } //compress
20
21
22
23
24
25
```

Dekotnpresija

```
1   void decompress
2   {
3     initialize(code table);
4     receive first code, call it prior;

5     print the string associated with pri
6     while (true)
7     {
8       receive code, call it current; if
         code then break;
9       search for current in the code tah
10      if not found
11      {
12        c=1st character of string
           associated with prior;
13        tempstring=concat(string
           associated vvith prior, c);
14        assign a code to tempstring;
           store both in the code table;
15         print tempstring;
16  }
17  else
18  {
19      c=1st character of string
         associated with current;
20      tempstring=concat(string
         associated With prior, c);
21      assign a code to tempstring;
         store both in the code table
22      print string associated with
         current
23       }
24    prior=current
25  } //while loop
    } //decompress
```

SLIKA 5.7 *Lempel-Ziv algoritmi kompresije i dekompresije*

Pogledajmo kako ovaj algoritam funkcioniše na konkretnom primeru. Pretpostavimo da su karakteri iz fajla čitani sledećim redosledom

```
ABABABCBABABABCBABABABCBA
```

U tabeli 5.8 prikazane su vrednosti relevantnih promenljivih i koje su akcije preduzete u svakom koraku, a u tabeli 5.9 rezultati kodne tabele nakon kompletiranja algoritma kompresije. Inicijalno, kodna tabela sadrži samo tri ulaza A, B i C, kojima su pridruženi kodovi 0, 1 i 2, respektivno. Dok algoritam dodaje nove stringove u kodnu tabelu, pretpostavljamo da kreira sukcesivne kodove, počevši od 3, za svaki novi dodati string. Dok budete izvodili svaki sledeći korak, imajte na umu da je tempstring baferovani string kome je dodato slovo c.

Kada korak I algoritma bude tražio AB u kodnoj tabli, neće uspeti da nade traženu kombinaciju. Salje kod za baferovani string A (0) i smešta AB (kod = 3) u kodnu tabelu; definiše se novi baferovani string kao B. Korak 2 traži BA u kodnoj tabeli i ne uspeva da ga pronade. Salje kod za B (1), smešta BA (kod = 4} u kodnu tabelu i definiše baferovani string kao A.

Tabela 5.8: Postupni rezultati algoritma za kompresovanje

Prolazak kroz petlju	Bafer	C	Šta je poslato	Šta je smešteno u tabelu	Nova vrednost u baferu
1	A	B	0 (kod za A)	AB (kod = 3)	B
2	B	A		BA (kod = 4)	A
3	A	B	1 (kod za B)		AB
4	AB	A		ABA (kod = 5)	A
5	A	B	3 (kod za AB)		AB
6	AB	C		ABC (kod = 6)	C
7	C	B	3 (kod za AB)	CB (kod = 7)	B
8	B	A	2 (kod za C)		BA
9	BA	B		BAB (kod = 8)	B
10	B	A	4 (kod za BA)		BA
11	BA	B			BAB
12	BAB	A		BABA (kod = 9)	A
13	A	B	8 (kod za BAB)		AB
14	AB	C			ABC
15	ABC	B		ABCB (kod= 10)	B
16	B	A	6 (kod za ABC)		BA
17	BA	B			BAB
18	BAB	A			BABA
19	BABA	B	9 (kod za BABA)	BABAB (kod= 11)	B

Tabela 5.9: Tabela koja se dobija algoritmom kompresije

Strine	AB	C	AB	BA	ABA	ABC	CB	BAB	BABA	ABCB	BABAB	BABC	CBA
Kod	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Ulazni string: ABABABCBABABABCBABABABCBA
Preneti kod: 0 1 3 3 2 4 8 6 9 8 7 0

Korak 3 traži AB u kodnoj tabeli i pronalazi. Ništa se ne šalje, a novi baferovani string je AB. Korak 4 algoritma traži ABA u kodnoj tabeli i ne uspeva da pronađe. Salje kod za baferovani string AB (3), smešta ABA (kod = 5) u kodnu tabelu, pa definiše novi baferovani string kao A.

Ovaj proces se nastavlja i u kodnu tabelu se smešta novi string. Osim toga, privremeni stringovi se sve češće pronalaze u kodnoj tabeli i sve rede se inicira prenos. Zbog ovoga, baferovani string postaje duži, a kada ne uspe da pronađe traženu kombinaciju u tabeli, preneti kod odgovara dužem stringu. Rezultat je bolja kompresija. U tabeli 5.8 nije prikazano nekoliko zadnjih koraka algoritma; ostavljamo ih Vama da ih provežbate.

Sledeći korak je opisivanje algoritma dekompresije. Zapamtite da algoritam dekompresije mora da radi sa inicijalnom kodnom tabelom karaktera (u našem slučaju, kodna tabela se sastoji od A, B i C, sa kodovima 0, 1 i 2) i dolazećim kodnim vrednostima. U našem primeru ulaz algoritma dekompresije je sekvenca 0 1 3 3 2 4 8 6 9 8 7 0. Pokazaćemo da će njegov izlaz biti isti kao i originalni string, koji je korišćen kao ulaz algoritma kompresije. Poseban kvalitet algoritma dekompresije je u tome što može da rekonstruiše istu kodnu tabelu na osnovu ovih ograničenih informacija. Pogledajmo kako ovo funkcioniše u prvih nekoliko koraka. Sledeći pasus pročitajte polako i pažljivo, detaljno proučavajući reference na algoritme koji su naznačeni u tabeli.

Inicijalno, algoritam dekompresije prima kod i naziva ga `prior` (linija 4 na slici 5.7). Štampa string koji je pridružen uz `prior`, koji pronalazi u kodnoj tabeli (linija 5). U ovom slučaju `prior` je 0 i štampa se slovo A. Nakon toga, algoritam ulazi u petlju. U tabeli 5.10 prikazane su relevantne vrednosti u svakom prolasku kroz petlju i šta se štampa. U prvom prolasku algoritam prima tekući kod 1 (linija 8 na slici 5.7). Tekući kod je u kodnoj tabeli, tako da algoritam izvršava linije 19 do 22, smeštajući AB/3 kao par tempstring/kod u kodnu tabelu i štampa string pridružen tekućem kodu (B). Primećujete da su, kao i kod kompresije, kodne vrednosti dodeljivane sukcesivno, počevši od 3. U drugom prolasku kod za `prior` je 1, a novi tekući kod je 3. Tekući kod je u kodnoj tabeli (nakon prvog prolaska), tako da algoritam ponovo izvršava linije 19 do 22. U kodnu tabelu smešta BA/4 kao par tempstring/kod i štampa string pridružen tekućem kodu (AB). U trećem prolasku kod za `prior` je 3 i novi tekući kod je ponovo 3. Tekući kod se već nalazi u kodnoj tabeli, tako da algoritam ponovo izvršava linije 19 do 22. U kodnu tabelu smešta ABA/5 kao par tempstring/kod i štampa string pridružen tekućem kodu (AB).

Tabela 5.10: Postupni rezultati algoritma dekompresije

Prolazak kroz petlju	Prior (string)	Current (string)	Da li je kod trenutno u tabeli?	C	Par tempstring/ kod	Šta se štampa (current, ili tempstring)
1	0 (A)	1 (B)	Da	**B**	**AB/3**	B (current)
2	1 (B)	3 (AB)	Da	**A**	**BA/4**	AB (current)
3	3 (AB)	3 (AB)	Da	**A**	**ABA/5**	AB (current)
4	3 (AB)	2 (C)	Da	**C**	**ABC/6**	C (current)
5	2 (C)	4 (BA)	Da	**B**	**CB/7**	BA (current)
6	4 (BA)	8	Ne	**B**	**BAB/8**	BAB (tempstring)
7	8 (BAB)	6 (ABC)	Da	**A**	**BABA/9**	ABC (current)
8	6 (ABC)	9 (BABA)	Da	**B**	**ABCB/10**	BABA (current)
9	9 (BABA)	8 (BAB)	Da	**B**	**BABAB/11**	BAB (current)
10	8 (BAB)	7 (CB)	Da	**C**	**BABC/12**	CB (current)
11	7 (CB)	0 (A)	Da	**A**	**CBA/13**	A (current)

Preostali koraci se izvode na sličan način. Ovde je značajno istaći način na koji se kodna tabela konstruiše. Poređenje onoga što je smešteno u kodnu tabelu sa onim što je tu smestio algoritam kompresije (tabela 5.8) pokazuje da je kodna tabela formirana na isti način. Zato se kodnom tabelom stvaraju isti stringovi i kod se dekompresuje. Kao što možete da vidite, iza prvog odštampanog slova (A) slede odštampana slova iz poslednje kolone tabele 5.10, koja odgovaraju inicijalnom stringu.

Kao finalnu napomenu, pokazali smo kodnu tabelu kao dvodimenzionalnu tabelu, sugerišući korišćenje linearnih pretraživanja kada se traže kodovi i stringovi. Ovo će u opštem slučaju smanjiti efikasnost oba algoritma u značajnoj meri. Umesto tabele, bolje je koristiti strukturu podataka, zasnovanu na principu rečnika, kod koje se pretraživanja mogu izvesti mnogo efikasnije. U stvari, Lempel-Ziv algoritmi predstavljaju primere opšte klase algoritama koji se nazivaju algoritmi kompresije zasnovani na rečnicima. Mi ovde nismo koristili strukturu podataka iz rečnika, jer to pripada kursu o strukturama podataka; detalji bi premašili planirani obim predstavljanja Lempel-Ziv algoritma. Ako želite da saznate više detalja o strukturama podataka zasnovanim na principima rečnika, pogledajte referencu [DrOl].

5.6 Kompresija slika

Reprezentacija slika

Izuzetan napredak u poslednjih nekoliko godina ostvaren je u integraciji multimedijalnih aplikadja i kompjuterskih programa i mreža. Pomoću jednog klika mišem možemo da pristupimo fotografijama, čuvenim umetničkim slikama koje su izložene u Luvru, pa, čak, i filmskim insertima sa Interneta, ili CD-ROM-a. Iako prenos slika na prvi pogled može da deluje kao samo mali korak napred u odnosu na prenos reči i rečenica, multimedijalne aplikacije jednostavno ne bi bile izvodljive bez nekih veoma sofisticiranih algoritama kompresije.

U ovom i narednom odeljku obradili smo dva popularna metoda kompresije koji se koriste prilikom prenosa i skladištenja vizuelnih slika. Prikazaćemo kako se jedna vizuelna slika poput fotografije može kompresovati, a zatim prelazimo na video, koji, u suštini, predstavlja niz nepokretnih slika prikazanih velikom brzinom tako da se stvori osećaj kretanja, Medutim, pre nego što pređemo na metode kompresije, najpre moramo da predstavimo način na koji se vizuelne slike mogu smeštati i zašto je kompresija neophodna.

Slike, bilo da je reč o fotografijama, ili slikama generisanim na kompjuterskom ekranu, sačinjene su od velikog broja veoma malih tačaka. Ove tačke se nazivaju još i elementi slike, ili pikseli. Ako je slika izradena u visokom kvalitetu, verovatno nećete moći da vidite ove tačke ako ne primaknete nos sasviro uz ekran, ili ako ne koristite lupu. One su veoma gusto pakovane, tako da nervni senzori u našim očima ne mogu da ih razaznaju. Kod slika izrađenih sa lošijim kvalitetom tačke mogu da se uoče. Na primer, pogledajte pažljivo fotografiju u novinama i moći ćete da razaznate individualne tačke. Ako udaljite sliku, tačke će početi da se spajaju i slika će izgledati kao kompaktna celina. Ovo bi moglo da se pokuša i sa televizijskim ekranom, ali svi se verovatno još iz detinjstva sećamo upozorenja: "Nemoj da sediš blizu televizora, jer ćeš tako oslepeti."

Raspravu počinjemo objašnjenjem načina na koji se pikseli mogu predstaviti u memoriji kompjutera. Kada smo predstavili prenos pomoću faksa, rekli smo da je slika sačinjena od belih i cmih piksela - mogli smo da koristimo ili O, ili 1 za predstavljanje svakog piksela. Medutim, filmovi, ili slike u crno-beloj tehnici ne zadovoljavaju današnje standarde. U stvari, fraza crno-belo se pogrešno koristi kada je reč o starim filmovima, ili fotografijama. Te slike se, u stvari, sastoje od nekoliko nijansi sive boje i svaki piksel mora da ima mogučnost prikazivanja drugačije nijanse. Obično se koristi 8-bitna šema za predstavljanje 256 nijansi sive boje (koje se nalaze izmedu bele i crne boje).

Predstavljanje slika postaje veoma složeno kada se doda boja. Visokokvalitetne slike dopuštaju korišćenje širokog opsega boja i suptilnih prelaza izmedu boja i zasićenosti boje. Slike laserskog kvaliteta možda ne uspevaju da postignu različite nijanse crvene, ili narandžaste boje. Na primer, personalni kompjuteri dopuštaju podešavanje rezolucije ekrana tako da se obezbede sve boje (true color), ili da se utvrdi boja desktopa, ekrana, ikonica i naslovnih linija. U nekim slučajevima možete da definišete i sopstvene boje.

Video tehnologija se zasniva na činjenici da se svaka boja ljudskom oku može predstaviti pomoću prikladne kombinacije osnovnih boja, crvene, zelene i plave (RGB). Ekran monitora sadrži tri fosfora,* po jedan za svaku osnovnu boju. Elektronika unutar monitora koristi tri elektronska mlaza, po jedan za svaki fosfor.

* Fosfor je supstanca koja emituje energiju u obliku svetlosti kada se njeni atomi pobude elektronskim mlazom.

Promenom intenziteta svakog mlaza moguće je podešavati emitovanu količinu osnovne boje na svakom fosforu. Moguće je postići doslovno sve boje iz vkllјivog spektra. Suština problema je u kreiranju strukture podataka koja predstavlja odgovarajuću mešavinu za svaki piksel.

Isto kao što se koristi osam bitova za predstavljanje 256 nijansi sive boje, možemo da koristimo 8-bitnu grupu za predstavljanje svake osnovne boje. Intenzitet svakog elektronskog mlaza se podešava u skladu sa 8-bitnom vrednošću koja daje željenu boju. LJ stvari, korišćenje osam bitova za svaku osnovnu boju znači da svaki piksel može da se predstavi pomoću 24 bita, što otvara mogućnost za 2^{24} različitih boja. Pošto Ijudsko oko ne može da razlikuje toliko boja, mislimo da je tru color dovoljan.

Napominjemo da postoji alternativna reprezentacija video slika koja se takode sastoji od tri 8-bitne grupe. Razlika je u tome što jedna grupa predstavlja *osvetljenost* (jarkost), a druge dve *obojenost* (bojuj. Osvetljenost i obe vrednosti boje izračunavaju se na osnovu RGB vrednosti. Na primer, u referenci [Ta03] moguća relacija ie navedena kao

$$Y = 0.30K + 0.59G + 0.11B$$
$$/ = 0.60R\text{-}0.28G\text{-}0.325$$
$$Q = 0.211?\text{-}0.52C + 0.315$$

Slova *R, C* i *B* predstavljaju vrednosti osnovnih boja. Slova *Y, I* i *Q* koristi NTSC* (National Television Standards Committee) za predstavljanje osvetljenosti i vrednosti boje. Postoje i drugi standardi i različite formule koje se tiču ovih veličina, ali one nisu relevantne za našu raspravu. Značajno je to da za svaku RGB vrednost postoji YIQ vrednost i obratno. Ako želite dodatne informacije o vrednostima obojenosti i osvetljenosti, pogledajte referencu [Pe93].

Prednost korišćenja osvetljenosti i obojenosti zasniva se na mogućnostima registrovanja Ijudskim okom, koje nije podjednako osetljivo na sve boje. Naš senzorni sistem je osetljiviji na osvetljenost, nego na obojenost. To znad da manji gubitak u vrednostima obojenosti, do koga dolazi u toku prenosa, možda neće biti mnogo primetan. Ovo je korisna informacija kada jer reč o kompresovanju slika.

(J narednoj raspravi nečemo voditi računa da li su pikseli predstavljeni pomoću RGB, ili YIQ vrednosti. Vodićemo račnna samo da li se svaki piksel može predstaviti sa tri 8-bitne grupe; naš glavni cilj je da redukujemo broj bitova za prenos, ili skladištenje.

U sledećem koraku razmatramo broj piksela na tipičnoj slici. Naravno, ovaj broj varira u zavisnosti od veličine slike. Medutim, da bismo imali neke brojke sa kojima možemo da radimo, pretpostavićemo da slika izlazi sa VGA ekrana veličine 640x480 piksela.t MaIo računice pokazuje da ova slika zahteva 24 640 480 = 7.372.800 bitova. Kako ovo utiče na saobraćaj preko Interneta? Uzimajuđ u obzir da se video često sastoji od 30 slika u sekundi i da postoji veći broj različitih slika koje se istovremeno prenose do različitih korisnika, broj bitova se povecava čak i do gigabitskih razmera.

* Grupa koja definiše standarde za lelevizijske prenose u SAD

t Današnji monitori mogu da prikazuju slike sa većim kvalitetom od 640x480. Osim toga, danas su uobičajene rezolucije 80OxSOO, 1.024x768, 1.152x864 i 1.280x1024.

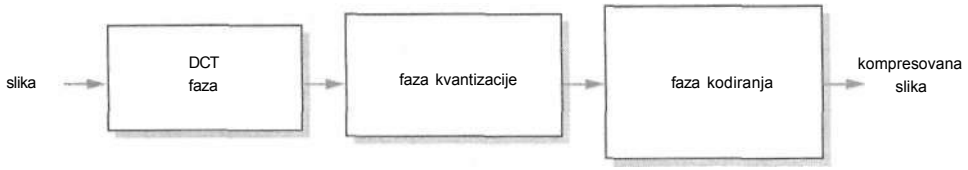
Bez načina za kompresovanje i značajno redukovanje broja bitova, tekuća tehnologija jednostavno ne bi mogla da podrži takav saobraćaj.

JPEG kompresija

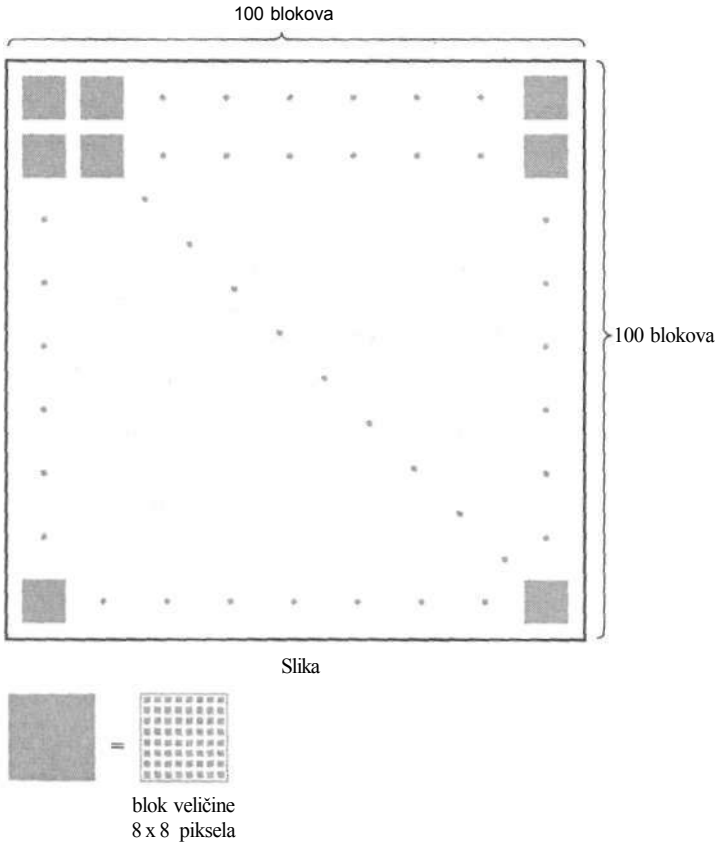
JPEG je akronim za Joint Photographic Experts Group, grupu koja je formirana zajedničkim naporima ISO, ITU i IEC - njen standard za kompresiju, opšte poznat kao **JPEG kompresija,** korišćen je i za crno-bele slike i za slike u koloru. JPEG se značajno razlikuje od prethodno predstavljenih tehnika kompresije. Prethodni metodi predstavljaju primere kompresije bez gubitaka. Odnosno, algoritam za dekompresiju može u potpunosti da "oporavi" informacije koje su ugradene u kompresovani kod. Kod JPEG kompresije postoje gubici: slika koja se dobija nakon dekompresije ne mora da bude identična originalnoj. Gubitak informacija nije prihvatljiv u slučajevima kao što je transfer izvršnih fajlova, ali može da se toleriše ako fajl sadrži sliku. Razlog je ograničenost ljudskog čula vida. Činjenica je da ljudsko oko ne može uvek da registruje suptilne razlike u bojama. Ovde može da se napravi poređenje sa uzorcima boja u farbari. Neko može da se dvoumi satima izmedu nekoliko uzoraka boja, a za njegovog partnera sve te boje mogu da izgledaju identično.

JPEG kompresija se sastoji iz tri faze (slika 5.8): diskretne kosinusne transformacije (DTC - discrete cosine transform), kvantizacije i faze kodiranja. Druga i treća faza su sasvim jednostavne, a prva je prilično složena. Najveći deo teorije je zasnovan na matematičkim zakonima i zahteva dobro predznanje iz matematike i poznavanje Furijeovih redova i diskretne kosinusne transformacije. Nećemo se baviti teorijskom razradom ovih tema; ako imate dobro predznanje \z matematike, možete da pogledate reference [Fe92] i [Ra90]. Ipak, pretpostavićemo neke jednačine, predstaviti nekoliko primera i objasniti kako i zašto se te jednačine mogu primeniti na slike.

DCT faza JPEG kompresija započinje deljenjem slike u nizove blokova veličine 8 8 piksela. Ako je veličina originalne slike 800x800 piksela, slika bi se sastojala od niza od po 100 blokova, i uzduž i popreko (slika 5.9). Ako je slika crno-bela, svaki piksel može da se predstavi kao 8-bitni **broj.**



SLIKA 5.8 *Tri faze JPEG kompresije*



SLIKA 5.9 *Slika sa 800 800 VCA ekrana podeljena na blokove veličine 8 8 piksela*

Svaki blok može da se predstavi kao dvodimenzionalni niz koji sadrži osam redova i osam kolona. Elementi niza su 8-bitni celi brojevi, od 0 do 255, zaključno. Faza diskretne kosinusne transformacije je primenjena na ovaj niz.

Ako je slika u boji, onda se svaki piksel predstavlja sa 24 bita, ili tri 8-bitne grupe (koje predstavljaju ili RGB, ili YIQ vrednosti; ovde nije bitno koje se od te dve vrednosti koriste). Ovaj blok od 8x8 piksela možemo da predstavimo koristeći tri dvodimenzionalna niza, sa po osam redova i osam kolona. Svaki niz predstavlja vrednosti piksela iz jedne od tri 8-bitne grupe. Diskretna kosinusna transformadja je primenjena na svaki niz.

Sada dolazimo do onoga što diskretna kosinusna transformacija, u stvari, radi. U osnovi, to je funkcija koja uzima dvodimenzionalni niz sa osam redova i kolona i kreira drugi dvodimenzionalni niz, takode sa osam redova i kolona. Na primer, ako P predstavlja niz vrednosti piksela, gde P[x] [y] predstavlja vrednost u redu *x* i koloni *y,* diskretna kosinusna transformacijadefiniše novi nizT, gde[T]i *[j]* predstavlja vrednost u i-tom redu, u; koloni na sledeći način:*

$$T[i][j]=0.25C(i)C(j)\sum_{x=0}^{7}\sum_{y=0}^{7}P[x][y]\cos\left(\frac{(2x+1)i\pi}{16}\right)\cos\left(\frac{(2x+1)j\pi}{16}\right) \quad (5\text{-}1)$$

za i = 0, 1, 2, ..., 7 ij = 0, 1, 2, ..., 7 i gde važi

$$c(i)=\begin{cases} 1/\sqrt{2} \text{ if } i = 0 \\ 1 \text{ suprotnom} \end{cases}$$

Ovde nećemo izvoditi ovu formulu. llmesto toga, pokušajte da shvatite šta ova formula radi nad matricom P i pod kojim okolnostima daje dobru kompresiju. Rezultujuča matrica T sadrži kolekciju vrednosti koje se nazivaju prostorne učestalosti (spatial frequency). Il suštini, ove učestalosti se tiču broja promena vrednosti piksela u funkciji pozicije piksela u bloku. Vrednost T[O] [0] se označava kao *DC koeficijent* i predstavlja prosečnu vrednost niza *P* (kada su *i* i *j* jednaki 0, kosinusi su 1). Ostale vrednosti u T nazivaju se *AC koeficijenti.* Kada *i* i*j* imaju veće vrednosti, vrednosti piksela mogu da predstavljaju umnoške kosinusnih funkcija sa većim učestalostima.

Zašto je ovo značajno? Pretpostavimo da su sve vrednosti u P iste. To bi odgovaralo slici koja sadrži samo jednu boju, bez ikakvih varijacija. U tom slučaju, svi AC koeficijenti odgovaraju sumi kosinusnih funkcija, koje se medusobno poništavaju (jer se vrednosti niza P mogu faktorisati sumiranjima). RezuUat je to da su svi AC koeficijenti jednaki 0. Ako postoje male varijacije u vrednostima u nizu P_1 onda će mnogi AC koeficijenti, mada ne svi, biti jednaki 0. Ako postoji veliki broj varijacija u vrednostima niza P, samo nekoliko AC koeflcijenata će imati vrednost 0.

Na slici 5.10 prikazani su rezultati primene diskretne kosinusne transformacije na dva različita niza. Prvi niz (slika 5.10a) sadrži P vrednosti koje se menjaju uniformno. Ovo bi odgovaralo slici sa uniformnim promenama boje i manjim količinom finih detalja. U ovom slučaju u nizu T mnogi AC koeficijenti su jednaki 0. Primećujete kako AC koeficijenti u opštem slučaju postaju manji dok se udaljavaju od gornje leve pozicije u nizu. Vrednosti koje su dalje od te pozicije odgovaraju većim prostornim učestalostima, ili finijim detaljima na slici. Pošto ova određena slika ima malo finih detalja, ove vrednosti su male i bliske 0.

U drugom slučaju (slika 5.10b) P vrednosti se često menjaju. Ovo odgovara slici sa većim brojem različitih boja u manjoj oblasti. Samim tim, na slici postoji više finih detalja. U tom slučaju su AC koeficijenti različiti od nule.

[+] Diskrctna kosinusna transformacija ima opštiju definiciju, u zavisnosti od veličine niza na koji se primenjuje. Pošto smo mi uzeli nizove dimenzija 8 8, ova formula odgovara našim potrebama.

P niz								*T* niz (vrednosti su zaokružene na najbliži celi broj)							
20	30	40	50	60	70	80	90	720	-182	0	-19	0	-6	0	-1
30	40	50	60	70	80	90	100	-182	0	0	0	0	0	0	0
40	50	60	70	80	90	100	110	0	0	0	0	0	0	0	0
50	60	70	80	90	100	110	120	-19	0	0	0	0	0	0	0
60	70	80	90	100	110	120	130	0	0	0	0	0	0	0	0
70	80	90	100	110	120	130	140	-6	0	0	0	0	0	0	0
80	90	100	110	120	130	140	150	0	0	0	0	0	0	0	0
90	100	110	120	130	140	150	160	-1	0	0	0	0	0	0	0

(a) P niz predstavlja male promene na slici.

P niz								*T* niz (vrednosti su zaokružene na najbliži celi broj)							
100	150	50	100	100	150	200	120	835	15	-17	59	5	-56	69	-38
200	10	110	20	200	120	30	120	46	-60	-36	11	14	-60	-71	110
10	200	130	30	200	20	150	50	-32	-9	130	105	-37	81	-17	24
100	10	90	190	120	200	10	100	59	-3	27	-12	30	28	-27	-48
10	200	200	120	90	190	20	200	50	-71	-24	-56	-40	-36	67	-189
150	120	20	200	150	70	10	100	-23	-18	4	54	-66	152	-61	35
200	30	150	10	10	120	190	10	2	13	-37	-53	15	-80	-185	-62
120	120	50	100	10	190	10	120	32	-14	52	-93	-210	-48	-76	80

(b) P niz predstavlja velike promene na slici.

SLIKA 5.10 *Rezultati diskretne kosinusne Iransformacije primenjene na dva različita niza*

U opštem slučaju, ako češće dolazi do promena vrednosti na slici i manje uniformno u funkciji njihove pozicije, AC koeficijenti odgovaraju vrednostima sa većim amplitudama. Osim toga, veći broj viših prostornih učestalosti dobija nenulte vrednosti. LI suštini, AC vrednosti predstavljaju manje varijacije piksela. Zato se slike sa mnoštvom finih detalja teže kompresuju od slika sa neznatnim varijacijama boje.

Nema velike koristi od prostornih učestalosti ako ne postoji način da se upotrebe za obnavljanje originalnih vrednosti piksela. U stvari, postoji inverzna formula koja konvertuje prostorne učestalosti nazad u vrednosti piksela.

$$P[i][j]=0.25C(i)C(j)\sum_{x=0}^{7}\sum_{y=0}^{7}T[x][y]\cos\left(\frac{(2x+1)i\pi}{16}\right)\cos\left(\frac{(2x+1)j\pi}{16}\right) \quad (5\text{-}2)$$

Ovde nečemo proveravati, niti dokazivati ovu tvrdnju. Ostavljamo Vama da vežbate pisanje programa koji primenjuje jednadnu 5.2 na prostorne učestalosti sa slike 5.10. Ako to tačno uradite, izračunaćete vrednosti originalnih piksela.

Faza kvantizacije Faza kvantizacije obezbeduje način za ignorisanje malih promena na slici, koje možda neče biti primetne. Ona defmiše još jedan dvodimenzionalni niz (neka se zove Q) koji se dobija deljenjem vrednosti niza Tnekim brojem, pa zaokruživanjem na najbliži celi broj. Na orimer. nretnostavimo da ie

$$\boldsymbol{T} = \begin{vmatrix} 152 & 0 & -48 & 0 & -8 & 0 & -7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -48 & -0 & 38 & 0 & -3 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -8 & 0 & -3 & 0 & 13 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -7 & 0 & 2 & 0 & -1 & 0 & 7 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \qquad \textbf{(5-3)}$$

Ako se svaka vrednost podeli sa IO i zaokruži na najbliži celi broj, imaćemo

$$\boldsymbol{Q} = \begin{vmatrix} 15 & 0 & -5 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -5 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \qquad \textbf{(5-4)}$$

Ovo smo uradili da bismo dobili sledeći niz sa manje različitih brojeva i da bi se dobili konzistentniji uzorci. Na primer, niz Q ima više nula i zato če se kompresovati bolje nego T. Naravno, ovo nameće logično pitanje kako da se vratimo sa Q na T radi dekompresije. Odgovor je jednostavan: ne možemo da se vratimo. Deljenjem vrednosti niza T i zaokruživanjem gubimo originalne informacije. Ako pokušamo da izvedemo operacije u suprotnom smeru i ako pomnožimo Q sa 10, dobićemo

$$\boldsymbol{T} = \begin{vmatrix} 150 & 0 & -50 & 0 & -10 & 0 & -10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -50 & 0 & 40 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 0 & 0 & 10 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \qquad \textbf{(5-5)}$$

Ne postoji način da znamo da dve vrednosti -1Ou koloni 1, u stvari, odgovaraju vrednostima -8 i-7 iz niza Tu jednačini 5.5 Da smo primenili jednačinu 5.2 na niziz jednačine 5.5, dobili bismo vrednosti piksela koje predstavljaju aproksimacije originalnih vrednosti. Drugim rečima, izgubili bismo nešto boje. Ipak, ako su gubici mali, ne moraju da budu primetni.

U praksi, deljenje vrednosti niza T nekom konstantnom vrednošću nije praktično i često izaziva velike gubitke. U stvari, mi pokušavamo da sačuvamo što više informacija iz gornjeg levog dela niza, jer one predstavljaju niže prostorne učestalosti - odgovaraju manje suptilnim karakteristikama slike koje bi se odmah uočile ako bi se promenile. Vrednosti u donjem desnom uglu predstavljaju finije detalje, čije promene možda neće biti mnogo uočljive. Zato se obično defmiše niz kvantizacije (nazovimo ga *U)* sa manjim vrednostima u gornjem levom delu i većim vrednostima u donjem desnom delu. Zatim se Q definiše pomoću formule

$$Q[i][j]=\text{Round}(T[i][j]/U[i][j]) \text{ za } i = 0, 1, 2, \ldots, 7 \text{ i } j = 0, 1, 2, \ldots, 7$$

gde je Round funkcija koja vrsi zaokruživanje na najbhži celi broj.

Na primer, ako koristimo niz T iz jednačine 5.3 i

$$\boldsymbol{U} = \begin{vmatrix} 1 & 3 & 5 & 7 & 9 & n & 13 & 15 \\ 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & ? & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 13 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & .23 & 25 & 27 & 29 \end{vmatrix} \quad \textbf{(5-6)}$$

kvantizacija se dobija kao

$$\boldsymbol{Q} = \begin{vmatrix} 152 & 0 & -10 & 0 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \quad \textbf{(5-7)}$$

Obrtanjem procesa i množenjem svakog elementa u Q sa odgovarajućim elementom u U dobija se

$$\boldsymbol{T} = \begin{vmatrix} 152 & 0 & -50 & 0 & -9 & 0 & -13 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -50 & 0 & 36 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -9 & 0 & 0 & 0 & 17 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix} \quad \textbf{(5-8)}$$

Ovo je i dalje aproksimacija originalnog niza T iz jednačine 5.3, ali u opštem slučaju dovodi do manjih gubitaka u oblasti niza sa nižim prostornim učestalostima, tako da će se sačuvati uočljiviji aspekti originalne slike. Možda postoje veći gubici u oblastima sa finijim detaljima, ali oni će biti manje primetni. Osim toga, niz kvantizacije Q ima značajan stepen redundanse, tako da je moguća bolja kompresija.

Napominjemo da JPEG ne propisuje sadržaj niza U. Sadržaj obično umnogome zavisi od konkretne primene; uloženi su veliki napori da se pronađe U koje omogućava kompresiju sa minimalnim gubicima.

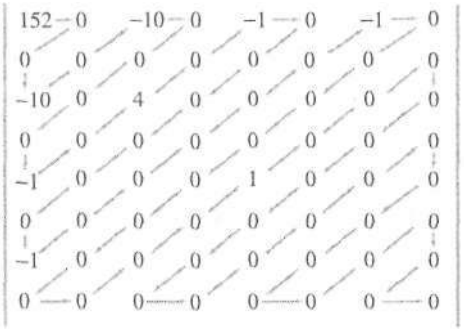
Faza kodiranja Veći deo dosadašnje rasprave o JPEG kompresiji uključivao je složene transformacije i rezultate kvantizacije. Još uvek se nismo bavili kompresijom. Ona se konačno izvodi u fazi kodiranja. Glavnu funkciju faze kodiranja predstavljaju linearizacija dvodimenzionalnih podataka iz Q niza i kompresovanje tih podataka radi prenosa. Logičan pristup može da podrazumeva linearizaciju koja se postiže prenosom jednog reda u jednom trenutku. Sa svim nulama koje se javljaju, možemo da koristimo run-length]<odiranje. Iako će ovo sigurno funkcionisati, postoji bolji način.

Na slici 5,11 ilustrovano je kako se mogu urediti elementi niza Q iz jednačine 5.7. Ako su elementi niza uredeni tako da se šalje po jedan red u jednom trenutku, počevši od prvog reda, imamo sledeće nizove 0 (sa dužinom većom od 8):

- Niz dužine 9 iz redova 1 i 2
- Niz dužine 13 iz redova 3 i 4
- Niz dužine 15 iz redova 7 i 8

Medutim, ako uredimo elemente kao što je naznačeno strelicama na slici, imaćemo sledeće nizove (sa dužinom većom od 8):

- Niz dužine 11
- Niz dužine 24



SLIKA 5.11 *Redosled po kome su preneti elementi niza*

Kao što možete da vidite, postoji mnogo duži niz kod drugog redosleda. Ne nameravamo da to iskoristimo kao formalni dokaz da je ovaj metod linearizacije elemenata optimalan. Sa druge strane, videli smo praćenjem strelica sa slike 5.11 da elementi koji predstavljaju više prostorne učestalosti teže mogu da se grupišu. To nije slučaj kada se prenose po jedan red, ili po jedna kolona u jednom trenut]<u. Pošto niz kvantizacije U često ima veće vrednosti u oblastima sa većom prostornom učestalošću, veća je verovatnoća da će vrednosti kvantizacije biti jednake O. Grupisanjem vrednosti sa većom prostornom učestalošću dobijamo veću verovatnoću za formiranje dugačkih nizova sa nulama, što omogućava bolju kompresiju.

JPEG može da koristi neki tip Hafmanovog koda, ili aritmetičkog koda za nenulte vrednosti u slučajevima kada DTC faza i faza kvantizacije daju odredene nenulte vrednosti sa većom učestalošću. Konačno, pošto nizovi dimenzija 8 8 predstavljaju samo mali deo slike, mora da se prenese veliki broj nizova. Na mnogim slikama sukcesivni nizovi predstavljaju susedne blokove koji se malo razlikuju. Postoji čak i mogućnost da se prenose razlike izmedu nenultih elemenata, umesto da se prenose sami elementi, što uvodi dodatnu dimenziju u kompresiju.

U suštini, JPEG kompresija je složena; stepen Irampresije umnogome zavisi od same slike **i** niza kvantizacije. Niz kvantizacije nije propisan, već zavisi od aplikacije. Pod određenim uslovima, JPEG može da stvori koeficijent kompresije 20:1, ili, čak, i bolji (što znači da velidna prenetog fajla iznosi pet odsto od veličine originala). Mogući su i veći koeficijenti, ali su tada gubici uočljiviji.

Trenutno se radi na još jednom standardu (JPEG 2000), koji je zasnovan na tehnologiji elementarnih talasa. Kao i obično, detalji standarda su složeni, ali tehnologija elementarnih talasa je slična Furijeovim redovima po tome što se funkcija rastavlja na sastavne komponente. Za više informacija o tehnologiji elementarnih talasa pogledajte referencu SaOO . Naravno, rasprava o JPEG ltompresiji može da se vodi sa mnogo više detalja; ako ste zainteresovani za takve detalje, možete da pogledate reference [SaOO], [HaOl], [Ho97] i [He96], ili možete da posetite sajt www.jpeg.org/.

GIF failovi

Odeljak završavamo kratkim predstavljanjem još jednog formata za slike koji se zove GIF **(Graphics Interchange Format).** Dok je JPEG dizajniran za rad sa slikama u punom koloru (224 različitih boja), GIF redukuje broj boja na 256. U osnovi, 256 boja se smešta u tabelu, uz pokušaj da se što vernije pokrije opseg boja sa slike. Zatim, 24-bitna vrednost piksela se menja 8-bitnim indeksom za zapise tabele sa bojama koje najviše odgovaraju originalnoj boji. Rezultujuće vrednosti bita se, nakon toga, izlažu varijaciji Lempel-Ziv kodiranja radi kompresije.

GIF fajlovi imaju velike gubitke ako se premaši granica od 256 boja, a ispod te granice nema gubitaka. Ovaj format je obično najprikladniji za grafičke sadržaje sa relativno malim brojem boja i sa relativno oštrim granicama izmedu boja. Tu se ubrajaju stripovi, grafikoni i crteži olovkom. GIF format nije prildadan za slike sa mnoštvom varijacija i senčenja, tj. za slike sa fotografskim kvalitetom.

5.7 Kompresovanje multimedijalnih informacija

MPEG

Pošto smo opisali kompresovanje nepokretnih slika, logično je da predemo na video, ili pokretne slike. Medutim, pre toga moramo da razumemo kako se postiže kretanje u video klipu. Kretanje, bilo da je reč o velikom ekranu, televiziji, ili video klipu sa CD-ROM-a, ili Interneta na kompjuterskom monitoru, u suštini nije ništa drugo nego brzo prikazivanje nepokretnih slika. Širom sveta koriste se različiti standardi za video, ali najčešće korišćeni standard defmiše NTSC, sa prikazivanjem 30 nepokretnih slika u sekundi. Ovo je dovoljno da se oko zavara, tako da se stvara "predstava" o pravom kretanju. Slike koje se stvaraju sa manjom brzinom prikazivanja daju "iseckano" kretanje, koje je karakteristično za neke stare filmove.

Grupa koja definiše standarde za video kompresiju je **Moving Pictures Expert Group (MPEG).** Kao i JPEG, predstavlja rezultat saradnje izmedu ISO, IEC i ITU. Ljudi često koriste frazu MPEG kompresija kada misle na video kompresiju. Ipak, MPEG nije jedan standard, već je reč o nekoliko standarda.

- MPEG-I, ranije poznat kao ISO/IEC-11172, bio je dizajniran za video i CD-ROM i rane direktne satelitske sisteme za emitovanje.
- MPEG-2, ranije poznat kao ISO/IEC-13818, korišćen je za složenije aplikacije, kao što su multimedijalna okruženja i high-definition televezija (HDTV), a usvojen je u i satelitskim prenosima.
- MPEG-4, ranije poznat kao ISO/IEC-14496, namenjen je za video konferencije preko kanala sa niskim opsegom (postojao je i MPEG-3, koji je originalno bio namenjen za HDTV, ali je HDTV dodat MPEG-2 standardu).
- U vreme dok pišem ovu knjigu priprema se standard MPEG-7, koji treba da podrži široki spektar aplikacija, a zasniva se na pretpostavci da bi multimedijalni podaci mogli da zauzmu sve veći deo propusnog opsega signala u toku ovog veka. MPEG-7 će obezbediti multimedijalne alatke za definisanje i pristup sadržaju i trebalo bi da omogući, na primer, pretraživanje pesama na osnovu "mrmljanja" u mikrofon kompjutera, pretraživanje slika skiciranjem grafike koja podseća na traženu sliku, ili skeniranje logo znaka kompanije i pretraživanje multimedijalne baze podataka reklama da bi se proverilo čiji je logo.
- Takode se priprema i standard MPEG-21. Postoji veliki broj učesnika na polju multimedije i svako od njih može da koristi svoje modele, pravila i procedure, a MPEG-21 bi defmisao zajednički radni okvir koji bi olakšao interakciju i saradnju između različitih grupa.

Na ovom nivou rasprave nećemo se baviti različitim varijacijama MPEG standarda, već ćemo se posvetiti standardu MPEG-I (ubuduće, kada kažemo MPEG, znajte da je reč o ovoj varijaciji standarda). Kao i ranije, ako želite detaljnije informacije od onih koje ovde predstavljamo, možete da pogledate reference [SaOO], [Ho97] i [HaOl]. U stvari, MPEG kompresuje audio **i** video zapise zasebno, a mi ćemo se ovde baviti samo video kompresijom. Opis MPEG audio kodiranja možete da pronadete u referenci [Ho97].

Pošto video predstavlja, u stvari, niz nepokretnih slika, deluje logično da MPEG koristi JPEG kornpresiju, ili njenu varijaciju za kompresovanje svake slike. lako je ovo, u suštini, tačno, korišćenje samo JPEG kompresije za svaku nepokretnu sliku ne obezbeduje dovoljan stepen kompresije za većinu aplikacija. U prethodnom odeljku istakli smo da nepokretna slika može da sadrži 7.372.800 bitova. Sa koeficijentom kompresije 20:1, možemo da redukujerno sliku na 368.640 bitova. Medutim, zapamtite da NTSC video standardi definišu prikazivanje 30 slika u sekundi, tako da bi se u svakoj sekundi moralo preneti 30x368.640 = 11.059.200 bitova. To je mnogo, posebno ako se prenos obavlja preko deljenih kanala koje koriste i drugi korisnici takođe radi pristupa video sadržajima.

Ono što MPEG čini izvodljivim jeste dodatna redundansa (privremena redundantnost), koja se nalazi u sukcesivnim kadrovima. U osnovi, to znači da bez obzira koliko akcije vidite u videu, razlika izmedu dva susedna kadra je obično veoma mala. Čak i popularni akcioni heroji zahtevaju nekoliko sekundi za izvodenje potpune akcije u prostoriji. To je 60 kadrova, a naš junak može da prede samo nekoliko stopa od jednog kadra do drugog. Ako posmatrate scenu bez mnogo akcije, susedni kadrovi mogu da budu skoro identični. Pošto se informacije JPEG kompresije nalaze u jednoj slici, MPEG mora da radi sa privremenom redundansom. Ovim problemom smo se bavili i ranije kada smo predstavili relativno kodiranje. U suštini, ova tehnika šalje osnovni kadar, pa kodira sukcesivne kadrove izračunavanjem razlike (sa malom količinom informacija), a zatim ih kompresuje i prenosi. Prijemna strana može da rekonstruiše kadar na osnovu prvog osnovnog kadra i primljenih razlika.

MPEG ovo izvodi do određene mere, ali, kao što ste mogli da pretpostavite, "stvari" su mnogo složenije. Izračunavanje razlika u odnosu na prethodni kadar funkcioniše dobro za postavljanje likova koji se kreću u Vašem vidokrugu, jer se oni nalaze i u prethodnom kadru. Na primer, potpuno nova scena ne može da se izrazi na ovakav način. Razlika između nove i stare scene je velika, tako da će biti neophodno slanje nove scene. Sledeći primer uključuje objekte koji su bili skriveni u prethodnoj sceni iza osobe koja se kreće. Kako se osoba kreće na sceni, tako se prikazuju objekti koji su ranije bili zaklonjeni.

MPEG identifikuje tri različita tipa kadrova:*

- **I kadar (unutrašnji kadar)** Ovo je samosadržinski kadar koji za sve namene i svrhe predstavlja JPEG-kodiranu sliku.
- P kadar (predviđeni kadar) Ovaj kadar je sličan onome koji smo upravo predstavili, po tome što se kodira izračunavanjem razlika izmedu tekućeg i prethodnog kadra.
- B kadar (bidirekcioni kadar) Sličan je P kadru, osim što je intrepoliran izmedu prethodnog i budućeg kadra (da, ovo zvući malo čudno, ali doći ćemo i do toga).

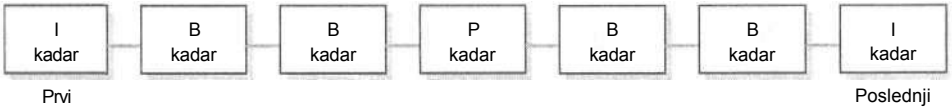
* Ustvari, postoji i četvrti tip kadra, nazvan DC kadar, koji može da se koristi za brzo pretraživanje na uredajima kao što su video rekorderi. Nema nikakvu ulogu u trenutnoj raspravi, tako da ga nećemo objašnjavati.

Potrebno je razmotriti kako i zašto se u jednoj sekvenci javljaju različiti tipovi kadrova i kako se P i B kadrovi konstruišu na osnovu drugih kadrova. Počećemo isticanjem da se I kadar mora javljati periodično u sklopu sekvence kadrova. Postoji nekoliko razloga za to. Kao što smo istakli, izračunavanje razlika vredi ako postoje male razlike izmedu kadrova. Medutim, male razlike iz fiksnog kadra su, obično, "vezane" za kraći vremenski period. Eventualno, scena može da se promeni, ili da u prikaz udu novi objekti. Ako bismo pokušali da sve merimo relativno u odnosu na prvi kadar, imali bismo malo uspeha. Ako se koristi relativno diferenciranje i mere razlike samo između susednih kadrova, onda bi eventualna greška mogla da bude preneta i na sve naredne kadrove. Drugi razlog se tiče emitovanja sadržaja, kada neko može da se uključi u prenos u bilo kom trenutku. Da je korišćeno relativno kodiranje u odnosu na prvi kadar, a Vi se uključite kasnije, ne biste imali sa čim da poredite naredne kadrove. Periodično postavljanje I kadrova osigurava da se razlike mere relativno u odnosu na tekuće scene. Osim toga, ovim je moguće eliminisati propagaciju grešaka.

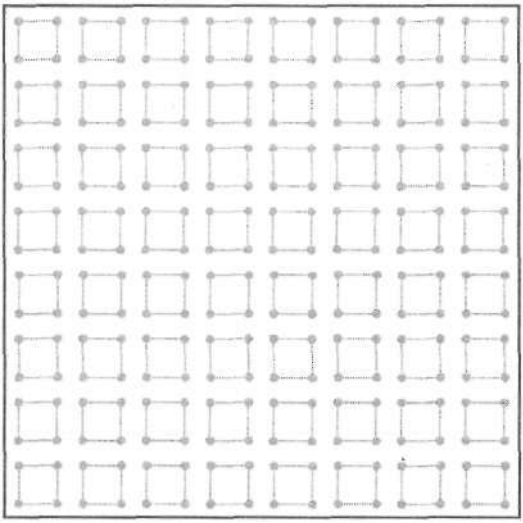
Na slici 5.12 data je tipična MPEG sekvenca kadrova. Izmedu dva I kadra "uglavljeni" su četiri B kadra i jedan P kadar. U opštem slučaju, broj B kadrova može da bude promenljiv, mada obično postoji jedan P kadar izmedu dve grupe B kadrova. P kadar predstavlja, u suštini, razliku izmedu prethodnog I kadra i B kadrova koji su interpolirani izmedu najbližih I i P kadrova. Na primer, prva dva B kadra su interpolirana od prvog I kadra i P kadra. Poslednja dva B kadra su interpolirana od poslednjeg I kadra i P kadra.

Nameće se logično pitanje kako možemo da interpoliramo B kadrove iz kadrova koje još nismo primili. Na slici 5.12 kadrovi su dati onim redosledom kojim se prikazuju, a ne onim kojim se prenose. P kadar će biti poslat pre prva dva B kadra, a drugi I kadar se šalje pre poslednja dva B kadra. P kadar i I kadrovi mogu da se baferuju, a B kadrovi koji će biti primljeni mogu da se dekodiraju na strani na kojoj će se prikazivati.

P kadrovi su kodirani pomoću metoda poznatog pod nazivom *motion-compensated prediction,* koji se zasniva na konceptima specificiranim u ITU preporuci H.261. Funkcioniše deljenjem slike na kolekciju makroblokova, koji sadrže po 256 piksela (16 po horizontali i 16 po vertikali). Uz pretpostavku da svaki piksel ima jednu vrednost za osvetijenje i dve za obojenost, makroblok može da se predstavi sa tri niza dimenzija 16x16. Da bi se "stvari" ubrzale, dva niza za obojenost se, u stvari, redukuju na nizove dimenzija 8x8. Na slici 5.13 prikazano je kako je to izvedeno. Niz dimenzija 16x16 je prikazan kao kolekdja nizova 2x2 sa vrednostima obojenosti. Prosečna vrednost svakog skupa od četiri vrednosti za obojenost menja četiri vrednosti i rezultat je niz 8x8. Naglašavamo još jednom da postoje odredeni gubici u ovom procesu, ali oni često nisu primetni.



SLIKA 5.12 *Tipična MPEG sekvenca*

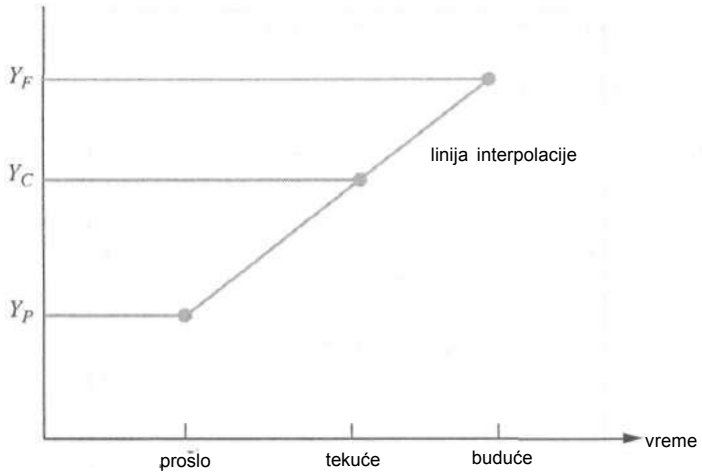


SLIKA 5.13 *Redukcija niza obojenosli dimenzija 16 16 na niz dimenzija 8 8*

Pre slanja P kadra, algoritam proučava svaki makroblok i locira najpodudarniji makroblok u prethodnom I kadru. On ne mora obavezno da bude na istoj relativnoj poziciji,* jer pretpostavljamo da se slike pomeraju od jednog kadra do drugog. MPEG ne defmiše šta se smatra najboljim poklapanjem; ono zavisi od konkretne aplikacije. Kada se pronade najpodudamiji makroblok, algoritam izračunava razlike izmedu makroblokova i izračunava vektor kretanja *{motion vector)* - u suštini, horizontalni i vertikalni pomeraj, koji se pamti zajedno sa razlikama. Ovo se izvodi za sve makroblokove u kadru, a rezultati su kodirani i preneti na sličan način kao u slučaju JPEG kompresije. Na prijemnoj strani prilikom dekodiranja prenete razlike se koriste za rekonstruisanje makioblokova, a vektor kretanja se koristi za utvrđivanje njihove pozicije u kadru.

Dekodiranje B kadra je slično, samo što se makroblokovi interpoliraju iz podudamih makroblokova u premodnom i budućem kadru. *Interpolađja* je način predviđanja vrednosti na osnovu dve postojeće vrednosti i obično je tema kojom se bavi numerička analiza. Na slici 5.14 ilustrovano je ono što pod tim podrazurnevamo. Ako horizontalna linija predstavlja vreme, a znamo vrednosti nekih veličina ($Y_{„}$ i Y_{l}:, respektivno) u nekim proteklim i budućim trenucima, možemo da procenimo vrednost (Y_c) u tekućem trenuuku. Jedan način podrazumeva crtanje linije koja povezuje tačke Yp i Y_F, i izračunavanje y koordinate te linije u tekucfem trenuuku. Ovo je primer lineame interpoladje. Postoje razni načini za izvođenje interpolacije, ali to je posebna tema.

* Ovo ne znači da algoritam pretražuje celi kadar tražed najbolje poklapanje. Obićno se traži makroblok na istoj relativnoj poziciji i u susednim pozicijama. Tako se ubrzava algoritam kodiranja.



SLIKA 5.14 *Korišćenje interpolacije za procenu vrednosti*

Ovde je značajno što se i prošli i budud kadrovi mogu pretraživati radi pronalaženja poklapanja makroblokova, pri čemu se izračunavaju razlike i vektori kretanja za svaki pronadeni makioblok. Vrednost makrobloka i njegova tačna postavka u budućem kadru mogu da se interpoliraju iz podudarnih makroblokova i vektora kretanja. Kao što smo prethodno istakli, ovo je posebno korisno kada ne postoji dobro poklapanje za makroblok iz prethodnog kadra, a može da se desi kada se u prikazu pojave novi objekti. Međutim, taj objekat će se verovatno nalaziti i u budućem kadru i ta informacija može da se iskoristi za izračunavanje i postavljanje makrobloka u tekućem kadru.

Potrebno je napomenuti da, i pored standarda, MPEG kodiranje i dekodiranje zahtevaju značajnu procesorsku snagu. Pronalaženje poklapanja i izračunavanje vektora poklapanja za veći broj bitova nije zanemarljivo. Ipak, kod mnogih multimedijalnih aplikacija video zapis se snima samo jednom i smešta na neki medijum. Pošto se ovo izvodi pre prikazivanja, vreme potrebno za kodiranje nije mnogo bitno. Međutim, video se često prikazuje u realnom vremenu. Dekodiranje mora da se izvede brzo, ili će kretanje delovati "iseckano". Pogledajte neke multimedijalne aplikacije na nekom starijem kompjuteru i verovatno ćete primetiti da slike "ne teku glatko" kao kod pravog videa.

Brži procesor definitivno može da pomogne, ali CPU obavlja još mnogo drugih "stvari", pa to nije dugoročnije rešenje za multimedijalne aplikacije, koje beleže veoma brz razvoj. Jedan od najznačajnijih uspeha u poslednjih nekoliko godina bila je MMX tehnologija, koja je ugrađena u Pentium čipove za multimedijalne aplikacije. Ona omogućava dekodiranje aplikacija kao što je MPEG (u stvari, MPEG-2), tako da se postiže značajno poboljšanje performansi. Opisivanje MMX tehnologije je predmet kursa o računarskoj arhitekturi, ali, zbog njenog uticaja na MPEG-2 aplikacije, vredi je ovde pomenuti. Više detalja o MMX tehnologiji i njenom uticaju na izvodenje multimedijalnih sadržaja možete da pronadete u referenci [Pe97].

MP3

Poglavlje o kompresiji završavamo pregledom tehnologije koja je uključena u neke high-profile novinske "priče". Tehnologija je MP3, protokol za kompresovanje audio zapisa. Napominjemo da MP3 sam za sebe nije kontraverzan. Jednostavno, definiše način za kompresovanje audio fajlova. Kontroverznu crtu unose nelicendrano korišćenje i slobodna razmena MP3 kodirane muzike. Raspravu o kontroverznom aspekatu ostavljamo drugima, a mi ćemo dati opšti pregled MP3 tehnologije.

Pre nego što zađemo u dublje razmatranje, navešćemo neke osnovne informacije o audio zapisima. U odeljku 3.6 opisana je tehnika za konvertovanje analognih signala u digitalne pod nazivom PCM (impulsna kodna modulacija). Predstavili smo frekvencije semplovanja i način kako su povezane sa frekventnim opsegom originalnog audio signala. Ono što tada nismo pomenuli jesu ograničenja Ijudskog čula sluha. U opštem slučaju, većina ljudi može da čuje samo zvukove čije se frekvencije nalaze izmedu 20 Hz i 20 kHz. Naše čulo sluha jednostavno ne može da "uhvati" frekvencije izvan tog opsega. Zbog toga, uobičajena PCM tehnika za kreiranje CD-kvalitetnih zvukova koristi 16-bitne semplove i frekvenciju semplovanja od 44,1 kHz. Prema Nikvistovoj teoremi, ovo je dovoljno da se signal iz našeg čujnog opsega kompletno rekonstruiše.

Malo računice pokazuje da jedna sekunda PCM-kodirane muzike zahteva 16x44.1x1000 « 700.000 bitova, i to samo za jedan kanal. Za dvokanalni stereo vrednost se udvostručava na 1,4 megabita. Tako bi 2-minutni zapis zahtevao otprilike 1,4 megabita x 120 sekundi «168 megabita. Ne samo da bi ovo bio ogrornan fajl, već bi morao i da se prenese bitskom brzinom od 1,4 Mbps da bi mogao da se reprodukuje kao zvuk. Moguća su dva načina za redukovanje ukupnog broja bitova: redukovanje broja bitova po semplu i redukovanje frekvencije semplovanja. Nažalost, oba pristupa odgovaraju manje preciznoj digitalizaciji originalnog analognog signala, tako da bi se izgubilo na kvalitetu zvuka. Tu nastupa kompresija.

MP3 se, u stvari, odnosi na MPEG sloja 3 za audio kompresiju, a usvojen je kao ISO/IEC standard 1992. godine. Kako ova postavka predlaže, MPEG dopušta tri razlidta sloja audio kompresije, Slojevi se razlikuju po složenosti kodiranja, koeficijentima kompresije i rezultujućem kvalitetu zvuka, i to na sledeći način:

- Sloj 1 daje koefidjent kompresije od oko 4:1, a zvuk može da se reprodukuje na bitskoj brzini od 192 Kbps za svaki kanal.
- SIoj 2 daje koeficijent kompresije od oko 8:1 i dizajniran je za bitske brzine od 128 Kbps po kanalu.
- Sloj 3 (MP3) daje koeficijent kompresije od oko 12:1 i prikladan je za bitske brzine od oko 64 Kbps po kanalu.

Kako MP3, u stvari, funkcioniše? Nažalost, ne možemo da damo kompletan odgovor na to pitanje, jer su u celu "priču" uključene neke veoma sofisticirane tehnike. Ipak, daćemo opšti pregled glavnih koncepata. Najveći deo onoga što je dizajnirano u MP3 zasniva se na psihoakustičnom modelu. Psihoakustika je, u osnovi, studija o ljudskom čulu sluha i može da identifikuje šta možemo da čujemo i koje zvuke možemo da razlikujemo.

Kao što smo ranije istakli, u opštem slučaju čujemo zvuke iz opsega od 20 Hz do 20 kHz. Sledeći problem je koliko dobro možemo da razlikujemo razlidte zvukove iz tog opsega.

Većina nas može da registruje razliku, na primer, između kontrabasa i flaute, jer postoji velika razlika u frekvencijama zvukova koje proizvode. Medutim, model ukazuje da kada frekvencije dva signala postanu veoma bliske 1 judsko čulo sluha neće mod da ih razlikuje. Na primer, većina ljudi ne može da razlikuje zvuk od 2.000 Hz od zvuka od 2.001 Hz.

Sledeći problem je čujno maskiranje **(auditory masking).** Ako je jedan zvuk na odredenoj frekvenciji bio izuzetno jak, a drugi slične frekvencije je bio slab, možda nećemo moći da čujemo slabiji zvuk. Ako ovo uporedimo sa ćulom vida, to je kao kada ne uspevate da vidite mali tamni objekat koji se nalazi ispred jakog svetla. Isključite svetlo i taj objekat se prikazuje. Razlog za to je činjenica da je signal jarkog svetla toliko jak da nadjačava reflektovano svetlo od manjeg objekta.

U audio svetu kao primer može da posluži sviranje klavira, kada se jako pritisne jedna dirka i zbog glasnog zvuka ne uspevamo da registrujemo lagani pritisak na susednu dirku. Glasniji tonovi nadjačavaju tiše. Možda ovo objašnjava zašto retko možete da sretne radnika za pneumatskim čekićom koji sluša klasičnu muziku. Osnovna ideja na kojoj se MP3 zasniva je da se "hvata" audio signal, utvrduje se šta ne može da se čuje, uklanjaju se te komponente iz audio zapisa i vrši se digitalizacija preostalog dela signala. Ukratko, uldanja se sve što ne može da se čuje.

Prvi korak u MP3 kompresiji je kodiranje po podopsezima *(subband coding),* koje je prikazano na slici 5.15. To znači da se audio zapis uvodi u psihoakustični model i **banku filtera.** Banka fdtera je kolekcija fdtera koji kreiraju nizove koji predstavljaju komponente signala u okviru specifičnog frekventnog opsega. Postoji po jedan filter za razne frekventne opsege; zajedno, oni rastavljaju originalni signal na podopsege, sa različitim, nepreklapajućim frekventnim opsezima. Teorija banki filtera i kodiranja po podopsezima pnpada veoma visokom nivou i zahteva složena izračunavanja slična onima kod diskretne kosinusne transformacije, predstavljene u odeljku 5.6. Nećemo se baviti time, ali ako ste zainteresovani, pogledajte referencu [SaOO] u kojoj možete da pronađete detaljan opis kodiranja po podopsezima. Glavna ideja je da se formira niz bitova za svaki opseg signala.

Međutim, zbog čujnog maskiranja, podopsezi neče biti kodirani na potpuno isti način. Ako su signali u jednom podopsegu veoma glasni, potrebna je dobra rezolucija njihovih amplituda.

SLIKA 5.15 ***MP3 kodiranje***

Drugim rečima, potrebno je više bitova. Ako su signali u susednim podopsezima slabi i ako ih maskiraju glasniji signali, potrebna je manja rezolucija za te slabije signale - mogu se kodirati sa manje bitova. Ovde se uklapa psihoakustični model. On analizira audio zapis i utvrduje pragove maskiranja. Proces je složen, jer zavisi od mapiranja signala u reprezentaciju frekventnog domena, pri čemu se koriste Furijeove transformacije. Ipak, svrha je da se utvrdi koje frekvencije dominiraju, a koje neće biti čujne. Ako odredene frekvencije nisu u čujnom opsegu, nema potrebe da se taj deo signala rekonstruiše, tako da je moguće koristiti manje bitova. Ovim je postignuta kompresija bez uočljivog gubitka u kvalitetu zvuka. Na kraju se svaki podopseg kodira korišćenjem različitog broja bitova.

Tabela 5.11: Pregled tehnika kompresije

Tehnika **kompresije**	**Tip** korišćene redundanse	**Kako se vrši kompresija**
Hafmanov kod	Odredeni karakten se češće javljaju od ostalih.	Koriste se kraći uzorci bitova za češće korišćena slova i duži za ona koja se ne koriste često.
Run-length kodiranje	Podaci sadrže dugačke stringove istog karaktera, ili bita.	Dugački niz odredenog bita, ili karaktera se menja sa dužinom tog niza.
Faksimil kompresija	Traže se i dugački stringovi istog bita i učestalost sa kojom se specifični stringovi javljaju.	Linija se deli na nizove belih i crnih piksela. Svaki niz se kodira korišćenjem modifikovanog Hafmanovog algoritma.
Relativno kodiranje	Izmedu dva susedna dela podataka mogu da postoje veoma male razlike.	Kodiraju se manje razlike izmedu uzastopnih okvira, umesto da se kodiraju konkretni okviri.
Lempel-Ziv kodiranje	Stringovi odredenih karaktera se javljaju češće od ostalih.	Pojave stringa koji se ponavlja menjaju se generisanim kodovima.
JPEG	Manji delovi slike često sadrže manju količinu detalja.	Kompresovanje nepokretnih slika se izvodi primenom diskretne kosinusne transformacije na blokove veličine 8x8 piksela, rezultati se kvantifikuju i kvatifikovani koeficijenti učestalosti se kodiraju.
MPEG	Susedni kadrovi često sadrže skoro identične scene.	Koriste se metodi slični JPEG kompresiji, ali se koristi prednost redundantnosti između susednih kadrova za kompresiju medukadrova izračunavanjem razlika u susednim kadrovima i korišćenjem tehnika za predvidanje kretanja.
MP3 (MPEG audio kompresija sloja 3)	Komponente signala se maskiraju drugim, moćnijim signalima, ili se odredene komponente nalaze van čujnog opsega.	Koriste se složeni psihoakustični model i banke filtera za utvrdivanje delova audio signala koji neće biti čujni i koji se uklanjaju.

Konačno, signali iz podopsega ulaze u fazu kvantizacije. Kao što je opisano u odeljku za JPEG kompresiju, faza kvantizacije redukuje ukupan broj vrednosti u svakom podopsegu i nastoji da eliminiše male razlike koje ne bi trebalo da budu opažene. Na kvantifikovane vrednosti može da se primeni Hafmanov kod i kreiraće se MP3 zapis.

Ovo razmatranje nije bilo preterano detaljno, jer je za to neophodno dosta matematike, zajedno sa teorijom o kodiranju po podopsezima i psihoakustičnom modelu. Ako Vas interesuje teorija, obavezno pogledajte referencu [SaOO].

5.8 Zaključak

Ovde predstavljene tehnike kompresije predstavljaju šeme koje se koriste u praksi. U referencama [SaOO], [Ho97], [Fe97] i [He96] pominju se i druge tehnike, a reference [Ho97], [SaOO] i [He96] u potpunosti su posvećene tehnikama kompresije (i predstavljaju dobro štivo za one koji su ozbiljno zainteresovani za ovu temu). Zapamtite da su tehnike kompresije dizajnirane za različite tipove prenosa. U tabeli 5.11 dat je kratak pregled onih kojima smo se bavili u ovom poglavlju.

Pitanja i zadaci za proveru

1. Šta je Hafmanov kod?
2. Šta je no-prefix svojstvo Hafmanovog koda?
3. Šta je frekventno-zavisni kod?
4. Šta se podrazumeva pod run-length kodiranjem?
5. Da li su sledeće tvrdnje tačne, ili netačne (zašto)?
   a. Hafmanov algoritam može efikasno da kompresuje tekstualni fajl sa nasumično rasporedenim karakterima.
   b. Algoritam Lempel-Ziv kompresije može efikasno da kompresuje veliki fajl u kome se nalazi izvorni kod programa.
   c. Metod kompresije koji funkcioniše dobro za jedan tip fajla često dobro funkcioniše i kod ostalih tipova fajlova.
   d. Metodi kompresije bi uvek trebalo da kompresuju fajl bez gubljenja informacija iz fajla.
   e. Kada se primeni na fajl, metod kompresije može da kreira veći fajl od originalnog.
   f. Koeficijenti kompresije za faksimil kompresiju su različiti u zavisnosti od slike koja se kompresuje.
   g. MP3 i JPEG kompresije su slične po tome što ne mogu da rekonstruišu originalni fajl.
6. Šta je aritmetička kompresija?
7. Koja je svrha zaključnog karaktera u aritmetičkoj kompresiji?
8. Šta je relativno kodiranje?

9. Šta je Lempel-Ziv kodiranje?
10. Šta je faksimil kompresija?
11. I Lempel-Ziv i Hafmanov algoritam su slični po tome što koriste ponavljanja. Po čemu se razlikuju?
12. Koje su glavne razlike izmedu JPEG i MPEG metoda kompresije?
13. Navedite razlike izmedu 1,PiB kadrova u kontekstu MPEG kodiranja.
14. Šta je MP3 fajl?
15. Šta je psihoakustični model?
16. Sta se podrazumeva pod čujnim maskiranjem? Kako ono može da pomogne kompresovanje zvučnih fajlova?
17. Koje tehnike kompresije u ovom poglavlju nemaju gubitke informacija? Koje gube podatke?

Vezbe

1. Možete li da izmislite 4-bitni kod sličan onome iz tabele 5.1?
2. Izvedite Hafmanov kod za slova čija je učestalost pojavljivanja data u sledećoj tabeli.

Slovo	Učestalost (0M)
A	15
B	25
C	20
D	10
E	10
F	20

Bez konstruisanja, koliko različitih Hafmanovih kodova može da se kreira?

3. Kompletirajte tabelu 5.5 i pronađite konačni interval za string iz tog primera.
4. Opišite probleme koji se javljaju ako frekventno-zavisni kod promenljive dužine, kao što je Hafmanov, nema no-prefix svojstvo.
5. Kompresujte sledeći niz bitova korišćenjem run-length kodiranja. Koristite pet bitova za kodiranje svakog dugačkog niza. Izrazi u zagradama označavaju nizove.

   1 (33 nule) 1 (25 nula) 1 1 1 (44 nule) 1 (2 nule) 1 (45 nula)

   Izrazite dužinu kompresovanog niza kao procenat dužine originalnog niza.
6. Pomoću run-length kodiranja, koliko nula mora da se javi u nizu da bi kod izvršio kompresiju?
7. Navedite primer situacije u kojoj bi run-length kodiranje bilo bolje (ili gore) od Hafmanovog koda.

8, Komentarišite sledeću konstataciju:

U oblasti megabitskih, ili gigabitskih prenosa šeme kompresije štede samo najmanje delove sekunde. Zbog toga, vreme koje se štedi nije vredno dodatnog kompresovanja bitova,

9. lskoristite Hafmanov kod iz tabele 5.3 i interpretirajte sledeći niz bitova (počevši od krajnjeglevogbita).

110011100100010001111O110

10. Koji su od sledećih kodova Hafmanovi? Zašto?

Karakter	Kod	Karakter	Kod	Karakter	Kod
A	01	A	10	A	1
B	001	B	001	B	01
C	10	C	11	C	000
D	110	D	101	D	001
E	010	E	000	E	0001

11. Kompletirajte korake aritmetičkog kodiranja započetog u tabeli 5.5.
12. Primenite aritmetičku kompresiju na string DCBED, koristeći verovatnoće definisane u tabeli 5.4. Koji se realni broj može koristiti za kompresovanje stringa?
13. Ponovite prethodnu vežbu, ali pretpostavite da su verovatnoće 0.15 (A), 0.25 (B), 0.2 (C), 0.1 (D) i 0.3 (E).
14. Pretpostavite da imate 10 karaktera sa podjednakom verovatnoćom pojavljivanja. Možete li da predložite jednostavniji način za implementiranje aritmetičkog kodiranja?
15. Na osnovu verovatnoća iz tabek 5.4, procenite kom stringu odgovara realna vrednost 0.45734? Pretpostavite da string ima pet karaktera.
16. Postoje oprečna mišljenja da li se aritmetička kompresija koristi za utvrđivanje jednog realnog broja između O i 1. Zato mogu da se koriste standardni formati za skladištenje brojeva u pokretnom zarezu i kompresovani kod uvek ima isti broj bitova. Sta mislite o tome?
17. Ako se koristi faks kompresija, kako izgleda kompresovani kod za niz od 1.300 belih piksela, a kako za 1.300 crnih piksela?
18. Popunite detalje specifične za programski jezik i implementirajte Lempel-Ziv algoritme na slici 5.7.
19. Kako bi se izveo algoritam sa slike 5.7 da smo koristili kompletan alfabet?
20. Kompletirajte tabelu 5.8.
21. Izvršite algoritme Lempel-Ziv kompresije i dekompresije, počevši sa sledećim stringom. Kreirajte tabele slične tabelama 5.8, 5.9 i 5.10.

BBABAABBAABACBACBACBABAAABAA

22. Napišite program koji primenjuje jednačinu 5.2 na prostorne ućestalosti za slike 5.10a i 5.10b.

23. Izvršite diskretnu kosinusnu transformaciju (jednačina 5.1) na sledeći niz piksela. Trebalo bi da napišete program koji izvodi potrebna izračunavanja.

Reference

[Cr90] Crichton, M. *Jurassic Park.* New York: Ballantine Books, 1990.

[DrOl] Drozdeck, A. *Data Structures and Algorithms in C++.* Pacific Grove, CA: Brooks/Cole, 2001.

[Fe92] Feig, E., and S. Winograd. "Fast Algorithms for Discrete CosineTransformations". *IEEE Transactions on Signal Processing,* vol. 40 (September 1992), 2174-2193.

[Fe97] Fernandez, J. *MIME, UUENCODE and ZIP: Decompressing and Decoding Internet Files.* New York: MIS Press, 1997.

[HaOl] Halsall, F. *Multimedia Communications.* Reading, MA: Addison-Wesley, 2001.

[He96] HeId, G. *Data and Image Compression,* 4th ed. New York: Wiley, 1996.

[Ho97] Hoffman, R. *Data Compression in Digital Systems.* New York: Chapman and Hall, 1997.

[Hu52] Huffman, D. "A Method for the Construction of Minimum Redundancy Codes". *IRE Proceedings,* vol. 40 (September 1952), 1098-1101.

[Pe93] Pennebaker, W. B and J.L. Mitchell, JPEG Still Image Data Compression Standard. New York: Van Nostrand Reinhold, 1993.

[Pe97] Peleg, A, S. Wilkie, and U. Weiser. "Intel MMX for Multimedia PCs". Communications of the ACM, vol. 40, no. 1 (January 1997), 25-38.

[Ra90] Rao, K.R. and P. Yip. *Discrete Cosine Transform: Algorithms, Advantages, Applications.* Boston: Academic Press, 1990.

[SaOO] Sayood, K. *Introduction to Data Compression,* 2nd ed. San Francisco: Morgan Kaufman, 2000.

[St03] Stallings, W. *Computer Organization and Architecture.* 6th ed. Englewood Cliffs, NJ: Prentice-Hall, 2003.

[Ta03] Tanenbaum, A. S. *Computer N'etutorks.* 4th ed. Englevrood Cliffs, NJ: Prentice-Hall, 2003.

[We84] Welch, T. "ATechnique for High-Performance Data Compression". *Computer,* vol. 17, no. 6 (May 1984), 8-19.

POGLAVLJE

6 Integritet podataka

lako je tačno da su današnja deca izložena većoj količini informacija i raznovrsnijim iskustvima nego njihovi vršnjaci u prošlosti, to ne znači automatski da su sofisticiranija. Uvek znamo više nego što razumemo i sa gomilom informacija koje su predstavljene mladitn Ijudima praznina izmedu znanja i razumevanja i između iskustva i učenja postaje veća nego što je bila u prošlosti.

—**David Elkind** (1709-1784), američki psiholog

6.1 Uvod

U prethodnim poglavljima su opisani mehanizmi koji su neophodni za smeštanje i prenos informacija. Svi ti metodi, bez obzira koliko su sofisticirani, ne mogu da garantuju efikasne i sigume komunikacije. Uzmimo za primer sledeću poruku koja je preneta elektronskom poštom:

Your brothel in New Orleans needs money. *(Vašem bordelu u Nju Orleansu je neophodan novac.)*

Vaša reakcija može da bude raznolika, od zbunjenosti do krajnjeg užasa. Šta ako Vaša supruga prva vidi ovu poruku? Ona ne zna da je originalna poruka glasila:

Your brother in New Orleans needs money. *(Vašem bratu iz Nju Orleansa je neophodan novac.)*

Šta se desilo? Poslata poruka je bila krajnje bezazlena, a problem je nastao u toku prenosa. Slovo r iz reči *brother* je bilo ASCII kodirano kao 1110010. Nažalost, zbog nekih električnih smetnji, došlo je do promene četiri srednja bita 1001 u 0110, tako da je primljena kombinacija 1101100, što predstavlja kod za slovo *l*.

Mislim da je nesporno da sistem koji dopušta isporučivanje ovakvo izmenjenih poruka nije poželjan. Međutim, činjenica je da se greške javljaju. Svaka poruka koja se prenosi elektronskim putem je podložna smetnjama. Jaka sunčeva svetlost, električni udari, fluktuacije u napajanju, ili udarac ašovom u kabl mogu da izazovu neverovatne i nepredvidljive "stvari" u toku prenosa. Međutim, ne možemo da dopustimo da, na primer, astronauti prime netačne instrukcije za navigadju, ili da neka švajcarska banka na neki račun deponuje milion dolara više nego što je

Mogućnost detektovanja promena u toku prenosa naziva se **detekcija grešaka.** U većini slučajeva, kada se greške detektuju, poruka se odbacuje, obaveštava se pošiljalac i poruka se ponovo šalje. Naravno, nema nikakvih garancija da ponovo neće doći do oštećenja poruke i zato je neophodno razviti protokole koji omogućavaju razmenu poruka (bez obzira na njihov status) između pošiljaoca i primaoca. U stvari, nema nikakvih garancija da neće biti, oštećene i same informacije o statusu poruke. Ovo zahteva razradu detaljnih protokola, ali o njima će biti više reči u Poglavlju 8.

Ponovno slanje poruke ponekad nije praktično. Možda nema dovoljno vremena, kao u slučaju real-time aplikacija. Jedan primer su sonde za svemirska istraživanja. Mogu da proteknu sati dok se ne prenesu značajni telemetrijski podaci iz udaljene tačke u svemiru. Osim toga, ako je sonda veoma daleko, signali mogu da budu veoma slabi i, samim tim, podložni smetnjama. Dok sonda primi zahtev za ponovno slanje, može da se pomeri na neku drugu tačku sa koje originalni podaci više ne mogu da se pribave. Osim toga, verovatnoća da će doći do oštećenja signala je veoma visoka kod prenosa na velikim udaljenostima i velika je verovatnoća da poruka nikada neće biti primljena bez neke greške. Sledeći primer je gledanje, ili slušanje multimedijalnih zapisa u realnom vremenu. Ako gledate video i nešto se desi u nekoliko kadrova, nije praktično da se vraćate nazad i ponovo gledate te kadrove. To bi bilo kao da gledate film na DVD-ju i neko neprestano premotava film unazad. U nekim slučajevima, kada se greška detektuje, moguće ju je ispraviti bez ponovnog prenosa. Ovo se naziva korekcija grešaka. Pošiljalac nikada neće saznati da je došlo do oštećenja poruke i da je ona ispravljena. Krajnji zaključak je da se poruke eventualno mogu tačno isporučiti.

U ovom poglavlju se bavimo integritetom podataka, mogućnošću da se utvrdi kada je došlo do njihovog oštećenja. U odeljku 6.2 predstavićemo proveru parnosti, metod za detektovanje grešaka u određenim bitovima. To je jednostavan i prilično naivan pristup i, mada se obično ne implementira samostalno, ipak igra važnu ulogu u nekim složenijim šemama. U odeljku 6.3 upoznaćete cikličnu proveru redundantnosti, komplikovani metod koji se zasniva na interpretiranju nizova bitova kao polinoma, koji se nakon prijema dele da bi se utvrdilo da li postoje greške. Videćete da je ovaj metod izuzetno tačan i da, i pored složenih izračunavanja, može efikasno da se implementira. Osim toga, ovo je često korišćeni metod za detektovanje grešaka.

Konačno, u odeljku 6.4 predstavićemo metod za korekciju grešaka pod nazivom Hamingov kod. To je metod koji koristi višestruke provere parnosti na takav način da se greška, ako je došlo do promene jednog bita, detektuje u jedinstvenoj kombinaciji provere parnosti. Kada se utvrdi lokacija na kojoj je došlo do greške, bit može da se promeni i na taj nadn je poruka ispravljena.

6.2 Jednostavne tehnike za detekciju grešaka

Provera parnosti

Tehnike za detektovanje grešaka zahtevaju slanje dodatnih bitova čije vrednosti zavise od podataka koji su poslati. Ako se podaci promene, vrednosti dodatnih bitova neće odgovarati novim podacima (barne teorijski). Verovatno najčešće korišćeni pristup je provera parnosti, koja uključuje brojanje bitova sa vrednošću 1, a zatim se postavlja jedan dodatni bit, tako da ukupan broj bitova sa vrednošću 1 bude paran **(parna parnost),** ili neparan **(neparna parnost).**

Dodatni bit se naziva bit pamosti. Našu raspravu zasnivamo na parnoj parnosti, a ćitaoci mogu da konstruišu analognu diskusiju za neparnu parnost.

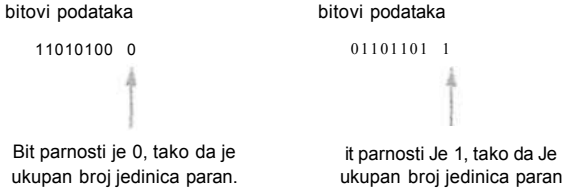
Ilustracije radi, pretpostavimo da je ukupan broj bitova sa vrednošću 1 neparan. Ako se bit pamosti defrniše kao 1, ukupan broj 1 bitova je sada paran. Slično tome, ako je ukupan broj 1 bio paran, bit parnosti je 0. Razmotrite nizove bitova sa slike 6.1. Prvi niz ima četiri 1 bita. Zato je bit parnosti O. Drugi niz ima pet 1 bitova, pa je njegov bit parnosti 1.

ANALIZA PROVERE PARNOSTI Provera parnosti detektuje samo greške u jednom bitu. Bit parnosti je prenet zajedno sa bitovima podataka, a prijemnik proverava parnost. Ako prijemnik pronade neparan broj bitova, došlo je do greške. Ipak, greške u samo jednom bitu su veoma retke kod električnih prenosa. Na primer, pretpostavimo da je došlo do greške zbog kratkog strujnog udara, ili statičkog elektriciteta, čije je trajanje izraženo u hiljaditim delovima sekunde. Sa ljudskog stanovišta, to je skoro neprimetno. Ali, ako su podaci prenošeni brzinom od 1 Mbps, u tom hiljaditom delu sekunde 10.000 bitova je izloženo nastalim smetnjama. Kada je oštećeno više bitova, kažemo da je došlo do navalne greške (burst error).

Kako provera parnosti funkcioniše u slučaju proizvoljnih navalnih grešaka? Pretpostavimo da se u toku prenosa menjaju vrednosti dva bita. Ako su oba bila 0, menjaju se u 1. Dve dodatne jedinice i dalje u zbiru daju paran broj jedinica. Slično tome, da su oba bila 1, promenili bi se u 0, tako da bi postojale dve jedinice manje, ali bi ukupan broj i dalje bio paran. Da su bitovi imali suprotne vrednosti i promenili ih, i dalje bi imali suprotne vrednosti. I ovoga puta ukupan broj jedinica ostaje isti. Krajnji zaključak je da provera parnosti ne može da detektuje dvostruke greške.

U opštem slučaju, provera parnosti može da detektuje greške koje se javljaju na nepamom broju bitova. Ako se promeni vrednost parnog broja bitova, provera parnosti neće moći da detektuje grešku. Zaključak je da provera parnosti detektuje greške sa verovatnocom od 50 odsto za navalne greške, ali to nije dovoljan procenat za komunikacione mreže.

Da li je zbog ovoga provera parnosti beskorisna? Odgovor je negativan, i to zbog dva razloga. Prvo, neke organizacije kompjuterske memorije srneštaju bitove iz bajta, ili reči na različitim čipovima. Tako se prilikom pristupa reči koriste različite putanje. U takvim slučajevima smetnje na jednoj putanji mogu da izazovu grešku u jednom bitu. Takve arhitekture često koriste dodatne memorijske čipove za proveru parnosti (detaljnija rasprava o njima ne može da se uklopi u previđeni obim ovog teksta). Drugi razlog je to što provera parnosti predstavlja osnovu za tehniku korekcije grešaka, koju ćemo predstaviti u odeljku 6.4.



SLIKA 6.1 *Detektovanje grešaka u jednom bitu pomoću provere parnosti*

Čeksume

Sledeći pristup deli sve bitove podataka u 32-bitne* grupe i svaku tretira kao celobrojnu vrednost. Te vrednosti **se,** zatim, sabiraju, tako da daju čeksumu (checksum). Svaki prenos u sabiranju koji bi zahtevao više od 32 bita se ignoriše. Zatim, proces koristi mod 232 vrednosti sume. Dodatna 32 bita predstavljaju čeksumu, koja se dodaje podacima pre nego što se pošalju. Prijemni uređaj deli rezultat sa onim što je smešteno u dodatna 32 bita. Ako se rezultati ne poklapaju, došlo je do greške.

Ovaj pristup je efikasniji od jednostavne provere parnosti i detektuje sve vrste navalnih grešaka, jer nasumična promena brojeva obično menja vrednost njihove sume. Ipak, ovaj pristup ne detektuje sve greške. Jednostavan primer je greška koja izaziva da se jedna od 32-bitnih vrednosti poveća za određeni iznos, a neka dmga 32-bitna vrednost se smanji za isti iznos. Suma tih vrednosti ostaje nepromenjena, i pored toga što je došlo do greške.

6.3 Detekcija grešaka pomoću ciklične provere redundantnosti

U ovom odeljku upoznaćete metod poznat pod nazivom **ciklična provera redundantnosti** (CRC-cyclic redundancy check), koji je mnogo tačniji i od provere parnosti i od metoda koji koristi čeksume. Osim toga, pokazaćemo kako se može efikasno implementirati.

CRC je neuobičajen, ali "parnetan" metod koji proveru grešaka izvodi deljenjem **polinoma.** Vaša prva reakcija je verovatno: "U kakvoj je vezi deljenje polinoma sa prenosom niza bitova?". U stvari, metod interpretira svaki niz bitova kao polinom. U opštem slučaju, niz bitova

$$b_{n-1}b_{n-2}b_{n-3}\ldots b_2b_1b_0$$

interpretira se kao polinom

$$b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + b_{n-3}x^{n-3} + \ldots + b_2x^2 + b_1x + b_0$$

Na primer, niz bitova 10010101110 interpretira se kao

$$x^{10} + x^7 + x^5 + x^3 + x^2 + X^1$$

Pošto je svaki koeficijent b, ili 0, ili 1, pišemo *x′* kada je *b,* jednako 1, a ne zapisujemo ih kada je *b′* jednako 0.

U sledećim koracima opisan je CRC metod. Pretpostavljamo da je kod svih izračunavanja korišćen mod 2.

1. Na dati niz bitova dodajte nekoliko nula na kraj niza (kasnije ćemo reći koliko i zašto) i nazovite ga *B.* Neka *B(x)* bude polinom koji odgovara nizu *B.*
2. *B(x)* se deli sa nekim ugovorenim polinomom *C(x)* **(generator polinoma)** i utvrđuje se ostatak *R(x).*

* Može da se korisle i grupe od osam, ili 16 bitova.

3. Defmiše se *T(x)* = B(x) - RfxJ. Kasnije ćemo pokazati da *T(x)/G(x)* generiše ostatak 0 i da se oduzimanje može izvesti zamenom prethodno dodatih 0 bitova nizom bitova koji odgovara polinomu *R(x).*

4. Prenosi se T, niz stringova koji odgovara polinomu *T(x).*

5. Neka *T* predstavlja niz bitova koje primalac dobija i neka je *T(x)* pridruženi polinom. Primalac deli *T(x)* sa *C(x).* Ako je ostatak O, primalaczaključuje da je T = *T* i da nije bilo grešaka. U suprotnom, on zaključuje da je došlo do greške i zahteva ponovni prenos.

Pre nego što počnete da očajavate, moramo da damo odgovore na neka pitanja. Zašto izvršavamo sve ove korake? Da li postoji ikakva validacija donošenja zaključaka u prijemniku nako deljenja *T'(x)* sa *G(x)l* Koliko je ovaj metod tačan? Da li i pošiljalac i primalac moraju da produ kroz sve ove korake prilikom slanja svakog okvira? Medutim, ne možemo da damo odgovore na ova pitanja bez nekoliko preliminarnih objašnjenja. Pretpostavljamo da posedujete predznanja o polinomima i njihovim operacijama korišćenjem realnih brojeva, ali ćemo, ipak, dati kratko objašnjenje o deljenju polinoma po modulu 2.

Deljenje polinoma

Na slici 6.2 prikazan je primer deljenja polinoma *T(x)/G(x),* gde je

$$T(x) = x^{10} + x^9 + x^7 + x^5 + x^4$$

i

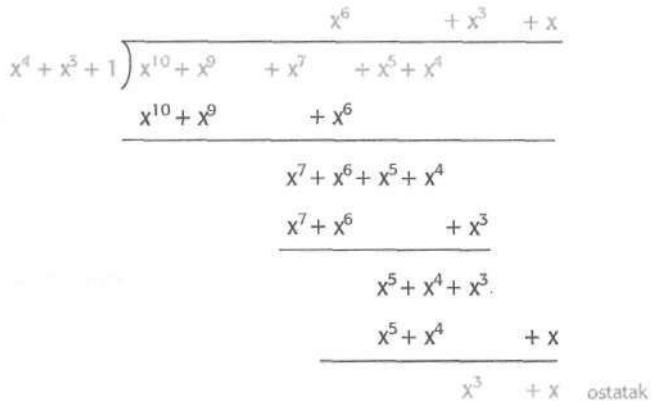
$$G(x) = X^4 + x^3 + 1$$

Ovo je slično običnom deljenju polinoma koje poznajete iz algebre, osim što se koristi aritmetika modula 2. Sabiranje i oduzimanje po modulu 2 definiše se na sledeći način:

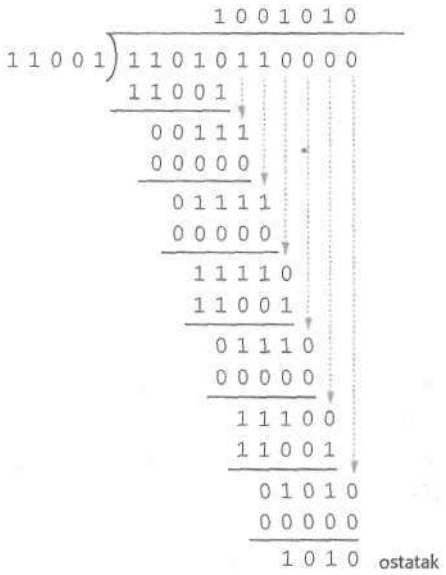
$0 + 0 = 0$ $1 + 0 = 1$ $0 + 1 = 1$ $1 + 1 = 0$

i

$0 - 0 = 0$ $1 - 0 = 1$ $0 - 1 = 1$ $1 - 1 = 0$



SLIKA 6.2 *Izmčunavanje (xlO + x9 + x7 + x5 + x4)/(x4 + x3 + 1)*



SLIKA 6.3 *Sintetičko deljenje (xlO + x9 + x7 + x5 + x4)/(x4 + x3 + 1)*

Primećujete da su sabiranje i oduzimanje po modulu 2 identični operaciji isključivo ILI (exdusive OR). Ovo je značajna činjenica koju ćemo koristiti kasnije u raspravi o implementaciji CRC-a.

Na slici 6.3 prikazano je sintetičko deljenje istih polinoma. Možda se sećate iz algebre da postoji jedna prečica koja koristi samo koefkijente polinoma (u ovom slučaju nizove bitova). Zapamtite da treba da koristite nule na mestima gde nedostaju članovi polinoma, tako da je 11010110000 lista koeficijenata za polinom $x^{10} + x^9 + x^7 + x^5 + X^4$, a 11001 za polinom $X^4 + x^3 + 1$.

Način kako CRC funkcioniše

Pogledajte sada kako CRC funkcioniše. Pretpostavimo da želite da pošaljete niz bitova 1101011, a da je generator polinoma *G(x) - X*^{4} + r^3 + *1;* kasnije ćemo predstaviti neke kriterijume za izbor *G(x).*

1. Dodajte nule na kraj niza. Broj 0 je isti kao i stepen generatora polinoma (u ovom slučaju 4). Tako dobijamo niz 11010110000.

2. Podelite *B(x)* sa *G(x).* Na slikama 6.2 i 6.3 pokazan ju rezultat za ovaj primer, sa ostatkom *R(x) = X*^{3} + *x,* ili bitskim ekvivalentom 1010. Napomenimo da ovo može da se predstavi jednačinom

$$\frac{B(x)}{G(x)} = Q(x) + \frac{B(x)}{G(x)}$$

gde Q(x) predstavlja količinik. Ekvivalentno, možemo da zapišemo

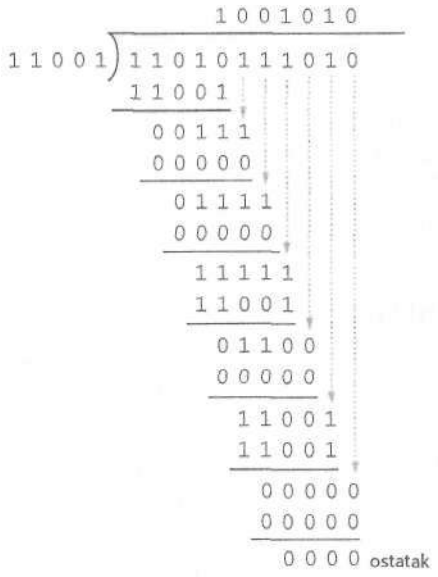
$B(x) = G(x) \times Q(x) + R(x)$

3. Definišite $T(x) = B(x) - R(x)$. Pošto oduzimanje uzima razlike izmedu koeficijenata odgovarajućih članova, razliku izračunavamo oduzimanjem bitova koji su pridruženi svakom polinomu. U ovom slučaju imamo

```
11010110000      niz bitova B (bit string B)
    −1010        niz bitova R (bit string R)
11010111010      niz bitova T (bit string T)
```

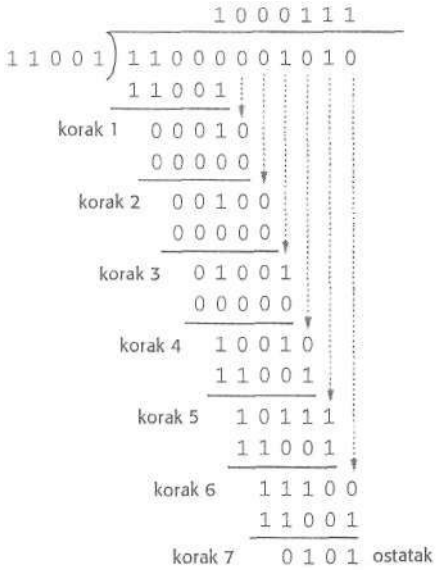
Primećujete da je niz T, u stvari, isti kao i string *B* sa dodatim nulama zamenjenim sa *R*. Sledeća važna čmjenica, koja je pokazana na slici 6.4, je da dobijamo ostatak O ako podelimo *T(x)* sa GCxJ,.* Nakon toga, pošiljalac prenosi niz T.

4. Ako niz T stigne bez grešaka, deljenje sa *C(x)* će dati ostatak O. Ali, pretpostavimo da je niz T oštećen u toku prenosa. Na primer, neka su neki bitovi na sredini promenjeni u 0, tako da je primljen niz 11000001010. Primalac vrši sintetičko deljenje sa *G(x)* i ostatak je različit od 0 (slika 6.5). Pošto je ostatak različit od 0, primalac zaključuje da je došlo do greške.



SLIKA 6.4 *Deljenje T(x) sa* G(Y)

* Postoji analogija sa korišćenjem celih brojeva - ako su *p\ q* celi brojevi i ako je *r* cclobrojni ostatak dobijen deljenjem *p* sa *q,* onda je *p - r* deljivo sa *q.* Na primer, 8/3 generiše oslalak 2, a S - 2 je deljivo sa 3.



SLIKA 6.5 *Deljenje primljenog polinoma sa G(x)*

(Napomena: Ovo nije isto kao da kažemo da deljenje oštećenog niza sa *G(x)* uvek daje ostatak različit od O. To može da se desi, ali, ako je *G(x)* pažljivo odabran, to će se desiti veoma retko. Ovo je sledeća tema koju ćemo obraditi.)

Analiza CRC-a

Mehanizam CRC-a je sasvim jednostavan. Još uvek nismo odgovorili na pitanje ima li ikakve koristi od ovog metoda. Da li će primalac raoći da detektuje oštećeni okvir? Oslanjamo se na pretpostavku da će deljenje oštećenog okvira sa *G(x)* dati ostatak različit od nule. Ali, da li je to uvek tačno? Da li je moguće da se bitovi u nizu *T* promene tako da posle deljenja sa *G(x)* ostatak bude nula?

Kompletan i detaljan dokaz zahteva poznavanje svojstava faktorizacije prstena polinoma (oblast apstraktne matematike) i ovde se nećemo baviti time. Umesto toga, sledi kraća rasprava koja će Vam pomoći da steknete osećaj za način na koji funkcioniše. Za početak, definišimo preciznije šta tražimo. Promena bitova u nizu *T* je analogna sabiranju nekog nepoznatog polinoma sa *T(x).* Dakle, ako *T* predstavlja primljeni niz i ako je *T(x)* pridruženi polinom, onda je *T(x) = T(x) + E(x),* gde je *E(x)* nepoznat primaocu niza *T.* U prethodnom primeru

niz T= 11010111010 odgovara polinomuT(x) = $x^{10} + x^9 + x^7 + x^5 + x^4 + x^3 + x$

niz *E* = 00010110000 odgovara polinomu E(x) = $x^7 + x^5 + x^4$

niz *T* = 11000001010 odgovara polinomu T(x) = $x^{10} + x^{J} + x^3 + x$

Ne zaboravite da se sabiranje izvodi pomoću operacije isključivo ILI. Na primer, sabiranje članova x^7 iz *E(x)* i *T(x)* daje $x^7 + x^7 = (1 + 1) \times x^7 = 0$.

Moramo da damo odgovor na pitanje kada *T(x)* + *E(x)/G(x)* daje ostatak nula. Pošto je (T(X) + *E(x))/G(x)* = *T(x)/G(x)* + *E(x)/G(x)* i pošto prvi član daje ostatak nula, drugi član određuje ostatak. Zbog toga, postavljeno pitanje može da se preformuliše tako da glasi za koje polinome *E(x)* deljenje *E(x)/G(x)* daje ostatak nula.

Sada možemo da tvrdimo sledeće:

Nedetektovane greške u toku prenosa odgovaraju greškama za koje je *G(x)* faktor polinoma E(x).

Sledeče pitanje je pod kojim je uslovima *G(x)* faktor polinoma *E(x).* Proučićemo najpre najjednostavniji primer, gde se rnenja samo jedan bit u nizu T. U ovom slučaju *E(x)* ima samo jedan član - X^k za neki celi broj *k.* Jedini način da *G(x)* bude faktor X^k je da je *G(x)* jednako *x* podignut na neki stepen. Sve dok biramo *G(x)* sa najmanje dva člana, to neće biti moguće. Tako će CRC moći da detektuje sve jednostruke greške.

Zatim, razmotrimo navalnu grešku dužine *k≤r* = stepen *G(x).** Pretpostavimo da je polinom *T(x)* predstavljen kao

$$t_n t_{n-1} \ldots \underbrace{t_{f+fc} t_{jtj+fc.2} \bullet \bullet \ldots t_t t_j}_{k} \ldots t^{\wedge} t_g$$

k bitovi na koje smetnje utiču (k affected bits)

a $t_{i+k-1} \ldots i\ t_i$ su prvi i poslednji bit koji će biti oštećeni. Bitovi izmedu ova dva bita se proizvoljno menjaju. To znači da je

$$E(x) = x^{i+k-1} + \ldots + x^i = x^i \times (x^{k-1} + \ldots + 1)$$

Zato je

$$\frac{E(x)}{G(x)} = \frac{x^i \times (x^{k-1} + \ldots + 1)}{G(x)}$$

Pretpostavimo sada da je *G(x)* izabrano tako da x nije faktor *G(x).* Zbog toga, *G(x)* i x^i iz prethodnog razlomka nemaju zajedničke faktore. Ako je *G(x)* faktor brojioca, onda mora da bude faktor $(x^{k-1} \ldots + 1)$, Pošto smo izabrali $k < r$, onda je $k - 1 < r$ i *G(x)* ne može da bude faktor polinoma sa manjim stepenom.

Zato donosimo sledeći zaključak:

Ako *x* nije faktor *G(x),* onda se detektuju sve navalne greške koje imaju dužinu manju, ili jednaku stepenu *G(x).*

Razmotrite sledeću navalnu grešku bilo koje dužine u kojoj je oštećen neparan broj bitova. Pošto £(x) ima član za svaki oštećeni bit, sadrži neparan broj članova. Zbog toga, E(I) (isključivo ILI neparnog broja jedinica) daje 1. Sa druge strane, pretpostavimo da je $x + 1$ faktor polinoma *G(x).* U tom slučaju, možemo da zapišemo *G(x) = (x + 1)* x *H(x),* gde je *H(x)* neki izraz.

* Stepen polinoma je najveći eksponent za *x.*

Pogledajte sada šta se dešava ako se pretpostavi da su se javile neke nedetektovane greške. Sećate se da nedetektovana greška znači da je *G(x)* faktor polinoma *E(x).* Zamenom *G(x)* sa *(x* + 1) x *H(x)* dobija se da je *E(x) = (x* + 1) x *H(x)* x *K(x).* Sada, ako se ova jednadna izračuna za Jc = 1, faktor *x* + 1 izjednačava E(IJ sa 1.

Jasno je da oboje ne može da se javi istovremeno. Ako i dalje pretpostavljamo da je *x* + 1 faktor polinoma *C(x),* onda pretpostavka o nedetektovanoj grešci koja oštećuje neparan broj bitova nije opravdana. Drugim rečima:

> Ako je *x* + 1 faktor *G(x),* onda su sve navalne greške koje oštećuju neparan broj bitova detektovane.

Poslednji slučaj koji razmatramo je navalna greška sa dužinom većom od stepena polinoma *G(x).* Na osnovu prethodne rasprave zaključujemo da je

$$\frac{E(x)}{G(x)} = \frac{x^j \times (x^{k-1} + \ldots + 1)}{G(x)}$$

Međutim, pošto ovoga puta pretpostavljamo da je *k* > 1 = r = stepen *G(x),* moguće je da je *G(x)* faktor $(x^{k-1} + \ldots + 1)$. Kolike su šanse da će se ovo desiti? Razmotrimo najpre slučaj *k* - 1 = r. Pošto je stepen polinoma G(xJ takode *r,* onda činjenica da je *G(x)* faktor $(x^r + \ldots + 1)$ znači da je #(xj = $(x^r + \ldots + 1)$. Sada đanovi izmedu x^r i 1 defmišu bitove na kojima je došlo do grešaka. Pošto postoji *r* - 1 takvih đanova, postoji 2^{r-1} 1 mogućih kombinacija oštećenih bitova. Ako pretpostavimo da se sve kombinacije javljaju sa podjednakom verovatnoćom, postoji verovatnoća od $1/2^{r-1}$ da kombinacije tačno odgovaraju članovima polinoma *G(x).* Drugim rečima, verovatnoća da neke greške neće biti detektovane iznosi $1/2^{r-1}$.

Slučaj za *k* - 1 > *r* je složeniji i ovde ga nećemo analizirati. Medutim, moguće je pokazati da verovatnoća da neke greške neće biti detektovane iznosi $1/2^r$. U referencama [Pe72] i [Mo89] možete da pronadete detaljniju analizu kodova za detekciju grešaka.

CRC se široko koristi u lokalnim mrežama (LAN mrežama), gde postoje standardni polinomi za *GIx),* poput sledećih:

CRC-12: $x^{12} + x'' + X^3 + x^2 + x + 1$

CRC-16: $x'^s + x^{,5} + x^2 +]$

CRC-ITU: $x^{16} + x^n + X^5 + 1$

CRC-32: $X^{32} + x^{26} + x^{2i} + x^{22} + x^{>6} + x^n + x''$
$+ x''^{>} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

U opštem slučaju, CRC je veoma efikasan ako se G(x) ispravno izabere. Specifično, G(x) može da se izabere tako da *x* nije faktor, ali *x+1* jeste. U ovom slučaju CRC detektuje sledeće greške:

- $ Sve navalne greške dužine r manje od stepena polinoma G(x)
- $ Sve navalne greške koje utiču na neparan broj bitova
- $ Sve navalne greške čija je dužina jednaka r + 1, sa verovatnoćom $(2^{r-1} - 1)/2^{r-1}$
- $ Sve navalne greške čija je dužina veća od r + 1, sa verovatnoćom $(2^r - 1)/2^r$

Na primer, CRC-32 polinom detektuje sve navalne greške čija je dužina veća od 33, sa verovatnoćom $(2^{32} - 1)/2^{32}$. Ovo je ekvivalentno 99,99999998 odsto tačnosti. Nije loše.

Implementacija CRC-a pomoću cikličnih pomeranja

Pronalaženje metoda za detekciju grešaka sa zadovoljavajućom tačnošću predstavlja samo jednu polovinu "bitke". Drugu predstavlja pronalaženje načina za efikasnu implementaciju. Ako uzmete u obzir bezbroj okvira koji se prenose preko mreže, shvatićete da je efikasna implementacija od suštinskog značaja.

Nakon što savladate osnovne principe CRC-a, Vaša prva reakcija može da bude pisanje programa koji vrši deljenje polinoma. Medutim, dok se takav program izvrši, verovatno će pristići još nekoliko okvira. Dok se provere svi ti okviri, verovatno će pristići još veći broj okvira i nastaće pravo "usko grlo". To bi bilo isto kao da kasirka traži proveru cene za svaki otkucani artikl; u međuvremenu, iza Vas bi se stvorio dugačak red ljudi sa zajedljivim komentarima, a istopili bi se i svi sladoledi koje ste kupili!

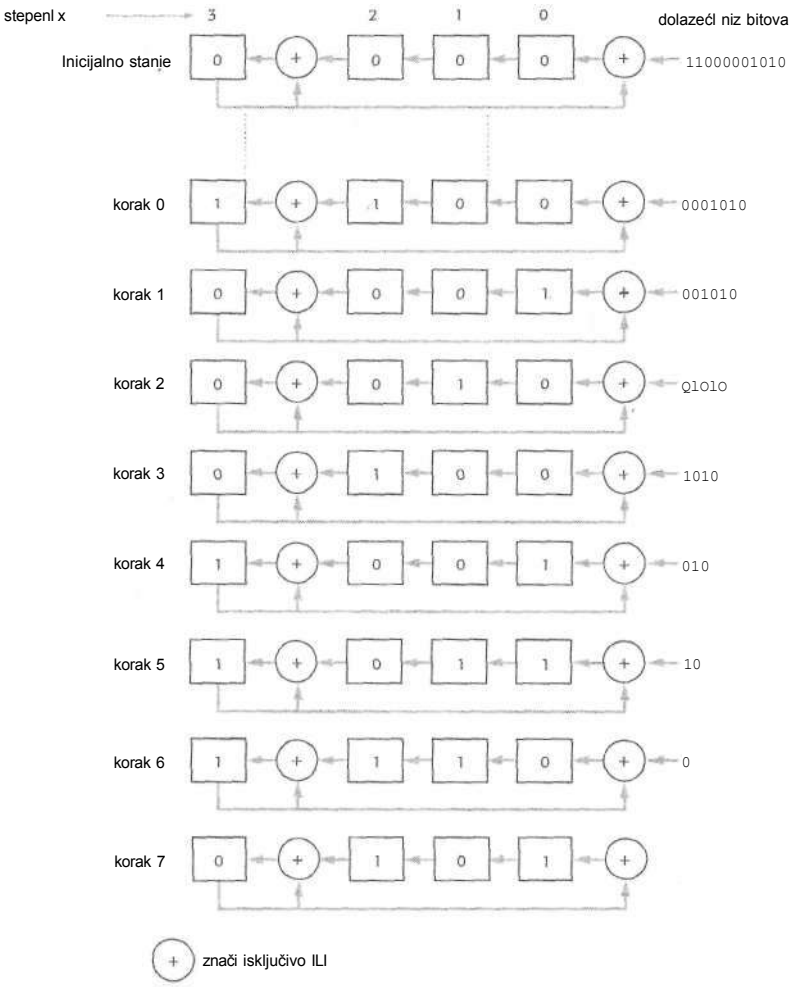
Postoji li način da podelimo dva polinoma i brzo dobijemo ostatak? Da li uopšte moramo da prolazimo kroz kompletan proces deljenja ako nam je potreban samo ostatak? Količnik se nigde ne koristi. Pogledajmo pažljivije sliku 6.5, na kojoj je prikazano sintetičko deljenje. Ceo proces može da se zamisli kao sekvenca pomeranja i isključivih ILI operacija izmedu delitelja i delova deljenika.

Cesta implementacija CRC-a koristi kolo koje se konstruiše u zavisnosti od generatora polinoma *G(x}*. Pošto postoje standardni polinomi, ova kola mogu masovno da se proizvode. KoIo sadrži pomerački registar i izvodi operaciju isključivo ILI u skladu sa sledećim pravilima:

- $ G(x) = $b_r x^r + b_{r-1} x^{r-1} + \ldots + b_2 x^2 + b_1 x + b_0$! gde je *b,* ili O ili 1, a *i* = O, ... r. Broj bitskih pozicija Li registru je r. Krajnja desna pozicija odgovara članu *bO,* a krajnja leva *b^X"'*.
- $ Isključivo ILI kolo se nalazi desno od bilo koje pozicije kojoj je pridružena vrednost b| jednaka 1.
- $ Niz bitova se uvodi u registar jedan po jedan, počevši od krajnje desne pozicije.
- $ Kada se uvede novi bit, svi koji se već nalaze u registru pomeraju se ulevo za jednu poziciju. Svaki bit se propušta kroz isključivo ILI kolo gde postoji, formirajući jedan operand za operaciju isključivo ILI.
- $ Bit sa krajnje leve pozicije se propušta kroz svako isključivo ILI kolo, formirajuđ drugi operand svake operacije isključivo ILI.

Na slici 6.6 prikazani su registar i kola isključivo ILI za polinom $G(x) = x^4 + x^3 + 1$. Primećujete simbol operacije isključivo ILI desno od pozicija koje odgovaraju članovima x^3 i 1, a nema ih desno od pozicija koje odgovaraju članovima x^2 i *x.* Inicijalno, u registru se nalaze sve nule.

Na ovoj slici prikazana su ista izračunavanja kao i na slici 6.5. LI koraku O prvi bit dolazečeg niza (deljenik sa slike 6.5) pomeren je na krajnju levu poziciju u registru. LI koraku I krajnji levi bit se propušta kroz svako isključivo ILI kolo i sve se pomera ulevo.



SLIKA 6.6 *Deljenje korišćenjem cikličnih pomeraja*

Primećujete da su sadržaji registra identični rezultatu prve operacije isključivo ILI iz koraka 1 sa slike 6.5.

Svaki korak definiše isti proces pomeranja ulevo i izvodenja operacije isldjučivo ILI. Sadržaji registra u svim koracima odgovaraju rezultatima slično označenih koraka sa slike 6.5. Kada bitovi iz dolazećeg niza budu propušteni kroz registar, u registru će se naći ostatak (korak 7 na slikama 6.5 i 6.6).

Detektovanje grešaka pomoću CRC-ja je tačan i široko korišćen metod. Može efikasno da se implementira, a potrebno vreme je proporcionalno dužini niza. Standardni generator polinomi omogućavaju hardversku implementaciju celokupnog metoda (u čipovima), čime je dalje poboljšana njegova efikasnost.

6.4 Hamingovi kodovi: Korekcija grešaka

Kao što smo ranije istakli, kada se greške detektuju, obično postoje dve opcije: ponovno slanje originalnog okvira i ispravljanje oštećenog okvira. Druga opcija pored metoda za detektovanje grešaka zahteva i precizno utvrdivanje bitova na kojima su se greške desile. Ovo nije moguće izvesti jednostavnom proverom parnosti.

Korigovanje jednostruke greške

Metod koji je razvio R. W. Hamming uključuje kreiranje specijalnih kodnih reči na osnovu podataka koji se šalju. **Hamingov kod** zahteva umetanje višestrukih bitova parnosti u niz bitova pre nego što se pošalje. Bitovi parnosti proveravaju parnost na strateškim lokacijama. Ideja je sledeća: ako se bitovi promene, njihove pozicije određuju jedinstvene kombinacije grešaka prilikom provere parnosti. Kada se okvir pošalje, primalac ponovo izračunava bitove parnosti. Ako dode do bilo koje greške, kombinacija grešaka može da pokaže koji je bit promenjen. Nakon toga, primalac može da postavi bitove na tačne vrednosti. Ova tehnologija je prilično česta prilikom adresiranja memorije i prenosa bitova iz registara u RAM i nazad.

Ilustrovaćemo primenu Hamingovog koda na najjednostavnijem slučaju detektovanjem i korigovanjem greške na jednom bitu. Pretpostavimo da okviri sadrže osam bitova. Označimo ih kao m^1 *m^l vr? mⁱ m⁵ rrfi* m^7 m^8- Sledeći korak je definisanje bitova parnosti za proveru parnosti na selektovanim pozicijama. Nameću se logična pitanja koliko ćemo provera parnosti koristiti i koje su pozicije obuhvaćene jednom proverom.

Ako koristimo jednu proveru parnosti, ona će ili uspeti, ili neće uspeti. Na osnovu toga, možemo da zakljudmo da se greška javlja, ili ne javlja. Ako koristimo dve provere tačnosti, moguća su četiri ishoda: obe će biti neuspešne, obe će biti uspešne, prva neće biti uspešna, a druga hoće, ili druga neće biti uspešna, a prva hoće. Ova četiri ishoda mogu da se koriste za predstavljanje četiri dogadaja: nema greške, ili postoji greška u jednom bitu na tri moguće pozicije. Pošto postoje više od tri bitske pozicije, dve provere parnosti nisu dovoljne.

U opštem slučaju, ako se koristi n provera parnosti, postoji 2" mogućih kombinacija neuspeha i uspeha. Svakoj bitskoj poziciji moramo da pridružimo jedinstvenu kombinaciju koja će primaocu omogućiti analiziranje provera parnosti i donošenje zaključka o poziciji na kojoj je došlo do greške (ako je došlo do greške). Medutim, da bi se uzele u obzir sve bitske pozicije, potrebno nam je n tako da je 2" veće od broja poslatih bitova. Osim toga, moramo da zapamtimo da svaka dodatna provera parnosti zahteva slanje još jednog bita.

U tabeli 6.1 prikazana je relacija između n i broja poslatih bitova, uz pretpostavku da se šalje 8-bitni okvir. Kao što je pokazano, ako se koriste četiri provere parnosti, postoji 16 mogućih kombinacija za uspešne i neuspešne provere parnosti.

Tabela 6.1: Broj kombinacija za uspešne i neuspešne provere parnosti u funkciji od *n*

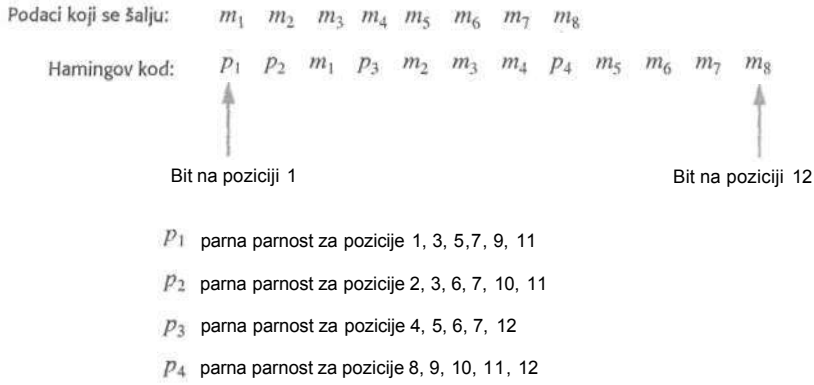
n (Broi provera parnosti)	Broj poslatih bitova	2" (broj mogućih kombinacija uspešnih i neuspešnih parnosti)
1	9	*I*
2	10	4
3	11	8
4	12	16

Četiri dodatna bita parnosti sa osam originalnih bitova podrazumevaju stvarno slanje 12 bitova. Dakle, moguće je da će se desiti 13 različitih dogadaja: nema greške, ili greška postoji u jednom bitu na nekoj od 12 pozicija.

Sledeći korak je pridruživanje kombinacije jedinstvenom dogadaju. Da bi se to izvelo, konstruišu se četiri bita parnosti $p\backslash p^1$, p^3, i p^4 i umeću se u kadar na način koji je prikazan na slici *6.7.* Svaki bit parnosti uspostavlja parnu parnost za selektovane pozicije naznačene na slici. Zašto se bitovi parnosti postavljaju na tim pozicijama? Kako da utvrdimo pozicije za sve provere parnosti?

Da bismo dali odgovore na ova pitanja, proučićemo pozicije uključene u svaku proveru parnosti. Prva provera uključuje sve bitove na neparnim pozicijama. Te pozicije, ako se zapišu u binarnom obliku, imaju 1 kao bit najmanje težine. Ako u binarnom obliku zapišete pozicije obuhvaćene drugom proverom, one će imati 1 na pozicijama bita druge najmanje težine. Slično tome, treća i četvrta provera imaju 1 na pozicijama koje odgovaraju bitovima treće i četvrte najniže težine, respektivno.

Kako nam ovo može pomoći? Kreirajte 4-bitni binarni broj koji se sastoji od fc^4, b^3, b^2, i b^l gde je fy = O ako je provera parnosti za *p,* uspešna, a u suprotnom je fy = 1 *[i* = 1, 2, 3, ili 4). U tabeli 6.2 prikazana je relacija izmedu pozicija sa pogrešnim bitovima, nevalidnim proverama parnosti i 4-bitnim brojem.



SUKA 6.7 *Hamingov kod za jednostruke greske*

Tabela 6.2: Pozicije bitova sa greškom i pridružene greške parnosti

Pozicija bita sa greškom	Netačna provera parnosti	b4, b3, b2 i bl
Nema greške	Ni jedna	0000
1	P_1	0001
2	P_2	0010
3	P_1 i P_2	0011
4	P_3	0100
5	P_1 i P_3	0101
6	P_2 i P_3	0110
7	P_1, P_2 i P_3	0111
8	P_4	1000
9	P_1 i P_4	1001
10	P_2 i P_4	1010
11	P_1, P_2, i P_4	1011
12	P_3, i P_4	1100

Kao što je pokazano u tabeli, postoji poklapanje izmedu 4-bitnog binarnog broja i pozicija sa pogrešnim bitom.

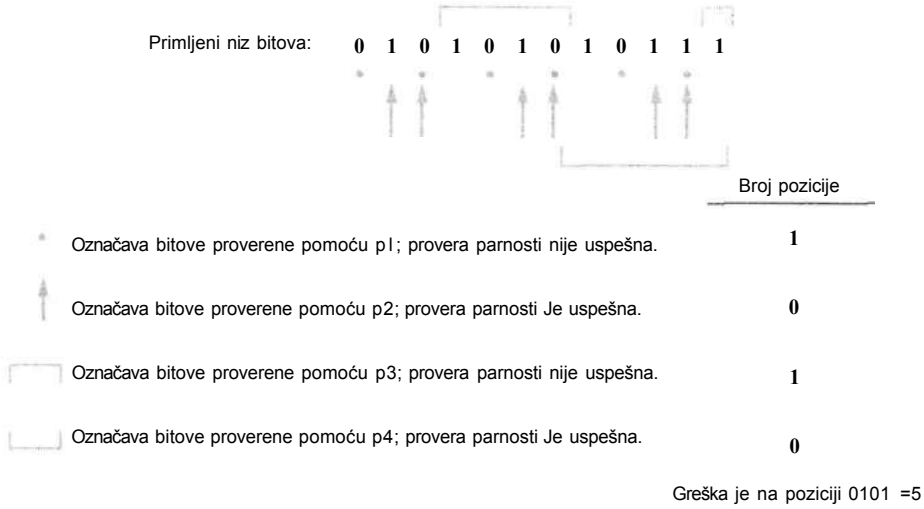
Kada primalac dobije preneti okvir, izvršava provere parnosti. Kombinacija neuspeha i uspeha odreduje da li je došlo do greške i, eventualno, na kojoj se poziciji greška pojavila. Kada primalac zna gde se greška javila, on je ispravlja tako što menja vrednost bita na toj poziciji.

Ilustracije radi, razmotrite primer sa slike 6.8. Ovde vidimo da Je inicijalni okvir bio 0110-0111, a da je preneti Hamingovkod 0101-1101-0111. Ako želite da se uverite da li bitovi parnosti daju parnu parnost na odgovarajućim pozicijama, morate sami da izvršite potrebna izračunavanja.

Na slici 6.9 pokazano je da je primljeni okvir 0101-0101-0111, Sada, ako izvršite proveru parnosti, videčete da su provere za pl i p3 netačne, tj. postoji neparan broj jedinica na pozicijama 1, 3, 5, 7, 9 i 11 i na pozicijama 4, 5, 6, 7 i 12. Znači, prema tabeli 6.2, greška je u bitu 5. Pošto bit 5 ima vrednost 0, primalac ie menia u 1 i okvir je ispravlien.

Podaci:	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**1**				
	m_1	m_2	m_3	m_4	m_5	m_6	m_7	m_8				
Hamingov kod:	**0**	**1**	**0**	**1**	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**1**
	p_1	p_2	m_1	p_3	m_2	m_3	m_4	p_4	m_5	m_6	m_7	m_8

SLIKA 6.8 Mz *bitova pre prenosa*



SLIKA 6.9 *Provera parnosti u okviru nakon prenosa*

Korigovanje višestrukih grešaka

O kodovima za korekciju jednostrukih grešaka možemo da damo sličan komentar kao i o kodovima za detekciju jednostrukih grešaka. Greške u samo jednom bitu su retke prilikom razmene podataka. Takvi kodovi postoje, ali ih nećemo ovde predstaviti. Broj dodatnih bitova može da postane toliko veliki da se koriste samo u specijalizovanim slučajevima. Ako ste zainteresovani, raspravu o tim kodovima možete da pronadete u referencama [Ko78] i [Ha80].

Postoji još jedna klasa metoda za korekciju grešaka koju vredi pomenuti: Bose-Chaundhuri-Hocquenghem (BCH-iz očiglednih razloga) kodovi i Reed-Solomonovi kodovi, koji predstavljaju potklasu BCH kodova. Oba metoda koriste koncept kodnih reči, kolekcije od *N* bitova, iza koje sledi *M* kontrolnih bitova greške, koji se izračunavaju na osnovu bitova podataka. Kodna reč ima $N + M$ bitova i postoji 2^N mogućih kodnih reči. Samo su N bitovi proizvoljni, a preostalih *M* bitova je strogo određeno. To znači da, iako postoji 2^{N+M} mogućih bitskih sekvenci, samo 2^N tih kombinacija (manji broj u poredenju sa ukupnim brojem) predstavlja legitimne kodne reči. Na primer, ako je N = 8 i *M* = 4, postoji 28 = 256 legitimnih kodnih reči od mogućih 212 = 4096 bitskih sekvenci.

Ključni koncept ovih kodova podrazumeva da postoji odredeno *rastojanje* između dve kodne reči, tj. da postoje različiti bitovi u tim ključnim rečima. Na primer, dva Hamingova koda 0101-1101-0111 i 1001-1001-0111 imaju rastojanje 3, jer se razlikuju u tri bita (prvi, drugi i šesti bit). Svaki skup kodnih reči ima *minimalno rastojanje.* Da bi se ono utvrdilo, izračunavaju se rastojanja izmedu svih parova kodnih reči. Najmanja izračunata vrednost predstavlja minimalno rastojanje tog skupa kodnih reči.

Minimalno rastojanje je značajno, zato što je direktno povezano sa brojem oštećenih bitova koje je moguće detektovati i ispraviti. U opštem slučaju, ako je *d* minimalno rastojanje, ovaj metod može da detektuje sve greške koje utiču na manje od *d* bitova (takva promena daje neispravnu kodnu reč) i može da ispravi sve greške koje utiču na manje od *d/2* bitova.

Na primer, pretpostavimo da je minimalno rastojanje skupa kodnih reči d = 10. Znači, bilo koje dve legitimne kodne reči moraju da se razlikuju najmanje u 10 bitova. Greške koje utiču na manje od *d/2* = 5 bitova mogu da se isprave. Da biste ovo videli na konkretnom primeru, pretpostavićemo da je kodna reč poslata i da je došlo do greške u četiri bita. U tom slučaju, rezultat neće predstavljati validnu kodnu reč (da bi se kreirala validna kodna reč, mora da se promeni najmanje 10 bitova). Osim toga, smatraćemo da primalac pretpostavlja da će sve greške uticati na manje od pet bitova. Primalac neispravne kodne reči mora da pronađe najbližu legitimnu kodnu reč; kada je pronađe, zaključuje da je to ispravna poslata kodna reč. Sve druge kodne reči bi morale da imaju namanje šest pogrešnih bitova da bi podsećale na primljenu reč.

Naravno, trik je u tome da se kodne reči izaberu tako da minimalno rastojanje bude što veće. Na taj način se maksimizira broj bitova koje je moguće detektovati i ispraviti. BCH kodovi, a posebno Reed-Solomonovi kodovi, upravo to i rade. Ipak, za detaljnije opise ovih metoda neophodno je poznavanje poljaa Galoa (Galois), teorije o matricama i generator polinoma, tako da ćemo zainteresovane čitaoce uputiti na reference [GrOl], [Wi95], i [Sw02]. Medutim, ne dopustite da Vas složenost ovih metoda navede na pomisao da je reč o nekim apstraktnim teorijama visokog nivoa, koje imaju samo nekoliko primena. Kada sledeći put budete slušali muziku sa CD-a, ili gledali film sa DVD-ja, setite se da oni koriste prednosti Reed-Solomonove tehnologije.

6.5 Zaključak

U ovom poglavlju su prvenstveno obradene dve teme: detektovanje i korigovanje grešaka nastalih prilikom prenosa informacija. Detekcija jednostavno podrazumeva utvrđivanje da li je došlo do greške; ako je došlo, onda se oslanja na druge protokole za ugovaranje dodatnih razmena da bi bile dobijene tačne informacije. Korekcija podrazumeva izvršavanje promena nakon što se podaci prime bez dodatnih prenosa.

Predstavljena su tri metoda.

- **Bitovi parnosti** Ovaj metod detekcije grešaka namenjen je prvenstveno detekciji jednostrukih grešaka, a navalne greške detektuje sa verovatnoćom 50 odsto. Medutim, može da bude koristan kada se bitovi prenose zasebno, kao kod nekih memorijskih arhitektura. Osim toga, predstavlja osnovu za tehnike korekcije grešaka.
- **Ciklična provera redundantnosti** Ovaj metod detekcije grešaka zasnovan je na teoriji deljenja polinoma. Nizovi bitova se interpretiraju kao polinomi. CRC bitovi su kreirani tako da, kada se poruka podeli sa generator polinomom, dobijate ostatak deljenja jednak nuli. Deljenjem primljene poruke sa generator polinomom i proverom ostatka deljenja sa velikom tačnošću može da se utvrdi da li je došlo do grešaka u toku prenosa. Ovaj metod se često koristi i lako se implementira korišćenjem cikličnih pomeračkih kola i registra. Odredeni polinomi su proglašeni standardima.

- **Hamingov kod** Ovaj kod za korekciju grešaka uspostavlja kolekciju bitova parnosti na strateškim pozicijama. Ako dode do jednostruke greške, njegova pozicija utiče na jednistvenu kombinaciju provera parnosti. Ovo ne samo da omogućava detektovanje greške, već i utvrđivanje pozicije bita na kome je došlo do greške. Ako se zna pozicija greške, onda se lako može i otkloniti.

Dakle, šta je bolje: detektovanje, ili korekcija grešaka? Kao što ste mogli i da očekujete, odgovor je da ni jedno nije bolje, bar u opštem smislu. Tehnike za korekciju u opštem slučaju zahtevaju veće troškove i ne mogu u svim primenama da opravdaju svoje postojanje, ukoliko se greške retko javljaju. Obično je mnogo jeftinije jednostavno zatražiti ponovni prenos oštećenih informacija. Tipično, većina kompjuterskih mreža se ubraja u ovu kategoriju.

Ako se greške češće javljaju, dodatni troškovi zbog ponovljenih prenosa počinju da predstavljaju problem. U takvim slučajevima je možda jeftinije uključiti dodatne bitove korekcije, umesto da se medijum opterećuje preteranim redundantnim prenosima.

Učestalost pojave grešaka nije jedini faktor koji treba uzeti u obzir. I ponovno slanje okvira zahteva određeno vreme. Trajanje zavisi od brojnih faktora, kao što su količina saobraćaja, brzina prenosa podataka i rastojanje. U većini slučajeva, kratko kašnjenje kod prijema email poruka, ili fajla sa IAN servera nije "strašno", a ponekad nije ni primetno. Medutim, u real-time okruženjima kod kojih se poruke moraju isporučiti pravovremeno da bi se izbegle katastrofe, čak ni ovako mala kašnjenja nisu dopuštena. Real-time aplikacije, kao što su gledanje, ili slušanje multimedijalnih zapisa, ne mogu sebi da "priušte luksuz" ponovnog slanja oštećenih podataka. Podaci moraju da se vide, ili čuju čim se pošalju. Sonde za svemirska ispitivanja, kod kojih signali putuju po nekoliko sati do svojih odredišta, ozbiljno su hendikepirane ako se poruka mora ponovo preneti, posebno ako postoji velika verovatnoća da će se smetnje ponovo pojaviti. Zamislite astronauta koji kaže: "Halo, NASA, ne čujemo vas. Sta ste rekli u vezi pretećeg sudara sa vanzemaljskim brodom?".

Pitanja i zadaci za proveru

1. Šta je bit parnosti?
2. Koja je razlika izmedu parne i neparne parnosti.
3. Koja je razlika izmedu korekcije grešaka i detekcije grešaka.
4. Sta je navalna greška?
5. Da li su sledeće tvrdnje tačne, ili netačne (zašto)?
   a. Nije neuobičajeno da se izgube jedan, ili dva bita u toku prenosa.
   b. Iako je tačna tehnika, CRC je vremenski složena tehnika, jer zahteva dodatne troškove.
   c. Generator polinom može da se izabere proizvoljno sve dok i pošiljalac i primalac znaju o kom polinomu je reč.
   d. CRC će detektovati navalnu grešku proizvoljne dužine sve dok je broj bitova koji se menjaju neparan.
   e. Kodovi za korekciju grešaka su efikasniji od kodova za detekciju grešaka, jer ne zahtevaju rentransmisiju podataka.
6. Šta je ciklična provera redundantnosti?

7. Pod kojim uslovima CRC detektuje sledeće greške?
   a. jednostruke greške
   b. dvostruke greške
   c. navalne greške čija je dužina manja, ili jednaka stepenu generator polinoma
   d. navalne greške čija je dužina veća od stepena generator polinoma
8. Koje uslove mora da ispuni generator polinom? Zašto?
9. Klasifikujte greške koje će CRC metod uvek detektovati.
10. Klasifikujte greške koje CRC metod neće detektovati.
11. Sta je pomerački registar?
12. Sta je Hamingov kod?
13. Definišite rastojanje izmedu dve kodne reči.

Vežbe

1. Navedite argument koji pokazuje da jednostavna provera parnosti detektuje greške samo u slučajevima kada se promeni vrednost neparnog broja bitova.
2. Pretpostavite da neki statički elektricitet koji traje 0,01 sekundu utiče na komunikacionu liniju za 56 Kbps modem. Na koliko bitova ova smetnja može da utiče?
3. Pretpostavite da se 128 bitova podataka proverava pomoću metoda za detekciju grešaka koji koristi čeksumu. Navedite primer koji pokazuje da je moguće da se promene vrednosti dva bita i da greške ne budu detektovane. Navedite još jedan primer koji pokazuje da je moguće promeniti tri bita podataka, a da greške ostanu nedetektovane. Da li je moguće da se promene svi bitovi, a da greške i dalje ne budu detektovane?
4. Zašto se za 0 - 1 = 1 koristi oduzimanje po modulu 2?
5. Koji polinom odgovara sledećem nizu bitova?

   0110010011010110

6. Koristeći metode opisane na slikama 6.2 i 6.3, izračunajte ostatak deljenja sledečib polinoma:

$$\frac{x^{12} + x^{10} + x^7 + x^6 + x^5 + x^3 + x^2}{x^7 + x^4 + x^2 + x^1}$$

7. Pretpostavite da želite da pošaljete podatke 100111001 i da je generator polinom $X^6 + x^3 + 1$. Koji je niz bitova stvarno poslat?
8. Nacrtajte ciklični pomerački registar i isključiva ILI kola za CRC-12 i CRC-16 standardne polinome.
9. Koristeći ciklična pomeranja, izračunajte ostatak deljenja sledecih polinoma:

$$\frac{x^{12} + x^{10} + x^7 + x^6 + x^5 + x^3 + x^2}{x^7 + x^4 + x^2 + x^1}$$

10. Proučite dokumentaciju za LAN na Vašem univerzitetu, ili u Vašoj kompaniji i utvrdite koji se metod za detekciju grešaka (ako postoji) koristi.

11. Pretpostavite da generator polinom ima član x kao faktor. Navedite primer greške koja neće biti detektovana.

12. Pretpostavite da želite da generišete Hamingov kod za korekciju jednostruke greške za 16-bitni niz podataka. Koliko Je bitova parnosti neophodno? Sta se dogada ako se koristi 32-bitni niz podataka?

13. Primljeni su sledeći 12-bitni Hamingovi kodovi (za korekciju jednostrukih grešaka). Koje ASCIl kodirano slovo predstavlja ovaj niz?

```
110111110010
```

14. Konstruišite Hamingove kodove za svako sledeće slovo: A, 0 i {

15. Pretpostavite da pošiljalac ima sledeće okvire podataka:

Broj okvira	Podatak
I	0 1 1 0 1 0 0 1
2	1 0 1 0 1 0 1 1
3	1 0 0 1 1 1 0 0
4	0 1 0 1 1 1 0 0

    Pretpostavite da pošiljalac konstruiše Hamingov kod za svaki okvir, formirajući dvodimenzionalni niz (svaki red sadrži jedan Hamingov kod), i da šalje jednu po jednu kolonu. Sta primalac dobija ako se zbog greške u četvrtoj koloni prime sve nule? Primenite metod za korekciju grešaka na primljene podatke i ispravite ih.

16. Razvijte Harningov kod koji može da ispravi sve jednostruke greške i da detektuje sve dvostruke greške u 8-bitnom nizu podataka.

17. Pretpostavite da 4-bitni broj $b_4\ b_3\ b_2\ b_v$ koji je opisan u tabeli 6.2, formira broj veći od 12. Sta to znači?

18. Napišite program koji uzima osam bitova podataka i kreira 12-bitni Hamingov kod.

19. Koliko iznosi minimalno rastojanje izmedu dve kodne reči koje su definisane dodavanjem bita pame parnosti?

20. Koliko iznosi minimalno rastojanje Hamingovog koda definisanog u odeljku 6.4?

21. Pretpostavite da se osam bitova dodaje na 32 bita radi kreiranja 40-bitne kodne reči. Koji procenat ukupnog broja 40-bitnih kombinacija predstavlja legitimne kodne reči?

22. Koliko bitova u okviru greške može da detektuje Hamingov kod?

Reference

[GrOl] Gravano, S. *Introduction to Error Control Codes.* Oxford and New York: Oxford University Press, 2001.

[Ha80] Hamming, R. W. *Coding and Information Theory.* Englewood Cliffs, NJ: Prentice Hall, 1980.

[Ko78] Kohavi, Z. *Sivitching and Finite Automata Theory,* 2nd ed. NewYork: McGraw-Hill, 1978.

[Mo89] Moshos, G. *Data Communications: Principles and Problems.* St. Paul, MN: West, 1989.

[Pe72] Peterson, W.W., and EJ. Weldon. *Enor Correcting Codes,* 2nd ed. Cambridge, MA: MITPress, 1972.

[Sw02] Sweeney, P. *Error Control Coding: From Theory to Practice.* New York: Wiley, 2002.

[Wi95] Wicker, S. *Error Control Systems for Digital Communications and Storage.* Englewood Cliffs, NJ: Prentice Hall, 1995.

POGLAVLJE 7

Zaštita podataka

Ko može da ospori da je privatnost dragocena? Uvek je bila oznaka privilegije i karakteristika istinske urbane kulture. Još od pečinskog doba, preko koliba, puebla i tvrđava, čovek je nastojao da sebi izgradi dom u kome će biti zaštićen i imati sopstveni kutak. Tako je bilo u svim dobima. Sirotinja je morala da se "tiska" po gradovima samo iz nužde. AH, sa napretkom civilizatije, bilo je Ijudi koji su sebi mogli da priušte luksuz povlačenja na mirna mesta.

—**Phyllis McGinley** (1905-1978), američki pesnik, autor

7.1 Uvod

Da li ste ikada naručili nešto preko Interneta, ili ste tražili da saznate stanje na svojoj kreditnoj kartici, ili bankovnom računu preko mreže? Možda ste tražili kolegine radove na Intemetu? Koliko puta ste zastali i pomislili da je ono što vidite, u stvari, preneto iz neke baze podataka na Vašu lokalnu poziciju i da, teorijski, svako može da vidi ono što Vam se prikazuje? To uključuje Vaše ocene, brojeve kreditnih kartica, bankovnih računa i stanja i sve ostalo što može da se smatra privatnim podacima.

Kada odete u banku, obično nećete pokazivati svoje uplate i stanje računa osobi koja čeka u redu sa Vama. Zbog istih razloga, elektronski transfer novca izmedu banaka mora da bude bezbedan, tako da neautorizovani ljudi ne mogu da dobiju pristup Vašim finansijskim podacima. Ne samo banke, veći mnogi drugi komunikacioni sistemi moraju da budu bezbedni. Idealno bi bilo da se pristup obezbedi samo autorizovanim, a da se zabrani neautorizovanim osobama. Ali, kako da informacije budu bezbedne ako se šalju preko satelita, ili pomoću mikrotalasa? Informacije slobodno "putuju" kroz vazduh i skoro je nemoguće sprečiti neuatorizovani prijem. Čak i kod kablovskog prenosa može da bude teško sprečiti nekoga da pronade izolovanu tačku u ormaru, ili podrumu i da se ne priključi na kabl.

Uobičajeni pristup za osiguravanje prenosa, koji je dovoljno jak, ne vodi računa o neautorizovanom prijemu. Zašto brinuti o nečemu što ne može da se spreči? Umesto toga, ovaj pristup menja (šifruje) poruke, tako da nisu razumljive za neautorizovane ljude koji ih presretnu. Šifrovanje je uobičajeno kod kablovske televizije (CATV). Svako ko ima CATV može da prima signal filmskih stanica, ali je on

Ako platite odgovarajuću sumu lokalnoj kablovskoj kompaniji, ona će deskremblovati signal, ili Vam obezbediti uredaj koji će to uraditi. Tek tada možete da gledate raspoložive filmove.

U ovom poglavlju se uglavnom bavimo problemima bezbednosti, uključujnći protokole da bi se osiguralo bezbedno rukovanje informacijama i metode koji šifruju i dešifruju osetljive podatke. Odeljak 7.2 počinje opisima nekih uobičajenih algoritama za šifrovanje koji koriste javne ključeve. Ako znate ključ šifrovanja i korišćeni metod, možete da obrnete proces i da dešifrujete podatke. Neki koriste termin *kriptosislemi sa simetričnim Mjučem,* jer su često ključevi za šifrovanje i dešifrovanje isti. Kasnije ćete videti da postoje i druge mogućnosti. Naravno, ovaj pristup podrazumeva da ključ mora da bude zaštićen. Ne sme da padne u pogrešne ruke. Kako možemo na siguran način da obezbedimo ključ i za pošiljaoca i za primaoca? Neki od mogudh pristupa su predstavljeni u odeljku 7.3.

LI odeljku 7.4 obradićemo šifrovanje javnim ključem. Nećemo brinuti da ključ ne dospe u pogrešne ruke, jer ga ionako svako zna. Ideja je da, ako neko zna ključ i metod za šifrovanje, i dalje ne bude sposoban za dešifrovanje poruke. Osoba mora da zna metod za dešifrovanje. Da, to znači da možda nećete moći da dešifrujete ni poruke koje ste sami šifrovali. U odeljku 7.4 predstavićemo metod šifrovanja zajedničkim javnim ključem, RSA algoritam i probleme verifikacije i autentifikacije kod tehnika koje koriste javne ključeve.

U odeljku 7.5 opisaćemo Secure Socket Layer (SSL) i Transport Layer Security, najčešće implementirane pristupe za osiguravanje transfera preko Intemeta. Oni koji se bave takvim aktivnostima mogu da prepoznaju bezbednu konekciju po tome što URL počinje sa *https,* umesto uobičajenim prefiksom *http.*

Sa stanovišta korisnika, najveće probleme verovatno prave kompjuterski virusi i ostale pretnje za sistem. Virusi napadaju kompjutere, često uništavajući informacije na njima. Među ostale pretnje se ubrajaju ljudi koji pokušavaju da upadnu na odredeni sistem i da pronađu privatne informacije. LJ odeljku 7.6 ćemo predstaviti firevvalle, koji se široko koriste za zaštitu cele infrastrukture od spoljašnjeg sveta. Objasnićemp kako firevvalli funkcionišu i neke dizajnerske pristupe za njih. U odeljku 7.7 saznaćete šta su virusi, kako funkcionišu i kako su napredovali u pokušajima da se izbegnu procedure detekcije.

Konačno, u odeljku 7.8 se bavimo ostalim bezbednosnim pretnjama: "crvima" i hakerima. Navešćemo razlike između "crva" i virusa, predstaviti maliciozne hakere i opisati Internet "crv", opštepoznati dogadaj koji je izazvao ozbiljne probleme pre par godina na mnogim kompjuterima.

Bezbednost je izuzetno značajna oblast, tako da postoje tomovi informacija o toj temi. Llključeni su tehnički, pravni i sociološki aspekti. Ovi problemi često dovode do podela, koje su, recimo, prouzrokovane razlidtim političkim filozofijama, u SAD udružile Vladu i Nacionalnu bezbednosnu agenciju (NSA - National Security Agency) u pronalaženju bezbednih šema za šifrovanje i načina za njihovo zaobilaženje. LI poslednje vreme postoje sporenja i o slobodi pristupa u akademskim mrežama, tako da istraživači pronalaze načine da "razbiju" bezbednosne sisteme. Da li to rade zbog akademskih, ili zbog nezakonitih razloga? Da biste bili sigurni, postoji mnogo toga što mora da se "pokrije" u ovoj oblasti, a mi samo možemo da obezbedimo uvodne informacije o različitim spornim detaljima.

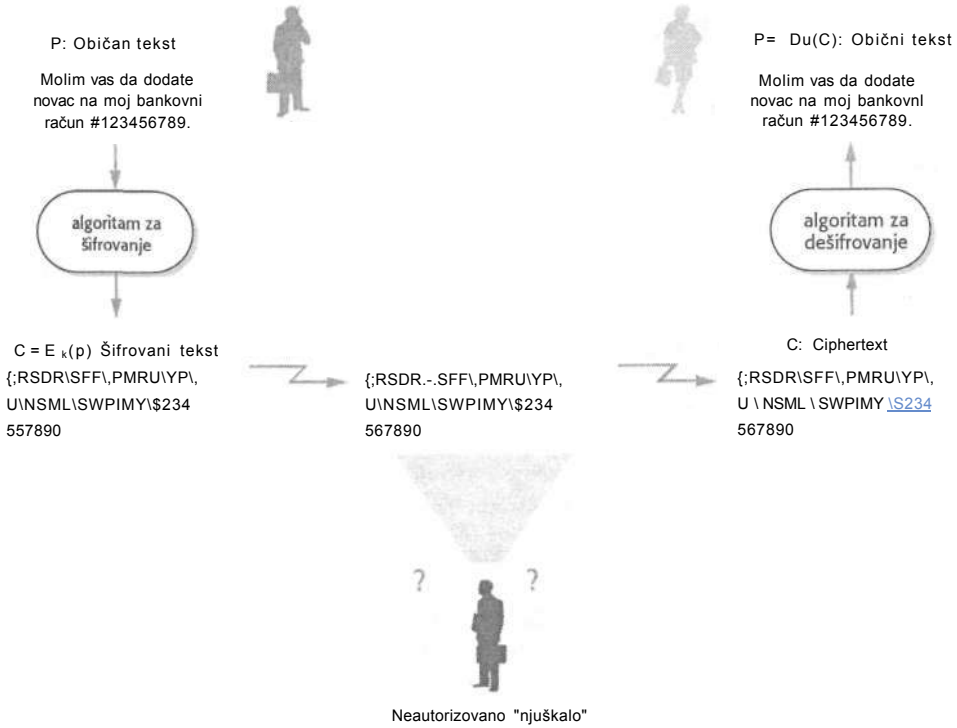
7.2 Algoritmi za šifrovanje

Metodi za detekciju i korekciju grešaka sprečavaju da ljudi dobiju netačne informacije. Sledeći potencijalni problem je nelegalni, ili neautorizovani prijem informacija. Takvi slučajevi uključuju uobičajenog pošiljaoca i primaoca i treću stranu koja presreće prenete informacije koje joj nisu namenjene (slika 7.1). Najgore je to što ni pošiljalac, ni primalac možda neće znati da je neko neautorizovano presreo njihove informacije sve dok treća strana ne zloupotrebi te informacije, bilo za ucenjivanje, kriminalne prevare, ili narušavanje nacionalne bezbednosti. Ali, do tada šteta je već udnjena. Jasno, ako nameravamo da šaljemo osetljive informacije pomoću nekog medijuma, biće nam polreban neki način za osiguravanje privatnosti.

Uloženi su veliki napori da se informacije koje eventualno mogu da presretnu neautorizovani korisnici učine nerazumljivim. Prevodenje informacija u drugačiji, neprepoznatljivi oblik naziva se šifrovanje (encryption). Autorizovani primalac mora da razume informacije, tako da mora da ima mogućnost menjanja šifrovanih podataka u njihov originalni oblik. Taj proces nazivamo đešifrovanje (decryption). Cesto se koriste i termini obični tekst (plaintext) za originalnu poruku i šifrovani tekst (dpherteXt) za šifrovanu pomku.

Proces Je ilustrovan na slici 7.2. Pošiljalac koristi ključ šifrovanja (obično neki karakter, ili numeričku konstantu) za promenu običnog teksta (P) u šifrovani tekst (C). Simbolički, to može da se zapiše kao C = $E]_1[P)$, gde E i k predstavljalju algoritam i ključ šifrovanja, respektivno. Ako neka neautorizovana osoba dobije C, nema nikakve koristi od nerazumljivih informacija. Eventualno, primalac dobija C i đešifruje ga da bi dobio originalnu poruku. To se simbolički zapisuje kao P = $D_{„}$.(C), gde su D i fe' algoritam i ključ za đešifrovanje. U opštem slučaju, P = $D_{„}(E_k(P))$. LI mnogim slučajevima (mada ne uvek), $k = k'$.

SLIKA 7.1 *Slanje nesigurnih poruka*



SLIKA 7.2 *Slanje šifrovanih poruka*

Kao i obično, nameću se brojna pitanja. Kako funkcionišu algoritmi za šifrovanje i dešifrovanje? Da li je šifrovana poruka stvarno nerazumljiva za neautorizovanog primaoca? Da li neautorizovani primalac koji zna kako je poruka šifrovana može da dešifruje poruku? Idealno bi bilo da šifrovanu poruku nije moguće dešifrovati bez poznavanja algoritma za dešifrovanje i ključa. Nažalost, krajnje bezbedni kodovi su analogni nepotopivim brodovima kao što je *Titanih* čim budete sigurni da su bezbedni, pokazaće se da grešite.

Cezarovo (Caeser) šifrovanje

Jedan od najranijih i najjednostavnijih kodova menja svaki karakter običnog teksta drugim karakterom. Izbor zamene zavisi samo od karaktera običnog teksta. Ovaj metod se naziva **monoalfabetsko šifrovanje,** ili **Cezarovo šifrovanje,** jer, navodno, datira još iz vremena Julija Cezara. Na primer, možete da dodate 1 (ključ šifrovanja) na ASCII kod svakog karaktera. Tako A postaje B, B postaje C i tako redom. Ovaj pristup se široko koristi, recimo u dečijirn TV šouima, u prstenovima za dekodiranje i na poledini kutija sa žitaricama. Na slici 7.2 je korišćeno Cezarovo šifrovanje. Možete da li da zamislite razumne motive za kasnije zamene?

Algoritam za dešifrovanje normalno obrće korake koji su korišćeni za šifrovanje. U prethodnom primeru, oduzimanjem 1 od svakog ASCII koda u šifrovanom tekstu dobijaju se karakteri originalnog običnog teksta. Ovo je slučaj kada su ključevi za šifrovanje i dešifrovanje identični. Treba da istaknemo da smo u primeru mogli da menjamo ASCII kodove za bilo koju konstantnu vrednost.

Iako se jednostavno opisuje i definitivno daje nerazumljive poruke, Cezarovo šifrovanje se retko koristi u ozbiljnim aplikacijama. Relativno se lako dekodira bez poznavanja originalnog metoda za šifrovanje, jer kod ne radi ništa drugo nego prikriva često korišćena slova, ili kombinadje. Na primer, najčešće korišćena slova engleskog alfabeta su *E, T, O, A* i N. Ako se neko slovo često javlja u šifrovanom tekstu, postoji velika verovatnoća da je to neko od navedenih slova, a ne Q, iliZ. To predstavlja dobru poiaznu osnovu za "razbijače" kodova.

Ilustracije radi, razmotrite sledeći šifrovani tekst (sa slike 7.2). Zamislite da nikada niste videli originalnu poruku.

```
{;RSDR\SFF\,PMRU\YP\,U\NSML\SVVPIMY\$234567890
```

Karakteri koji se najčešće javljaju u ovom primeru su \ - sedam puta, S - četiri puta i R, P i M - po tri puta. Zato postoji velika verovatnoća da su to zamene za *E, T, O, A* i *N,* ili, ćak, za blanko znak.

Sledeči korak bi uključivao isprobavanje različitih kombinacija karaktera običnog teksta na mestima gde se nalaze šifrovani karakteri. Na primer, nakon nekoliko pokušaja, možete da otkrijete da parcijalno šifrovani string izgleda ovako (dešifrovani karakteri su svetliji):

```
{;EADE\AFF\,QNEU\YO\,U\NANL\AVVOINY\$234567890
```

U nastavku možete da primetite da većina poruka ima blanko znakove izmedu reči i da najčešće korišćeni karakter (\) predstavlja blanko znak. Pokušajte da generišete sledeći string

```
{;EADE AFF ,ONEU YO ,U NANL AVVOINY $234567890
```

Zatim, možete da tražite YO i AFF i zapitajte se koliko se dvoslovnih reči završava slovom O i koliko troslovnih reči počinje sa A, iza čega sledi duplo slovo. Nema ih mnogo, tako da možete da pokušate da zamenite Y sa *T* i *F* sa D. Sada imate

```
{;EADE ADD ,ONEU TO ,U NANL AVVOINT $234567890
```

Nekim pronicljivim nagadanjima dobijamo napola dešifrovanu poruku. Neće biti teško dovršiti poruku (ovo je slično popularnoj igri pogadanja sa vešanjem, ili televizijskom šou programu "Točak sreće"). Ovde je važno istaći da bezbedan kod ne sme da zadrži odredene slovne sekvence, ili učestalost sa kojom se odredena slova javljaju u originalnoj poruci.

Polialfabetsko šifrovanje

Jedan od načina da se promene učestalost pojavljivanja slova i uobičajene selcvence je da se koristi **polialfabetsko šifrovanje.** Poput monoalfabetskog šifrovanja, svaki karakter se menja drugim karakterom. Razlika je u tome što se određeni karakter običnog teksta ne menja uvek istim šifrovanim karakterom. Zamenu možemo da izaberemo ne samo na osnovu konkretnog karaktera običnog karaktera, već i na osnovu njegove pozicije u poruci.

Sledeći kodni segment prikazuje jednostavan primer. Nizovi P i C predstavljaju obične i šifrovane karaktere, respektivno, a K je celobrojni ključ.

```
for (int i = 0; i<length of P; i++)
    C[i]=P[i]+K+(i mod 3);
```

Pretpostavite da je K= 1. Zatim se dodaje 1 na ASCII kodove karaktera na pozicijama 0, 3, 6 i tako dalje; 2 se dodaje na kodove karaktera na pozicijama 1, 4, 7, i tako dalje; 3 se dodaje na kodove karaktera na pozicijama 2, 5, 8 i tako redom. U slučaju stringa THEMTHENTHEY šifrovani string je UJHNVKFPWIG\. Izgleda kao da ovo šifrovanje rešava problem ponavljanja, ali činjenica je da ga, u stvari, samo redukuje. I dalje se javljaju ponavljanja i određeni šabloni. Na primer, string THE je šifrovan na tri načina: UJH, VKF i WIG. Ako postoji više THE podstringova, šifrovane verzije će se češće pojaviti. Možete da koristite vrednost veću od 3 u kodnom segmentu. Tako se omogućava veći broj moguđh načina za šifrovanje THE podstringova, čime je obezbeđen manji broj ponavljanja. Medutim, ako je string dugačak, i dalje će postojati ponavljanja.

Osim toga, postoje i drugi šabloni u šifrovanom tekstu. Možete li da ih uočite? Prva slova u svakom ekvivalentu šifrovanog teksta za podstring THE *(U, V* i W) su sukcesivna. Isto važi i za druga po redu slova (/, *Ki I) i* treća po redu slova (H, *F* i G), iako su u poslednja dva slučaja slova preuredena. Ipak, šabloni i dalje postoje i, ako profesionalac pokuša da "razbije" kod, to je velika olakšica za utvrdivanje korišćenog meloda šifrovanja. Naravno, postoje i daigi načini da se šabloni "razbiju"; predstavićemo ih nešto lasnije.

Šifrovanje premeštanjem

Sifrovanje premeštanjem preureduje slova običnog teksta poruke, umesto da menja karaktere šifrovanim karakterima. Jedan od nadna da se ovo uradi je da se karakteri običnog teksta smeštaju u dvodimenzionalni niz sa m kolona. Prvih m karaktera običnog teksta se smešta u prvi red niza, drugih m karaktera u drugi red i tako dalje. Zatim, utvrdujemo permutacije brojeva od 1 do m i pišemo ih kao $p_{,,}$ p_2, ■■■, $p_{,,,}$. Permutacije mogu da budu slučajne, ili ih odreduje neki tajni metod. U svakom slučaju, finalni korak je prenošenje svih karaktera iz kolone $p_{,,}$ zatim iz kolone p_2 i tako redom. Poslednji preneti sluip karaktera je skup iz kolone $p_{,,,}$.

Ilustracije radi, pretpostavite da su karakteri sledeće poruke smešteni u dvodimenzionalni niz sa pet kolona (tabela 7.1).

```
FOLLOW THE YELLOW BRICK ROAD
```

Tabela 7.1: Dvodimenzionalni niz koji se koristi za šifrovanje premeštanjem

		Brojevi kolona		
1	2	3	4	5
F	O	L	L	O
W		T	H	E
	Y	E	L	L
O	W		B	R
I	C	K		R
O	A	D		

Pretpostavimo da su brojevi kolona uređeni kao 2, 4, 3, 1, 5, tj. da se karakteri u koloni 2 prvi prenose, zatim slede karakteri iz kolona 4, 3, 1 i 5, respektivno. U tom slučaju preneta poruka izgleda ovako:

```
0 YWCALHLB LTE KDFW OIOOELRP
```

Preneta poruka uopšte ne liči na originalnu, ali ako primalac zna brojeve kolona i ako zna permulacije brojeva kolona, lako može da rekonstruiše originalnu poruku. To se izvodi smeštanjem dolazećih karaktera u kolone, uz pracenje redosleda permutacija. U ovom primeru dolazeči karakteri bi najpre bili smešteni u kolonu 2, zatirn 4, 3, 1 i 5. Ovo je sledeći primer u kome se algoritam dešifrovanja definiše obrtanjem koraka algoritma za šifrovanje.

Problem kod šifrovanja premeštanjem je da nije dovoljno sigurno. Pre svega, zadržana su originalna slova. Po prijemu, neautorizovani primalac može da analizira šifrovani tekst i da primeti visoku učestalost pojavljivanja najčešće korišćenib slova. To je već naznaka da nije korišćena zamena slova i da je možda korišćeno šifrovanje premeštanjem. Sledeći korak u "razbijanju" koda mogu da budu grupisanje karaktera i njihovo smeštanje u različite kolone. Primalac neće pokušavati nasurnice da utvrdi uredenje kolona, ali će pokušati da otkrije najčešće korišćene sekvence, kao što su THE, ING, ili IS po redovima. Ovaj proces može značajno da redukuje broj nagadanja i da obezbedi dosta informacija neautorizovanom, ali izuzetno motivisanom primaocu.

Šifrovanje na nivou bitova

Svi prenosi ne podrazumevaja sekvence karaktera. Zato se svi metodi šifrovanja ne bave samo manipulisanjem, ili zamenom karaktera. Neki funkcionišu na nivou bitova. Jedan metod definiše ključ šifrovanja kao string bitova. Izbor je odreden nasumice i tajno. String se prenosi podeljen u podstringove - dužina svakog od njih jednaka je dužini ključa šifrovanja. Nakon toga se šifruje svaki podstring izvodenjem operacije isključivo ILI izmedu podstringa i ključa za šifrovanje.

U ovom slučaju se dešifrovanje ne izvodi obrtanjem koraka za šifrovanje, već se koraci ponavljaju. Drugim rečima, da bi se dešifrovalo nešto što je šifrovano pomoću operacije isključivo ILI, ona se ponovo izvršava izmedu šifrovanih podstringova i ključa za šifrovanje.

Na slici 7.3 demonstrirano je da dvostruka primena operacije isključivo ILI daje originalni string. Ali, da li to uvek funkcioniše tako? Da! Da biste videli zašto, neka p_t bude neki proizvoljni obični string i neka © predstavlja operaciju isključivo ILI. Za vreme procesa šifrovanja/dešifrovanja, na *pi* je dva puta primenjena operacija isključivo ILI, ili sa 0, ili sa 1. Ako je bila 0, imamo

$$(p_i \oplus 0) \oplus 0 = (p_i) \oplus 0 = p_i$$

Ako je bila I, imamo

$$(p_i \oplus 1) \oplus 1 = p_i \oplus (1 \oplus 1) = p_i \oplus (0) = p_i$$

U svakom slučaju, dvostruka primena operacije isključivo ILI generiše originalni bit p_t.

Sigurnost ovog koda zavisi delimično od dužine ključa za šifrovanje. Krađ ključ znači da je originalni string podeljen na više podstringova i da je svaki šifrovan nezavisno. Kada postoji veđ broj podstringova, postoji veća šansa da se jave neka ponavljanja. Pošto se šifruju istim ključem, ponavljaju se i šifrovani podstringovi. Kao i ranije, ponavljanja mogu da pomognu neautorizovanim primaocima da lakše "razbiju" kod. Duži ključevi za šifrovanje "razbijaju" ovaj šablon, ali i dalje postoje neki drugi šabloni. Na primer, pretpostavimo da se koristi n-bitni ključ za šifrovanje n bitova običnog teksta. Ako postoje dva n-bitna dela običnog teksta koji se razlikuju za samo jedan bit, i dve generisane n-bitne šifrovane komponente takode će biti različite samo za jedan bit. Ozbiljni kriptolozi mogu da iskoriste ovu činjenicu kao olakšicu za "razbijanje" kodova.

U ekstremnom slučaju, dužina ključa za šifrovanje je ista kao i dužina poruke koja se šalje. Tada se svaki bit šifruje korišćenjem jedinstvenog bita u ključu. Ako su bitovi ključa stvamo nasumično izabrani, u šifrovanom stringu neće postojati nikakvi šabloni. Osim toga, ako se ključ nikada ne koristi više puta, nema šanse da se pronadu šabloni izmedu različitih šifrovanih tekstova i tako se dobija kod koji je stvarno nemoguće "razbiti" bez isprobavanja svih mogućih ključeva za šifrovanje. Ovakvi kodovi se često označavaju kao **one-time pads.** Nedostatak dugačkih ključeva je što je obavezna komunikacija sa primaocom, pa je zato metod donekle nezgrapan. Osim toga, može da se koristi samo jednom.

1101100101001	Običantekst
1001011001010	KJJuč za šifrovanje
0100111 10001 1	Šifrovani tekst = običan tekst na kome je primenjena operacija isključivo lLI sa ključem za šifrovan
100101 1001010	Ključ za dešifrovanje (isti kao i ključ za šifrovanje)
1101100101001	Obični tekst = šifrovani tekst na kome je primenjena operacija isključivo lLI sa ključem za šifrovanj

SLIKA 7.3 *Korišćenje operacije isključivo ILI nad bitovima*

Standardi za šifrovanje podataka

DES Do sada predstavljeni metodi šifrovanja nisu bili preterano složeni. U stvari, kraći ključevi nisu toliko dobri, jer šifrovani tekst sadrži mnogo naznaka koje neautorizovanoj osobi mogu da olakšaju "razbijanje" koda. Medutim, kada se koriste duži ključevi, šifrovani tekst je teži za "razbijanje". U ekstremnom slučaju, kod je skoro nemoguće "razbiti". Ipak, otežana implementacija dugačkih ključeva otežava korišćenje tog pristupa.

I dalje se koriste pristupi sa kratkim ključevima (relativno kratkim u odnosu na šifrovanu poruku); i oni koriste složene procedure za šifrovanje podataka. Jedan takav metod, nazvan **Data Encryption Standard (DES),** razvijen je u IBM-u početkom 70-ih godina prošlog veka na osnovu eksperimentalnog kriptografskog sistema pod nazivom *Lucifer.* Usvojio ga je kao standard 1977. godine National Bureau of Standards (sada NIST), a koristi ga Vlada Sjedinjenih Američkih Država za komercijalne i informacije koje se ne smatraju poverljivim. ANSI ga je odobrio kao standard privatnog sektora 1981. godine. Primenjuje se u bankarskim transakcijama i za kodiranje PIN brojeva za mašine za automatsko obaveštavanje; medu poznatije korisnike ubrajaju se Department of Justice, Department of Energy i Federal Reserve System. Logika ovog široko korišćenog metoda je ugrađena hardverski (VLSI čipovi), tako da je još brži.

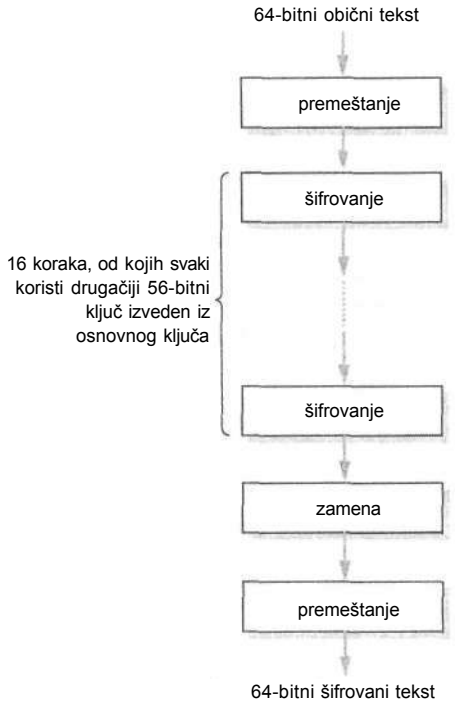
DES je primer **blokovskog šifrovanja.** Poruka se deli na 64-bitne blokove - svaki od njih se šifruje. Koristi 56-bitni ključ* i složene kombinacije premeštaja (preuredenje bitova), zamena (jedne grupe bitova drugom), operacija isključivo ILI i nekoliko drugih obrada bloka za eventualno kreiranje 64 bita šifrovanih podataka. Sve u svemu, 64-bitni blok prolazi kroz 19 sukcesivnih koraka, gde izlaz svakog koraka predstavlja ulaz za sledeći korak.

Na slici 7.4 prikazani su osnovni koraci. Prvi korak vrši premeštanje bitova izmedu 64 bita podataka i 56-bitnog ključa. Sledećih 16 koraka (označenih kao *šifrovanje* na sliti) uključuje razne operacije, koje ćemo ukratko opisati. Svaki korak je isti, osim što koristi drugačiji ključ izveden iz originala. Značajno je napomenuti da izlaz iz jednog koraka predstavlja ulaz za sledeći korak. Pretposlednji korak (označen kao *zamena* na slici) vrši zamenu prva 32 bita sa poslednja 32 bita. Poslednji korak vrši drugo premeštanje. U stvari, reč je o obrnutom premeštanju u odnosu na prvi korak. Rezultat su 64 bita šifrovanih podataka.

Na slici 7.5 prikazane su primarne operacije svakog od 16 središnjih koraka. String bitova predstavljamo na slici slovom i numeričkim subskriptom. Subskript ukazuje na broj bitova u stringu. Na primer, K_{56} se odnosi na 56-bitni string koji se koristi kao ključ, dok je X_{48} 48-bitni string koji predstavlja rezultat neke meduoperacije. Iako na slici koristimo isti simbol X_1 on predstavlja različite podstringove u svakoj fazi. Ovaj pristup deluje razumnije nego da se koriste različiti nazivi za svaku operaciju.

Prvo, DES deli C_{64} (64 bita koji će se šifrovati) na pola. Prva 32 bita su L_{32}, a preostala 32 bita su označena kao R_{32}. Zatim, R_{32} se proširuje u 48-bitni string premeštanjem nekih bitova i dupliranjem odredenih bitova. Rezultat je označen kao R_{48}, čime se ukazuje da je u potpunosti izveden iz stringa R_{32}. Osim toga, algoritam menja 56-bitni ključ deljenjem na pola i izvršavanjem cikličnih pomeranja u svakoj polovini.

* U stvari, koristi se 64-bitni ključ ali se za detekciju grešaka koristi osam bitova. Zato se u procesu šifrovanja koristi samo 56 bitova.

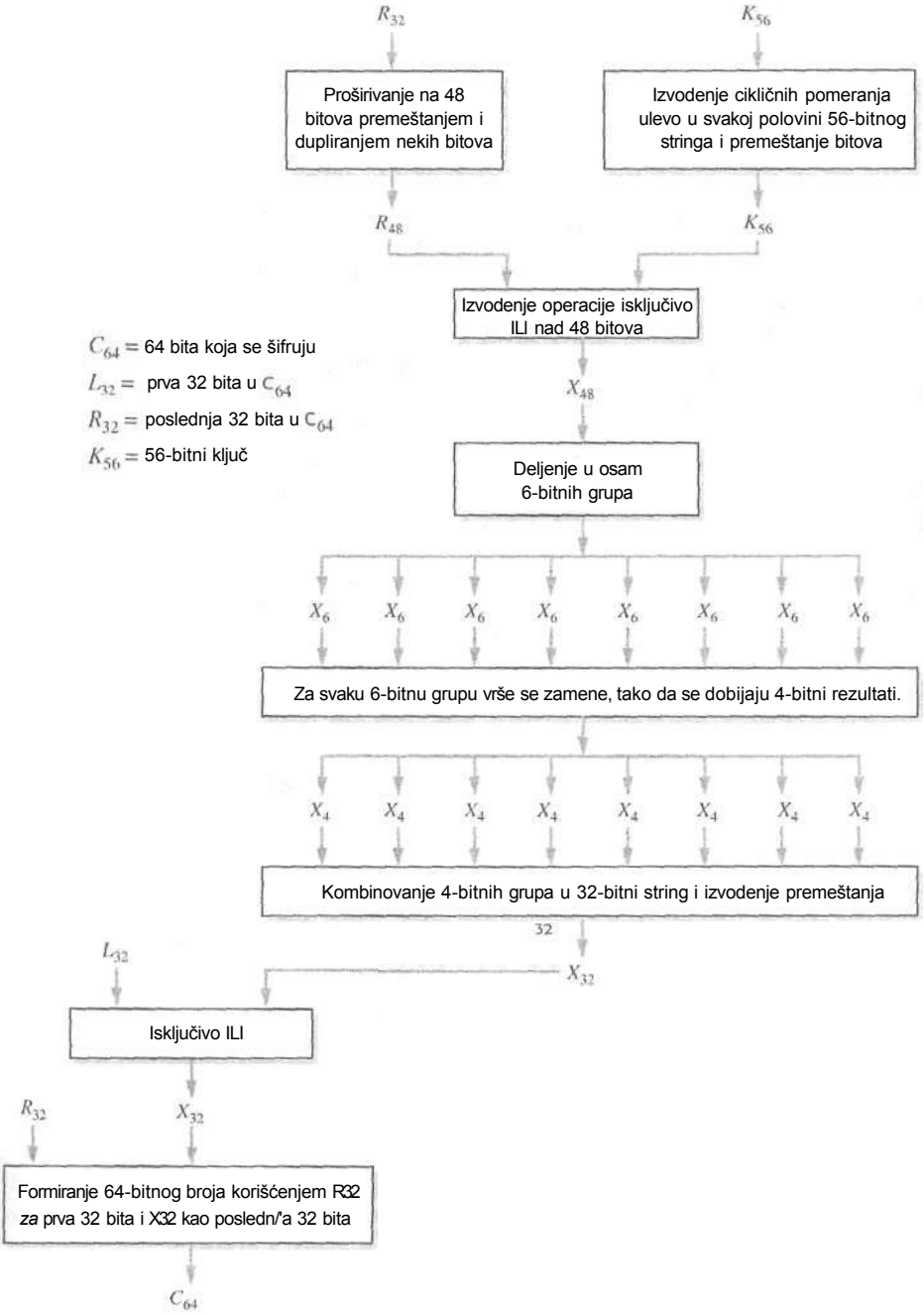


SLIKA 7.4 *Opšti pregled DES procesa*

Broj cikličnih pomeranja zavisi od toga koji se od 16 mogućih koraka trenutno izvršava. Poenta je u tome da se u svim koracima koristi drugačiji ključ. Nakon pomeranja, ključ je ispremeštan. Rezultat je označen kao K_{56}

Zatim, algoritam vrši operaciju isključivo ILI izmedu R_{48} i prvih 48 bitova ključa K_{56}. Rezultat je označen kao X_{48}. Nakon toga, X_{48} se deli na osam 6-bitnih grupa (X_6). Svaka 6-bitna grupa prolazi kroz algoritam zamene i dobija se 4-bitna grupa označena sa X_4. Rezultujućih osam 4-bitnih grupa se kombinuje i vrši se novo premeštanje, tako da se dobije sledeća 32-bitna grupa, označena kao X_{12}. Zatim, algoritam primenjuje operaciju isključivo ILI između tog stringa i L_{32}. Ponovo rezultat označavamo kao X_{32}. Konačno, algoritam kreira 64-bitni string, koristeći R_{32} kao prva 32 bita i X_{32} kao poslednja 32 bita. Celokupni proces se izvodi 16 puta. Svaki put ulaz predstavlja izlaz prethodnog koraka i koristi se drugačiji ključ.

Zbunjujuće? Dobro, i trebalo bi da bude. IBM-ova namera nije bila da dizajnira metod koji svako može lako da razume. Ideja je da se dizajnira metod koji se sastoji od većeg broja konvolutivnih koraka i skoro da ga je nemoguće reprodukovati bez prethodnog poznavanja ključa za šifrovanje. Izostavili smo mnogo detalja, kao, na primer, kako se izvode premeštanja i kako se 6-bitne grupe menjaju 4-bitnim grupama. Detalji izvodenja zamena uglavnom zavise od uspostavljenih pravila za zamenu i tabela nazvanih S-boxovi, koje definišu kako se određeni string mapira u drugi string.



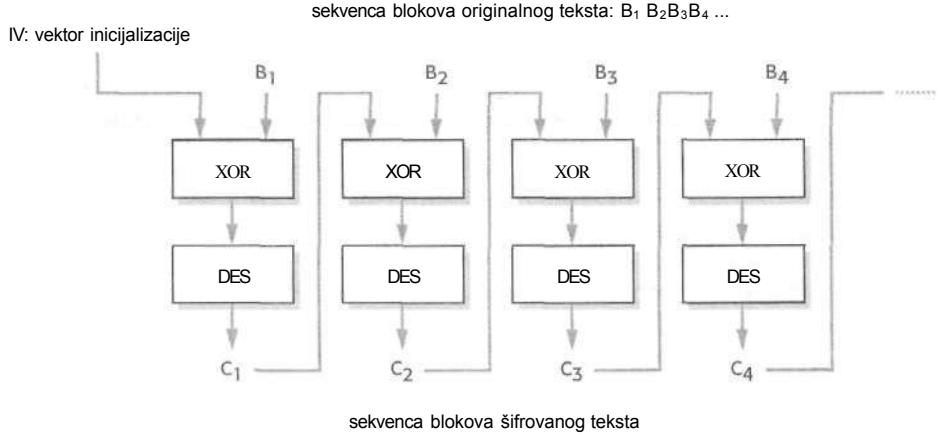
SLIKA 7.5 *Jedan od 16 koraka DES procesa*

Ako ste zainteresovani, u referenci [St95] možete da pronadete detaljan opis DES algoritma.

DES može da funkcioniše u nekoliko modova, uključujući ECB **(electronic codebook)** mod i CBC **(chiper block chaining)** mod.* U ECB modu algoritam jednostavno šifruje svaki od 64 bita običnog teksta na opisani način za kreiranje odgovarajućeg 64-bitnog šifrovanog bloka. Ako se isti 64-bitni blok javlja više puta u originalnom tekstu, uvek generiše isti 64-bitni blok šifrovanog teksta. Naravno, na taj način se kreira šablon ako je originalni tekst dovoljno dugačak, koji može da se koristi kao nagoveštaj za nekoga ko pokušava da "razbije" kod.

CBC mod narušava ovaj šablon. Pre šifrovanja jednog bloka originalnog teksta algoritam najpre vrši operaciju isključivo ILI izmedu tog bloka i prethodno šifrovanog bloka (slika 7.6). Zatim se rezultat šifruje. Za prvi blok originalnog teksta operacija isključivo ILI se vrši sa **vektorom inicijalizacije,** koji se definiše u sklopu implementacije. Značajno je to da svaki blok šifrovanog teksta zavisi ne samo od odgovarajućeg bloka originalnog bloka, već i od svih prethodnih blokova. Sledeći način razmišljanja podrazumeva da svaki blok šifrovanog teksta zavisi i od bloka originalnog teksta i pozicije bloka u originalnom tekstu. Kao rezultat, ako se isti blok originalnog teksta javlja na različitim pozicijama, najverovatnije će generisati različite blokove šifrovanog teksta i tako se narušava prethodno opisani šablon.

Dakle, koliko je DES dobar? Postoje razni načini da se algoritam šifrovanja napadne, ali cilj je stalno utvrđivanje ključa i originalne pomke. Postoje različite tehnike, koje često zavise od šablona u šifrovanom kodu, ili od slabosti algoritma koji dopušta generisanje šablona. Na primer, **diferencijalna kriptoanaliza** uključuje traženje parova blokova originalnog teksta koji se razlikuju na određene načine. Ako se te (ili slične) razlike jave u šifrovanim parovima, postoji šablon koji je moguće iskoristiti.



SLIKA 7.6 CBC *mod za DES šifrovanje*

* Za više detalja o drugim modovima pogledajte referencu [MoOlD.

Sećate se da je polialfabetsko šifrovanje iz prethodnog odeljka pokazalo takve šablone. Drugi metod napada je, jednostavno, brutalna sila, metod kod koji isprobavaju svi mogući ključevi sve dok se ne otkrije pravi ključ. Očigledno, ovo je mnogo teže ako je broj mogućih ključeva veliki. U referenci [MoOlD predstavljeni su drugi načini za napade na šifrovane sisteme.

Većgodinama, mnogi istraživači koji proučavaju DES pokušavaju da pronadu slabosti, ili šablone koji bi se mogli iskoristiti za "razbijanje" koda. Ipak, zabeležili su samo manji uspeh. Osim toga, brutalni napadi se smatraju teškim, jer sa 56-bitnim ključem postoji 2^{56} » 7.2 x 10" mogućih vrednosti ključa. Nažalost, sa procesorima koji rade na nekoliko gigaherca i današnjim paralelnim sistemima ovaj broj je dostižan. U stvari, 1998. godine Electronic Frontier Foundation je kreirala **DES** Cracker, specijalno dizajnirani kompjuter, koji je koštao 250.000 dolara. Ovo je možda vđiki novac za Vas, ili za mene, ali ne predstavlja nikakav problem za organizovani kriminal, terorističke grupe, ili neprijateljske vlade koje mogu da ostvare koristi od "razbijanja" zaštite Vladinih i privatnih institucija visokog nivoa. U saradnji sa svetskom mrežom personalnih kompjutera, Fondacija je mogla da isproba milijarde ključeva u sekundi i da "razbije" DEC kodove za nekoliko sati. Zbog toga je DES napušten.

Originalna standardizacija DES-a je uvek bila kontroverzna (videti referencu [Ko77]). Kada su istraživači u IBM-u poćeli da pripremaju rešenje problema, koristili su 128-bitni ključ Međutim, na zahtev NSA, ključ je redukovan na 56 bitova. Razlozi za tu redukciju nisu javno objavljeni. 128-bitni ključ bi imao mnogo veći broj moguđh kombinacija ($2^{128} \sim 3 \times 10^{38}$) i značajno bi otežao brutalne napade. *

Sledeći faktor koji je doprineo kontroverznosti algoritma je to što su neki Ijudi smatrali da obrazloženja za zamene u DES algoritmu nikada nisu u potpunosti ubedljiva. Strahovalo se da u tim zamenama možda postoji nešto što može da naruši integritet šifre. Zato su postojale spekulacije da za NSA nije bilo dobro da postoji kod koji se ne može "razbiti" bez velikih problema. Zapamtite, s obzirom na široko korišćenje elektronske pošte, raspoloživost DES čipova i sve veći proboj digitalnih govornih prenosa, postoji velika količina DES-šifrovanih informacija. Pretpostavka o nemogućnosti dešifrovanja informacija kada je to potrebno navela je NSA da malo odstupi od prvobitnog stanovišta. Da bi "stvari' bile još gore, postojali su neki izveštaji da je Vlada pokušavala da spreči neka istraživanja, ili publikacije koje su bile posvećene sigurnijim šiframa.

Trostruki **DES** "Razbijanje" DES koda nije predstavljalo nikakvo iznenađenje, jer su mnogi još ranije predvidali da će do toga doći. Zato se prešlo na pronalaženje alternativnih načina šifrovanja. Jedan od tih načina je **trostruki DES,** koji, kao što naslućujete iz naziva, podatke šifruje tri puta. Zasnovan je na ANSI standardu X9.52, a funkcioniše tako što se DES algoritam tri puta uzastopno primenjuje na obične tekstualne poruke. Na primer, pretpostavite da £j,(M) i $D_k(M)$ odgovaraju DES algoritmima šifrovanja i dešifrovanja, respektivno, i da koriste ključ *k* koji se primenjuje na poruku. Trostruki DES je izračunat korišćenjem £/,$_3$(D_{fi2}(E_{kl}(M))).

* Koliko iznosi 3 x 10^{38}? Pretpostavimo da sistem može da isproba milijardu ključeva u mikrosekundi (10^{1fi} ključeva u sekundi). I dalje bi bilo potrebno oko 3 x 10^{23} sekundi da se isprobaju svi ključevi. To je oko 9,5 x 10^{15} godina, a to je više vremena od poslednjeg Velikog praska (možda u vreme onog pre poslednjeg, ali to niko ne zna).

ANSI standard X9.52 defmiše tri opcije za vrednosti ključeva: (1) sve tri vrednosti su medusobno nezavisne, (2) Je$_1$ i k_2 su nezavisni, ali je IJ$_1$ = /<:, i (3) sva tri ključa su jednaka. Naravno, u poslednjem slučaju trostruki DES se svodi na inicijalni DES, jer se prva dva koraka medusobno poništavaju. To znači da je trostruki DES kompatibilan sa običnim DES algoritmom, a to je razlog zašto se algoritam dešifrovanja definiše kao u drugom koraku.

Trostruki DES definiše tehniku koja koristi 168-bitni ključ. Ključevi k_v k_2 i fes jednostavno predstavljaju prvi, srednji i poslednji deo 56-bitnog ključa. Prednost ovog metoda je što se on oslanja na postojeći algoritam koji se pokazao kao solidan i nije ispoljio neke ozbiljne nedostatke. Osim toga, može da se uvede u postojeće DES sisteme zbog postojeće kompatibilnosti. Iako mnogi ljudi predviđaju da će 168-bitni ključevi dugo biti imuni na napade, drugi su trostruki DES kritikovali da Je isuviše *spor,* jer traje oko tri puta duže od običnog DES algoritma. Naravno, spor je relativan pojam, pa neki ljudi veruju da je dodatno vreme vredno poboljšanja sigurnosti.

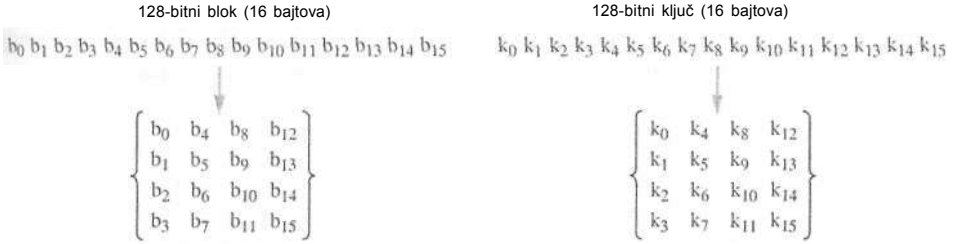
AES i Rijndael algoritam Uvidevši da su dani DES algoritma "odbrojani", NIST je 1997. godine poslao zahtev svetskoj kriptografskoj zajednici sa predlogom novog Advanced Encryption standarda (AES).* Kasnije je NIST primio 15 takvih predloga i objavio ih kao prvi AES kandidat konferencije u avguslu 1998. godine. Nakon analize algoritama kandidata, broj kandidata je redukovan na pet. Razmatrani su algoritmi MARS, RC6, Rijndael, Serpent i Twofish (referenca [Sc98]). Svaki od tih algoritma je #odvrgnut daljim analizama; u oktobru 2000. godine NIST objavljuje da je kao novi AES izabran Rijndaelov algoritam. Krajem maja 2002. godine potvrdeno je usvajanje algoritma kao zvaničnog Vladinog standarda.

AES Rijndaelov algoritam su razvili dr Vincent Rijmen i dr Joan Daemen (što objašnjava naziv algoritma). Kao i za DES, koriste se šifrovanje blokova i ključevi od po 128, 192 i 256 bitova. Tako su alternativne oznake AES standarda AES-128, AES-192 i AES-256. U poslednjem slučaju postoji aproksimativno 1,1 x 10^{77} mogućih ključeva. Broj je toliko veliki da ga je leško pojmiti. Osim toga, algoritam može da šifruje blokove veličine 128, 192 i 256 bitova. U stvari, može da se koristi kombinacija ključa i veličine bloka, tako da postoji devel mogućih varijacija. Predvideno je da Rijndael algoritam ima jačinu najmanje trostrukog DES-a, ali je njegova prednost što se potrebne operacije brže izvode. Pošto je dizajniran da zameni DES, umesto trostrukog DES-a, mnogi predviđaju da će se AES i trostruki DES neko vreme paralelno koristiti.

Dizajn Rijndaelovog algoritma je zasnovan na drugoj blokovskoj šifri nazvanoj SQlIARE.* Detalji su složeni, a kompletno razumevanje Rijndael algoritma zahteva veoma dobro poznavanje matematike, posebno teorije polja, polja Galoa, nesvodivih polinoma i klasa ekvivalencije. Iako ovde ne bi bilo praktično obrađivati te teme, daćemo kratak pregled Rijndaelovog algoritma po glavnim koracima.

* Detalje pogledajte na htip://csrc.nist.gov/encryption/aes/overview.

t Detalje pogledajte na http://www.esat.kuleuven.ac.be/?rijmen/square/index.html.

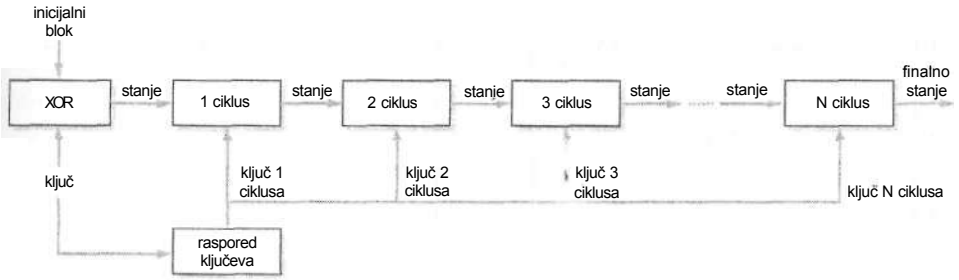


SLIKA 7.7 *Interpreliranje bloka i ključa pomoću matrica*

Nekoliko dobrih referenci obezbeđuje više detalja, ali čitalac mora da ima predznanja koja se stiču na jednom, ili dva kursa iz apstraktne algebre. Te reference su [MoOl], [Da02], http://xsrc.nist.gov/ encryption/aes/rijndael/Rijndael.pdf i http://csrc.nist.gov/publications/fips/ fipsl97/fips-197.pdf.

Da bismo dali opšti pregled Rijndaelovog algoritma, najpre ćemo navesti neke preliminarne koncepte. Pošto ovaj algoritam koristi blokovsko šifrovanje, šifruje se jedan po jedan blok originalnog teksta. Kao što smo ranije istakli, veličina bloka i ključa može da bude 128, 192 i 256 bitova. Medutim, da bismo pojednostavili našu raspravu, koristićemo 128-bitni ključ i blokove od po 128 bitova i komentarisaćemo druge opcije na odgovarajućim mestima. Rijndaelov algoritam "zamišlja" 128-bitni blok originalnog teksta kao sekvencu od 128-bitnih bajtova *(b_0* do b_{15}), koju organizuje u okviru matrice (slika 7.7) sa četiri reda i četiri kolone. Na sličan način se interpretira 128-bitni ključ. I kod većih ključeva i blokova koriste se četiri reda, ali sa povećanim brojem kolona, tako da se smeste svi potrebni podaci.

Prvi korak Rijndaelovog algoritma izvršava operaciju isključivo ILI izmedu inicijalnog bloka i ključa. Rezultat, nazvan *slanje,* uvodi se u prvi od IO ciklusa šifrovanja (slika 7.8).



SLIKA 7.8 *Opšti pregled Rijndaelovog algoritma*

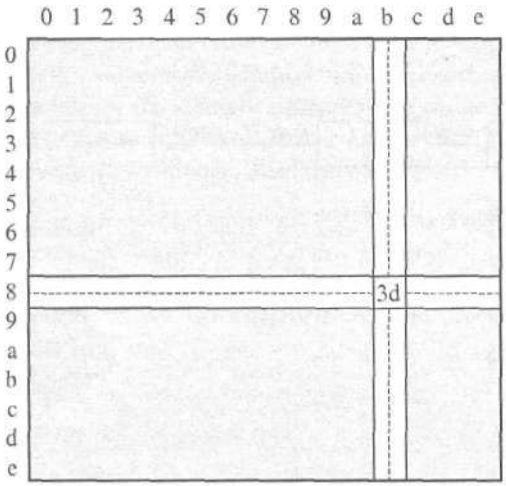
U svakom ciklusu se menja ulazno stanje definisano operacijama sa matricama i kreira se izlazno stanje (u suštini druga matrica), koje se uvodi u sledeći ciklus aktivnosti. Na kraju zahtevanog broja ciklusa šifrovanje se završava. Ako su blokovi veći, koristi se veći broj ciklusa, ali osnovni princip ostaje isti.

Svaki ciklus koristi *zaseban ključ* ciklusa kojim se menja sadržaj ulaznog stanja. Ključevi ciklusa su definisani kroz rutinu za proširivanje ključa, koja započinje interpretiranjem ključa kao sekvence 4-bajtnih reči (po jedna reč u svakoj koloni). U ovoj diskusiji počinjemo sa **četiri** reči: $W_0 = It_0Jt_1Jt_2Jt_3$, $W_1 = fe_4{:}fe_5{:}fe_6{:}fe_7$, $W_2 = k_s{:}k_9{:}k_w Jt_n$, $W_3 = k_{12}{:}k_{13}{:}k_u{:}k_{15}$. Peta i sve naredne reči su kreirane na osnovu prethodno definisanih reči korišćenjem rekurzivne formule. Ovde nećemo zalaziti u detalje, ali sledećim koracima opšteg pregleda utvrđuje se W_I, za $4 < i < 44$, jer postoji 10 ciklusa, a potreban nam je ključ koji sadrži četiri reči za svaki ciklus, što daje ukupno 44 reči, uključujud četiri inicijalne reči. Za veći broj ciklusa potrebno je više reči.

1. Ako i nije deljivo bez ostatka sa 4, onda se wi definiše kao rezultat operacije isključivo ILI izmedu w_u i w_u. **Ako** je i deljivo bez ostatka sa 4, izvodi se sledeći korak.

2. Izvršava se ciklična permutacija bajtova u w_{iA}. Ako su bajtovi bili *[a, b, c, d),* permutacijom se dobija *(b, c, d, d).*

3. Izvodi se zamena svakog bajta u reči iz prethodnog koraka drugim bajtom. Svaka zamena bajta je odredena bajtom koji se menja i tabelom vrednosti zamene nazvanom *S-box.* S-box sadrži redove i kolone (slika 7.9) vrednosti bajta i svaki red i kolona su indeksirani 4-bitnom vrednošću. Da bi se utvrdila odgovarajuća zamena, Rijndaelov algoritam koristi prva četiri bita za utvrdivanje reda u S-boxu i druga četiri bita za utvrđivanje kolone. Vrednost bajta u kojoj se taj red i ta kolona seku koristi se kao zamena za originalni bajt.



SLIKA 7.9 *S-box zamena*

Rijndaelov algoritam definiše celokupni sadržaj S-boxa, ali je na slici 7.9 pokazano samo da bi se vrednost bajta 3b zamenila bajtom *3d,* koji je smešten u 8. redu i *d* koloni.

4. Izvršava se operacija isključivo ILI sa konstantom ciklusa koju definiše Rijndaelov algoritam.

Kada se generišu sve reči, Rijndaelov algoritam koristi prve četiri u inicijalnoj operaciji isključivo ILI i svaku sledeću grupu od četiri reči u svakom narednom ciklusu, tj. i-ti ciklus koristi reč W_I.

Dakle, šta Rijndaelov algoritam radi u okviai jednog ciklusa? U osnovi, postoje četiri koraka u okviru svakog ciklusa, označena kao zamena bajta (byte substitution — BSB), pomeranje reda (shift row — SR), mešanje kolone (mix column — MC) i povećavanje ključa ciklusa (round key addition — RKA).* Svaki ciklus se izvršava na sledeći način.

1. BSB: Svaki bajt u tekućoj matrici stanja menja se drugim bajtom koji je odreden S-boxom. Ovo je slično zameni u okviru procedure proširenja ključa; u stvari, koristi se isti S-box.

2. SR: Pomeraju se bajtovi u svakom redu matrice (osim u prvom) za naznačeni broj pozicija ulevo. Elementi koji se pomeraju iza krajnje leve kolone ponovo ulaze u krajnje desne kolone. Drugim rečima, ovo je levo ciklično pomeranje. Broj pozicija za koje se pomeranje izvodi zavisi od broja reda i ključa i veličine bloka, onako kako to propisuje Rijndaelov algoritam. U našem primeru redovi 2, 3 i 4 su pomereni za 1, 2 i 3 pozicije bajtova, respektivno. Na primer, SR korak transformiše matricu

$$\begin{bmatrix} 45 & 6a & 3b & 67 \\ 76 & da & d4 & 4f \\ fa & 2d & 31 & 9b \\ f5 & 4d & 33 & 78 \end{bmatrix} \text{ u matricu } \begin{bmatrix} 45 & 6a & 3b & 67 \\ da & d4 & 4f & 76 \\ 31 & 9b & fa & 2d \\ 78 & f5 & 4d & 33 \end{bmatrix}$$

3. MC: Ovo je najteži korak za opisivanje, zato što se sva teorija zasniva na izračunavanjima. Opšta ideja je da se svaka kolona interpretira kao četvoročlani polinom i da se pomnoži sa drugim polinomom sa fiksnim brojem članova, $c(x) = 03^{*3} + 01x^2 + 0Ix^1 + 02$ (unapred utvrđeni polinom u okviru Rijndaelovog algoritma). Medutim, množenje se izvodi po modulu $X^4 + 1$, što predstavlja još jedan propisani polinom Rijndaelovog algoritma. To znači da, ako je $p(x)$ polinom koji se množi, onda je rezultat koji ćemo koristiti drugi polinom $q(x)$, čiji je stepen manji od 4, a razlika $p(x) - q(x)$ je jednako deljiva sa $X^4 + 1$. Matematičari kažu da *je* $p(x)$ ekvivalentno $cf(x)$ modulo $X^4 + 1$. Sledeđ faktor koji komplikuje celu "priču" je činjenica da se koeficijenti polinoma interpretiraju kao 8-bitne binarne vrednosti i ti brojevi se ne množe onako kako ste možda pretpostavili. Detaljna objašnjenja su komplikovana, ali svaka 8-bitna vrednost predstavlja element matematičke strukture koja se naziva polje Galoa (poseban tip konačnih polja)' i može se interpretirati kao polinom čiji su koeficijenti ili 0, ili 1. Osim toga, pravila za množenje ovakvih vrednosti su složena i defmisana su karakteristikama teorije polja Galoa, složenom granom matematike.

* U stvari, u poslednjem ciklusu nema MC koraka.

t U opštem slučaju, *polje* definiše matematičku strukturu koja sadrži kolekciju elemenata i operacija (sabiranje, oduzimanje, množenje i deljenje) koje imaju određena svojstva.

Pošto su ove teme isuviše složene za našu raspravu, ne možemo da navedemo detalje izračunavanja. Glavna ideja je da se svaka kolona stanja (interpretirana kao polinom) pomnoži sa fiksnim polinomom (korišćenjem napredne matematičke logike) da bi se generisao proizvod polinoma. Taj proizvod polinoma se interpretira kao 4-bajtna sekvenca, koja se postavlja umesto originalne kolone.

4. RKA: Ovaj korak izvršava operaciju isključivo ILI između tekućeg stanja i ključa ciklusa.

Kada se završi svih 10 ciklusa, rezultujuće stanje predstavlja šifrovanu verziju inicijalnog stanja. Naravno, mi smo ovde izostavili dosta detalja. Na primer, kako se bira S-box? Koje su konstante ciklusa korišćene u metodu za proširivanje ključa? Kakav značaj imaju polinomi $03x^3 + 01x^2 + 0Ix^1 + 02$ i $X^4 + 1$ u MC koraku? Kako Rijndaelov algoritam definiše množenje polinoma? Odgovori na sva ova pitanja zahtevaju dobro poznavanje apstraktne algebre, uključujući teme kao što su prstenovi, polja Galoa, nesvodivi polinomi, polinomi sa koeficijentima u poljima Galoa i srodne transformacije. Krajnji zaključak je da je Rijndaelov algoritam formiran na veoma moćnoj i složenoj matematičkoj teoriji. Čitaoci sa odličnim poznavanjem matematike mogu da pogledaju detaljnija objašnjenja u referencama [MoOl], [Da02], http://xsrc.nist.gov/encryption/aes/rijndael/Rijndael.pdf i http://csrc.nist.gov/publications/fips/fipsl97/fips-197.pdf.

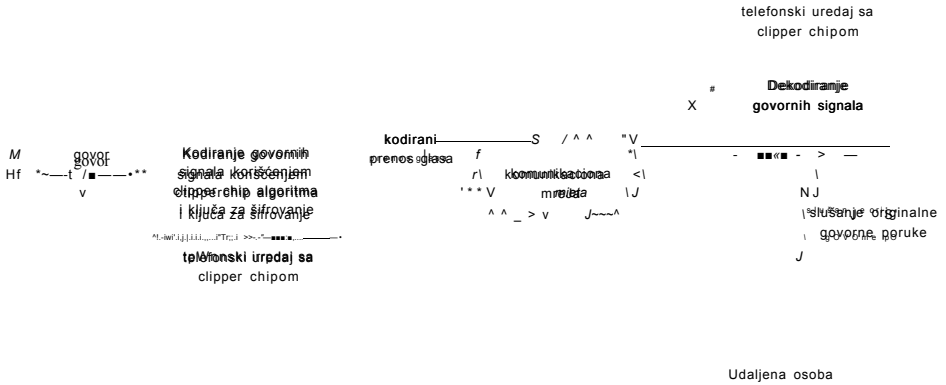
Clipper Chip i Skipjack algoritam

U aprilu 1993. godine pažnju javnosti je privukla objava administracije predsednika SAD Bila Klintona da se planira nova bezbednosna inicijativa. Inicijativa je podrazumevala planove za novu tehnologiju pod nazivom Clipper Chip,* kompjuterski čip dizajniran po nacrtima Vlade, sa ugrađenom šifrom, koji je mogao da se koristi u bezbednosnim uređajima. Ti uredaji su mogli da se koriste u običnoj komunikacionoj opremi, kao što su telefoni, ili faks mašine.

Način njihovog funkcionisanja je sasvim jednostavan (slika 7.10). Clipper Chip sadrži algoritam za šifrovanje koji je dizajniran u njegovim mikrokolima. Pretpostavimo da zovete nekoga telefonom i da želite da započnete bezbednu konverzaciju, tj. konverzaciju koja se ne može presresti, niti je može razumeti neka treća strana. Sve što treba da uradite je da pritisnete dugme i Vaš bezbednosni uredaj i onaj na prijemnoj strani razmenjuju ključeve za šifrovanje.[f] Nakon toga, bezbednosni uredaj rutira sve što kažete, zajedno sa ključem za šifrovanje do Clipper Chipa, koji kodira govorne signale. Telefonski sistem zatim prenosi kodiranu poruku. Na prijemnoj strani bezbednosni uredaj dekodira govorni prenos, koristeći svoj Clipper Chip (to može jer je razmenjen ključ za šifrovanje), i ponovo uspostavlja originalnu govornu poruku. Krajnji rezultat je da osoba na prijemnoj strani čuje Vaš glas kao i u slučaju svih ostalih telefonskih konverzacija.

* Iako se obično koristi termin *CUpPeV$_1$* zvanični naziv tehnologije je bio *Capstone.*

t Postoje različiti protokoli za razmenu ključeva za šifrovanje koji onemogućavaju presretanje ključa u toku prenosa. U sledečem odeljku ćemo predstaviti opciju pod nazivom Diffie-Hellman razmena ključa.



SLIKA 7.TO *Clipper Chip šifrovanje*

Medutim, treća strana koja eventualno može da upadne u konverzaciju dobija samo kodirane govorne signale.

Ova inicijativa je pokrenuta zbog dva glavna razloga. Prvi je bio potreba obezbedivanja privatnih telefonskih razgovora i zaštite svih osetljivih informacija koje se prenose preko telefona, faksa, ili kompjutera. Drugi je bio odgovor na potrebu zakonskog gonjenja kada osetljive informacije odgovaraju ilegalnim aktivnostima. Ovo deluju kao sasvim opravdani razlozi i zato se postavlja pitanje čemu kontroverze.

Prvo, Clipper Chip su dizajnirali inženjeri u NSA, bez ikakvog učešća privatnih industrija (ref. [Si96]). Uzimajući u obzir narastajuću tenziju izmedu civilnih "slobodnjaka" i Vladinih službenika, sama ova činjenica je bila dovoljan razlog za kontroverze, ali postoji tu još mnogo toga. Metod koji je korišćen za šifrovanje je takozvani *Skipjack algoritam,** koji je razvijen u NSA i čiji su detalji poverljivi. To je izazvalo sumnje u nekim kmgovima koji su smatrali da algoritam ne može da se testira na isti način na koji su testirani neki drugi algoritmi čiji su detal/i *javno* dostupni. Neki su ovo smatrali narušavanjem zakona Computer Security Act, koji je usvojen 1987. godine u Kongresu, a čija je svrha bila da ograniči ulogu NSA u uspostavljanju standarda. Pristalice poverljivih algoritama su smatrali da čuvanje odredenih detalja u tajnosti nema nikakve veze sa namerom da se algoritam na taj način čini još bezbednijim. Umesto toga, namera je bila da se spreči neautorizovana konstrukcija uredaja koji bi bili kompatibilni sa autorizovanim, ali koji ne bi implementirali određene zakonske regulative.

Ovo nas dovodi do sledećeg kontroverznog aspekta, koji uključuje same ključeve za šifrovanje. Pretpostavimo da neko koristi telefon, ili druge komunikacione uredaje dok se nalazi pod sumnjom da obavlja neke nelegalne aktivnosti.

•"Preskočićmo" detalje ovog algoritma, ali zainteresovani čitaoci mogu da pronadu više detalja na Web sajtu htlp://csrc. nist.gov/encryption/skipjack/skipjack.pdf.

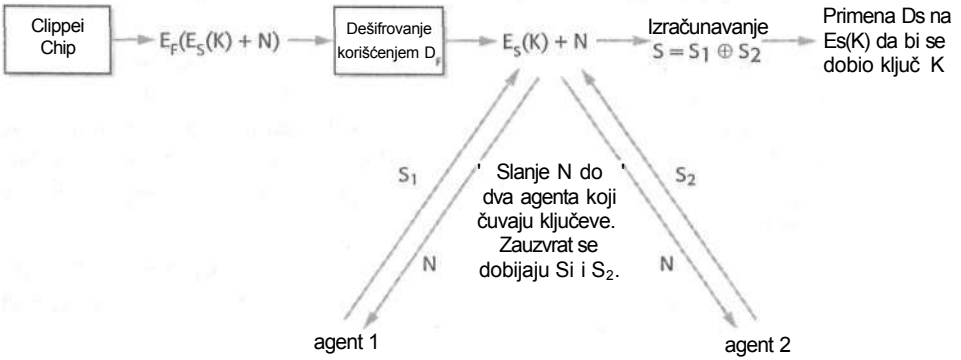
Uobičajena sredstva koja se koriste u zakonske svrhe su prisluškivanje telefonskih linija i praćenje komunikacije. Ako je osumnjičeni šifrovao sve komunikacije, prislušni uređaj ne može da obezbedi korisne informacije. Osim toga, metod šifrovanja je dobar i ne može da se "razbije" za odgovarajuće vreme. Suština problema leži u tome da li obični gradani imaju prava na tehnike šifrovanja koje se ne mogu "razbiti" i kada pripadnici službi za očuvanje reda i zakona imaju pravo da prisluškuju privatne razgovore. Ovo je tema koja izaziva žučne debate. FBI i druge slične agencije zauzimali su se za uključivanje Clipper Chipa, koji bi im omogućio utvrđivanje ključa za šifrovanje i, samim tim, dekodiranje šifrovanih informacija.

Svaki Clipper Chip ima sledeće informacije:

- K: 80-bitni ključ sesije koji se koristi za šifrovanje prenete poruke. Ovo službe za očuvanje reda i zakona moraju da znaju da bi im se omogućilo efikasno prisluškivanje.
- F: 80-bitni ključ "familije". Svi čipovi u grupi koriste ovaj jedinstveni ključ.
- N: 30-bitni serijski broj koji je jedinstven za svaki čip
- S: 80-bitni tajni ključ, koji je, takode, jedinstven za svaki čip, a koriste ga službe za očuvanje reda i zakona

Poslednji ključ S nalazi se u centru kontroverznosti. Kao što je istaknuto, svaki Clipper Chip generiše šifrovanu govornu poruku £„(Poruku). Osim toga, generiše zakonski odredeno polje *Ep(E$_s$(K) + N).** Poslednji izraz je veoma važan i zato ćemo ga pažljivije proučiti. U suštini, Clipper Chip kreira sopstveni ključ sesije kao izlaz, čak i u šifrovanoj formi. Sve što je potrebno je metod za njegovo dobijanje. Na slici 7.11 opisani su neophodni koraci.

Kada sud odobri upotrebu prislušnog uredaja, D_F se primenjuje na zakonski regulisano polje da bi se dobilo $E_S(K) + N$. Ključ "familije" F nije tajni, tako da on ne predstavlja problem; teorijski, svako može da ga ima.



SLIKA 7.11 *Utvrđivanje ključa za šifrovanje*

* Kao što je već rečeno u ovom odeljku, *E* se koristi za predstavljanje metoda za šifrovanje. Subkript zavisi od ključa kojim se vrši šifrovanje. + u izrazu $E_S(K)$ + N odnosi se na konkatenaciju N-ovih bitova na £$_S$(K).

U ovom trenutku službenici mogu da izvuku serijski broj čipa i šifrovani ključ sesije $E_5(K)$. Ostaje da se primeni D_5 da bi se dobio ključ sesije. Pošto je S tajni ključ, to nije jednostavno, ali postoji način. Međutim, najpre ćemo proučiti kako se S kreira.

Tajni ključ S je, u stvari, definisan sa druga dva ključa, u skladu sa formulom $S = S_1 \copyright S_2$ (© je operacija isključivo ILI na nivou bitova). Ključevi S_1 i S_2 su, takode, tajni i održavaju ih dve zasebne agencije za čuvanje ključeva, koje su su ovlašćene za čuvanje i zaštitu dragocenih informacija. Kada se konstruiše odredeni Clipper Chip, uvek je prisutan po jedan član iz svake agencije. Svaki predstavnik selektuje nasumice izabrani 80-bitni ključ, koji se izlaže nizu izračunavanja. Jedan agent daje ključ S_1 a drugi S_2. Ni jedan agent ne zna šta je onaj drugi izabrao. Nakon toga se nalazi rezultat operacije $S = S_1$ ffi S_2 i programira u čip. Jedan agent beleži S_1 i N (serijski broj čipa), a drugi S_2 i N. Svaki prenosi ove dragocene informacije do svojih agencija, gde se čuvaju na bezbednim lokacijama. Ovde je značajno napomenuti da se tajni ključ S ne smešta na jednom mestu. Tako je obezbeden dodatni nivo sigurnosti, jer ni S_1, ni S_2 ne mogu, sami za sebe, da obezbede nikakve korisne informacije. U stvari, kompjuteri koji izračunavaju S i programiraju čip mogu čak da budu i uništeni u đlju daljih mera zaštite.

Kada službenici za očuvanje reda i zakona dobiju serijski broj čipa, mogu da pošalju kopiju svakoj agenciji koja čuva ključ, zajedno sa dokazom da je prislušni uredaj autorizovan. Svaka agencija kao odgovar šalje svoj deo ključa zajedno sa naznačenim serijskim brojem. Službenici mogu da dobiju oba ključa S_1 i S_2, izračunaju S i primene D_5 na $E_3(K)$ da bi bio dobijen ključ sesije K. U toj tački mogu da dešifruju sve poruke koje su šifrovane pomoću E_K i prislušni uredaj je uspešan.

Agenti koji su nadležni za komponente ključa zaduženi su i za mehanizme za smeštanje i zaštitu ključeva. Mnogi smatraju da bi ove agencije morale da budu odvojene od agencija za očuvanje reda i zakona. Glavno pitanje je pravo ljudi na privatnost. Sporno je to što bi postojala mogućnost zloupotrebe ako agencija za očuvanje reda i zakona može sama da čuva komponente ključa, ili ih dobija od druge srodne agencije. Sa nezavisnim agencijama koje nemaju ništa sa zvaničnim službama za očuvanje reda i zakona bolje su zaštićena prava gradana na privatnost. Ako Vas interesuje više detalja o mehanizmima sistema za čuvanje ključeva, pogledajte referencu [De36J.

Uzaključku ovog odeljka naglašavamo da NISTdefiniše kriptografske standarde u svojim Federal Information Processing Standards (FIPS) publikacijama. U tekstovima o kriptografskim standardima možete da naidete na termin odobren od FlPS-a (FIPS approved), što znači da je standard naveden ili u FIPS publikacijama, ili ga je usvojio FIPS i u dodatku, ili drugim dokumentima referencira se sa FlPS-om. U vreme kada je ova knjiga nastajala postojala su četiri algoritma koja je odobrio FIPS; AES, DES, trostruki DES i Skipjack.

7.3 Distribuiranje i zaštita ključa

Svi do sada predstavljeni metodi su pretpostavljali da je ključ za dešifrovanje izveden (ili jednak) iz ključa za šifrovanje. Samim tim, najbolji metod za šifrovanje na svetu ne vredi ništa ako se ključ ne može čuvati na tajnom mestu. Zato se suočavamo sa problemom kako pošiljalac prenosi ključ do primaoca **(distribuiranje ključa,** ili *razmena ključa).* Na primer, Clipper Chip razmenjuje ključeve pre nego što otpočne bezbedna konverzacija. Možda će Vaš prvi predlog biti da pošiljalac jednostavno pošalje ključ. Međutim, šta ako ga presretne neautorizovani primalac? Onda ćete

možda predložiti da se ključ šifruje, ali koji metod šifrovanja primeniti? Kako pošiljalac obaveštava primaoca koji je metod šifrovanja koristio? Ovo ne rešava problem; jedva da ga redefiniše.

Čuvanje ključa na bezbednoj lokaciji nije nimalo jednostavan zadatak, ali postoje opcije i za to. Na primer, dve osobe koje treba da komuniciraju mogu da se sretnu na nekoj skrivenoj lokaciji i **da** se dogovore o ključu. Međutim, ponekad ovakvi sastanci nisu dopušteni. Sledeća opcija može **da** bude transport ključa pod oružanom pratnjom. Ovo definitivno nameće sliku ljudi sa aktn tašnama vezanim lisicama za njihove ručne zglobove, sa "opasnim" raomcirna u pratnji, koji neprestano drže ruke u džepovima sakoa.

Shamirov metod

Jedan od metoda za distribuciju ključa **Shamirov metod** koristi se u scenariju koji se razlikuje od prethodno opisanog. Pretpostavimo da su šifrovane informacije toliko poverijive da se slanje, ili prijem.ključa ne mogu poveriti samo jednoj osobi. Želimo da se ključ utvrdi tako da najmanje *k* osoba bude obavezno prisutno. Dalje, pretpostavljamo da će bilo kojih *k* osoba sa odgovarajućom dozvolom biti dovoljno, tj. ne postoje zahtevi za prisustvom bilo koje određene osobe, ili osoba.

Smeštanje ključa na jednom mestu nije prikladno, jer se tako narušava uslov da mora da bude prisutno *k* osoba. Ključ bi se mogao podeliti na k delova, koji bi se mogli distribuirati do *k* osoba. Ako bi svaka osoba dobila po jedan deo ključa, stvorilo bi se ograničenje u vezi toga ko mora da bude prisutan (samo osobe sa različitim delovima ključa). Ako se nekoliko delova ključa daa istoj osobi, manje od *k* - 1 osoba ima preostale delove, što znači da za utvrdivanje ključa nije neophodno *k* osoba, što je u suprotnosti sa polaznom pretpostavkom.

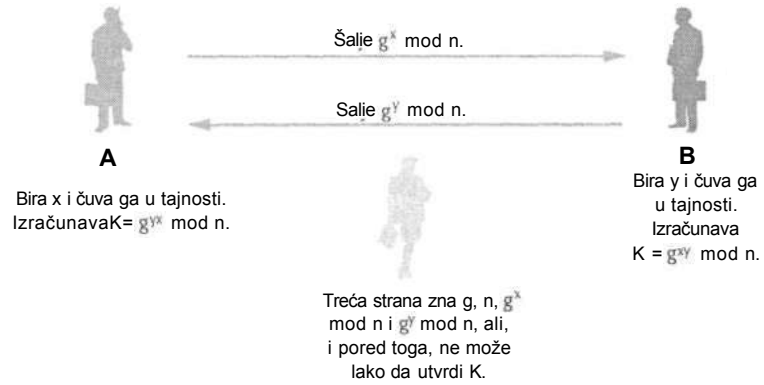
Shamirov metod (ref. [Sh79]) zasnovan je na interpolaciji polinoma. Pretpostavimo da Je $p(x) = a_0 + a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$ polinom stepena k - 1. Osim toga, pretpostavimo da su (X_l, y,), (x^2, y_2), ..., (x_k, y_k) poznate tačke na grafiku *p(x)* i da je $X_1 * X_1$ svaki put kada je *i ćj.* Ovih *k* tačaka određuje polinom *p(x)* na jedinstven način i na osnovu njih možemo da utvrdimo vrednosti koeficijenata

Kod Shamirovog metoda polinom *p(x)* je konstruisan tako da se jedan od njegovih koeficijenata (na primer, a_a) koristi kao ključ za šifrovanje. Svako ko ima dozvolu za slanje, ili prijem informacija dobija tačno jednu tačku sa grafika *p(x),* čime je obezbeđena jedinstvenost x-koordinata tačaka krive. Svaka grupa od *k* osoba može da obezbedi *k* jedinstvenih tačaka. Sve tačke zajedno omogućavaju utvrdivanje polinoma, a, samim tim, i ključa.

Ako je prisutno manje od *k* osoba, nema dovoljno podataka za utvrdivanje jedinstvenog polinoma. I pored toga, manje subverzivne grupe mogu da iskoriste svoje podatke za utvrdivanje relacija izmedu koeficijenata a_u što može da pruži nagoveštaj za vrednost ključa. Shamirov metod uklanja takve mogućnosti, tako što se sva izračunavanja izvode pomoću modularne aritmetike.

Diffie-HeHman razmena ključa

Diffie-Hellman razmena ključa funkcioniše tako što pošiljalac i primalac razmenjuju izračunate vrednosti na osnovu kojih može da se utvrdi vrednost ključa za šifrovanje.



SLIKA 7.12 *Diffie-Hellman razmena ključa*

U izračunavanjima se koriste drugi brojevi koji ne moraju da budu tajni. Na primer, pretpostavimo da se ljudi slože da koriste dva cela broja *g* i n za izračunavanje ključa za šifrovanje. Na slici 7.12 prikazana je razmena između osoba A i B i ilustrovano je ko šta zna.

Najpre A bira ceo broj *x* i izračunava i šalje vrednost g* mod *ndo B.* Slično tome, B nezavisno bira vrednost *y* i do A šalje vrednost g^y mod n. Ako treća strana prisluškuje njihovu konverzaciju, pretpostavimo da zna *g* i n, zajedno sa onim što se prenosi. LJ meduvremenu, A dobija g^y mod *n* i stepenuje tu vrednost na *x,* tako da se dobije g^{yx} mod *n. B* dobija g^x mod n i stepenuje tu vrednost na *y,* tako da se dobije g^{xy} mod *n.* Na osnovu svojstava modularne aritmetike, g^{yx} mod n je isto što i g^{xy} mod *n.* Znači, A i B koriste isti ključ za šifrovanje.

Sta je sa trećom stranom? Kao što smo istakli, ta osoba zna *g* i n i može da utvrdi vrednost g^x mod n i g^y mod *n,* osluškujući komunikacionu liniju. Medutim, pošto A i B kriju vrednosti *x* i *y,* treća strana ne može da kompletira izračunavanja neophodna za utvrđivanje ključa za šifrovanje. Ovo, samo po sebi, ne obezbeduje sigurnu liniju. Logično pitanje je da li treća strana može na osnovu informacija koje ima da zaključi koje su vrednosti za *x* i *y.* U stvari, ona mora da izračuna logaritam od g^x mod *n,* ili g^y mod *n.* Postoje uslovi koje *g* i n moraju da ispune da bi takva izvodenja bila veoma teška. Na kraju krajeva, i *g* i *n* moraju da budu veoma veliki brojevi (možda sa po hiljadu bitova). Ovde nećemo objašnjavati uslove koji moraju biti ispunjeni, jer bi nas to odvelo duboko u teoriju brojeva. Ako ste zainteresovani za ovu temu, pročitajte detaljnija objašnjenja u referenci [Sc94].

Diffie-Hellman razmena je osetljiva na problem koji je poznat kao man-in-the-middle napad. Kod ovog napada "uljez" se postavlja između osoba A i B. Kada A šalje g^x mod *n,* "uljez" presreće tu vrednost i menja je vrednošću $g^{x'}$ mod *n,* koju prosleduje do B. Toga nisu svesni ni A, ni *B.* Kada *B* pošalje g^y mod *n,* "uljez" ponovo presreće tu vrednost, menja je u $g^{y'}$ mod *n* i šalje do *A. Ni A,* ni *B* nisu svesni toga.

Osoba *A* vrši, zatim, šifrovanje ključem g^{xy} mod n, koji koristi i "uljez". Slično tome, *B* koristi ključ g^{xy} mod n koji "uljez" takode koristi za komuniciranje sa B. I A i B misle da komuniciraju, a, u stvari, "uljez" presreće sve njihove poruke, dešifruje ih i ponovo ih šifruje, pre nego što ih prosledi do druge strane.

7.4 Šifrovanje javnim ključem

Svi prethodno predstavljeni metodi šifrovanja imaju jednu zajedničku karakteristiku: ako neautorizovani primalac presretne šifrovani tekst i ako zna korišćeni algoritam za šifrovanje i ključ (£„), onda je lako utvrditi metod za dešifrovanje (D„). Na primer, ako je za šifrovanje korišćena Cezarova šifra, definisana dodavanjem vrednosti *k* na ASCII kodove originalnog teksta, dešifrovanje se vrši oduzimanjem vrednosti *k* od ASCII kodova šifrovanog teksta. Slični komentari mogu da se daju i za druge metode koje smo do sada predstavili.

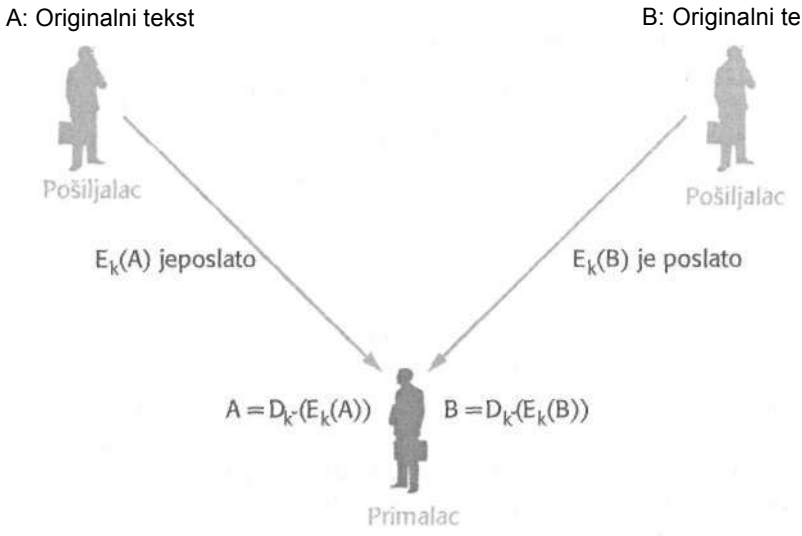
Defmitivno postoji opravdanje zbog koga poznavanje E„ čini trivijalnim dešifrovanje. Ipak, kao i kod mnogih drugih "stvari", to nije onako kako bi trebalo da bude. Diffie i Hallman (ref. [Di67]) su 1976. godine predložili korišćenje metoda za šifrovanje kod koga se algoritam za dešifrovanje i ključ ne mogu lako utvrditi, čak i ako su poznati algoritam i ključ za šifrovanje. Cilj je bio da se neautorizovana osoba koja poznaje algoritam i ključ za šifrovanje onemogući da dešifruje šifrovani tekst.

Postoji još jedna prednost ovakvih metoda. Pretpostavimo da neko treba da pribavi poverljive podatke iz više izvora (slika 7.13). Umesto da svi izvori koriste različite metode za šifrovanje, svi mogu da iskoriste isti metod E_k. Jedino primalac zna metod za dešifrovanje D_v. U stvari, E_k može da se učini javnim algoritmom. Pošto $D_{l,}$- ne može da se izvede na osnovu javnog ključa, nema opasnosti. Sve dok je D_v privatan, čak i različiti pošiljaoci ne mogu da dešifruju tude poruke, iako su svi koristili isti metod za šifrovanje.

Takvi sistemi se nazivaju **kriptosistemi sa javnim ključem.** Obično se koriste u bankama za prijem poverljivih finansijskih zahteva od klijenata, ili u vojnim komandnim centrima za pribavljanje izveštaja sa različitih lokacija. Često se koriste i na Web sajtovima koji nude usluge e-komerca.

RSA algoritam

RSA algoritam, nazvan po svojim kreatorima Rivestu, Shamiru i Adlemanu, a opisan u referenci [Ri78], koristi modularnu aritmetiku i faktorizaciju veoma velikih brojeva. Šifrovani tekst se iznenađujuće lako izračunava i veoma teško "razbija", čak i kada je £„ poznato. Jedan deo teorijske osnove ovog algoritma zasnovan je na matematičkoj teoriji brojeva, specifično u obliku rezultata poznatih pod nazivima Fermatova teorema i Ojlerova (Euler) teorema. Ovde nećemo zalaziti u raspravu o teoriji brojeva, ali, ako ste zainteresovani, u referencama [St95] i [MoOl] možete da pronadete pregled značajnijih teorijskih rezultata.



SLIKA 7.13 *Više pošiljalaca koristi isti metod za šifrovanje*

Da bismo opisali kako ovaj metod funkcioniše, razmotrićemo poruke koje se sastoje samo od velikih slova. Medutim, metod može da se generalizuje tako da uključi veđ skup karaktera. U narednim koracima opisan je RSA algoritam za šifrovanje i uključen je primer radi ilustracije funkcionisanja algoritma.

1. Pridružuje se prost kod slovima, kao što su 1 do 26 za A do Z, respektivno.
2. Bira se broj *n* koji će predstavljati proizvod dva velika prosta broja *p* i *q* (prosti brojevi nemaju faktore, osim same sebe i broja 1). U praksi, veliki prosti broj sadrži 200, ili više cifara. Medutim, mi ćemo, radi uštede prostora, uzeti primer n = p x ^ = 11 x 7 = 77.
3. Pronalazi se broj k koji je relativno prost u odnosu na *(p* - 1) x *[q* - 1). Dva broja su relativno prosta ako nemaju zajedničke faktore, osirn 1. U našem primeru biramo *k* = 7, što je relativno prost broj u odnosu na *[p* - *l)* x *(q* - 1) = 10 x 6 = 60. Broj *k* je ključ za šifrovanje. Možda ćete se zapitati možemo li uvek da pronađemo broj *k* koji ima ovakvo svojstvo. Odgovor je potvrdan. Dobro poznati rezultati u teoriji brojeva to dokazuju.
4. Poruka se deli na komponente. U opštem slučaju, svaka komponenta sadrži više slova da bi se izbeglo ponavljanje komponenata. Medutim, u našem primeru koristićemo samo jedno slovo po komponenti. Ako je poruka glasila "HELLO", komponente su H, E, L, L i O.
5. Za svaku komponentu nadovezuju se svi binarni kodovi za svako slovo u komponenti i niz bitova se interpretira kao ceo broj. Ovde svaka komponenta sadrži samo jedno slovo. Zato će celi brojevi biti 8, 5, 12, 12 i 15 (brojevi koji su originalno dodeljeni slovima).

6. Poruka se šifruje stepenovanjem svakog broja na stepen k. Medutim, sva aritmetika se izvodi po modulu n. U našem primeru to zahteva sledeća izračunavanja:

$$8^7 \bmod 77;\ 5^7 \bmod 77;\ 12^7 \bmod 77;$$

$$12^7 \bmod 77;\ 15^7 \bmod 77$$

Rezultat je šifrovana poruka. Ovde izračunavanja daju 57, 47, 12, 12 i 71, respektivno (ubrzo ćemo objasniti kako se vrše izračunavanja). Napomenimo da postoje dva broja 12 koja označavaju slovo koje se ponavlja. Ako komponenta sadrži nekoliko slova, ponavljanja poput ovog se izbegavaju.

Primalac dobija šifrovanu poruku 57, 47, 12, 12 i 71. Kako je dešifruje? Slede koraci metoda za dešifrovanje i nastavljamo primer.

1. Pronalazi se vrednost *k'* za koju je *k* x fe' - 1 = 0 mod *(p* - 1) x *[cj* - 1). To znači da je *{k* x *k')* - 1 jednako deljivo sa *[p* - 1) x *[q* - 1). Vrednost za *k'* je ključ za dešifrovanje. U ovom primeru *[p* -1) x *[cf* - 1) = 60 i fe' = 43, što lepo funkcioniše, tj. 7 x 43 - 1 = 300, što je deljivo sa 60. Možda ćete se zapitati može li se uvek pronaći vrednostJe'. Da! To je potvrđeno Ojlerovom i Fermatovom teoremom.

2. Svaki šifrovani broj iz koraka 6 stepenuje se na fe' i izvodi se aritmetika po modulu *n.* Rezultat su originalni brojevi iz koraka 5. U našem primeru ovo zahteva sledeća izračunavanja:

$$57^{43} \bmod 77;\ 47^{43} \bmod 77;\ 12^{43} \bmod 77;$$

$$12^{43} \bmod 77;\ 71^{43} \bmod 77$$

Rezultat su originalni brojevi 8, 5, 12, 12 i 15.

Korišćenjem ranije notacije $E_k(x) = x^k$ itod n i $D_k(y) = y^{k'}$ mod *n* imamo $D_{k'}(E_k(x)) = (x^k)^{k'}$ mod *n.* Sve dok su k i k' izabrani kao što smo o p i s a $(x^k)^{k'}$ n o d n daje *x.* Potvrda za to može da se nade u teoriji brojeva.

Algoritmi za šifrovanje i dešifrovanje su iznenadujuće jednostavni. Oba uključuju eksponencijalnu i modularnu aritmetiku. Medutim, kako se izračunava tačna vrednost broja kao što je 71^{43}? To iznosi otprilike 10^{79} i, u stvari, predstavlja veoma mali broj u poredenju sa onima koji se koriste u praksi. Ovo defmitivno deluje kao zastrašujuće izračunavanje. Nas ipak interesuje samo modularna aritmetika, tako da možemo da skratimo objašnjenja izračunavanja samo na ona koja možete da izvedete pomoću bilo kog kalkulatora. Uustrovaćemo izračunavanje 71^{43} mod 77.

Prvi korak je zapisivanje eksponenta kao sume stepena 2. Na taj način, dobijamo

$$71^{43} = 71^{32+8+2+1} = 71^{32} \times 71^{8} \times 71^{2} \times 71^{1} \qquad (7\text{-}1)$$

Sada, $71^2 = 5041 = 36 \bmod 77$. Ovo znači da 5041 i 36 imaju isti celobrojni ostatak kada se dele sa 77. Pošto jednačina 7.1 zahteva samo vrednost po modulu, možemo da zamenimo 71^2 sa 36. Osim toga, možemo da zapišemo 71^8 kao $(71^2)^4$.

Pošto nam je potrebna samo modularna vrednost, ovo je isto kao i 36^4. Slično tome, modularni ekvivalent za 71^{32} je $(71^2)^{16}$, ili 36^{16}. Tako se jednačina 7.1 redukuje na

$$71^{43} = 36^{16} \times 36^4 \times 36 \times 71 \quad \text{mod } 77. \qquad (7\text{-}2)$$

Kao što možete da vidite, značajno smo redukovali potrebna izračunavanja. Medutim, možemo da idemo i dalje. Nastavljajući na sličan način, imamo $36^2 = 1296 = 64 \text{ mod } 77$. Zato možemo da zapišemo $36^{"} = (36^2)^2 = 64^2 \text{ mod } 77$ i $36^{K'} = (36^2)^8 = 64^8 \text{ mod } 77$. Sada se jednačina 7.2 redukuje na

$$71^{43} = 64^8 \times 64^2 \times 36 \times 71 \quad \text{nod } 77 \qquad (7\text{-}2)$$

Naravno, mogli bismo da nastavimo ovaj proces tako da dobijemo

$$71^{43} = 64^8 \times 64^2 \text{ x } 36 \text{ x } 71 \text{ mod} 77$$

$$15^4 \text{ x } 15 \text{ x } 36 \text{ x } 71 \text{ mod } 77$$

$$71^2 \text{ x } 15 \text{ x } 36 \text{ x } 71 \text{ mod } 77$$

$$= 36 \text{ x } 15 \text{ x } 36 \text{ x } 71 \text{ mod } 77$$

$$= 15 \text{ mod } 77$$

Nema izračunavanja koja se ne mogu proveriti bilo kojim kalkulatorom.

RSA algoritam se relativno lako implementira, ali da li je bezbedan? Algoritam za šifrovanje zahteva *n* i *k,* a algoritam za dešifrovanje zahteva *n* i *k'.* Pretpostavimo da presretnete šifrovanu poruku i da znate n i *k.* Ne bi trebalo da bude teško da se utvrdi *k'.* Medutim, imajte na umu da se *k'* bira tako da je *(k* x *k')* x 1 = 0 mod *(p* - 1) x *(q* - 1). Zato je potrebno samo da pronadete *p* i Cj_1 faktore broja *n.* Ako je *n* veoma veliki broj, recimo od 200 cifara, ovo je veoma teško (ili bar zahteva mnogo vremena) da se izvede.*

Digitalni potpisi

Sledeća interesantna primena kriptosistema sa javnim ključem je za verifikaciju. Na primer, kada podižete novac iz banke, morate da popunite formular i da ga potpišete. Vaš potpis verifikuje Vaš identitet. Ako kasnije tvrdite da nikada niste naložili podizanje novca, banka može da Vam izda forraular sa Vašim potpisom. Naravno, uvek možete da tvrdite da je to falsifikat i da tužite banku, U slučaju da odete na sud, banka može da dovede grafologa koji će da potvrdi da je potpis zaista Vaš. Tako ćete izgubiti parnicu i službenici za zajmove u banci verovatno neće prihvatiti Vaš zahtev za hipoteku za novu kuću.

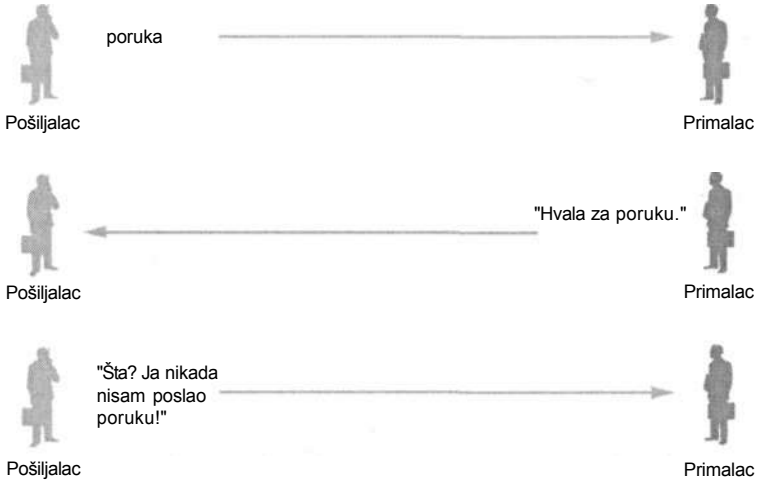
* Možda če Vas interesovati izazov na RSA sajtu (www.rsasecurity.com/rsalabs/challenges/factoring /numbers.html), gde se nudi novćana nagrada za faktorizaciju velikih brojeva. Na primer, u vreme kada smo pisali ovaj tekst u RSA laboratorijama je nudena nagrada od 200.000 dolara za faktorizaciju 617-cifrenog (2048-bitnog) broja
25195908475657893494027183240048398571429282126204032027777137836043662020707595556264018525880784406918290641249515082189298559149176184502808489120072844992687392807287776735971418347270261896375014971824691165077613379859095700097330459748808428401797429100642458691817195118746121515172654632282216869987549182422433637259085141865462043576798423387184774447920739934236584823824281198163815010674810451660377306056201619676256133844143603833904414952634432190114657544454178424020924616515723350778707749817125772467962926386356373289912154831438167899885040445364023527381951378636564391212010397122822120720357. Da li je neko zainteresovan?

Sada razmotrite nešto drugačiji scenario. Šaljete elektronski zahtev za Vaš račun u švajcarskoj banci za transfer veće svote novca na račun Vaše bivše supruge. Sta banka može da preduzme ako kasnije budete tvrdili da nikada niste poslali takav zahtev, posebno ako je Vaša bivša supruga u međuvremenu podigla sav novac i preselila se u Boliviju? Na fajlu ne postoji rukom napisani potpis koji grafolog može da analizira. Banka može da potvrdi da je za autorizaciju zahteva morala da bude uneta lozinka koju samo Vi znate. Naravno, lozinka je čuvana i u nekim kompjuterima u banci da bi se mogla utvrditi njena autentičnost. Možete da tvrdite da je neko uzeo Vašu lozinku iz banke i da Vam nije obezbedena odgovarajuća zaštita.

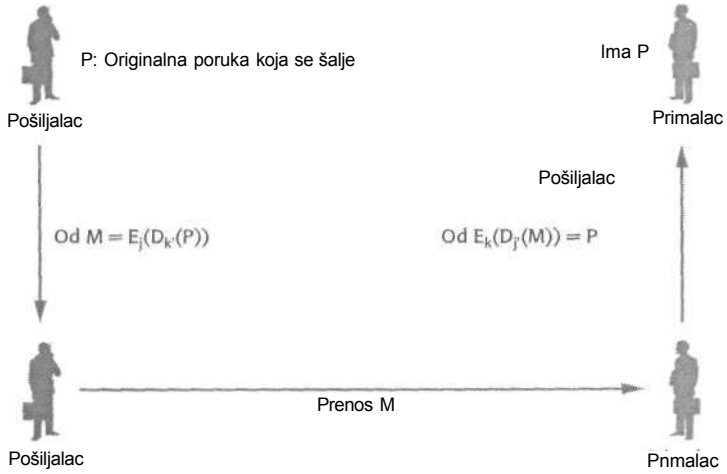
Da li postoji način da banka dokaže da nije bilo nikakve greške i da ste Vi postavili zahtev? Na slici 7.14 ilustrovan je opšti problem. Neko šalje poruku, pa dobija odgovor, ali, zatim, tvrdi da je nikada nije poslao. Može li primalac lažno da svedoči? Verifikovanje identiteta pošiljaoca naziva se autentifikacija.

Jedan od mogućih metoda autentifikacije podrazumeva korišćenje digitalnih potpisa. U suštini, to uključuje šifrovanje poruke tako da način šifrovanja zna samo pošiljalac. To je slično šifrovanju lozinkama, samo što se i one smeštaju u fajlovima primaoca radi verifikacije. Ključ za šifrovanje poseduje samo pošiljalac. Pošiljalac može da tvrdi da mu je neko ukrao ključ, ali, pošto primalac nema nikakav zapis ključa, on ne može da bude optužen za kradu. Ovo je kao da ste izgubili ključeve od svoje kuće. U krajnjem slučaju, sami ste odgovorni.

Na slici 7.15 pokazano je kako se šalje šifrovana poruka koja sadrži digitalni potpis. Metod koristi dva para metoda šifrovanja javnim ključem/dešifrovanja. Označavamo ih kao (E_k, D_k) i (E_j, D_j'), gde su *j* i *k* javni ključevi, a fe' (koji zna samo pošiljalac) i j' (koji zna samo primalac) privatni ključevi.



SLIKA 7.14 *Pošiljalac demantuje Aa je poslao poruku*



SLIKA 7.15 *Slanje poruke uz korišćenje digitalnog potpisa*

Osim toga, parovi treba da lmaju sledeca svojstva:

$$E_{11}(DUP)) = D_{11}(HP)) = P \quad \text{i} \quad HD_1(P)) = D_1(E_1(P)) = P$$

Već smo istakli da šifrovanje prati dešifrovanje koje daje originalnu poruku, ali mora da važi i obratno, tj. najpre dešifrovanjem, pa šifrovanjem mora opet da se dobije original.

Pretpostavimo da pošiljalac želi da pošalje šifrovanu poruku i da se identifikuje. Ako je P obična tekstualna poruka, pošiljalac izračunava $Ej(D_k\{P))$ i šalje rezultat. * Primalac primenjuje *Df* na poruku. Pošto su *Df* i *Ej* inverzne operacije, rezultat je D,,.(P).' Primalac pamti $Di_1(P)$ u slučaju da pošiljalac eventualno demantuje da je ikada poslao poruku. Zatim, primalac primenjuje E_k, tako da se dobije £,,(D,,(P)) = P i poruka je primljena.

Pretpostavimo sada da pošiljalac demantuje da je poslao poruku. Da bi bio potvrden identitet pošiljaoca, primalac arbitru (nekome ko će da utvrdi ko laže) obezbeduje i $D_k[P)$ i P. Arbitar primenjuje E_1, (šifrovanje javnim ključem) na $D_{11}(P)$ i dobija P. Tako se dokazuje da je poruka P izvedena iz $D_k(P)$. Osim toga, pošto je $D_k(P)$ utvrdeno pomoću privatnog ključa koji se ne izvodi iz £„ arbitar zaključuje da je $D_{/;}${P) mogao da konstruiše samo neko ko je imao privatni ključ fe'.

Pošto je pošiljalac jedina osoba koja to zna, samo on može da se okrivi.

* Kasnije ćemo u ovom poglavlju predstaviti digitalne sažetke poruke (message digest), brojeve koji su jedinstveni za svaku poruku. Često proces izračunava vrednost digitalnog sažetka poruke i primenjuje Dk' na tu vrednost. Ako je ta vrednost pravilno izračunata, ovo je siguran način da se dokument potpiše i autentifikuje.

t Ako je samo ID pošiljaoca bio promenjen samo prvi put, primalac sada ima običan tekst i ID koji je izmenio D_k..

Kao što ste mogli i da pretpostavite, ovo nije jedini način za kreiranje digitalnog potpisa - u referenci [KaOlQ možete da pronadete poređenja i razlike izmedu nekih drugih šema. Osim toga, pošto postoje stanadardi za skoro sve ostalo, postoji i jedan za digitalne potpise. NIST FIPS publikacija 186-2 (http://csrc.nist.gov/publications/fips/fips816-2/fipsl86-2.pdf) daje pregled standarda Digital Signature Standard (DSS). Tehnika propušta originalnu poruku kroz hash algoritam (predstavićemo ga uskoro) pod nazivom SHA-I (Secure Hash Algorithm 1, definisan u FIPS publikaciji 180-1, http://csrc.nist.gov/publications/fips/fipsl80-l/fipsl80-l.pdf), koji utvrduje jednistveni broj za poruku. Taj broj se koristi kao ulaz za algoritam šifrovanja privatnim ključem radi formiranja digitalnog potpisa. Naravno, postoje razlozi za ovakav pristup i detalji koje moramo da obradimo. Ovo je naša sledeća tema.

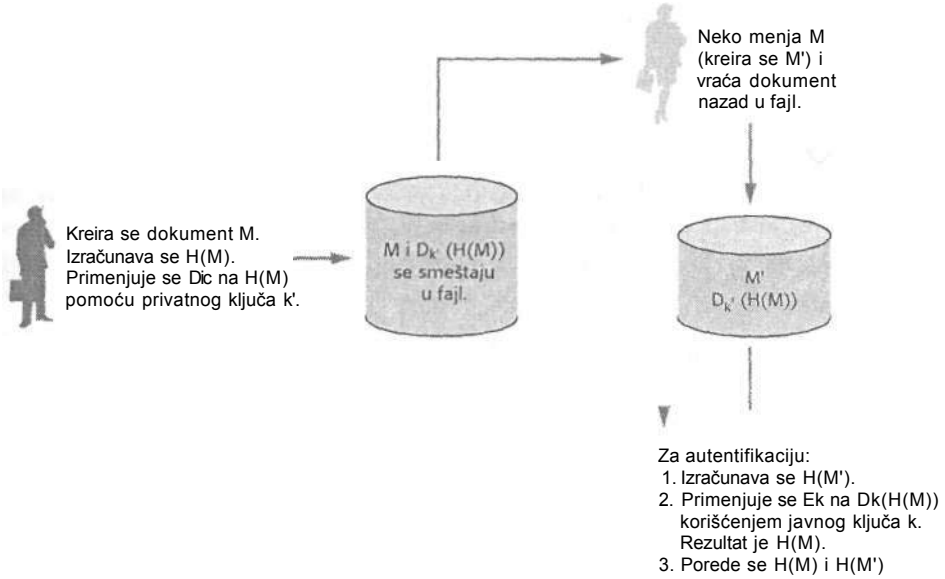
Autentifikacija i digitalni sažetak poruke

Mogućnost autentifikacije pošiljaoca poruke je sigurno značajna u eri elektronskog transfera. Medutim, pomoću metoda koji smo prethodno opisali vrši se autentifikacija šifrovanjem cele poruke. Zato se gubi jasna granica izmedu obezbeđivanja zaštite i autentifikacije. U nekim slučajevima ne brine nas toliko skrivanje sadržaja poruke koliko autentičnost. Ovo je korisno u slučajevima kada se u poruci nalaze ugovor, pismo preporuke, ili bilo šta sa zakonskim implikacijama. Da li postoji način da se osigura da kada se dokument dobije u elektronskoj formi ne može da se promeni bez detekcije?

Jedan pristup koristi **hash funkciju** *H* (još se naziva i *message digest* - digitalni sažetak poruke) za dodeljivanje jedinstvene vrednosti fiksne dužine dokumentu. Ako je M dokument, onda H(M) predstavlja **hash vrednost,** ili **vrednost digitalnog sažetka poruke** (message digest). Zatim se primenjuje algoritam dešifrovanja sa privatnim ključem, predstavljen sa D_v na H(M), i smešta se rezultat D,,{H(M)), zajedno sa dokumentom (slika 7.16). Ako je reč o autentifikaciji dokumenta, onda se radi sledeće:

1. Izračunava se vrednost digitalnog sažetka dokumenta sa kojim se radi.
2. Primenjuje se algoritam šifrovanja javnim ključem na dešifrovanu vrednost digitalnog sažetka D,,{H(M)), koja je smeštena zajedno sa dokumentom - koristi se algoritam šifrovanja javnim ključem koji odgovara agoritmu za dešifrovanje sa privatnim ključem primenjenim na H(M). Rezultat je £,,(D,,(H(M))) = H(M).
3. Ove dve vrednosti se porede. Ako se ne slažu, neko je "petljao" nešto sa dokumentom.

Ovo jednostavno objašnjenje nameće logično pitanje koliko Je teško nešto promeniti u dokumentu, a da se ne promeni vrednost digitalnog sažetka poruke? Na primer, prosta hash funkcija sumira vrednosti bajtova (interpretirane kao celi brojevi) u okviru dokumenta. Promena dva bajta može da izmeni sadržaj dokumenta, a da vrednost digitalnog sažetka ostane ista. Ovo je od izuzetnog značaja za razvoj metoda za autentifikaciju.



SLIKA 7.16 *Autentifikacijd dokumenta*

Odgovor leži u kreiranju jednosmernih hash funkcija. Jednosmerna hash funkcija H ispunjava sledeće uslove:

- Neka M bude poruka, ili dokument proizvoljne dužine. Onda je *H* funkcija koja dodeljaje jedinstvenu vrednost fiksne dužine za M. Matematički, to se zapisuje kao H(M) = V.
- H(M) se lako izračunava. Ovo je važno za razvoj efikasnog algoritma.
- I pored konkretne vrednosti V, teško je utvrditi M za koje je H(M) = V. Ovaj kriterijum je razlog za jednosmernu kvalifikaciju. Direktna implikacija je to što je, ako je M poruka i V = H(M), teško pronaći drugu poruku M' za koju je H(M') = V.
- Teško je pronaći dve poruke M_1 i M_2 za koje je $H(M_1) = H(M_2)$. Ovo možda deluje isto kao i prethodni kriterijum, ali, kao što ćemo uskoro pokazati, postoji značajna razlika.

Moguće je postaviti pitanje neophodnosti ovako strogih uslova hash funkcije. Na kraju krajeva, čak i ako poruka, ili dokument mogu da se promene bez izmene vrednosti digitalnog sažetka poruke, zar promena ne bi bila primetna? Odgovor je odrečan. U dobro poznatom đanku Yuval [Yu79] je 1979. godine opisao tzv. *napad rodendana* (birthday attack), tehniku koja može da se koristi za generisanje dva dokurnenta sa istom vrednošću digitalnog sažetka, ako ta vrednost nije dovoljno jaka.

Naziv tehnike ima veze sa dobro poznatim problemom iz teorije verovatnoće. Problem je

> Ako se u prostoriji nalazi proizvoljan broj ljudi, koliko njih mora da bude pristuno da bi verovatnoća da još neko ima rodendan kad i Vi bila veća od 0.5?

Odgovor je polovina ukupnog broja dana u godini, ili 183.

Sličan problem glasi ovako:

> Ako se u prostoriji nalazi proizvoljan broj ljudi, koliko njih mora da bude prisutno da bi dve osobe imale rodendan istog dana sa verovatnoćom većom od 0.5?

Za mnoge je često iznenađenje što odgovor glasi 23. Razlog za manju vrednost je to što se u ovom slučaju rodendan "ne vezuje" za tačno odredeni datum.

Ovo je specijalni slučaj opštijeg problema, koji može da se izrazi na sledeći način:

- Neka su X = (X_1, X_2, X_3, ..., x_k} i Y = {y_{lf} y_2, y_3, ..., yi,} dva skupa brojeva.
- Svaki broj je nasumično izabran i nalazi se izmedu 1 i *T",* respektivno (m je neki pozitivni ceo broj).
- Neka je *P(m, k)* verovatnoća da X i Y imaju najmanje jedan zajednički broj. Drugim rečima, X_1 = y, za neko *i* i j.
- Koliko mora da bude *k* da bi *P[m, k)* > 0.5?

Ispostavlja se da je odgovor na ovo pitanje k » $2^{m/2}$. Ovde nećemo navoditi matematički dokaz za to, ali, ako imate dovoljno dobro predznanje iz matematike, detalje možete da proverite u referenci [St95].

U kakvoj je ovo vezi sa autentifikacijom dokumenata? Pretpostavimo da beskrupulozna osoba X radi u gradskom zavodu za građevinsku inspekciju i da pomaže objavljivanje liste inspekcijskih izveštaja - priprema svaki izveštaj na osnovu izveštaja svakog inspektora i prosleđuje ga do inspektora radi odobrenja. Nakon odobrenja, inspektor pokreće program za izračunavanje vrednosti digitalnog sažetka poruke i vraća dokument do X radi eventualnog objavljivanja. Pretpostavimo da neki projekat nije dobio prolaznu ocenu inspektora, ali je X potplaćen da i, pored toga, objavi pozitivan izveštaj. Kako X to može da izvede?

Pretpostavimo da message digest algoritam generiše vrednost između 1 i 2^{n}[4]. Napad rođendana se ogleda u tome što X priprema dva izveštaja. Jedan je validan, ali nepoželjan, a drugi nije validan, ali je poželjan za izvodača. X može da kreira nekoliko varijadja svakog izveštaja, koji bi, u suštini, imali isti sadržaj. Na primer, X može da pronade nekoliko sinonima za neke reči, ili na nekim mestima zamenice može da zameni odgovarajućim imenicama. Cak i zamena dva uzastopna blanko znaka daje promenu originalnog izveštaja. Ako X može da identifikuje 32 mesta u svakom izveštaju na kojima je moguće izvesti zamene, postoji 2^{32} varijacija svakog izveštaja. Toliko postoji mogućih kombinacija zamena.

Ako se message digest algoritam primeni na svaki izveštaj, generiše se 64-bitni broj, a postoji šansa 50:50 da X može da pronade validan i nevalidan izveštaj, generišući istu vrednost digitalnog sažetka. Sa stanovišta izračunavanja, to ne bi trebalo da bude preterano teško. Naravno, 2^{32} je veliki broj mogućih varijacija za ručno izračunavanje digitalnog sažetka, ali za kompjuterski program to nije nikakav problem.

Zato X može da obezbedi validan izveštaj za inspektora, koji će generisati vrednost digitalnog sažetka. Kada X dobije nazad validni izveštaj, on može da zameni nevalidni izveštaj istom vrednošću digitalnog sažetka i da tako objavi lažni izveštaj.

Nedostatak ovog procesa nije u samoj šemi, već u veličini broja digitalnog sažetka. Na primer, ako su vrednosti dva digitalna sažetka bili 128-bitni brojevi, verovatnoća pronalaženja dva podudarna broja iz 2^{32} varijacija svakog izveštaja je izuzetno mala. U stvari, X treba da ima 2^{M} verzija svakog dokumenta da bi postojala verovatnoća 50:50 da će pronaći poklapanje. Ogromno vreme koje je neophodno za ovoliko pokušaja čini prevare skoro neizvodljivim. Naravno, ova poslednja ocena gubi na verodostojnosti kako se proizvode sve brži i moćniji kompjuteri.

Sledeće logično pitanje je koje su vrste šema za hašingovanje na raspolaganju. Jedan takav algoritam je *MD5 algoritam,* koji je razvio Ron Rivest u MIT (ref. [Ri92]). Algoritam generiše 128-bitne vrednosti digitalnog sažetka poruke. Poruka se deli na 512-bitne blokove (neka povećanja mogu da budu neophodna da bi se dobio kompletan 512-bitni blok) i vrše se operacije na svakom bloku. Svaki blok prolazi kroz četiri ciklusa različitih operacija na nivou bitova i faktorizacije vrednosti kroz sinusnu funkciju. Eventualno, generiše se rezultat koji može da se koristi kao ulaz za šifrovanje sledećeg bloka. Kao i kod ranije predstavljenih šifrovanja blokova, reč je o složenom procesu i ovde se nećemo baviti detaljima; ako ste zainteresovani za detaljnija objašnjenja, pogledajte reference [St95] i [Sc94]. Sledeći algoritam je **Secure Hash Algorithm (SHA-I),** koji su razvili NSA i NIST. Bezbedniji je po tome što daje 160-bitnu vrednost digitalnog sažetka. Kao i MD5, radi sa 512-bitnim blokovima.* Proces uključuje najpre deljenje 512-bitnog bloka u 16 32-bitnih reči. Nakon toga se grupe reči i unapred definisane konstante izlažu brojnim ciklusima operacija logičkog I, ILl, isključivog ILI i operacijama pomeranja, tako da se na kraju dobija 160-bitna vrednost. Više detalja o ovoj temi može da se pronade u referencama [St03] i [Sc94] i na Web sajtu ttp://csrc.nist.gov/publications/fips/fipsl80-l/fipsl80-l.pdf.

Program Pretty Good Privacy

Jedna od najčešće korišćenih Internet aplikacija (ako ne i najčešće korišćena) je email. Iako se obično koristi za razmenu neformalnih informacija, da bi se održao kontakt sa prijateljima, ima izuzetno važnu ulogu u brojnim poslovnim operacijama. Zbog toga, postoji velika potreba da se informacije iz emaila zaštite i da se izvrši autentifikacija izvora i sadržaja email poruka. Osim toga, pošto se korisnici emaila oslanfaju na mnoštvo različitih email paketa i sistema, neophodno je obezbediti opcije za šifrovanje i autentifikaciju koje neće zavisiti od platforme, ili paketa. Jedan takav program koji pruža tražene opcije naziva se **Pretty Good Privacy (PGP).**

PGP je program za zaštitu emaila koji je napisao Philip Zimmerman i možete da nabavite njegovu komercijalnu, ili freeware verziju (posetite www.pgpi.org/ i www.pgp.com/). PGP uključuje šifrovanje javnim ključem, autentifikaciju, digitalne potpise i kompresiju.

* FIPS publikacija 180-1 definiše i pravila za proširivanje poruke ako njena ukupna dužina nije deljiva sa 512 bez ostatka.

Pokreće se na raznim platformama i koristi algoritme koji su detaljno provereni, kao što su RSA za šifrovanje javnim ključem, SHA-I i MD5 za digitalne potpise i IDEA algoritam za regularno šifrovanje (još jedan algoritam za šifrovanje blokova; videti referencu [Sc94]).

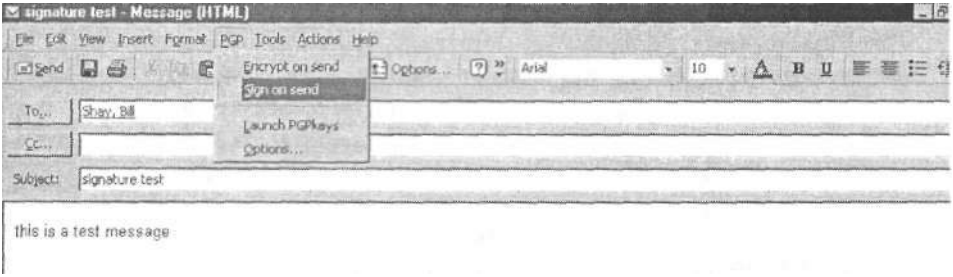
Kada je prvi put razvijen, PGP je bio pomalo ozloglašen. Pošto ga je neko postavio na Internet, bio je javno dostupan širokoj svetskoj zajednici. Problem je bio to što je početkom 90-ih prošlog veka Vlada Sjedinjenih Američkih Država smatrala algoritme za šifrovanje čiji su ključevi imali više od 40 bitova vojnom opremom. Svrstavala ih je u istu kategoriju kao i sva vojna sredstva (oružje i municija), stvari koje Vlada ne bi htela da vidi kao nešto što se javno izvozi. Upotreba tih algoritama izvan SAD smatrana je pretnjom za nacionalne interese i zato je podlegala strogim zakonima za izvoz. Vlada je zaključila da je dostupnost PGP-a preko Interneta omogudla njegov "izvoz" u strane države i da su time narušeni zakoni o izvozu. Radi rešavanja tih problema, neke novije verzije ovog programa su izrađene izvan Sjedinjenih Američkih Država. Od tada nema zakonskih problema, ali i dalje postoje slučajevi u kojima proizvodi koji koriste kriptografske tehnike podležu režimu izvoznih licenci. Bureau of Export Administration je izvršio nekoliko promena polise da izvoznici iz SAD ne bi bili oštećeni u zonama slobodne trgovine koje omogućavaju izvoz u nevladine institucije Evropske unije. Ipak, i dalje postoje neka ograničenja u vezi izvoza softvera za šifrovanje u zemlje koje su Sjedinjene Američke Države proglasile zemljama koje podržavaju teroriste. To je komplikovan problem i nema sumnje da će se politika menjati u skladu sa tekućim svetskim dogadajima. Zainteresovani čitaoci mogu da pročitaju nešto više o tim promenama na Web sajtu www.bxa.doc.gov/Encryption.

U svakom slučaju, naša namera je da ovde opišemo neke glavne karakteristike PGP-a i način njegovog funkcionisanja. Kao što smo prethodno istakli, PGP može da se preuzme kao freeware i može da se instalira na brojnim platformama, kao što su LlNIX, Windows 2000/ME/XP i Macintosh. Dizajniran je da se koristi kao plug-in za nekoliko paketa za email. Na primer, ako za email koristite Microsoft Outlook, interfejs se modifikuje tako da možete da koristite mogućnosti PGP-a bez napuštanja Outlooka.

Na slici 7.17 prikazan je primer zasnovan na PGP freeware verziji 6.5.8, koja je pokrenuta na Windows sistemu; kod drugih verzija mogu da postoje neke manje varijacije. Pretpostavimo da želite da pošaljete poruku (tekst: *This is a test message)* i da ste je potpisali digitalnim potpisom. Poruku konstruišete na uobičajeni način. Medutim, ako Je PGP instaliran kao plug-in, postoji novi ulaz, označen kao PGP, kome možete pristupiti pomoću opcija padajućeg menija. Ova stavka nije postojala pre instaliranja PGP-a. Selektovanjem stavke PGP možete da vidite nekoliko opcija, medu kojima je i Sign on Send (videti sliku 7.17). To znači da će PGP Vašoj poruci pridružiti digitalni potpis nakon što kliknete Send, ali pre nego što Outlook stvarno pošalje poruku.

Pretpostavimo sada da ste tačno to i uradili. Kada primalac otvori poruku, videće sledeću poruku, koja ukazuje da Je poruka digitalno potpisana.

```
---BEGIN PGP SIGNED MESSAGE---
Hash: SHA1

this is a test message
---BEGIN PGP SIGNATURE---
```



SLIKA 7.17 *Pristupanje optijama PGP-a iz Microsoft Outlooka*

```
Version: PGPfreeware 6.5.8 for non-commercial use
<http://www.pgp.com>

iOVAwUBPVHAkz012x9/xPKqEQL7UQCg65yJ8I4c5o7s37iMvLcqq-
RtokhAAn3E2
EzQd3vhFE41QGj308zvDSawR
= knKs

---END PGP SIGNATURE---
```

Dok je poruka otvorena, primalac može da selektuje padajud meni PGP i izabere opciju Decrypt/Verify (koja nije prikazana na slici 7.17). Nakon biranja te opcije, prozor u kome se poruka nalazi menja se u oblik koji podseća na naredni prikazani primer, ukazujući na validnost potpisa.

```
*** PGP Signature Status: good
*** Signer: William Shay <shayw@uwgb.edu>
*** Signed: 8/7/02 7:51:31 PM
*** Veri.ed: 8/7/02 8:50:46 PM
*** BEGIN PGP VERIFIED MESSAGE ***
this is a test message
*** END PGP VERIFIED MESSAGE ***
```

Sledeći sličnu proceduru, možete da šifrujete svoje email poruke. Sastavite email pomku i selektujte padajući meni PGP, ali ovoga puta izaberite opciju Enciypt on Send (videti sliku 7.17). Osoba koja prima poruku u tom slučaju vidi nešto slično sledećim linijama.

```
---BEGIN PGP MESSAGE---
Version: PGPfreeware 6.5.8 for non-commercial use
<http://www.pgp.com>

qANQR1DBwU4D1bHGsRJGiAYQCACzq5cJFQTYx/CgqG6lK.blkBArlOYTIAE+M
xoVAuK1LsrNu1HHkUkRCrEH2eHwVwl9FngHvCOxo0cniiI5GRhOE02EjOeMqeO
PYasuAWvWDGLQykNF6xqAeGWAWIt0PUr3PANIUshqS5ss7TQ1mG5KgpoKMt-
```

```
SzBXqDk]2nS/Dm01m/tDf+OyqRnYnS/1nZ8xHCilyQ0seDuiNSPVYpVeah550
iZfZHF7gkstd7+dSZYISuy0SeRWNUii0VW8tDfFUwpb/X0UtPH/z22waG-
wiW8id1Vz07Zm1zSM6Kw1Ma8RUf82xKM90em7pv74u/zJu8zl97oGrsJBt9MQG
4/TKRgCACFAuSzyOmsUZrNWqvS1AuRYe9zY27ktXi5THt88PfB+qE855fzx0f
Axh8ovr564Cz4duK0cPcdIhIQre1guZdq71 SMBGzOdReDkOkXSfZ5r7snUP.AN
q4ks8K1BozB5I5irSTMs62Yj U08RRDXJ1URSRgS8t0yonXpiQj M6+eSz2QT0X
tWF29klFQLeA6rX3q1zp2nhNwIdt3cI2xb09OTRKTrdqtbVcske2gzsKfDL3JO
FJJuK/2KZ9kcTxaiWC7WV2auLwqZ9Jrts8FwRD0B15n2hE4V+Bi0nTMn/u+XZ
MGHGVblRqFlZcHP0MzDrIVTa8jxhzLHQ3KHZF7K75GyS29LD8EY2RRLKb-
VZyz6odIxokEfHbqZV22sP7sC8IsydvQ9wHbsIjavXi/NeqU=
=3+11

---END PGP MESSAGE---
```

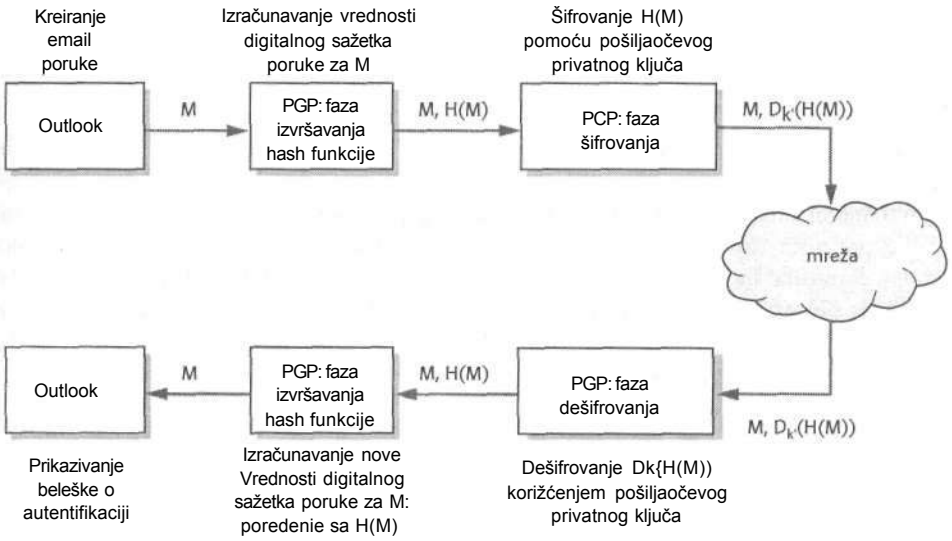
Kada primalac selektuje opciju Decrypt/Verify iz PGP menija, dobija originalnu poruku:

this is a test message

Iako su ovde šifrovanje i dodavanje digitalnog potpisa prikazani kao zasebni primeri, moguće je izvesti oboje u okviru iste poruke. Jednostavno, izaberete obe opcije iz PGP menija. Osim za email aplikacije, šifrovanje i potpisivanje, PGP može da se primeni direktno i na fajlove. Na operativnom sistemu Windows možete da kliknete desnim tasterom miša naziv fajla i videćete opciju PGP. Nakon selektovanja PGP opcije, prikazaće se moguće opcije za šifrovanje, potpisivanje, ili oboje istovremeno. Možete i da *uklonite* (wipe) fajl. Ovo je slično njegovom brisanju. Ipak, kod normalnog brisanja uklanjaju se samo reference na podatke iz fajla. Stvarni podaci i dalje ostaju netaknuti i obično mogu da se povrate upotrebom nekih pomočnih programa za održavanje diska. Ovo je korisno ako slučajno izbrišete fajl. Osim toga, ovim se pomaže pripadnicima službi za očuvanje reda i zakona kada treba da oporave fajlove sa nelegalnim materijalom (kao što je dečija pornografija) sa kompjutera koji su koristili kriminalci. Kada uklonite fajl, podaci se uklanjaju sa drajva. Ne postoji način da se povrate.

PGP se sasvim sigurno lako koristi, ali kako on funkcioniše? Pre svega, i pošiljalac i primalac moraju da instaliraju PGP. Sledi problem ključeva. Kada prvi put instalirate PGP, instalaciona rutina nudi opciju za generisanje para privatnog/javnog ključa. Nakon instaliranja PGP-a, možete da generišete i novi par privatni/javni ključ pokretanjem pomoćnog modula PGPkeys. Kada se modul pokrene, tražiće da izaberete tip ključa (na primer, RSA, ili Diffie-Hellman/DSS), veličinu ključa i datum isteka. Tako možete da prilagodite PGP svojim potrebama. Osim toga, zahtevaće da izaberete *frazu propusnicu* (pass phraze), string koji obezbeduje pristup privatnom ključu. Ovo je bezbednosni mehanizam koji samo Vama omogućava pristup privatnom ključu. Program zahteva da unesete frazu propusnicu svaki put kada digitalno potpišete dokument sa privatnim ključem, ili dešifrujete i šifrujete poruke.

Kada PGP generiše par ključeva, snima ih u takozvani *keyring* (određeni fajlovi na hard disku). Privatni ključevi su smešteni u fajl pod nazivom *secring.skr,* a javni ključevi se smeštaju u fajl pod nazivom *pubring.pkr.* Javni ključ može da se postavi na server tako da i drugi korisnici mogu da mu pristupe. Alternativno, javni ključ može da se eksportuje u fajl, a zatim da se pošalje do bilo koga sa kim želite da razmenjujete bezbedne email poruke, ili čak mogu da se "uskladište" na Vašem Web saitu.



SLIKA 7.18 *Korišćenje PGP-a za autentifikaciju poruke kreirane u Outlooku*

Kada PGP potpiše poruku, obavlja dve "stvari" (slika 7.18): izračunava vrednost digitalnog sažetka poruke H(M) za poruku M, koristeći SHA-I algoritam, i ta vrednost se šifruje pomoću pošiljaočevog privatnog ključa. Tako se generiše D^(H(M)) sa slike 7.18. Kada se prime poruka i šifrovana vrednost digitalnog sažetka, izvode se dve komplernentarne akcije. Prvo se vrednost digitalnog sažetka dešifruje pomoću pošiljaočevog privatnog ključa (koji primalac takode mora da ima) i ponovo se dobija H(M). Zatim se H(M) poredi sa izračunatom vrednošću na strani primaoca. Ove dve vrednosti moraju da se poklope da bi bila potvrđena autentičnost poruke.

Naravno, postoji još mnogo štošta što može da se kaže o PGP programu. Kako se kontrolišu keyring fajlovi? Kako se preuzima javni ključ koji je postavljen na serveru? Koje još algoritme za šifrovanje PGP podržava? Kako je moguće utvrditi da li je osoba koja obezbeduje javni ključ ona za koju se predstavlja? Ovde postoji prostor za eventualne prevare. Lista pitanja može da se nastavi. Mnoga od njih su specifična za PGP i upućujemo Vas na informacije iz izvora koji se više bave njima (reference [Ga94], [Zi95] i [OpOl] iWebsajtwww.pgpi.org/).

7.5 Zaštita na transportnom sloju i autentifikacija servera

Zbog ogromnog broja kompanija koje danas svoje poslove obavljaju preko Interneta, zaštita je postala jedan od najvažnijih problema. Možda ste nekada nešto kupovali preko Web sajta unošenjem broja svoje kreditne kartice u pretraživač i klikom na odgovarajuće dugme. Možda ste uplaćivali, ili podizali novac sa Vašeg bankovnog računa. Naravno, tada ste morali da unesete broj računa i PIN.

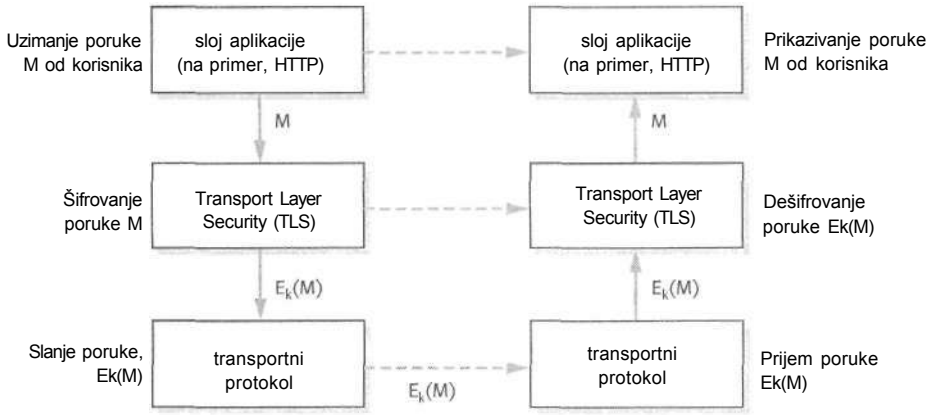
Svaki put kada unosite privatne informacije iz sigurnosti svog doma, kako znate da neko ne nadgleda Vašu konekciju i ne kopira lnformadje? Sajtovi kojima pristupate moraju da garantuju sigumost ovakvih privatnih informacija. Pitanje je kako mogu da osiguraju ono što im šaljete iz privatnosti svog doma.

S obzirom da je mnogo pažnje posvećeno šifrovanju, logično je da se rešenje traži upravo u metodima za šifrovanje. Medutim, to nameće druga pitanja. Na primer, kako dobiti pristup odgovarajućem metodu za šifrovanje i ključevima iz privatnosti svog doma? Morate ih imati lokalno pre nego što pošaljete privatne informacije posredstvom Internet provajdera. Ipak, nemojte da mislite da jedini oblik prevare uključuje nekoga ko prati Vašu konekciju. Kako da znate da je sajt na koji se povezujete stvarno legitiman? Samo zato što sajt ima lepo odraden grafički dizajn i polja za unos broja kreditne kartice ne znači da je reč o legitimnoj firmi. Kako da budete sigurni da operatori Web sajta neće uzeti broj Vaše kartice i nestati?

Odgovor na ova pitanja daju protokoli kao što su Secure Sockets Layer **(SSL)** i **Transport Layer Security (TLS)** i X.509 sertifikati. X.509 sertifikat obezbeđuje standardnizovani sadržaj za informacije koje se koriste za autentifikaciju servera; ubrzo ćemo i njih predstaviti.

Zaštita na transportnom sloju

I SSL i TSL su protokoli koji se nalaze izmedu sloja aplikacija predstavljenog HTTP protokolom (koriste ga pretraživači) i transportnog sloja TCP (Transport Control Protocol of the Internet). SSL je originalno razvio Netscape. Dva najvažnija cilja pri uvodenju tog protokola bila su obezbeđivanje pomočnih rutina za šifrovanje radi bezbedne razmene privatnih informacija i obezbeđivanje autentifikacije servera tako da korisnici mogu da smatraju da server jeste ono za šta se "izdaje". Na slici 7.19 prikazano je gde se uklapa TLS (ili SSL). Pošto se TLS (ili SSL) umeće izmedu slojeva, on izvodi šifrovanje poruke M koja se generiše na sloju aplikacije.



SLIKA 7.19 *TLS (Transport Layer Security) protokol*

Rezultat $E_k(M)$ se prosleđuje do transportnog sloja, koji ga prosleduje do transportnog sloja na strani servera. Transportni sloj prenosi $E_k(bA)$ do TLS-a, gde se dešifruje nazad u M i prenosi do aplikacije na serveru.

Razvijeno je nekoliko verzija SSL protokola; svaka je donela nova poboijšanja i veći broj opcija u odnosu na prethodnu. Osim toga, TLS je razvio IETF kao eventualnu zamenu za SSL. Prema IETF, razlike između ovog protokola i SSL 3.0 nisu dramatične, ali su dovoljno značajne da TLS 1.0 i SSL 3.0 ne mogu zajedno da funkcionišu. Ovde se nećemo baviti tim razlikama, ili razlikama između različitih SSL verzija, ali ćemo dati neke kraće komentare. Osim tih komentara, nećemo navoditi nikakve druge razlike izmedu ovih protokola; zainteresovane čitaoce upućujemo na reference [ThOO] i [ReOl].

- Netscape je razvio SSL, a IETF je razvio TLS, koji je zasnovan na SSL 3.0.
- Iako TLS i SSL u opštem slučaju ne mogu da komuniciraju, TLS ima mogućnost da se vrati na SSL 3.0.
- Neke razlike niskog nivoa odreduju način na koji se defmišu proširenja bajta kada je neophodno kreirati blokove odgovarajuće dužine između SSL-a i TLS-a.
- TLS podržava dodatne kodove upozorenja (to su kodovi koji korisnika upozoravaju kada nešto nije u redu).
- IETF je izbacio Fortezza* algoritme koje SSL podržava.

X.509 sertifikat

Počinjemo od toga kako se server autentifikuje. Pretpostavimo da ste se povezali na neki Web sajt i da treba da unesete neke privatne informacije, kao što je broj kreditne kartice. Kako možete da budete sigurni da je sajt to što tvrdi da jeste? Za mnoge ljude je i dalje sigurnije da odu u prodavnicu i da na kasi daju svoju kreditnu karticu, jer u tom slučaju znaju ko pristupa njihovom računu. Ipak, pretpostavimo da na Vaša vrata zakuca trgovački putnik i da odlučite da kupite nešto od onoga što on nudi. Da li ćete imati dovoljno poverenja da mu date broj svoje kreditne kartice? On možda tvrdi da predstavlja veliku uglednu kompaniju, ali kako da budete sigurni? Može Vam pokazati čak i svoju identifikacionu karticu iz kompanije, ali kako da budete sigurni da nije falsifikovana? Ovde postoji ozbiljan problem poverenja.

Povezivanje na Internet je po mnogo čemu slično. Kako da znate da li možete da vemjete sajtu? Odgovor obezbeduje **X.509 sertiflkat.** Reč je, u suštini, o dokumentu koji sajt obezbeduje kao dokaz svog identiteta. To je nešto slično elektronskom ID-u. Naravno, mora da postoji i nešto više od toga, jer svako može da kreira ID validnog izgleda.

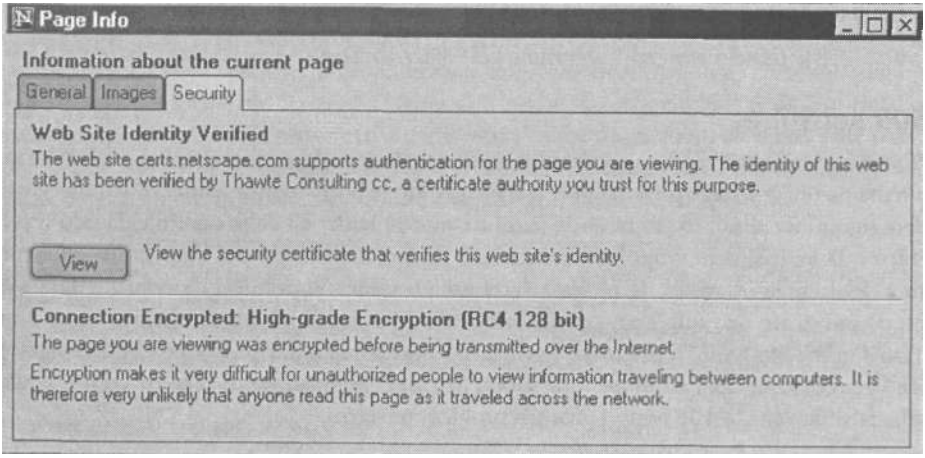
Sledeći deo "slagalice" je telo nadležno za izdavanje sertifikata **(CA - certificate authority),** entitet koji je ovlašćen za izdavanje sertifikata. Razmotrimo analogni primer. Kompanija intervjuiše kandidate za određeno mesto. LJ listi kandidata su navedene kvalifikacije i stečene diplome, ali kompanija ne donosi konačnu odluku o zapošljavanju (bar ne pozitivnu) samo na osnovu tih informacija.

* Fortezza algorilme, zasnovane na Capstone, raz\'ila je NSA; detalji specifikacije su poverljivi.

Kandidat mora da obezbedi preporuke i odgovarajuće dokaze o stečenoj diplomi sa fakulteta. Nakon toga, kompanija proverava preporuke i univerzitet da bi se utvrdilo da li je sve što je kandidat naveo tačno. Naravno, mora da postoji poverenje i u ono što osobe navedene u preporukama, ili sa univerziteta kažu. Poslodavac pretpostavlja da su njihove informacije pouzdane. Vremenom poslodavac nauči da prepozna kome može da veruje.

Tako je i sa CA. Kao potrošač, čuvate listu onih kojima možete da verujete. Ako CA izda sertifikat za Web sajt, možete da verujete da je Web sajt ispitan i da je identifikovan kao legitiman. Drugim rečima, CA predstavlja nekoga ko daje verodostojne preporuke za taj Web sajt.

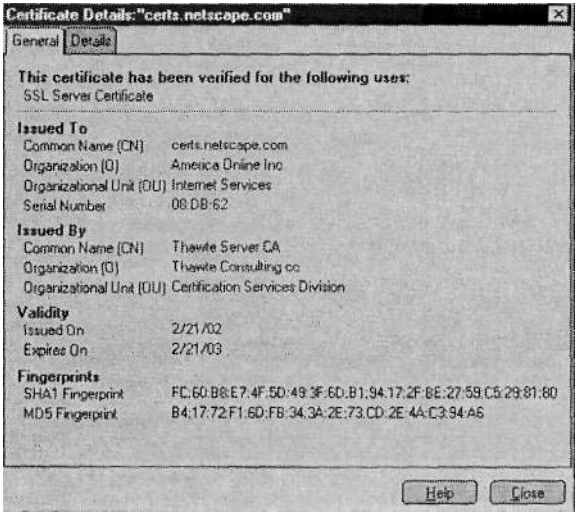
Vaša sledeća pomisao može da bude: "Naručio sam nešto preko Interneta, ali ne sećam se da sam proveravao listu CA, niti sam pogledao da li postoji sertifikat". To je zato što to umesto Vas obavlja pretraživač. Na primer, pretpostavimo da koristite Netscape i da ste se povezali na https://certs.netscape.com.* Primećujete da se koristi *https* umesto uobičajenog prefiksa *http* u URL-u. Dodatno slovo *s* ukazuje da ste se povezali na siguran sajt koji ima sertifikat i koji je odobrio CA. Da biste videli ko je izdao sertifikat, selektujte opciju Page Info iz menija View. Pređite na karticu Security; na slici 7.20 prikazan je jedan mogući primer. Tu je kao CA naveden Thawte Consulting. Ako želite da vidite konkretan sertifikat koji CA izdaje, kliknite View i videćete nešto slično onome na slici 7.21.'



SLIKA 7.20 *Informacije tela nadležnog za izdavanje sertifikata*

* Sve legitimne adrese koje počinju sa *htlps* mogu da posluže.

t Osim toga, možete da vidite da li su ove informacije dobijene korišćenjem Internet Explorera (IE). Kada se povežete na siguran sajt, IE na dnu statusne linije prikazuje ikonicu sa katancom. Ako duplo kliknete tu ikonicu i selektujete različite kartice i opcije, mod ćete da vidite informacije iz sertifikata.



SLIKA 7.21 *Digitalni sertifikat*

Thawte je jedno od brojnih proverenih CA tela koje Vaš pretraživač priznaje. Da biste videli listu i drugih CA u Internet Exploreru (verzija 6), selektujte Internet Options iz menija Tools. Zatim, predite na karticu Content, kliknite Certificates i pređite na neku od mogućih kartica u rezultujućem prozoru. Ako koristite Netscape 7.0, selektujte Preferences iz menija Edit. Otvorite kategoriju Privacy and Security i selektujte opciju Certificates. Kliknite Manage Certificates i pređite na karticu Authorities. Verisign & Thawte su dva najveća CA, ali videćete i mnoge druge.

Sertifikat sa slike 7.21 prikazuje CA telo koje je izdalo sertifikat, period u kome je sertifikat validan (sertifikati imaju vreme isteka) i dva otiska (fingerprints). To su vrednosti digitalnog sažetka generisane SHA-I, ili MD5 algoritmom, a koriste se za autentifikaciju. Pošto svako može da kreira sertifikat, moramo da utvrdimo koji su sertifikati autentični.

Kada se korisnik poveže na Web sajt, pretraživač radi sledeće (u stvari, radi još mnogo čega - uskoro ćemo dati detaljniji opis):

- Preuzima sertifikat servera.
- Proverava datum izdavanja i isteka sertifikata. Da li se tekući datum nalazi u tom periodu?
- Proverava otiske. Da li autentifikuju dokument?
- Proverava CA. Da li je na listi proverenih tela?

Sva ova pitanja moraju da imaju potvrdne odgovore, ili se, u suprotnom, korisnik obaveštava da postoji neki problem. Na primer, ako pretraživač ne može da pronade CA u listi, prikazače poruku sličnu onoj na slici 7.22. Tada se korisniku prepušta odluka o narednoj akciji. Ako korisnik odluči da, ipak, nastavi dalje, sve dalje akcije izvodi na sopstveni rizik.

SLIKA 7.22 *Bezbednosno upozorenje*

Usaglašavanje

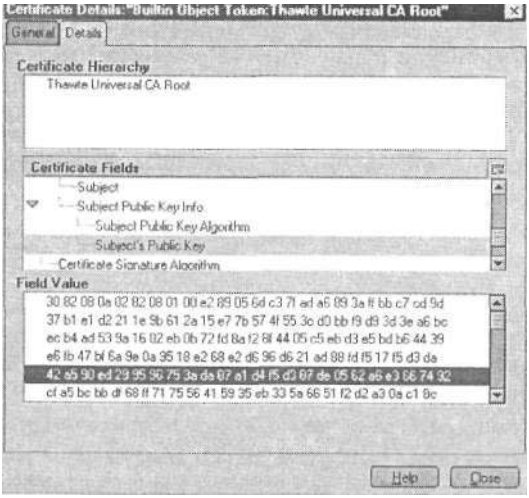
Sledeći korak je opisivanje onoga šta TLS (ili SSL) radi kada pokušava da se poveže na siguran sajt (označen prefiksom *https*). Inicijalni set razmena između klijenta i servera naziva se *usaglašavanje* (handshake), a definiše rutine za dešifrovanje i uključuje razmenu ključeva i procedure za proveru autentičnosti. Usaglašavanje je definisano sledećim koracima. Kompletan opis možete da pronađete u referencama [Th00] i [Re01].

1. Klijent šalje informacije do servera. Informacije uključuju najnoviju raspoloživu verziju SSL-a (ili TLS-a), liste algoritama za šifrovanje i za razmenu ključa i listu metoda za šifrovanje koje klijent podržava. Algoritmi za šifrovanje su DES, trostruki DES, RC2, RC4, IDEA i Fortezza. Algoritmi za razmenu ključa su RSA, Diffie-Hellman i Fortezza. Ove informacije ne definišu šta će se uraditi, nego samo šta može da se uradi. Klijent takođe šalje ID sesije i neke nasumično generisane podatke (čiju ćemo namenu uskoro opisati). ID sesije može da bude nula, ili različit od nule. Vrednost 0 znači da klijent zahteva novu sigurnu sesiju, a nenulta vrednost da klijent zahteva ažuriranje parametara tekuće sesije. To je dodatna mera zaštite u slučaju da neautorizovana osoba nadgleda prenos. Periodična promena rutina otežava praćenje.

2. Server šalje broj verzije, specifikaciju šifrovanja i razmene ključa i algoritam kompresije koji je izabran na osnovu klijentovog predloga. Selektuju se najnovije (i najsigurnije) verzije koje obe strane mogu da podrže. Server takođe šalje neke nasumično generisane podatke i svoj sertifikat.

3. Do sada se nije desilo ništa posebno, jer su klijent i server, jednostavno, razmenili informacije. To se menja u koraku 3. Pošto je klijent primio serverov sertifikat, mora da proveri validnost sertifikata i autentičnost servera. Naravno, u okviru toga mora da bude potvrdena i autentičnost sertifikata. Klijent izvršava sledeće korake. Problem u bilo kom koraku izaziva upozorenje koje se prosleduje do korisnika. Svi koraci moraju da budu kompletirani da bi se nastavilo dalje.

   - Današnji datum se poredi sa datumom izdavanja i isteka sertifikata. Ako se tekuć datum ne nalazi u tom periodu, sertifikat nije validan.
   - Proverava se da li se CA telo koje je izdalo sertifikat nalazi u listi proverenih CA.
   - Pošto sertifikat može da bude falsifikovan, klijent mora da proveri njegovu autentičnost. CA koji izdaje sertifikat prilaže i digitalni potpis za njega. U te svrhe se koriste prethodno opisani metodi za utvrđivanje vrednosti digitalnog sažetka. Nakon toga se vrši šifrovanje te vrednosti pomoću privatnog ključa. Koriste se dva algoritma: SHA-I i MD5. Ako se otkrije sigurnosni propust kod jednog, drugi obezbeduje dodatne mere zaštite.
   - Pristupa se javnom ključu izabranog CA tela i on se primenjuje na digitalni potpis da bi se dobila originalna vrednost digitalnog sažetka. Izvršavaju se procedure za proveru autentičnosti slične onima koje smo prethodno opisali i utvrduje se da li je dobijena tačna vrednost digitalnog sažetka. Možda se pitate odakle klijentu ključ od CA. Setite se da su ovi ključevi javno poznati i da se čuvaju zajedno sa listom CA. Na primer, ako ste koristili prethodno opisane procedure za dobijanje liste CA, možete da selektujete bilo koji od njih, pa kliknite View. Prelaskom na karticu Details i selektovanjem polja Public Key Info možete da dobijete prikaz sličan onome na slici 7.23 - prikazan je deo javnog ključa koji je korišćen za RSA (ostatak javnog ključa može da se vidi skrolovanjem vrednosti u polju Field Value).



SLIKA 7.23 *Prikazivanje javnog ključa CA*

- Naziv domena u sertifikatu se poredi sa nazivom domena na serveru. Ovaj korak omogućava detektovanje napada tipa man-in-the-middle, kod koga se "uljez" ubacuje između klijenta i servera. Da je umetanje izvedeno u ranoj fazi procedure usaglašavanja, "uljez" bi mogao da uspostavi sopstveni sigurni kanal sa klijentom i serverom i tako bi mogao da vidi sve informacije koje obe strane šalju. Klijent bi mislio da je "uljez" server, a server bi mislio da je "uljez" klijent.

4. Klijent kreira 48-bajtnu sekvencu (označenu kao *pre-master secret),* šifruje je, koristeći serverov javni ključ,* i šalje je do servera. Klijent će koristiti ovu sekvencu za generisanje simetričnog ključa za šifrovanje za bezbednu sesiju. Server prima poslatu sekvencu, dešifruje je pomoću svog privatnog ključa i izvodi slična izračunavanja za generisanje ključa.

5. Ako je neophodno, server može da zatraži autentifikaciju klijenta. Proces je sličan autentifikaciji servera i ovde nećemo navoditi njegove detalje. Jednostavno, napominjemo da neke komunikacije (na primer, transfer novca između dve fmansijske institucije) zahtevaju autentifikaciju obe strane. Većina korisnika koji su naručivali nešto preko Interneta nije morala da se autentifikuje na ovakav način. Kada korisnik potvrdi unos broja svoje kreditne kartice, server proverava broj, koristeći iste metode kao bilo koji drugi prodavac, bilo na Internetu, ili u tradicionalnim prodavnicama.

6. I klijent i server koriste pre-master secret sekvencu za generisanje *master secret* sekvence. Da bi se izračunala ta sekvenca, klijent šalje nasumično generisane podatke, ti se podaci primaju na serveru (sećate se koraka 1 i 2) i izvršava se hash procedura sa pre-master secret sekvencom, koja daje 48-bajtnu sekvencu. Server izvršava analogni postupak. I klijent i server uvode master secret sekvencu u hash algoritme za eventualno generisanje ključeva sesije koji se koriste za šifrovanje podataka koje će kasnije razmenjivati u toku sesije.

7. Klijent šalje serveru drugu poruku, u kojoj potvrđuje kreiranje ključa sesije i saopštava da će sve buduće poruke biti šifrovane pomoću ključa i algoritma naznačenih u prethodnoj razmeni. Server šalje analogne informacije do klijenta. Kada i klijent i server prime ove poslednje poruke, uspostavlja se sigurna sesija i sigurna komunikacija koja uključuje metode za šifrovanje i naznačeni ključ može da otpočne.

Kao i u slučaju prethodnih tema, moguće je navesti mnogo više detalja; upućujemo zainteresovane čitaoce na reference [ThOO] i [ReOl], a mogu da posete i Web sajtove www.ietf.org./html.charters/tls-charter.html i http://developer.netscape.com/docs/manuals/security.sslin/contents.htm.

* Korišćenje javnog ključa predstavlja samo jedan od mogućih načina za slanje pre-master sekvence. U zavisnosti od specifikacija za razmenu ključa iz prethodnih koraka, mogla je da se koristi i neka varijacija Diffie-Hellman, ili Fortezza algoritma.

7.6 Firevvalli

Do sada predstavljene mere zaštite dizajnirane su za zaštitu informacija koje se koriste za obavljanje poslova preko Interneta. Medutim, ljudi i mašine su izloženi riziku i kada ne pokušavaju da obave bilo kakve aktivnosti. Kompjuteri koje koristite i mašine koje obezbeduju mrežne servise izloženi su riziku od napada malicioznih ljudi koji žele da ometu njihove aktivnosti, ili da im nanesu štetu. To je realnost današnje svakodnevnice; važi i za cyberspace. Kako sprečiti napade na kompjuter? Hi, možda realnije, kako sprečiti uspešne napade?

Prvi korak u borbi sa napadima je upoznavanje mrežne arhitekture i načina na koji se napadi prenose kroz mrežu. Skoro svaka organizacija održava svoju mrežu i omogućava pristup Internetu preko sopstvenih mašina. Osim toga, svaka mašina na internoj mreži ima svoju Internet Protocol (IP) adresu, tako da je vidljiva i za druge kompjutere van lokalne mreže. Cim ste vidljivi, automatski postajete meta potencijalnih napada.

Kako zaštititi kompjuter koji ima IP adresu? Kao analogni primer možete da uzmete javne sisteme - na primer, aerodrome, ili medijske dogadaje viskog nivoa. Postavljaju se kontrolne tačke i Ijudi koji žele pristup moraju da produ kroz njih, gde se skeniraju i traže "stvari" koje mogu da predstavljaju pretnju. Primoravajući svakoga da prode kroz jednu, ili više kontrolnih tačaka, službenici mogu bolje da kontrolišu bezbednost objekta, ili dogadaja. Ako je broj kontrolnih tačaka mali, zaštita se bolje defmiše, jer službenici mogu da fokusiraju svoje napore na manji broj lokacija.

Mnoge mreže koriste sličan koncept. Iako svaki kompjuter u lokalnoj mreži ima pristup Internetu, sav Internet saobraćaj mora da prode kroz jednu, ili više specijalizovanih mašina koje se nazivaju firewalli. Svrha firewalla je da prouči saobraćaj koji prolazi preko njega i da detektuje eventualne pretnje. Prolazak se zabranjuje za sve što može da predstavlja potencijalnu pretnju. Ostatak normalno prolazi. Zvuči jednostavno, ali teži deo podrazumeva donošenje odluke šta predstavlja pretnju, posebno zbog toga što su mnoge pretnje maskirane tako da izgledaju kao legitimni saobraćaj. Kasnije ćemo analizirati neke primere.

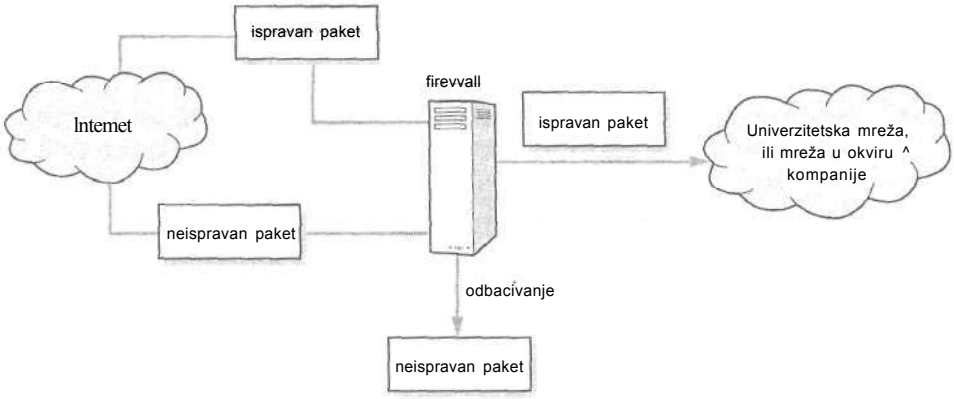
Iako su u Poglavlju 11 detaljnije predstavljeni Internet Protokol i njegov prateći protokol Transport Control Protocol (TCP), postoji nekoliko detalja koje moramo istaći pre nego što pređemo na predstavljanje načina funkcionisanja firewalla.

- IP i TCP odgovaraju slojevima 3 i 4, respektivno.
- Internet saobraćaj je organizovan u veći broj paketa koji putuju izmedu rutera. IP paketi sadrže podatke i adresu koja definiše odakle paket potiče (izvor) i gde treba da se isporuči (odredište). U paketu se nalaze i polja koja defmišu da li TCP, ili neki drugi protokol sloja 4, kao što je UDP, koristi IP.
- Podaci aplikacije se obično dele na jedan, ili više Internet paketa.
- Jedan server na mreži može da obezbedi različite servise. Klijent koji želi odredeni servis mora da naznači IP adresu servera i **broj porta** kojim se identifikuje traženi servis. Brojevi portova za neke uobičajene servise su dobro poznati, kao što su port 23 za Telnet (za daljinsko logovanje), port 21 za FTP (za transfer fajlova), port 80 za HTTP (za aktivnosti na Webu) i port 25 za SMTP (email).

Filtriranje paketa

Filtriranje paketa je verovatno najjednostavniji pristup za dizajniranje firewalla. Funkcioniše na osnovu jednostavne pretpostavke (slika 7.24): proučava sa sadržaj zaglavlja svakog paketa i donosi se odluka da li paket treba, ili ne treba da prođe. Odluka se obično donosi na osnovu sadržaja adresnog polja paketa, broja porta, ili transportnog protokola koji je naveden. Evo nekih raogućih primera odluka:

- Dopušteni su svi paketi koji imaju 23 kao oznaku TCP porta. Na taj način je omogućeno logovanje na bilo koju mašinu u lokalnoj mreži preko Telneta (pomoćna rutina za daljinsko logovanje).
- Dopušteni su svi paketi koji imaju 23 kao oznaku TCP porta i čija se adresa nalazi u listi adresa koju firewall održava. Na taj način je omogućeno logovanje na bilo koju naznačenu mašinu iz liste preko Telneta. Sto je još značajnije, firewall blokira sve pokušaje logovanja na neke druge lokalne mašine. Administrator mreže defmiše koje će se mašine naći na listi.
- Dopušteni su svi dolazeći paketi sa odredišnom adresom naznaćenom u Hsti koju fuevrall održava. Tako je omogućeno propuštanje zahteva za servisima koje obezbeđuju serveri iz liste.
- Dopušteni su svi odlazeći paketi koji sadrže odredišnu adresu naznačenu u listi koju firewall održava. Ovo se razlikuje od prethodnog primera po tome što se odnosi na odlazeće pakete. Kompanija mora da definiše stroga pravila koja dopuštaju zaposlenima da pristupaju samo određenim destinacijama radi obavljanja svojih radnih zadataka. Ovakve polise mogu da zabrane zaposlenima neobavezno surfovanje Webom, ili pristup sajtovima koji nisu povezani sa njihovim poslom.
- Dopušteni su svi odlazeći paketi sa odredišnom adresom koja odgovara nekoj od IP adresa u okviru organizacije. Ovo možda deluje ćudno na prvi pogled. Na kraju krajeva, odlazeći paket potiče sa nekog kompjutera u okviru organizacije i, samim tim, izvor odgovara nekoj IP adresi te organizacije. Nažalost, to nije tačno.



SLIKA 7.24 *Filtriranje paketa*

Neki napadi na sajtove su zasnovani na lažnim IP adresama (IP address spoofing), kada IP pake sadrži lažnu IP adresu u polju izvora i pokušava da sakrije pravi izvor poruke. Kasnije ćemc predstaviti neke vrste napada, ali ideja je da se blokiraju svi odlazeći paketi koji sadrže lažni izvorne adrese.

Napomenimo da ovi primeri definišu kriterijume koji će biti korišćeni za donošenje odluke c propuštanju paketa. Ovo povlači podrazumevanu akciju koja ukazuje da paket treba blokirati ako nije đrugačije naznačeno. Postoji i druga opcija: inicijalno se dopušta propuštanje svih pake ta, ako nije drugačije naznačeno. U ovom slučaju moguće je doneti neke od sledećih odluka:

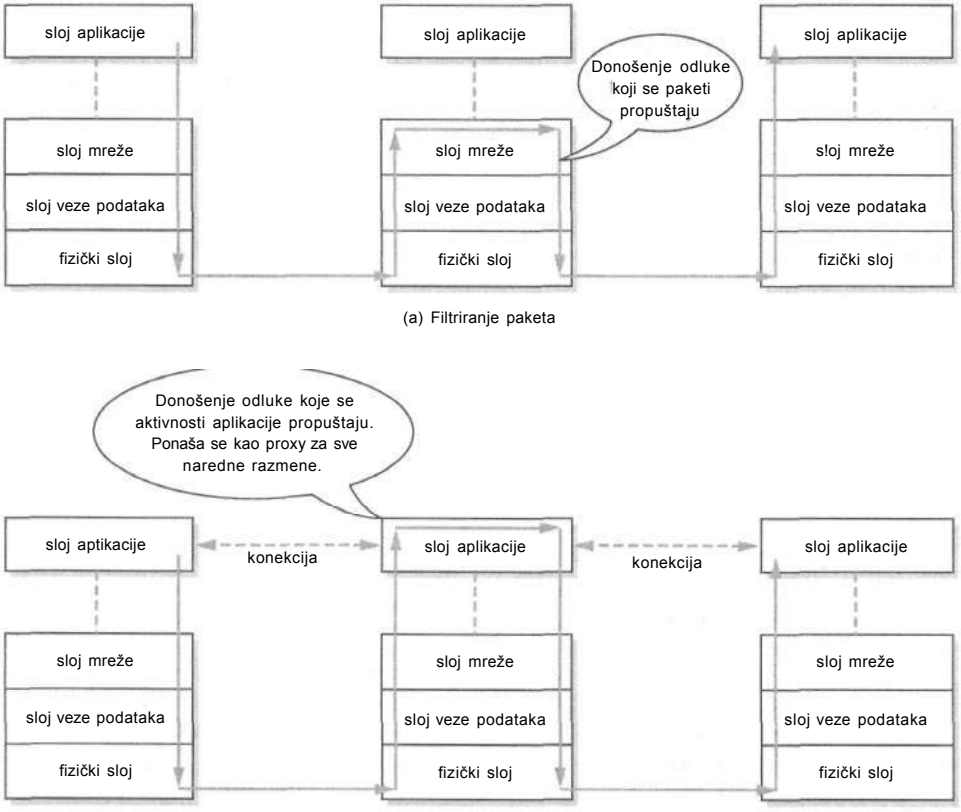
- Blokiraju se svi dolazeđ paketi sa 23 kao oznakom TCP porta. Na ovaj način su zabranjeni svi pokušaji logovanja na mašine kompanije preko Telneta.
- Blokiraju se svi dolazeći paketi koji sadrže izvomu adresu naznačenu u listi koju firewall održava. Ako administrator mreže utvrdi da su određeni udaljeni sajtovi problematični, ili da neko odatle šalje ogromne količine beskorisnih email poruka, firevvall može da bJokira sve što dolazi sa tog sajta. Covek se stvarno dobro oseča kada stavi takve *sajtove na listu za blokiranje!*
- Blokiraju se svi dolazeći paketi koji sadrže odredišnu adresu naznačenu u listi koju firewall održava. Na ovaj način, administrator mreže može da obezbedi servere sa dodatnim nivoom zaštite, a da se, i pored toga, dopusti pristup nekim drugim serverima. Time se eflkasno blokira pristup svim serverima iz liste za sve one koji se nalaze na drugoj strani firewalla (tj. opštoj Internet zajednici).
- Blokiraju se svi odlazeći paketi koji sadrže izvornu i odredišnu adresu iz listi koje firewall održava. Možda je šef laboratorije otkrio da mnogi koriste laboratorijske kompjutere za povezivanje na server koji omogucava igranje igara preko Interneta. Ako takve aktivnosti nisu u skladu sa pravilima organizacije, mogu se sprečiti zabranom propuštanja odlazećih paketa iz laboratorije do odgovarajućih servera.

Koji je pristup bolji? Ako se inicijalno dopušta propuštanje paketa, administrator mreže mora neprestano da proverava odakle mogu da poteknu novi napadi, ili u kom obliku mogu da se jave. Osim toga, mora se voditi računa o namšavanju lokalnih polisa. Sa svakom novom mogućnošću, firewall mora da se ažurira. Naravno, administrator ne mora da bude svestan svih mogućnosti za napade i zato je ovaj pristup manje bezbedan.

Ako se inicijalno blokiraju svi paketi, administrator mora da utvrdi kako da konfiguriše legitimne zahteve. Ništa neće proći ako nije eksplicitno definisano na firewallu. Ovaj pristup je bezbedniji, ali administrator mora da poznaje sve legitimne potrebe korisnika. Ako nešto izostavi, onda firewall može da blokira legitimne pakete i tako inicira neprestane žalbe korisnika. Naravno, firewall može da se ažurira, ali korisnik u tom slučaju može da kaže: "Ne daju mi ništa da uradim dok se najpre ne požalim".

Firevvalli tipa Application-Level Gateway

Sledeći tip firewalla funkcioniše na višem nivou. Filtriranje paketa prvenstveno funkcioniše na sloju 3, gde se odluke donose na osnovu sadržaja paketa (slika 7.25), i ne obezbeduje fleksibilnost kada se vrši diskriminacija zasnovana na akcijama koje se izvode u okviru odredene aplikacije.



SLIKA 7.25 *Mogući dizajni fireualla*

Na primer, možete da dopustite korisniku da se poveže na mašinu koristeći protokol za transfer fajlova radi preuzimanja fajlova. Međutim, želite da zabranite postavljanje fajlova. Ovo je tipično za mnoge sajtove koji služe kao repozitorijumi fajlova za korisnike Interneta.

Filtriranje paketa ne može da zadovolji ovakve zahteve. Eventualno, paketi namenjeni portu 21 (FTP) mogli bi da se blokiraju, ali to bi blokiralo sve FTP zahteve, a ne samo postavljanje fajlova na server. Kao alternativa, može da posluži ugradivanje dodatne logike u firewall koji može da razume aplikacije i njihove mogućnosti (slika 7.25b). Ovakav tip firewalla se označava kao **application-level gateway** (poznat je i pod nazivom *proxy server).* Ovaj firewall pokreće specijalni program dizajniran za svaki tip aplikacije čije je zahteve neophodno pratiti. Proučavaju se zahtevi specifični za sloj aplikacije, a zatim se dopuštaju, ili odbijaju, u zavisnosti od informacija koje obezbeduje firewall. Inicijalno ne obezbeduje zaštitu za sve aplikacije; gateway program mora da se instalira za svaku individualnu aplikaciju.

Na primer, pretpostavimo da klijent želi da uspostavi FTP konekciju sa serverom na kome je postavljen firevvall. Zasebni program se izvršava na tom firewallu i presreće sve zahteve od klijenta. Ako klijent šalje get zahtev (zahtev za uzimanje informacija), zahtev se postavlja u IP paket i rutira se ka firewallu, gde se izvlači iz paketa i proučava na sloju aplikacije firewalla. Ako firewall odobri takav zahtev, onda se zahtev prosleđuje do nižih slojeva, gde se postavlja u drugi IP paket i šalje do odredišta. Ako je umesto get zahteva korišćen zahtev put (zahtev za postavljanje informacija), firevvall blokira transfer ako takvi zahtevi nisu dopušteni. Naravno, sve ove informacije moraju da budu definisane u okviru pravila firewalla, koje odreduju polise organizacije.

Glavna prednost firewalla tipa application-level gateway u odnosu na fdtriranje paketa je povećana fleksibilnost donošenja odluka u skladu sa aplikacijom. Medutim, postoje i dodatni troškovi. Firewall mora da pokreće odgovarajuće aplikacije. Osim toga, firewall rastavlja konekciju izmedu klijenta i servera na dve zasebne konekcije: jednu izmedu klijenta i firewalla, a drugu izmedu firewalla i servera. Firewall igra ulogu servera kada "razgovara" sa klijentom, a u komunikaciji sa serverom ponaša se kao klijent. Ovo zahteva dosta dodatnih aktivnosti.

Ispitivanje sadržaja paketa na osnovu prethodnog stanja

Pristup koji mnogi sve više koriste uključuje ispitivanje sadržaja paketa na osnovu prethodnog stanja (stateful inspection). Slično kao pristup filtriranja paketa, funkcioniše na sloju mreže i uvodi dodatne troškove za proxy. Filter paketa donosi odluku isključivo na osnovu sadržaja paketa. Nažalost, lukaviji hakeri mogu da falsifikuju (lažiraju) sadržaj paketa i tako zaobidu bezbednosne mere. Pristup ispitivanja sadržaja na osnovu prethodnog stanja vrši ispitivanje sadržaja u kontekstu onoga što se desilo ranije.

Možda se kao najjednostavniji primer može uzeti komanda ping. Na primer, pretpostavimo da ste izdali komandu ping w.x.y.z iz komandne linije. Ovde w.x.y .z predstavlja neku IP adresu. Mogući odziv na komandu može da izgleda ovako:

64 bytes from w.x.y.z: icmp_seq_0 ttl_109 time_106.0 ms

64 bytes from w.x.y.z: icmp_seq_l UL109 time__94.3 ms

64 bytes from w.x.y.z: icmp_seq_2 ttl_109 time_109.7 ms

64 bytes from w.x.y.z: icmp_seq_3 ttl_109 time„109.6 ms

64 bytes from w.x.y.z: icmp_seq„4 ttl_109 time_109.5 ms

U osnovi, ovaj odziv ukazuje da uredaj sa specifičnom IP adresom može da se kontaktira; taj uredaj pokazuje koliko je vremena bilo potrebno da paket stigne do njega i vrati se nazad. Ping je uobičajena alatka koju administratori koriste za proveravanje da li je server pokrenut i koriste je neki Internet sajtovi za prikupljanje statističkih podataka o dostupnosti. Ovde je značajno napomenuti da komanda ping šalje Echo Request paket do naznačenog sajta, a sajt reaguje slanjem niza Echo Response paketa.

Naravno, svi ovi paketi moraju da prođu kroz firewall. Kod ovog pristupa Echo Response paket nije dopušten ako prethodno nije postojao Echo Request paket. Osim toga, odredišna i izvorna adresa u odzivu moraju da odgovaraju izvoru i odredištu u zahtevu, respektivno. Ideja je da nema smisla slati Echo Response paket ako prethodno nije zahtevan. Pošto su takvi paketi korišćeni u prošlosti prilikom napada odbijanja servisa (videti odeljak 7.8), ovaj pristup omogućava dodatne mere zaštite.

U opštem slučaju, na firewallu postoji baza pravila koja sadrži informacije slične onima koje su predstavljene u tabeli 7.2. Kolone Izvor i Odredište predstavljaju sajtove, a zvezdica (*) ukazuje na džoker znak. Kolona Servis definiše protokole koji su uključeni u pravilo, a kolona Akcija definiše da li se paket koji je pridružen uz ove tri kolone prihvata, ili odbacuje. U opštem slučaju, pravila mogu da imaju i druge parametre. Na primer, polje Time of Day može da se uključi u bazu pravila, tako da se naznači da su neki paketi dopušteni samo u odredeno vreme. Osim toga, administrator može da instalira *ograničenje brzine,* tako da se dopusti pristup servisima, ali sa redukovanim bitskim brzinama. Ovo može da se koristi u kombinaciji sa određenim korisničkim grupama koje ne zahtevaju prenos velikih fajlova. Sporije bitske btzine neće imati mnogo efekta za fajlove koji su im neophodni za obavljanje posla, ali će obeshrabriti korisnike da preuzimaju svoje omiljene pesme, ili filmove. Naravno, kod pravih firevvalla obično postoji mnogo veći broj zapisa nego što je predstavljeno u ovoj tabeli.

Tabela 7.2: Primer baze pravila za firewall

Pravilo br.	Izvor	Odredište	Servis	Akcija	Komentar
1	*Any (Bilo koji)	ServerA.nekaOrg.com ServerB.nekaOrg.com ServerC.nekaOrg.com	HTTP, Telnet, FTP	Accept (Prihvatanje)	Dopušta HTTP, Telnet i FTP pristup bilo kom od servera A, B, ili C sa bilo kog spoljašnjeg izvora.
2	GroupA, GroupB	*Any	Echo reply, Echo request	Accept	GroupA i GroupB su grupe specifične za sajt (možda laboratorijske, ili nastavne). Dopuštaju pakete Echo Request i Echo Reply sa tih sajtova.
3	*Any	PrintServerA	*Any	Drop (Odbacivanje)	Ne dopušta pristup na PrintServerA iz bilo kog spoljašnjeg izvora.
4	*Any	GroupA	Kazaa Drop		Ne dopušta pristup na Kazaa sa bilo koje mašine iz GroupA. Tako se zabranjuje postavljanje fajlova na Internet zajednicu sa bilo kog kompjutera iz GroupA
N	*Any	*Any	*Any		Ako se ne primenjuje ni jedno drugo pravilo, ovo je podrazumevano.

Drop

U odsustvu bilo kojih drugih informadja, firewall donosi odluku da li će prihvatiti, ili odbaciti paket, u zavisnosti od baze pravila. Medutim, kada paketi počnu da se prihvataju, definisan je kontekst za prihvatanje, ili odbacivanje novih paketa. Razmotrimo sledeći primer:

1. Udaljeni klijent zahteva FTP konekciju sa serverom A.
2. Server A potvrduje zahtev i uspostavlja se FTP kontrolna sesija. Klijent sada može da zahteva postavljanje, ili preuzimanje fajlova.
3. Firevvall kreira stanje defmisano IP adresom klijenta kao izvorom zahteva, IP adresom servera kao odredištem i portom broj 21 (port za FIP konekcije). Stanje se održava kao ulaz u internoj tabeli stanja.
4. Naredni paketi koji pristižu na firewall proučavaju se u kontekstu prethodnih aktivnosti. Na primer, ako server pokušava da inicira preuzimanje fajla za klijenta, paket ne može automatski da prode. Razlog je činjenica da konkretni kontekst (stanje konekcije) ukazuje da je klijent zahtevao uspostavljanje konekcije i jedino klijent može da inicira transfer podataka.
5. Pretpostavimo da klijent želi da provuče drugi paket preko konekcije koja pristupa portu sa drugim brojem. Taj klijent možda pokušava da napadne sistem, ili pokušava da pošalje paket sa potencijalno štetnim sadržajem. Izvor, odredište i broj porta u tom paketu ne odgovaraju ni jednom postojećem stanju u tabeli. Zato firewall ne prihvata automatski konkretni paket. Umesto toga, podvrgnuće ga originalnim kriterijumima iz baze pravila.

Glavna ideja za ispitivanje paketa na osnovu prethodnog sadržaja je da se korisnik spreči da postavi neki legitimni zahtev i da dobijeni odziv iskoristi za pokretanje napada. Ono što dobro funkcioniše na jednom sajtu možda neće odgovarati potrebama nekog drugog. Naravno, zato nije moguće reći koji je pristup najbolji. Sve što može da se uradi je da se neprestano prati razvoj novih tehnologija. U referencama [Po02] i [St03] možete da pronadete detaljniji opis firewal-Ia, dok je u referenci [CoOIa] dat pregled raznih firewalla.

7.7 Virusi

Do sada smo razmatrali integritet i zaštitu podataka u toku njihovog prenosa preko nekog medijuma. Dali smo odgovore na pitanja u vezi detektovanja oštećenih podataka i maskiranja podataka da ih neautorizovane osobe ne mogu razumeti. Sledeća ozbiljna pretnja po bezbednost i integritet informacija jesu kompjuterski virusi i "crvi".

Strogo govoreći, virusi i "crvi" mogu biti manji problem za kompjutersku mrežu, a veći za operativni sistem. Iako mreže zasigurno olakšavaju širenje nekih virusa, ako je kompjuter povezan na mrežu, to ne znači da će virusi jednostavno "uskočiti" u kompjuter i "pojesti" hard diskove. U stvari, dva glavna razloga zbog kojih virusi i "crvi" postoje jesu propusti u zaštiti operativnih sistema i nemamo ponašanje korisnika.

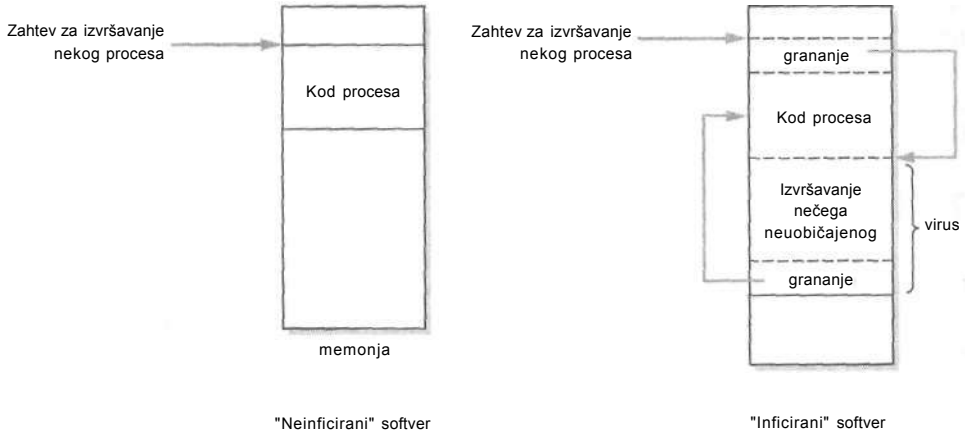
Sa druge strane, mrežne konekcije nisu bezopasne, što mogu da posvedoče žrtve Internet "crva". Pristup elektronskim oglasnim biltenima i međusobna povezanost velikog broja kompjutera širom sveta čine problem virusa i "crva" krajnje ozbiljnim. Zato se u poglavlju o zaštiti moraju predstaviti i, u krajnjem slučaju, makar pomenuti njihove mogućnosti.

'Inficiranje" fajlova

Virus je kolekcija instrukcija pridružena izvršnom fajlu koja radi nešto za šta originalni izvršni fajl nije dizajniran. Na PC-jima on se obično pripaja fajlovima sa ekstenzijom .exe, Oi .com. Na Macintoshu krak resursa (resource fork) fajla je obično meta virusa. Kada se virus pripoji uz fajl, kažemo da je fajl "inficiran".

Crvi (worms) su po mnogo čemu slični virusima, ali obično se javljaju kao zasebni programi. Kao i virusi, upadaju na sistem i potencijalno narušavaju bezbednost sistema.

Na slici 7.26 prikazan je jedan od načina za razlikovanje "inficiranog" i "neinficiranog" fajla. Ako ste zainteresovani za druge načine, pogledajte reference [Sp90] i [Ka94]. "Neinficirani" fajl sadrži izvršni kod koji se može izvršiti kada se pozove. Međutim, kod "inficiranog" fajla virus umeće naredbu grananja na sopstveni kod. Kada korisnik pokrene "inficirani" fajl radi izvršavanja nekog zadatka, komanda grananja prenosi kontrolu najpre na kod virusa.



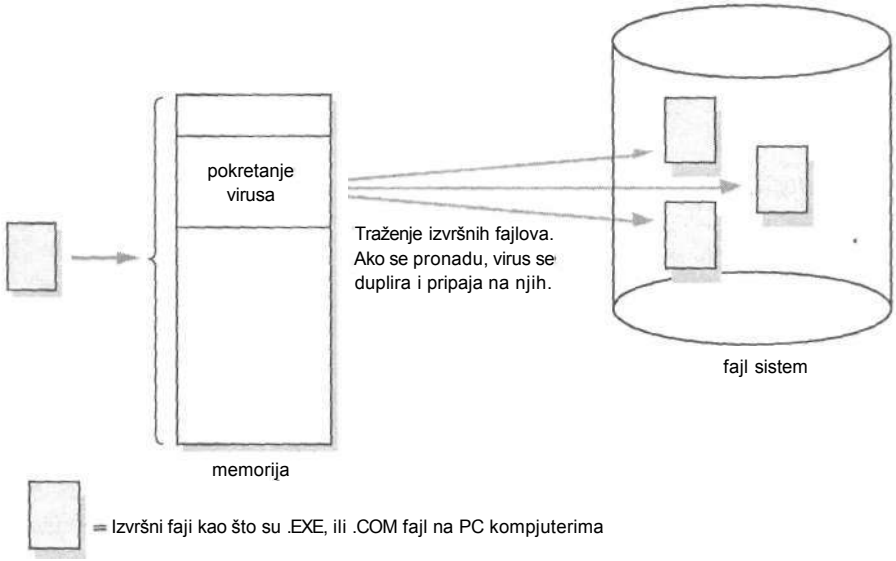
SLIKA 7.26 *Virus na izvršnom fajlu*

Virus izvršava svoj "posao", pa pomoću sledeće naredbe grananja kontrolu vraća na izvršavanje traženog zadatka. Sa stanovišta korisnika, zahtevani zadatak je izvršen. Osim ako virus nije uradio nešto krajnje očigledno, kao što je brisanje hard diska, korisnik možda neće ni znati da on postoji.

Šta sve virus može da uradi? Nažalost, skoro sve. Može da izvede "bezopasne" akcije, kao što je prikazivanje božićne jelke na monitoru u vreme praznika.* Može da bude izuzetno destruktivan i da izbriše podatke sa hard diska, ili da uništi fajl sistem. U tim slučajevima njegov efekat obično može da se minimizira, ali samo ako ste redovno bekapovali svoje podatke! Ako niste, onda ste u ozbiljnoj nevolji.

Najgori virusi ne izazivaju masovna razaranja odmah po "inficiranju". U stvari, oni su veoma "suptilni" i izazivaju male promene (obično neprimetne) u fajlovima svaki put kada se pokrenu. Tokom vremena, promene se uvećavaju i na kraju bivaju primećene. Do tada, informacije su već oštećene. Da "stvar" bude još gora, ako ste marljivo bekapovali podafte, moguće je da su i oni "inficirani". Oporavljanje "neinficiranih" verzija fajlova može da bude veoma težak proces.

Kako se virus pripaja na izvršni fajl? Prvi korak je donošenje "inficiranog" fajla na kompjuter i njegovo pokretanje. Kada se pokrene, pridruženi virus može da "inficira" fajlove na različite načine. Na primer, može da istraži fajl sistem, proveravajući gde postoje drugi izvršni fajlovi (slika 7.27).



SLIKA 7.27 *Samostalno dupliranje virusa*

* Neki bezopasni virusi mogu, u stvari, da nanesu veliku štetu, čak i ako eksplicitno ne uništavaju, ili ne oštediju postojeće informacije. Uskoro ćemo navesti nekoliko primera.

Svaki put kada pronade izvršni fajl, virus može da izvrši instrukcije za samostalno dupliranje i smeštanje kopije u fajl, kao što je prikazano na slici 7.27.

Takvi virusi mogu lakše da se detektuju od ostalih. Proces traženja izvršnog fajla i njihova promena zahteva dodatnu aktivnost diska. Zbog toga, ako primetite veću aktivnost diska dok obavljate neke jednostavne zadatke, posumnjajte na virus. Hi, bilo bi još bolje da kupite neki antivirusni softver koji će pratiti aktivnosti koje se odvijaju na kompjuteru.

Virusi koji su rezidentni u memoriji

Umesto da skenira fajl sistem na disku, virus može da se kopira u memoriju i tamo čeka da se neki izvršni fajl smesti u memoriju. Kada se fajl smesti u memoriju, on ga napada. Hvatanje fajlova, jedan po jedan, dok se unose u memoriju predstavlja mnogo suptilniji oblik napada - slično lukavom grabljivcu koji se skriva i čeka svoj plen, koji dolazi neočekivano, ne sluteći da će postati nečiji obrok.

Kako se virusi koji su rezidentni u memoriji aktiviraju? Reč je o neaktivnom programu koji može da se aktivira samo po pozivu. Na PC-jima neki virusi koriste prednost mehanizama internih prekida. Tipično, BIOS (basic input/output system) i rutine servisa operativnog sistema lociraju se preko tabele prekida, ili vektora prekida. *Tabela prekida* je kolekdja adresa za servisne rutine. Kada korisnik zahteva neki servis, ili kada se desi neki asinhroni dogadaj koji zahteva akciju, operativni sistem locira traženi servis pronalaženjem njegove adrese u tabeli i započinje izvršavanje programa na toj lokaciji. Virusi koji su rezidentni u memoriji menjaju tabelu prekida tako što kreiraju adrese za lociranje virusa umesto servisa (slika 7.28).

SLIKA 7.28 *Virus rezidentan u memoriji*

Kada dode do prekida, mtina koja se locira preko adrese iz tabele u stvari je virus, koji obavlja svoje funkcije. Kao i ranije, može da pokuša da maskira ono što je uradio pozivanjem tražene servisne rutine, tako da korisnik misli da sve protiče u najboljem redu.

Razvoj virusa

Istorija kompjuterskih vimsa datira još od 1949. godine, iz vremena kada većina]judi nije ni znala da kompjuteri postoje. Džon Fon Nojman (John Von Neumann) je napisao rad pod nazivom "Teorija i organizacija složenih automata", u kome je predstavio teoriju po kojoj se kompjuterski programi mogu umnožavati. Tako je dao model kompjuterskog virusa. Pošto su jedini kompjuteri tog vremena bile velike mainframe mašine, nije bilo mnogo interesovanja za Fon Nojmanovu teoriju. Ipak, to se kasnije promenilo.

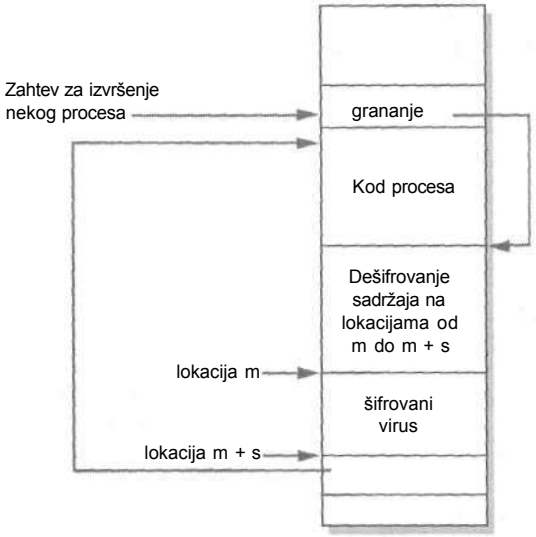
Pre sredine 80-ih prošlog veka kompjuterski virusi bukvalno nisu ni postojali. Otada im se povećava broj i postaju sve složeniji, a razvijeni su i metodi za njihovo detektovanje i eliminisanje. Prodavci softvera nude razne antivirusne pakete dizajnirane za lociranje poznatih virusa na personalnim kompjuterima. Kako antivirusni paketi postaju sofisticiraniji, tako napreduju i virusi. Nameće se logično pitanje zašto se virusi sve teže detektuju. U referencama [Na97] i [Ka94] data je istorija razvoja virusa i antivirusnih programa. Ovde predstavljamo tu temu u sažetom obliku.

Podnjemo proučavanjem načina za detektovanje poznatih virusa. Rani virusi iz polovine 80-ih godina prošlog veka bili su sasvim jednostavni i malobrojni. Antivirusni programi su funkcionisali tako što su tražili potpis vimsa, sekvencu bajtova koja se nalazi u virusu (koja odgovara instrukcijama na mašinskom jeziku). Pošto nije bilo mnogo viaisa, broj potpisa nije bio veliki, tako da njihovo pronalaženje nije predstavljalo ozbiljan problem.

Kako Je vreme prolazilo, desile su se dve "stvari" koje su otežale detekciju. Prvo, kreiran je veći broj virusa. Drugo, široka upotreba softverskih proizvoda i sve veći kapacitet diskova obezbedili su više mesta za skrivanje virusa. Prosto pretraživanje potpisa virusa medu fajlovima na disku postaje izuzetno dug proces. Autori antivirusnih paketa ističu da su virusi uglavnom kratki programi i da se, obično, postavljaju na početak, ili kraj izvršnog programa (setite se slike 7.26). Zahvaljujući ovoj činjenici, ti autori mogu da koncentrišu svoja istraživanja na početak i kraj takvih fajlova, što značajno povećava efikasnost njihovog rada.

Programeri virusa su počeli da shvataju da moraju nekako da sakriju potpis virusa, ili bar da ga maskiraju da ne bude prepoznatljiv. Zato su počeli da koriste šifrovanje - propuštanjem koda vimsa kroz algoritam za šifrovanje (čak i jednostavno Cezarovo šifrovanje) potpis se menja i antivirusni program ne uspeva da ga pronade. Jedini problem je što program čije su mašinske instrukcije šifrovane nije hteo da se izvršava.

Ovo je moglo da se zaobide umetanjem algoritma za dešifrovanje u fajl sa virusom (slika 7.29). Kada se proces pozove, komanda grananja u ulaznoj tački procesa prebacuje kontrolu na algoritam za dešifrovanje. Algoritam se izvršava i dešifruje sadržaj memorije u kojoj je smešten šifrovani virus. Kada se dešifruje, virus može da se izvrši, obavi svoj posao, a zatim vraća kontrolu originalno pozvanom procesu.



SLIKA 7.29 *Šifrovani virus na izvršnom fajlu*

Pošto se ovo dešava u memoriji, virus zadržava šifrovani oblik u okviru fajla i njegov potpis ostaje maskiran. Da bi "stvari" bile još teže za autore antivirusa, virus može da se napiše tako da svaki put kada "inficira" novi fajl koristi drugi ključ za šifrovanje. Zbog toga, on može da izgleda drugačije kod svakog novog "inficiranog" fajla.

Programeri antivirusnih paketa su reagovali dizajniranjem programa koji traže šablone bajtova karakteristične za algoritme za dešifrovanje. Problem je što je kod algoritma kratak i često je veoma sličan kodu legitimnih programa. Na primer, da li sledeći kod predstavlja algoritam za dešifrovanje Cezarove šifre, ili je reč o jednostavnom ažuriranju informacija u okviru niza?

```
for (i=0; i<s; i++)
    m[i]+=k;
```

Može da bude i jedno i drugo; teško je razaznati njegovu namenu bez poznavanja konteksta u kome se koristi.

Da bi bila eliminisana lažna upozorenja, antivirusni programi koriste tehniku poznatu pod nazivom *x-raying.* U suštini, program uzima sumnjivi šifrovani virus i propušta ga kroz kolekciju poznatih algoritama za dešifrovanje, koji su poznati po tome da se koriste sa virusima. Nakon toga, prelazi se na proučavanje šifrovanog rezultata i traže se potpisi virusa. Ovaj metod se pokazao kao uspešan.

Programeri virusa su uzvratili kreiranjem **polimorfnih virusa,** koji, u suštini, mutiraju nakon "inficiranja" novog fajla. To je slično biološkim virusima (kao što je virus SIDE), koji često mutiraju, tako da su otporni na lečenje obolelog. Svaki put kada "inficira" novi fajl, polimorfni virus koristi mehanizam mutacije za promenu koda koji odreduje algoritam za dešifrovanje. Iskusniji programeri znaju da postoje brojni načini da se defmiše kod koji obavlja specifični zadatak.

Polimorfni virusi jednostavno koriste prednost te činjenice. Zbog toga, svaka kopija polimorfnog virusa ne samo da se šifruje drugim ključem, već i za dešifrovanje koristi drugi algoritam. Ovo predstavlja ozbiljan problem za antivirusni program koji traži specifičnu sekvencu bajtova karakterističnu za viruse, ili za algoritme za dešifrovanje.

Kako se virus može locirati ako uopšte ne znate šta tražite? Zabeleženi su određeni uspesi u analiziranju nizova bajtova koje kreiraju mehanizmi za mutaciju i detektovani su neki šabloni. Problem je što su neki složeniji mehanizmi sposobni za generisanje više od bilion različitih oblika algoritama za dešifrovanje. Ogroman broj moguđh potpisa čini metode za detekciju traženjem potpisa praktično neprimenljivim.

Ipak, svaki virus mora eventualno da se dešifruje i da se otkrije da bi bio izvršen. To je ključni element koji omogućava uspešnu reakciju na polimorfne viruse. Problem je što je isuviše kasno ako antivirusni program mora da čeka na aktiviranje virusa. Neke tekuće tehnike za detektovanje virusa koriste tehnologiju generićkog dešifrovanja (GD-generic deoyption). U stvari, reč je o tehnici koja nastoji da prevari virus da se prerano otkrije, pre nego što dobije šansu da nanese neku stvarnu štetu. Preciznije rečeno, *CD antivirusni softver* sadrži CPU emulator. Kada ispituje fajl, pokreće softverski program koji simulira izvršenje fajla. Ako se u fajlu nalazi polimorfni virus, on se dešifruje i otkriva šablon svog potpisa. Periodičnim pozivanjem rutina za traženje potpisa virus može da se detektuje - čak i ako uspe da se izvrši pre nego što se detektuje, neće naneti nikakvu štetu, jer je reč o simulaciji.

Međutim, šta ako fajl ne sadrži virus? U nekoj tački simulacija mora da se okonča, ili pravi posao fajla nikada neće biti završen. Simulacija može da izvrši specifičan broj instrukcija, a zatim se okončava ukoliko virus nije pronađen. Ipak, ovaj metod nije u potpunosti otporan na greške. Virus može na svom početku da sadrži razne vrste beskorisnih instrukcija. Primer su NOP instrukcije (koje bukvalno ne izvršavaju nikakve funkcije), ili instrukcije koje vrše sabiranje sadržaja registra sa O. Svrha tih instrukcija je da odlože stvarnu aktivnost virnsa i da stvore utisak bezazlenog programa, bar za neko vreme. Ukoliko je simulacija isuviše kratka, neki virusi mogu proći neopaženo. Sa druge strane, ako traje isuviše dugo, korisnik postaje nestrpljiv dok čeka kompletiranje programa za detekciju i obično odmah piše žalbe prodavcu o lošem odzivu programa.

"Rat" izmedu autora virusa i svih ostalih se nastavlja. Tekući antivirusni softver je efektan i efikasan, ali neprestano se javljaju novi virusi, primoravajući korisnike da neprestano ulažu značajnu količinu novca i vreme u zaštitu svojih resursa.

Izvori virusa

Odakle virusi potiču? U početku, kreirali su ih individualci koji su pokušavali da zauzmu sistem. Blio da je svrha bila podvala, ili neprincipijelni i maliciozni destruktivni čin, obično nije bilo nikakvih posledica. Za žrtvu razlozi nisu bili bitni.

Poput bilo kog biološkog virusa, kompjuterski vinisi se šire deljenjem. Videli ste kako se fajl "inficira". Razmotrite šta se dešava ako se "inficirani" fajl iskopira na neki prenosivi medijum i ako se taj medijum ubaci u neki daigi kompjuter.

Kada se "inficirani" fajl pokrene, drugi fajlovi u kompjuteru mogu takođe da se "inficiraju". Ako se bilo koji od tih fajlova ponovo iskopira na neki prenosivi medijum i prenese na neki drugi kompjuter, virus se dalje širi. Zbog toga, morate da budete veoma obazrivi kada nabavljate softver.

Sve veća upotreba mreža i komunikacija samo je doprinela povećanju ozbiljnosti problema. Šta se dešava ako "inficirani" fajl dospe na Web sajt, u komercijalni softver, ili na fajl server na mreži? Virus sada može da se proširi na hiljade kompjutera u veoma kratkom periodu. Brzina njegovog mogućeg širenja je skoro zastrašujuća.

Nemojte na osnovu ove rasprave da mislite da su Web sajtovi, mreže i komercijalno distribuirani softver "raj" za viruse. Respektabilni menadžeri i prodavci ulažu ogromne napore da obezbede softver koji sigurno nije "inficiran" virusima. Medutim, ne postoje nikakve garancije za potpuni uspeh. Došlo je 1988. godine do incidenta kada je jedan komercijalni softverski paket sadržavao virus koji je prikazivao mirovnu poruku, a zatim se sam brisao. Ovo je primer "bezazlenih" virusa - oni su bezazleni zato što ne uništavaju fajlove i ne kradu dragocene informacije. Medutim, prodavac mora da popravi narušeni ugled i da povrati poverenje kupaca u svoje proizvode. Kompanija čiji su proizvodi bili "inficirani" može da propadne, tako da njeno zatvaranje postane neizbežno. Za proizvodača i njegove zaposlene ovakvi incidenti uopšte nisu bezazleni.

S obzirom da su virusi neizbežni, kako "izaći na kraj" sa njima? Kao i u slučaju raznih bolesti, prevencija je najvažnija. Danas imate na raspolaganju brojne antivirusne pakete. Ponekad paketi za detekciju virusa skeniraju medijum koji se umetne u drajv, tražeći vimse. Ako se virus detektuje, generiše se upozorenje i, u nekim slučajevima, kao, na primer, kod flopi diska, medijum se izbacuje. Korisnik može da zahteva od paketa da ukloni virus, ili da zameni "inficirane" fajlove "neinficiranim" bekapovanim fajlovima (uz proveru da li su validni).

7.8 Pretnje i napadi

Internet "crv"

Jedan od najčuvenijih upada izveo je Internet "crv". U nekoliko interesantnih dostupnih članaka opisani su "crv" i način njegovog funkcionisanja. Da, to je stara "priča", ali je vredi pročitati zbog potrebe da javnost upozna rizike kojima se izlažu povezani kompjuteri. Mi ćemo navesti samo opšti pregled, a zainteresovani čitaoci mogu da potraže reference [Sp89], [Ro89], [Se89] i [De90].

U novembru 1988. godine jedan diplomac sa Cornell univerziteta je na Internet otpustio "crv", koji je zauzeo hiljade Sun 3 i VAX kompjutera koji su koristili različite varijante 4 BSD UNIX operativnog sistema. Ovaj "crv" je pripadao takozvanoj bezazlenoj vrsti; nije uništavao nikakve informacije, niti je odavao otkrivene lozinke.

Sa druge strane, došlo je do ozbiljnog proboja sistema zaštite. "Crv" se brzo replicirao preko Interneta, zagušujući komunikacije i primoravajući mnoge sisteme na "gašenje". Osim toga, zbog njega su mnogi stručnjaci provodili dane u pokušajima da otkriju izvor problema. Inicirana je FBI istraga koja je trebalo da utvrdi da li je došlo do narušavanja zakona iz 1986. godine (Computer Fraud and Abuse Act).

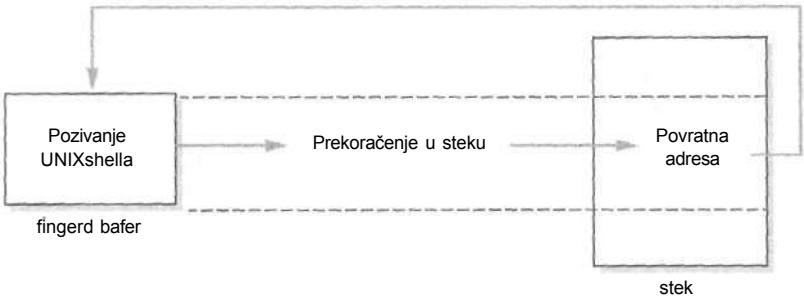
Slučaj je stigao do suda i federalna porota je proglasila optuženog krivim - osuden je na tri godine uslovno, kažnjen je novčano sa 10.000 dolara i morao je da provede 400 sati na društveno korisnom radu [Mo90]. Sve to je "zaradio" zbog jednog bezazlenog "crva". Kreiranje kompjuterskih "crva" i virusa je kriminalno delo, koje u većini slučajeva ispituje FBI. Podleže federalnim zakonima, koji su mnogo strožiji od većine zakona koji se primenjuju na nivou saveznih država.

Sam "crv" je napisan u programskom jeziku C, a napada UNIX sisteme koristeći propuste u softveru. Koristi se nekoliko tehnika, čije opise možete da pročitate u referenci [Sp89]. U jednom pristupu koristi se uslužni program pod nazivom fingerd koji jednom korisniku omogućava pribavljanje informacija od drugog korisnika. Dizajniran je tako da prihvati jednu ulaznu liniju sa udaljenog sajta (zahtev) i da pošalje nazad izlaz koji odgovara zahtevu (slika 7.30). Propust koji je ovde iskorišćen ogledao se u tome što ulazna komanda programa fingerd (C komanda gets) nije proveravala prekoračenje bafera. Zato se "crv" koji se pokreće na udaljenoj mašini mogao povezati na program fingerd i poslati spedjalno konstmisanu poruku koja izaziva prekoračenje ulaznog bafera programa fingerd.

Na slici 7.31 pokazano je šta se desilo. Preneta poruka je prekoračila ulazni bafer i prepisala delove sistemskog steka. Medutim, stek sadrži povratnu adresu (pozivne procedure) koja će se referencirati kada se fingerd završi. Zbog prekoračenja, ova adresa se menja na neku tačku u kojoj su smeštene neke instrukcije u baferu. Zato, kada se fingerd završi, kontrola se vraća na "program" koji je lociran na osnovu nove povratne adrese. Taj "program" se efektivno menja sa UNIX shellom (interfejs, ili komanda interpretatora). Rezultat je bilo to što je "crv" bio povezan na shell. Od te tačke "crv" komunicira sa shellom i eventualno šalje svoju kopiju, tako da se "inficira" sledeća mašina. Nakon toga, on nastavlja da ispituje sistemske fajlove, u potrazi za konekcijama ka novim mašinama koje bi mogao da "inficira".

Napada se i fajl sa lozinkom, radi pokušaja dešifrovanja korisničkih lozinki. Dešifrovanjem lozinke crv dobija mogućnost da napadne sve ostale kompjutere na kojima taj korisnik ima naloge.

SLIKA 7.30 *Program fingerd*



SLIKA 7.31 *Upad na sistem*

Ovde je interesantno napomenuti da su sve lozinke šifrovane pomoću DES-a. Teorijski, dešifrovanje lozinki bez ključa trebalo bi da bude skoro nemoguće. "Crv" ipak koristi jednostavniji pristup. lednostavno nagađa lozinke, šifruje ih i poredi sa originalnim šifrovanim lozinkama. Teorijski, broj mogućih lozinki je ogroman, tako da je ovakav metod traženja nepraktičan. Međutim, "crv" koristi reči iz online rečnika i u mnogim slučajevima uspeva da pronađe poklapanje. U nekim slučajevima više od 50 odsto lozinki ostaje neotkriveno (ref. [Sp89]). Naravoučenije: Ne koristite lozinke koje se mogu naći u rečniku.

Kompjuterski hakeri

Povezanost ogromnog broja kompjutera otvorila je brojna vrata za sledeću bezbednosnu pretnju - za kompjuterske hakere. U osnovi, haker je neko ko piše programe "čisto" iz zadovoljstva. Medutim, neki ljudi hakere vide kao neprincipijelne ljude koji pokušavaju da dobiju neautorizovani pristup kompjuterskom sistemu, često iskorišćavanjem bezbednosnih propusta u operativnim sistemima i otkrivanjem korisničkih lozinki. Zašto neprincipijelni hakeri rade to što rade? Teško je dati odgovor na to pitanje. Neki ljudi veruju da za većinu hakera upad na sistem predstavlja izazov, ili običnu zabavu. Takvi hakeri ili ne shvataju, ili ih nije briga za posledice njihovih akcija. Za neke otkrivanje privatnih informadja ne nosi istu težinu kao provala u nečiji dom. Ipak, to ne mora da znači da su takvi ljudi manje opasni. Postoje verovanja da kriminalac koji je pustio Internet "crva" nije imao maliciozne namere, jer nije pokušao da uništi informacije. Nesumnjivo, njegov efekat je bio dalekosežan i značajno je poremetio rad kompjutera. Ali, postoje i ljudi koji hakuju kompjutere radi krađe, ili promene informacija. Oni se često izlažu riziku da budu uhvaćeni, ili možda i ne brinu zbog toga. [La87] procenjuje motive nekih hakera i pokazuje da postoje neki opasni ljudi koji se bave hakovanjem.

Sveprisutnost mreža je stvorila mogućnosti za ozbiljnije i potencijalno opasnije probleme (ref. [Na91]). Na primer, hakerski incident koji je opisan u referencama [St88] i [St89] privukao je pažnju javnosti. Počelo je sa greškom u obračunu od 75 centi u Lawrence Berkeley Laboratory u avgustu 1986. godine, a završilo se hapšenjem hakera iz Zapadne Nemačke.

Haker je napadao mnoge kompjutere na MILNET-u (vojna mreža) i prenosio je informacije KGB-u, tajnoj policiji Sovjetskog Saveza. Nadležni sud Zapadne Nemačke je 1990. godine osudio hakera zbog špijunaže.

Iako su reference [St88] i [St89] dosta stare, ipak predstavljaju fascinantno štivo. Detaljno opisuju aktivnosti mnogih ljudi koji su nadgledali akcije "uljeza" i eventualno ga pratili sve do lokacije u Zapadnoj Nemačkoj. "Uljeza" je opisan kao veoma spretan, ali i uporan, jer je koristio primitivne napade sa pogadanjem lozinki. Kao i u slučaju Internet "crva", mnogi od tih napada su bili uspešni, jer su kao lozinke korišćene uobičajene reči koje se mogu naći u rečniku.

Ostale pretnje

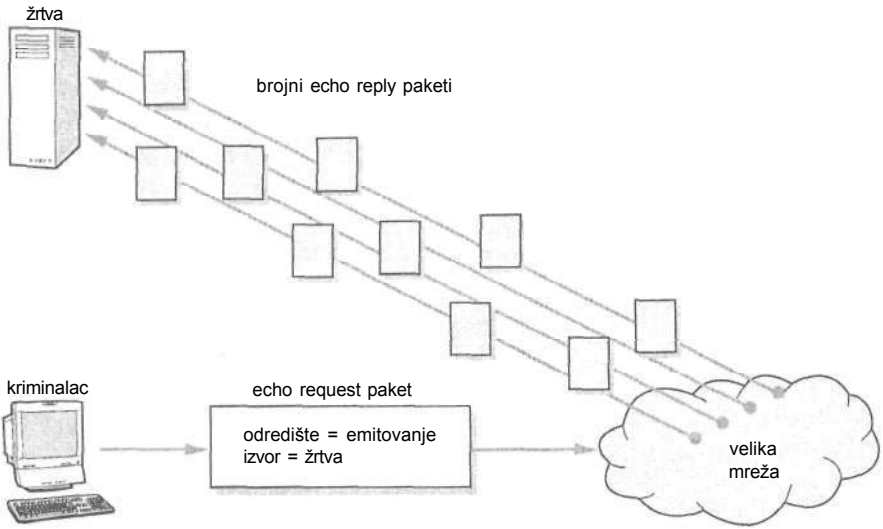
Napadi na kompjutere i mreže ne samo da se nastavljaju, nego su i sve učestaliji. Hihg-profile sajtovi mogu da budu meta više od milion napada u toku jednog meseca, mada se većina tih napada uspešno odbija. Ozloglašena vrsta napada je *napad odbijanja servisa* (DoS-denial of service). Jedan oblik DoS napada, nazvan **smurf napad,** obasipa mrežu sa toliko "dubreta" da nije moguće normalno funkcionisanje. Tako mrežni resursi postaju nedostupni legitimnim korisnicima, izazivajući gubitke u obavljanju poslova, vremenu i novcu. Ovo se izvodi prilično jednostavno i zasniva se na uobičajenoj mrežnoj komandi ping. Korisnik unosi komandu ping IP-adresa koja izaziva slanje ICPM* Echo Request paketa do hosta. Host reaguje slanjem sekvence Echo Response paketa nazad do korisnika. Osnovna ideja je da se utvrdi da li je sajt dostupan i koliko je vremena potrebno za slanje i povratak paketa.

Na slici 7.32 prikazano je kako to može da se zloupotrebi. Neko koga smatramo kriminalcem (nivo ove knjige nam onemogućava da ga nazovemo pravim imenom) šalje Echo Request paket na nezaštićenu mrežu. Međutim, on menja dve "stvari". Kao odredišnu adresu postavlja emisionu adresu, tako da se paket šalje do svih adresa u okviru date mreže. Osim toga, u polje izvorne adrese postavlja adresu žrtve. Svaka mašina koja dobije Echo Request paket reaguje onako kako bi trebalo - šalje niz Echo Reply paketa do izvora. Ako su Echo Request paketi poslati do velikog broja mašina, do žrtve stiže ogroman broj Echo Reply paketa, zatrpavajući server i efektivno gaseći normalne operacije servera.

Sledeći tip DoS napada je takozvani napad jwžfeg_532^&gmgaafa, koji se zasniva na protokolu koji TCP koristi za postavljanje konekcije. U Poglavlju 11 detaljnije analiziramo ovaj protokol; u suštini, njegovi koraci se sastoje iz sledećeg:

1. Klijent šalje TCP SYN segment, zahtevajući uspostavljanje konekcije.
2. Server šalje potvrdu o prijemu zahteva.
3. Klijent šalje nastavak za kompletiranje konekcije.

* To je Internet protokol koji ćemo predstaviti u Poglavlju 11



SLIKA 7.32 *Napad odbijanja servisa*

Sva tri koraka su neophodna za uspostavljanje konekcije. Kod napada poplavom SYN segmenata kriminalac šalje TCP SYN zahtev, ali, kao i kod smurf napada, koristi lažnu IP adresu. Server šalje potvrdu do prevarenog korisnika. Taj klijent nikada ne šalje nastavak, jer nije ni inicirao zahtev. Ovo izgleda kao kraj "priče", ali nakon što server pošalje svoju potvrdu, on kreira neku strukturu podataka kao pripremu za ono što misli da je eventualna konekcija. Ako nastavak nikada ne stigne, ističe vreme definisano za datu strukturu podataka i ona se eventualno otpušta. Medutim, ako kriminalac šalje lažne zahteve za uspostavljanje konekcije brzinom koja je veća od brzine kojom ističe vek strukture podataka, server žrtve može da ostane bez memorije, ili da dostigne granicu kada više ne može da prihvata konekcije od legitimnih korisnika.

DoS napadi nisu jedini bezbednosni problem. Packet sniffer je program koji prati pakete dok putuju preko mreže. Može da iskopira privatne informacije kao što su lozinke i kasnije ih može iskoristiti za pokretanje napada. CERT Coordination Center (uskoro će biti objašnjen) izveštava da su korisnici kablovskih modema izloženi visokom riziku, jer više korisnika u istom susedstvu koristi istu liniju. Pretnju mogu da predstavljaju i pretraživačke mašine. Prema referenci [HeOl], neki hakeri su pribegli korišćenju pretraživačkih mašina za pokretanje napada. Sledeća pretnja za privatnost informacija je spywere. Ti programi ulaze na nečiji sistem slično virusima i mogu da budu deo drugih programa koje su preuzeti i instalirani sa nekog manje poznatog sajta. Dok je pokrenut, spywere program ne utiče na postojeće fajlove, niti ometa tekuće aktivnosti na kompjuteru. Međutim, on beleži sve što radite (programe koje pokrećete, fajlove kojima pristupate, Internet sajtove koje posećujete) i podnosi izveštaje vlasniku spywere programa. Reč je o skrivenom programu koji može da se izvršava bez Vašeg znanja.

Kao što ste do sada verovatno pretpostavili, napadi na sisteme ne prestaju, a ne smanjuje se ni broj ljudi kojih ih iniciraju. Administratori mreža moraju da budu veoma oprezni prilikom konfigurisanja firewalla i moraju nepreslano da prate pojave svih novih napada. To je ozbiljan probkm. Neki hihg-profile sajtovi podnose izveštaje o više od jednog miliona odbijenih napada u toku samo jednog meseca. Sa sve većim prisustvom mreža u doslovno svim delovima sveta i sa sve veđm brojerrt Ijudi koji pišu, ili nabavljaju programe za iniciranje takvih napada, ovo problem koji sigurno neće nestati u dogledno vreme.

Jedan rnogući pristup u borbi sa ovakvim napadima jeste razvijanje protokola koji će obezbediti potvrdu identiteta i klijenta i servera pre nego što se konekcija uspostavi. Jedan takav proizvod (Kerberos), razvijen je u MIT, koristi kriptografske tehnike. Ima sličnosti sa TLS/SSL, ali je razvijen za klijent/server okruženja, za razliku od TLS/SSL, čija je primarna namena obezbedivanje pristupa Webu. To je bezbednosni standard sa otvorenim kodom. Najnovija inkarnarija, verzija 5, nastala je u saradnji sa IETF; specificirana *je* u RFC 1510. Kerberos je detaljnije predstavljen u referenci [CoOIb].

Sledeći bezbednosni paket je IPSec. Dok su paketi kao što su Kerberos, TLS/SSL i PGP bili razvijeni za rad na višim slojevima, IPSec je dizajniran za rad na IP sloju. Tako je omogućena nezavisnost od aplikacije i krajnjih korisnika. IPSec omogućava autentifikaciju IP paketa i šifrovanje i obezbeduje sredstva za upravljanje ključem. Pitanjima IP zaštite se bavimo u Poglavlju 11. Više detalja možete da pronadete i u referenci [PeOO].

Postoje organizacije koje su zadužene za praćenje sumnjivih aktivnosti. Jedna od njih je CERT Coordination Center, smeštena na Carnegie Mellon University. Originalno je nosila naziv Computer Emergency Response Team (datira iz 1988. godine) a stvorena je iz Defense Advanced Projects Research Agency (DARPA). CERT finansiraju U.S. Department od Defense (američko ministarstvo odbrane) i neke druge federalne civilne agencije. CERT beleži i obezbeduje mnoštvo informacija o napadima, zajedno sa preporukama kako se izboriti sa njima. Više detalja možete da pronadete na Web sajtu www.cert.org.

Iako je ovo poglavlje duže od većine ostalih u knjizi, samo smo "zagrebali" po površini problema zaštite. Postoje brojna tehnička, sociološka i pravna pitanja. Neki tvrde da je siguran samo kompjuter koji nije priključen na mrežu i koji je isključen. Iako je ova izjava delimično izrečena u šali, u njoj ima i istine. Mnogi Ijudi čuvaju svoje privatne informacije na kompjuterima koje koriste kao servere. To bi bilo isto kao kada biste poverljive dokumente stavili na sto u dnevnoj sobi, skinuli sva vrata i ispred kuće postavili tablu "svi posetioci su dobrodošli".

7.9 Zaključak

LI ovom poglavlju smo se se bavili problemima zaštite, uključujući šifrovanje, distribuciju ključa, autentifikaciju, bezbednosne pakete i protokole, firewalle, viruse i napade. Akcenat je stavljen na tehnike šifrovanja, zato što je šifrovanje vitalno ne samo za prikrivanje podataka koje želimo da sakrijemo od "uljeza", već i za digitalno potpisivanje dokumenata i njihovu autentifikaciju. Tehnike šifrovanja se dele na dve kategorije: sa privatnim i sa javnim ključem. Ključ LI kriptosistemima sa privatnim ključem mora da se čuva na tajnorn mestu, zato što "uljez" koji uspe da utvrdi ključ šifrovanja može da dešifruje sve poruke.

Tabela 7.3 Pregled algoritama za šifrovanje i generisanje digitalnog sažetka

Algoritam	Komentari
AES (Rijndael)	U skorije vreme definisani standard koji koristi 128-, 192- i 256-bitne blokove kao matrice i izvršava višestruke cikluse složenih operacija. Svaki ciklus uključuje S-box zamene, operacije pomeraja, isključivo ILI i druga matematička izračunavanja koja se zasnivaju na množenju polinoma i teoriji polja Galoa. Mnogi aspekti ovog algoritma mogu da budu izuzetno teški za razbijanje.
Šifrovanje na nivou bitova	Ključ je niz bitova. Operacije isključivo ILI između ključa i sukcesivnih delova poruke daju šifrovanu poruku. Svaki blok šifrovanog teksta se dešifruje pomoću još jedne operacije isključivo ILI, sa istim stringom. Ovaj metod se ne smatra teškim za "razbijanje", jer se zadržavaju razlike izmedu delova poruke u originalnom obliku. Na primer, ako su Pl i P2 dva različita dela obične tekstualne poruke koji se razlikuju na odredenim bitskim pozicijama, onda se i šifrovane verzije razlikuju na identičnim bitskim pozicijama.
Cezarovo šifrovanje	Ovaj metod menja jedan karakter nekim drugim. Uobičajene slovne sekvence su održane i zato se ova šifra lako "razbija". Nije prikladna za ozbiljnije aplikacije, ali se ponekad koristi kao deo složenije šeme.
Data Encryption Standard (DES)	Koristi složenu kolekciju premeštanja, zamena i operacija isključivo ILI. Metod je ranije bio kontroverzan, jer su neki smatrali da ga je National Security Agency namerno oslabila u odnosu na originalni predlog. Spekulisalo se da NSA ne želi metod za šifrovanje koji se ne može "razbiti" bez većih problema. DES algoritam je razbijen 1988. godine pomoću DES Cracker mašine kreirane u Electronic Frontier Foundation. Danas se ovaj algoritam smatra zastarelim.
Poliafabetsko šifrovanje	Poput Cezarovog šifrovanja, jedan karakter se menja nekim drugim karakterom. Razlika je to što izbor zamene za uobičajene karaktere zavisi od pozicije slova u poruci.
RSA algoritam	Ovaj metod predstavlja primer kriptografskog sistema sa javnim ključem koji vrši šifrovanje tretirajući niz bitova kao broj i stepenujući ga na visoki stepen korišćenjem modularne aritmetike. Ključ za dešifrovanje (privatni ključ) se veoma teško utvrduje, čak i kada je ključ za šifrovanje (javni ključ) opštepoznat.
Šifrovanje premeštanjem	Ovaj metod ne pokušava da prikrije karaktere, već ih preureduje. Njegova glavna namena je da posluži kao komponenta nekih složenijih šema.
Trostruki DES	Primenjuje DES algoritam tri puta uzastopno. Ako tri instance algoritma koriste različite ključeve, većina ovaj metod smatra teškim za "razbijanje".

U javnim kriptografskim sistemima ne vodimo računa o tome ko sve zna ključ. Pretpostavlja se da njegovo poznavanje ne može da bude korisno za utvrdivanje rutina za dešifrovanje. U tabeli 7.3 dat je pregled šema za šifrovanje koje su predstavljene u ovom poglavlju.

Osim algoritama za šifrovanje, u ovom poglavlju su predstavljeni sledeći bitni koncepti:

- **Distribucija i deponovanje ključeva** Da bi se kodovi dešifrovali, privatni ključevi moraju da se distribuiraju do odgovarajućih primalaca. Shamirov metod uključuje korišćenje ključa kao dela izraza sa polinomima. Tada se distribuiraju tačke sa grafika polinoma. Da bi se utvrdio ključ, minimalni broj ljudi mora da obezbedi dovoljan

broj tačaka za utvrdivanje originalnog polinoma. Diffie-Hellman razmena ključa poziva pošiljaoca i primaoca da razmene izračunate vrednosti na osnovu kojih može da se izvede ključ za šifrovanje. Deo postupka izračunavanja se drži u tajnosti; nedostatak kompletnih instrukcija trećoj strani značajno otežava izvodenje ključa. Ipak, Diffie-Hellman razmena je osetljiva na napade tipa man-in-the-middle. Ključevi za Clipper Chip su utvrdeni izvodenjem operacije isključivo ILI izmedu nasumičnb generisanih nizova bitova. Svaki od tih nizova se smešta u zasebnoj agenciji za čuvanje ključeva i obe agencije su neophodne za izvodenje ključa.

- **Digitalni potpisi** Digitalni potpisi predstavljaju način za autentifikaciju autora šifrovane poruke. Ideja je da se šifrovanje vrši privatnim, a dešifrovanje javnim ključem. Ako autor poruke kasnije poriče da je poslao tu poruku, primalac može da obezbedi i šifrovani i originalni tekst. Pošto je privatni ključ poznat samo autoru, jedino je autor mogao poslati tu poruku.

- **Autentifikacija** Ovo podrazumeva utvrdivanje da li je dokument autentičan, ili je falsifikat. Izračunava se vrednost digitalnog sažetka (message digest), broj koji na jedinstven način odreduje sadržaj dokurnenta. Taj broj se zatim potpisuje pomoću privatnog ključa kao kod kriptosistema sa javnim ključem. Skoro svaki pokušaj falsifikovanja dokumenta kreira dokument čija se vrednost digitalnog sažetka razlikuje od one koja je potpisana. Pomenuli smo dva algoritma koja generišu vrednosti digitalnog sažetka. MD5 algoritam daje 128-bitnu vrednost digitalnog sažetka. Poruka se deli na blokove od po 512 bitova (možda će biti neophodno proširivanje kako bi se dobio kompletan blok od 512 bitova), a zatim se izvode operacije na svim blokovima. Svaki blok prolazi četiri ciklusa operacija, u okviru kojih se izvode različite operacije na nivou bitova i vrši se faktorizacija na osnovu sinusne funkcije. Eventualno, generiše se rezultat koji se koristi kao ulazna vrednost za šifrovanje sledećeg bloka. Secure Hash Algoridim (SHA-I), slično MD5, radi sa blokovima od po 512 bitova. Svaki blok se deli na 16 32-bitnih reči. Grupe reči i preddefinisane konstante se propuštaju kroz različite cikluse logičkih operacija I, ILI, isključivo ILI i operacija pomeraja, tako da se na kraju generiše 160-bitna celobrojna vrednost.

- **Clipper Chip** Clipper Chip prate kontroverzni komentari, jer koristi algoritam koji je definisan u NSA. Uključuje i protokol koji pod određenim okolnostima omogućava pristup korisnikovom privatnom ključu (obično kada se sumnja na neke nelegalne aktivnosti). Pošto mnogi ljudi sumnjaju u Vladine aktivnosti, posebno kada je u pitanju privatnost, čip je pokrenuo brojne diskusije izmedu grupa za zaštitu privatnosti i službi za očuvanje reda i zakona, koji se spore da li treba voditi računa o privatnosti kada je reč o nelegalnim aktivnostima.

- **Prety Good Privacy** PGP je freeware prograin za zaštitu emaila koji je razvio Philip Zimmerman. Uključuje šifrovanje javnim ključem, autentifikaciju, digitalne potpise i kompresiju. Pokreće se na različitim platformama i koristi algoritme koji su detaljno ispitani, kao što su RSA za šifrovanje javnim ključem, MD5 za digitalne sažetke i IDEA za regularno šifrovanje. Pomalo je ozloglašen, zbog toga što je javno dostupan preko Interneta, tako da ga mogu koristiti i u stranim zemljama (izvan Sjedinjenih Američkih Država). Vlada Sjedinjenih Američkih Država je protestvovala zato što

smatra da je kriptografski softver deo vojne opreme i da se na ovaj način krši federalni zakon o izvozu. Skorije verzije su kreirane izvan Sjedinjenih Američkih Država da bi se rešio ovaj problem. Osim toga, pravila o izvozu ovakvog softvera su malo olakšana.

- **Transport Layer Security** TLS i Secure Socket Layer (SSL) su protokoli koji autentifikuju servere (i klijente, po potrebi) i pregovaraju o ključevima za šifrovanje i algoritmima neophodnim za sigurne prenose. Protokol se oslanja na X.509 sertifikat koji serveru izdaje telo nadležno za izdavanje sertifikata. Sertifikat potvrđuje autentičnost servera. Kada se server autentifikuje, informacije iz sertifikata, kao što je javni ključ, klijent može da iskoristi za slanje nazad do servera. Nakon toga, ove informacije se koriste i na strani klijenta i na strani servera za utvrdivanje ključa za naredno šifrovanje.

- **Firewalli** Firewall izdvaja kompjutere u okvim organizacije od Intemeta. Sva komunikacija izmedu kompjutera u okviru organizacije i udaljene mašine mora da prode kroz firewall. Firevvall može da blokira prenose na osnovu sadržaja IP paketa, ili akcija na sloju aplikacije. Takode, može da prihvati, ili odbaci TCP konekcije izmedu njih. Sve češće korišćeni pristup proučava pakete u kontekstu prethodnih aktivnosti (ispitivanje sadržaja na osnovu prethodnog stanja). Odredeni paketi se propuštaju ako su reakcije na prethodno legitimne zahteve.

- **Virusi, "crvi" i hakeri** Virusi su programi koji se pripajaju drugim programima. Njihove akcije mogu da budu različite i moguće je da budu visoko destruktivne. Poput virusa, "crvi" predstavljaju invazije na sistern, ali oni, u stvari, nisu deo nekog drugog programa. Kompjuterski hakeri su osobe koje pokušavaju da razbiju bezbednosne mere sistema. Mogu da pokušaju da ukradu privatne informacije, ili da postave viruse, ili "crve" koji će naneti neku štetu sistemu. Dva čuvena slučaja upada bili su Internet "crv", koji je napao hiljade kompjutera na Internetu, i upad programera iz Zapadne Nemačke u kompjutere MILNET-a. Broj napada na kompjutere širom sveta je u neprestanom porastu; među te napade ubrajaju se napadi odbijanja servisa, lažiranje adrese, praćenje sadržaja paketa i spywere.

- **Pravna, sociološka, etička i politička pitanja** Razvoj sigurnog koda je ozbiljan posao. Istakli smo neke kontroverze koje prate NSA i razvoj tehnika za šifrovanje. U stvari, komercijalni proizvodi za šifrovanje se tretiraju kao vojna oprema, tako da podležu istim zakonima kao i borbeni avion F-16. Ako Vas uhvate da ih prodajete van Sjedinjenih Američkih Država, možete biti obeleženi kao medunarodni trgovac oružjem. Predsednik Klinton je 1996. godine izdao izvršnu naredbu koja je trebalo da prebaci nadležnost za izvoz komercijalnih proizvoda za šifrovanje sa State Departmenta na Commerce Department, nakon što su U Commerc Departmentu donete regulative za implementaciju te naredbe. Nije dopušteno izvoženje svih algoritama za šifrovanje i neke informacije u vezi odredenih kriptografskih algoritama su i dalje strogo poverljive. Osim toga, ministarstvo trgovine i dalje može da se pozove na State Department i druge agencije radi revizije nekih izvoznih poslova.

Ovo je kontroverzno i veoma složeno pitanje. I dalje traju debate između onih koji se zalažu za zaštitu prava na privatnost i onih koji su protiv prava na privatnost u slučajevima kada se sumnja na nelegalne aktivnosti.

Pitanja i zadaci za proveru

1. Navedite razliku izmedu šifrovanja i dešifrovanja.
2. Navedite razliku između šifrovanog i običnog teksta.
3. Sta je Cezarova šifra?
4. Navedite razliku izmedu monoalfabetskog i polialfabetskog šifrovanja.
5. Da li su sledeće tvrdnje tačne, ili netačne (zašto)?
   a. Cezarova šifra nema nikakvu stvarnu vrednost kada je neophodna ozbiljna zaštita.
   b. Šifrovanje javnim ključem dopušta različitim ljudima da koriste isti ključ za šifrovanje, čak i kada se pretpostavlja da niko ne treba da zna šta je druga osoba poslala.
   c. Svaka blokovska šifra koja šifruje jedan po jedan blok u suštini predstavlja šifrovanje zamenom, gde se jedan blok menja drugim.
   d. Melod za šifrovanje koji koristi duži ključ je sigurniji od metoda koji koriste kraći ključ.
   e. Najozbiljniji virusi veoma brzo uništavaju ogromne količine podataka.
   f. Šifrovanje pomoću operacija isključivo ILI na sekcijama originalne poruke je siguran metod sve dok se bitovi ključa distribuiraju nasumično.
   g. Pod odredenim okolnostima, trostruki DES metod šifrovanja je bolji od običnog DES-a.
   h. Firewalli se obično koriste za zaštitu mreža od dolazećih virusa.
   i. Virusi koji ne uništavaju informacije, ili na drugi način ne ugrožavaju kompjuterski sistem su bezazleni.
   j. Napad na mrežu nije ozbiljan, ako nije dizajniran da zahvati privatne informacije, ili da smešta nelicencirane programe na kompjuter.
6. Šta ješifrovanje premeštanjem?
7. Ako je ključ za šifrovanje dovoljno dugačak, tehnike kao što je šifrovanje na nivou bitova je zaista nemoguće "razbiti". Zašto se više ne koriste?
8. Sta je Data Encryption Standard (DES)?
9. Šta je DES Cracker?
10. Objasnite razliku izmedu ECB (electronic codebook) moda i CBC (chiper block chaining) moda kada se primenjuje šifrovanje na nivou blokova.
11. Sta je vektor inicijalizacije?
12. Kakve kontroverze prate DES?
13. Zbog čega je Clipper Chip kontroverzan?
14. Koja je svrha korišćenja zasebnih agenata za čuvanje ključeva koji se koriste u Skipjack algoritmu?

15. Šta podrazumeva Shamirov metod za distribuciju ključa?
16. Šta podrazumeva šifrovanje na nivou blokova?
17. Koja je svrha S-boxa?
18. Kako izgleda napad tipa man-in-the-middle?
19. Po čemu se šifrovanje javnim ključem razlikuje od regularnog šifrovanja?
20. Šta je digitalni potpis?
21. Koje su glavne karakteristike RSA algoritma?
22. Zbog čega je RSA veoma težak za "razbijanje"?
23. Po čemu se autentifikacija razlikuje od digitalnih potpisa?
24. Kakav je značaj napada rodendana?
25. Šta je Prety Good Privacy?
26. Zašto bi se kriptografski algoritmi klasifikovali kao vojna oprema i zbog čega za njih važe stroga pravila za izvoz?
27. U čemu je razlika između uklanjanja i brisanja fajla?
28. Šta je X.509 sertifikat?
29. Svako može da kreira sertifikat autentičnog izgleda i da tvrdi da potiče od proverenog CA tela. Kako klijent može da utvrdi da li je sertifikat verodostojan?
30. Šta je firewall?
31. Navedite tri tipa firewalla, i opišite kako funkcionišu.
32. Firewalli filtriraju dolazeće podatke tražeći moguće pretnje po bezbednost sistema. Takode, ispituju se i odlazeći podaci. Zašto se to radi?
33. U čemu se ogleda nedostatak filtriranja paketa kada se proveravaju dolazeći paketi na moguće pretnje po bezbednost sistema?
34. Sta se podrazumeva pod ispitivanjem sadržaja na osnovu prethodnog stanja?
35. Navedite razlike izmedu virusa i crva.
36. Kako izgleda inficirani fajl?
37. Navedite neke moguće načine za sprečavanje širenja kompjuterskih virusa.
38. Šta je fingerd program na UNIX-U?
39. Sta su virusi rezidentni u memoriji?
40. Sta je bio Internet crv?
41. Sta Je napad odbijanja servisa?
42. Šta je spywere?
43. Šta radi packet sniffer?

Vežbe

1. Kako su utvrđene zamene za slova sa slike 7.2?
2. Razmotrite dve šifre zamene. Jedna dodaje vrednost *i* na ASCII kod karaktera običnog teksta. Druga dodaje vrednost j na karakter običnog teksta. Sva sabiranja se vrše po modulu

256. Razmotrite sada metod dvostrukog šifrovanja koji dodaje *i* na svaki karakter običnog teksta, a zatim na rezultujuđ šifrovani karakter dodaje vrednost j kako bi se dobio novi šifrovani karakter. I ta izračunavanja se izvode po modulu 256. Koliko je siguran metod dvostrukog šifrovanja u poređenju sa prethodnim metodima jednostrukog šifrovanja?

3. Napišite program za šifrovanje i dešifrovanje pomoću Cezarove šifre. Program kao ulaz zahteva unošenje ključa za šifrovanje.
4. Sledeća poruka je šifrovana pomoću Cezarove šifre. Kako glasi originalna poruka?

   fcvceqoowpkecvkqpucpfeqorwvgtpgvyqtmu
5. Napišite algoritam za dešifrovanje šifrovanog teksta koji je kreiran pomoću polialfabetskog šifrovanja iz sekcije 7.2.
6. Razmotrite šifrovanje premeštanjem primenjeno na tabelu 7.1. Kako glasi preneta poruka ako se kolone preurede u redosled 5, 1, 4, 2, i 3?
7. Napišite program za prihvatanje binarnog stringa i binarnog ključa. Nakon toga program treba da koristi ključ za šifrovanje stringa koristeći šifrovanje na nivou bitova.
8. Uzmite string 00101101010100001111111010011101 i ključ 10110. Pomodi ključa šifrujte a zatim dešifrujte string koristeđ šifrovanje na nivou bitova (koristeći operaciju isključivo ILI).
9. Prilagodite algoritam za šifrovanje na nivou bitova tako da radi u CBC (chiper block chaining) modu. Primenite ga koristeći string i ključ i vežbe 8.
10. Definišite algoritam za dešifrovanje za algoritam šifrovanja iz vežbe 9.
11. Pretpostavite da originalna poruka glasi "ABCDEF" i uzmite bilo koji 8-bitni ključ. Ako šifrovanje izvedete primenom operacije isključivo ILI izmedu ASCII koda svakog karaktera originalne poruke i ključa, kakvi se šabloni javljaju u rezultujućem šifrovanom tekstu?
12. Pretpostavimo da pokušavate da razbijete metod šifrovanja koji je koristio 64-bitni ključ. Hz preptostavku da se primenjuje brutalni napad, koliko ključeva u sekundi morate da isprobate da bi se metod razbio za jedan mesec?
13. Zašto trostruki DES defmiše algoritam dešifrovanja kao drugi od tri algoritma?
14. Konstruišite sliku sličnu slici *7.7* za blok od 256 bitova.
15. Ponovite proces šifrovanja poruke "HELLO" koji je diskutovan u sekciji 7.4 koristeći drugi ključ za šifrovanje, ali sa istom vrednosti za *n.*
16. Pretpostavimo da ste presreli sledeću šifrovanu poruku:

    20 5 21 3 49 4 49 3 4 15

    Takode, znate da je ključ šifrovanja $k = 7$, i da je utvrden korišćenjem $n = 55$. Dešifrujte ovu poruku. Pretpostavite da su slova A do Z inicijalno kodirana sa 1 do 26, a da je blanko znak inicijalno kodiran sa *27.*
17. Koristeći vrednosti $n = 47$, $g = 5$, $x = 10$, i $y = 12$, potvrdite da Diffie-Hellman razmena ključa funkcioniše. Sta je ključ za šifrovanje?
18. Izračunajte $95^{91} \bmod 121$.
19. Napišite algoritam (na programskom jeziku po sopstvenom izboru) koji će izračunavati $a^b \bmod n$, gde su *a, b* i n vrednosti tipa int.

20. Dizajnirajte klasu largelnt koja može da smešta cele brojeve od po *n* cifara (n je pozitivan celi broj). Klada treba da uključuje metod koji množi dve largelnt klase i stvara treću koja sadrži proizvod. Zabeležite koliko traje algoritam množenja za različite vrednosti n.
21. Jednostavni metod koji izračunava vrednost digitalnog sažetka razbija poruku na 128-bitne blokove, a zatim ih sumira. Da li je ovo efikasan algoritam sažimanja? Zašto jeste ili nije?
22. Preuzmite PGP sa web sajta i instalirajte ga na svom kompjuteru. Zamolite prijatelja da uradi isto. Koristite PGP za razmenu sigurnih i potpisanih poruka.
23. Pronadite web sajt koji nudi bezbednu konekciju. Sta se koristi kao javni ključ?

Reference

[CoOIa] Conry-Murray, A. "Firewalls for AU." *Netivork,* vol. 16, no. 6 (June 2001), 42-47.

[CoOIb] Conry-Murray, A. "Kerberos: Computer Security's Hellhound." *Network,* vol. 16, no. 7 (JuIy 2001), 40-45.

[Da02] Daemen, J., and V. Rijmen. *The Design of Rijndael: AES, the Advanced Encryption Standard,* Heidelberg, Germany: Springer-Verlag, 2002.

[De90] Denning, P., ed. *Computers Under Attack: lntruders, Worms, and Viruses,* Reading, MA: Addison-Wesley, 1990.

[De96] Denning, D.E., and D.K. Branstad. "ATaxonomy for Key Escrow Enayption Systems." *Communications of the ACM,* vol. 39, no. 3 (March 1996), 34-40.

[Di67] Diffie, W., and M. E. Hellman. "New Directions in Cryptography." *IEEE Transactions on Information Theory,* vol. 13 (November 1967), 644-654.

[Ga94] Garfinkel, S. PGP: *Pretty Good Privacy.* Sebastopol, CA: CReilly & Associates, 1994.

[HeOl] Hernandez,*}.,* et al. "Search Engines as a Security Threat." *Computer,* vol. 34, no. 10 (October 2001),25-30.

[Ka94] Kane, P. *PC Security and Virus Protection Handbook.* New York: M&T Books, 1994.

[KaOl] Kaliski, B. "RSA *Digital Signatures.*" *Dr. Dobb's Journal,* vol. 26, no. 5 (May 2001), 30-36.

[Ko77] Kolata, G.B."Computer Encryption and the National Security Agency Connection" *Science,* vol. 197(JuIy 1977), 438-440.

[La87] Landreth, B., and H.Rheingold. *Out of the Inner Circle: A Hacker's Guide* to *Computer Security,* Bellevue, WA: Microsoft Press, 1987.

[Mo90] Montz, L. "The Worm Case: From Indictment to Verdict." *InComputers Under Attack: lntruders, Worms, and Viruses,* ed. Peter Denning. Reading MA: Addison-Wesley, 1990.

[MoOl] Mollin, R. *An lntroduction to Cryptography.* Boca Raton, FL: Chapman & Hall/CRC Press, 2001.

[Na91] National Research Council. *Computers at Risk.* Washington, DC: National Academy Press, 1991.

[Na97] Nachenberg, C, "Computer Virus-Antivirus Coevolution." *Communications of'the ACM,* vol. 40, no. 1 (January 1997), 46-51.

[OpOl] Oppliger, R. *Secure Messaging with PGP and S/MIME.* Norvvood, MA: Artech House, 2001.

[PeOl] Perlman, E., and C. Kaufman. "Key Exchange in IPSec: Analysis of IDE." IEEE *Internet Computing,* vol. 4, no. 6 (November/December 2000), 50-56.

[Po02] Pohlmann, N,, and T. Crothers. *Firevjall Architecture for the Enterprise.* New York: Wiley, 2002.

[ReOl] Rescorla, E. *SSL and TLS: Designing and Building Secure Systems.* Reading, MA: Addison-WesIey, 2001.

[Ri78] Rivest, R.L., A. Shamir, and L. Adleman. "On a Method for Obtaining Digital Signatures and Public Key Cryptosystems." *Communications of the ACM,* vol. 21, no. 2 (Februarv 1978), 120-126.

[Ri92] Rivest, R.L. "The MD5 Message Digest Algorithm." RFC 1321, April 1992.

[Ro89] Rochlis, J., and M. Eichin. "With Microscope and Tweezers: The Worm from MIT's Perspective." *Communications ofthe ACM,* vol. 32, no. 6 (June 1989), 689-698.

[Sc94] Schneier, B. *Applied Cryptography.* New York" Wiley, 1994.

[Sc98] Schneier, B. "The Twofish Encryption Algorithm." *Dr. Dobb's Journal,* vol. 23, no. 12 (December 1998), 30-38.

[Se89] Seeley, D. "Password Cracking: A Game of Wits." *Communications of the ACM,* vol. 32, no. 6 (June 1989), 700-703.

[Sh79] Shamir, A. "How to Share a Secret." *Communications ofthe ACM,* vol. 22, no. 11 (November 1979), 612-613.

[Si96] Simonds, F. *Netuiork Security: Data and Voice Communications.* New York: McGraw-Hill, 1996.

[Sp89] Spafford, E. "The Internet Worm: Crisis and Aftermath." *Communications ofthe ACM,* vol. 32, no. 6 (June 1989), 678-687.

[Sp90] Spafford, E., K. Heaphy, and D. Ferbrache. "A Computer Virus Primer." In *Computers Under Attack: lntruders, Worms, and Viruses,* ed. Peter Denning. Reading, MA: Addison-Wesley, 1990.

[St88] Stoll, C. "Stalking the Wily Hacker." *Communications ofthe ACM,* vol. 31, no. 5 (Mayl988), 484-497.

[St89] Stoll, C. *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage.* NewYork: Doubleday, 1989.

[St95] Stinson, D. *Ciyptography: Theory and Practice.* Boca Raton, FL: CRC Press, 1995.

[St03] Stallings, W. *Ciyptography and Netmork Security: Principles and Practice,* 3rd ed. Upper Saddle River, NJ: Prentice-Hall, 2003.

[ThOO] Thomas, S. *SSL and TLS Essentials: Securing the Web.* New York: Wiley, 2000.

[Yu79] Yuval, G. "How to Swindle Rabin." *Ciyptologia,* vol. 3, no. 3 (JuIy 1979), 187-190.

[Zi95] Zimmerman, *P. The Official PGP User's Guide.* Cambridge, MA: MIT Press, 1995.

POGLAVLJE

8

Kontrola toka

Znanje u obliku raspoloživosti informacija već sada je neprocenljivo važno za produktivnost i nastaviće da bude glavna (možda i najznačajnija) "stavha" u svetskoj trci za moć. Moguće je da će se nacije jednog dana boriti za kontrolu nad informacijama na isti način hao što su se u prošlosti borile za teritorije, a kasnije za eksploataciju sirovina i jeftine radne snage.

—**Jean Fran^ois Lyotard** (1924-1998), francuski filozof

8.1 Uvod

Skoro sve što smo do sada predstavili odnosilo se na prenos od pošiljaoca do primaoca. Bilo da smo razmatrali digitalne, ili analogne signale, kompresiju, nadmetanje, zaštitu, ili integritet, osnovna tema se u opštem slučaju odnosila na jedan paket, ili okvir. Većina komunikacija je složenija i zato je neophodno razmotriti sledeća pitanja.

- Sta ako je preneta poruka veoma dugačka? Primeri su veliki fajlovi sa podacima, ili, na primer, kopija govora sa nekog političkog skupa. Tretiranje cele poruke kao jednog entiteta u toku jednog prenosa monopolizuje medijum. Ovo je dobro sa aspekta pošiljaoca, **ali** ne i sa aspekta svih ostalih koji čekaju da prenesu svoje podatke.
- Kako reagujemo na oštećene prenose? Prethodno smo istakli da primalac jednostavno zahteva ponavljanje prenosa. Ali, kako to primalac, u stvari, inicira? Da li protokol koji pošiljalac koristi u potpunosti zavisi od primaočeve sposobnosti da ga obavesti o oštećenim okvirima? Da li pošiljalac zaključuje da je okvir ispravno primljen ako ne primi nikakav zahtev od primaoca? Šta se dešava ako primaočev zahtev za ponavljanje prenosa bude oštećen, ili izgubljen?
- Šta ako kompjuteri pošiljaoca i primaoca ne funkcionišu na istim brzinama? Na primer, možete da preuzmete fajl sa nekog superkompjutera na personalni kompjuter star 10 godina. Hi, možda je primalac uposleniji od pošiljaoca. U opštem slučaju, postavlja se pitanje kako sprečiti pošiljaoca da zaspe primaoca sa više podataka nego što ovaj može da ih kontroliše.
- Šta se dešava ako se poslati okvir izgubi u toku prenosa? Na primer, okvir je oštećen u delu u kome se pamti adresa primaoca. U tom slučaju, okvir se nikada neće isporučiti

nikada ne stigne. Da li to znači da je okvir izgubljen, ili da ništa nije poslato? Kako primalac može da napravi razliku izmedu ta dva slučaja?

- U prethodnim primerima razlika između pošiljaoca i primaoca je bila jasno odredena, Šta se dešava ako oba istovremeno žele i da pošalju i da prime podatke? To je u neku ruku slično situaciji u kojoj istovremeno i govorite i slušate nekoga. Možda se svako povremeno nade u takvoj situaciji, ali činjenica je da tada propustimo deo onoga što nam neko govori. Kada je reč o komunikacijama, ne smemo da dopustimo da primalac izgubi njemu poslate informacije.

U ovom poglavlju predstavićemo dve značajne funkcije koje su neophodne za uspostavljanje i održavanje efektivne komunikacije: kontrolu grešaka i kontrolu toka. Kontrola grešaka defmiše kako uredajjroverava da li u okviru postoje greške i šta se radi ako se one detektuju. U odeljcima 6.2 i 6.3 pnkazani su nacini za detektovanje gresaka, ali nije receno sta se radi nakon detekcije. Uobicajeni pristup podrazumeva da prijemni uređai ukazuje da ie došlo do greške. Uređai pošiljaoca nakorLtoga može da obavi razlidte "stvari" i mi ćemo ovde razmotriti nekoliko protokola. U suštini, ponovo se zahteva sIanfe"6k>ifa7TiKoda se ovakav tip kontrole erešaka često naziva automatski zahtev za retransmisijom (ARQ-automatic repeat reauest).

Kontrolom toka definiše se način na koji se više okvira šalje i prati i kako uređali izvode kontrolu gresaka. U opštem slučaju, protokoli kontrole toka obezbeduju da svi povezani okviri stižu na svoje odredište odgovarajudm redosledom.

Protokoli mogu da budu predstavljeni ili na krajnje jednostavan, ili na složen način. U odeljcima 8.2 i 8.3 predstavljeni su relativno prosti protokoli kontrole toka. Postoji široki opseg različitih protoko-Ia, od onih koji šalju po jedan okvir u jednom trenutlcu.(stop-and-wait), do onih koji šalju sve okvire odjednom (neograničeni tok). U ovim odeljcima je razmotrena i upotreba specifalnih signala, ili specifičnih vrednosti bajtova koje ukazuju kada treba poslati podatke. To je analogno regulisanju saobraćaja na visokoprometnim putevima. Sve dok je upaljeno zeleno svetlo, saobraćaj može da se odvija. Ali, kada saobraćaj dostigne odredeni stepen zasićenja, pali se crveno svetlo.

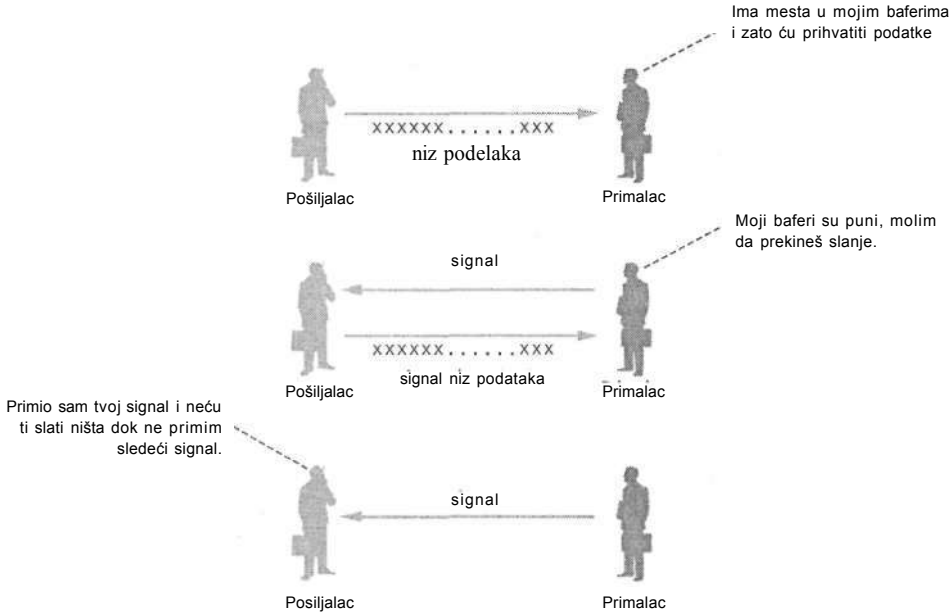
U odeljcima 8.4 i 8.5 prikazan je složeniji pristup koji numeriše okvire i šalje ih samo po nekoliko istovremeno. Nakon toga, pošiljalac čeka na potvrdu pre nego što pošalje još okvira. Protokol g-back-n predstavljen u odeljku 8.4 pretpostavlja da okviri stižu istim redosledom kojim su poslati. Protokol selektovanog ponavljanja, predstavljen u odeljku 8.5, dopušta slučajeve u kojima se neki okviri mogu odložiti i isporučiti mimo redosleda. U odeljku 8.6 analiziran je algoritam za te pristupe i definisane su formule za utvrdivanje efektivnih brzina prenosa podataka.

Predstavljanje protokola i načina njihovog funkcionisanja je jedno, a potvrda da su tačni je nešto sasvim drugo. Kod jednostavnih algoritama verifikacija je često laka, ali složeniji zahtevaju neke specijalizovane alatke. U odeljku 8.7 (pošto je prilično teorijski, može se "preskočiti" bez gubitka kontinuiteta) prikazani su neke alatke za verifikaciju, kao što je Petri net, i modeli konaćnih stanja.

Ovo poglavlje je više teorijsko nego većina ostalih i ne odnosi se na specifične mrežne protokole. Ipak, to ga ne čini manje značajnim. Zapravo, zbog toga je možda i značajnije, jer daje osnovu za razmatranje protokola veze podataka i protokola transportnog sloja, koje je predstavljene u narednim poglavljima. Ti protokoli će Vam biti jasniji ako već razumete kontrolu toka, potvrde, prozore, brojeve okvira, okvire koji stižu van redosleda, izgubljene okvire i tajmere. Pokušajte sada da razumete koncepte, a njihovu implementaciju ćemo prikazati nešto kasnije.

8.2 Signaliziranje

Ovaj odeljak uvodi relativno elementarne pristupe za kontrolu toka koji su korisni za jednostavnije komunikacione sisteme. Prvi pristup, sasvim jednostavan, predstavlja *signaliziranje* (slika 8.1). Pošiljalac prenosi podatke sve dok je primalac sposoban da ih prihvati. Ipak, primalac možda neće biti sposoban da prihvata podatke neprestano. Na primer, baferi koji prihvataju podatke mogu da se napune, ili primalac možda neće biti spreman zbog zauzetosti nekim drugim poslovima. U takvim slučajevima primalac šalje signal do pošiljaoca. Po prijemu tog signala, pošiljalac prekida prenos. Osim toga, protokol dopušta slanje originalnog signala kada primalac ponovo bude spreman za prihvatanje novih podataka. Ovaj pristup je analogan neproduktivnom argumentu kao kada jedna osoba kaže: Stani! Ne želim više da te slušam."



SLIKA 8.1 *Kontrola toka kada se koristi signaliziranje*

DTE-DCE kontrola toka

U odeljku 4.4 predstavljen je jedan način za signaliziranje spremnosti i prihvatanja podataka preko EIA-232 interfejsa. Uključuie slanje signala preko specifičnih linija (DTO i DSR), čime se ukazuje na spremnost uredaja za prihvatanje podataka. Kada DTE želi da pošalje nešto do DCE, šalie drugi slignal fRTS) kojim zahteva dozvolu za slanie. Nakon toga, čeka na Clear to Send signal (CrS) koji mu dopušta slanje. Detalji su predstavljeni u odeljku 4.4, tako da ih nećemo ovde razmatrati.

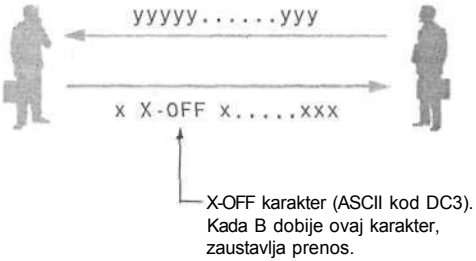
X-0N/X-OFF

EIA-232 interfejs je složen po tome što zahteva zasebne linije za zasebne signale. Sledeći pristup podrazumeva slanje signala kao dela prenetih podataka; to se naziva signalizirarije u opsegu (in-band signaling). U ovom slučaju, primalac treba da analizira dolazeće podatke, tražeći bilo koje specijalne signale na koje mora da reaguje.

ASCII set karakteia definiše dva kontrolna karaktera za kontrolu toka (videti tabelu 2.6 u odeljku 2.5). Simbolički, .to su DC3 (heksadecirnalni kod 13) i DCl (heksadecimalni kod 11); još se nazivaju i X-OFF i X-QN. respektivno..* Ponekad se koriste za kontrolu toka izmedu radne stanice i servera. Na slici 8.2 pokazano je kako ovo funkcioniše.

Na slici je pretpostavljeno da se koriste full-duplex komunikacije, tako da nema razlike između pošiljaoca i primaoca. A i B šalju i primaju podatke jedan od drugoga. Ako se baferi u A napune, može da reaguje umetanjem X-OFF karaktera u podatke koje šalje do B. Kada X-OFF karakter stigne, B ga vidi i prestaje da šalje podatke do A (ipak, napomenimo da A i dalje šalje podatke za B). Ako A kasnije oslobodi mesto u baferu, može da pošalje X-ON karakter do B, čime se šalje signal za B da može da nastavi prenos.

Kada jedan uredaj šalje X-OFF karakter, nastavlja da prima podatke još neko kratko vreme, zbog toga što postoji manje kašnjenje između vremena kada se X-OFF karakter pošalje i vremena kada drugi uredaj može da reaguje.



SLIKA 8.2 *Kontrola toka pomoću in-band signaliziranja*

* Ovi karakteri obićno odgovaraju control-S i control-Q sekvencama sa tastature kada se koriste odredeni emulatori terminala.

Zbog toga, uređaj obično šalje podatke kada njegovi baferi premaše neki prag.

Ovaj protokol ste prvi put verovatno sreli sasvim slučajno. Na primer, uobičajeno je da na ekranu, kada se emulator terminala koristi za povezivanje na server, dobijate prikaz sadržaja tekstualnog fajla. Povremeno, bilo zbog greške, ili nenamemo, možete da naidete na komandu koja prikazuje sadržaj binarnog fajla kao da je reč o izvršnom fajlu. Rezultat su čudni karakteri na ekranu, neki beep zvukovi i nasumično kretanje kursora. U nekim slučajevima radna stanica prestaje da reaguje na naredne unose sa tastature - tastatura se koči. Pošto većina bajtova u binarnom fajlu ne odgovara karakterima koji se mogu odštampati, radna stanica često reaguje na njihov sadržaj na neočekivani način. Kodovi za prebacivanje na početak reda i vertikalni i horizontalni tabulator izazivaju nasumično pomeranje kursora. Karakteri mogu da sadrže i BEL kod (heksadecimalno 07), koji izaziva beep zvuk (ovim se narušava uobičajeno verovanje da su beepovi upozorenja da će se radna stanica uništiti).

Problem "zamrzavanja" se javlja ako jedan od bajtova u fajlu sadrži X-OFF karakter. Pošto radna stanica primi taj karakter, ona reaguje zaustavljanjem prenosa nazad ka serveru. Ako i dalje zadajete ulaz preko tastature, kodovi neće biti poslati i ekran radne stanice se "zamrzava". Moguća rešenja su isključivanje radne stanice, ili prelazak u lokalni mod radne stanice i čišćenje komunikacije.

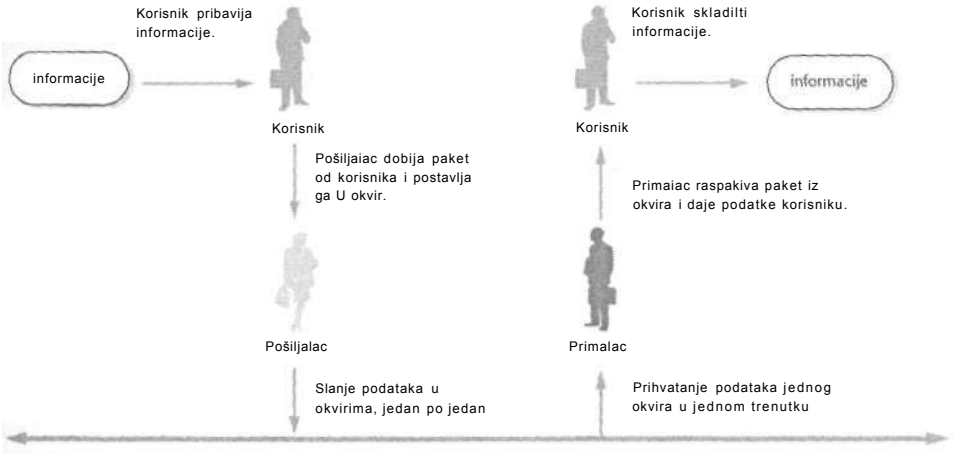
Sledeća uobičajena upotreba ovog protokola dešava se kada se na ekranu prikazuje neki veliki fajl. Da bi se sprečilo "isklizavanje" informacija sa ekrana, unesite control-S (zadržite taster Ctrl i pritisnete S) sa tastature da bi se ekran "zamrznuo". Control-S šalje X-OFF karakter, čime se zaustavlja prenos fajla. Kasnije, kada prodtate ono što ste hteli, možete da unesete control-Q, čime se šalje X-ON karakter koji dopušta nastavak prenosa fajla.

8.3 Kontrola orijentisana okvirima

Protokoli kao što je X-ON/X-OFF orijentisani su bajtovima i tipični su za asinhrone komunikacije (videti odeljak 4.3). Tj., prenos može da se startuje i pauzira u bilo kom bajtu. Sinhrone komunikacije (videti odeljak 4.3) su orijentisane okvirima i zahtevaju više organizacije. Informacije se šalju i "oporavljaju" u većim blokovima, a ne kao niz bajtova. Pošto uredaj mora da bude sposoban za baferovanje svih bajtova u okviru koji primi, različiti protokoli se koriste za ograničavanje broja okvira koje je moguće poslati. Naravno, postoje različiti načini za primenu ograničenja.

Zatim, treba uzeti u obzir činjenicu da onaj ko šalje i prima informacije obično ne vodi računa o okvirima i njihovoj strukturi. Ako se fajl prenosi preko Internet konekcije, sigurno nećete hteti da se bavite tim detaljima. Jednostavno, izdajte komandu, ili kliknite dugme za slanje fajla, a softveru prepustite sve ostale detalje. Zbog toga, većina protokola deli informacije tako da se šalju u okvirima sa odgovarajućim formatom, a zatim ih šalje. Na slici 8.3 ilustrovano je kako se ovo izvodi u tipičnom slučaju.

Neko, ili nešto koga nazivamo korisnikom ima informacije koje mora nekome da pošalje. Napomenimo da u ovom kontekstu ne moramo da mislimo na stvarnu osobu.



SLIKA 8.3 *Slanje i prijem podataka između dva korisnika*

Ovde termin *korisnik* predstavlja sledeći viši sloj u protokolu koji obuhvata više slojeva. Ovo ima smisla, jer kod protokola koji obuhvata više slojeva svaki sloj se smatra korisnikom sloja koji se nalazi ispod njega. Na primer, kod OSI modela sloj sesije je korisnik transportnog sloja, a sloj mreeže je korisnik za sloj veze podataka. Oba ova primera su relevantna, jer su protokoli za kontrolu toka značajan deo sloja veze podataka i transportnog sloja. Pošiljalac uzima dovoljno informacija (paket) od korisnika i postavlja u okvir, a zatim prenosi taj okvir. Primalac prihvata okvir, rastavlja paket i prenosi informacije do korisnika kome su potrebne. Proces se ponavlja za naredne okvire, sve dok ima informacija koje treba preneti.

Tipično, pošiljalac, primalac i korisnici sa slike 8.3 definišu sukcesivne slojeve u nekom komunikacionom softveru. Ovo je tipična interakcija između veze podataka (pošiljalac i primalac) i sloja mreže (korisnika) u OSI modelu. Kontrola toka postoji i na protokolima viših slojeva, kao što je TCP/IP (opisujemo ga kasnije u ovoj knjizi). U ovoj tački, gde pošiljalac, primalac i korisnik postoje, to nije značajno. Kontrola toka postoji u različitim modelima i na različitim slojevima. Medutim, važno je razumeti da je kontrola toka obično deo interakcije izmedu dva susedna sloja istog protokola.

Neograničenijjcotokol

Najlakšinrotokol, neograničeni protokol, pretpostavlja da primalac ima neograničaai kapacitet *Za* prihvatanie okvira, ili za prpcesiranie dovoliaom brzinom tako da uvek postoji slobodan prostor u baferu.

```
void senddata;
{
  while there are packets to send
  {
    get packet from the user;
    put packet into a frame;
    send(frame);
  }
}
```

Kod posiljaoca

```
Void receive_data;
{
 while there are frames to receive
  {
   wait for frame to arrive;
   receive(frame);
   Extract packet from the frame;
   Give packet to the user;
  }
}
```

Kod primaoca

SLIKA 8.4 *Neograničena kontrola toka*

Slika 8.4 pokazuje da je logika pošiljaoca i primaoca napisana delimično u C-u.* Pošiljalac i primalac koristi primitivne pozive send i receive. Ovo su obično pozivi na sloju ispod pošiljaoca i primaoca koji vode računa o detaljima potrebnim za prenos, ili prihvatanje okvira.

Pošiljalac neprestano izvršava petlju sve dok postoje informacije za slanje. Sa svakim prolaskom kroz petlju od korisnika se uzima paket, postavlja se u okvir i okvir se šalje. Okviri se neprestano šalju i ne pokušava se da se ograniči broj okvira koji se šalje. I primalac neprestano izvršava petlju. Pretpostavljamo da je primalac uvek sposoban za prihvatanje okvira. Sa svakim prolaskom kroz petlju primalac čeka (tj. ili se suspenduje, ili je u stanju čekanja) sve dok ne stigne sledeći olcvir. Po pristizanju okvira primalac se budi i prihvata okvir. Paket se izvlaći iz okvira i prenosi do korisnika, a zatim se vraća u stanje ćekanja do dolaska sledećeg okvira.

Ovaj pristup ne razmatra probleme koje smo prethodno razmotrili. Nema nikakvih pokušaja da se proveri da li su okviri oštećeni, izgubljeni, ili kasne, niti se kontroliše broj poslatih okvira. Pretpostavlja se da svi okviri stižu bez oštećenja istim redosledom kojim su poslati. Ovo je slično našoj zavisnosti od toga kada će pošta isporučiti pristigla pisma.

Protokol stop-and-wait

Protokol stop-and-wait (stani i čekai) razlikuie se od prethodnoe protokola na dva načina. Prvo. svaki put kada primalac dobije okvir šalje potvrdu nazad do pošiljaoca. Potvrda ie takođe okvir koji se šalje nazad do pošiljaoca. Nakon što pošalje okvir, pošiljalac čeka na potvrdu pre negojtp pošalje sledeći okvir. Tako, umesto da se okviri šalju brzo jedan iza drugog u nizu, protokol šalje jedan okvir, čeka na potvrdu, šalje sledeći, čeka potvrdu za njega i tako dalie. U nekim slučajevima, stop-and-wait protokol predstavlja suprotni ekstremni slučai prethodnoe metoda.

* Nemamo nameru da ovde navodimo sintaksno ispravan kod. Kombinujemo sintaksu C-a sa neformalnim naredbama kako bi se predstavilo značenje programa, bez zalaženja u detalje konkretnog piogramskog jezika. Dobra vežba može da bude modifikovanje svih protokola koje predstavljamo u sinlaksno i logički tačnc programe.

Za razliku od neograničenog protokola koji šalje maksimalan broj okvira u jedinici vremena, ovaj protokol šalje minimalan broj okvira.

Na slici 8.5 predstavljeni su protokoli pošiljaoca i primaoca. Protoko] primaoca je sličan neograničenom protokolu. Sadrži beskonačnu pedja u kojoj se čeka na dolazak okvira. Kada okvirstigne, primalac proverava da li je oštećen. Za našu diskusiju nije bitno da li korisd CRC, ili neki drugi oblik provere parnosti (o tome je bilo reči u Poglavlju 6). Važno je to što može da detektuje oštećeni okvir.

Ako okvir nije oštećen, primalac definiše polje greške u strukturi zapisa pod nazivom ack kao 0. Nastavlja da izvlači paket iz okvira, i predaje ga korisniku. Ako je okvir bio oštećen, polje greške ack je 1. Primalac ne izvlači paket iz okvira i korisnik ne dobija nikakve podatke. *U* svakom slučaju, primalac šalje potvrdu nazad do pošiljaoca. Pošto polje greške definiše status primljenog okvira, pošiljalac može da reaguje na odgovarajući način.

Pošiljalac neprestano izvršava pedju, šaljuci okvire i čekajući na potvrde, Pre slanja okvira, pošiljalac mora da odluči da li će poslati novi okvir, ili će ponovo poslati stari. To radi preko iokalne promenljive pod nazivom damaged, čija vrednost ukazuje na status postednjeg poslatog okvira (1 znači da j'e okvir bio oštećen; O znači da nije bio oštećen). Inicijalna vrednost je O. Na početku pedje, pošiljalac proverava damaged. Ako *je 0,* pošiljaiac uzima novi paket od korisnika, postavlja ga u okvir i šalje ga. Ako je damaged bilo I_1 pošiljalac šalje tekući okvir (onaj koji je prethodno poslat). U svakom sluča/u, nakon slanja okvira čeka na potvrdu. Kada potvrda stigne, pošiljalac proverava polje greške, čiju je vrednost definisao primalac. Ako polje greške ukazuje na oštecenje u prediodnom okviru, pošiljalac definiše damaged = 1.

```
void send_data;
{
  damaged0;
  while there are packets to send
  {
    if (/damaged)
      /* !0 is the same as true in C */
    {
     Get packet from the user;
     Put packet into a frame;
    }
    send(frame);
    Wait for acknowledgment to arrive;
    receive(ack);
    if ack.error
      damagedl;
    else
      damagedO;
  }
}
```

Kod posiljaoca

```
void receivejJata;
{
   vvfille there are packets to receive
   {
     Wait for frame to arrive;
     receive(franie) ;
     Examine frame for transmission error;
     if no transmission error
     {
        ack.error0;
        Extract packet from the frame;
        Give packet to the user;
     }
     else
        ack.errorl;
      send(ack);
   }
}
```

Kod primaoca

SLIKA 8.5 *Stop-and-ivait kontrola toka*

Kada se sledeći put pellja izvrši, pošiljalac ne uzima nove podatke od korisnika, već ponovo šalje okvir. Ako je damaged = 0, pošiljalac uzima novi paket u sledećem prolasku i šalje ga.

Kao što smo ranije opisali, stop-and-wait protokol deluje kao poželjniji izbor od neograničenog orotokola. iDak, ima i neke nedostatke:

- Ako je poslati okvir izgubljen, primalac nikada ne šalje potvrdu i pošiljalac čeka beskonačno dugo.
- Ako se izgubi potvrda primaoca, dešava se isto,
- Ako se potvrda ošteti, pošiljalac može da donese pogrešan zaključak i tako dolazi do greške u protokolu.
- Pošiljalac sigurno neće zasuti primaoca ogromnim brojem okvira, ali zato može da ode u drugu krajnost. I pošiljalac i primalac provode dosta vremena čekajući. To je kao kada predavač diktira jedno po jedno pitanje za pismeni zadatak. Student nosi pitanje kući, vraća se u školu, predaje rad, čeka da ga predavač oceni, zatim uzima sledeće pitanje i neprestano ponavlja ceo postupak. U nekim slučajevima bi bilo efikasnije poneti sve zadatke kući, završiti ih i vratiti se sledećeg dana.

U sledeće dve sekcije proučavamo ove primedbe. Pre nego što pređemo na složenije protokole, upoznaćemo efikasnost protokola.

Efikasaast protokola

Efikasnost može da se meri na nekoliko načina. Na primer, koliko protokol zahteva prostora u baferu? Za stop-and-wait protokol nikada nije potrebno mesta za više od jednog okvira, tako da je bafer kapaciteta iednog okvira sasvim dovoljan. Kod neograničenog protokola okviri mogu da stižu,brže nego što ih primalac može prihvatiti. Zato se oni moraju privremeno smeštati u bafer. Broi smeštenih okvira ie određen brzinom koiom stižu i koliko brzo primalac može da ih podeli. U svakom slučaju, stop-and-wait protokol zahteva manje prostora i zato je iz ove perspektive efikasniji.

Sledeća korisna mera je **efektivna brzina prenosa podataka.** To ie stvarni broi bitova podataka (za razliku od izvorne bitske brzine) koji se pošalje u iedinici vremena. Da bi se izračunala erektivna brzina podataka, dehmo broj poslatih bitova (N) sa protekhm vremenom lzmedu slanja dva okvira.

Ilustrujmo to jednim primerom. Uzimite u obzir naredne defmicije, sa konkretnim vrednostima u zagradama koje će biti korišćene u primeru:

R = bitska brzina (10 Mbps, ili 10 bitova/^sec)

S = brzina signala (200 metara/|j.sec)

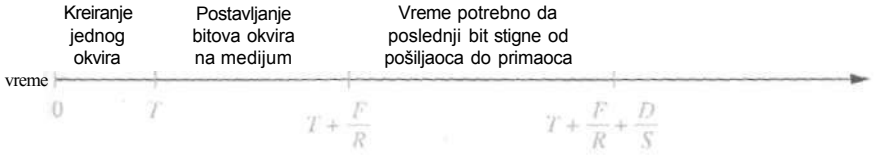
D = rastojanje izmedu pošiljaoca i primaoca (200 metara)

T = vreme potrebno za kreiranje jednog okvira (1 irsec)

F = broj bitova u okviru (200)

N = broj bitova podataka u okviru (160)

^A = broj bitova u potvrdi (40)



SLIKA 8.6 *Vreme potrebno za slanje okvira do primaoca*

Počinjemo utvrdivanjem vremena koje je neophodno za konstruisanje i slanje okvira (slika 8.6). Pretpostavljamo da pošiljalac startuje od trenutka O. Pošiljalac je dobio informacije i postavlja ih u okvir za vreme *T.* Sledeći korak je prenos okvira. Pošto je *R* bitska brzina, onda je *IfR* vreme potrebno za prenos jednog bita. Zato je F/R vreme potrebno za prenos jednog okvira. Ukupno potrošeno vreme do sada iznosi *T + F/R.*

Kada je pošiljalac poslao okvir, bitovima treba vremena da stignu do primaoca. Vreme putovanja iznosi *D/S.* Nakon što se prenese poslednji bit, potrebno mu Je *DfS* vremenskih jedinica da stigne do primaoca. Zbog toga, primalac prima poslednji bit u vreme = T + *F/R* + *D/S.* Napomenimo da okvir u putu provede *P/R* + *D/S* vremena.

Kod stop-and-wait protokola primalac mora da pošalje i potvrdu. Sličan argument pokazuje da je vreme potrebno da pošiljalac primi potvrdu jednako *T* + *AfR* + *DfS.* *

Sledeće pitanje: Koliko vremena protekne između slanja dva okvira podataka? Kod neograničenog protokola pošiljalac počinje da kreira sledeći okvir čim prenese poslednji bit prethodnog okvira. Kod stop-and-wait protokola on mora da sačeka potvrdu za svaki poslati okvir. Tako vreme između slanja dva uzastopna okvira iznosi

$$\text{vreme} = T + \frac{F}{R} \tag{8-1}$$

za neograničeni protokol i

$$\text{vreme} = \left(T\frac{F}{R} + \frac{D}{S}\right) + \left(T\frac{A}{R} + \frac{D}{S}\right)$$

$$= 2\left(T + \frac{D}{S}\right) + \frac{F+A}{R} \tag{8-2}$$

za stop-and-wait protokol.t

* Strogo govoreći, količina vremena koja jc neophodna primaocu da konstruiše okvir potvrde ne iznosi T. Medutim, to vreme zavisi od brzine CPU-a, efikasnosti kompajliranog koda i softverskog planiranja. Da bismo pojednostavili "stvari", pretpostavljamo da i pošiljalac i primalac mogu da konstruišu okvirza isto vreme.

t Ovde je pretpostavljeno da primalac šalje potvrdu čim primi okvir. Medutim, to nije uvek tako.

Dakle, deljenjem broja bitova podataka sa proteklim vremenom između slanja dva okvira dobijamo

efektivna brzina prenosa podataka (neograničeni protokol) $= \frac{N}{T + \frac{F}{R}}$

$$= \frac{160\text{bits}}{1\mu\text{ sec} + \frac{200\text{bits}}{10\text{bits}/\mu\text{ sec}}} \qquad [8\text{-i J}$$

$$\approx 7.6\text{bits/m sec} = 7.6\text{ Mbps}$$

i

efektivna brzina prenosa podataka (stop-and-wait protokol) $= \frac{N}{2\left(T + \frac{D}{S}\right) + \frac{F+A}{R}}$ **(8-4)**

$$= \frac{160\text{bits}}{2\left(1\mu\text{sec} + \frac{200\text{meters}}{200\text{meters}/\mu\text{sec}}\right) + \frac{200\text{bits} + 40\text{bits}}{10\text{bits}/\mu\text{sec}}}$$

$$\approx 5.7\text{bits/msec} = 5.7\text{Mbps}$$

Važno je napomenuti da kapacitet izvome bitske brzine ne garantuje da će se toliko podataka preneti. U ovom primeru stop-and-wait protokol realizuje samo oko 57 odsto izvorne bitske brzine. Efektivna brzina prenosa podataka umnogome zavisi od protokola, veličine okvira, rastojanja koje okviri prelaze i tako dalje. Na primer, povećavanjem veličine okvira povećava se i efektivna brzina prenosa podataka (uz pretpostavku da sa povećanjem okvira postoji propordonalno povećanje broja bitova podataka). Ovo možda nije toliko očigledno na osnovu prethodne jednačine, ali isprobajte i vidite šta će da se desi. Možete li da utvrdite kakav efekat ima povećanje drugih promenljivih na efektivnu brzinu prenosa podataka?

Ove mere predstavljaju samo deo celokupne slike i ne možemo da tvrdimo da je neograničeni protokol bolji samo zato što obezbeduje veću efektivnu brzinu prenosa podataka. Među ostalim faktorima koje treba uzeti u obzir nalaze se korisnid koje protokoli opslužuju, količina podataka koja se prenosi i činjenica da i drugi dele isti medijum. Činjenica je da ova dva protokola predstavljaju dva ekstremna slučaja (ili se šalje sve odjednom, ili se šalje jedan po jedan okvir) i svi ostali protokoli se nalaze negde izmedu. Predstavićemo ih u naredna dva odeljka.

8.4 Go-Back-n: Protokol klizajućih prozora

Prethodni protokoli su prikladni ako broj okvira i rastojanje izmedu uređaja nisu veliki. Ako se broj okvira poveća, neograničeni protokol može da "poplavi" medijum i zatrpa primaoca. Jednačina 8.4 pokazuje šta se dešava sa stop-and-wait protokolom ako se rastojanje poveća. Pošto je *D* u imeniocu, efektivna brzina prenosa podataka se smanjuje. Teorijski, biranjem dovoljno velike vrednosti za D brzina postaje proizvoljno mala.

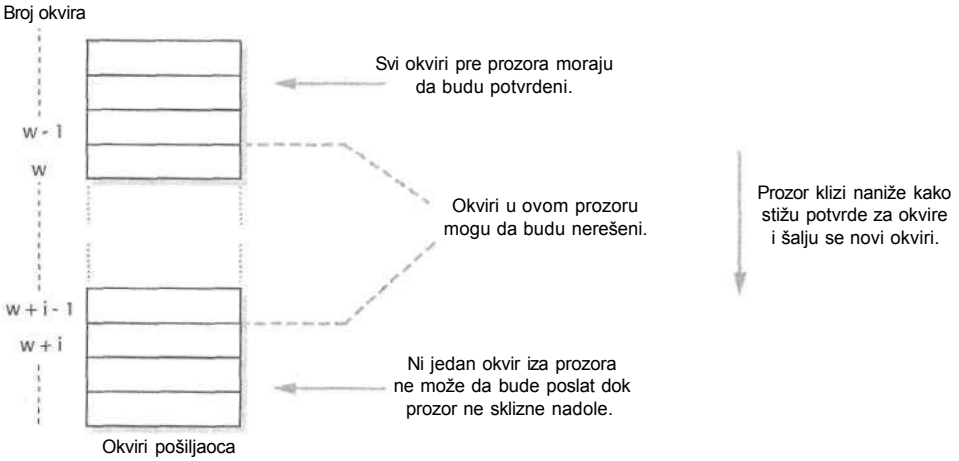
Pošto se komunikacije često odvijaju preko velikih rastojanja i uključuju velike količine podataka, potrebni su altemativni protokoli. Jedan mogući pristup je kompromis između neograničenog i stop-and-wait protokola, poznatkao **protokol klizajucih prozora.** On numeriše okvire koji se šalju i definiše **prozor** kao podskup uzastopnih olcvira. Ako prozor sadrži i okvira, počevši od w (w i i su celi brojevi), važe sledeće konstatacije (slika 8.7):

- Svaki okvir koji ie numerisan broiem manjim od w ie već poslat i potvrden.
- Ni jedan okvir koji ie numerisan brojem većim od w + *i* još uvek nije poslat.
- Svi okviri iz prozora su poslali, ali možda nisu stigle potvrde.za njih. Oni za koie nisu stigle potvrde predstavljaju nerešene okvire **(outstanding frames).**

Inicijalno, prozor sadrži okvire koji startuju sa okvirom O. Kako korisnik obezbeđuje pakete, prozor se proširuje tako da uključi nove okvire, koji se zatim šalju. Ipak, ograničenje veličine prozora predstavlja granicu za broj nerešenih okvira. Kada se ta granica dostigne, pošiljalac ne uzima nove pakete od korisnika. Kada stigne potvrda za nerešene okvire, prozor se smanjuje kako bi se isključili potvrđeni okviri. Nakon toga se prozor ponovo proširuje kako bi uključio nove okvire za slanje.

Dok se prozor menja, prethodni uslovi moraju uvek da budu ispunjeni i prozor uvek mora da sadrži sukcesivne okvire. Na primer, ako je stigla potvrda za okvir w + 1, ali nije stigla za okvir *w,* prozor ne može da se promeni sve dok ne stigne potvrda za *w.* Cak i ukoliko su stigle potvrde za sve okvire osim za *ui,* prozor se neće promeniti. Okviri se isključuju iz prozora istim redosledom kojim su i uključeni.

Ovaj pristup predstavlja kompromis, jer omogućava slanje više (mada ne nužno svih) okvira pre nego što stigne potvrda za sve poslate okvire. Maksimalna veličina prozora defmiše broj okvira koji mogu da budu nerešeni.



SLIKA 8.7 *Protokol klizajućih prozora*

Ako je prozor veličine 1, u suštini imamo stop-and-wait protokol. Ako je veličina prozora veća od ukupne količine okvira, dobijamo neograničeni protokol. Podešavanje veličine prozora može da pomogne kontroli saobraćaja na mreži i da odredi zahteve za baferovanje.

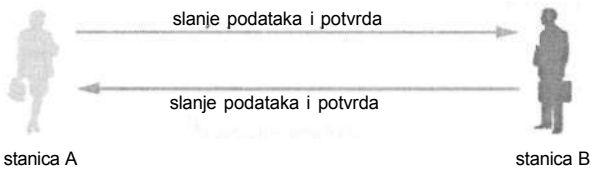
Postoje dve uobičajene implementacije protokola klizajućih prozora. **Go-back-n** protokol zahteva prijem okvira istim redosledom kojim su i poslati.* **Protokol selektivne retransmisije** to ne zahteva. Go-back-n je jednostavniji, jer primalac izbacuje svaki okvir, osim onog čiji prenos treba ponoviti. Selektivna retransmisija zahteva od primaoca da bude sposoban da zadrži okvire koji su primljeni van redosleda pre nego što ih prosledi do višeg sloja odgovarajućim redosledom.

Format okvira

Kod ovih protokola smo ispustili prethodnu pretpostavku i eliminisali jasnu granicu izmedu pošiljaoca i primaoca. Naime, pretpostavili smo realniji model kod koga dva uredaja (A i B) međusobno šalju (i primaju) okvire (slika 8.8). To je konverzacija, ili full-duplex komunikacija. Zato protokol mora da bude sposoban ne samo da šalje, već i da prima okvire.

Pogledajmo ukratko šta okvir stvarno sadrži. Na slici 8.9 prikazana su tipična polja. U kasnijim poglavljima ćemo videti neke specifične formate. U okviru postoje sledeća polja:

- **Izvorna adresa (Source)** Ovo je adresa uređaja koji šalje okvir. Često je neophodna kako bi prijemni uredaj znao gde da pošalje potvrdu.
- **Odredišna adresa (Destination)** Ovo je adresa na koju treba poslati okvir. Neophodna je da bi uredaj mogao da utvrdi koji su okvin njemu namenjeni.
- **Broj okvira (Number)** Svaki okvir ima sekvencu brojeva koja počinje od O. Ako ovo polje ima K bitova, najveći broj je 2^K - 1. Više od 2^K okvira izaziva komplikacije.



SLIKA 8.8 *Dvosmema komunikacija izm.edu uredaja A i B*

* Razlozi zbog kojih okviri mogu da stignu van redosleda su različiti. To je nalik poštanskim ispostavama. Pismo koje ste poslali u ponedeljak verovatno će stići pre onog koje ste poslali u utorak, ali nemojte previše da se oslanjate na to. Gust saobraćaj, hardverske, ili softverske greške i oštećeni okviri mogu da doprinesu kašnjenju, ili, čak, gubitku nekih okvira.

- **ACK** Celobrojna vrednost ovog okvira je broj okvira koji je potvrden. Napomenimo da pošto uredaj može i da šalje i da prima okvire, može da izbegne slanje zasebne potvrde uključivanjem potvrde u okvir sa podacima. Ovo je poznato kao piggybadting.

- **Tip okvira (Type)** Ovo polje definiše tip okvira. Na primer, podaci koji se prenose u okviru su odredenog tipa. Medutim, ponekad se potvrda prenosi u zasebnom okviru. Piggybacking može da se koristi samo kada postoje podaci za slanje; bez podataka, protokol koristi zasebne potvrde, koristeći okvir tipa "ACK". Koristi se okvir i tipa "NAK" (negativna potvrda) za problematične situacije. Na primer, protokol šalje NAK okvir kada je primljeni okvir oštećen, ili ako je stigao pogrešan okvir. U svakom slučaju, protokol omogućava obaveštavanje drugog uređaja da nešto nije proteklo kako treba.

- **Podaci (Data)** Ovim su predstavljeni podaci u okviru.

- **CRC** Ovo odgovara bitovima koji se koriste za proveru grešaka (videti odeljak 6.3).

Karakteristike

Protokol go-back-n ima nekoliko identifikujućih karakteristika:

- Brojevi okvira moraju da se nalaze izmedu 0 i 2^K - 1 *(K* = broi bitova u polju Number), zaključno. Ako postoji više od 2^K okvira, brojevi okvira sedupliraju. Na primer, pretpostavite da Je $K = 6$ i da postoji više od 64 okvira za slanje. Okviri od O do 63 su numerisani kao O do 63. Medutim, brojevi od 64 do 127 takode su numerisani kao O do 63. U opštem slučaju, svi okviri su numerisani po modulu 2K. Videćemo da ta karakteristika postavlja ograničenja za veličinu prozora koja uredajima omoeućava ispravno interpretiranje broieva okvira.

Ovo zahteva i neznatna podešavanja u načinu defmisanja prozora. I dalje je obavezno da prozor sadrži sukcesivno numerisane okvire.* Medutim, sada smatramo O sledećim okvirom nakon 2^K 1. Na primer, ako je $K = 6$, onda su okviri numerisani kao 62, 63, 0, 1, 2 i tako dalje, po modulu $26 = 64$.

- Prijemni uređai uvek očekuie da primi okvire u skJadu *sa redosledom (mod* 2^K*)* bro/eva okvira. Ako se okviri prime van redosleda, okviri se ignorišu i šalje se NAK *za okvir koji* /e bio očekivan. Nakon toga se čeka na dolazak odgovarajućeg okvira.

Source	Destination	Number	ACK	Type	...Data...	CRC

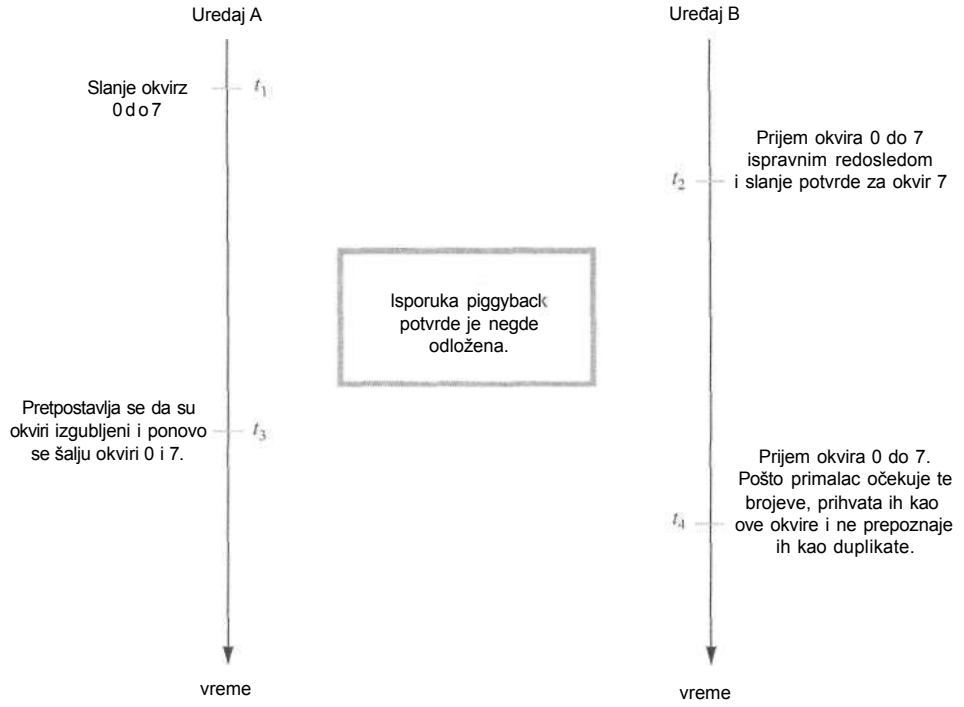
SLIKA 8.9 *Tipični format okvira*

* Ubuduće, kada kažemo broj okvira, mislimo na vrednost koja sc nalazi u polju Number datog okvira.

- □□□ j□ pristigli okvir oštećen, prijernni uredaj ga ignoriše i šalje NAK za niega.
- Prijemni uređai ne šalje eksplicitnu potvrdu za svaki primlieni okvir. Ako uređaj pošiljaoca primi potvrdu za okvir j, a kasnije primi potvrdu za okvir *k (k >j),* on pretpostavlja da su okviri izmedu njih ispravno primljeni. Tako se redukuje broj potvrda i proreduje saobraćaj na mreži. Naravno, uredaj koji šalje potvrde mora da bude siguran da je ova pretpostavka validna.
- Uređaj koristi piggyback pristup svaki put kada je moguće poslati potvrdu za najskorije primlieni okvir. Međutim, ako se nikakvi okviri sa podacima ne šalju u tom periodu, uredaj mora da pošalje zaseban okvir za potvrdu. **ACK tajmer** se postavlja svaki put kada stigne okvir sa podacima. On odbrojava unazad i zaustavlja se kada uredaj ima neke podatke za slanje. Razlog za to je činjenica da, kada okvir stigne, za njega mora da se pošalje potvrda u periodu koji je definisan ACK tajmerom. Ako nema odlazećih okvira, tajmer odbrojava do O. Kada se dostigne vrednost 0 (istek), uredaj šalje zasebni okvir potvrde, umesto piggyback potvrde. Ako uredaj šalje okvir sa podacima u vreme odbrojavanja tajmera, tajmer se zaustavlja, jer potvrda odlazi sa tim okvirima.
- Uredaj pošiljaoca baferuje pakete iz svih okvira u prozoru u slučaju da ih je neophodno ponovo poslati. Paketi se uklanjaju iz bafera kada stigne potvrda za njih.
- □□□ uređai ne primi potvrdu u odredenom periodu, pretpostavlja se da ie nešto krenulo naopako i da jedan, ili više nerešenih okvira nije stigao do svog odredišta. Tada se koristi tajmer okvira, po jedan za svaki okvir, koji se postavlja nakon slanja svakog okvira. Tajmer okvira odbrojava unazad i zaustavlja se kada stigne potvrda za dati okvir. Ako tajmer okvira istekne, protokol inicira ponovno slanje svakog okvira u prozoru.

  Razlog za slanje svih nerešenih okvira je to što prijemni uređaj odbacuje sve okvire sa pogrešnim brojem. Ako je prijemni uredaj dobio prvi okvir iz prozora, uredaj pošiljaoca je dobio potvrdu za njega. Ako nema potvrde, pretpostavlja se da se nešto desilo u meduvremenu. Takođe, smatra se da je prijemni uredaj, ako nije dobio prvi okvir, odbacio sve naredne okvire. Zato je neophodno ponovo poslati sve okvire. Ukoliko postoji n okvira, vraća se na početak prozora i šalje ih ponovo. Odatle naziv go-back-n.

Koliko okvira protokol može da ima sa nerešenom statusom u jednom trenutku? Dmgim rečima, kolika je maksimalna veličina prozora? Ako su okviri numerisani od 0 do 2^K - 1, veličina prozora ne može da bude veća od 2^K. Ako bi bila veća, postojalo bi više od 2^K nerešenih_oJbdra. Zato bi postojali nerešeni okviri sa istim brojem. Kada uredaj pošiljaoca primi potvrdu sa tim brojem, ne postoji način da se utvrdi koji je od dva moguća okvira potvrden. Na primer, pretpostavimo da je $K = 3$ i da je prvih devet okvira nerešeno. Prvih osam okvira je numerisano sa O do 7. Poslednji je numerisan kao 0. Ako uređaj pošiljaoca primi potvrdu za okvir 0, ne zna da li ta potvrda odgovara prvom, ili poslednjem okviru.

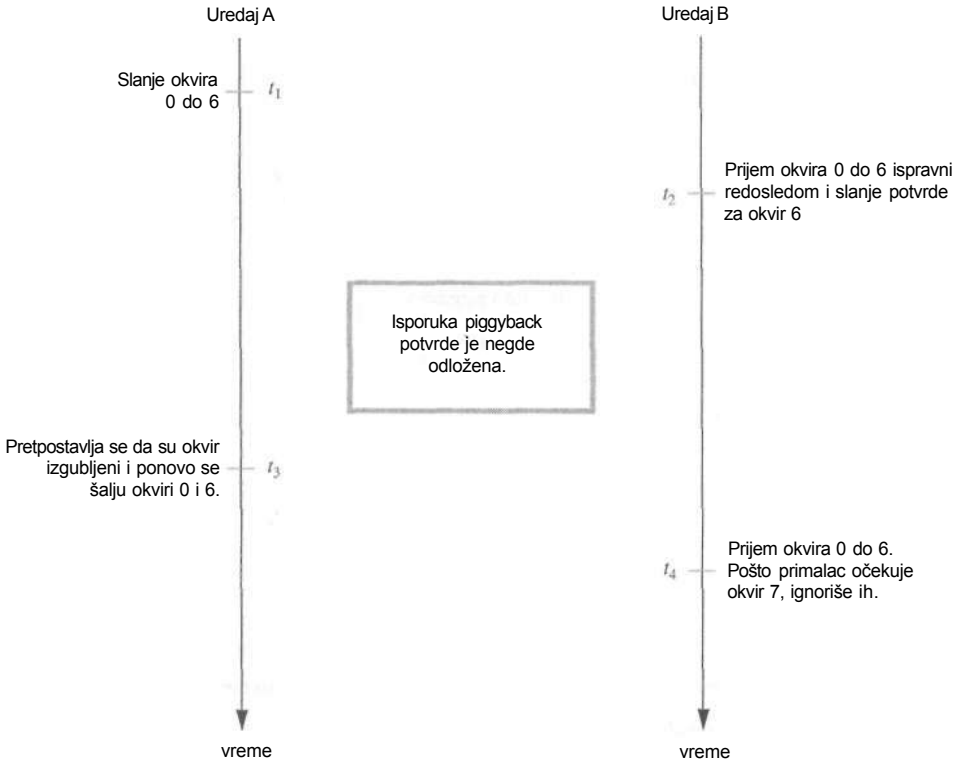


SLIKA 8.10 *Greška u protokolu kada je veličina prozora jednaka 2K*

Na osnovu ovoga, zaključujemo da veličina prozora mora da bude manja, ili jednaka 2^K. Medutim, ako je veličina prozora jednaka 2^K, nesrećna sekvenca dogadaja i dalje može da izazove greške u protokolu. Pretpostavite da je $K = 3$ i razmotrite dogadaje prikazane na slici 8.10. Pretpostavite da su oba uređaja razmenila okvire pre trenutka ^. U trenutku t, uredaj A šalje okvire 0 do 7 do uredaja B. Uređaj B ih prima ispravnim redosledom i u trenutku t_2 šalje potvrdu za najskorije primljeni okvir, pod brojem 7. Nažalost, ova potvrda se gubi negde u toku prenosa, zbog hardverske, ili softverske greške, ili nekog gremlina na linijama.

Uređaj B ne može da zna da je potvrda izgubljena i čeka na okvir koji sledi iza okvira 7 (okvir 0). Uredaj A, sa druge strane, ne prima potvrdu i ne zna da li su okviri stigli, ili ne. Prateći protokol, on inicira ponovno slanje okvira 0 do 7 u trenutku t_3. U trenutku t_4 uredaj B prima okvir 0. Problem je to što je ovaj okvir 0 duplikat prethodnog okvira 0. Ali, uredaj B očekuje novi okvir 0 i ne postoji način da on utvrdi da je primio duplikat. Zato prihvata duplikat kao novi okvir i dolazi do greške u protokolu.

Problem se javlja zato što dva sukcesivna prozora sadrže iste brojeve okvira. Uredaj B ne može da zna u kom prozoru se okvir nalazi. Redukovanjem veličine prozora na I moguće je ispraviti ovaj problem. Na slici 8.11 pokazano je šta se dešava ako se slični dogadaji dese kod smanjene veličine prozora. Ovde uredaj Ašalje okvire 0 do 6 u trenutku tl i uredaj B ih sve prima.



SLIKA 8.11 *Uspešno izvršeni protokol kada je veličina prozora 2K 1*

U trenutku t_2 uređaj B šalje potvrdu za okvir 6 i potvrda se izgubi negde u toku prenosa. Razlika je u tome što uređaj B sada očekuje okvir 7. Kada A ponovo pošalje okvire 0 do 6 (u trenutku t_3), oni u B stižu u trenutku t4. Pošto ih ne očekuje, B ih ignoriše. Eventualno, šalje još jednu potvrdu, koju A prima (nadamo se).* Uređaj A pomera svoj prozor tako da uključi okvir 7 i protokol se nastavlja. Na osnovu ovoga, zaključujemo da veličina prozora mora da bude manja od 2^K, ili će doći do greške u go-back-n protokolu.

Algoritam

Konačno smo spremni da predstavimo detaljniji opis go-back-n protokola. Na slici 8.12 pokazan je parcijalno kodirani C program koji sadrži logiku i nazive promenljivih. Slika 8.7 treba da Vam pomogne da razumete upotrebu promenljivih protokola *w* i *i.*

```
#define MAX=2K;                          /* K = broj bitova u polju frame.number */
#define N=MAX 1;                         /* N je maksimalna velicina prozora i
                                            najveci broj okvira */
#define increment(x) x=(x+1) % MAX;      /* Inkrementiranje x po modulu MAX */
void go_back_N;
{
  int w=8;                               /* Prva pozicija u prozoru */
  int i=0;                               /* Tekuca pozicija prozora */
  int laSt=N;                            /* Broj okvira poslednjeg primljenog okvira */
  packettype buffer MAX ;                /* Baferi za pakete */
  while (zemlja se rotira oko svoje ose)
  {
    ceka se na dogadjaj;
    if (dogadjaj je "paket od korisnika") && (i<J)
  {                                      /* Ako se okvir uklapa u prozor, salje se */
    uzimanje paketa od korisnika i smestanje u buffer [(w+i) % MAX];
    konstruisanje okvira sa frame.ack= last, frame.type=data i frame.number=(w+i) % MAX;
    send(frame);
    reset frametimer(frame.number);      /* Definisanje tajmera za ocekivani ACK ovog
                                            okvira */
    stop acktimer;                       /* Zaustavljanje ACK tajmera jer je ACK poslat
                                            sa podacima */
    i++;                                 /* Povecavanje velicine prozora za 1 */
    continue;                            /* Prelazak na krai vvhile petlje */
  }
  if (dogadjaj je "istekao acktiraer")
  {                                      /* Nikakvi okviri nisu poslati u medjuvremenu.
                                            Slanje specijalnog ACK okvira */
    Konstruisanje i slanje okvira sa frame.type=ack i frame.ack=last;
    continue;

  >
if (dogadjaj je "istekao frametimer")
  {                                      /* Nije primljen ACK u medjuvremenu;
                                            ponovno slanje svih okvira u prozoru */
    for (j=w; j is "izmedju" w and (w+i-1) % MAX; increment(j) )
    {
      Konstruisanje i slanje okvira podataka kao i ranije sa paketom iz buffer [J];
      reset frametimer(j);               /* Startovanje tajmera za ocekivani ACK ovog
                                            okvira */
    }
stop acktimer;                           /* Zaustavljanje ACK tajmera; ACK je poslat
continue;                                   sa podacima */
  }
  if (dogadjaj je "stigao osteceni okvir")
  {
     Konstruisanje i slanje okvira sa frame.type=nack i frame.ack=last i njegovo slanje;
     stop acktimer;                      /* Zaustavljanje ACK tajmera; ACK je poslat */
     continue;
```

SLIKA 8.T 2 *Go-back-n protokol*

```
  if (dogadjaj je "stize neosteceni okvir")
  {                                    /* Uklanjanje svih okvira "izmedju" w i
                                          frame.ack iz prozora */
     receive(frame);
     for (j=w; j is "izmedju" w and frame.; increment(j) )
  {
    i--;
    stop frametimer(j);                /* Zaustavljanje tajmera okvira; ACK je
                                          primljen */
  }
  w=(frame.ack+1) % MAX;
  if (frame.type==data) && (frame.number==((last+1) % MAX )
  {                                    /* Ako je okvir podataka primljen u nizu,
                                          prenosi se do "gospodara" */
                                       /* Ignorisu se svi okviri primljeni van
                                          redosleda */
    mcrement(last);
    izvlacenje paketa iz okvira i prenosenje do korisnika;
    if acktimer is not active then
      reset acktimer;                  /* Startovanje ACK tajmera za okvir koji je
                                          prihvacen */
    continue;
  }
  if  (frame.type==nak)
  {                                    /* ponovno slanje svih baferovanih paketa */
    for (j=w; j is "izmedju" w and (w+i-1) % MAX; increment(j) )
    {
       konstruisanje i slanje okvira podataka kao i ranije sa paketom iz buffer [j];
       reset frametimer(j);            /* Startovanje tajmera za ocekivani ACK za o\
    }                                     okvir */
    stop acktimer;                     /* Zaustavljanje ACK tajmera; ACK je
    continue;                             poslat sa podacima */
  }
  if (frame.type==data) && (frame.number!=(last+1) % MAX)
  {     /* Slanje NAK za ocekivani okvir */
    konstruisanje i slanje okvira sa frame.type=nak i frame.ack=last;
    stop acktimer;
     }                                 /* Zaustavljanje ACK tajmera; ACK je poslat i
                                          okviru */
  }                                    /* kraj dogadjaja "stigao neosteceni okvir" J
  }                                    /* kraj while petlje */
}                                      /* kraj go_back_N */
```

SLIKA 8.12 *Go-back-n protokol*

Kao i ranije, nismo ni pokušavali da navedemo sintaksno ispravan kod, niti smo brinuli o tome da li se kod može ispravno kompajlirati. Namera nam je bila da opišemo kako protokol funkcioniše, bez zalaženja u detalje specifične za programski jezik. Važno je zapamtiti da oba uredaja pokreću kopiju algoritma dok razmenjuju okvire. Naime, svaki uredaj reaguje na dogadaje koji mu odgovaraju slanjem i primanjem okvira. Pažljivo i polako pročitajte narednu diskusiju i algoritam; algoritam je složen.

Algoritam sadrži petlju koja se kontroliše uslovom koji treba da važi u dužem vremenskom periodu. Ako uslov postane netačan, dolazi do greške u protokolu. Kako algoritam izvršava petlju, on reaguje na dogadaje koji se dešavaju. Ako se u toku jednog prolaska kroz petlju desi više događaja, algoritam nasumično bira neki od tih dogadaja i reaguje na njega. Mi ne vodimo

računa o tome kako se taj lzbor vršl - to zavisi od sistema. Moguce je da ce reagovati na druge dogadaje kroz naredne prolaske kroz petlju.

Sa svakim prolaskom kroz petlju, uredaj čeka da se desi neki dogadaj. Sledi pet dogadaja i reakcija protokola na njih:

1. Korisnik je isporučio paket. Ako je veličina prozora (i) sa maksimalnom veličinom (AT), ništa se ne dešava i događaj čeka dok se veličina prozora ne smanji. Ako Je veličina prozora manja od N, protoko] kreira okvir sa podacima koji sadrže paket. Takode, definiše piggyback potvrdu poslednjeg poslatog okvira (`frame.ack = last`) i definiše broj okvira kao (`frame.number = (w + 1) % MAX`, gde je $MAX = 2^K$). Izraz `(w + 1) % MAX` takode definiše i bafer u kome je paket smešlen. Nakon baferovanja paketa i slanja okvira, veličina prozora se inkrementira za 1 (++) i resetuje se odgovarajući tajmer okvira. Takode, zaustavlja se ACK tajmer. ACK tajmer, kao što je prethodno objašnjeno, detektuje dugačke periode vremena u kojima se ne šalju nikakvi okviri. Čim se pošalje neki okvir, ACK tajmer se zaustavlja. Tajmer okvira treba da detektuje dugački period u kome naznačeni olcvir nije potvrđen. Resetovanjem tajmera podnjemo odbrojavanje."

2. ACK tajmer je istekao. Kada nisu poslati nikakvi okviri sa podacima, drugi uredaj ne prima nikakve piggyback potvrde. Da bi se drugi uređaj obavestio o tome šta je tekući uredaj primio, protokol šalje specifični okvir potvrde kada ACK tajmer istekne. Njegova namena je da potvrdi najskorije primljeni okvir (`frame.ack=last`).

3. Tajmer okvira je istekao. Ako protokol nije primio potvrdu već duže vremena, možda je nešto "krenulo naopako". Možda su potvrde izgubljene, ili su izgubljeni okviri u tekućem prozoai. Pošto protokol ne zna šta se desilo, on pretpostavlja najgori slučaj i ponovo šalje sve okvire izprozora ("izmedu"t bafer je izmedu wi `(w + i • 1) % MAX`). Takode, resetuje sve tajmere okvira kako bi se obezbedilo dovoljno vreme da poslati okviri stignu do svojih odredišta i da se vrate njihove potvrde, pre nego što se donese zaključak da se ponovo nešto desilo sa njima u toku prenosa. Na kraju, zaustavlja se ACK tajmer, jer je poslata i piggyback potvrda.

4. Stigao je oštečeni okvir. Oštećeni okvir se ignoriše. Ako je oštećeni okvir bio očekivani, protokol eventualno ignoriše sve naredne. Zato protokol mora da obavesti drugi uređaj da je došlo do problema, tako da može da pošalje sve svoje baferovane okvire.

' Način implementacije tajmera nije bitan za našu diskusiju. Može da postoji interni prekidni takl, ili protokol može jednostavno da kreira listu zapisa za svaki tajmer, sa vrcmenskim pečatom za svaki zapis, a zatim se lista periodično pioverava. Ove detalje ostavljarao onima koji žele da implementiraju protoko] radi vežbanja programiranja.

t Mi smo definisali "izmedu" w i `(w+i-1 )%MAX` po modulu MAX. Ako je $w <$ `(if+i-I)%MAX`, "izmedu" ima svoje konvencionalno značenje. Ako Je $w >$ *(w+i-l)%MAX,* "izmedu" uključuje brojeve od *w* do MAX - 1 i 0 do *(ui+i-1)*%MAX. Na primer, preipostavimo da je MAX =16. Ako Je *w* =3 i *(w+i-l) % 6* = 12, "izmedu" znači vrednosti od 3 do 12, zaključno. Ako je *w* - 12 i `( W - I ) % 6 = 3`, "izmedu" znači vrednosti 12, 13, 14, 15, 0, 1, 2 i 3.

Protokol ovo izvodi slanjem okvira tipa NAK. Uredaj je poslao ACK za poslednji ispravno primljeni okvir i zaustavlja ACK tajmer.

5. Stigao je neoštećeni okvir. Ovo je najsloženiji deo protokola. Prvo što protokol radi jeste prijem okvira. Zatim proverava da li je uz njega poslata i piggvback potvrda i uklanja sve okvire na koje se eventualna potvrda odnosi. To se postiže smanjivanjem veličine prozora za 1 za svaki okvir "izmedu" w i frame. ack. Takode, zaustavlja tajrnere okvira za potvrdene okvire. Nakon toga, redefmiše početak prozora (w) kako bi se locirao prvi okvir koji nije potvrđen: (frame.ack + 1) % MAX.

   Ako okvir sadrži podatke, protokol utvrduje da li je to očekivani okvir ((last + 1) % MAX). Zapamtite, promenljiva last predstavlja najskorije primljeni okvir. Dakle, broj iza njega je očekivani okvir. Ako je primljeni okvir očekivani, protokol izvlači paket i daje ga korisniku. Takođe, uvećava vrednost promenljive last, tako da se pamti novi okvir koji je najskorije primljen. Zatim se postavlja ACK tajmer, koji definiše vreme u okviru koga treba da stigne potvrda.

   Ako je okvir tipa NAK, protokol ponovo šalje sve okvire iz prozora, kao da je istekao tajmer svih okvira. Ako se u okviru nalaze podaci, ali to nije očekivani okvir, protokol ga ignoriše, ali šalje NAK okvir.

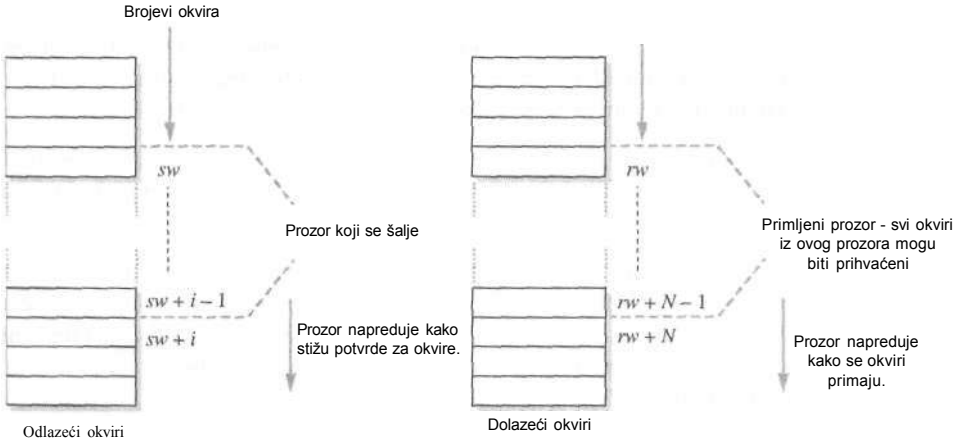
8.5 Selektivna retransmisija: Protokol klizajucih prozora

Protokol go-back-n dobro funkcioniše, posebno preko pouzdanih medijuma. Ukoliko u retkim situacijama dode do gubljenja, oštećenja, ili kašnjenja okvira, pretpostavka da su stigli ispravnim redosledom kojim su i poslati obično je tačna. U nekoliko slučajeva kada postoji problem, ponovnim slanjem svih nerešenih okvira gubi se malo vremena. Kako se pouzdanost smanjuje, dodatni troškovi ponovnog slanja kompletnog prozora zbog jednog oštećenog okvira, ili okvira koji stiže van redosleda postaju preterani. Ovde se nameće logično pitanje zašto se ne dopusti prijemnom uređaju da prihvati okvire van redosleda i da ih sortira kada svi stignu. Odgovor je omoeućen sledećim protokolom klizajućih prozora, nazvanim *selektivna retransmisija.*

Karakteristike

Protokol selektivne retransmisije je sličan go-back-n protokolu na sledeće načine:

- Formati okvira su slični i okviri se numerišu pomoću K-bitnog polja (videti sliku 8.9).
- Pošiljalac ima prozor defmisan maksimalnim brojem nerešenih okvira.
- Protokol selektivne retransmisije uvek šalje potvrdu zajedno sa novim podacima, ako je to moguće, i ne izdaje eksplicitne potvrde za sve okvire. Ako je okvir potvrđen, uredaj pošiljaoca smatra da su primljeni i svi okviri pre njega.
- Protokol koristi NAK okvire za sve oštećene okvire, ili okvire koji su stigli van redosleda.
- Koristi tajmere za slanje specijalnih potvrda za okvire u periodima slabog saobraćaja i za ponovno slanje okvira koji nisu potvrdeni duže vremena.



SLIKA 8.13 *Slanje i primanje prozora za protokol selektivne retransmisije*

Ovde prestaju sličnosti. Verovatno najuočljivija razlika je to što protokol selektivne retransmisije definiše dva prozora, po jedan i za predajnu i za prijemnu stranu protokola (slika 8.13). Tako, svaki uredaj koji koristi protokol selektivne retransmisije ima i predajni i prijemni prozor. Prozor na strani pošiljaoca je isti kao kod go-back-n protokola. On definiše koji okviri mogu da budu nerešeni.

Prijemni prozor definiše koji okviri mogu da se prime. Kao i kod prozora pošiljaoca, okviri u prijemnom prozoru su numerisani sukcesivno (mod 2^K, gde je K = broj bitova koji se koriste za brojeve okvira). Dakle, prijemni uredaj ne mora obavezno da prima okvire istim redosledom. Okvir koji stiže van redosleda može da se primi sve dok pripada istom pozoru. Medutim, setićete se da je deo odgovornosti protokola i isporuka paketa do korisnika sa ispravnim redosledom. Zato protokol mora da baferuje svaki okvir u prozoru. Okviri koji stižu van redosleda se baferuju sve dok ne stignu njihovi prethodnici. Nakon toga, protokol može da ih isporuči u tačnom redosledu.

Sledi lista daljih razlika izmedu selektivne retransmisije i go-back-n protokola, zajedno sa reagovanjem protokola sa selektivnom retransmisijom.

- Ako se pristigli okvir rtalazi u prijemnom prozoru, on se baferuje. Medutim, ne daje se korisniku sve dok ne stignu svi njegovi prethodnici (u granicama prozora). Tako, svaki put kada se okvir baferuje, protokol proverava slotove prozora pre dolaska novog okvira. Ako sadrže pakete, protokol ih isporučuje korisniku i pomera prozor unapred.
- Svaki put kada stigne okvir van redosleda, protokol šalje NAK za okvir koji je očekivao. Razlog za to je činjenica da okvir van redosleda signalizira da se nešto desilo sa očekivanim okvirom. NAK obaveštava pošiljaoca o mogućem gubitku. Zapamtite da, sve dok se primljeni okvir nalazi u prozoru, on može da bude prihvaćen.

- [/ Ako istekne vreme za potvrdu okvira, ponovo se šalju samo okviri čiji su tajmeri istekli. Kod go-back-n protokola inicira se slanje svih nerešenih okvira. Kod selektivne retransmisije prijemni uredaj je možda primio druge okvire i, osim ako nije isteklo njihovo vreme za potvrdu, nema potrebe za njihovim ponovnim slanjem.

- "" Ako protokol primi NAK, ponovo šalje naznačeni okvir. Go-back-n ponovo šalje sve nerešene okvire. Razlog za slanje samo jednog okvira je isti kao i slučaju isteka tajmera za okvir.

J

- Piggyback potvrda ne mora da se odnosi na najskorije primljeni okvir. Umesto toga, može da potvrdi okvir koji je primljen pre onog koji se nalazi na početku prijemnog prozora (tj. poslednjeg isporučenog korisniku). Ako bi se potvrdio najskorije primljeni okvir, uredaj pošiljaoca ne bi mogao da zaključi da su primljeni i prethodni okviri. Zapamtite, najskoriji okvir može da stigne i van redosleda. Efektivno, to bi uvelo potrebu izdavanja potvrda za sve okvire. Potvrđivanjem okvira koji je poslednji isporučen korisniku omogućava se uređaju pošiljaoca da zaključi da su prethodni okviri isporučeni i, samim tim, primljeni. Ponovo, rezultat je manji broj potvrda.

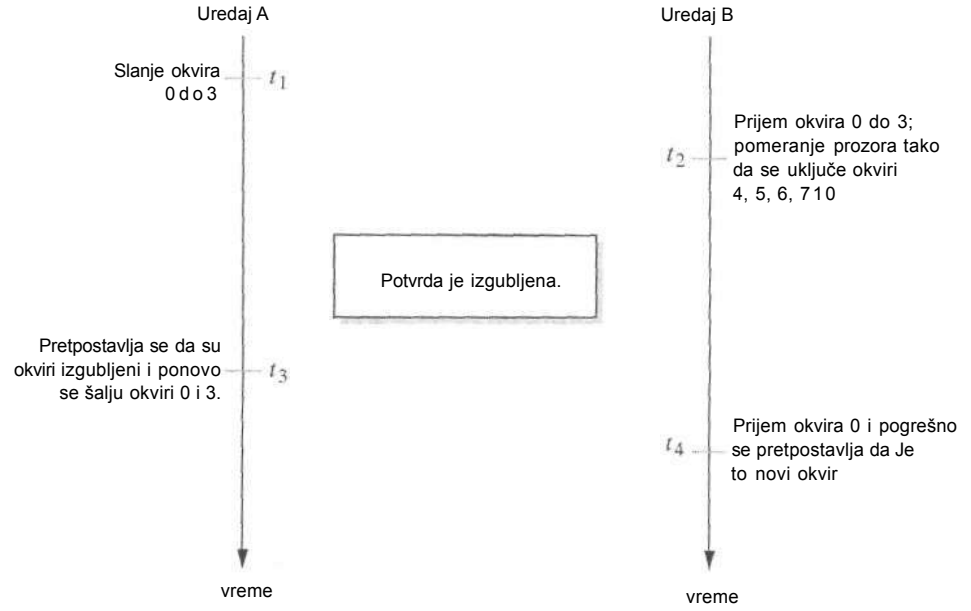
Kod go-back-n algoritma smo videli da su postojala ograničenja u veličini prozora. Specifično, veličina prozora je morala da bude strogo manja od 2^K, ili je moglo da dode do greške u protokolu zbog odredenih dogadaja. Ograničenja mogu da postoje i kod protokola selektivne retransmisije. Pretpostavimo da je maksimalna veličina prozora na predajnoj i prijemnoj strani jednaka. U tom slučaju, za oba važi ograničenje da moraju da budu manji, ili jednaki polovini od 2^.2*-[1]).

Da bismo videli šta bi se desilo u suprotnom, razmotrimo par primera. U oba primera ćemo koristiti JC = 3, tako da je $2^K = 8$. U prvom primeai pretpostavljamo da uredaj pošiljaoca ispunjava ograničenje i da ima maksimalnu veličinu prozora 4. Ali, razmotrimo šta se dešava ako je prozor na prijemnoj strani veći, recimo 5 (slika 8.14).

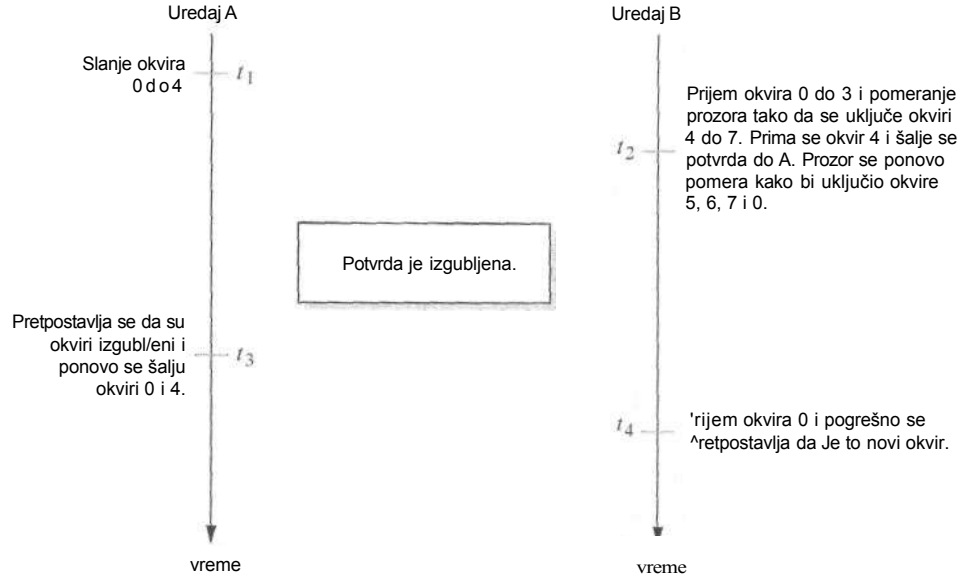
U trenutku tl uredaj Ašalje maksimalni broj okvira, od O do 3. Pošto uređaj B ima prozorveličine 5, može da prihvati sve okvire numerisane od O do 4, zaključno. U trenutku t_2 B prima okvire 0 do 3. Pošto se nalaze u prozoru, prihvataju se i prenose do korisnika. Nakon toga, B pomera prozor, tako da uključuje okvire 4, 5, 6, 7 i 0.

U međuvremenu, potvrda koju B šalje je izgubljena. Eventualno, A više ne može da čeka i pretpOvStavlja da je nešto "krenulo naopako". U skladu sa protokolom, A ponovo šalje okvire 0 do 3 (trenutak t_3). Pošto se okvir 0 nalazi u prijemnom prozoru, *B* ga prihvata (trenutak t_4), ne shvatajući da je to duplikat prethodnog okvira 0. Dolazi do greške u protokolu.

Sličan problem može da se desi ako veličina prijemnog prozora ispunjava ograničenje, ali ne i veličina prozora na stani pošiljaoca. Na primer, pretpostavimo da je ovoga puta prozor kod A veličine 5, a da je veličina prozora kod B 4 (slika 8.15). U trenutku I_1 A šalje okvire 0 do 4. Pošto je veličina prozora kod B 4, on može da prihvati samo okvire 0 do 3.



SLIKA 8.14 *Greška u protokolu: Veličina prijemnog prozora je veća od* $2^{K}-1$



SLIKA 8.15 *Greska u protokolu: Veličina prozora na strani pošiljaoca je veća od* 2^{K}"

Ali, pretpostavimo da okvir 4 kasni. U meduvremenu stižu okviri 0 do 3 i oni se prihvataju (trenutak t_2). B pomera svoj prozor kako bi uključio okvire 4 do 7.

Kada okvir 4 eventualno stigne, nalazi se unutar novog prozora i on se prihvata. Prozor se ponovo pomera i sada uključuje okvire 5, 6, 7 i 0. U ovoj tački B šalje nešto do A sa uključenom potvrdom. Ponovo se dešava isto kao kod prethodne potvrde. A se ponovo "umara", čekajući i ponovo šalje okvire 0 do 4 (trenutak t_3). Okviri se konačno probijaju (nema gremlina) i, pošto se okvir 0 nalazi unutar prijemnog prozora, prihvata se (trenutak ti). B ponovo ne prepoznaje da je reč o duplikatu prethodnog okvira 0 i dolazi do greške protokola.

Svaki od ovih problema može da se ispravi izjednačavanjem veličine oba prozora na 4. LI stvari, ovim problemi mogu da se eliminišu i biranjem prozora veličine 5 i 3, umesto 5 i 4 (ili 3 i 5, umesto 4 i 5). Problem se javlja kada se prijemni prozor pomera do tačke u kojoj uključuje nove okvire sa brojevima koji već postoje u prozoru pošiljaoca. Ovo može da se desi kada je suma veličina prozora veća od 2^K (ukupan broj različito numerisanih okvira). Redukovanjem veličina prozora to se neće desiti i tako smo eliminisali problem. Obično se koriste prozori iste veličine (2^{K1}).

Algoritam

Na slici 8.16 prikazan je delimično C-kodirani program za protokol selektivne retransmisije. Dizajniran je slično go-back-n protokolu po tome što neprestano izvršava petlju, reagujući na događaje koji se dešavaju. Prozori na obe strane su iste veličine, $N = 2^{KA}$.

Algoritam ima nekoliko dodamih promenljivih kojih nema u algoritmu za go-back-n. Osim bafera kod pošiljaoca (sbuf f er), postoji bafer i na prijemnoj strani (rbuf f er). Pošto su oba prozora veličine N_I oba bafera su definisana tako da mogu da prihvate N elemenata. Tako se stvara još jedna razlika u odnosu na go-back-n algoritam. Kod go-back-n algoritma paketi se smeštaju u slotove bafera sa brojevima okvira kao subscriptima. Ovde postoji dva puta više okvira nego što bafer ima slotova. Da bi se izbegao preterani broj bafera, subscript bafera je jednak broju okvira mod N.

Sledeća promenljiva koje nije bilo u ranije algoritmu jeste statusni niz. Pošto se pristigli okviri baferuju nasumičnim redosledom, koristimo statusni niz za utvrđivanje da li je slot bafera prazan. Vrednost `status [i] = 1` znači da bafer pod brojem i sadrži paket. Vrednost 0 znači da je prazan.

Kako algoritam prolazi kroz petlju, tako reaguje na događaje koji se dešavaju. Ovde navodimo listu dogadaja i reakcija na njih. Zbog sličnosti sa go-back-n protokolom, ovde nećemo detaljno diskutovati sve korake. Koncentrisaćemo se samo na one delove koji se razlikuju u odnosu na go-back-n.

1. Korisnik ima isporučeni paket. Protokol reaguje u većoj meri isto kao i go-back-n protokol. Ako je veličina prozora na strani pošiljaoca postavljena na maksimalnu Vrednost, ništa se ne dešava. U suprotnom, paket se baferuje, kreira se okvir, a zatim se okvir šalje. Takode, uz paket korisnik uključuje i potvrdu za okvir koji je poslat pre onog na početku prijemnog prozora (`frame.ack = prior(rw)`). Makro pod nazivom `priorvrši` oduzimanje 1 po modulu 2^K. Za razliku od makroa increment, ne menja promenljivu koja mu je preneta.

```
#define MAX=2^K;                    /* K = broj bitova u polju frame.number */
#define N=MAX/2;                    /* N je maksimalna velicina prozora posiljaoca,
                                       stvama velicina prijemnog prozora i broj bafera */
#define increment(x) x=(x+1) % MAX; /* Inkrementiranje x po modulu N */
#define prior (x) (x==0 ? MAX-I : x-1)/* Vraca celi broj pre x mod N */
void selective_repeat
{
  int frame_no=0;                   /* Odrzava brojeve okvira za odlazece okvire */
  int sw=0;                         /* Prva pozicija u prozoru posiljaoca */
  int rw=0;                         /* Prva pozicija u prozoru primaoca */
  int i=0;                          /* Tekuca pozicija u prozoru posiljaoca */
  packettype sbuffer[N];            /* Baferi za pakete kod posiljaoca */
  packettype rbuffer [N];           /* Baferi za pakete kod primaoca */
  int status[N];                    /* Status okvira u prijemnom prozoru. 1
                                       znaci da je stigao; 0 znaci da nije */
  while (Pakao nije zamrznut)
    {
      ceka se na dogadjaj;
      if (dogadjaj je "paket od korisnika") && (i<N)
      {                             /* Ako se okvir uklapa u prozor, salje se *;
          uzimanje paketa od korisnika i smestanje u sbuffer[(sw+i) % N];
          konstruisanje okvira sa frame.ack= prior(rw), frame.type=data i
            frame.number=framejio;
          send(frame);
          increment(frame_no);      /* Definisanje broja sledeceg odlazeceg
          okvira */
          reset frametimer(frame.number);   /* Definisanje tajmera za ocekivani ACK
                                               ovog okvira */
          stop acktimer;            /* Zaustavljanje ACK tajmera; ACK je poslat
                                       sa podacima */
          i++;                      /* Povecavanje velicine prozora posiljaoca za
                                       1 */
          continue;                 /* Prelazak na kraj while petlje */
    }
     if (dogadjaj je "istekao acktimer")
    {                               /* Nikakvi okviri nisu poslati u medjuvremenu.
                                       Slanje specijalnog ACK okvira */
         Konstruisanje i slanje okvira sa frame.type=ack i frame.ack=prior(rw);
         continue;
    }
    if (dogadjaj je "istekao frametimer")
    {                               /* Nije primljen ACK u medjuvremenu; ponovno
                                       slanje okvira */
        fn = broj okvira koji odgovara tajmeru;
        Konstruisanje i slanje okvira podataka kao i ranije sa paketom iz sbuffer fn % N ;
        reset frametimer(fn);       /* Startovanje tajmera za ocekivani ACK ovog
         stop acktimer;                okvlra */
         continue;                  /* Zaustavljanje ACK tajmera; ACK je poslat */
    }
    if (dogadjaj je "stigao osteceni okvir")
    {
       Konstruisanje i slanje okvira sa frame.type=nack i frame.ack=prior(rw);
       njegovo slanje;
       stop acktimer;               /* Zaustavljanje ACK tajmera; ACK je poslat */
       continue;
```

```
}
if (dogadjaj je "stize neosteceni okvir")
{                                   /* Uklanjanje svih okvira "izmedju" sw i
                                       frame.ack iz prozora posiljaoca */
    receive(frame);
    for (j=w; j is "izmedju" sw and frame.ack; increment(j) )
    {
        i--;
        stop frametimer(j);         /* Zaustavljanje tajmera okvira; ACK je primljen */
    }
    Sw=(frame.ack+1) % MAX;
    if (frame.type==data) && (frame.number /= rw)
{
    konstruisanje okvira sa frame.type=nak i frame.ack=prior(rw) i njegovo slanje;
    stop acktimer;                  /* Zaustavljanje ACK tajmera; ACK je poslat */
}
if (frame.type==data) && (frame.number is in prijemni prozor) &&
    (status[frame.number % N]==0)
{                                   /* Ako je okvir u prozoru i jos nije stigao,
                                       baferuje se */
    izvlacenje paketa iz okvira i postavljanje u rbuffer frame.number % N ;
    status[frame.number % N]=1;
    for (; status[rw % N)==1; increment(rw) )
    {                               /* Uzimanje primljenih paketa smestenih u
                                       sukcesivnim slotovima prozora za korisnika */
       izvlacenje paketa iz rbuffer[rw % N] i davanje korisniku;
       status[rw % N]=0
    }
    reset acktimer;                 /* Startovanje ACK tajmera za okvire koji su
    }                                  prihvaceni*/
    if (frame.type==nak) && (framenum=(frame.ack+1) % MAX is in prozor
    posiljaoca)
    {                               /* Ponovno slanje okvira koga prijemna
                                       stanica ocekuje da primi */
      konstruisanje i slanje okvira sa paketom iz sbuffer[framenum % N];
      reset frametimer(framenum % N);  /* Startovanje tajmera za ocekivani ACK
      stop acktimer;                      za ovaj okvir */
      continue;
    }                               /* Zaustavljanje ACK tajmera; ACK je poslat
                                       sa podacima */
  }
}
}
```

SLIKA 8.16 *Protokol selektivne retransmisije*

2. ACK tajmer je istekao. Protokol šalje ACK okvir potvrde za okvir pre onog koji se nalazi na početku prijemnog prozora.

3. Istekao je tajmer okvira. Umesto da se ponovo šalju svi nerešeni okviri, protokol šalje samo okvir koji odgovara tajmeru koji je istekao. Način utvrđivanja okvira zavisi od toga kako su tajmeri implementirani. Kao i ranije, mogli su da postoje prekidni mehanizam koji identifikuje brojeve okvira, ili neka lista sa vrednostima perioda i brojevima okvira.

4. Stiže oštećeni okvir. Protokol šalje NAK okvir sa potvrdom za okvir koji je prethodio okviai smeštenom na početak prijemnog prozora.

5. Stiže neoštećeni okvir. Nakon prijema okvira, protokol iz prozora uklanja sve okvire koji su potvrdeni. Ako okvir sadrži podatke, protokol proverava da li stigli pravilnim redosledom. Drugim rečima, da li je broj okvira jednak broju koji odgovara početku prijemnog prozora? Ako nije, nazad se šalje NAK okvir.

Zatim, protokol proverava još dva uslova: da li je primljeni okvir u prozom i da li njegov paket još uvek nije baferovan. Paket je mogao da bude baferovan ako je okvir stigao ranije, ali je njegova potvrda (ACK) kasnila do uredaja pošiljaoca. U tom slučaju, za okvir je isteklo vreme i protokol bi inicirao njegovo ponovno slanje. Proverom vrednosti status[frame.number % N] izbegava se dodatni posao izvlačenja paketa koji je većbaferovan.

Ako su oba uslova ispunjena, paket se izvlači i smešta u bafer. Zatim, protokol utvrduje da li može da pomeri napred prijemni prozor i da isporuči pakete do korisnika. U tu svrhu proverava sukcesivne pozicije u nizu `status`. Zaustavlja se kada pronade prvi prazan slot u prozoru.

Konačno, ako je okvir tipa NAK, protokol proučava vrednost u `frame.ack`. Kada se pošalje NAK okvir, u polju frame.ack nalazi se broj okvira koji prethodi onome koji se nalazi na početku prijemnog prozora. Ovo znači da se nešto desilo i da prijemni protokol nije dobio okvir koji je očekivao ((frame.ack+1) % MAX). Ako je ovaj okvir i dalje u prozoru pošiljaoca, protokol mora da ga pošalje. Možete li da konstruišete scenario u kome se ovaj okvir ne nalazi u prozom pošiljaoca?

8.6 Efikasnost protokola klizajućih prozora

U odeljku 8.3 analizirani su neograničeni i stop-and-wait protokoli i pokazano je da na protokol može da utiče stvarna količina podataka koja se prenese u jedinici vremena (efektivna brzina prenosa podataka). Videli smo da efektivna brzina prenosa podataka zavisi i od izvorne bitske brzine, rastojanja između uredaja, veličine okvira i dmgih faktora. Kompletna analiza protokola klizajućih prozora je mnogo teža, jer i drugi faktori utiču na efektivnu brzinu prenosa podataka. Medu te faktore ubrajaju se učestalost gubljenja, ili oštećenja okvira, vrednosti tajmera koje se koriste za utvrdivanje potrebe za slanjem specijalnih ACK okvira i broj okvira sa podacima koji "putuje" u suprotnom smeru sa uključenom potvrdom.

Analizu protokola klizajućih prozora dajemo sa određenim pretpostavkama. Pretpostavićemo da nema izgubljenih, ili oštećenih okvira. Takode, pretpostavljamo konzistentan saobraćaj u oba smera kako bi se maksimalno koristile piggyback potvrde. Zahvaljujući toj pretpostavci, možemo da zanemarimo ACK tajmere, jer neće biti korišćeni. Ako ste zainteresovani za kompletnu analizu protokola klizajućih prozora, pogledajte reference [Ta96] i [Wa91].

Ako su sve "stvari" iste, efektivna brzina prenosa podataka za klizajući prozor treba da se nađe negde izmedu brzine za neograničeni i stop-and-wait protokol. Ali, koju efektivnu brzinu možemo da očekujemo od protokola klizajućih prozora? Kako na nju utiče veličina prozora?

Sećate se sledećih definicija iz odeljka 8.3 i vrednosti koje su korišćene u primeru:

R = bitska brzina (10 Mbps, ili 10 bitova/nsec)

S = brzina signala (200 metara/|isec)

D = rastojanje izmedu pošiljaoca i primaoca (200 metara)

T = vreme potrebno za kreiranje jednog okvira (1 nsec)

F = broj bitova u okviru (200)

N = broj bitova podataka u okviru (160)

A = broj bitova u potvrdi (40)

Dodajmo još jednu promenljivu u ovu listu:

W = veličina prozora (4 okvira)

Za početak, razmotrićemo dva moguća slučaja kod protokola klizajućih prozora. Prvi je kada prozor pošiljaoca nikada ne dostiže svoju maksimalnu veličinu. To se dešava kada prva potvrda stiže pre nego što se pošalju svi okviri iz prozora. Ako se to desi i uz pretpostavku da nema kašnjenja na drugom kraju, pošiljalac nikada ne mora da čeka na potvrdu. Drugim rečima, stari okviri se uklanjaju iz prozora istom brzinom kojom se u njega dodaju novi. Zato se protokol pošiljaoca ponaša kao neograničeni protokol,

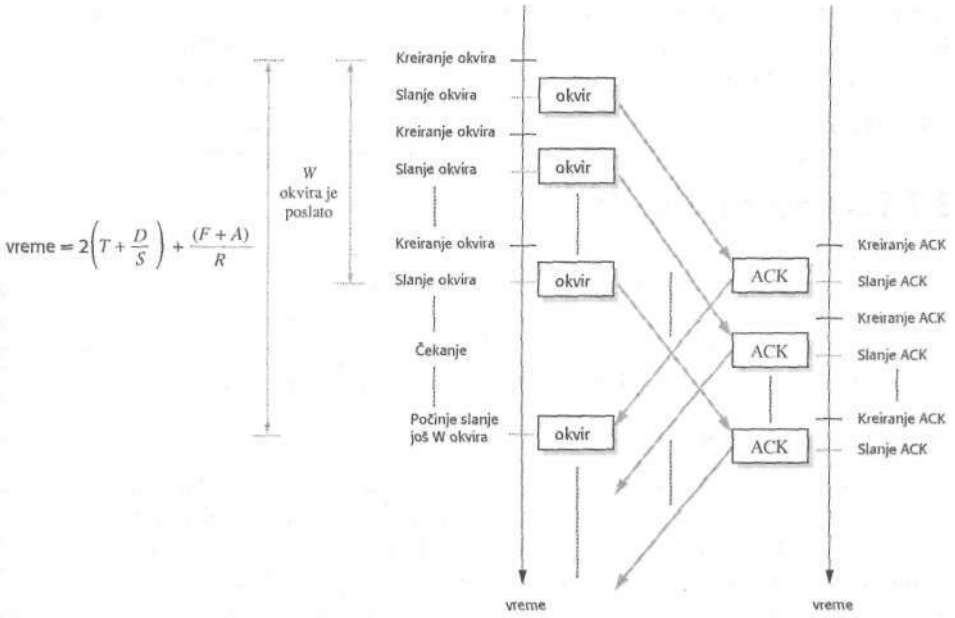
U drugom slučaju, kada su svi *W* okviri poslati i prva potvrda još uvek nije stigla (slika 8.17), protokol mora da čeka na njega. Kada potvrda stigne, protokol može da pošalje novi okvir. Ako potvrde stižu istom brzinom kojom se okviri sa podacima šalju, poslaće se još *W* okvira pre nego što protokol bude morao ponovo da čeka. Drugim rečima, protokol šalje W okvira, čeka na prvu potvrdu, šalje još *W* okvira, čeka na potvrdu i tako redom. Sada protokol podseća na stop-and-wait protokol. Međutim, umesto da šalje i čeka individualne okvire, on šalje prozor ispunjen okvirima.

Matematički, ova dva slučaja mogu da se razlikuju poredenjem vremena za slanje W okvira sa vremenom potrebnim da se pošalje jedan okvir i da se primi potvrda. Na osnovu jednačine 8.1, vreme potrebno za kreiranje i slanje jednog okvira je T + *F/R.* Tako je vreme potrebno za slanje W okvira W x (T + F/R). Na osnovu jednačine 8.2, vreme potrebno za slanje jednog okvira i prijem potvrde (uz pretpostavku da se potvrda odmah vraća) je *2(T + D/S) + (F + A)/R = 2(T + D/S) + 2F/R = 2(T + D/S + F/R).* U ovoj jednačini smo zamenili *F* za A_1 jer se potvrde šalju u okvirima sa podacima (veličine F), umesto da se šalju posebni ACK okviri (veličine A).

Tako imamo

1 slučaj (neograničeni protokol): $W\left(T+\frac{F}{R}\right)>2\left(+\frac{D}{S}+\frac{F}{R}\right)$

2 slučaj (prozoru orijentisani stop-and-wait): $W\left(T+\frac{F}{R}\right)<2\left(T+\frac{D}{S}+\frac{F}{R}\right)$



SLIKA 8.17 *Slanje svih okvira iz prozora i čekanje*

U prvom slučaju, na osnovu jednačine 8.3, imamo

$$\text{efektivna brzina prenosa podataka} = \frac{N}{T + \frac{F}{R}} \quad \text{(neograničena verzija)}$$

Pošto naše proste vrednosti ispunjavaju uslov u slučaju 1, efektivna brzina prenosa podataka iznosi

$$\frac{160\text{bits}}{1\mu\text{sec} + \frac{200\text{bits}}{10\text{bits}/\mu\text{sec}}} \approx 7.6 \text{ bits/msec} = 7.6 \text{ Mbps}$$

Efektivna brzina prenosa podataka za slučaj 2 je izvedena na osnovu formule 8.4. Ova jednačina je izvedena pod pretpostavkom da je poslat samo jedan okvir. Pošto mi sada šaljemo *W* okvira za isto vreme, menjamo *N* sa *W* x *N*. Uz zamenu *A* sa *F,* dobijamo

$$\text{efektivna brzina prenosa podataka} = \frac{W \times N}{2\left(T + \frac{D}{S}\right) + \frac{2F}{R}} \quad \text{(prozoru orijentisani stop-and-wait protokol)}$$

S obzirom na to da naše vrednosti ne ispunjavaju uslov u slučaju 2, njihovom zamenom u ovoj jednačini ne bismo dobili smisaoni rezultat. Ako povecamo rastojanje (D) sa 200 metara na 5.000 metara, vrednosti će ispuniti uslov slučaja 2. Sa tim vrednosima dobijamo

$$\text{efektivna brzina prenosa podataka} = \frac{4 \times 160\text{bits}}{2\left(1\mu\,\text{sec} + \frac{5000\ \text{meters}}{200\ \text{meters}/\mu\,\text{sec}}\right) + \frac{2 \times 200\text{bits}}{10\text{bits}/\mu\text{sec}}}$$

8.7 Tačnost protokola

U prethodnom odeljku smo predstavili neke protokole i uslove pod kojima mogu ispravno da funkcionišu. Primećujete da kažemo "mogu da funkcionišu". Ovo je svakako neophodno, jer još uvek nismo dokazali da stvarno funkcionišu. Obezbeđivanje formalnog dokaza, ili verifikacije funkcionisanja protokola je veoma teško i zato takve dokaze ostavljamo nekim naprednijim kursevima za dizajn protokola. U ovom odeljku uvodimo dve osnovne alatke za verifikaciju.

Konačni automati

Veći deo onoga što smo smatrali kontinuelnim, ili analognim u stvari je kolekcija zasebnih, ili diskretnih dogadaja. Kao najčešće primenjivani primer mogu da posluže pokretne slike. Dok jedemo kokice, pijemo sokove, protežemo se i zevamo, akcija koju gledamo na ekranu deluje kao tečno, ili kontinuelno kretanje. U stvarnosti, reč je o brzom prikazivanju slika koje se prikazuju kroz projektor. To omogućava da vidimo film na novi način, kao kolekciju individualnih slika. Ovo nije najpoželjniji način za gledanje nekih klasika, ali to je upravo način na koji ljudi koji rade na filmu, kao što su tehničari za specijalne efekte, moraju da vide taj film. Oni vide sekvencu slika koje će biti podeljene, isečene, ili izmenjene kako bi se postigao odgovarajući efekat.

I kompjuterski algoritmi mogu da se posmatraju kao sekvence "slika". Kompjuteri koji ih izvršavaju predstavljaju digitalne uredaje. Njihove akcije se kontrolišu i sinhronizuju internim taktom i vođeni su programima koje pokreću. Svaki impuls takta defmiše novi set internih vrednosti i za kraći period (dužina impulsa takta) ništa se ne menja. U suštini, cela arhitektura je "zamrznuta" u vremenu i kolekcija internih vrednosti definiše sliku **stanja automata.** Te vrednosti se menjaju sa sledećim impulsom takta, defmišući novo stanje automata. Ovaj proces se neprestano izvršava, definišući sekvencu stanja automata.

Slično tome, algoritam može da se posmatra kao sekvenca stanja. Svako stanje je definisano delimično vrednostima programskih prornenljivih u jednom trenutku. Teorijski, možemo da kategorizujemo (izlistamo) sva moguća stanja i dogadaje koji mogu da izazovu promenu sa jednog stanja u drugo. Termin konačni automat **(finite state machine),** koji se ponekad naziva i model konačnog stanja, odgovara ovoj kategorizaciji. Dogadaj koji izaziva promenu stanja naziva se prelazno stanje.

Posmatranje algoritma na diskretni način omogućava predstavljanje algoritma preko grafa koji se naziva dijagram prelaza stanja **(STD - state transition diagram).** Sećate se da se usmereni graf sastoji od niza temenih tačaka (verteksa) i ivica. Svaki verteks predstavlja stanje i obično je vizuelno predstavljen tačkom, ili kmgom.

Svaka ivica je uredeni par verteksa i obično se vizuelno predstavlja strelicom od prvog verteksa do drugog. Preko teorije grafova možerao da analiziramo dijagram prelaza stanja i možemo da izvedemo zaključke o dostupnosti odredenih stanja, ili mogućih sekvenci dogadaja (prelaza).

Na slici 8.18 prikazan je dijagram prelaza stanja. Ima šest različitih stanja, a strelice pokazuju moguće prelaze. Na primer, ako je sistem trenutno u stanju Sl, moguća su tri dogadaja: jedan koji izaziva prelazak u stanje S_2, a druga dva izazivaju prelaske u stanje S_4, ili S_5.

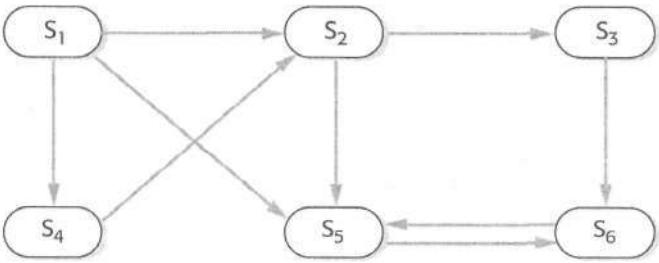
Analiziranjem grafa možemo da donesemo zaključke o sistemu koji je predstavljen grafom. Na primer, primećujete da nema ivica koje ukazuju na stanje S_1. To znači da nema prelaza u stanje S_1. Ako ovaj graf predstavlja algoritam dizajniran za reagovanje na dogadaje, ovo opažanje može da ukaže na nedostatak u logici algoritma. Naime, algoritam ne reaguje ni na jedan dogadaj koji bi ga odveo u stanje S_1. Ako je ovo u suprotnosti sa onim što znamo o sistemu, detektujemo nedostatak.

Ovaj graf pokazuje još jedan potencijalni problem. Pretpostavimo da se desi dogadaj koji izaziva prelazak u stanje S_5. On može da reaguje samo na dogadaje koji izazivaju prelazak u stanje S_6. Kada se nade tu, može samo da se vrati u stanje S_5. Drugim rečima, kada ovaj model jednom dospe u stanje S_5 ili S_6, ostaće u jednom od ta dva stanja zauvek. To može da odgovara beskonačnoj petlji, ili samrtnom zagrljaju (deadlock), čekanju na dogadaj koji se nikada neće desiti. Kao i u prethodnom slučaju, ovo najverovatnije predstavlja propust u algoritmu.

SID za pojednostavljeni go-back-n protokol

Kako se ovaj dijagram može primeniti na konkretni protokol? Najpre ćemo razmotriti go-back-n protokol kod koga je veličina prozora na strani pošiljaoca 1 i polje Frame Number je 1-bitno. Pretpostavljamo da nema pauza, niti grešaka u toku prenosa, da svi podaci idu samo u jednom smeru (od pošiljaoca do primaoca) i da primalac šalje potvrdu za svaki primljeni okvir. U suštini, to je stop-and-wait protokol sa brojevima okvira. Dešavaju se sleded dogadaji:

1. Slanje okvira O
2. Prijem okvira 0; slanje ACK 0



SLIKA 8.18 *OpUi dijagram prelaza stanja*

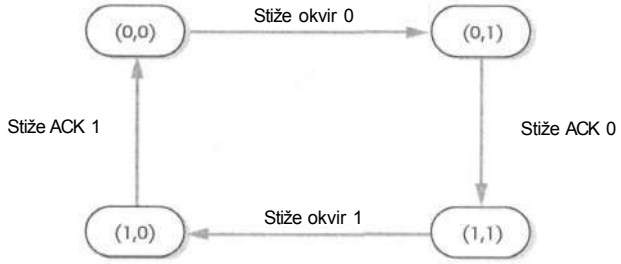
3. Prijem ACK O; slanje okvira 1
4. Prijem okvira 1; slanje ACK 1
5. Prijem ACK 1; slanje okvira 0

Kod ovog protokola možemo da razlikujemo četiri stanja. Označena su uređenim parovima (x, *y)* na slici 8.19. Vrednost *x* je ili O, ili 1, u zavisnosti od ACK broja na koji pošiljalac čeka. Slično tome, y je Oi 0, ili 1, u zavisnosti od broja okvira koga primalac očekuje. Stanje (0, 0) znači da je pošiljalac poslao okvir 0 i da oćekuje potvrdu. Takode, znači da primalac očekuje okvir 0.

Dolazak okvira 0 je događaj koji izaziva prelazak iz stanja (0, 0) u (0, 1). Primalac je primio okvir 0, poslao je potvrdu i sada očekuje okvir 1. Medutim, pošiljalac i dalje čeka na potvrdu za okvir 0. Kada potvrda stigne, pošiljalac je prihvata, šalje okvir 1 i ponovo čeka na sledeću potvrdu. To je stanje (1, 1), jer primalac i dalje očekuje okvir 1. Slanje i primanje okvira i potvrdivanje se nastavlja, tako da se stanja na slici 8.19 javljaju u smeru kretanja kazaljki na časovniku.

Pronicljiviji čitalac će možda postaviti pitanje zar ne postoji više mogućih stanja koja prate ovaj protokol. Na primer, postoji određeni period nakon što pošiljalac primi potvrdu i pre nego što pošalje naredni okvir. Zar ne bi trebalo da postoji stanje u kome pošiljalac čeka na korisnika koji mu obezbeduje pakete? Da! U stvari, mogli bismo da idemo u krajnost i da definišemo stanje koje odgovara izvršenju svakog koraka u sklopu algoritma. Ali, da li se to isplati?

Definisanje stanja je značajan dizajnerski problem. Idealno, mogli bismo da definišemo stanja koja predstavljaju značajne korake u evoluciji sistema i da ne vodimo računa o beznačajnim, ili trivijalnim razlikama. Ali, utvrdivanje šta je značajno često nije nimalo jednostavno i umnogome zavisi od toga šta se modeluje. Često postoje razni nivoi dorade kojima STD može da se izloži. Daćemo jedan primer koji pokazuje kako se vrše dorade i kako STD može da ukaže na nedostatke sistema. Naša namera je da ovde predstavimo samo uvodne koncepte, tako da se nećemo baviti detaljnim razradama STD-a. Ako ste zainteresovani za više detalja ili diskusiju na višem nivou, predlažemo reference [Ta96], [Wa91], [Li87] i [Ru89].



SLIKA 8.19 *STD za stop-and-wait protokol sa brojevima okvira*

Dijagram prelaza stanja za go-back-n protokol sa greškom

Razmotrimo prethodnu verziju go-back-n protokola. Ovoga puta pretpostavljamo da je veličina prozora 2, što, prema odeljku 8.4, može da dovede do greške. Da bi Vam bilo lakše, na slici 8.20 prikazujemo algoritam sa odgovarajućim ograničenjima (na primer, pošiljalac prima samo ACK, ili NAK, primalac prima samo podatke i brojevi okvira su naizmenično O i 1).

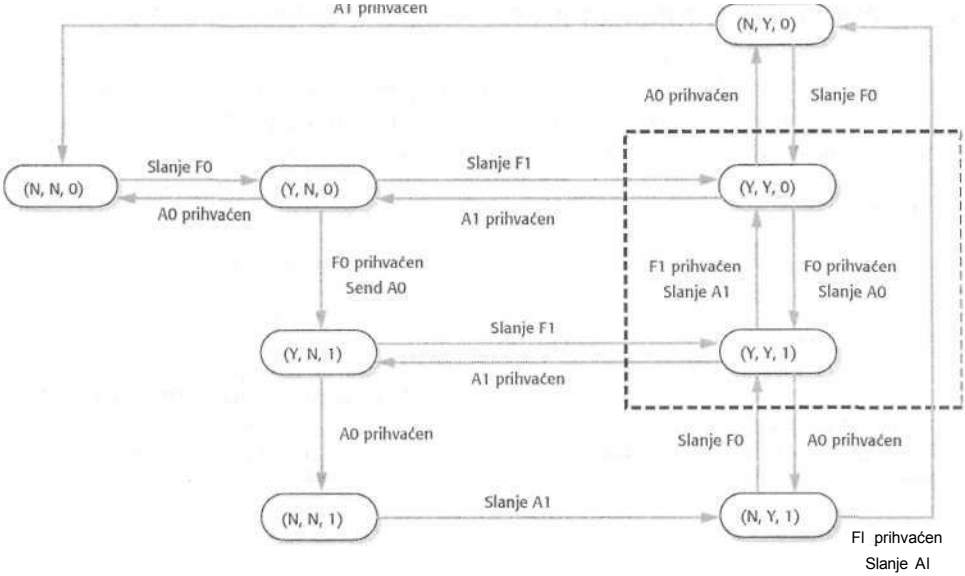
```
void send_data;
{
#define increment(x) x=(x==0 ? 1 : 0);
int w=0;
int i=0;
packettype buffer[2];
while postoje paketi za slanje
{
  cekanje na dogadjaj;
  if (dogadjaj is "paket od korisnika") &&
  (i<2)
  {
    uzimanje paketa od korisnika i
        smestanje u
        buffer[(w+i) % 2];
    konstruisanje i slanje okvira sa
        frame.number=(w+i) % 2;
    reset frametimer(frame.nimiber);
    i++;
    continue;
  }
  if (dogadjaj is "istekao frametimer")
  {
    ponovno slanje jednog ili oba okvira
        iz prozora;
    resetovanje jednog ili oba frametimera;
    continue;
  }
  if (dogadjaj Is "stize neosteceni okvir")
  {
    receive(frame);
    uklanjanje svih potvrdjenih okvira iz
        prozora;
    decrement i za broj uklonjenih okvira;
    stop frametimers za potvrdjene okvire;
    w=(frame.ack+1) % 2;
    if (frame.type=nak)
  {
        ponovno slanje okvira iz prozora;
        reset frametimers;
    }
  }
}
```

Kod posiljaoca

```
void receive_data;
{
#define increment(x) x=(x==0 ? 1 : 0);
int last=-1;
while postoje paketi za prijem
{
    cekanje na dogadjaj;
    if (dogadjaj is "stize osteceni okvir")
    {
        konstruisanje okvira sa
          frame.type=nak
        i frame.ack=last i slanje okvira;
        stop acktimer;
        continue;
    }
    if (dogadjaj is "stize neostecenl okvir")
    {
        receive(frame);
        if (frame.number != last)
        {
          increment(last);
          izvlacenje paketa iz okvira i
          prosledjivanje do "gospodara";
          if acktimer not aktivan then
            reset acktimer;
          continue;
    }
    if (frame.number==last)
    {
        konstruisanje i slanje okvira sa
          frame.type=nak i frame.ack=last;

        stop acktimer;

    }

    if (dogadjaj is "istekao acktimer")
    {
        konstruisanje i slanje okvira sa
          frame.type=ack i frame.ack=last;
        continue;
    }
}
```

Kod primaoca

SLIKA 8.20 *Go-back-n protokol za jednosmerni transfer podataka (veličina prozora = 2)*



SLIKA 8.21 *Prva aproksimacija STD-a za go-back-n protdkol (ueličina prozora = 2)*

Na slici 8.21 predstavljena je prva aproksimacija STD-a koji prikazuje neka stanja i prelaze stanja. U ovom slučaju kategorizujemo sva stanja na koja pošiljalac i primalac čekaju. Mi predstavljamo svako stanje kao uredenu trojku *(a, b,* c), definisanu na sledeći način:

- Ako pošiljalac čeka na ACK za okvir 0, onda je *a* = Y. U suprotnom, *a* = N.
- Ako pošiljalac čeka na ACK za okvir 1, onda je *b* = Y. U suprotnom, *b* - N.
- Ako primalac čeka na okvir 0, onda Je c = 0. U suprotnom *c* = 1.

Na primer, pretpostavimo da se model nalazi u stanju (N, N, 0). Pošiljalac ne očekuje ACK, a primalac čeka na okvir 0. Ako pošiljalac šalje okvir 0, model prelazi u stanje (Y, N, 0). Pošiljalac sada očekuje ACK za okvir 0. Dok se nalazi u ovom stanju, mogu se desiti dva druga dogadaja. Prvi je da pošiljalac šalje okvir 1 kada model prelazi u stanje (Y, Y, 0). Drugi dogadaj je kada okvir 0 stiže i biva prihvaćen, a primalac šalje ACK. U ovom slučaju primalac sada čeka na sledeći okvir istanje je (Y, N, 1).

Trebalo bi da izdvojite vremena da ispratite neke strelice i da razumete zašto se stanja menjaju onako kao što je prikazano. Kako to budete radili, otkrićete neke anomalijske dogadaje. Na primer, razmotrite sledeća dva načina za prelazak iz stanja (N, N, 0) u stanje (Y, Y, 0):

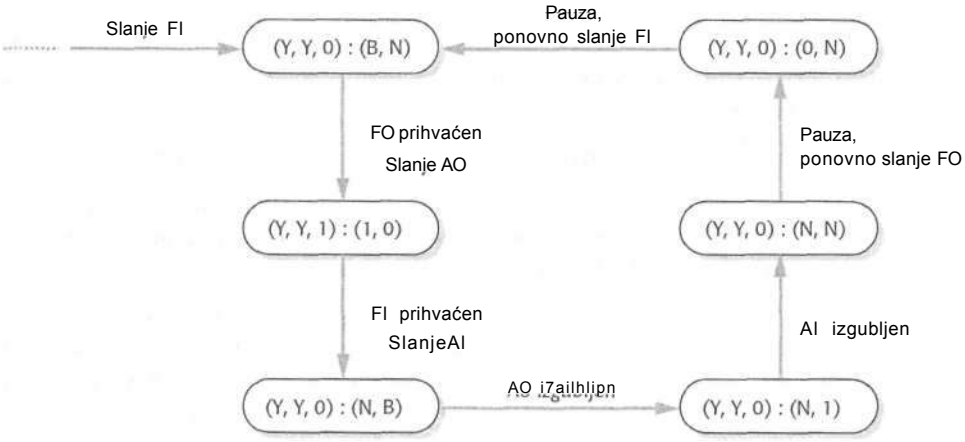
1. Slanje okvira FO i Fl
2. Slanje okvira FO; FO prihvaćen i AO poslato; slanje Fl; Fl prihvaćen, a Al poslato.

U prvom slučaju sledeći može da bude samo jedan događaj: FO stiže i biva prihvaćen. Drugi prikazani događaji (AO i Al prihvaćeni) ne mogu da se dese, jer još nije poslata ni jedna potvrda. FO ne može da stigne jer je već stigao (pretpostavićemo da okviri ne mogu da se kloniraju dok putuju, što bi rezultovalo armijom okvira). Poenta je da se na slici 8.21 ne pravi razlika između slučajeva 1 i 2; prikazani su slučajevi koji mogu da budu nemogući u zavisnosti od toga kako se došlo do stanja.

Problem je to što nismo redefinisali naše definicije stanja radi tačnog prikazivanja sistema. Jedno moguće rešenje je da se dorade defmicije stanja, tako da uključuju okvire koji su, u stvari, u tranzitu, tako da možemo da pravimo razliku izmedu dva slučaja (i drugih). Takode, kreiraće se dodatna stanja i prelazi stanja, tako da dijagram postane složeniji.

Na slici 8.22 prikazana je parcijalna dorada uokvirenog dela STD-a sa slike 8.21. Takođe, ova dorada pokazuje kako STD može da locira nedostatke u našem dizajnu. Dalje definišemo svako stanje naznačavanjem ne samo šta uredaji čekaju, već šta je, u stvari, u tranzitu. Ovo predstavljamo dodavanjem uredenog para *(x, y)* za svako stanje. Promenljiva x definiše okvire koji su, u stvari, u tranzitu (0 za okvir 0, 1 za okvir 1, B za oba i N za ni jedan). Slično tome, y defmiše koje se potvrde naalze u tranzitu. Na primer, stanje (Y, Y, 0): (B, N) znači da pošiljalac čeka na potvrdu za okvire 0 i 1 i da su ti okviri i dalje u tranzitu. Primalac čeka na okvir 0 i nema potvrda u tranzitu. Takode, uključili smo dodatne događaje koji izazivaju promenu stanja (postoje i drugi događaji koje nismo prikazali, ali i ovi su sasvim dovoljni za naše potrebe).

Zatim ćemo pokazati kako ovaj redefmisani model može da ukaže na problem. Sećate se sa kursa o strukturama podataka da putanja kroz graf predstavlja listu čvorova gde su svaka dva susedna čvora u listi povezana ivicom. Na STD-u putanja defmiše niz događaja. Graf na slici 8.22 prikazuje putanju (u stvari, ciklus) u kojoj se javljaju oba stanja (Y, Y, 0) i (Y, Y, 1).



SLIKA 8.22 *Parcijalna dorada STD-a sa slike 8.21*

Razmotrite šta se dešava ako se neprestano prati ciklus. Ovim se defmiše niz dogadaja koji izaziva primaoca da naizmenično očekuje i prima okvire O i 1. Medutim, ni jedan od tih dogadaja ne odgovara slanju novih okvira (samo ponovno slanje starih). Ovo znači da primalac neprestano prihvata nove okvire, iako nikakvi novi okviri nisu poslati. Naime, primalac prihvata stare okvire kao da su stari, kao što smo diskutovali u odeljku 8.4.

U opštem slučaju, STD-i mogu da se koriste za praćenje sekvenci dogadaja i medustanja. Ako putanja predstavlja promene koje ne bi trebalo da se jave kod odredenih dogadaja, postoji propust u modelu.

Model Petri net

Poput modela konačnog stanja, **Petri net** koristi graf za predstavljanje stanja i prelaza, ali način na koji to radi je drugačiji. Petri net se sastoji iz četiri dela:

1. **Mesta** Vizuelno se predstavljaju krugovima, mestima koja odgovaraju delu stanja. Ovo je jedna razlika u odnosu na model konačnog stanja. Svaki verteks STD-a predstavlja kompletno stanje; kod Petri neta može da postoji nekoliko mesta koja predstavljaju kompletno stanje. tlbrzo ćemo dati i primer.

2. **Prelazi** Vizuelno se predstavljaju kratkim horizontalnim, ili vertikalnim linijama i pokazuju prelazak izmedu mesta.

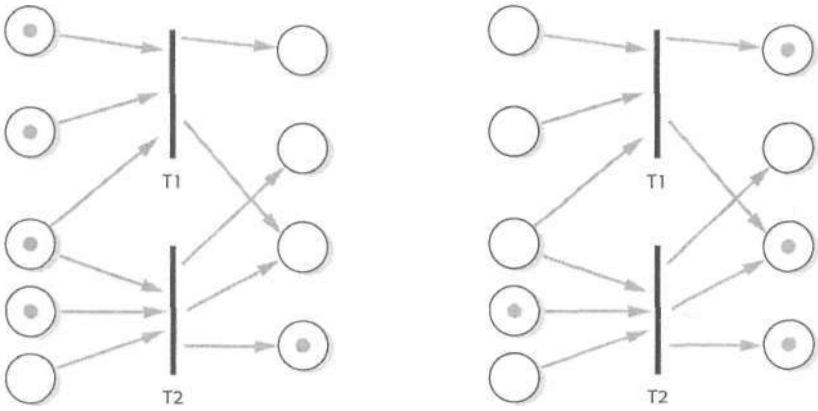
3. **Strelice** Strelice pobezuju mesto sa prelazom, ili obrnuto. Mesto u izvoru strelice naziva se ulazno mesto prelaza na koji strelica ukazuje. Svako mesto na koje strelica ukazuje predstavlja izlazno mesto prelaza u izvoru strelice.

4. **Tokeni** Tokeni, predstavljeni podebljanim tačkama unutar mesta, kolektivno definišu tekuće stanje sistema.

Petri net može da se predstavi grafom. Verteks grafa može da bude ili mesto, ili prelaz, a ivica je predstavljena strelicom. Kod STD-a prelazi stanja su definisani pomeranjem sa jednog verteksa na drugi, duž ivice. Zato je sledeći korak definisanje pravila po kojima token može da se kreće:

- Prelaz je osposobljen ako svako od njegovih ulaznih mesta sadrži token.
- Svaki osposobljeni prelaz može da ispali token. Tokeni se uklanjaju iz ulaznih mesta i smeštaju se u svim izlaznim mestima. Nakon ispaljivanja, može da postoji manje, ili više tokena, u zavisnosti od broja ulaznih i izlaznih mesta.
- U jednom trenutku je moguće ispaljivanje iz jednog prelaza. Medutim, ako je osposobljeno više prelaza, izbor prelaza je neodreden. Za naše potrebe, ovo znači da se izbor vrši proizvoljno. Pošto ispaljivanje prelaza odgovara stvarnim dogadajima, ne želimo da postoje pravila koja bi diktirala redosled kojim se javljaju.

Na slici 8.23a prikazan je Petri net pre ispaljivanja. Postoje dva prelaza, Tl i T2, ali je samo Tl osposobljen (sva ulazna mesta imaju tokeni). Na slici 8.23b prikazan je Petri net nakon ispaljivanja. Tokeni su uklonjeni iz svakog ulaznog mesta za T1.



(a) Pre ispaljivanja (b) Nakon ispaljivanja

SLIKA 8.23 *Petri net pre i posle ispaljivanja*

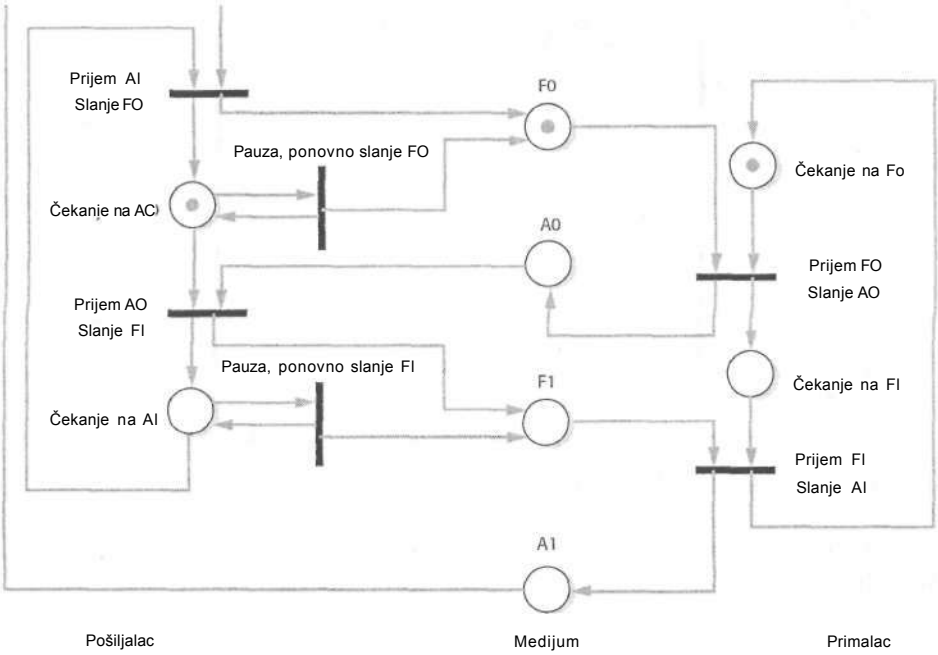
Zatim, token se postavlja u svako izlazno mesto za Tl. Drugi tokeni u mestima pridruženim različitih prelaza ostaju gde jesu.

Pogledajmo kako možemo da koristimo Petri net za modelovanje protokola. Onaj koji ćemo koristiti je go-back-n protokol sa slike 8.20, promenjen tako da se koristi prozor veličine 1. Na slici 8.24 prikazan je deo Petri neta za njega. Kao i ranije, izostavili smo neke delove Petri neta za uprošćavanje dijagrama i naše diskusije.

Umesto da se pokušava da se opiše stanje sistema u jednom verteksu, delimo sistem na delove i predstavljamo stanje svakog. U ovom slučaju, sistem se sastoji od pošiljaoca, primaoca i medijuma izmedu njih. Stanje sistema zavisi od toga šta pošiljalac i primalac čekaju i šta se nalazi na medijumu, kao i kod prethodno dorađenog STD-a. Pošiljalac ima dva stanja, svako predstavljeno mestom. PošiljalaččekanapotvrduzaokvirO ("Čekanjena AO"), iliza okvir 1 ("Čekanjena Al"). I primalac ima dva stanja: čekanje na okvir 0 (FO), ili okvir 1 (Fl). Četiri mesta u sredini predstavljaju ono što se nalazi na medijumu. FO i Fl su mesta koja odgovaraju okvirima 0, ili I koji se prenose. AO i Al odgovaraju potvrdama za okvire 0, ili I koji se prenose.

Tokeni sa slike 8.24 prikazuju tekuće stanje sistema. Pošiljalac šalje okvir 0, koji se trenutno nalazi na medijumu. Primalac čeka na njega, a pošiljalac čeka na potvrdu za njega.

Razmotrimo sada prelaze. Prelazi odgovaraju dogadajima koji mogu da se dese, a njihova ulazna mesta odgovaraju stanjima koja moraju da postoje pre nego što se dogadaj desi. Na primer, pogledajte prvi prelaz za pošiljaoca označen kao "Prijem A1, Slanje FO". Ako ima dva ulazna mesta, "Cekanje na A1" za pošiljaoca i "Al" za medijum, ispaljivanje prelaza znači da je pošiljalac primio ACK za okvir 1 i da je poslao sledeći okvir FO. Medutim, da bi se ovo desilo, pošiljalac mora da čeka na Al i Al mora da se nade na putu.

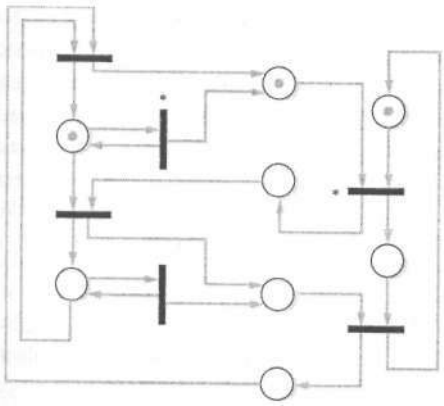


SLIKA 8.24 *Parcijalni Petri net za go-back-n sa veličinom prozora 1 na strani pošiljaoca*

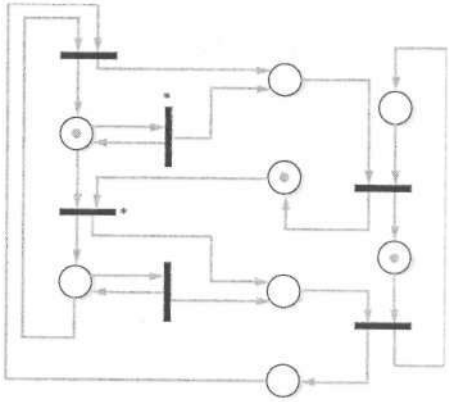
Da bi se prelaz ispalio, tokeni moraju da se nadu u ova dva ulazna mesta. Slične argumente možemo da damo i za druge prelaze za pošiljaoca i primaoca.

Ovaj Petri net ima i dva prelaza pauze (time-out). Svaki ima jedno ulazno mesto koje odgovara pošiljaocu koji čeka na ACK. Na primer, pretpostavimo da pošiljalac čeka na AO (token na tom mestu). Odgovarajući time-out prelaz je osposobljen. Ipak, to ne znači da će ispaliti token. To znači da može da ispali. Ako tajmer okvira istekne, dolazi do ispaljivanja time-out prelaza. Kada se to desi, token se uklanja iz ulaznog mesta i drugi se postavljaju na dva izlazna mesta. Jedno od njih je FO mesto za medijum, koje ukazuje da je okvir 0 poslat. Drugo izlazno mesto je isto kao i ulazno mesto, što znači da pošiljalac ponovo čeka na AO.

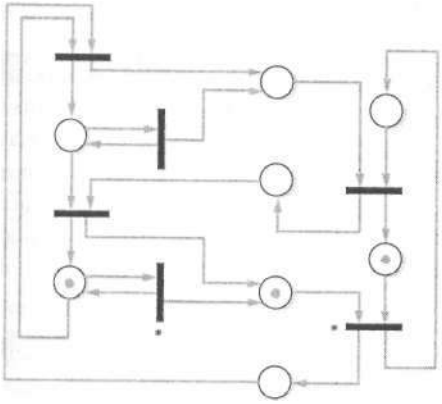
Zbunjeni ste? Ispratimo kretanje tokena u sekvenci tipičnih dogadaja. Delovi (a) do (d) na slici 8.25 prikazuju Petri netove koji odgovaraju sukcesivnim ispaljivanjima prelaza. Mesta, prelazi i strelice su kao na slici 8.24, ali smo eliminisali oznake da bi dijagram bio jednostavniji. Postavka tokena na slici 8.25a je ista kao na slici 8.24. Pošiljalac čeka na AO, primalac čeka na FO, a FO je na medijumu. Zajedno definišu stanje sistema. U ovoj tački su osposobljena dva prelaza (označena sa *): prvi time-out prelaz za pošiljaoca i prelaz za primaoca označenog (na slici 8.24) sa "Prijem FO, Slanje AO".



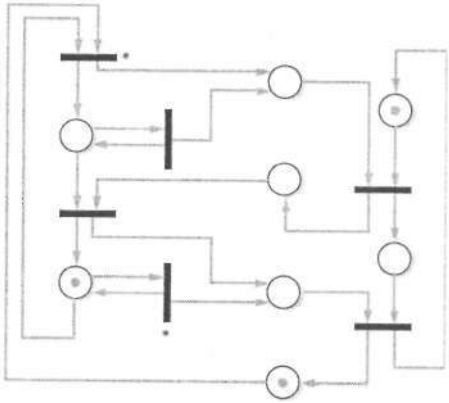
(a) Slanje FO



(b) Slanj'e AO



(c) Slanj'e FI



(d) Sianje AI

* označava osposobljene prelaze

SLIKA 8.25 *Sekvenca ispaljivanja za normalnu razmenu ohvira i potvrda*

Pretpostavimo da drugi prelaz ispali token. Tokeni se uklanjaju iz dva ulazna mesta i postavljaju se u nova mesta, kao što je prikazano na slici 8.25b. Sistem se nalazi u novom stanju. Pošiljalac i dalje čeka na AO, koji se sada nalazi na medijumu. Primalac sada čeka na Fl. Osposobljena su dva prelaza sa slike 8.25b. To su time-out i prijem AO. Drugi prelaz ispaljuje tokene i oni se pomeraju na svoje pozicije prikazane na slici 8.25c. Sistem se nalazi u novom stanju. Pošiljalac je poslao Fl, koji se nalazi na medijumu, i čeka na A1. U meduvremenu, primalac i dalje čeka na Fl. Ako primalac dobije okvir, tokeni se pomeraju na pozicije prikazane na slici 8.25d. Pošiljalac i dalje čeka na AO, koji je u putu, i primalac čeka na FO. Ako potvrda stigne, Petri net se ponovo menja i tokeni se postavljaju kao na slici 8.25a. Ako su okviri i potvrde razmenjeni bez grešaka, ova četiri Petri neta opisuju promene stanja sistema.

Ranije smo istakli da Petri net sa slike 8.24 ne uključuje prelaze za sve moguće dogadaje. Na primer, uvek postoji token na jednom mestu za medijum, što znači da na medijumu uvek nešto postoji. Naravno, to nije tačno. Token na medijumu može da se izgubi, ili može da bude uništen, tako da rezultat bude stanje u kome i pošiljalac i primalac čekaju, a na medijumu se ne nalazi ništa (nešto što Petri net ne prikazuje). Ovo stanje se lako rešava postavljanjem svih mesta na medijumu kao ulaznih mesta za novi prelaz sa oznakom "izgubljen". Ni jedno od tih stanja neće imati izlazna mesta. Svaki put kada se nešto nade na medijumu, ovi prelazi su osposobljeni. Ako se ispali, token se uklanja iz ulaznog mesta. Bez izlaznog mesta, taj token nestaje. Eventualno, time-out prelaz može da ispali token i postavi ga ponovo na jedno od ulaznih mesta.

Sledeći način na koji možemo da doradimo Petri net je da podelimo prelaze pošiljaoca na dva zasebna prelaza. U našem modelu pošiljalac dobija ACK odmah nakon što pošalje sledeći okvir. Pretpostavljamo da uvek postoje okviri za slanje, ali to ne mora da bude tako. Mogli bismo da definišemo nova mesta pošiljaoca koja odgovaraju situacijama u kojima pošiljalac mora da čeka na pakete od svog korisnika. Razmotrite neke od tih slučajeva i ponovo nacrtajte Petri net (videti vežbe na kraju ovog poglavlja).

Kao i kod STD-a, Petri net može da se analizira radi traženja grešaka u protokolu. Na primer, ako tokeni nikada ne mogu da dospeju na određena mesta, odredena stanja ne mogu da se predstave Petri netom. Ako se zna da su moguća, naš model sadrži grešku. Sledeća greška bi se pokazala kada bi se token kretao kroz odredena mesta bez zaustavljanja u mestima izmedu. Na primer, zamislite sekvencu ispaljivanja koja daje Petri nete kod kojih se token naizmenično pomera izmedu mesta primaoca na slici 8.24. Ako, u ovim istim Petri netovima jedno mesto pošiljaoca nikada ne dobija token, postoji greška, jer Petri net ukazuje da primalac dobija okvire, ali ih pošiljalac ne šalje.

Ako ste zainteresovani za dalje proučavanje, ili za neke druge primere Petri neta, pogledajte reference [Ta96], [Wa91] i [Pe81],

8.5 Zaključak

U prethodnim poglavljima su analizirani detalji koji su neophodni za prenos jednog okvira, ali u ovom poglavlju je razmatran prenos više okvira. Prikazani su protokoli koji se bave sledećim temama:

- praćenje više okvira i njihovih potvrda
- reagovanje na okvire koji stižu oštećeni
- reagovanje u situacijama kada okvir, ili potvrda nikada ne stižu, ili kasne

Predstavili smo četiri protokola za kontrolu toka: stop-and-wait, neograničeni, go-back-n i selektivna retransmisija. Druga dva su primeri protokola klizajućih prozora. Definišu prozor za uredaj pošiljaoca koji sadrži okvire koji mogu biti poslati, ali ne moraju još uvek da budu potvrdeni.

Na neki način, isključujući tajmere, ACK i NAK, svi ovi protokoli mogu da se posmatraju kao varijacije jednog protokola klizajućih prozora (tabela 8.1). Na primer, go-back-n protokol u suštini predstavlja selektivnu retransmisiju kod koje prijemni prozor ima samo jedan okvir. Kod stop-and-wait protokola oba prozora imaju samo jedan okvir. Kod neograničenog protokola ne postoji granica za veličine okvira.

Oba protokola klizajućih prozora reaguju na okvire koji stižu oštećeni, ili van redosleda, slanjem negativne potvrde (NAK). Uredaj koji primi NAK mora ponovo da pošalje prethodno poslate okvire. U okviru go-back-n protokola ponovo se šalju svi nerešeni okviri, dok se kod selektivne retransmisije ponovo šalje samo okvir za koji je poslat NAK. Svaki protokol se oslanja na tajmere - ako ne stigne potvrda u odredenom periodu, uredaj pretpostavlja da su jedan, ili više okvira izgubljeni i ponovo ih šalje.

Kod oba protokola klizajućih prozora postoje ograničenja za veličinu prozora. U opštem slučaju, kod go-back-n protokola, ako postoji 2^K različitih brojeva okvira, prozor pošiljaoca mora da ima manje od 2K okvira.

Tabela 8.1: Poredenje protokola za kontrolu toka

	Stop-and-wait protokol	Neograničeni protokol	Go-back-n protokol	Protokol selektivne retransmisije
Veliana prozora kod pošiljaoca	Jedan okvir	Neograniceni broj okvira	Manje od 2^K	Manje, ili jednako 2^K, minus veličina prijemnog prozora (ali obično iznosi 2^{K1})
Veličina prijemnog minus prozora	Jedan okvir	Neograničeni broj okvira	Jedan okvir	Manje nego, ili jednako 2^K, veličina prozora na strani pošiljaoca, ali obično iznosi 2^{K1})
Komentari	Čeka se na zasebnu potvrdu za svaki okvir pre nego što se pošalje sledeći. Ovaj pristup je sporiji, jer postoje veliki periodi čekanja.	Šalju se svi okviri, bez obzira koliko ih ima. Ovo može da stvori probleme zagušenjem saobraćaja na mreži i može da zatrpa prijemnu stranu. Kao što smo opisali, nismo uključili mogućnosti da se okviri oštete, ili izgube u toku prenosa.	Prozor sadrži okvire koji su poslati, ali za njih još uvek nije stigla potvrda. Pretpostavlja se da prijemna strana prihvata i isporučuje okvire onim redosledom kojim pristižu. Ovo je jednostavnije od selektivne retransmisije i funkcionisaće dobro ako se okviri retko gube, ili stižu van redosleda. Pošiljalac će ponovo poslati sve okvire u prozoru ako stigne NAK.	Sioženiji je od go-back-n protokola, jer možda ne mora ponovo da šalje okvire koji kasne. Oni se prihvataju u bafere, a zatim isporučuju pravilnim redosledom do korisnika, kada stignu svi prethodni okviri. Ovo je korisno kod prenosa na velikim rastojanjima gde zbog kašnjenja i gustog saobraćaja okviri često stižu van redosleda. Ako stigne NAK, pošiljalac treba ponovo da pošalje samo okvir koji je specifično naznačen.

Kod selektivne retransmisije suma veličina prozora na strani pošiljaoca i primaoca ne sme da bude veća od 2^K. Narušavanje ovog ograničenja ne znači nužno i grešku u protokolu, ali znači da su greške moguće kod odredenih sekvenci dogadaja.

Algoritmi su složeni, a navođenje formalnih matematičkih dokaza prelazi predviđeni obim ovog teksta. Međutim, u odeljku 8.7 predstavljene su dve alatke koje mogu da se koriste za verifikaciju: dijagram prelaza stanja i Petri net. Obe koriste usmerene grafove, koji predstavljaju stanja i prelaze izmedu stanja sistema, ali se razlikuju po tome šta rade. Dijagrami prelaza stanja koriste čvorove za stanja i ivice za prelaze stanja. Izvršenje algoritma može da se izvede definisanjem putanja kroz dijagram prelaza stanja. Petri net modeli su složeniji. Oni koriste mesta, prelaze, strelice i tokene. Kolekcija tokena odgovara mestima koja definišu stanje sistema. Tokeni koji se pomeraju sa jednog mesta na drugo podležu pravilima koja omogućavaju ispaljivanje odredenog prelaza. U oba slučaja, analiza modela može da se iskoristi za detektovanje mogućih anomalija.

Pitanja i zadaci za proveru

1. Sta je kontrola grešaka sa automatskim ponavljanjem zahteva?
2. Šta je kontrola toka?
3. Šta su X-ON i X-OFF karakteri?
4. Sta je neograničena kontrola toka?
5. Šta je stop-and-wait kontrola toka?
6. Da li su sledeća tvrđenja tačna, ili netačna (zašto)?
   a. Neograničena kontrola toka u opštem slučaju ima bolju efektivnu brzinu prenosa podataka u odnosu na stop-and-wait kontrolu toka.
   b. Neograničena kontrola toka predstavlja, u stvari, odsustvo kontrole toka.
   c. Neoganičena i stop-and-wait kontrola toka predstavljaju specijalne slučajeve protokola klizajućih prozora.
   d. Protokoli klizajućih prozora mogu da funkcionišu sa bilo kojom veličinom prozora.
   e. Go-back-n algoritam ponovo šalje nekoliko okvira ako samo jedan od njih stigne na odredište sa oštećenjem.
   f. Kod protokola selektivne retransmisije veličina prijemnog prozora je nezavisna od veličine prozora na strani pošiljaoca.
   g. Petri net i konačni automati predstavljaju dva različita načina za postizanje iste "stvari".
7. Navedite razliku izmedu bitske brzine i efektivne brzine prenosa podataka.
8. Koju značajnu ulogu igraju potvrde u protokolima za kontrolu toka?
9. Sta se podrazumeva pod kontrolom toka pomoću protokola sa klizajućim prozorima?
10. Sta je piggyback potvrda?
11. Zašto se kod numeracije okvira koristi modulama aritmetika, umesto da se brojevi jednostavno uvećavaju po potrebi?
12. Navedite tipična polja u okviru sa podacima.

13. Navedite razliku izmedu tajmera okvira i ACK tajmera.
14. Kakva je svrha određivanja veličine prozora u protokolima klizajućih prozora za kontrolu toka?
15. Navedite razlike izmedu ACK, NAK i okvira sa podacima.
16. Koje su glavne razlike izmedu go-back-n protokola i protokola selektivne retransmisije?
17. Kakvo ograničenje u veličini prozora na strani pošiljaoca postoji kod protokola selektivne retransmisije?
18. Kod protokola selektivne retransmisije kakva relacija postoji izmedu veličine prozora na strani pošiljaoca i primaoca?
19. Šta je konačni automat (ili model konačnog stanja)?
20. Sta je dijagram prelaza stanja?
21. Šta je Petri net?
22. Definišite termine *mesto, prelaz, strelica* i *tohen* u kontekstu Petri net modela.
23. Šta se misli kada se kaže da prelaz ispaljuje?

Vežbe

1. Sta se dešava ako A i B sa slike 8.2 umetnu X-OFF karaktere u svoje nizove podataka?
2. Kod X-ON/X-OFF protokola zašto jedan uredaj šalje X-OFF karakter pre nego što se baferi napune, umesto da sačeka da budu puni?
3. Modifikujte neograničeni protokol sa slike 8.4 tako da odražava sledeće promene:
   a. Pošiljalac ima fiksni broj okvira za slanje.
   b. Okvir može da bude oštećen.
4. Kolike su efektivne brzine prenosa podataka za neograničeni i stop-and-wait protokol sa sledećim vrednostima?

   R = kapacitet (16 Mbps)

   S = brzina signala (200 metara/irsec)

   D = rastojanje izmedu pošiljaoca i primaoca (200 metara)

   T = vreme potrebno za kreiranje jednog okvira (2 usec)

   F = broj bitova u jednom okviru (500)

   N = broj bitova podataka u jeđnom okviru (450)

   A = broj bitova potvrde (80)
5. Za svaku promenljivu (osim za brzinu signala) iz vežbe 4 kako bi 10-tostruko povećanje vrednosti uticalo na efektivnu brzinu prenosa podataka i za neograničeni i za stop-and-wait protokol?
6. Scenario na slici 8.10 pokazuje da kod jednosmernog go-back-n protokola može da dode do greške, jer je veličina prozora jednaka maksimalnom broju numeracija okvira u sekvenci. Opišite drugi način kod koga u protokolu može da dode do greške pod istom pretpostavkom.
7. Razmotrite go-back-n algoritam na slici 8.12 sa veličinom prozora 7. Opišite akcije protokola za slanje i prijem, definišući vrednosti promenljivih i sadržaje bafera u

sledećim slučajevima. Kakvo je tekuće stanje svakog protokola nakon reagovanja na specifični događaj?

a. Uredaj A šalje okvire 0 do 6. Uređaj B ih prima pravilnim redosledom, ali okvir 4 je oštećen.
b. Uredaj A šalje okvire 0 do 6, a uredaj B ih prima ispravnim redosledom. Uredaj B šalje jedan okvir sa podacima do A (A ga ispravno prima) nakon prijema okvira 4, ali pre primanja okvira 5.
c. Isti scenario kao i pod (b), ali u toku prenosa dolazi do oštećenja okvira koji se šalje do A
d. Uređaj A ima 12 okvira za slanje do B, ali B nema ništa za slanje do A.

8. Kako može da dode do greške go-back-n algoritma sa slike 8.12 pod svakim od sledećih uslova?
   a. uklanjanje provere za istek ack tajmera
   b. uklanjanje provere za istek tajmera okvira
   c. uklanjanje provere za uslov (i<N) u prvoj naredbi glavne petlje
   d. uklanjanje naredbe w= (f rame. ack+1) % MAX iz koda kod provere poslednjeg dogadaja
9. Reprodukujte scenarije sa slike 8.14 i 8.15 ako su veličine oba prozora jednake 4 i pokažite da neće doći do greške u protokolu.
10. Uzmite protokol selektivne retransmisije sa slike 8.16 sa veličinom prozora 4. Opišite akcije protokola za slanje i prijem, navodeći vrednosti promenljivih i sadržaje bafera u sledećim slučajevima. Kakvo je tekuće stanje svakog protokola nakon reagovanja na naznačene događaje?
    a. Uredaj A šalje okvire 0 do 3. Stižu svi osim okvira 2. Okvir 2 je izgubljen.
    b. Uredaj A šalje okvire 0 do 3. U B stižu redosledom O, 1, 3, 2.
    c. Uređaj A šalje okvire 0 do 3. Uredaj B prima okvire O i 1 i šalje piggyback potvrdu, koju A prima.
    d. Isti scenario kao i pod (c), ali potvrda se gubi
11. Uzmite protokol selektivne retransmisije sa slike 8.16. Konstruišite scenario kod koga važi uslov frame.type==nak, ali ne važi uslov koji sledi iza njega.
12. Kako može da dode do greške u algoritmu selektivne retransmisije sa slike 8.16 pod svakim od narednih uslova?
    a. Uklanja se provera za istek ack tajmera.
    b. Uklanja se provera za istek tajmera okvira.
    c. Uklanja se provera za uslov (i<N) u prvoj naredbi glavne petlje.
    d. Uklanja se naredba *w={frame.ack+1)* % MAX iz koda kod provere poslednjeg događaja.
13. Uzmite analizu protokola klizajućih prozora iz odeljka 8.3 i pretpostavite sledeće vrednosti:

    R - kapacitet (10 Mbps, ili 10 bitova/irsec)

    S = brzina signala (200 metara/|j.sec)

D = rastojanje izmedu pošiljaoca i primaoca (nepoznato)

T = vreme potrebno za kreiranje jednog okvira (1 |isec)

F = broj bitova u jednom okviru (200)

N = broj bitova podataka u jednom okviru (160)

W = velidna prozora (4 okvira)

Na kom rastojanju prva potvrda stiže tačno u isto vreme kada se pošalje poslednji okvir iz prozora? Kolika je efektivna brzina prenosa podataka?

14. Uzmite odgovor na vežbu 13. Šta se dešava sa efektivnom brzinom prenosa podataka ako se poveća veličina prozora i ako se smanji?
15. Ponovite vežbu 14 za svaki od preostalih parametara.
16. Uzmite dijagram prelaza stanja sa slike 8.21. Zašto ne postoji prelaz iz stanja (Y, N, 1) u stanje (Y, N, 0)?
17. Proširite uokvirenu oblast sa slike 8.21 tako da uključi stanja (Y, N, 0) i (Y, N, 1) i doradite proširenje da dobijete rezultat sa slike 8.22.
18. Uzmite Petri net sa slike 8.24. Razmotrite slučaj u kome pošiljalac prima ACK, ali možda nema okvir za slanje - mora da čeka na korisnika da mu obezbedi paket. Kako Petri net izgleda?
19. Nacrtajte sekvencu Petri net modela (slično onima sa slike 8.25) za sledeći niz dogadaja:
    a. Slanje FO
    b. . Pauza, ponovno slanje FO
    c. Primalac dobija FO i šalje AO.
    d. Pošiljalac dobija AO i šalje F1.
    e. Pauza, ponovno slanje Fl
20. Do pauze može da dode nakon što primalac dobije prvi okvir i pošalje ACK, ali pre nego što pošiljalac dobije taj ACK. Pošiljalac može ponovo da pošalje okvir koji primalac ne očekuje. Modifikujte Petri net sa slike 8.24 tako da se uzme u obzir i ova mogućnost.

Reference

[Li87] Lin, F., P. Chu, and M. Liu. "Protocol Verification Using Reachibility Analysis: The State Space Explosion Problem and Relief Strategies." *Proceedings of the ACM SIGCOMM 1987 Workshop* (1987), 126-135.

[Pe81] Peterson, J. *Petri Net Theory and the Modeling of Systems.* Englevrood Cliffs, NJ: Prentice-Hall, 1981.

[Ru89] Russel, D. *The Principles of Computer Netivorking.* New York: Cambridge University Press, 1989.

[Ta96] Tanenbaum, A. S. *Computer Netmorks: A First Course.* Boston: Richard D. Invin, 1991.

POGLAVLJE

9 Lokalne mreže

Što **u** bankama podataka postoji više zapisa o nama, utoliko manje postojimo.

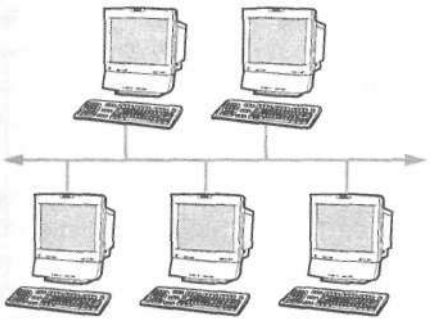
—**Marshall McLuhan** (1911 1980), kanadski teoretičar komunikacija

9.1 Uvod

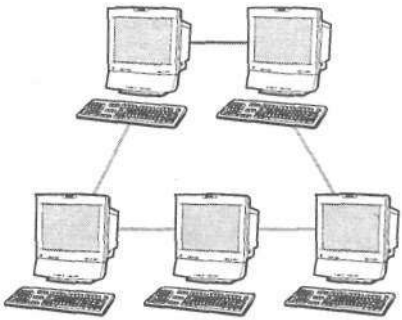
Do sada smo se fokusirali na komunikacije izmedu dva uređaja i, uz izuzetak multipleksiranja i "nadmetanja" u odeljcima 4.5 i 4.7, nismo uzimali u obzir širu sliku sa većim brojem povezanih uređaja. Počevši od ovog poglavlja, predstavićemo strategije povezivanja i protokole koji su neophodni za održavanje komunikacije izmedu većeg broja uređaja. Ovo poglavlje se bavi prvenstveno protokolima sloja 2 koji defmišu operacije lokalnih (LAN) mreža. U narednim poglavljima opisujemo načine za povezivanje više LAN mreža i prikazujemo protokole koji defmišu Internet.

Podnjemo sa opštim pregledom različitih LAN topologija (konfiguracija), prikazanih na slici 9.1. Istorijski, dve najčešće korišćene topologije su bile magistrala i prsten. Kasnije ćemo videti da su tehnologije povezivanja obezbedile veći dijapazon mogućnosti. Kod **topologije magistrale** (slika 9.1a) jedna komunikadona linija - obično koaksijalni kabl, ili optički fiber, predstavlja primarni medijum, koji nazivamo segment. Svaki uređaj koji treba da pošalje nešto do drugog uredaja izvodi to preko segmenta. Ipak, samo jedan uređaj može da šalje podatke u jednom trenutku i zato je neophodno obezbediti neku vrstu protokola za "nadmetanje". Kod **topologije prstena** (slika 9.1b) svi uredaji su uredeni u prsten, u kome je svaki uređaj direktno povezan sa svoja dva suseda. Ako uredaj želi da pošalje okvir do drugog uređaja, okvir mora da prode kroz sve ostale uredaje koji se nalaze izmedu njih (bilo u smeru, ili suprotno od smera kretanja kazaljki na časovniku). Ovo je pomalo nalik tračevima koji idu od jednog suseda do drugog.

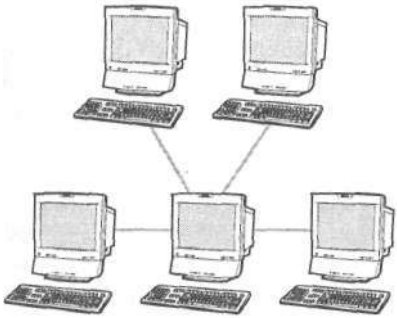
Ostale topologije su topologija zvezde i potpuno povezana topologija. Kod **topologije** zvezde (slika 9.1c) jedan uredaj služi kao logički komunikacioni centar za sve ostale. Sva komunikacija izmedu bilo koja dva uredaja mora da se odvija preko njega. Konačno, **potpuno** povezana **topologija** (slika 9.Id) direktno povezuje svaka dva para uređaja. Potpuno povezane topologije



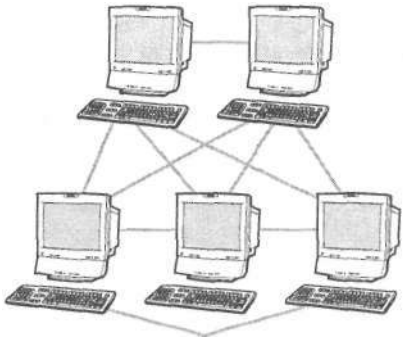
(a) Topologija magistrale



(b) Topologija prstena



(c) Topologija zvezde

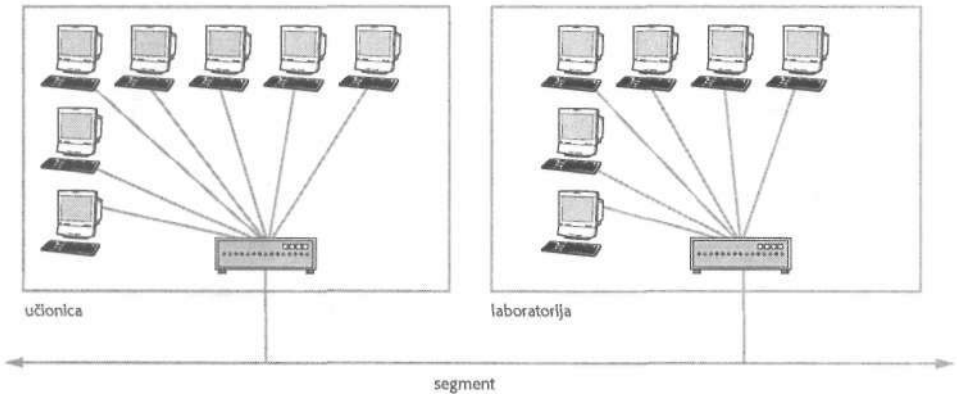


(d) Potpuno povezana topologija

SLIKA 9.1 *Mrežne topologije*

Prednost topologije magistrale je što je jednostavna. Segment može da se prostire preko jedne ili više zgrada, sa vodovima koji prolaze kroz specifične laboratorije, ili udonice (slika 9.2) i povezuju personalne kompjutere pomoću komutatora, ili hubova (uređaji koje ćemo kasnije opisati u ovom poglavlju). Takode, može da se prostire duž proizvodne linije u fabrici, povezujući uredaje koji su neophodni za sklapanje nekog proizvoda, kao što je automobil. Zbog linearne organizacije, dodavanje novih, ili uklanjanje starih uredaja je relativno jednostavno. Nedostatak linerane organizacije je što u jednom trenutku samo jedan uredaj može da šalje podatke. U nekim slučajevima ovo ograničenje ne predstavlja problem. Ipak, kako se broj uredaja povećava, moguća su ozbiljna "uska grla".

Topologija prstena omogućava slanje podataka iz više uredaja korišćenjem jednog, ili više tokena koji kruže prstenom. Sećate se da smo u odeljku 4.7 istakli da je *token* specijalni okvir koji uredaju koji ga poseduje omogućava slanje podataka. Najčešće implementirane mreže sa topologijom prstena koriste samo jedan token, mada neki protokoli obezbeduju više tokena. Topologije prstena su ranije bile uobičajene za kancelarijska okruženja, gde je bilo neophodno obezbediti komunikaciju izmedu više personalnih kompjutera, ili komunikaciju sa fajl serverom, ili deljenim štampačem.



SLIKA 9.2 *Topologija magistrale koja povezuje nekoliko lokacija*

Prvi korak u predstavljanju IAN operacije je definisanje protokola koji uređaji koriste za medusobnu komunikaciju. Ako mislite da ovo pomalo podseća na diskusiju iz prethodnog poglavlja, potpuno ste u pravu. Odeljak 9.2 se nadovezuje na taj materijal - u njemu je prikazan dugogodišnji ISO standard za veze izmedu podataka na osnovu koga su izvedeni mnogi kasniji standardi.

U odeljcima 9.3 i 9.6 predstavićemo opštepoznate IEEE standarde za mreže sa topologijom magistrale i prstena: nekoliko varijacija Etherneta i token ring mrežu. Ethernet je postao dominantan način za povezivanje uredaja u LAN okruženjima, a prošao je kroz nekoliko evolucija kako bi zadovoljio napredak tehnologija i bitskih brzina. Iako je ranije bio dosta čest, token ring se sve rede koristi. Ipak, pruža sjajan kontrast idejama koje definiše Ethernet. Postoji i treći IEEE standard, poznat kao *token bus.* Ima topologiju Etherneta, ali je kod "nadmetanja" neophodan token koji kruži izmedu uređaja. Iako je reč o standardu, retko se koristi, tako da ga nećemo opisivati u ovom izdanju knjige.

Svi ti LAN standardi imaju jednu zajedničku karakteristiku: uređaje povezuju pomoću UTP kabla, koaksijalnog kabla, ili optičkog fibera. Medutim, verovatno se sećate da smo u odeljku 2.4 naglasili da fizičke konekcije više nisu neophodne za ostvarivanje komunikacije izmedu uređaja i da su bežične tehnologije došle do tačke u kojoj su postale i ekonomične i uobičajene. To je naravno delimično omogućilo razvoj standarda kao što je IEEE 802.11, bežični standard o kome će biti reči u odeljku 9.7.

9.2 Kontrola veze između podataka

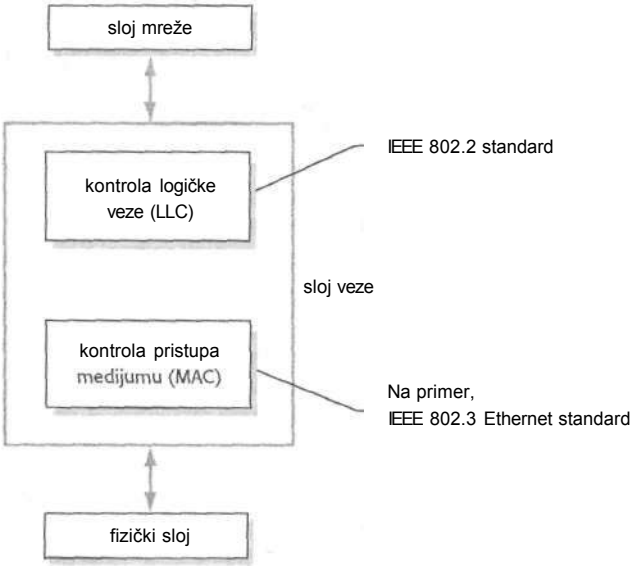
Da biste u potpunosti razumeli kako funkcionišu LAN standardi, važno je da razumete gde se uklapaju u slojeviti dizajn i kakva je njihova veza sa ostalim temama koje smo do sada obradili. Sećate se iz odceljka 4.1 sedmoslojnog OSI referentnog modela. Najniža tri sloja obično definišu mrežne operacije: fizički, sloj veze i sloj mreže.

Sloj veze izvršava servise koji su neophodni za sloj mreže i pretpostavlja postojanje fizičkog sloja. Sloj veze je odgovoran za tačnu komunikaciju izmedu dva čvora na mreži. Ovo uključuje formate okvira, proveru grešaka i kontrolu toka. U opštem slučaju, sve ove teme su nezavisne od mrežne topologije. Na primer, provera grešaka, ili algoritmi za kontrolu toka ne vode računa da li se okvir šalje preko magistrale, ili prstena.

Zboga toga se sloj veze dalje deli na dva podsloja, koja su prikazana na slici 9.3: na **kontrolu logičke** veze (LLC - logical link control) i na **kontrolu pristupa medijumu** (MAC - medium access control). LLC vodi računa o logičkim vezama izmedu uredaja, dok MAC kontroliše pristup prenosnom medijumu. Primamo, LLC obezbeduje servise za sloj mreže i poziva MAC radi obavljanja zadataka koji su specifični za odredeni tip mreže.

IEEE 802.3 Ethernet i IEEE 802.5 token ring standardi koje ćemo predstaviti nešto kasnije predstavljaju MAC protokole. Vidimo kako uslojavanje softvera omogućava funkcionisanje istog višeg nivoa sa različitim protokolima nižeg nivoa. Mnoge od tema koje smo obradili ne zavise od mrežne topologije, što obezbeduje veliku fleksibilnost i mogućnost širokog plasiranja na tržištu.

Mnogi različiti protokoli veze podataka mogu da se postave iznad MAC-a, ali interesantno je da svi imaju zajedničkog "pretka". Početkom 70-ih godina prošlog veka IBM je razvio protokol Synchronous Data Link Control (SDLC). Pre njega protokoli su bili **orijentisani bajtovima.** To znači da su okviri bili interpretirani kao sekvence bajtova; svaki bajt je sadržao neke informatije i bio je definisan pomoću EBCDIC koda (sećate se odeljka 2.5). Ponekad bi se u sklopu niza podataka pojavili i neki kontrolni bajtovi, tako da se dešavalo da se i podaci pogrešno interpretiraju kao kontrolne informacije. Iako su postojali načini da se ovaj problem prevazide, IBM je razvio **bitovima orijentisani protokol,** SDLC, kod koga sve nije moralo da se posmatra kao sekvenca bajtova.



SLIKA 9.3 *Dorada sloja veze*

Ovaj pristup obezbeduje malo bolju fleksibilnost kod prenosa većeg broja različitih tipova podataka. SDLC koristi go-back-n kontrolu toka i bio je deo IBM-ove System Network Architecture (SNA), ekvivalenta OSI modelu. Korišćen je tipično u komunikacijama IBM terminala i kompjutera.

IBM je prosledio SDLC protokol do ISO radi proglašavanja standardom. Međutim, u ISO je modifikovan i kreiran je novi standard, poznat kao **HDLC (High-Ievel Data Link Control);** predstavljao je prvi formalni standard za bitovima orijentisane protokole za kontrolu veze između podataka. Efektivno, SDLC je IBM-ov ekvivalent za HDLC (ako posmatrate sa druge strane, HDLC je ISO ekvivalent za SDLC). Ipak, ovde se "priča" ne završava. IBM je poslao SDLC i do ANSI radi usvajanja. Kao i sve dobre organizacije sa standardizaciju, ANSI je modifikovala protokol i nazvala ga Advanced Data Communications Control Procedure (ADCCP).

ITU je usvojio i modifikovao HDLC za korišćenje sa njegovim X.25 mrežnim interfejs standardom (prikazan je u Poglavlju 13). Originalno je označen kao LAP (skraćeno od Link Access Protocol), ali je kasnije promenjen u LAPB (B je oznaka za balansirani). Omogućio je povezivanje uredaja na mreže sa komutacijom paketa. Varijacija LAPB je LAPD, kontrola veze za Integrated Services Digital Network (ISDN). ISDN je u potpunosti digitalni komunikacioni sistem koga defmiše ITU. Neki su predvidali da će eventualno zameniti telefonski sistem, ali razvijene su i druge tehnologije i, naravno, do toga nije došlo. LAPD omogućava komunikaciju uredaja preko ISDN D kanala (prikazan je u Poglavlju 13). IEEE ima protokol koji je izveden iz HDLC protokola. Naziva se LLC (Logical Link Control) i koristi se u lokalnim mrežama. Takode, omogućava povezivanje više LAN mreža, tako da se kreira WAN mreža.

Kao što možete da vidite, protokola za kontrolu veze izmedu podataka ima sasvim dovoljno, ali za sve njih je karakteristična jedna "stvar". Mnogo duguju HDLC protokolu. Pošto nemamo dovoljno prostora da obuhvatimo sve te protokole (a verovatno ne bi bilo ni mnogo koristi od toga za Vaše obrazovanje), objasnićemo njihovog "pretka" HDLC. Detalji o nekim drugim protokolima mogu da se nađu u referencama [Fo03] (za LLC) i [St99] (za LAPD).

High-level Data Link Control (HDLC) protokol

HDLC je bitovima orijentisani protokol koji podržava i half-duplex i full-duplex komunikacije (videti odeljak 4.3). Kao što smo ranije istakli, *orijentisan bitovima* znači da protokol sve okvire tretira kao nizove bitova. Drugim rečima, ne prepoznaje ih, niti ih interpretira kao bajtove (kao X-ON/X-OFF), kao što je bio slučaj kod nekih ranije analiziranih protokola.

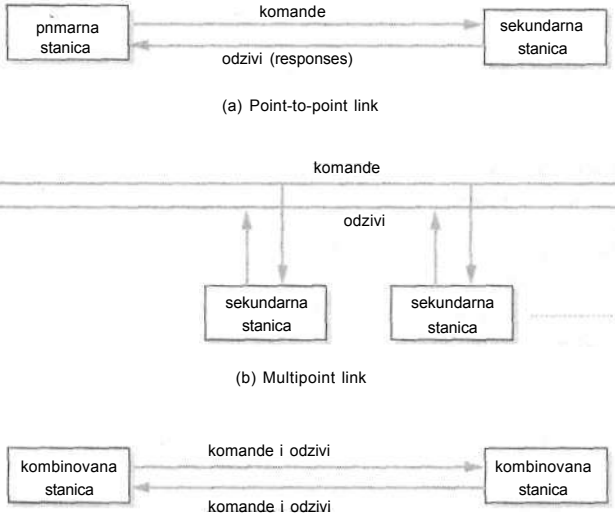
HDLC protokol izvršavaju tri tipa uredaja:

- **Primarna stanica** (ponekad se naziva i *host stanica,* ili *kontrolna stanica)* Upravlja tokom podataka tako što izdaje komande za druge uredaje, ili reaguje na njihove odzive. Kasnije ćemo videti i neke primere. Može da uspostavlja i kontroliše konekcije izmedu više uredaja.
- **Sekundardna stanica** (ponekad se naziva *ciljna stanica,* ili *gostujuća stanica)* Može da se odazove samo jednoj primarnoj stanici u jednom trenutku. Ne izdaje nikakve komande drugim uredajima (iako može da šalje podatke).

- **Kombinovana stanica** Kao što se vidi iz samog naziva, može da se ponaša i kao primarna i kao sekundarna stanka. Može da izdaje komande i da reaguje na komande koje je izdala neka druga kombinovana stanica.

Uredaji koji izvršavaju HDLC mogu da komuniciraju u jednom od tri moguća moda:

- **Mod normalnog odziva (NRM - normal response mode)** U NRM primarna stanica kontroliše komunikacije. Naime, sekundardna stanica može da šalje podatke samo kad joj primarna stanica to naloži, ili dopusti. Ovaj operacioni mod je uobičajen kod dve konfiguracije. Kod point-to-point linka (slika 9.4a) primarna stanica komunicira sa jednom sekundardnom stanicom. Kod multipoint linka (ponekad se označava i kao multidrop link) primarna stanica može da komunicira sa nekoliko sekundarnih stanica (slika 9.4b). Naravno, mora da kontroliše i da održava zasebne sesije za svaku sekundardnu stanicu.
- **Mod asinhronog odziva (ARM - asynchronous response mode)** Kao i NRM, ARM uključuje komunikaciju izmedu primarne i jedne, ili više sekundarnih stanica. Ovde sekundardna stanica ima veću nezavisnost. Može da šalje podatke, ili kontrolne informacije do primarne stanice, bez eksplicitnih instrukcija, ili dozvola za to. Medutim, ne može da šalje komande. Odgovornost za uspostavljanje, održavanje i eventualno okončavanje konekcija i dalje je poverena primarnoi stanici.



SLIKA 9.4 *HDLC konflguracije*

- **Asinhroni balansirani mod (ABM)** ABM se koristi u konfiguracijama za povezivanje kombinovanih stanica (slika 4.9c). Svi uredaji mogu da šalju podatke, kontrolne informacije, ili komande. Ovo je tipična konekcija izmedu dva kompjutera i kod X.25 interfejs standarda (predstavljen je u Poglavlju 13).

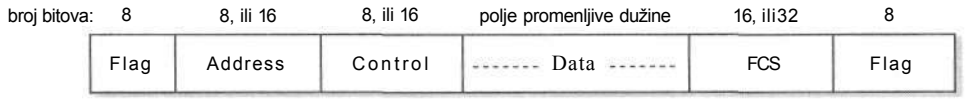
ftjffliat ***oWm*** *HDLC okviri su slični opštim formatima, koje smo ranije predstaviH. Na slici 9.5* prikazan je format okvira. Neka polja mogu da imaju dve različite veličine. Manja veličina definiše standardni format, dok veća daje proširen format. Odluka o formatu koji će biti korišćen donosi se nakon uspostavljanja linka.

Postoje tri različita tipa okvira. Razlikuju se po sadržaju polja Control i da li okvir sadrži stvarne podatke. Najpre ćemo prikazati polja koja su zajednička za sve tipove okvira, a zatim ćemo razmotriti razlike izmedu okvira.

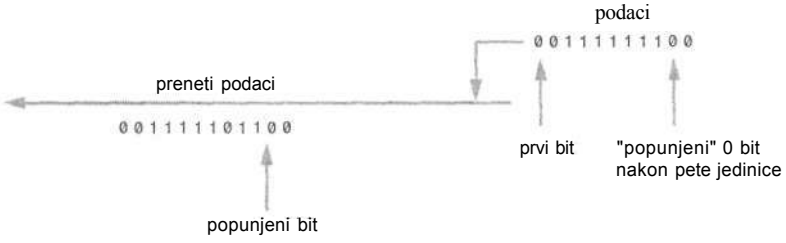
Polje Flag označava početak i kraj svakog okvira i sadrži specijalni uzorak bitova 01111110. Uredaj koji primi ovaj uzorak "zna" da je HDLC okvir na putu. Pošto veličina okvira može da bude različita, uredaj proučava dolazeće bitove i traži ovaj uzorak kako bi detektovao kraj okvira. Ovaj uzorak predstavlja problem: pošto je protokol orijentisan bitovima, polja sa podacima (i druga polja) mogu da sadrže proizvoljne kombinacije bitova. Ako se uzorak flega nađe u nekom drugom polju, zar ga uređaj neće pogrešno interpretirati kao kraj okvira? Definitivno ne želimo da ograničimo podatke tako da ne mogu da sadrže odredene uzorke bitova.

Srećom, ovaj problem ima relativno lako rešenje, poznato pod nazivom **popunjavanje bitova** (bit stuffmg). Uredaj koji šalje podatke nadgleda bitove izmedu flegova pre nego što ih pošalje. Ako detektuje pet uzastopnih jedinica (slika 9.6), umeće dodatnu nulu nakon pete jedinice. Tako se "razbija" potencijalni uzorak flega i sprečava slanje takvih bitova. Sada imamo pogrešne podatke koje prijemni uredaj mora da ispravi. Svaki put kada naide na nulu iza pet uzastopnih jedinica, on pretpostavlja da je ubačena i uklanja je. Pošto polje flega nije predmet popunjavanja bitova na strani pošiljaoca, to je jedino mesto na kome je moguć ovakav uzorak bitova.

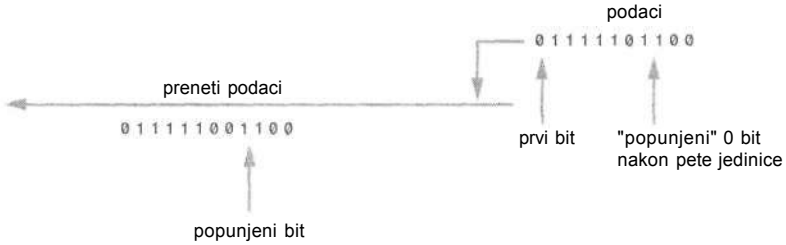
Polje Address je jasno samo po sebi. Ima 8 bitova kod standardnog formata i 16 bitova u proširenom formatu. Prošireni format omogućava identifikovanje većeg broja uređaja. Ako primarna stanica pošalje okvir, polje Address definiše sekundardnu stanicu kojoj se okvir šalje. Ovo je neophodno kod multipoint konfiguracija, kada postoji veći broj sekundarnih stanica. Ako sekundarna stanica šalje okvir, polje Address sadrži identitet pošiljaoca. Pošto postoji samo jedna primarna stanica i sekundardne stanice ne mogu da šalju podatke izmedu sebe, odredišna adresa nije potrebna. Izvorna adresa je neophodna samo da bi primarna stanica znala odakle okvir potiče.



SLIKA 9.5 *Format HDLC okvira*



(a) Više od pet uzastopnih jedinica



(b) Pet uzastopnih jedinica

SLIKA 9.6 *Popunjavanje bitova*

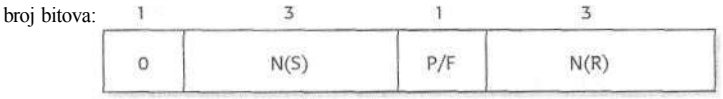
U nekim slučajevima polje može da sadrži **grupnu adresu,** ili **emisionu adresu** (broadcast address) (sve jedinice). Sve sekundardne stanice u preddefmisanoj grupi prihvataju okvir sa grupnom adresom. Svaka sekundardna stanica sa kojom primarna stanica uspostavlja link prihvata okvir sa emisionom adresom.

Polje Data sadrži podatke i promenljive je dužine. Videćemo da u nekim slučajevima nema podataka i da ovo polje ne postoji. Za CRC detekciju grešaka koristi se **Frame Check Sequnece (FCS).** Polje može da bude dugačko 16 bitova (standardni format), ili 32 bita (prošireni bit). Najčešće se sreće 16-bitno polje, koje se definiše pomoću CRC polinoma $x^{16} + x^{12} + x^5 + 1$, kao što je opisano u odeljku 4.3.

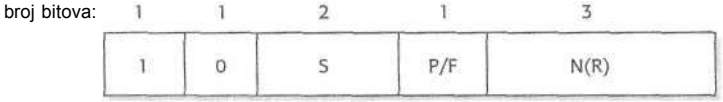
Polje Control ima osam (standardni format), ili 16 bitova (prošireni format) i koristi se za slanje informacija o statusu, ili za izdavanje komandi. Njegov sadržaj zavisi od tipa okvira. Postoje tri tipa: informacioni okvir, supervizorski okvir i nenumerisani okvir. Na slici 9.7 prikazan je standardni format svakog tipa okvira. Kod proširenog formata polja su veća, ili okvir sadrži neiskorišćene bitove; ovde razlike nisu bitne.

Prvi bit, ili prva dva bita definišu tip okvira. Kao što je pokazano na slici *9.7,* informacioni okvir uvek počinje sa O, supervizorski okvir sa 10, a nenumerisani okvir sa 11. Zahvaljujući ovim defmicijama, prijemni uređaji uvek mogu da utvrde tip pristiglog okvira.

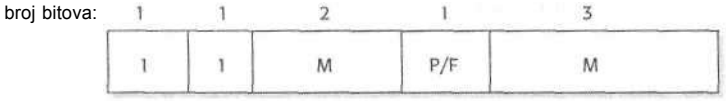
Informacioni okviri se koriste prvenstveno za prenos informacija (polje Data sa slike 9.5), koristeći ili go-back-n protokol, ili protokol klizajućih prozora sa selektivnom retransmisijom.



(a) Informacioni okvir



(b) Supervizorski okvir



(c) Nenumerisani okvir

SLIKA 9.7 *Kontrolna polja za HDLC okvire*

Polja N(R) i N(S) su slična onomima koja smo ranije nazivali frame.ack i frame.number, respektivno. N(R) (broj primljenih okvira) je piggyback potvrda koja ukazuje da su svi okviri do N(R) - 1 primljeni. Ekvivalentno, uređaj trenutno očekuje okvir numerisan kao N(R). Slično tome, N(S) je broj poslatih okvira. Polja N(R) i N(S) su obično dugačka tri (standardni okvir), ili sedam bitova (prošireni okvir). Zato se numerisanje okvira i aritmetika poput N(R) - 1 izvode po modulu 8 (2^3), ili po modulu 128 (2^7).

P/F bit označava Poll/Final bit. Njegovo značenje zavisi od toga da li bit šalje primarna (PoIl bit), ili sekundardna stanica (Final bit). Primarna stanica može da zahteva odziv sekundame stanice slanjem okvira sa P bitom postavljenim na 1. Na primer, primarna stanica možda hoće da zna da li sekundardna ima podatke za slanje, ili može da zahteva njen status u vezi nekog procesa. LI svakom slučaju, očekuje se odziv sekundardne stanice (uskoro ćemo videti i neke primere). Kada sekundardna stanica šalje okvir, F bit ukazuje da je tekući okvir poslednji u nizu okvira.

Supervizorski okviri se koriste u svim uredajima za ukazivanje na tekući status, ili za negativne potvrde (NAK) neispravno primljenih okvira. N(R) i P/F bitovi rade isto kao i kod informacionih okvira. Razlike postoje u 2-bitnom polju S, koje je definisano na sledeći način;

- **RR (Receive Ready): Spreman** za **prijem (00)** Kada uređaj treba da ukaže da je spreman i da može da primi informacije, on šalje RR okvir. Taj okvir se koristi i za periodičnu potvrdu primljenih okvira kada nema odlazećih podataka.

- **REJ (Reject): Odbacivanje (01)** Ovo je slično NAK potvrdama koje su predstavljene kod go-back-n protokola. Ovim poljem se od drugog uredaja zahteva slanje svih nerešenih okvira, počevši od onog čiji je broj defmisan sa N(R). Ovo može da se desi ako okvir stiže van redosleda, ili je oštećen.
- **RNR (Receive Not Ready): Nije spreman za prijem (10)** Ako su baferi uredaja skoro puni, ili ako je detektovana greška na njegovoj strani linka, uredaj može da zaustavi tok dolazećih okvira slanjem RNR okvira.
- **SREJ (Selective Reject): Selektivno odbacivanje (11)** Ovo je slično NAK potvrdama koje su predstavljene kod protokola selektivne retransmisije. Zahteva se da drugi uredaj pošalje okvir čiji je broj definisan sa N(R).

Dok informacioni i supervizorski okviri kontrolišu i upravljaju transfer okvira, *nenumerisani okviri* odreduju kako se protokol nastavlja. Na primer, ranije smo istakli da HDLC može da koristi go-back-n, ili selektivna retransmisija, da može da koristi okvire različitih veličina i da može da komunicira u jednom od tri moda. Kako uredaj odlučuje šta će da uradi?

Deo (c) na slici 9.7 prikazuje format nenumerisanog okvira. Dva polja označena sa M zajedno definišu pet flegova čije vrednosti definišu komunikacioni protokol. Prirnarna stanica šalje komande, a sekundardna reaguje na komande postavljanjem odgovarajućih flegova. U tabeli 9.1 date su neke moguće komande i odzivi koji se mogu kodirati u sklopu nenumerisanih okvira.

HDLC primer Sledeći primer opisuje proces uspostavljanja linka, razmenu okvira i poništavanje linka. Na slici 9.8 prikazana je moguća sekvencu razmene između dva uredaja, A (primarne stanice) i B (sekundame stanice). Pretpostavimo da se koristi go-back-n protokol. Vertikalne strelice označavaju vreme prolaska. Nakošene strelice izmedu njih ukazuju na slanje okvira i njihova odredišta. Tekst u izvoru strelice definiše sadržaj okvira. Tekst na kraju strelice definiše šta se dešava kada okvir stigne. Da bi se slika uprostila, nije naveden nikakav tekst u slučajevima kada je okvir prihvaćen bez greške.

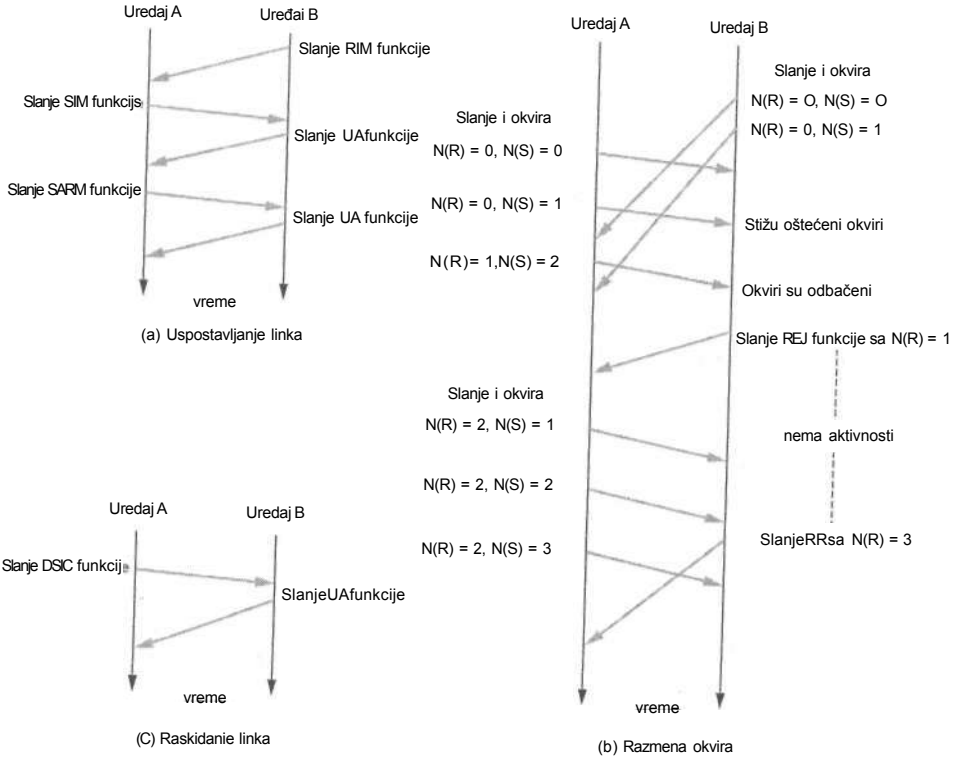
Za početak, na slici 9.8a prikazano je kako se konekcija može uspostaviti. Uredaj B startuje slanjem nenumerisanog okvira sa RIM funkcijom. Ovo je zahtev koji primarna stanica (A) šalje u vidu nenumerisanog okvira sa kodom funkcije SIM. Kada B primi SIM, počinje procedura inidjalizadje i potvrduje se prijem SIM-a slanjem drugog nenumerisanog okvira sa funkcijom UA. Kada A primi UA, zna da se B inicijalizuje. U tom slučaju, A odlučuje da se kao mod odziva koristi ARM i šalje drugi nenumerisani okvir sa tom funkcijom. Kada B primi okvir, ponovo šalje potvrdu slanjem drugog UA okvira. Kada A primi potvrdu, stanice su spremne za komunikaciju.

Na slici 9.8b prikazan je primer razmene okvira. Pošto je mod odziva ARM, A i B počinju da šalju informacione okvire (i okvire). B šalje svoja prva dva okvira sa brojevima N(S) = 0 i LU oba slučaja N(R) je 0. Pošto B još uvek nije ništa primio, očekuje prvi okvir (broj O). LI meduvremenu, A šalje svoja prva tri okvira, sa brojevima N(S) = 0, 1 i 2. U prva dva okvira N(R) = 0, jer A još uvek nije ništa primio.

Tabela 9.1: Funkcije HDLC nenumerisanog okvira

Funkcija	Značenje
SNRM: Set Normal Response Mode (C)	Komunikacija u modu normalnog odziva sa standardnim formatom okvira
SNRME: Set Normal Respons Mode Extended (C)	Komunikacija u modu normalnog odziva sa proširenim formatom okvira
SARM: Set Asynchronous Response Mode (C)	Komunikacija u modu asinhronog odziva sa standardnim formatom okvira
SARME: Set Asynchronous Response Mode Extended (C	Komunikacija u modu asinhronog odziva sa proširenim formatom okvira
SABM: Set Asynchronous Balanced Mode (C)	Komunikacija u asinhronom balansiranom modu sa standardnim formatom okvira
SABME: Set Asynchronous Balanced Mode Extended (C	Komunikacija u asinhronom balansiranom modu sa proširenim formatom okvira
DISC: Disconnect (C)	Inicira raskidanje veze izmedu dva uredaja. Raskidanje veze je kompletirano kada drugi uredaj reaguje sa UA funkcijom (videti dole).
RSET: Reset (C)	Svaki uredaj prati vrednosti N(R) i N(S) kako okviri prolaze. Ako dode do greške (recimo na višem nivou od HDLC), kontrola veze podataka može da reinicijalizuje razmenu okvira. RSET resetuje praćene vrednosti N(R) i N(S) na prethodno uspostavljenu vrednost.
SIM: Set Initialization Mode (	Nalaže drugom uredaju da inicijalizira svoje funkcije za kontrolu veze podataka.
UP: Unnumbered Poll (C)	Poziv (zahtev) za dobijanje informacija o statusu specifičnog uređaja
UI: Unnumbered Information (C, ili R)	Koristi se za slanje informacija o statusu. Obično se šalje iza UP, ili SIM.
XID: Exchange Identification (C, ili R)	Omogućava razmenu informacija o identifikaciji, ili statusu između dva uredaja.
RIM: Request Initialization Mode (R)	Zahtev sekundarne stanice primarnoj stanici da pošalje SIM
RD: Request Disconnect (R)	Zahtev sekundardne stanice primamoj stanici da inicira raskidanje veze slanjem DISC okvira
DM: Disconnect Mode (R)	Govori primamoj stanici da sekundardna stanica nije operabilna (nije priključena).
UA: Unnumbered Acknovvledgment (R)	Koristi se za potvrdu prethodno poslatih komandi, kao što su postavljanje moda, ili raskidanje konekcije.
TEST: Test (C, ili R)	Zahteva od drugog uredaja da pošalje test odziv. Uredaj koji šalje zahtev može da postavi nešto u polje podataka što bi prijemni uredaj vratio radi testiranja linka.
FRMR: Frame Reject (R)	Koristi se za ukazivanje da je pristigli okvir odbačen. REJ funkcija odbacuje oštećene okvire, ili okvire koji su primljeni van redosleda. FRMR se koristi, na primer, ako je kontrolno polje pogrešno definisano, ili ako Je stigla potvrda za okvir koji nikada nije poslat.

C = komanda; R = odziv



SLIKA 9.8 *Komuniđranje pomoću HDLC protokola*

Međutim, A prima okvir od B nakon što pošalje drugi okvir, Zato kod trećeg okvira A postavlja N(R) = 1, potvrdujući prijern okvira 0.

Zatim, pretpostavimo da drugi okvir koji A šalje stiže oštećen. B šalje supervizorski okvir sa kodom funkcije REJ i N(R) = 1, Ovaj okvir je zadužen za dve "stvari": potvrduje da je B primio okvir 0 od A i kaže da se došlo do greške i da A treba ponovo da pošalje sve, počevši od okvira 1. U međuvremenu, B i dalje očekuje okvir 1, tako da kada stigne sledeći okvir, B ga odbija, jer je van redosleda.

Eventualno, A prima REJ okvir i ponovo šalje okvire, počevši od naznačenog broja. Ako ima tri okvira za slanje, oni sadrže N(S) = 1, 2 i 3. Napomenimo da je u svakom od tih okvira N(R) sada 2 - dok smo prikazivali šta B radi sa oštećenim okvirom, A je primio još jedan (svoj drugi) i okvir. Ova tri okvira eventualno dopiru do B. Međutim, B je ušao u period neaktivnosti i ne može da prosledi nikakvu potvrdu. Izmedu dolaska okvira 1 i 2 njegov tajmer ističe.

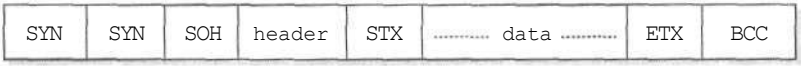
Šalje supervizorski okvir sa kodom funkcije RR, koja reafirmiše da je B i dalje spreman za prihvatanje okvira i potvrduje prijem okvira broj 2 (postavlja N(R) = 3). Korišćenje RR je ovde slično slušanju preko telefona ekscentrične tetke koja se žali na svoje susede. Držite telefonsku slušalicu na uvetu dok pravite kolače i povremeno izgovarate: "Da, tetka."

Ovakva razmena okvira se javlja sve dok obe stanice ne završe slanje. Stanica A donosi odluku da je vreme da raskine konekciju slanjem nenumerisanog olcvira sa kodom funkcije DISC (slika 9.8c). Kada B primi okvir, potvrđuje ga slanjem UA okvira. Kada A primi potvrdu, zna da su se obe strane složile da raskinu konekciju i okončaju link (ne spuštajte slušalicu bez tetkinog odobrenja; jednom ćete se naći u situaciji u kojoj sve zavisi od njene volje).

Binarny Synchronous Communications (BSC) protokol

Pre nego što završimo, predstavićemo ukratko **binarni** sinhroni komunikacioni **protokol,** koji se nekada označava kao **BSC,** ili bisync **protokol,** koga je razvio IBM. Koristi se kod sinhronih half-duplex komunikacija i koristi stop-and-wait kontrolu toka. Reč je o starom protokolu, ali predstavljamo ga da bismo mogli da napravimo poređenje sa bitovima orijentisanim protokolom.

Za razliku od prethodno predstavljenih protokola, BSC je *bajtovima orijentisan.* Naime, uredaj interpretira okvire kao sekvencu kontrolnih bajtova i bajtova sa podacima. Vrednosti bajtova mogu da se interpretiraju pomoću ASCII, ili EBCDIC skupova karaktera.* BSC koristi nekoliko različitih formata okvira. Na slici 9.9 prikazana su tri tipična okvira: format kontrolnog okvira i dva formata za okvire sa podacima.



(a) Netransparentni podaci

(b) Transparentni podaci

(c) Kontrolni okvir

SLIKA 9.9 *Format BSC okvira*

* Interesantno je napomenuti da BSC može da se koristi sa manje poznatim 6-bitnim kodom nazvanim Transcode (u stvari, možda i ne može).

U svakom slučaju, okvir počinje sa dva SYN karaktera. Oni omogućavaju primaocu okvira da podeli niz bitova na bajtove (da se naznači gde se jedan bajt završava, a drugi počinje).

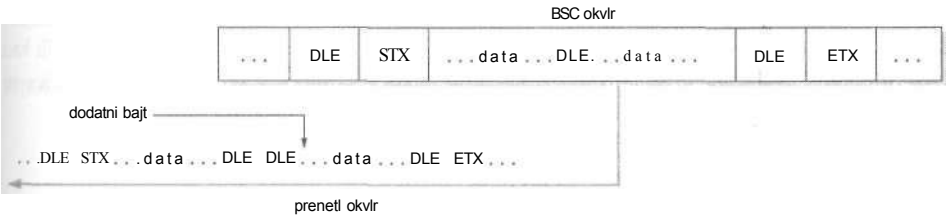
Iza SYN bajtova slede jedan, ili više kontrolnih bajtova. U kontrolnom okviru (slika 9.9c) kontrolne bajtove sačinjava najveći deo prenetih informacija. Kontrolne informacije su slične onome što smo prethodno naglasili. Kontrolni bajtovi mogu da obezbede potvrdu za ispravno primljene okvire, ili NAK za one koji su primljeni netačno, ili za postavljanje zahteva za odziv drugog uređaja.

U okviru sa podacima prvi kontrolni bajt je SOH (Start of Header početak zaglavlja). Govori prijemnom uredaju da sukcesivni bajtovi u pristiglom okviru sadrže informacije zaglavlja. Informacije zaglavlja mogu da budu različite, ali obično uključuju identitet, ili adresu uredaja pošiljaoca i primaoca. Na primer, identifikator odredišta je neophodan kada primarni uređaj šalje nešto preko multipoint linije, a identifikator prijemnog uredaja je neophodan kada primarni uredaj prima nešto od njega.

Iza informacija zaglavlja sledi STX (Start of Text — početak teksta) karakter (slika 9.9a), ili kombinacija DLE (Data Link Escape) i STX karaktera (slika 9.9b). U prvom slučaju STX ukazuje na početak teksta. To znači da sukcesivni bajtovi predstavljaju podatke. Medutim, pošto broj bajtova sa podacima može da bude različit, protokol mora da nađe načina da označi kraj bajtova podataka. To izvodi pomoću ETX (End of Text — kraj teksta) karaktera. Dakle, prijemni uredaj prima i prihvata bajtove kao bajtove podataka sve dok ne naide na ETX.

Kod aplikacija kod kojih podaci sadrže kodove karaktera koji se mogu štampati ovo je jednostavan način za ukazivanje njihovog kraja. Ali, šta je sa binarnim fajlovima čiji se podaci sastoje od nasumičnih uzoraka bitova? Sta se dešava ako se u podacima u okviru nade ETX karakter? Šta će sprečiti prijemni uredaj da ga interpretira kao kontrolni bajt i da se tako izgube preostali podaci? Uređaj koristeći BSC protokol ovo rešava postavljanjem STX karaktera zajedno sa DLE karakterom, koji se ponaša kao preklopnik. Kada prijemni uredaj vidi par DLE-STX, onesposobljava proveru kontrolnih bajtova, kao što je ETX. Osim toga, provera ostaje onesposobljena sve dok prijemni uredaj ne naide na sledeći DLE karakter. Kada se naiđe na sledeći DLE karakter, prijemni uredaj osposobljava dalju proveru postojanja ETX, ili STX karaktera.

Na prvi pogled deluje kao da još uvek nismo rešili problem. Sta ako u podacima postoji sekvenca koja je identična DLE karakteru? Protokol zaobilazi ovaj problem maskiranjem DLE karaktera u svojim podacima. Na slici 9.10 pokazano je kako zaobilazi.



SLIKA 9.10 *Popunjavanje bajta*

Uredaj koji šalje okvir proučava karaktere koji predstavljaju podatke. Svuda gde postoji DLE karakter umeće dodatni DLE karakter. Ovaj proces se naziva **popunjavanje bajta** (byte stuffing) i sličan je ranije prikazanom popunjavanju bitova; jednostavno, funkcioniše na drugom nivou. Kada prijemni uredaj naide na DLE, odmah traži sledeći. Ako nade sledeđ karakter, zna da je drugi lažan i prihvata prvi kao obične podatke. Ako ne pronađe, zna da DLE ne predstavlja podatke i osposobljava proveru kontrolnih karaktera. Podaci razgraničeni na ovakav način nazivaju se **transparentni podaci** - sadržaj polja Data je transparentan za prijemni uredaj. Podaci koji su razgraničeni samo sa STX i ETX karakterima predstavljaju *netransparentne podatke.*

Poslednji karakter iz formata predstavljenog na slici 9.9 je **BCC, Block Check Character.** Koristi se u sklopu BSC-ovog metoda za proveru grešaka. Sadržaj polja BCC zavisi od korišćenog metoda za proveru grešaka. Ako protokol koristi CRC proveru, BCC odgovara CRC-16 polinomu (videti odeljak 6.3). U drugim slučajevima BCC koristi **longitudinalnu proveru redundantnosti.** Ovaj metod vizuelizuje okvir kao dvodimenzionalni niz u kome svaki red predstavlja jedan bajt. Za svaku kolonu bit parnosti se odreduje na osnovu bitova iz te kolone i smešta se u polje BCC.

9.3 Ethernet: IEEE standard 802.3

Sada, kada znate kako dva uredaja mogu da razmenjuju okvire (nezavisno od medijuma na koji su povezani), sledeći logičan korak je prikaz tehnika za pristup medijumu (tj. MAC sloja). llbrzo nakon što su razvijene LAN mreže, IEEE je definisao tri formalna standarda: IEEE 802.3 za Ethernet, IEEE 802.4 za token bus i IEEE 802.5 za token ring.

Ethernet je bio dizajniran kao topologija magistrale kod koje su se uredaji "nadmetali" za segment, koristeći formu CSMA/CD protokola za "nadmetanje" (videti odeljak 4.7). Obično se koristio za povezivanje radnih stanica, servera, štampača i starih mainframe kompjutera. Deo istorije Etherneta datira još od 1973. godine. U okvim svoje doktorske teze, Robert Metcalfe je opisao najveći deo svojih istraživanja o LAN tehnologijama. Nakon odbrane teze, pridružio se Xerox Corporation i radio je u grupi koja je implementirala ono što je postalo poznato kao **Ethernet.** Ethernet je dobio naziv po **etru** (ether), imaginarnoj supstanci za koju se verovalo da okupira sav prostor i da Je to medijum pomoću koga se svetlosni talasi prostiru.

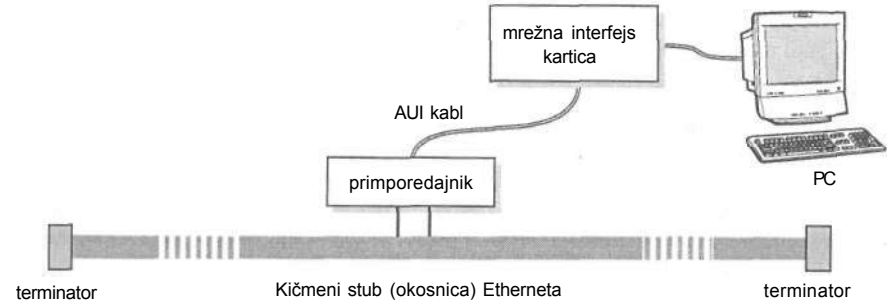
Kasnije su napisani koncepti Etherneta i predloženi su IEEE radi usvajanja za LAN mreže. Predlog je imao podršku Xeroxa, Intela i DEC-a. IEEE ga je usvojio kao standard i sada se karakteriše kao IEEE standard 802.3. Vredi napomenuti da su u to vreme u IEEE upućena još dva predloga. Jedan je podržo General Motors, a drugi IBM. S obzirom da su ovako uticajne organizacije predlagale odredene standarde, zvaničnici u IEEE su se dvoumili koji je od tri predloga najprikladniji kao LAN standard. Napravljen je kompromis i doneta su sva tri LAN standarda - druga dva su postala IEEE 802.4 (token bus) i IEEE 802.5 (token ring).

Većina današnjih LAN mreža je zasnovana na originalnim Ethernet idejama, a čitaoci će retko naići na token bus, ili token ring mreže. Zbog toga ćemo ovde predstaviti originalne Ethemet koncepte i različite nadogradnje, koje su bile uslovljene tehnološkim napretkom. Takode, zbog istorijskog značaja za ovu oblast i činjenice da predstavlja dramatično drugačiji prilaz povezivanju uređaja, kasnije ćemo predstaviti i token ring mrežu. Ipak, nećemo se baviti token bus standardom.

Koncepti

Iako je Ethernet namenjen za topologiju magistrale, u stvari postoji nekoliko načina za povezivanje uređaja. Na slici 9.11 pokazano je kako je originalno konfigurisana konekcija izmedu personalnog kompjutera i Ethernet segmenta (koaksijalnog kabla).* Iako koristimo primer personalnog kompjutera, treba da znate da se i drugi uredaji mogu povezivati pomoću koaksijalnog kabla. Elektronski **terminatori** su postavljeni na oba kraja kabla. Oni sprečavaju vraćanje eha signala kroz kabl, čime bi se stvorio lažni signal i došlo bi do konfuzije u komunikacijama. Efekat električnog eha je donekle sličan situaciji u kojoj se uključujete u neku radio emisiju i čujete sebe na radiju. Vaš glas obično malo kasni kako bi se onemogućilo da se nepristojne reči izgovore u toku žive emisije. Zato pričanje i slušanje svog zakasnelog glasa može da Vas dezorijentiše i da ne uspete da razaznate šta Vam domaćin emisije govori.

Personalni kompjuter je povezan na kabl pomoću dodatnog hardvera. Prvo, **primopredajnik** se spaja sa kablom pomoću konektora poznatog pod nazivom "vampire clarnp", uredaja sa pinom koji se uklapa u spoljašnji omotač kabla i uspostavlja kontakt sa jezgrom kabla. Primarna namena primopredajnika je da kreira interfejs između kompjutera i kabla. Jedna od njegovih funkcija je prenos bitova na kabl preko CSMA/CD protokola za "nadmetanje", koji omogućava utvrdivanje da li na medijumu postoji saobraćaj i da li je došlo do kolizije. Primopredajnik komunicira sa personalnim kompjuterom pomoću **primopredajnog kabla** (tranceiver cable). Neki ga nazivaju AtJI *(attachment unit interface) kabl.*



SLIKA 9.11 *Moguća Ethernet konekcija*

* Ovo se odnosi na debeli Ethernet, koji je poznatiji kao 10Base5 kabl. Uskoro ćemo predstaviti i ostale specifikacije kablova.

Kabl čini pet parova upredenih parica. Dva para se koriste za slanje podataka i kontrolnih informacija do kompjutera. Sledeća dva se koriste za prijem podataka i kontrolnih informacija. Peti par se koristi za povezivanje napajanja i za uzemljenje. Primopredajnik može da komunicira sa nekoliko uređaja preko multipleksera.

Kabl primopredajnika je povezan na personalni kompjuter preko mrežne interfejs kartice (NIC - network interface card), koja je instalirana na kompjuteru. NIC sadrži logička kola neophodna za baferovanje podataka i njihov prenos između kabla primopredajnika i memorije kompjutera. Vrši i proveru grešaka, kreira okvire, utvrđuje kada je potrebno ponoviti prenos nakon kolizija i prepoznaje okvire koji su namenjeni njenom kompjuteru. Ukratko, izvršava funkcije protokola za MAC sloj. Izvršavanjem tih zadataka rasterećuje procesorsku jedinicu kompjutera, tako da on može da se posveti svojim tipičnim aktivnostima.

Najpre ćemo navesti u nizu i opisati aktivnosti koje su neophodne da bi personalni kompjuter poslao podatke do drugog personalnog kompjutera. Uključeni su sledeći koraci:

1. Kompjuter pošiljalac izvršava mrežni softver koji u memoriju kompjutera postavlja informacije u formi paketa. Nakon toga, šalje signal NIC kartici preko interne magistrale da pajcet čeka na slanje.

2. NIC dobija»paket i kreira okvir u odgovarajućem formatu, smeštajući paket u polje Data novokreiranog okvira. Potom, čeka na signal od primopredajnika, koji nadgleda segment, čekajući šansu za slanje okvira.

3. Kada primopredajnik detekujte neaktivnost na kablu, šalje signal do NIC kartice, koja odmah šalje okvir do primopredajnika. Primopredajnik šalje bitove na kabl i osluškuje da li će doći do nekih kolizija. Ako ne dode do kolizije, pretpostavlja se da je prenos uspešno obavljen. Ako dode do kolizije, primopredajnik obaveštava NIC karticu. NIC kartica izvršava binarni eksponencijalni backoff algoritam, koji smo opisali u odeljku 4.7, kako bi utvrdila šta dalje da preduzme. Ukoliko se kolizije i dalje javljaju, šalje se signal mrežnom softveru, koji korisniku prikazuje poruku greške, ili izvršava neki algoritam kao odziv na grešku.

4. Primopredajnik na prijemnoj strani nadgleda saobraćaj preko kabla. Kopira okvire sa kabla usmerava ih ka NIC kartici na toj strani.

5. Nakon toga, NIC vrši CRC proveru grešaka. Ako nije bilo nikakvih grešaka, NIC proverava odredišnu adresu okvira. Ukoliko je okvir namenjen tom personalnom kompjuteru, NIC baferuje podatke iz okvira (paket) u memoriju i generiše prekid, obaveštavajući kompjuter da je paket stigao.

6. Personalni kompjuter izvršava mrežni softver i utvrduje da li paket može da se prihvati u skladu sa algoritmima kontrole toka, koje smo predstavili u Poglavlju 8. Ako može da se prihvati, kompjuter uzima paket iz memorije radi dalje obrade. Ukoliko ne može da se prihvati, mrežni softver reaguje u skladu sa protokolima sledećeg višeg sloja.

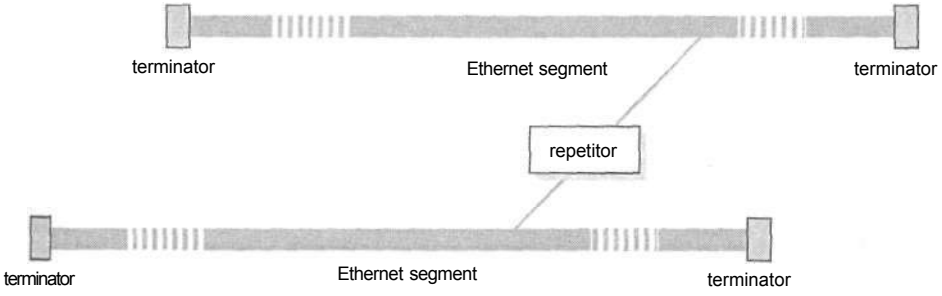
Svaki segment u okviru originalnog standarda imao je dužinu najviše 500 metara. Ovo ograničenje je bilo neophodno, zato što su se signali degradirali kako su se prostirali duž segmenta, a nakon 500 metara degradacija je bila značajna. Ali, šta se moglo uraditi ako je, na primer, ukupno rastojanje premašivalo tu granicu?

802.3 komitet je razmatrao taj problem **i** rešio ga tako što je dopušteno povezivanje više segmenata. Na slici 9.12 pokazan je jedan mogući način da se to postigne, korišćenjem repetitora (repeater), uredaja koji prihvata signal, regeneriše ga i prosleduje dalje. Regenerisanje omogućava prenos signala na većim udaljenostima. U opštem slučaju, originalni 802.3 standard jedopuštao povezivanje dva kompjutera sa najviše četiri repetitora.*

Format Ethernet okvira

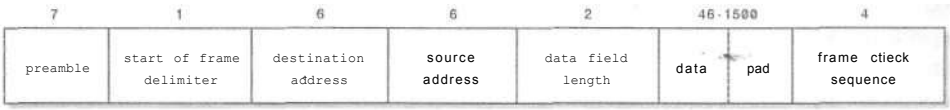
Na slici 9,13 prikazan je format Ethernet okvira. Kao i kod prethodnih formata okvira, nema nekih velikih iznenadenja. Sadrži uobičajene informacije:

- **Preamble (uvodni deo)** 7-bajtni uzorak koji se sastoji od nazimeničnih nula **i** jedinica **i** koristi se za sinhronizaciju. Sećate se da sinhronizacija uspostavlja učestalost semplovanja bitova. Ovo je slično vodećem vokalu u bendu koji odreduje tempo odbrojavanjern pre početka pesme.
- **Start of Frame Delimiter (delimiter za početak okvira)** Specijalni uzorak 10101011 koji ukazuje na početak okvira
- **Destination Address (odredišna adresa)** Ako je prvi bit 0, ovo polje definiše specifični uredaj. Ako je 1, odredišna adresa predstavlja grupnu adresu i okvir se šalje do svih uredaja u nekoj preddefmisanoj grupi određenoj adresom. Interfejs svakog uredaja zna svoju grupnu adresu i odaziva se kada je prepozna. Ako su svi bitovi jedinice, okvir se emituje do svih uredaja.
- **Source Address (izvorna adresa)** Definiše odakle okvir potiče.
- **Data Length Field (dužina polja sa podacima)** Definiše broj bajtova kombinacije polja Data i Pad.



SLIKA 9.12 *Povezivanje dva segmenta*

' Na mreži može da postoji veći broj segmenata. Ograničenje od četiri repetitora je primenjeno samo na putanje između dva uredaja.

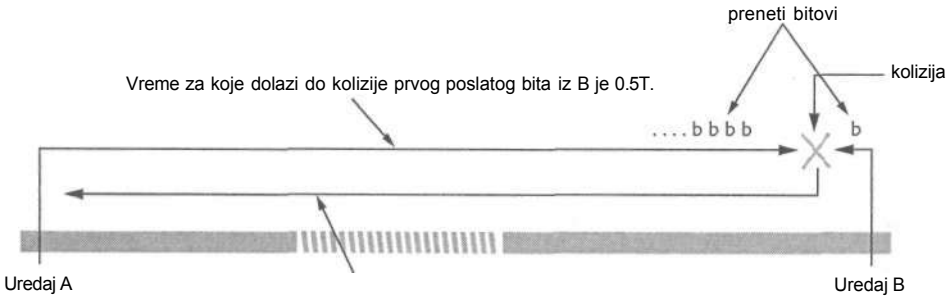


SLIKA 9.13 *Format Ethernet okvira*

- Data (podaci) Ovo polje je jasno samo po sebi.
- Pad (dopuna) Polje Data mora da ima najmanje 46 bajtova (uskoro ćemo reći nešto više o tome). Ako nema dovoljno podataka, dodaju se dodatni bajtovi (dopunjavaju) tako da se dostigne potrebni broj bajtova.
- Frame Check Sequence (provera okvira) Provera grešaka pomoću 32-bitnog CRC-a.

Na slici 9:13 možemo da vidimo gornju i donju granicu (od 46 do 1.500) bajtova za podatke i dopunu. Gomja granica se koristi kako bi se sprečili prenosi koji bi monopolizirali medijum na duže vreme. Donja granica postoji da bi se obezbedilo ispravno funkcionisanje tehnika za detekciju kolizije. Da biste videli šta je potrebno, razmotrite scenario sa slike 9.14. Uredaj A prenosi bitove na segment. Ti bitovi "putuju" preko segmenta i, neposredno pre nego što dopru do B, uredaj B započinje prenos svojih podataka. Dolazi do kolizije izmedu bitova iz A i B i šum koji nastaje zbog kolizije "putuje" nazad do A. A eventualno čuje da je došlo do kolizije i, ako i dalje šalje podatke, zna da su se njegovi okviri sudarili. Ako je A završio slanje svog okvira, onda ne zna da li su u koliziji učestvovali njegovi okviri. Ne može da zna da li je prenos bio uspešan. Ključno za ispravno funkcionisanje CSMA/CD protokola je da, u slučaju da dode do kolizije, pošiljalac može da je detektuje pre nego što završi slanje okvira.

Ovo znači da okvir mora da bude dovoljno dugačak tako da uredaj i dalje prenosi bitove kada se kolizija detektuje. Dakle, koliko je vremena potrebno da bi se kolizija detektovala?



SLIKA 9.14 *Maksimalno vreme za detektovanje kolizije*

Ovo vreme delimično zavisi od toga koliko je pošiljalac udaljen od mesta kolizije. Prethodno smo istakli da je svaki segment u originalnom standardu mogao da ima maksimalnu dužinu 500 metara i da su dva uređaja mogla da budu razdvojena najviše sa četiri repetitora. Tako je maksimalna udaljenost iznosila 2.500 metara. Osim toga, električni signali putuju preko bakamih vodova brzinom od oko 200 metara/|isec (mikrosekunde). To znači da je signal mogao da pređe 2.500 metara za oko 12,5 usec. Maksimalno vreme za vraćanje šuma do pošiljaoca iznosi još 12,5 usec. Dakle, u najgorem slučaju, uređaju je potrebno 25 frsec da detektuje koliziju.

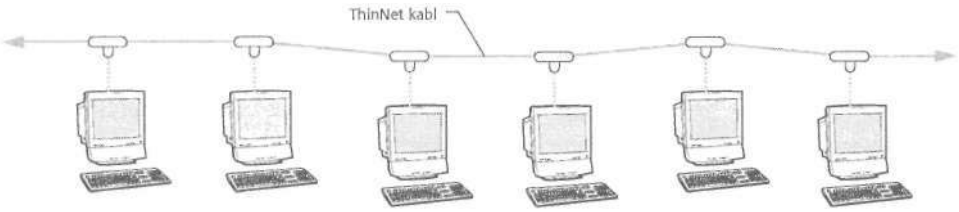
Medutim, postoji i jedan faktor koji komplikuje celu situaciju. Postoji kašnjenje kod svakog repetitora, jer se pre prosledivanja vrši regenerisanje bitova. Tačno kašnjenje zavisi od samog uređaja, ali dizajneri dopuštaju kašnjenje od po nekoliko ixsec za svaki repetitor. Dakle, osim 25 lisecza putovanje, mogu da postoje i četiri kašnjenja pre detekcije kolizije i još četiri kod bitova koji su nastaju kao rezultat kolizije. U najgorem slučaju, nakon što pošiljalac počne slanje, može da protekne 50 usec dok ne detektuje koliziju.

Ovo znači da svaki okvir treba da zahteva najmanje 50 usec za prenos. Sa brzinom prenosa od 10 Mbps, uredaj šalje 10 bitova svake nsec. U 50 iasec moguće je poslati 500 bitova. Dodavanjem nekih bitova, sigurnosti radi, minimalna veličina okvira je postavljena na 512 bitova, ili 64 bajta.

Fizičke implementacije 10 Mbps Etherneta

Nakon čitanja prethodnog teksta, možda ste se osvrnuli po laboratoriji, Oi učionici i rekli: "Osim po NIC karticama na personalnim kompjuterima, naše konekcije ne izgledaju tako. Gde su koaksijalni kablovi i repetitori?". Naravno, u pravu ste. Tehnološki napredak je promenio način povezivanja mreža. Mnoge originalne implementacije Etherneta su koristile 10Base5 kabl, 50-omski koaksijalni kabl prečnika 10 mm, koji je podržavao brzinu prenosa podataka od IO Mbps. Nazivan je i **Thick Wire Ethernet,** ili samo **ThickNet,** što je nedvosmisleno ukazivalo na debeli, nezgrapni kabl. Kablovi debljine 10 mm su se obično postavljali kroz podrume, ili ispod poda, ali su se teško savijali i vodili su se oko ćoškova, kroz plakare i druge tesne prostore. Zato nisu uvek mogli da se sprovedu blizu do uredaja koji je trebalo povezati. Zbog toga su postojale konfiguracije slične onoj sa slike 9.11.

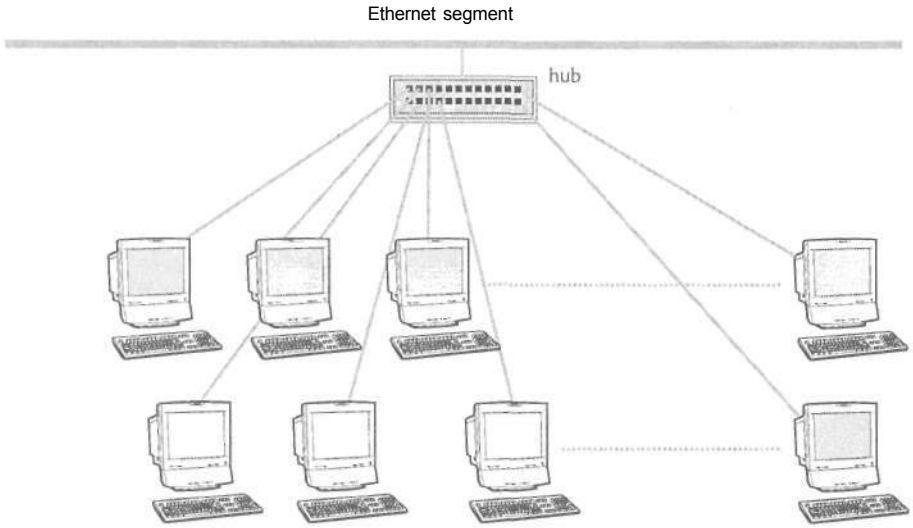
Da bi se omogućile jeftinije i fleksibilnije LAN mreže, prikladnije za personalne kompjutere, IEEE je kreirao modifikovani standard 802.3a, koji je koristio 10Base2 kablove. Kabl je podržavao prenos podataka na brzinama od 10 Mbps, ali je bio manjeg prečnika i lakše se savijao, što je bilo značajno za postavljanje kabla po ćoškovima i u ormarima. Pošto je bio jefitiniji, dobio je nadimak **Cheapernet,** mada je kao tipični naziv korišćen **Thin Wire Ethernet,** ili **ThinNet.** Dodatna fleksibilnost je omogućila drugačije konekcije u poredenju sa onima sa slike 9.11. Tanji, fleksibilniji 10Base2 kabl je mogao da se priključi direktno u kompjuter pomoću T-konektora (slika 9.15). Zahvaljujući ovome, logika primopredajnika je ugradena u NIC karticu koja je instalirana na personalnom kompjuteru. Ovaj metod je omogućio redukovanje cene LAN-a, jer je 10Base2 kabl mogao da se postavi do niza personalnih kompjutera, od kojih je svaki imao svoj T-konektor. Nedostatak 10Base2 kabla je bilo to što je zbog manje debljine imao veću električnu otpornost i nije mogao da se postavlja na velikim rastojanjima (oko 185 metara, što je malo u poredenju sa 500 metara kod 10Base5).



SLIKA 9.T 5 *ThinNet konekcije izvedene pomoću T-konektora*

Sa druge strane, minimalno rastojanje izmedu dva susedna uredaja za 10Base2 kabl je iznosilo oko pola metra, što je mnogo manje od minimalnih osam stopa izmedu primopredajnika povezanih 10Base5 kablom. Zato je 10Base2 prikladan za povezivanje više personalnih kompjutera u laboratorijama, jer se u jednoj prostoriji kompjuteri obično postavljaju veoma blizu jedni drugih.

Na slici 9.16 prikazana je sledeća konfiguracija koja je prerasla u IEEE standard 802.3i. Jedan uredaj nazvan hub (ponegde može da se sretne i naziv višeportni repetitor (multiport repeater)), ima više portova - na svaki od njih može da se priključi jedan uredaj preko lOBaseT kabla (kategorije 3, 4, ili 5 UTP), koji podržava brzine prenosa od IO Mbps. Hub je sa druge strane povezan i na Ethernet segment, iako je moguće povezati jedan hub na drugi, ili čak na komutator, ili ruter. Poslednja dva uredaja su zadužena za LAN konekcije; predstavićemo ih u par narednih poglavlja.



SLIKA 9.16 *Povezivanje personalnih kompjutera pomoću huba*

Primećujete da sa ovakvim pristupom više ne održavamo topologiju magistrale. Kako onda to utiče na protokole koje uredaji koriste? Možda je začuđujuće, ali ne utiče; uređaj i dalje može da izvršava protokole magistrale, što je defmitivno jako zgodno, jer mrežne kartice ne moraju da se redizajniraju. Hub se prvenstveno sastoji od elektronike koja regeneriše i mtira električne signale koje primi od jednog uredaja preko portova, do drugog uredaja sa kojim je povezan. Drugim rečima, sve što uredaj pošalje dostupno je svim ostalim uredajima, kao i kod magistrale. Osim toga, ako dva uređaja istovremeno šalju podatke, dolazi do kolizije,* takode kao kod magistrale. Kažemo da svi uredaji koji su povezani na isti hub imaju isti **domen kolizije.** Znači, bar sa stanovišta uredaja, nema nikakve razlike u odnosu na povezivanje na fizičku magistralu i moguće je koristiti protokol magistrale. Sa druge strane, magistrala može da se svede na jedan hub.

Ovakav tip konfiguracije ima nekoliko prednosti. Prvo, koristan je u zgradama gde fizička konfiguracija ne može da se postigne pomoću linearnih konekcija. Osim toga, na ovaj način je omogućena implementacija mreža uz iskorišćenje postojećih kablova, koji su često postavljeni u vreme izgradnje zgrade. Konačno, centralizovana kontrola komunikacija uprošćava dijagnostifikovanje i testiranje.

Sva tri standarda za kabliranje koriste električnu provodnost i Mančester kodiranje za digitalne signale. Optički standard je 802.3), koji definiše lOBaseF kabl, optički fiber kabl sa više modova rada, koji, kao i drugi, podržava brzinu prenosa od 10 Mbps. LI stvari, lOBaseF uključuje tri podskupa: lOBaseL (fiberlink), lOBaseFB (fiber backbone) i lOBaseFP (fiber passive). lOBaseFL povezuje uredaje na hub slično lOBaseT kablu, ali je njegova maksimalna dužina 2.000 metara. lOBaseFB se obično koristi za povezivanje hubova; i on ima maksimalnu dužinu 2.000 metara. lOBaseFP se takode koristi za povezivanje uredaja na hubove. Razlika u odnosu na lOBaseFL je to što se lOBaseFP koristi za manje instalacije i koristi pasivni sprežni element kao hub. Pasivni sprežni element nema eksterni izvor napajanja i ne može da poveže više od 33 uređaja. Osim toga, njegova maksimalna dužina iznosi 500 metara. Ovaj pristup nije mnogo uobičajen.

U tabeli 9.2 dat je pregled različitih strategija povezivanja za 10 Mbps Ethemet standard.

9.4 Fast Ethernet (100 Mbps)

Brzina prenosa od 10 Mbps ne samo da je bila uobičajena, već je smatrana i veoma velikom. Naravno, tehnologija stalno napreduje i standardi moraju da je prate. Samim tim se menjaju i potrebe korisnika. Brzina od 10 Mbps je bila zadovoljavajuća za većinu aplikacija koje su podrazumevale slanje emaila i razmenu manjih fajlova. Medutim, Web aplikacije, CAD (computer-assisted design) sistemi, veći fajlovi poput onih sa multimedijalnim sadržajem, baze podataka na serveru i sve veći broj korisnika počeli su da pomeraju granicu od 10 Mbps. Zato je uspostavljen novi standard koji je podržavao brzinu prenosa od 100 Mbps. Nazvan je **Fast Ethernet** (brzi Ethernet), što možda zvuči neadekvatno u današnje vreme, ali je tada, 1995. godine, predstavljalo ogromno poboljšanje. IEEE je dodao Fast Ethernet u svoju grupu 802.3 protokola i označio ga kao 802.3u.

* Ovo nije tačno kod nekih kasnijih tehnlogija, ali doći ćemo i do toga.

Tabela 9.2: Fizičke implementacije 10Mbps Etherneta

IEEE standard	Standardni kabl	Komentari
IEEE 802.3	10Base5	(ThickNet) 50-omski koaksijalni kabl, prečnika 10 mm. Maksimalna dužina segmenta iznosi 500 metara. Ostale granice su 4 repetitora, 5 segmenata i 100 uredaja po jednom segmentu. Rastojanje između primopredajnika mora da bude najmanje 8 stopa. Koristi se Mančester kodiranje. Primenjuje se kod fizičkih topologija magistrale.
IEEE 802.3a	10Base2	(CheaperNet, ThinNet) 50-omski koaksijalni kabl, prečnika 5 mm. Maksimalna dužina segmenta iznosi 185 metara. Dopuštena su najviše 4 repetitora, 5 segmenata i 30 uređaja po jednom segmentu. Minimalno rastojanje izmedu čvorova iznosi 0,45 metara. I ovaj standard koristi Mančester kodiranje. Primenjuje se kod fizičkih topologija magistrale.
IEEE 802.3i	lOBaseT	Kategorija 3, 4, ili 5 UTP kabla. Centralni hub. Mančester kodiranje. Maksimalna dužina UTP kabla je 100 metara. Koristi se kod fizičke topologije zvezde.
IEEE 802.3J	10BaseFx	Fiber sa više modova. Maksimalno rastojanje iznosi 200 metara za FL i FB, a 500 metara za FP. Koristi se kod fizičke topologije zvezde, ali FP koristi pasivni sprežni element kao hub. FP se ne sreće često.

Fast Ethernet odlikuje većina prethodno opisanih karakteristika. Formati Ethernet okvira i adresa su isti. Zato postoje ograničenja za maksimalnu i minimalnu veličinu okvira. Konačno, "nadmetanje" se i dalje zasniva na CSMA/CD. Drugim rečima, MAC sloj je isti.

Dakle, u čemu se ogledaju glavne razlike izmedu Fast Etherneta i 10 Mbps Etherneta? Jedna je očigledna - desetostruko povećanje bitske brzine. Ali, postoji još mnogo toga, uglavnom zbog činjenice da uvećanje bitske brzine za faktor 10 nije jednostavno. Možda izgleda lako razviti hardver koji generiše bitove sa 10 puta većom brzinom. Ipak, postoji problem. Medijum preko koga se bitovi prenose ima ograničenja.

10 Mbps Ethemet je dizajniran tako da koristi koaksijalni kabl i, kao što smo ranije istakli, bio je modifikovan za korišćenje UTP kablova. Sa druge strane, Fast Ethernet nije dizajniran za koaksijalne kablove. Svi njegovi standardi za kabliranje uključuju UTP, STP, ili optički fiber i dizajnirani su za fizičke topologije zvezde. Znači, razlike izmedu Fast Etherneta i 10 Mbps Etherneta ogledaju se prvenstveno u fizičkom sloju. Obradićemo sledeće Fast Ethernet standarde: 100BaseTX, 100BaseT4 i 100BaseFX.

100BaseTX

Počinjemo sa 100BaseTX, koji je dizajniran za korišćenje u kombinaciji sa kategorijom 5 UTP kablova. Kada je Fast Ethernet dizajniran za brzine od 100 Mbps preko UTP kabla kategorije 5, pojavio se jedan ozbiljan problem. Pošto je implementacija 10 Mbps Etherneta koristila Mančester kodiranje za digitalne podatke, izgledalo je razumno samo povećati frekvenciju tog kodiranja kako bi se uzela u obzir veća bitska brzina.

Međutim, to nije funkcionisalo zato što su visokofrekventni signali mnogo više podložni šumovima (EMI, elektromagnetnoj interferenci) kod kategorije 5 UTP kablova. Zato je bilo neophodno redukovati frekvenciju, što je opet stvaralo očigledan problem sa stanovišta prethodnog razmatranja povećanje bitske brzine. Jedna moguća opcija je ukljudvala korišćenje NRZ kodiranja. Pošto je signal bio fiksan za svaki bit (tj. nije bilo prenosa na sredini intervala), neophodna frekvencija je prepolovljena. Problem je bio mogud gubitak sinhronizacije ako je signal predstavljao dugački niz nula (videti odeljak 3.2).

Da bi bio rešen problem, lOBaseTX implementacija koristi šemu blokovskog kodiranja pod nazivom 4B/5B kodiranje. To je šema koja zamenjuje svaku polovinu bajta (tehnički termin je polubajt) sa pet bitova. Povećanje broja bitova deluje neuobičajeno i, u stvari, izgleda kao da će da pogorša problem, jer je sada neophodno preneti 25 odsto više bitova. Trik je u načinu implementacije.

1010 – 0010 – 0000 – 0000 – 0000 – 0000

Proučimo najpre kako kod funkcioniše. U tabeli 9.3 pokazano je kako se svaki polubajt menja sa odgovarajućih pet bitova. Uzimo sledeći niz polubajtova:

10110-10100-11110-11110-11110-11110

Primećujete šta se dešava. Dugački niz istog bita nije moguć. Bez obzira od kakvog niza polubajtova krenete, rezultujući 5-bitni blokovi uvek sadrže i nule i jedinice. U čemu se ogleda prednost ovakvog pristupa? Pošto više nemamo dugačke nizove istog bita, nema potrebe da se koristi Mančester kodiranje. Tako možemo da iskoristimo jednostavniju NRZI šemu kodiranja. Ipak, čak i sa NRZI kodiranjem i signalima niže frekvencije, postoje odredeni nivoi šuma kada se koriste UTP kablovi kategorije 5.

Da bi se dalje redukovali šumovi koji prate više frekvencije, razvijena je nova šema signaliziranja, pod nazivom **Multilevel Line Transmission Three Levels** (MLT-3).

Tabela 9.3: Kodiranje pomoću 4B/5B šeme

Bitovi podataka	5B kod	Bitovi podataka	**5Bkod**
0000	11110	1000	**10010**
0001	01001	1001	**10011**
0010	10100	1010	**10110**
0011	10101	1011	**10111**
0100	01010	1100	**11010**
0101	01011	1101	**11011**
0110	01110	1110	**11100**
0111	01111	1111	**11101**

Odstupa od prethodnih šema kodiranja na jedan značajan način: umesto da koristi signal sa dva stanja (predstavljena visokim i niskim nivoom, ili -1 i 1), MLT-3 definiše signal sa tri stanja. Ta stanja označavamo sa -1, 0 i 1. U opštem slučaju, MLT-3 signal kruži kroz stanja sledećim redosledom: -1, 0, +1, 0, -1, 0, +1, 0, -1 i tako redom. U stvari, ovo je slično vrednostima sinusne funkcije koje se ponavljaju u pravilnim intervalima.

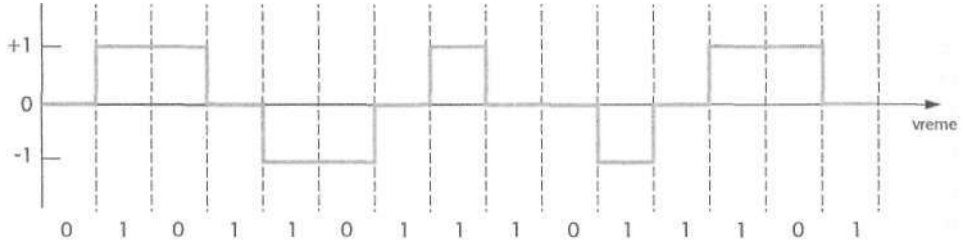
Svaki bit se u okviru MLT-3 šeme predstavlja na sledeći način:

- Ako je vrednost bita 1, MLT-3 signal prelazi na sledeće stanje u nizu.
- Ako je vrednost bita 0, MLT-3 signal zadržava trenutno stanje.

Na slici 9.17 prikazan je primer niza bitova i odgovarajućeg MLT-3 signala. Pretpostavljamo da signal startuje od 0. Pošto je prvi bit 0, signal ostaje u trenutnom stanju za jedan interval. Drugi bit je 1 i zato signal prelazi u sledeće stanje +1. Treći bit je 0 i signal ostaje tu gde jeste, na +1. Četvrti bit je 1 i signal prelazi u sledeće stanje 0. Peti bit je 1; signal se menja i prelazi na -I. Ovaj uzorak se ponavlja za sve bitove u nizu. Za svaku jedinicu signal prelazi u naredno stanje; za svaku nulu signal ostaje isti.

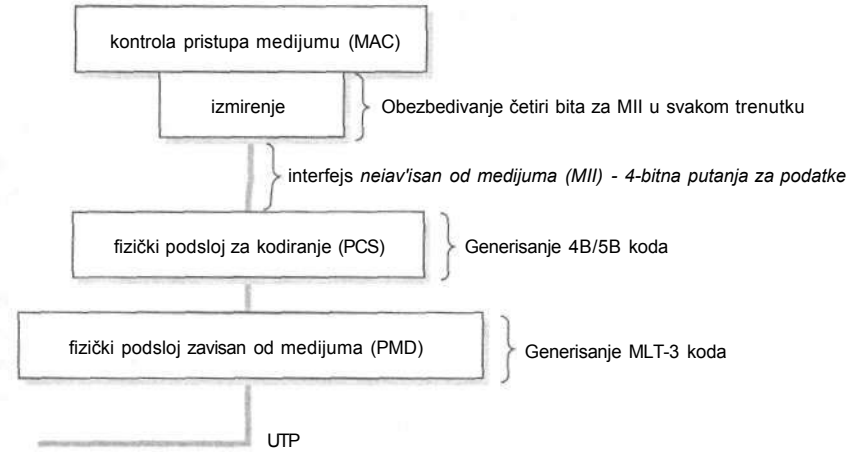
Kako ovo može da pomogne? Dobro osmotrite sliku 9.17. Potrebna su najmanje četiri intervala da MLT-3 signal prode jedan kompletan ciklus (-1 -» 0 -» +1 -> 0 -» -1). Poredenja radi, Mančester kod može da prode kompletan ciklus (nizak -» visok -» nizak) za jedan interval. To znači da MLT-3 kodiranje zahteva oko 25 odsto maksimalne frekvencije Mančester kodiranja. Generiše mnogo manji šum i funkcioniše sasvim zadovoljavajuće sa UTP kablovima kategorije 5.

Na slici 9.18 prikazana je delimična slojevita struktura Fast Ethernet protokola.* Karakteristike koje smo do sada opisali nalaze se ispod MAC sloja. **Podsloj izmirenja** (reconciliation sublayer) uzima po četiri bita u jednom trenutku i šalje ih do **fizičkog podsloja za kodiranje** (PCS-physical coding sublayer) preko **interfejsa nezavisnog od medijuma** (Mll-medium independent interface). MII defmiše konekciju izmedu NIC kartice i primopredajnika; ako je primopredajnik eksterni, zahteva kabl sa 40-pinskim konektorom.



SLIKA 9.17 *Multilevel Line Transmission-Three Levels (MLT-3)*

* Izostavili smo neke komponente, ali potpuno razumevanje svih komponenata zahteva poznavanje elektronskih i signalnih karakteristika koje nisu predmet razmatranja u ovoj knjizi. U referenci [Fo03] možete da pronadete više detalja.



SLIKA 9.18 *1 OOBaseTX fizički podslojevi*

Ako je primopredajnik ugraden u NIC karticu (što je obično slučaj), neophodan je MII kabl. U svakom slučaju, MII obezbeduje putanju podataka širine četiri bita. PCS vrši blokovsko kodiranje, menjajući svaki polubajt sa 5-bitnim kodom. Nakon toga, 5-bitni kod se šalje do fizičkog **podsloja zavisnog od medijuma** (PMD - physical medium dependent), na kome se generiše MLT-3 signal, koji se, zatim, prenosi na medijum. Sledeća karakteristika po kojoj se ast Ethernet razlikuje od prethodnih verzija Etherneta je činjenica da može da se pokreće u half-dulpex, ili full-duplex modu.* Pošto je originalni Ethernet koristio koaksijalni kabl i osnovno Mančester kodiranje, u jednom trenutku je bio moguć samo jedan prenos, inače je dolazilo do kolizije. Zato su uređaji morali da koriste half-duplex mod. To se promenilo sa primenom UTP kablova. Kablovi kategorije 5 obično sadrže četiri para upredenih parica, a lOOBaseTX ima dva; jedan za slanje, a drugi za prijem i detektovanje kolizija. Kada se uredaji povezuju pomoću lOObaseTX huba, koriste half-duplex mod. To je zbog toga što postoji zaseban par za slanje i za prijem, tako da je i dalje moguća kolizija izmedu dva prenosa na habu. Ipak, uredaji mogu da koriste full-duplex mod (istovremeno slanje i prijem) ako su povezani preko komutatora (switch). Komutatori su uređaji sloja 2 i koriste se za razdvajanje domena kolizije; odložićemo razmatranje o komutatorima i full-duplex operacijama do sledećeg poglavlja, u kome se bavimo komutiranirn Ethemetom.

lOOBaseFX

Naravno, Fast Ethernet sa kablovima kategorije 5 nije jedina opcija. Sledeća opcija je fiber i lOOBaseFX defmiše Fast Ethernet preko optičkog fibera sa više modova. Prednost fibera u odnosu na bakarne vodove je dužina segmenta.

' Ovo važi i za lOBaseT.

lOOBaseTX ima maksimalnu dužinu 100 metara, dok 100BaseFX linkovi mogu da budu dugački i do dva kilometra ako se koristi full-duplex mod. Kod implementacije su uključena dva fibera: jedan za prenos, a drugi za prijem i detekciju kolizije.

Postavka fizičkih slojeva za 100BaseFX je slična onoj za lOOBaseTX. Koristi 4B/5B blokovski dekoder za zamenu svakog polubajta sa pet bitova. Medutim, pošto fiber nema frekventna ograničenja kao upredene parice, ne zahteva fazu MLT-3 kodiranja. Umesto toga, koristi NRZI.

100BaseT4

Recimo da su u Vašoj organizaciji čelnici rešili da zamene 10 Mbps Ethernet sa Fast Ethernetom. Sada je za to neophodno samo instalirati uredaje i postaviti kablove kategorije 5 izmedu zidova, ispod podova, iznad plafonskih ploča, kroz ispuste i ko zna kuda još. Cesto je teško i skupo zameniti postojeće kabliranje novim. Da li to znači da ste, osim ako ne instalirate nove vodove kategorije 5, "osuđeni" na brzinu od 10 Mbps? Srećom, ne.

Specifikacija 100BaseT4 je dizajnirana za kablove kategorije 3. Ovo je delimično zgodno za zgrade koje su pravljene sa starijim vodovima kategorije 3. Zamene nisu neophodne. Naravno, nameće se pitanje kako da ih nateramo da rade. Frekventna ograničenja kategorije 5 su zahtevala neka inovativna rešenja da bi se postigle brzine od 100 Mbps. Ograničenja kategorije 3 su još ozbiljnija. Korišćenje 4B/5B dekodera, iza koga sledi MLT-3 šema kodiranja, može da posluži kod kategorije 5, ali stvara isuviše veliki šum kod vodova kategorije 3. Zato je neophodan potpuno drugačiji pristup.

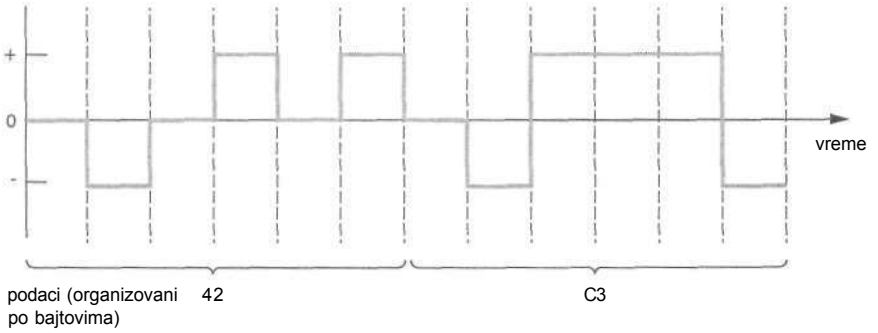
Kao i lOOBaseTX, 100BaseT4 koristi šemu kodiranja sa tri nivoa. Medutim, ovde je primenjen sasvim drugačiji pristup. Umesto da se koristi blokovsko dekodiranje, kao U slučaju 4B/5B, koristi se 8B/6T šema kodiranja. Ova šema pridružuje vrednost svakog bajta (osam bitova) jedinstvenom stringu od šest ternamih vrednosti, nazvanih tritovi (doslovni prevod bio bio mrve). Simbolički, svaka od tih vrednosti se označava kao +,Oi-. Možda izgleda čudno menjati osam vrednosti sa šest, ali imajte na umu da se osam bitova menja sa šest tritova. Postoji 2* = 256 mogućih 8-bitnih stringova i 3^6 = 729 moguclh 6-tritskih stringova. Dakle, asocijacije koriste samo nešto više od trećine mogućih tritskih stringova.

U tabeli 9.4 dati su neki tritski kodovi za manje vrednosti bajta. Kompletna tabela ima 256 slogova i ovde stvarno nema potrebe da se svi predstavljaju. Ako ste zainteresovani, u referenci [HaOl] možete da pronađete kompletnu tabelu.

Tabela 9.4 Delimična tabela za 8B/6T kodiranje

Bajt	Trit kod	Bajt	Trit kod	Bait	Trit kod	Bait	Trit kod
00	-+00-+	40	-00+0+	80	-00+-+	CO	-+0+-+
01	0-+-+0	41	0-00++	81	0-0-++	CI	0-+-++
02	0-+0-+	42	0-0+0+	82	0-0+-+	C2	0-++-+
03	0-++0-	43	0-0++0	83	0-0+-+-	C3	0-+++-

Na slici 9.19 prikazan je signal kome su dodeljene dve sukcesivne vrednosti bajtova.

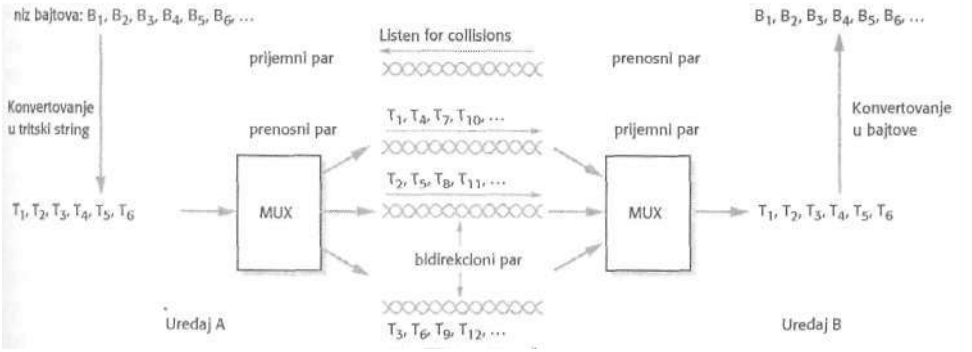


SLIKA 9.19 *8B/6T kodiranje*

Primećujete da je 16 bitova preneto u 12 sukcesivnih vremenskih intervala (po jedan za svaki trit), što daje smanjenje od 25 odsto u odnosu na slanje bitova sa NRZI kodiranjem. Pošto nam je potrebna brzina prenosa od 100 Mbps, signal mora da ima kapacitet promene na brzini bauda od 75 miliona. Na žalost, ovo i dalje zahteva frekvenciju koju kablovi kategorije 3 ne mogu da postignu.

Ipak, kao i kablovi kategorije 5, kablovi kategorije 3 obično imaju četiri para upredenih parica. 100BaseT4 funkcioniše tako što se podaci prenose preko tri para istovremeno. Specifikacija 100BaseT4 nalaže da svaki uređaj koristi jedan par za prenos, a drugi za prijem i detekciju kolizija. Druga dva para koriste se kao bidirekcioni medijum. Drugim rečima, kada uređaj A šalje podatke do uredaja B, za to koristi par za prenos i dva bidirekciona para. Ako uređaj B šalje podatke, on koristi svoj par za prenos (prijemni par za uredaj A) i dva bidirekciona para.

Na slici 9.20 ilustrovano je kako to funkcioniše. Uredaj A mora da pošalje sledeću sekvencu bajtova: B_1, B_2, B_3, B_4 i tako redom.



SLIKA 9.20 *Slanje podataka na 100BaseT4 preko četiri para upredenih parica*

Harvder uredaja A konvertuje svaki bajt u 6-tritski string, stvarajući niz tritova T_1, T_2, T_3, T_4 i tako redom koji se uvodi u multiplekser. Multiplekser ima tri izlaza. Stringovi T_1, T_4, i T_7 i tako dalje "putuju" duž prenosnog para uređaja A (koji je ukršten sa prijemnim parom uređaja B). Stringovi T_2, T_5, T_8, i tako dalje "putuju" preko dva bidirekciona para. Ako B šalje podatke, koristio bi svoj prenosni par i iste bidirekcione parove.

Pošto se prenos javlja paralelno, dovoljna je manja bitska brzina za svaki par. Samim tim, koristi se manja frekvencija za svaki par. Pretpostavimo da signal na svakom od tri para ima kapacitet promene na brzini bauda od 25 miliona. To daje brzinu od 75 miliona tritova u sekundi. Pošto se šest tritova proširuje u osam bitova, efektivna brzina prenosa podataka je 100 Mbps. Najznačajnije je to što se frekvencija neophodna za brzinu bauda od 25 miliona nalazi u granicama šuma za kategoriju 3 UTP kablova.

Nedostatak 100BaseT4 specifikacije je to što ne može da funkcioniše u full-duplex modu, zbog tipa signala koji se koriste na fizičkom sloju i činjenice da se za prenos podataka koriste tri od četiri para upredenih parica. Znači, za full-duplex mod bilo bi neophodno obezbediti najmanje šest parova, ili koristiti neku novu tehniku.

Domen kolizije

Poslednji problem kojim ćemo se baviti je domen kolizije. I 10 Mbps i Fast Ethernet održavaju minimalnu veličinu okvira od 512 bitova. Medutim, Fast Ethernet prenosi jedan okvir minimalne veličine za 5,12 irsec, 10 puta brže od 10 Mbps Etherneta. Ne zaboravite da je signalima i dalje potrebno vreme da pređu medijum i da CSMA/CD protokol zahteva da uredaj i dalje bude spreman za prenos ako dode do kolizije. To je problem za Fast Ethernet ako kolizija zahteva skoro 50 nsec za detektovanje, jer je uređaj mogao odavno da se zaustavi kada se kolizija detektuje. Zato domen kolizije mora značajno da se redukuje. Standard zahteva samo jedan, ili dva huba, u zavisnosti od toga da li je hub repetitor klase I, ili II. Repetitor klase II ima manje kašnjenje i brže prosleduje bitove. Međutim, on podržava samo jedan tip signala - ne može da poveže lOOBaseTX i 100BaseT4 kabl. Dva repetitora klase II mogu da se nalaze u istom domenu kolizije sve dok je link izmedu njih dugačak pet metara, ili kraći. Repetitor klase I, iako sporiji, može da povezuje segmente različitih tipova. Međutim, standard definiše samo jedan repetitor klase I u domenu kolizije.

U tabeli 9.5 možete da vidite konačni pregled tri specifikacije Fast Etherneta.

9.5 Gigabit Ethernet

Sledeći veliki korak u razvoju Etherneta bili su novo desetostruko povećanje bitske brzine i razvoj **Gigabit Ethernet** standarda. Kao što naslućujete na osnovu naziva, podržane su brzine od 1.000 Mbps (1 Gbps). Motivacioni faktori za razvoj Gigabit Etherneta su bili slični onima za razvoj Fast Etherneta: sve veći broj velikih fajlova, više multimedijalnih aplikacija (posebno real-time aplikacije), veći broj korisnika, sofisticiranije Web aplikacije i tako dalje. Sve veće potrebe korisnika su zahtevale porast i razvoj novih tehnologija.

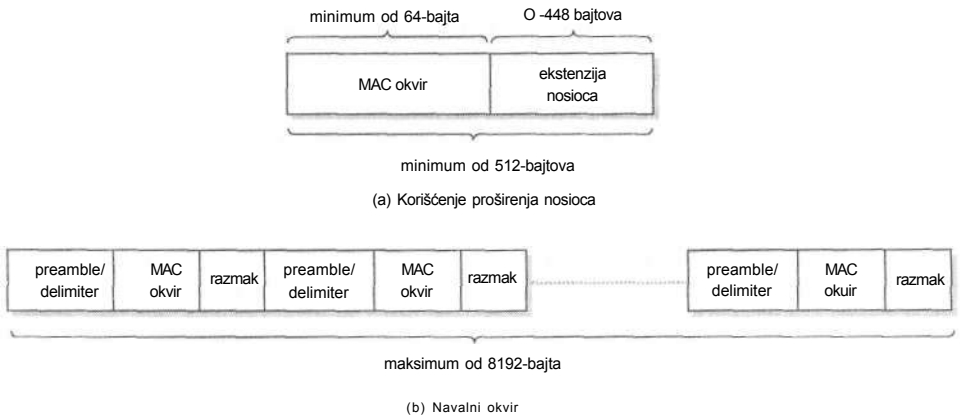
Tabela 9.5: Fizičke implementacije IEEE 802.3u 100 Mbps Fast Ethernet standarda

Specifikacija kabla	Komentari
100BaseTX	Koristi dva para upredenih parica kod kablova kategorije 5 i 4B/5B, iza kog sledi MLT-3 kodiranje. U domenu kolizije se koriste maksimalno dva repetitora klase II (ako se nalaze na rastojanju od pet metara, ili manje), ili jedan repetitor klase I. Maksimalna dužina segmenta je 100 metara.
100BaseFX	Optički fiber sa više modova rada. Koristi 4B/5B i NRZI kodiranje. U domenu kolizije koriste se maksimalno dva repetitora klase II, ili jedan repetitor klase I. Maksimalna dužina segmenta iznosi 136 metara ako su oba linka u istom domenu kolizije izvedena pomoću fibera, a 160 metara ako je drugi link UTP. Ako fiber povezuje dva komutatora, dužina segmenta može da bude 412 metara (half duplex), ili 2.000 metara (full duplex).
100BaseT4	Koristi četiri para upredenih parica kod kablova kategorije 3. Koristi se 8B/6T kodiranje i tritovi se prenose istovremeno preko tri para upredenih parica. Izmedu uredaja je moguće postaviti najviše dva repetitora klase II, ili jedan klase I. Ne može da se koristi u full-duplex modu. Maksimalna dužina segmenta iznosi 100 metara.

MAC podsloj

Gigabit Ethernet je dizajniran za korišćenje sa dva optička fibera (lOOOBaseSX i lOOOBaseLX) i preko bakarnog voda (lOOOBaseT i lOOOBaseCX). Ubrzo ćemo opisati fizičke slojeve, ali najpre ćemo se fokusirati na značajni problem MAC sloja koji nije predstavljen kod Fast Ethernet standarda. Gigabit Ethernet može da funkcioniše i u half-duplex i u full-duplex modu. U full-duplex modu nema kolizija i zato je CSMA/CD onesposobljen. Medutim, kolizije su moguće u half-duplex modu,i zato je domen kolizije problematičan. Videli smo kod Fast Etherneta da je uredaj mogao na 100 Mbps da prenese okvir minimalne velidne za 5,12 μsec. Zato je veličina domena kolizije smanjena na oko 200 metara za UTP kabl kategorije 5. Na gigabitskim brzinama uredaj prenosi okvir minimalne veličine za 0,512 μsec. Za UTP kablove kategorije 5 to redukuje veličinu domena kolizije na oko 25 metara, što ne predstavlja veliku oblast u kojoj bi se mreža formirala.

Ovo je pravi problem i postoje tri načina da se reši. Prvi je da se kolizije u potpunosti eliminišu tako da minimalna veličlna okvira ne bude definisana. To može da se postigne pomoću komutatora; predstavićemo ih u sledećem poglavlju. Druga opcija je povećanje minimalne veličine okvira. Iako se u mnogim implementacijama Gigabit Etherneta prešlo na full-duplex okruženja bez kolizija, standard i dalje podržava CSMA/CD sa povećanjem minimalne veličine okvira sa 512 na 4.096 bitova. Ovo je osmostruko povećanje i veličine okvira i vremena potrebnog za njegov prenos. Ipak, na ovaj način je omogućena dužina segmenta od 100 metara, slično Fast Ethernet specifikacijama. U full-duplex modu MAC okvir je isti kao i kod prethodnih standarda. U half-duplex modu slika 9.21a pokazuje kako standard povećava veličinu okvira, koristeći *ekstenziju nosioca* (carrier extension).



SLIKA 9.21 *Gigabit Ethernet okviri*

Na slici 9.13 je prikazan standardni Ethernet okvir. Ako je za podatke potrebno manje od 46 bajtova, polje Data se dopunjava tako da se postigne originalni minimum od 64 bajta. U ovom slučaju ekstenzija nosioca sadrži 448 bajtova neophodnih za postizanje minimuma od 512 bajtova. Ako je za podatke potrebno više od 46 bajtova, onda se ekstenzija nosioca smanjuje za po 1 bajt za svaki dodatni bajt podataka. Ukoliko su za podatke potrebna 494 bajta, ili više, onda okvir ispunjava zahteve za minimalnom dužinom i nema potrebe za ekstenzijom nosioca.

Ovo je sasvim opravdan pristup ako je potrebno poslati velike količine podataka. Medutim, ako većina okvira sadrži manju količinu podataka, ekstenzija nosioca je samo još jedna dodatna dopuna i nije efikasna. Na kraju krajeva, kreiranje okvira od 512 bajtova samo da bi se prenelo nekoliko podataka nije isplativo. To bi bilo isto kao da sa cisternom idete do prodavnice na ćošku da biste sipali u nju dva litra mleka. Da bi standard efikasnije funkcionisao, Gigabit Etherner koristi *sporadično slanje okvira* (frame bursting) (slika 9.21b). Ideja je da se prenese više okvira bez potrebe da svaki od njih prolazi kroz protokol za "nadmetanje". Standard to postiže kreiranjem *navalnog okvira* fburst frame), koji se sastoji iz jednog MAC okvira (uključujući ekstenziju nosioca), iza koga sledi niz dodatnih MAC okvira. Svi ostali MAC okviri u nizu nemaju ekstenziju nosioca i svaki par sukcesivnih okvira razdvaja 96-bitni razmak.

Uredaj prolazi kroz uobičajeni CSMA protokol i, kada dobije pristup medijumu, prenosi navalni okvir. Prvi ugrađeni MAC okvir ima ekstenziju nosioca, tako da će uredaj detektovati sve kolizije pre slanja drugog ugradenog MAC okvira. Ako ne dode do kolizije, uredaj prenosi preostale MAC okvire i razmake ugradene u navalni okvir bez prolaska kroz protokol "nadmetanja" za pristup medijumu za svaki pojedini okvir. Razmaci izmedu okvira sprečavaju oslobadanje medijuma; svi ostali uredaji koji se "nadmeću" za pristup medijumu vide da je zauzet sve dok prenosi navalni okvir.

Kada se Gigabit Ethernet pokreće u full-duplex modu, minimalna veličina okvira i navalni okviri nisu neophodni, jer ne dolazi do kolizija. U tom slučaju, Gigabit Ethernet koristi isti format MAC okvira koji koriste i prethodne verzije.

IOOOBaseX

Prva Gigabit Ethernet specifikacija je bila opisana u IOOOBaseX, 802.3z standardu. IOOOBaseX standard, u stvari, uključuje tri zasebna medijuma: kratkotalasni optički fiber (1000BaseSX), dugotalasni optički fiber (1000BaseLX) i zaštićeni bakarni kabl (1000BaseCX). Bakarni kabl ima drugačije električne karakteristike u poredenju sa UTP kablovima kategorije 5 i omogućava velike bitske brzine na rastojanjima do 25 metara. Obično se koristi za povezivanje centralizovanih uređaja na ispitnim mestima.

Razlika izmedu optičkih fibera sa kratkim i dugim talasima ogleda se u frekvenciji svetlosti koja se prenosi kroz fiber. Kratki talasi koriste signale sa višim frekvencijama na fiberu koji funkcioniše u više modova. Dugački talasi dopuštaju korišćenje fibera sa više modova, ili u jednom modu. Kratkotalasni laseri su slični onima koji se koriste za CD tehnologije. Jeftiniji su, ali signal brže slabi, što znači da su dopuštena kraća rastojanja. Dugotalasni laseri su skuplji, ali je integritet signala sačuvan i na većim udaljenostima.

Fizički sloj Gigabit Etherneta za IOOOBaseX podseća na onaj sa slike 9.18. Medutim, umesto interfejsa nezavisnog od medijuma, sa putanjom širine četiri bita, ovde se koristi interfejs zavisan od Gigabit medijuma (GMII - Gigabit medium independent interface), sa putanjom širine osam bitova. Sloj izmirenja obezbeđuje jedan bit podataka na svakoj putanji na svakih osam nanosekundi za fizički podsloj za kodiranje (PCS). Svaka 8-bitna putanja može da smesti 125 Mbps; kombinovana brzina prenosa iznosi 1 Gbps.

Kao i PCS kod Fast Etherneta, PCS za Gigabit Ethernet koristi šemu blokovskog kodiranja. Ipak, ovde se koristi 8B/10B šema kodiranja, koja menja osam bitova podataka sa IO bitova koda. Svaki 10-bitni kod ima ili četiri, pet, ili šest nula (i naravno, šest, pet, ili četiri jedinice). Razlog je isti kao i ranije; da bi se sprečili dugački nizovi nula, ili jedinica. 8B/10B šema je konceptualno slična 4B/5B šemi; medutim, obezbeđuje bolji *DC balans*. To znači da, kada se string proširi, rezultat uključuje sličan broj nula i jedinica i sličnu učestalost prelaza sa O na 1 i obratno.

Postoji nekoliko prednosti ove specifikacije. Prva je sinhronizacija koju smo ranije prikazali. Druga je to što postoji više 10-bitnih kodova nego 8-bitnih podataka. Zato se neki kodovi mogu koristiti za kontrolne funkcije. Sledeća prednost je što, kada se koristi u optičkim fiber komunikadjama, balans redukuje zagrevanje lasera zavisno od podataka. Ovo je značajno zato što ovakvo zagrevanje može da poveća učestalost pojavljivanja grešaka. Ipak, napomenimo da sam 8B/10B kod ne garantuje podjednak broj nula i jedinica u dugačkim stringovima - jedino garantuje da učestalost pojavljivanja nula, ili jedinda neće preći 60 odsto (jer postoji najviše šest nula, ili jedinica u bilo kom kodu). Da bi se DC balans dalje poboljšao, *PCS izračunava disparitet*. Kako se bitovi generišu, PCS prati da li je generisana nula, ili jedinica. Disparitet je pozitivan ako ima više jedinica, a negativan u suprotnom slučaju. Zatim, PCS konvertuje svaki 8-bitni bajt u 10-bitni kod koji zavisi ne samo od bajta, već i od tekuće vrednosti dispariteta. Korišćenjem dva moguća koda umesto jednog, PCS može da generiše približno jednak broj nula i jedinica. Tako se redukuje zagrevanje lasera zavisno od podataka i povećava se efikasnost lasera. Iako koncept konvertovanja osam bitova u 10 na prvi pogled deluje jednostavno, specifični kodovi su izuzetno složeni.

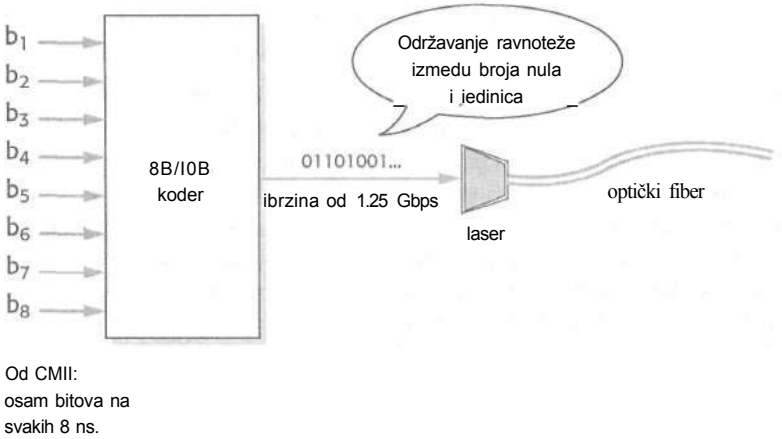
Kompletan proces je predstavljen na slici 9.22. 8B/10B koder prima osam bitova brzinom od 1 Gbps i kodira svaku grupu od osam bitova u 10-bitni kod. Nakon toga, laser prenosi sekvencu 10-bitnih kodova preko optičkog fibera na brzini od 1,25 Gbps.

lOOOBaseT

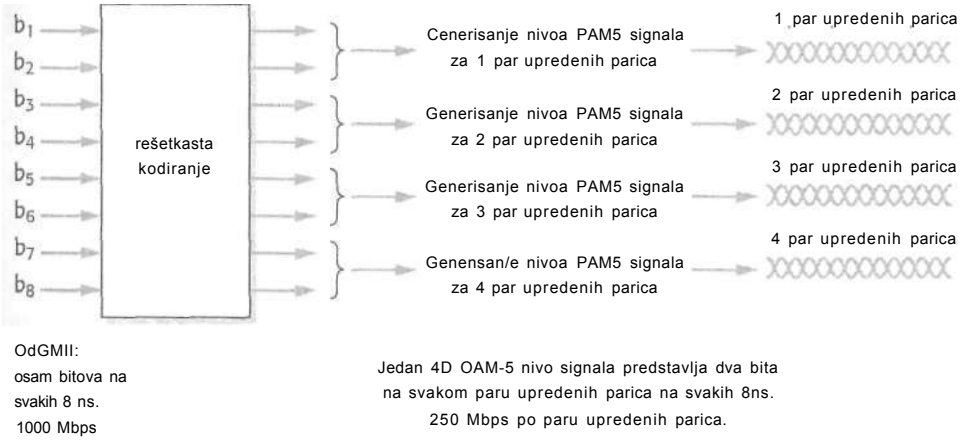
Gigabit Ethernet takode može da se pokreće preko UTP kablova kategorije 5. Medutim, generisanje gigabitskih brzina preko bakarnih vodova postavilo je neke ozbiljne izazove za 802.3ab komitet koji je bio odgovoran za lOOOBaseT specifikaciju. U odeljku 9.4, o Fast Ethemetu, predstavljeni su neki problemi koji prate visoke frekvencije neophodne za postizanje većih bitskih brzina preko LITP kablova. Problemi su se još više uvećali kada su dostignute gigabitske brzine. Prosto proširivanje pristupa koji se koristio za lOOBaseTX nije bilo moguće, zato što je povećanje bitske brzine zahtevalo frekvencije signala koje prelaze kategoriju 5 UTP kablova. Drugim rečima, nije funkcionisalo!

Pravi problem se sastojao u rešavanju šuma koji se generisao kod visokofrekventnih signala koji su premašivali specifikacije kategorije 5 UTP kablova. Jedan mogući pristup je uključivao korišćenje MLT-3 kodiranja za četiri para upredenih parica umesto samo jednog. Ako bi bilo moguće generisati 250 Mbps na jednom od četiri para upredenih parica, dobila bi se željena brzina od 1 Gbps. Iako je ova ideja uspela kod lOOBaseTX specifikacije sa tri para upredenih parica, kod lOOOBaseT bitovi su se generisali 10 puta brže i neophodna frekvencija za generisanje MLT-3 signala za 250 Mbps na svakom paai i dalje je bila previsoka. Da bi se to zaobišlo, lOOOBaseT koristi veoma sofisticirane tehnike: PAM5 kodiranje, rešetkasto kodiranje i Viterbi dekodiranje (slika 9.23).

PAM5 je akronim za *pulse amplitude modulation with five levels* (impulsna amplitudska modulacija sa pet nivoa). Osnovna ideja je da se u jednom trenutku uzme osam bitova od GMII, da se svaka 8-bitna grupa podeli na četiri 2-bitne grupe (po jedna za svaki UTP), a zatim da se dodeli odgovarajućem nivou signala (izabranom izmedu pet mogućih nivoa) za svaku 2-bitnu grupu. Tako četiri zasebna, konkurentna signala prenose osam bitova informacija.



SLIKA 9.22 *lOOOBaseXprenos*



SUKA 9.23 *1 OOOBaseT prenos*

Jedan 4D OAM-5 nivo signala predstavlja dva bita na svakom paru upredenih parica na svakih 8ns. 250 Mbps po paru upredenih parica. (One 4D PAM-5 level signal representing 2 bits on each wire every 8 ns. 250 Mbps per wire pair.)

Prvo pitanje: Zašto koristiti pet nivoa u okviru šeme za kodiranje? Imajte na umu da četiri para prenose osam bitova istovremeno. Pošto postoji 2"=256 mogućih vrednosti, potrebno je najmanje 256 kombinacija signala preko četiri UTP-a. Ako svaki par može da postigne četiri nivoa signala, postojalo bi 4" = 256 mogućih kombinacija signala preko četiri para. Iako je to dovoljno, nema mogućnosti za dodatne signale koji se koriste za kontrolne funkcije. Zato standard koristi pet nivoa, čime se dobija 5" = 625 kombinacija signala. Uobičajene oznake nivoa su -2, -1, 0, +1, i +2.

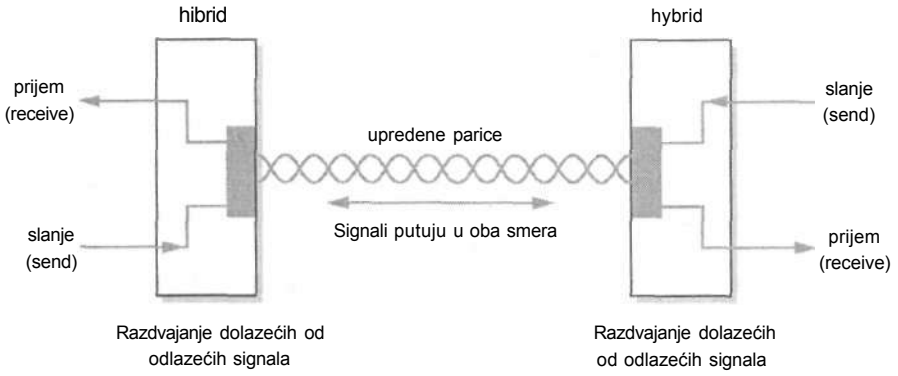
Da bi se postigle gigabitske brzine, svaki par mora da obezbedi 250 Mbps. Iako svaki nivo signala može da smesti dva bita, frekvencija koja je potrebna za slanje dva bita u jednom trenutku na brzini od 125 MHz i dalje ima koeficijent signal-šum koji premašuje granice IJTP kablova kategorije 5. Zato je 802ab komitet ugradio tehnike za korekciju grešaka i detektovanje. Tu nastupa rešetkasto (trellis) i Viterbi dekodiranje. *Rešetkasto kodiranje* (trellis envoding) je primer koda koji koristi dodatni nivo signala u PAM5 (dva bita zahtevaju samo četiri nivoa) za kreiranje redundansi u kodu. Redundanse se koriste za povećanje koeficijenta signal-šum za UTP signale. *Viterbi dehodiranje* na prijemnoj strani uključuje algoritme dizajnirane za traženje oštećenih signala i oporavak originala. Opis funkcionisanja rešetkastog kodiranja i Viterbi dekodiranja je izuzetno složen i zahteva dobro poznavanje signala, distribucije napona, tehnika za uobličavanje impulsa i ekvilizaciju signala i karakteristike signala i šuma. Ovi metodi su slični tehnologijama koje se koriste kod 56Kbps modema i dobro su se pokazali. Medutim, ovde to nećemo detaljnije opisivati, tako da zainteresovane čitaoce upućujemo na reference [Sc97], [GrOl] i [PrOl].

Poslednja karakteristika koju ističemo je full-duplex priroda lOOOBaseT specifikacije. Kod prethodnih standarda je korišćen unidirekcioni prenos, kod koga su zasebni parovi upredenih parica korišćeni za prenose u svakom trenutku. lOOOBaseTuključuje bidirekcioni prenos. Signali putuju u oba smera istovremeno (slika 9.24). Kao što možete da pretpostavite, problemi nastaju kada signali istovremeno putuju u suprotnim smerovima. Stvara se eho koji dovodi do izobličenja stvamih signala. Srećom, postoji rešenje za taj problem: lOOOBaseT implementira bidirekcioni mod pomoću *hibrida,* složenih uređaja koji mogu da eliminišu šum izazvan ehom. Hibridi mogu i da razdvoje lokalno prenete signale od onih koji dolaze od drugih uredaja.

U vreme pisanja ove knjige većina radnih stanica je i dalje koristila 10 Mbps, ili Fast Ethernet konekcije. Gigabit standard je dizajniran prvenstveno za fiber konekcije koje se koriste za povezivanje komutatora i obezbeduje putanje izmedu Fast Ethernet grupa: medutim, za povezivanje ovakvih uređaja na ispitnim mestima mogu se koristiti i zaštićeni bakarni vodovi (uređaji koji će u bliskoj budućnosti biti u praktičnoj primeni). U tabeli 9.6 je dat pregled Gigabit standarda.

Brzine veće od gigabita

Nakon diskusije o razvoju Etherneta od 10 Mbps do Gigabit Etherneta, postavlje se pitanje kuda dalje? Naravno, odgovor je *10 Gigabit Ethernet,* koji je razvio 802.3ae komitet i usvojen je kao standard u leto 2002. godine. Zašto bi se razvijalo sledeće desetostruko povećanje bitske brzine? Kao i ranije, odgovor je ispunjavanje zahteva. Radne stanice povezuju LAN mreže na bitskim brzinama od 10, ili 100 Mbps. Kućni korisnici prelaze sa 56Kbps modema na kablovske modeme i DSL tehnologije. Prosto rečeno, krajni korisnici generišu i troše više informacija nego ikada ranije i neprestano se povećava tok razmene informacija izmedu njih. Zato je neophodno dalje razvijati tehnologije koje se koriste u osnovama ovih komunikacionih sistema. 10 Gigabit Ethernet se koristi uglavnom kao "kičmeni stub" ne samo u LAN okruženjima, već i gradskim i WAN mrežama kod kojih je neophodno obezbediti široki propusni opseg signala.



SLIKA 9.24 *UTP full-duplex mod preko UTP-a*

Tabela 9.6: Fizičke implementacije Gigabit Ethernet standarda

Specifikacija kabla	Komentari
1000BaseT(IEEE802.3ab)	Zahteva četiri para upredenih parica kategorije 5 UTP kablova i pokreće se u full-duplex modu. Za signaliziranje su neophodne složene procedure kodiranja/dekodiranja: PAM5, rešetkasto i Viterbi. Maksimalna dužina segmenta iznosi 100 metara.
1000BaseCX(IEEE802.3z)	Medijum je specijalni zaštićeni bakami kabl. Maksimalno rastojanje iznosi 25 metara. Obično se koristi za povezivanje uredaja na ispitnim mestima u komunikacionom centru. Koristi 8B/10B kodiranje.
1000BaseLX(IEEE802.3z)	Dugotalasni optički fiber. Može da se koristi ili u više modova, ili samo sa jednim modom. Maksimalne dužine iznose 5.000 metara za fiber sa jednim modom i 550 metara za više modova. Koristi se 8B/10B kodiranje.
1000BaseSX(IEEE802.3z)	Kratkotalasni optički fiber. Može da se koristi sa fiberom sa više modova. Maksimalne dužine su 220 i 550 metara, u zavisnosti od prečnika fibera. Koristi 8B/10B kodiranje.

Prilikom uspostavljanja 10 Gigabit Ethemet standarda, 802.3ae komitet je utvrdio pet kriterijuma (videti http://grouper.ieee.Org/groups/802/3/ae/criteria.pdf):

- Potencijal za plasiranje na širokom tržištu
- Kompatibilnost sa postojećim 802.3 standardima
- Značajne razlike u odnosu na prethodnike
- Tehnička izvodljivost
- Ekonomska isplativost

Pored očiglednog 10-strukog povećanja brzine u odnosu na Gigabit Ethernet, 10 Gigabit Ethernet ima sledeće značajne karakteristike:

- Koristi isti format MAC okvira kao i njegovi prethodnici, čime je obezbedena kompatibilnost.
- Podržava samo full-duplex mod.
- Dizajniran je za funkcionisanje samo preko optičkog fibera.
- Ne uključuje CSMA/CD protokol kao raniji Etherneti. Više nema kolizija.
- Podržava prenos na većim rastojanjima (40 km za fiber sa jednim modom).
- Može da se pokreće preko SONT nosećeg sistema - preciznije rečeno, preko OC-192 kola.

802.3ae standard definiše dva različita tipa fizičkih slojeva: LAN PHY i WAN PHY. LAN PHY je dizajniran za međusobno povezivanje IAN mreža i za podršku postojećih gigabitskih aplikacija, ali na mnogo većim brzinama. Kao i u slučaju prethodnih verzija, LAN PHY sadrži PCS podsloj

koji komunicira sa podslojem izmirenja. Medutim, PCS sloj za 10 Gigabit Ethernet koristi 64B/66B šemu kodiranja, koja je slična ranije predstavljenim šemama 4B/5B i 8B/10B, osim što kodira 64 bita u 66 bitova. Istovremeno procesiranje više bitova je brže. Takode, kod ove šeme postoji manje dodatnih bitova (dva bita na svakih 66, za razliku od dva bita na svakih 10). Tako je omogućena bitska brzina koja bliže odgovara frekvencijama signaliziranja koje su neophodne za slanje bitova.

WAN PHY je dizajniran za proširenje Etherneta preko tradicionalnog LAN okaiženja i da bi se olakšalo povezivanje većih oblasti. Pošto se tekuće komunikacije na većim rastojanjima uglavnom odvijaju preko više protokola, korišćenje Ethemet tehnologije u komunikacijama na većim udaljenostima značajno uprošćava "stvari". Više nije neophodno konvertovati okvire u neke druge forme pre prenosa. Da, svet se "smanjuje" i sada vidimo potendjal za stvaranje svetskih informacionih sistema koji će biti povezani pomoću LAN tehnologija.

Glavna razlika između WAN PHY i LAN PHY je u tome što je podsloj WAN interfejsa (WIS-WAN interface sublayer), smešten između PCS i PMD podslojeva, dizajniran tako da omogući ransparentne komunikacije preko SONET OC-192 kola. Pošto su bitske brzine za OC-192 kola nešto ispod 10 Gbps, postoji prirodno poklapanje. WIS podsloj prima kodirane bitove sa PCS podsloja, ugraduje ih u SONET okvir, koga predaje PMD posloju. Tako Ethernet može da koristi SONET nosioce za transport preko WAN mreža - samim tim, opseg Ethernet mreža je proširen na mnogo veća rastojanja.

802.3ae standard ima opcione interfejse: interfejs nezavisan od 10 Gigabit medijuma (XGMII-IO Gigabit media independent interface) i XGMII pridružena jedinica interfejsa (XALII). XGMII dopušta full-duplex komunikacije između PCS podsloja i podsloja izmirenja i ima putanju podataka širine 74 bita. Putanja uključuje 32 bita u oba smera, kao i takt i kontrolne funkcije. Medutim, njen dizajn ne dopušta direktne konekcije čip-u-čip, ili čip-u-optički modul, zbog hardverskih ograničenja koja odreduju rastojanje koje je moguće pokriti magistralom. Alternativa je XAUI, sa manjim brojem linija u poredenju sa XGMII i dva i po puta većom brzinom od lOOOBaseX. Ima i drugačije električne karakteristike, koje proširuju domet XGMII-Ja.

Formalno odobrenje 10 Gigabit Ethernet standarda desilo se samo par meseci pre nego što je ovaj rukopis pripremljen i postoji samo nekoliko objavljenih knjiga i članaka o tome. Zainteresovani čitaoci mogu da pronadu više informacija na Web sajtovima http://grouper.ieee.org/ groups/802/3/ae/ i www.10gea.org/. Ovi sajtovi nude brojne linkove ka sajtovima koji obezbeduju detaljniju diskusiju o raznim aspektima 10 Gigabit standarda.

Da li nas očekuje 100 Gigabit Ethemet? Verovatno. Prvi korak u procesu standardizacije je identifikovanje sponzora (tehničkog udruženja, ili grupe u okviru IEEE) koji će nadgledati razvojni proces. U vreme kada je ova knjiga pripremana, još nije uspostavljen 100 Gigabit Ethernet standard, tako da se verovatno neće pojaviti u skorijoj budućnosti. Osim toga, gigabitski uređaji se često koriste u LAN okruženjima; 10-bitski uredaji su skupi i prvenstveno se koriste za telekomunikacione nosioce.

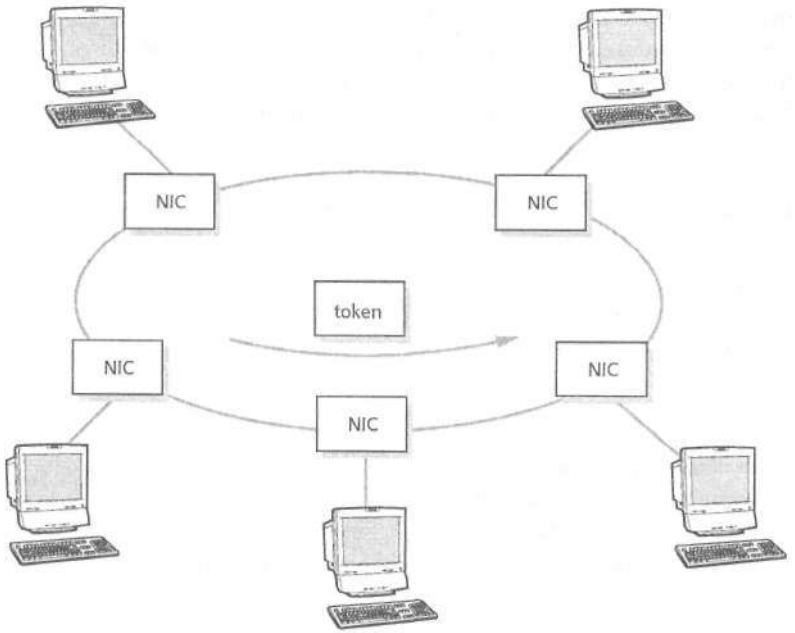
Neki predviđaju da će proći više godina pre nego što se cene spuste do te mere da 10-gigabit tehnologije postanu široko dostupne i dok ljudi ne počnu da smatraju da su im neophodne brzine od 10 gigabita. I pored toga, nije zahvalno davati bilo kakva predviđanja: jedino možemo da nastavimo da pratimo sve što se novo objavljuje.

9.6 Token ring: IEEE standard 802.5

Token ring LAN je defmisan IEEE standardom 802.5. Kao i Etheraet, token ring je MAC protokol koji je smešten izmedu sloja kontrole logičkog linka i fizičkog sloja. Brzine prenosa podataka za token ring mreže su navedene kao 4 Mbps i 1 Mbps, iako IBM token ring može da se pokreće na 4, 16, ili 100 Mbps. Prenos se odvija pomoću tehnika diferencijalnog Mančester kodiranja, koje su opisane u odeljku 2.4.

Uređaji na token ring LAN-u su povezani u prsten pomoću NIC kartica (slika 9.25). Uređaj može direktno da šalje podatke samo do svojih suseda i u većini slučajeva samo do jednog suseda (na slici 9.25 suprotno od smera kretanja kazaljki na časovniku). Ako uredaj želi da pošalje nešto do drugog uređaja u prstenu, okvir mora da prode kroz sve interfejse koji se nalaze izmedu ta dva uređaja. "Nadmetanje" u okviru prstena je izvedeno pomoću tokena (specijalnog okvira) koji cirkuliše izmedu uređaja. Specifičnosti prisvajanja tokena i slanja okvira objasnićemo nešto kasnije u ovom odeljku. Za sada, predstavljamo veoma uopšten i jednostavan opis procesa.

Kada token stigne do uredaja, moguće su dve "stvari". Ako uređaj nema podatke za slanje, on rutira token do susednog uređaja. Ako ima nešto za slanje, prisvaja token, uklanja ga sa prstena i šalje okvir na mestu tokena.



SLIKA 9.25 *Token ring mreža i cirkulisanje tokena*

Nakon toga, okvir "putuje" preko prstena i svaki uredaj proverava njegovu odredišnu adresu. Ako se odredišna adresa ne poklapa sa adresom tekućeg uređaja, on rutira okvir do svog suseda. Ako se poklapa, odredišni uredaj kopira okvir, postavlja u njega neke bitove statusa i rutira okvir do svog suseda. Okvir nastavlja da kruži kroz prsten sve dok ne stigne do uređaja koji ga je kreirao. Taj uređaj uklanja okvir sa prstena, generiše novi token i šalje ga nazad na prsten.

Ovde je moguće odmah uočiti dve "stvari". Prva.je da je "nadmetanje" za pristup prstenu uredenije nego kod Etherneta. Svaki uredaj zna kada može da šalje podatke i šalje ih do svog suseda. Znači, nema trošenja propusnog opsega zbog kolizija. Drugo, odmah se uočava da otkaz jednog uredaja izaziva otkaz cele mreže. Za razliku od Etherneta, svaki uređaj učestvuje u rutiranju tokena, ili okvira podataka. Ako dode do otkaza uredaja, on neće moći da rutira primljeni token, ili okvir sa podacima, tako da se konkretni okvir gubi iz prstena.

Formati okvira i tokena

Već smo rekli da je token specijalna vrsta okvira. Na slici 9.26 prikazani su formati tokena i okvira sa podacima. Četiri polja označena kao "destination address" (odredišna adresa), "source address" (izvoma adresa), "data" (podaci) i "frame check sequence" (provera okvira) imaju isto značenje kao i ista polja u ranije prikazanim formatima, tako da ih ovde nećemo ponovo objašnjavati. Odredišna adresa može da bude individualna, grupna, ili emisiona (broadcast). Polje Data teorijski nema maksimalnu dužinu, mada postoji ograničenje za količinu koju uredaj može da prenese bez prekida, tako da praktićna granica iznosi oko 5.000 bajtova podataka.

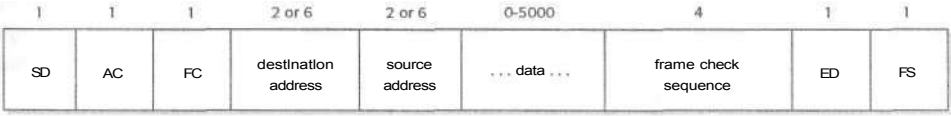
Svaki okvir ima **startni delimiter** (SD) i **završni delimiter** (ED) kojima se definišu granice tokena. SD ima specijalni uzorak signala JK0JK000. NuIe su binarne nule, onako kako su definisane diferencijalnim Mančester kodom. Simboli J i K odgovaraju specijalnim signalima. Da biste razumeli zašto se tu nalaze, setite se da diferencijalni Mančester kod definiše prelaz signala (sa visokog na niski, ili sa niskog na visoki) na sredini svakog intervala. J i K signali odstupaju od tog pravila. Signal J startuje kao 0, ali nema prelaza na sredini. Slično tome, signal K startuje kao I, ali nema prelaza. Ponekad se za ove signale koriste oznake **non-data-J** i **non-data-K.** Pošto ovi signali nisu u skladu sa Mančester kodovima za defmisanje bitova, nikada ne mogu da se jave kao deo bilo kojih informacija. Zato su korisni za označavanje specijalnih stanja, kao što su početak i kraj okvira.

Završni delimiter koristi JK1JK1IE uzorak signala. Simboli J i K su isti kao i kod SD-a. Jedinice su binarne jedinice. Dva preostala bita odgovaraju bitu prelaznog okvira (I) i bitu greške (E). Kao i ranije, komunikacija izmedu dva uredaja obično uključuje više okvira. Bit I je u poslednjem okviru 0, a u ostalim slućajevima ima vrednost 1. Bit E se postavlja na 1 svaki put kada se detektuje greška (kao kod sekvence za proveru okvira).

Drugi bajt u svakom okviru je polje Access Control (AC). Njegovi bitovi mogu da imaju različito značenje. Bit označen kao t označava *bit tokena* i odreduje tip okvira. Kod tokena je $t = 0$, a kod okvira sa podacima t = 1, tako da uredaj može da utvrdi šta je primio. Preostali bitovi služe za održavanje prstena i rezervaciju tokena, o čemu će nešto kasnije biti više reči.

(a) token veličlne trl bajta

broj bajtova



(b) Okvir sa promenljlvlm brojern bajtova

SD (startnl delimiter): J K O J K O O O
AC (kontrota prlstupa): p p p t m r r r
p p p: bitovi prloriteta
t: blttokena
m: blt za praćenje
r r r: bltovl rezervacije
ED (završnl delimiter): J K I J K I I E
FC (kontrola okvlra): f f z z z z z z
f: bitovi za tlp okvira
z: kontrolni bitovl
FS (status okvlra): a c x x a c x x
a: bit prepoznavanja adrese
c: blt koplranja okvlra
x: nedefinlsanl blt

SLIKA 9.26 *Formati tokena i okvira sa podacima*

Treći bajt u okviru je polje Frame Control (FC), koje je takode zaduženo za održavanje prstena.

Poslednji bajt je polje Frame Status (FS) koje uključuje dve kopije (za slučaj greške) bita za prepoznavanje adrese (Address Recognized Address-bit *a)* i bit kopiranja okvira (Frame Copied-bit *c).* Uređaj koji šalje okvir inicijalno postavlja bitove *a* i *c* na O. Ako se odredišni uredaj nalazi na prstenu, on postavlja bit *a* na 1, čime ukazuje da je adresa prepoznata. Ako odredišni uredaj iskopira okvir, postavlja i bit c na 1. Primećujete da prisustvo odredišnog uredaja ne znači automatski da je okvir iskopiran. Status koji je postavljen na višem sloju (kao što je LLC) može privremeno da zabrani prihvatanje novih okvira. Takve mogućnosti smo prikazali u Poglavlju 8, u odeljcima o protokolima klizajućih prozora.

Polje Frame Status "govori" uredaju pošiljaocu da li je odredišni uredaj na prstenu i, ako jeste, da li je iskopirao okvir. Ako je odredišni uređaj tu, ali nije iskopirao okvir, pošiljalac pretpostavlja da kasnije treba ponovo da pokuša da pošalje okvir.

Rezervisanje i pnsvajanje tokena

Proces zahvatanja tokena i slanja okvira sa podacima na prvi pogled deluje jednostavno. Kada uredaj pošalje okvir i u povratku ga ukloni sa prstena, on kreira novi token koji šalje do svog suseda, koji tada dobija šansu da prisvoji token. Nastavljajući na ovakav način, uredaji dobijaju šansu za prenos onim redosledom kojim su povezani. Ovakav proces postavlja i gornju granicu u dužini vremena za koje uredaj mora da čeka na token.* Ali, da li je moguće "pogaziti" ovaj redosled? Da li postoje načini da neki uređaj dobije viši prioritet i da tako dobije šansu za slanje pre ostalih uredaja? Ovo bi sigurno bilo korisno u slučajevima gde token ring opslužuje real-time uređaje, ili uredaje visokog prioriteta.

Da bi se odredili prioriteti i uredajima omogućilo zahvatanje tokena drugačijim redosledom od onog kojim su povezani, pretpostavljamo da svaki uređaj, kao i token koji cirkuliše, ima svoj prioritet. Prioritet uređaja je defmisan lokalno, a prioritet tokena je definisan sa tri *bita prioriteta* (Priority) u AC polju. Uredaj može da prisvoji token samo ako je njegov prioritet veći, ili jednak prioritetu tokena. Tako uređaji nižeg prioriteta moraju da proslede raspoloživi token, i pored toga što možda imaju nešto za slanje. Jedino uredaji sa većim, ili jednakim prioritetom od prioriteta tokena mogu da prisvoje token.

Ovaj sistem nameće neka značajna pitanja. Ko definiše prioritet tokena? Kako se to izvodi? Inicijalno, jedan uredaj šalje token sa prioritetom O. Nakon toga, sve nadalje se prepušta *sistemu za rezervacije,* protokolu koji se koristi za rezervisanje tokena i definisanje prioriteta.

Pretpostavimo da uredaj primi token sa višim prioritetom od svog prioriteta (uskoro ćemo videti kako može da dođe do toga), ili da primi okvir sa podacima. U bilo kom slučaju, uredaj ne može da šalje svoje podatke. Ipak, on može da postavi zahtev (rezervaciju) za tokenom kada on naide sledeći put. Da bi to uradio, uredaj proučava dolazeće *bitove rezervacije* (Reservation). Ako vrednost tih bitova ukazuje na niži prioritet od onog koji ima uredaj, on postavlja svoj prioritet i na taj način rezerviše token. Ako je vrednost bitova ukazala na viši prioritet, uredaj ovoga puta ne može da rezerviše token. Moguće je da je neki drugi uredaj sa višim prioritetom, ali i dalje nižim od prioriteta tokena, već rezervisao token i ne može da ga ukloni uredaj nižeg prioriteta (pokušajte da rezervišete svoj omiljeni apartman u hotelu sa pet zvedica ako Je premijer Engleske već rezervisao ceo sprat).

Kada uredaj povuče okvir i kreira novi token, on proučava bitove rezervacije u dolazećem okviru. Ako vidi da je neki uredaj rezervisao token, on definiše novi token sa prioritetom koji odgovara vrednosti rezervacije. Zatim, u lokalni stek smešta i stari i novi prioritet (koji sada postaje tekući prioritet). Nakon toga, označen je kao stanica za odlaganje (stacking station) - jedino taj uređaj može da vrati token na njegov originalni prioritet. Kada token počne da "putuje" preko prstena, prisvojiće ga prvi uredaj sa višim, ili jednakim prioritetom. Uredaji sa nižim prioritetom se ignorišu u korist onih sa višim prioritetima.

Na slici 9.27 prikazan je pseudoalgoritam koji opisuje sistem za definisanje prioriteta i rezervacije. Svaki uređaj izvršava algoritam po dolasku okvira i tokena.

* Ova tvrdnja je data uz pretpostavku da postoji ograničenje u dužini vremena za koje stanica može da zadrži token. U stvari, podrazumevana vrednost lajmera za zadržavanje tokena iznosi 10 (isec, čime je odredeno koliko dugo neka stanica može da kontroliše token.

```
While (Petar Pan zivi u Nedodjiji)
{
  cekanje na dogadjaj;
  if (dogadjaj je "okvir stigao")
  {
1   if (okvir potice Iz tekuce stanica)
    {
    povlacenje okvira;
2   if (frame.res > frarne.priority)
    {
        kreiranje i slanje tokena sa token.priority = frame.res i token.res = 0;
        postavljanje starog i novog prioriteta na stek i oznacavanje ove stanice
          kao stanice za odlaganje;
        continue;
        }/* kraj uslova • frame.res > frame.priority */
3       if (tekuca stanica je stanica za odlaganje ovog okvira)
        {
        kreiranje i slanje tokena sa token.priority = max(frame.res, stari prioritet sa steka);
        if koriscen frame.res izvesti zamenu tekuceg prioriteta sa vrha steka;
        if koriscen stari prioritet sa steka, povuci staru i tekucu vrednost sa steka i ukloni
          oznaku stanice za odlaganje ako je stek nakon toga prazan;
        continue;
        }/* kraj uslova - tekuca stanica je stanica za odlaganje ovog okvira */
        kreiranje i slanje tokena sa token.priority =frame.res i token.priority = frame.priority;
    }/* kraj uslova - okvir potice sa tekuce stanice */
    if (okvir potice sa nekog drugog mesta)
4   {
        if (postoji okvir za slanje) && (prioritet stanice > fname.res) smesti prioritet za
5         slanje u frame.res;
        slanje okvira;
        }
    }/* kraj uslova - dogadjaj je "okvir stigao" */
    if (dogadjaj js "toksn stigao")
    {
        if (tekuca stanica je stanica za odlaganje za ovaj token)
6       {
          kreiranje tokena sa token.priority = max(token.res, stari prioritet sa steka);
          if koriscen frame.res izvesti zamenu tekuceg prioriteta sa vrha steka;
            if koriscen stari prioritet sa steka, povuci staru i tekucu vrednost sa steka i ukloni
            oznaku stanice za odlaganje ako je stek nakon toga prazan;
        }
        if (postoji okvin za slanje)
7       {
          if (prioritet stanice >= token.priority)
8           prisvajanje tokena, kreiranje i slanje okvira;
          else
            if (prioritet stanice > token.res)
9             smestanje prioriteta posiljaoca u token.res;
        }/* kraj uslova - okvir za slanje */
          slanje okvira, ili tokena;
    /* kraj uslova - dogadjaj je "token stigao" */
  /* kraj while petlje */
```

SLIKA 9.27 *Token ring protokol za rezervaciju i prisvajanje tokena*

Algoritam je pomalo pojednostavljen i ne pokazuje kako uređaj prima okvire, ili kako šalje više okvira. Nameravali smo da se fokusiramo samo na postavljanje rezervacija i utvrdivanje prioriteta, ali, vežbe radi, možete i sami da pokušate da proširite algoritam.

U algoritmu bitovi rezervacije i prioriteta su označeni kao frame.res i frame.priority, respektivno. Slična notacija je korišćena i za bitove tokena. Kao i prethodni algoritmi, i ovaj je određen događajima, gde se pod događajem smatra dolazak okvira sa podacima, ili tokena. Ako stigne okvir sa podacima, uređaj utvrduje da li je on poslao okvir (uslov 1), ili je okvir poslao neki drugi uređaj (uslov 4). Ako okvir potiče od nekog drugog uredaja i ako tekući uredaj ima podatke za slanje, on pokušava da rezerviše token. Ako je prioritet uređaja viši od vrednosti frame.res (uslov 5), postavlja se rezervacija. Ako nije, nema rezervacije. U svakom slučaju, okvir se prosleduje do susednog uređaja.

Ako je okvir potekao od tekućeg uređaja, uređaj mora da povuče okvir i da kreira novi token. Pitanje na koje uređaj mora da daa odgovor je kakav prioritet treba da ima novi token. Postoje dva moguća odgovora.

Prvo, pretpostavimo da je neki uredaj sa višim prioritetom od prioriteta okvira rezervisao token (uslov 2). Uredaj tokenu daje prioritet jednak prioritetu uredaja koji je izvršio rezervaciju (vrednost frame. res). Takode, on postavlja frame.res = 0 kako bi se dala šansa za sledeće rezervacije. Ali, podizanje prioriteta tokena je "strašna" odgovornost. Svaki uredaj koji to radi ujedno je i odgovoran za njegovo snižavanje kasnije kada samo uredaji sa nižim prioritetom imaju nešto za slanje. U tom trenutku uređaj mora da prepozna token sa podignutim prioritetom i mora da snizi prioritet. Da bi se to postiglo, uredaj (označen kao stanica za odlaganje) smešta stari i novi (sada tekući) prioritet na stek, svaki put kada povisi prioritet.

Druga mogućnost se javlja kada je rezervaciju u dolazećem okviru kreirao uredaj sa nižim prioritetom od prioriteta okvira. To znači da u vreme dok je okvir putovao preko prstena, ni jedan uredaj sa višim prioritetom nije imao podatke za slanje. Zato je neophodno sniziti prioritet tokena. Ali, koji je uređaj nadležan za to? Ako tekući uredaj nije povisio prioritet tokena, on ne može ni da ga snizi. Zato on jednostavno kreira token sa istim prioritetom koji je naznačen u bitovima rezervacije dolazećeg okvira (poslednja naredba u okviru uslova 1). Moguće je da će token stići do stanice za odlaganje, koja može da snizi prioritet tokena.

Medutim, ako je tekući uređaj ujedno i stanica za odlaganje* (uslov 3), on kreira novi token sa nižim prioritetom. Sledeće pitanje je kakav je novi prioritet. Stanica za odlaganje poredi vrednost dolazeće rezervacije sa starim prioritetom sa steka i bira veću vrednost. Ako je vrednost rezervacije veća, ona se postavlja na mesto tekućeg prioriteta na steku. Uređaj i dalje predstavlja stanicu za odlaganje, jer još uvek nije obnovljen prioritet koji je postojao pre nego što je taj uredaj postao stanica za odlaganje. Ako vrednost rezervacije nije veća, token koristi stari prioritet sa steka. Stari i tekući prioriteti se skidaju sa steka i ako je stek nakon toga prazan, uređaj više nije stanica za odlaganje.

* Moguće je da postoji više stanica za odlaganje, zato što se prioritet tokena može podizati u više stanica. Nama Je ovde potrebna stanica koja je u najskorije vreme postala stanica za odlaganje. Stanica mora da to utvrdi poredenjem polja za prioritet okvira sa tekućim prioritetom na lokalnom steku konkretne stanice. Ako su isti, tekuća stanica može da snizi prioritet tokena. Ako nisu, onda je neka druga stanica podigla prioritet tokena i zato je ona odgovoma za njegovo snižavanje.

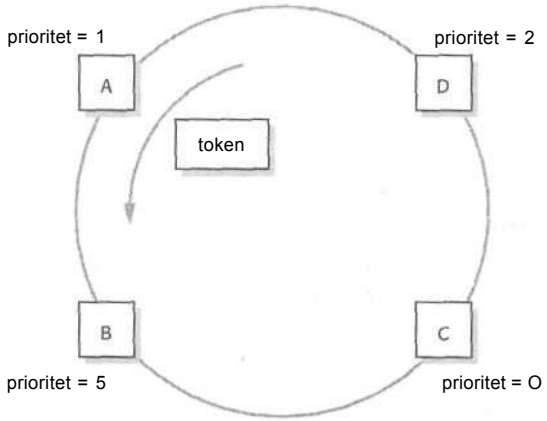
Napomenimo da se oznaka stanice za odlaganje uklanja samo ako je stek prazan, jer je ista stanica mogla da bude odgovorna i za prethodno povećanje prioriteta.

Razmotrimo sada šta se dešava kada token stigne. Ako je uredaj stanica za odlaganje (uslov 6), proces se nastavlja kao što smo malopre objasnili. Nakon toga, uredaj utvrđuje da li postoji okvir za slanje (uslov 7). Ako postoji, poredi prioritet tokena sa prioritetom uredaja (uslov 8). Ako uređaj ima dovoljno visok prioritet, on prisvaja token. Ako nema, poredi svoj prioritet sa vrednošću bitova rezervacije (uslov 9). Ako ima dovoljno visok prioritet, on postavlja rezervaciju. Konačno, šalje ili token, ili okvir (ako ga je kreirao) do svog suseda.

Ova diskusija je opšta i opisuje samo glavne aspekte sistema za definisanje prioriteta i rezervacija. Sada je vreme da vidimo i jedan primer. Pretpostavimo da su četiri uredaja spremna za slanje (slika 9.28) i da svaki od njih irna samo jedan okvir za slanje. Pretpostavimo da je uredaj C upravo zahvatio token i da šalje svoj okvir. Sada primenjujemo algoritam na situaciju sa slike 9.28.

Da biste lakše pratili algoritam, u tabeli 9.7 je dat pregled šta se dešava u svakom uređaju dok okvir, ili token "putuju" preko prstena. U prvoj koloni su navedeni koraci. U drugoj koloni je naznačen uredaj koji je primio okvir, ili token. U trećoj koloni su navedeni uslovi sa slike 9.27 koji su tačni kada okvir, ili token stignu. Četvrta kolona definiše da li ttredaj šalje okvir, ili token. Poslednje dve kolone defmišu vrednosti prioriteta i rezervacija (vrednosti od O do 7) odlazećeg okvira, ili tokena. Počinjemo od tačke u kojoj je uredaj C upravo zahvatio token i poslao svoj okvir.

Pratimo algoritam nakon što C šalje okvir, tako da prva linija u tabeli 9.7 odgovara dolasku okvira u D (korak 1). U ovom slučaju uslovi 4 i 5 su tačni. Drugim rečima, okvir nije potekao od uredaja D, D ima okvir za slanje, a njegov prioritet je viši od prioriteta dolazećeg okvira. Zato D smešta svoj prioritet (2) u frame.res i prosleduje okvir do svog suseda. Kada A primi okvir (korak 2), jedino je ispunjen uslov 4 (A ima niži prioritet).



SLIKA 9.28 *Rezervisanje tokena u token ring mreži*

Tabela 9.7: Aktivnosti koje se izvode dok token i okvir "putuju" preko prstena

Korak	Dolazak u	Uslov	Uredaj šalje	Prioritet	Rezervacija
1	D (okvir)	4 i 5	Okvir	0	2
2	A (okvir)	4	Okvir	0	2
3	B (okvir)	4 i 5	Okvir	0	5
4	C (okvir)	1 i 2	Token	5	0
5	D (token)	7 i 9	Token	5	2
6	A (token)	7	Token	5	2
7	B (token)	7 i 8	Okvir	5	**0**
8	C (okvir)	4	Okvir	5	0
9	D (okvir)	4 i 5	Okvir	5	2
10	A (okvir)	4	Okvir	5	2
11	B (okvir)	1	Token	5	2
12	C (token)	6	Token	2	2

Kada B primi okvir (korak 3), reaguje kao što je prethodno reagovao uredaj D i povećava f ranie. res na 5. Okvir konačno stiže nazad u C (korak 4), uredaj odakle je potekao, i tu se povlači sa prstena. Pošto su uslovi 1 i 2 ispunjeni, C kreira i šalje token sa prioritetom 5 i postaje stanica za odlaganje. D prima token (korak 5), ali ne može da ga zahvati, jer je prioritet tokena suviše visok. Medutim, postavlja vrednost rezervacije na 2 i šalje token A (korak 6). Uredaj A ima nizak prioritet, tako da šalje token do B (korak 7). Uređaj B prisvaja token (uslovi 7 i 8) i šalje okvir.

Korak cirkuliše kao i ranije i eventualno se vraća do B (korak 11). B povlači okvir i kreira token. Primećujete da polje Reservation u tokenu ima vrednost 2, a prioritet Je 5. Pošto B nije stanica za odlaganje (to je bio uređaj C u koraku 4), šalje token sa istim prioritetima i vrednostima rezervacije kao i u primljenom okviru. Kada C primi okvir (korak 12), vidi da je ispunjen uslov 6 i snižava prioritet tokena na 2. Stari prioritet se i dalje nalazi na steku i C i dalje ostaje stanica za odlaganje. Iako su preostali koraci slični onima koji su se već desili, trebalo bi da ih predete. Primećujete da se tabela 9.7 može proširiti do tačke u kojoj su svi uređaji poslali svoje okvire i kada je prioritet tokena ponovo vraćen na O. Neka ovo bude vežba za Vas.

Održavanje prstena

U dosadašnjoj diskusiji su opisane operacije token ring mreže u situaciji kada se ne dešavaju nikakve greške. Ipak, to je opasna pretpostavka. "Stvari" mogu da "krenu naopako". Na primer:

- Uredaj šalje kratki okvir preko dugačkog prstena (gde je poslednji bit poslat pre nego što se prvi vratio) i zato pada. Uredaj ne može da povuče okvir sa prstena. Okvir koji nije povučen je ispušteni okvir (orphane frame).

- Uređaj prima okvir, ili token i pada pre nego što uspe da ih pošalje. Pošto više na prstenu ne postoji token koji cirkuliše, svi uređaji beskonačno čekaju na token.
- Šum na liniji oštećuje okvir. Koji je uredaj odgovoran za njegovo ispravljanje?

Neki problemi mogu da se reše tako što se nekim uredajima daju dodatne odgovornosti; ti uredaji se označavaju kao **stanice** za **nadgledanje,** ili **monitori** (monitor station). Na primer, da bi se detektovao ispušteni okvir, uredaj inicijalno kreira okvir sa bitom praćenja u Access Control bajtu (videti sliku 9.26) postavljenom na O. Kada monitor primi okvir, ovaj bit se postavlja na I. Ispušteni okvir nije uklonjen sa prstena, tako da po drugi put dolazi do monitora sa bitom koji je već postavljen na 1. Monitor povlači okvir i generiše novi token.

Monitor može da detektuje izgubljeni token pomoću ugrađenog tajmera. Tajmer je definisan u skladu sa dužinom prstena, brojem uredaja i maksimalnom veličinom okvira. Svaki put kada monitor šalje okvir, ili token, on startuje svoj tajmer. Ako monitor ne primi ni jedan okvir, ili token pre nego što tajmer istekne, pretpostavlja da je izgubljen i generiše novi token.

Ipak, neke probleme ni monitori ne mogu da reše. Na primer, šta ako je monitor stanica na kojoj je došlo do kvara? Šta ako do gubitka tokena dode zbog raskidanja prstena? Slanje novog tokena neće rešiti problem.

Ovi problemi se rešavaju pomoću *kontrolnih okvira,* kao što je prikazano u tabeli 9.8. Kontrolni bitovi u FC bajtu (videti sliku 9.29) definišu funkciju okvira. Kada se u prsten postavi novi uredaj, šalje Duplicate Address Test okvir koji smešta sopstvenu adresu uredaja u polje Destination. Ovaj okvir obezbeduje jedinstvenost adrese uredaja na prstenu. Kada se okvir vrati, uredaj proverava bit prepoznavanja adrese (Address Recognized) u polju Status konkretnog okvira. Ako Je 0, nema drugog uredaja sa takvom adresom; ako je 1, adresa uređaja je duplikat. Uredaj se sam uklanja sa prstena i podnosi izveštaj o nastaloj grešci.

Ako se rnora izabrati nova stanica koja će imati ulogu monitora, jedan, ili više uređaja šalju ponude da postanu monitori. Posao će biti poveren stanici sa najvišom adresom.

Tabela 9.8: Kontrolni okviri na token ring mreži

Tip okvira	Kontrolni bitovi u oktetu Frame Controi	Značenje
Active Monitor Present	000101	Šalje informacije da je uredaj monitor operativan i inicira proceduru identifikacije suseda.
Beacon	000010	Lociranje grešaka na prstenu
Claim Token	000011	Bira novi monitor.
Duplicate Address Test	000000	Provera da li postoje duplirane adrese.
Purge	000100	Čišćenje prstena
Standby Monitor Present	000110	Izvodi proceduru identifikacije suseda.

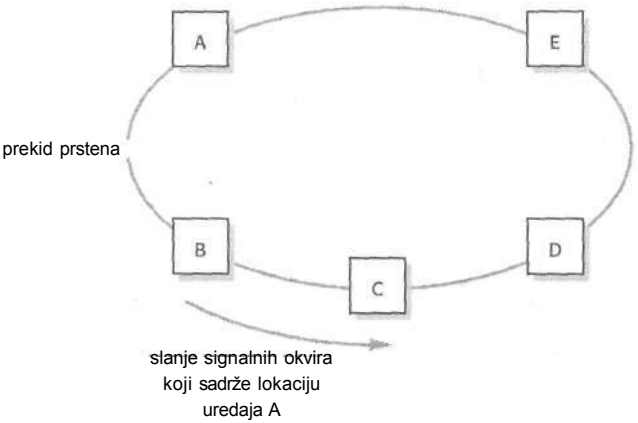
Problem je što ni jedan uređaj ne zna šta je. Da bi to utvrcHo, svaki šalje niz *Claim Token (CT) okvira*. Kada uređaj primi Claim Token okvir, poredi izvornu adresu tokena sa sopstvenom. Ako je izvorna adresa tokena viša, uređaj zaustavlja slanje sopstvenih okvira i ponavlja one koje prima. Ako je izvoma adresa tokena niža, uredaj je povlači sa prstena i nastavlja da šalje sopstvene okvire. Dakle, CT okvir nikada ne prelazi preko uredaja koji se "nadmeće" koji ima višu adresu i jedini CT okviri koji cirkulišu kroz prsten potiču od uređaja sa višim adresama. Kada uređaj primi svoj CT okvir, smatra da je propisno izabran kao monitor - nakon veoma kratke kampanje i bez pomoći nekih specijalnih interesnih izvora.

Kada se uredaj izabere kao monitor i povremeno nakon toga, on šalje *Active Monitor Present (AMP)* okvir koji obaveštava druge uređaje da postoji aktivna stanica za nadgledanje. Ako zbog nečega dode do kvara monitora, neće biti poslat ni jedan AMP okvir. Drugi uredaji imaju tajmere koji ističu kada se za određeno vreme ne detektuje ni jedan AMP okvir. Nakon toga, ti uredaji šalju ponude da postanu novi monitori slanjem CT okvira, kao što je prethodno navedeno.

Pre nego što monitor kreira i pošalje novi token, najpre šalje *Purge okvir*. U meduvremenu, on povlači sve što primi, uključujući vraćeni Purge okvir, kako bi bio siguran da je prsten čist pre nego što pošalje novi token, ili AMP okvir.

Standby Monitor Present (SMP) okvir je deo procedure za identifikaciju suseda. Kada monitor pošalje AMP okvir, on postavlja a bitove u polju Status na 0. Prvi uređaj koji primi okvir (sledeći sused u silaznom nizu) beleži izvornu adresu i postavlja a bitove u polju Status na I pre nego što prosledi okvir. Taj uredaj zna ko je njegov prethodni sused. Svi a bitovi su postavljeni na 1 kako bi drugi uredaji bili obavešteni da okvir koji stiže do njih nije okvir njihovog neposrednog suseda. Nakon prijema AMP okvira od prethodnog suseda, uredaj šalje SMP okvir (takode sa svim a bitovima postavljenim na 0). Njegov naredni sused prihvata okvir, beleži izvornu adresu, postavlja a bitove na 1 i prosleduje ga. Nakon nekog vremena, šalje sopstveni SMP okvir i njegov naredni sused reaguje slično. Ovaj kaskadni efekat uzrokuje da svaki uredaj šalje sopstveni SMP okvir kako bi obavestio naredni susedni uredaj o svom identitetu. Pošto su a bitovi postavljeni u prvom uredaju koji prima svaki SMP okvir, svaki uređaj može da napravi razliku od okvira koji potiče od prethodnog suseda i okvira koga je prethodni sused samo prosledio.

Beacon (signalni) okvir se koristi za informisanje uredaja da je došlo do problema i da je zaustavljen protokol za prosleđivanje tokena. Prethodno smo rekli da svaki uredaj ima tajmer koji detektuje odsustvo AMP okvira. Kada se to desi, uredaj ne zna da li je došlo do greške u monitoru, ili je došlo do raskidanja prstena. Ukoliko je kvar u prstenu, slanje CT okvira neće rešiti problem. Uredaj koji je detektovao problem šalje niz signalnih okvira sa adresom svog prethodnog suseda. Ako se okviri vrate, uredaj pretpostavlja da nije došlo do prekida prstena (ili da je eventualni problem otklonjen) i počinje da šalje CT okvire kao i ranije. Ako se signalni okviri ne vrate za odredeno vreme, uredaj zaključuje da je negde došlo do prekida i podnosi izveštaj o grešci višem sloju u protokolu. Ako uredaj primi signalne okvire od drugog uredaja, on suspenduje slanje sopstvenih okvira i ponavlja one koje je primio. Eventualno, ako postoji prekid, jedini uredaj koji šalje signalne okvire (slika 9.29) je onaj koji se nalazi ispod prekida.



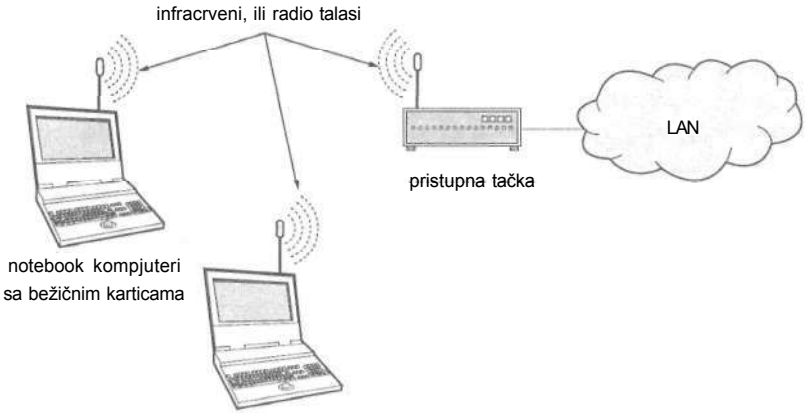
SLIKA 9.29 *Lociranje prekida na prstenu*

Održavanje protokola neophodnih za token ring mreže zasigurno je nedostatak kada se token ring poredi sa Ethernetom. Sa druge strane, kontrolisani pristup medijumu preko tokena eliminiše sve kolizije i zato čini token ring mreže popularnom opcijom u ranom periodu umrežavanja. Ovo se pokazalo kao tačno kada su neočekivana kašnjenja izazivala preterane kolizije, koje su stvarale ozbiljne probleme. Osim toga, token ring mreže su omogućile prenose zasnovane na prioritetima - prednost za administratore mreža kojima je odgovaralo uspostavljanje prioriteta u mrežnom saobraćaju. Ipak, napredak tehnologija i mrežnog dizajna redukovao je (i u nekim slučajevima eliminisao) kolizije u Ethernet okruženjima. Osim toga, bitske brzine na kojima Ethernet može da funkcioniše obezbeduju dovoljan propusni opseg, koji u većini slučajeva omogućava pravovremenu isporuku okvira na njihova odredišta. Zbog toga, iako i dalje postoje neke token ring mreže, Ethernet tehnologije su, ipak, postale dominantne u LAN okruženjima u poslednjih nekoliko godina.

9.7 Bežične mreže: IEEE standard 802.11

Do sada smo u ovom poglavlju predstavljali razne načine za povezivanje uredaja u LAN mrežama, ali postoji još jedna opcija - nema potrebe povezivati ih, bar ne u fizičkom smislu. Naravno, govorimo o bežičnim komunikacijama. Većina Ijudi koja koristi televiziju, radio i satelite zna da je to uobičajena opcija i da je neophodna u komunikacijama na velikim rastojanjima. Bežične tehnologije su se proširile i na LAN okruženja i zato je IEEE razvio **802.11** standard za bežični **LAN** (VVLAN - wireless LAN).

Konceptualno, princip je jednostavan: na slici 9.30 prikazano je tipično uredenje. Uredaji kao što su notebook kompjuteri sadrže bežične mrežne kartice koje mogu da šalju i primaju ili radio, ili infracrvene talase. Talasi se prostiru kroz slobodni prostor izmedu uredaja i na taj način je omogućena komunikacija između njih. Takode, moguće je uspostaviti komunikaciju sa kabliranom mrežom pomoću uredaja koji se označava kao **pristupna tačka** (AP-access point).



SLIKA 9.30 *Bežične WLAN konekcije*

Ima iste mogućnosti za slanje i prijem infracrvenih, ili radio talasa, ali ima i fizičku konekciju sa LAN mrežom. Preko AP uređaja, notebook kompjuteri imaju pristup bilo kom serveru, štampaču, ili drugom uredaju na LAN mreži. Prednost je što korisnik notebook kompjutera ne mora da traži zidne utičnice, niti da se bavi kablovima. Jednostavno, dovoljno je da pronade pogodno mesto u dometu pristupne tačke i može da obavi svoj posao. Nedostatak bežičnih LAN prenosa je što su bitske brzine u opštem slučaju niže nego kod LAN mreža sa fizičkim konekcijama. Videćete da postoje i neki problemi kod kontrole pristupa medijumu koji ne postoje kod Etherneta, ili token ring mreža.

Infracrveni i radio talasi

Bežični LAN koristi dva oblika komunikacija: infracrvene i radio talase. Kao što je objašnjeno u Poglavlju *2,* infracrveni i radio talasi su deo elektromagnetnog talasnog speklra. Frekvencije infracrvenih talasa nalazi se samo ispod vidljivog svetla, gde radio talasi imaju manje frekvencije. Infracrveni uređaji su opremljeni sa LED i laserskim diodama koje emituju infracrvene svetlosne talase. Ovi talasi mogu da se usmere direktno ka prijemniku (point-to-point), ili mogu da se reflektuju od zidova, ili tavanice (difuzno). Implementiranje point-to-point infracrvenog LAN-a je mnogo teže, jer predajni i prijemni uredaji moraju da budu poravnati. Sa dmge strane, difuzni infracrveni LAN može da koristi reflektivna svojstva infracrvenih talasa. Prijemnik u pristupnoj tački može da se fokusira na zid, ili tavanicu. Uređaj koji prenosi infracrveni signal je usmeren na zid, ili tavanicu, odakle se talasi reflektuju i eventualno dopiru do pristupne tačke. To je slično daljinskom upravljaču za uključivanje televizora, ili promenu kanala. Nije neophodno da se daljinski upravljač usmeri ka TV-u, već može da se odbija od plafona, ili zida iza Vas. Ovo je izuzetno pogodno ako volite da vežbate dok nazimenično pratite svoju omiljenu seriju i slušate spotove sa MTV-a.

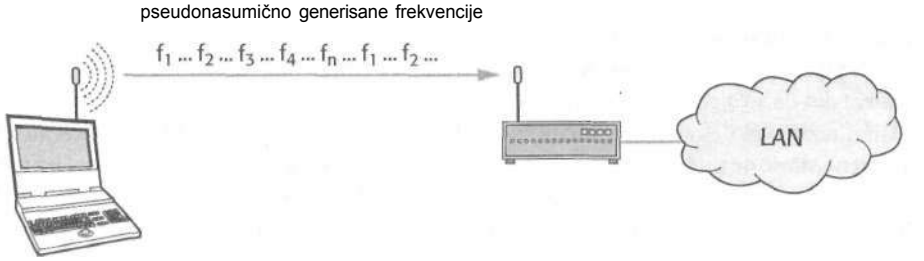
Postoje neke prednosti infracrvenih sistema. Na primer, infracrvene signale nije regulisao FCC kao što je to slučaj sa radio signalima. To znači da za opremu koja se zasniva na infracrvenim talasima nisu neophodne licence. Sledeća prednost je što infracrveni talasi ne prodiru kroz čvrste objekte, tako da su sigurniji od neovlašćenog prisluškivanja spolja. Zahvaljujući ovom svojstvu, uredaji u različitim delovima zgrade mogu da koriste isti infracrveni signal bez interferenci. Na infracrvene talase ne utiče radio interferenca, mada jaka sunčesva svetlost, ili izvori toplote mogu da utiču na njih. Medutim, nemogućnost prodiranja kroz čvrste objekte je nedostatak ako aplikacija zahteva komunikaciju koja prelazi granice čvrstih objekata.

Uredaji koji koriste radio talase ugrađeni su u predajnike i antene. Ipak, korišćenje radio talasa je nešto složenije nego što ste mogli da pomislile. Razlikuje se od konvencionalnog radija, kod koga se signal emituje na datoj nosećoj frekvenciji i Vi podešavate radio aparat za njegov prijem. Možda većznate da postoje problemi kada se koristi jedna noseća frekvencija. Jedan problem je interferenca od drugih uređaja. Ako ste nekada slušali uredaj blizu nekog jakog uređaja, kao što je motor, znate u čemu je problem. Isključite motor i slušaćete radio bez smetnji. Još ozbiljniji problem je što neko može namerno da emituje signale na odredenoj frekvenciji koji ometaju prenos. To može da ima ozbiljne posledice kod vojnih radio komunikacija koje se eventualno koriste za kontrolu naoružanja. Sledeći problem je što je neautorizovanim "uljezima" lakše da presretnu signale. Dovoljno je da se podese na odgovarajuću noseću frekvenciju i da slušaju. Tu ne možete ništa da uradite.

Da bi se rešili ovi problemi, 802.11 standard koristi široki spektar (spread spectrum), tehnologi-ju koja se koristi ne samo za bežične LAN mreže, već i za bežične i mobilne telefone. Umesto da se koristi uski frekventni opseg, prenosi u širokom spektru emituju energiju signala preko šireg opsega frekvencija (tj. većeg propusnog opsega). To ih čini manje sklonim interferencama, koje obično utiču samo na manji broj frekvencija. Osim toga, obezbedena je bolja zaštita. "Uljez" koji prisluškuje na odredenoj frekvenciji dobija samo manji deo signala, koji mu izgleda kao šum. Ovo je interesantan koncept, ali se nameće pitanje kako je moguće proširiti prenos preko širokog opsega frekvencija. 802.11 standard definiše dva tipa tehnologija širokog spektra za WLAN fizički sloj: directe-sequnce spread spectrum i frequnecy-hopping spread spectrum.

Principi na kojima se zasniva FHSS (frequency-hopping spread spectrum) tehnologija razvijeni su još početkom Drugog svetskog rata; patenti su priznati Georgu Antheilu i Hedy Keisler Markey (poznatijoj kao Hedy Lammar).* Uredaj koji koristi FHSS defmiše niz frekvencija /,, $/_2$, /,, ..., $/_n$ koje se sve nalaze u emisionom opsegu (slika 9.31). Uredaj za prenos u odredenom periodu koristi frekvenciju/!, pa prelazi na frekvenciju/$_2$.

■ Ako ste ljubitelj starih filmova, ili muzike, prepoznaćete imena Hedy Lammar i George Antheil. Ona je glumila u brojnim filmovima tokom 30-ih i 40-ih godina prošlog veka, a on je bio kompozitor. Interesantno je da su oni izumili šemu za kontrolu torpeda na velikim udaljenostima na takav način da ih neprijatelj nije mogao delektovati, ili ometati prenose. Šema je kasnije prerasla u oblik komunikacija širokog spektra.



SLIKA 9.31 *FHSSprenos*

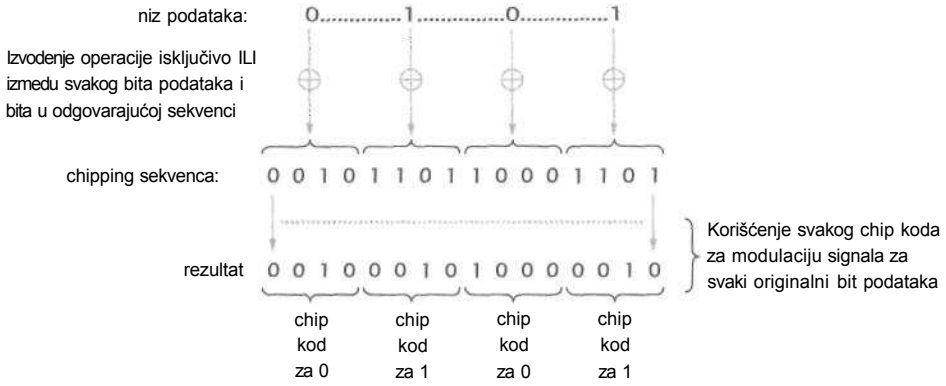
I na toj drugoj frekvenciji vrši prenos za određeni period i onda prelazi na /3. Ovaj šablon se nastavlja sve dok uređaj vrši prenos preko frekvencija u fiksnom periodu. Kada završi prenos preko frekvencije /n, ponovo počinje od /,. Svako ko pokuša da prisluškuje na konkretnoj frekvenciji čuće periodične nailaske signala koji se razlikuju od šuma. Ukoliko zna sekvencu frekvencija i trenutkeza prebacivanje frekvencija, "uljez" neće moći da presretne, ili omete signal.

Uređaj utvrduje ove frekvencije pomoću pseudonasumičnog generatora brojeva. Naime, koristi se algoritam sa inicijalnom vrednošću (inicijalna ulazna vrednost algoritma) za generisanje sekvence frekventnih vrednosti. Pošto formula koja se koristi za frekvencije nije u potpunosti nasumična, dobar generator daje brojeve koji imaju brojne karakteristike istinski nasumičnih vrednosti. Prijemni uređaj koristi isti algoritam i generiše isti set frekvencija. Tako je moguće podešavanje na odgovarajuću frekvenciju i prelazak na sledeću kada dode do sledeće frekvencije.

Za razliku od konvencionalnih radio emisija, FHSS prenos ne zahteva licenciranje kod FCC, sve dok je jačina signala manja od 1 vata, koji je dovoljan za komunikaciju na manjim rastojanjima. FHSS komunikacije za bežične LAN mreže obično funkcionišu između 2,4 i 2,483 GHz i koriste do 79 zasebnih kanala, čime su obezbeđena 22 različita šablona (redosled frekvencija). Učestalost sa kojom uredaj menja frekvencije menja se u skladu sa komunikacionim polisama uspostavljenim u konkretnoj zemlji. U Sjedinjenim Američkim Državama uredaji moraju da menjaju frekevencije bar 2,5 puta u sekundi.

DSSS (direct-sequence spread spectrum) tehnologija funkcioniše na nešto drugačiji način. U kraćim periodima FHSS koristi uskopojasne prenose, ali u dužim periodima koristi široki propusni opseg. DSSS proširuje jedan bit podataka na više njih. Zahvaljujud tome, predajnik funkcioniše na većim bitskim brzinama, tako da se signal prostire preko šireg propusnog opsega. Sledeći koraci ilustruju kako ovo funkcioniše.

1. Predajni uredaj startuje sa stringom podataka.
2. Za svaki bit podataka generiše pseudonasumično izabrani niz bitova, nazvan **chipping sekvenca,** sa n bitova.
3. Kombinuje svaki bit podataka i odgovarajuću chipping sekvencu za kreiranje *chip koda* dužine n bitova. Na slici 9.32 prikazano je kako se to izvodi sa n = 4 bita.



SLIKA 9.32 *DSSS prenos*

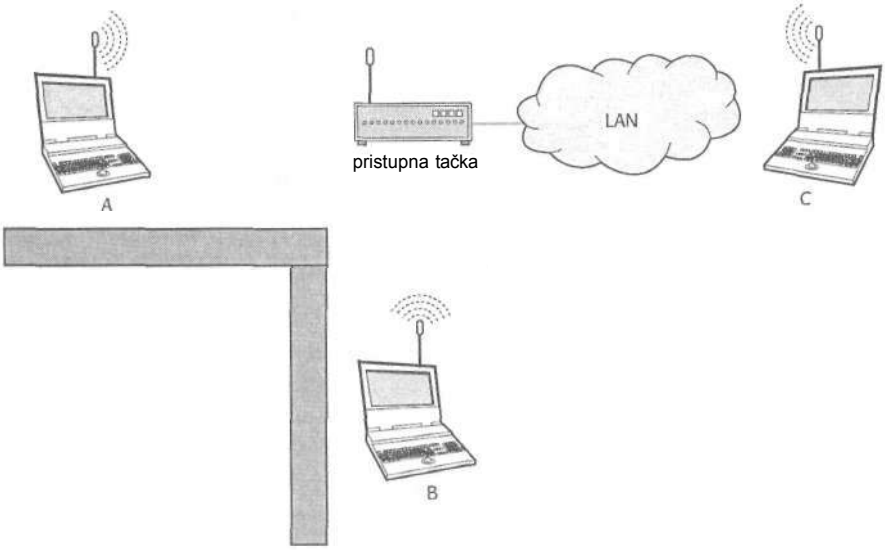
Proces izvodi operaciju isključivo ILI izmedu bita podataka i svakog bita u chipping sekvenci. Rezultat je sekvenca 4-bitnih chip kodova, po jedan za svaki bit. U ovom slučaju svaki 4-bitni chip kod odgovara 4-bitnoj grupi nasumičnog šablona, ako je bit podataka bio 0, i predstavlja komplement 4-bitne grupe nasumičnog šablona, ako je bit podataka bio 1. 802.11 standard koristi 11-bitnu chipping sekvencu koja se naziva *Barkerov kod* za generisanje chip kodova.

Ovde je značajno napomenuti da predajnik mora da prenese n bitova za svaki bit podataka. Na primer, da bi se postigla bitska brzina od 1 Mbps, uredaj mora da ima mogućnost prenosa n Mbps. Sa druge strane, veća bitska brzina zahteva signale iz većeg propusnog opsega. Tu se dešava rasipanje signala preko širokog opsega. 802.11 standard primenjuje binarni metod faznog pomeranja za chip kod kako bi se postigla bitska brzina od I Mbps i kvadraturni metod faznog pomeraja za postizanje bitske brzine od 2 Mbps. Nakon toga, signal se moduliše na nosećoj frekvenciji u opsegu 2,4 do 2,483 GHz pre prenosa. Specifični detalji zavise od šema modulacije; dodatne detalje možete da pronadete u referencama [ScOO] i [Du03].

Nadmetanje"

Pošto smo predstavili komunikaciju bežičnih uredaja, postavlja se logično pitanje šta se dešava ako dva, ili više uredaja pokušaju istovremeno da iniciraju prenos. Pošto oba uredaja dele isti medijum (slobodni prostor), doći će do kolizije njihovih signala. Ovo je isti problem sa kojim su se suočili i kreatori Ethernet tehnologija i delovalo je logično da se iskoristi isto rešenje: CSMA/CD. Medutim, postojao je problem. CSMA/CD je radio kod Etherneta zato što se moglo pretpostaviti da uredaj rnože da osluškuje da li postoji saobračaj na medijumu i da može da detektuje kolizije ako oba uredaja istovremeno iniciraju prenos. Takve pretpostavke nisu moguće kod WLAN-a.

Na slici 9.33 opisano je ono što nazivamo problem *skrivene stanice* i pokazano je šta može da se desi. Dva uredaja (A i B) šalju informacije do pristupne tačke, koristeći infracrvene talase. Ako postoji čvrsta prepreka (zid) između njih, ni jedan uredaj ne može da registruje signal onog drugog.



SLIKA 9.33 *Nedetektovane kolizije na WLAN-U*

Znači, ne može da se koristi CSMA/CD. Jedno moguće rešenje je da se ne koriste infracrveni talasi, već da se ostane na radio talasima, jer oni nemaju ograničenja zbog čvrstih objekata. Medutim, to i dalje neće funkcionisati zbog drugog scenarija koji postavlja svaki uređaj na suprotne strane pristupne tačke (uredaji A i C). Iako se pristupna tačka nalazi u dometu A i C signala, uređaji A i C mogu da budu toliko udaljeni da njihovi signali ne mogu da dopru jedni do drugih. U tom slučaju, ni jedan uredaj ne može da detektuje kolizije, tako da CSMA/CD opet ne može da posluži.

Zato je za WLAN neophodno alternativno rešenje. 802.11 standard uključuje protokol MAC sloja pod nazivom Distributed Coordination Function (DCF),* koji implementira **Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA).** Ipak, ni ovaj protokol ne može da izbegne sve kolizije, ali ih bar značajno redukuje. Protokol funkcioniše na sledeći način:

1. Izvorni uredaj treba da pošalje okvir sa podacima do neke odredišne tačke. Prvo, osluškuje medijum. Ako je medijum zauzet, čeka da se oslobodi i koristi strategiju upornosti onako kako je opisano u odeljku 4.7. Ako izvorni uredaj registruje slobodan medijum, čeka dodatni period. Ovo dodatno vreme se koristi za prioritizaciju aktivnosti. Dodatno vreme čekanja je definisano sa SIFS (short interframe space) i DIFS (DFC interframe space), koji je duži od SIFS.

*Postoji i drugi metod pristupa poznat kao Point Coordination Function (PCF), koji se koristi kada su uključena i vremenska ograničenja. Više detalja možete da pronađete u referencama [Fo03] i [ScOO].

Kada se uređaj bori za okvir medijuma, on čeka na DIFS (uskoro ćemo videti zašto). Kada dočeka DIFS, ako je medijum i dalje slobodan, on šalje RTS (Recjues to Send) okvir do odredišne tačke. Kao što sam naziv ukazuje, izvor traži dozvolu od odredišta za slanje okvira. RTS okvir uključuje trajanje perioda koji je uredaju neophodan za slanje. Uskoro ćemo videti kako se to koristi.

2. Odredišna tačka prima RTS okvir i mora da reaguje slanjem CTS (Clear to Send) okvira nazad do izvornog uredaja. Medutim, mora i da se "nadmeće" za medijum, jer u borbu za medijum mogu da budu uključeni i drugi izvori. Sećate se, drugi uredaji možda nisu registrovali poslati RTS okvir. Odredište nastavlja onako kako je opisano u prethodnom koraku, osim što, kada detektuje da je medijum slobodan, čeka na SIFS. Ideja je da ako daigi uredaj pokušava da pošalje svoj RTS okvir, on će biti primoran da čeka na DIFS. Pošto se na SIFS čeka kraće, odredište dobija viši prioritet u odnosu na ostale uredaje koji pokušavaju da iniciraju prenos. CTS okvir uključuje i dužinu perioda definisanog u RTS okviru.

3. Kada izvorni uredaj dobije CTS okvir, primio je dozvolu za slanje podataka. Sada, pre nego što nastavimo, odgovorimo na logično pitanje šta se dešava ako dode do kolizije dva RTS okvira. Ovo može da se desi zato što dva uređaja mogu istovremeno da izvedu prethodno opisani korak 1 - oba moraju da pošalju RTS okvir, a ni jedan nije detektovao zahtev ovog dmgog, niti rezultujuću koliziju. Međutim, odredište to detektuje i zato ne šalje CTS okvir. Pošto ni jedan uredaj u naznačenom periodu ne dobija CTS okvir, oba pretpostavljaju da je došlo do kolizije, ćekaju nasumično izabrani period i ponovo pokušavaju.

   Iako drugi uredaji možda neće registrovati RTS okvir, svaki uređaj u dometu odredišta registmje CTS okvir i zna da je neki drugi uređaj dobio dozvolu za slanje. Zato se svi uzdržavaju od pokušaja da pristupe medijumu radi slanja svog RTS okvira. Drugim rečima, uredaj kome je potvrdena dozvola za slanje ima ekskluzivnu kontrolu nad medijumom u odredenom periodu. Osim toga, pošto CTS okvir sadrži dužinu perioda, svi drugi uređaji znaju koliko moraju da čekaju.

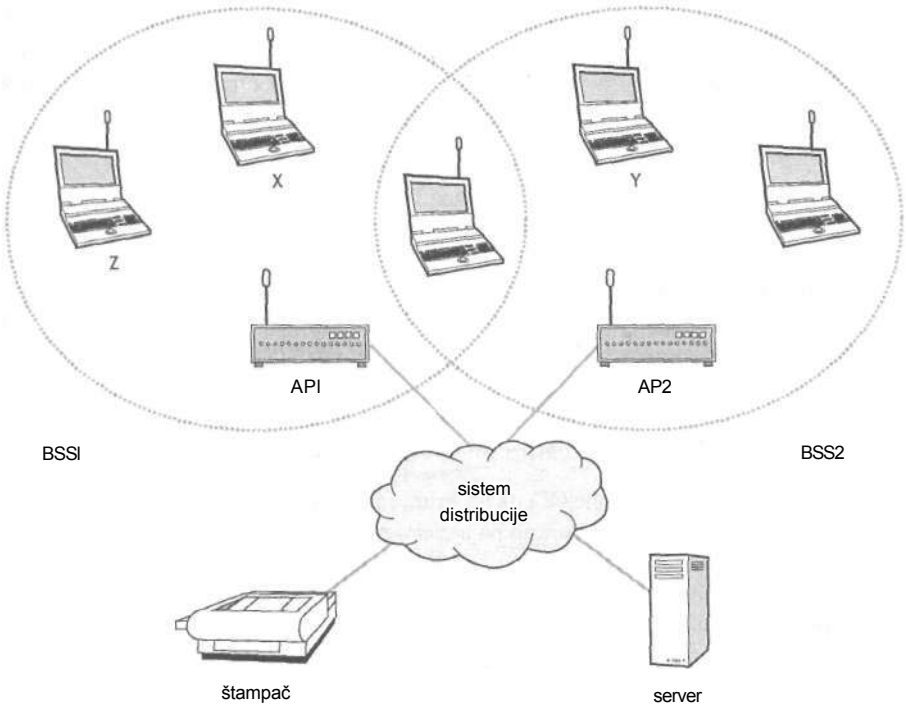
4. Kada uredaj primi CTS okvir, on šalje svoj okvir sa podacima. Ovde se primenjuje izbegavanje kolizije. Pošto su drugi uredaji svesni neposrednog prenosa i znaju koliko još moraju da čekaju, ne bi trebalo da dode do bilo kakvih kolizija.

5. Kada odredište primi okvir sa podacima, vraća ACK okvir. Tako izvor zna da su podaci primljeni.

6. Kada istekne definisani period, svi uređaji ponovo počinju da se bore za pristup medijumu.

Adresiranje

802.11 protokol, kao i drugi protokoli, ima tačno definisan format okvira. Međutim, pre nego što dođemo do toga, moramo bolje da razumemo mehanizam adresiranja za 802.11, koji definiše četiri različita adresna polja za svaki okvir. Zašto četiri? Da bismo dali odgovor na ovo pitanje, najpre moramo da objasnimo komponente koje sačinjavaju bežični LAN. Na slici 9.34 prikazana je jedna moguća konfiguracija.



SLIKA 9.34 *Infrastruktura bežičnog LAN-a*

Skup bežičnih uredaja sa jednim AP uredajem označen je kao **BSS** (basic service set - osnovni servisni skup).* Pošto je moguće da postoji više AP uređaja, moguće je da postoji i više BSS-ova. Osirn toga, više BSS-ova može da se poveže preko sistema distribucije (DS - distribution system). Standard 802,11 ne defmiše arhitekturu DS-a. DS može da bude LAN, kolekcija LAN mreža, ili neki drugi tip mreže (uključujući i bežičnu), ali u svim slučajevima omogućava komunikaciju jednog uređaja u BSS-u sa jednim uređajem iz drugog BSS-a, ili omogućava pristup serverima, štampačima i tako dalje. Kada uredaj pošalje okvir, postoje četiri različite mogućnosti. Prikazane su u tabeli 9.9, u kontekstu slike 9.34; naznačeno je kako su definisana adresna polja.

Zbog različitih mogućih scenarija, 802.11 okvir ima četiri adresna polja, čije vrednosti zavise od konkretnog scenarija. Ova polja su neophodna da bi se razlikovao izvor od predajnika prijemnik od odredišla.

* Tehnički, definicija BSS-a je malo šira i uključuje sve uredaje koji se "nadmeću" za pristup istom medijumu (tj. koriste istu radio frekvenciju). Medutim, za naše svrhe ova definicija je sasvim dovoljna.

Tabela 9.9: Adresna polja u 802.11 okviru

Slučaj	Opis	Addressl (prijemnik)	Address2 (predainik)	Address3	Address4
1	X šalje okvir do Z. Okvir ostaje u sklopu BSSI.	Z	X	BSS1	
2	X šalje okvir do Y. Okvir prvo ide do API.	API	X	Y	
3	API šalje okvir koji potiče od X do AP2 preko bežičnog DS-a. Okvir je namenjen uredaju Y.	AP2	API	Y	X
4	AP2 šalje okvir koji potiče iz X do Y.		Y	AP2	X -

U prvom slučaju našeg primera X šalje okvir do Z. Oba se nalaze u istom BSS-u i AP nije uključen u prenos. Addressl označava odredište okvira, a Address2 označava izvor. Ovde nismo pravili razliku izmedu izvora i predajnika (oba su X), niti izmedu prijemnika i odredišta (oba su Z). Address3 jednostavno predstavlja ID BSS-a, a Address4 se ne koristi.

U drugom slučaju X šalje okvir narnenjen uređaju Y. Pošto se X i Y nalaze u različitim BSS-ovima, protokol je složeniji. U ovom slučaju Addressl definiše APl, a Adress2, kao i ranije, označava izvor. Razlika je u tome što ovde prijemnik (AP2) nije i eventualno odredište. Polje Address3 definiše Y kao odredište. Ovo je neophodno kako bi APl rutirao okvir.

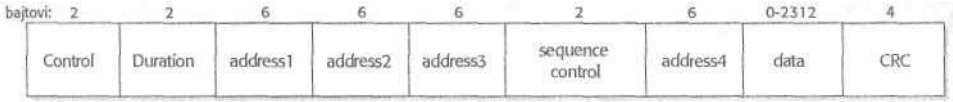
Treći slučaj se primenjuje ako sistem distribucije sledi i bežični standard. U tom slučaju, APl (polje Address2) prenosi okvir, a AP2 (polje Addressl) treba da primi taj okvir. Medutim, ni jedan od njih nije niti originalni izvor, niti eventualno odredište. Polja Address3 i Address4 naznačavaju X i Y, respektivno.

Poslednji slučaj se primenjuje kada okvir napušta DS dok ga AP2 prenosi do Y. U tom slučaju, odredište i prijemnik su isti (Y), ali su predajnik (AP2) i izvor (X) različiti.

Sva četiri slučaja imaju nešto zajedničko. Addressl uvek definiše uredaj koji treba da primi okvir. Dakle, dok uredaj prati radio signale od kojih se okvir sastoji, dovoljno je samo da proveri polje Addressl kako bi video da li treba da reaguje. Drugo, polje Address2 uvek definiše uredaj koji prenosi okvir. Ovo je neophodno zbog ACK okvira koji treba da se pošalje nakon uspešnog prijema. Prijemni uredaj mora da zna gde da pošalje ACK okvir. Glavne razlike se ogledaju u sadržaju polja Address3 i Address4.

Format okvira

Na slici 9.35 prikazani su sadržaj 802.11 okvira i postavka četiri adresna polja koja smo malopre opisali. Polje Duration ukazuje na period u kome se odvija prethodno opisana razmena RTS/CTS okvira. Polje Data je jasno samo po sebi, a polje CRC se koristi za kontrolu grešaka.



SLIKA 9.35 *802.11 okvir*

Polje Sequence Control sadrži sekvencu brojeva koji se koriste za kontrolu toka. Dakle, jedino polje koje je potrebno detaljnije razmotriti je Control.

Polje Control je dugačko dva bajta i sadrži sledeće elemente:*

- To **DS bit** Ovaj bit se postavlja kada okvir ide ka DS-U. Fleg je 1 u slučajevima 2 i 3 iz tabele 9.9, a O u suprotnom.
- From **DS bit** Ovaj bit se postavlja kada okvir dolazi sa DS-a. Fleg je 1 u slučajevima 3 i 4 iz tabele 9.9, a O u suprotnom. Ovaj i prethodni fleg određuju način na koji protokol interpretira adresna polja.
- **More Fragment bit** Bežični prenosi su skloniji greškama nego električne, ili optičke komunikacije. Zbog toga, standard 802.11 dopušta smanjenje maksimalne veličine okvira ako je učestalost pojave grešaka velika. To se radi zbog toga što je oštećenje manjeg okvira manje neefikasno nego oštećenje većeg okvira, jer je potrebno ponoviti prenos manje količine podataka. To stvara probleme ako količina korisničkih podataka nalaže korišćenje okvira veličine koja premašuje maksimalnu vrednost. Medutim, protokol dopušta deljenje okvira na fragmente. Fragmenti mogu da se šalju nezavisno, a zatim se na odredištu sastavljaju kako bi se ponovo kreirao originalni okvir. More Fragment bit je postavljen kada je tekući okvir podeljen na fragmente i kada sledi više fragmenata.
- **Retry bit** Ovaj fleg se postavlja kada se okvir prenosi drugi put (na primer, zbog toga što nije stigla potvrda).
- **Dvobitni broj verzije** Definiše tekuću verziju korišćenog protokola.
- **Dvobitno polje Type i 4-bitno polje Subtype 2-bitno polje** Type označava da li tekud okvir sadrži podatke, kontrolne informacije, ili Je reč **o** upravljačkom okviru. Polje Subtype dalje artikuliše namenu okvira. Na primer, 2-bitni kod Ol ukazuje na kontrolni okvir. Kontrolni okviri se koriste za pristup medijumu.

* Ovde nismo naveli sve elemente polja Control, jer neki od njih zahtevaju detaljnija objašnjenja standarda nego što smo mi obezbedili. Više detalja možete da pronadete u referencama [ScOO], [Fo03],i [Be02].

U primere mogu da se svrstaju RTS, CTS i ACK okviri. Polja Subtype omogućavaju razlikovanje okvira. Okviri sa podacima su jasni sami po sebi. Upravljački okviri se koriste za konfigurisanje mreže.

Slede neki primeri korišćenja upravljačkih okvira:

- **Konfiguracija BSS-a** Sećate se sa slike 9.34 da je moguće da se dva BSS-a preklapaju. To znači da uredaj mora da se pridruži AP uredaju ako mu je neophodan pristup sistemu distribucije. Da bi se to izvelo, uredaj šalje Assodation Request okvir do naznačenog AP uredaja. Ako AP prihvati pridruživanje, reaguje slanjem okvira Association Response.

- **Pronalaženje AP uredaja** Kako uredaj zna koji AP uredaji postoje? Da bi detektovao i identifikovao AP uredaje, svaki uredaj šalje Probe Request okvir. Svaki AP detektuje okvir i reaguje slanjem Probe Response okvira. llredaj koji je inicirao ispitivanje nakon toga može da selektuje AP (možda onaj sa najjačim signalom) i može da mu se pridruž, kao što je opisano u prethodnom koraku.

- **Roaming** Uredaji su mobilni, što znači da mogu da prelaze iz jednog BSS-a u drugi. Kada se to desi, uređaj može da pošalje Reassociate Request okvir do novog AP uređaja. AP može da reaguje sa Reassociate Response okvirom. Disassociation okvir razdružuje uredaj od prethodno pridruženog AP-a. Medutim, mobilni uredaj se možda više ne nalazi u dometu starog AP-a. Zato je novi AP odgovoran za slanje Disassociation okvira do starog AP-a u ime mobilnog uredaja. Ponovno pridruživanje treba da se izvrši automatski. Ako se uredaj pomeri i detektuje oslabljeni signal iz postojećeg AP-a, mora da pronade AP sa jačim signalom i da mu se pridruži. Ovo je analogno funkcionisanju mobilnih telefona kada korisnik prelazi iz jedne ćelije u drugu, osim što je ovde reč o mnogo manjim razmerama.

- **Bezbedne komunikacije** Uredaj može da se uključi u bezbednu komunikaciju slanjem Authenticate okvira. Nakon toga, dva uređaja mogu da razmenjuju informacije u skladu sa pravilima bezbednih komunikacija. Kada se bezbedna komunikacija završi, okončava se slanjem Deauthentication okvira.

Wired Equivalent Privacy (WEP) protokol

Kao što se moglo i pretpostaviti, bezbeđnost je problem bežičnih okruženja, posebno kada se koriste niskofrekventni radio talasi. U takvim slučajevima, prednost "povezivanja" uredaja razdvojenih fizičkim barijerama može da postane ozbiljan bezbednosni problem. Zbog prirode radio talasa male snage, WLAN je verovatno sigurniji ako se nalazi na farmi koja je jedini objekat na području od 10.000 jutara. Međutim, ako se WLAN postavlja u stanu, ili kancelariji u zgradi koja se nalazi u gusto naseljenom područuju, osetljiv je na napade. Projekat na kome je radio Peter Shipley (posetite www.dis.org/filez/openlans.pdf) prikazuje eksperiment u okviru koga su članovi tima, koristeći najnoviji hardver, mogli da lociraju WLAN jednostavno vozeći se kroz susedstvo i pokušavajući da detektuju signal. Zaključili su da je moguće uspostaviti konekciju sa vrhova brda, ili visoke zgrade sa mrežom koja je udaljena više od 20 milja. Većina tih mreža uopšte nije implementirala bezbednosni protokol.

802.11 standard uključuje bezbednosni protokol poznat pod nazivom Wired Equivalent Privacy (WEP). Ovde ćemo navesti samo neke značajne karakteristike, a zainteresovani čitaoci mogu da pronadu više detalja u referenci [Sw03].

- WEP obezbeduje autentifikaciju i šifrovanje komunikacije između uredaja i AP-a. Algoritam za šifrovanje koristi 40-bitni tajni ključ i dodaje 24-bitni vektor inicijalizacije kako bi se kreirao 64-bitni ključ. Za svaki okvir se koristi drugačiji vektor inicijalizacije, tako da se dobijaju različiti ključevi šifrovanja svakog okvira. Novija oprema omogućava šifrovanje i sa 128-bitnim ključem.
- Protokol ne definiše algoritam koji se koristi za razmenu ključa. Pretpostavlja se da su ključevi ugovoreni pre bilo kakve razmene. U okviru Shipleyevog projekta, skoro polovina WLAN mreža koja je implementirala WEP koristi podrazumevani ključ za šifrovanje.
- Metod za šifrovanje koristi algoritam poznat kao RC4 (videti referencu [Sc95]). 40-bitni ključ i 24-bitni vektor inicijalizacije su ulazne vrednosti za algoritam koji generiše sekvencu pseudonasumičnih ključeva. Svaki od tih ključeva se propušta kroz operacije isključivo ILI sa sekcijom običnog teksta iz okvira. Svaki vektor inicijalizacije je poslat u sklopu okvira (u originalnoj formi). Prijemni uredaj koristi vektor inicijalizacije i 40-bitni ključ za generisanje iste sekvence ključeva i izvodi se operacija isključivo ILI sa dolazećim šifrovanim tekstom za ponovno kreiranje originalnog teksta.

Grupa u University of California - Berkley je istakla nekoliko propusta u WEP šifrovanju (posetite www.isaac.cs.berkley.edu/isaac/wep-faq.html). Na primer, mali 24-bitni vektor inicijalizacije znači da postoji veća verovatnoca da će 64-bitni ključ, a, samim tim, i sekvenca ključeva, biti ponovljeni u uslovima gustog saobraćaja. "Uljez" može da presretne dva, ili više okvira šifrovanih istim ključem I da koristi statističke tehnike (često prvi korak u "razbijanju" koda) za analiziranje uzoraka. Referenca [Ar02] ukazuje na slabost standarda 802.11 u pogledu zaštite i ističe neke moguće pristupe za rešavanje tih problema.

Varijacije standarda 802.11

Originalni 802.11 standard koristi jedan od tri fizička sloja: infracrveni, DSSS, ili FHSS. Svi mogu da se izvršavaju brzinom od 1 Mbps, Oi 2 Mbps, što je veoma sporo u poređenju sa današnjim protokolima za kablirana LAN okruženja. Kao i u slučaju Ethernet standarda, postoji nekoliko revizija standarda 802.11 kako bi se obezbedile brže komunikacije. To su 802.11a, 802.11b i 802.Hg. Sve su zasnovane na CSMA/CA i imaju velike razlike u fizičkim slojevima.

Na primer, 802.11b, koji je poznat i kao Wi-Fi (skraćenica za *wirelless fidelity),* koristi radio talase iz opsega od 2,4 GHz i postiže brzine od 11 Mbps. Koristi samo DSSS, jer veće brzine ne mogu da se postignu sa FHSS. Sledeća razlika se ogleda u šemi modulacije. Originalni standard 802.11 obično koristi PSK modulaciju, a 802.11b koristi CCK (complementary code keying) metod.

Poredenja radi, standard 802.11a prenosi okvire u opsegu od 5 GHz i koristi oblik multipleksiranja sa deljenjem frekvencije, umesto tehnologija širokog spektra. Teorijski, bitska brzina iznosi 54 Mbps, ali brzine koje se postižu u praksi ne prelaze 30 Mbps. Medutim, veće brzine mogu da budu skupe na kraćim rastojanjima. Opseg od 5 GHz je značajan, jer mnogi uređaji (mobilni telefoni, Bluetooth uređaji, čak i mikrotalasne pećnice) većkoriste frekvendje iz opsega od 2,4 GHz. Sa druge strane, neke vojne aplikacije koriste opseg od 5 GHz.

Na sledećem standardu se još uvek radi (u vreme kada je ova knjiga pripremana) - to je 802.1 Ig. Dizajniran je za funkcionisanje u opsegu od 2,4 GHz i ima bitsku brzinu od 54 Mbps. Dati su predlozi za korišćenje šeme multipleksiranja sa deljenjem frekvencije za veće bitske brzine i CCK šeme zbog obezbedivanja kompatibilnosti sa prethodnim standardom 802.11b. Međutim, 802.1 Ig još uvek nije ratifikovan kao zvanični standard.

Ovaj odeljak ne sadrži kompletan opis bežičnog LAN protokola. Reč je o veoma složenom protokolu. Nismo naveli ništa više od opšteg pregleda najznačajnijih tema; postoji još mnogo toga što može da se kaže o ovom protokolu (videti reference [ScOO], [Fo03] i [Be02]).

9.8 Zaključak

U ovom poglavlju smo predstavili lokalne mreže i fokusirali smo se prvenstveno na protokole sloja 1 i 2. Videli ste da sloj veze izmedu podataka sadrži dva podsloja: LLC i MAC. LLC je nezavisan od specifične lokalne mreže i defmiše point-to-point komunikacije izmedu dva uređaja u LAN okruženju. Mnogi protokoli funkcionišu na ovom sloju, mada većina koristi HDLC (High-level Data Link Control) protokol.

HDLC je primer bitovima orijentisanog protokola. Njegovi okviri se tretiraju kao nizovi bitova. Definisao ga je ISO za point-to-point, ili multipoint konekdje; koristi se za half-duplex i full-duplex komunikacije, a može da koristi ili go-back-n protokol, ili protokol selektivne retransmisije. Definiše različite tipove okvira i koristi ih za razmenu podataka, komandi, ili kontrolnih informacija.

Poredenja radi, ilustrovali smo i BSC, nešto stariji bajtovima orijentisani protokol kod koga se okviri tretiraju kao nizovi bajtova. Takode se koristi za point-to-point, ili multipoint konekcije, mada se obično koristi u half-duplex komunikacijama sa stop-and-wait protokolom za kontrolu toka. Kao i HDLC, definiše različite tipove okvira za razmenu podataka, komandi i kontrolnih informacija.

U većem delu ovog poglavlja opisani su protokoli MAC sloja i fizičkog sloja koji se koriste za implementiranje LAN mreža. IEEE je definisao ćetiri razlidta IAN standarda: 802.3 Ethernet, 802.4 token bus, 802.5 token ring i 802.11 bežični LAN. Ethemet je dominantni standard i razvijen je u nekoliko različitih verzija (Fast Ethernet, Gigabit Ethernet, 10 Gigabit Ethernet) kako bi se iskoristile prednosti novih, bržih tehnolgija. Iako se u današnje vreme token bus i token ring mreže retko koriste, predstavili smo token ring radi poredenja sa Ethernetom. Bežični standard, koji je relativno nov, eliminiše potrebu za fizičkim konekcijama, ali je sporiji od kabliranih varijanti. U tabeli 9.10 prikazan je pregled najvažnijih karakteristika ovih IAN standarda.

Tabela 9.10: Pregled LAN standarda

IEEE standard	Komentari
IEEE 802.3	Koristi 10Base5 (ThickNet) 50-omski koaksijalni kabl, prečnika 10 mm. Maksimalna dužina segmenta iznosi 500 metara. Ostale granice su četiri repetitora, oet segmenata i 100 uredaja po jednom segmentu. Rastojanje izmedu primopredajnika mora da bude najmanje osam stopa. Koristi Mančester kodiranje. Primenjuje se kod fizičkih topologija magistrale.
IEEE 802.3a	Koristi 10Base2 (CheaperNet, ThinNet) 50-omski koaksijalni kabl, prečnika 5 mm. Maksimalna dužina segmenta iznosi 185 metara. Dopuštena su najviše četiri repetitora, pet segmenata i 30 uredaja po jednom segmentu. Minimalno rastojanje izmedu čvorova iznosi 0,45 metara. I ovaj standard koristi Mančester kodiranje. Primenjuje se kod fizičkih topologija magistrale.
IEEE 802.3i	Kategorija 3, 4, ili 5 UTP kabla (lOBaseT). Centralni hub. Mančester kodiranje. Maksimalna dužina UTP kabla je 100 metara. Koristi se kod fizičke topologije zvezde.
IEEE 802.3J	Fiber sa više modova (10BaseFx). Maksimalno rastojanje iznosi 200 metara za FL i FB, a 500 metara za FP. Koristi se kod fizičke topologije zvezde, ali FP koristi pasivni sprežni element kao hub. FP se ne sreće često.
IEEE 802.3U	Koristi dva para upredenih parica kod kablova kategorije 5 (100BaseTX) i 4B/5B, iza koga sledi MLT-3 kodiranje. U domenu kolizije se koriste maksimalno dva repetitora klase II (ako se nalaze na rastojanju od pet metara, ili manje), ili jedan repetitor klase I. Maksimalna dužina segmenta je 100 metara.
IEEE 802.3U	Optički fiber sa više modova rada (lOOBaseFX). Koristi 4B/5B i NRZI kodiranje. U domenu kolizije koriste se maksimalno dva repetitora klase II, ili jedan repetitor klase I. Maksimalna dužina segmenta iznosi 136 metara ako su oba linka u istom domenu kolizije izvedena pomoću fibera, a 160 metara ako je drugi link UTP. Ako fiber povezuje dva komutatora, dužina segmenta može da bude 412 metara (half duplex), ili 2.000 metara (full duplex).
IEEE 802.3U	Koristi četiri para upredenih parica kod kablova kategorije 3 (100BaseT4). Koristi se 8B/6T kodiranje i tritovi se prenose istovremeno preko tri para upredenih parica. Između uredaja je moguće postaviti najviše dva repetitora klase II, ili jedan klase I. Ne može da se koristi u full-duplex modu. Maksimalna dužina segmenta iznosi 100 metara.
IEEE 802.3at	Zahteva četiri para upredenih parica kategorije 5 UTP kablova (1000BaseT) i pokreće se u full-duplex modu. Za signaliziranje su neophodne složene procedure kodiranja/dekodiranja: PAM5, rešetkasto i Viterbi. Maksimalna dužina segmenta iznosi 100 metara.
IEEE 802.3Z	Medijum je specijalni zaštićeni bakarni kabl (lOOOBaseCX). Maksimalno rastojanje iznosi 25 metara. Obično se koristi za povezivanje uredaja na ispitnim mestima u komunikacionom centru. Koristi 8B/10B kodiranje.
IEEE 802.3Z	Dugotalasni optički fiber (1000BaseLX). Može da se koristi ili u više modova, ili samo sa jednim modom. Maksimalne dužine iznose 5.000 metara za fiber sa jednim modom i 550 metara za više modova. Koristi se 8B/10B kodiranje.
IEEE 802.3Z	Kratkotalasni optički fiber (lOOOBaseSX). Može da se koristi sa fiberom sa više modova. Maksimalne dužine su 220 i 550 metara, u zavisnosti od prečnika fibera. Koristi 8B/10B kodiranje.

IEEE standard	Komentari
IEEE 802.3ae	10 Gigabit Ethernet. Full-duplex komunikacije isključivo preko optičkog fibera, kod kojih su kolizije eliminisane. U vreme kada je ova knjiga pripremana, 10 Gigabit Ethernet Je najverovatnije korišćen samo kod nosilaca koji obezbeduju povezanost LAN okruženja sa manjim bitskim brzinama.
IEEE 802.5	Token ring mreža. Uredaji su povezani u logički prsten, a okviri "putuju" preko njega, sledeći rutu do svojih odredišta. Uredaj ne može da prenosi podatke sve dok ne primi token (specijalni okvir koji cirkuliše prstenom kada se ne prenose nikakvi podaci). Moraju se implementirati upravljačke rutine koje nastupaju u situacijama kada dode do problema sa tokenom.
IEEE802.11	Bežični LAN. Koristi infracrvene, ili jedan od dva moguća oblika radio talasa širokog spektra. Na brzinama od 1, ili 2 Mbps WLAN je sporiji od ostalih mreža sa fizičkim konekcijama. Pošto nema garancija da će uredaji detektovati kolizije, uredaji prilikom "nadmetanja" za pristup medijumu koriste CSMA/CA.
IEEE 802.1 lx	802.1 la funkcioniše u opsegu od 5 GHz i teorijski može da postigne bitske brzine od 54 Mbps. Ipak, u praksi su postignute niže vrednosti. 802.1 Ib, poznat I kao Wi-Fi, funkcioniše u opsegu od 2,4 GHz i ima bitsku brzinu od 11 Mbps. 802.1 Ig se i dalje razvija, ali planira se da funkcioniše u opsegu od 2,4 GHz i da postigne bitske brzine od 54 Mbps.

Pitanja i zadaci za proveru

1. Navedite razliku između bitovima orijentisanog i bajtovima orijentisanog protokola.
2. Koja tri komunikaciona moda postoje kod HDLC protokola? Opišite sva tri.
3. Šta je dopunjavanje bitova i zašto je neophodno?
4. Navedite razlike izmedu primarne, sekundardne i kombinovane stanice.
5. Navedite razlike između glavnih tipova HDLC okvira.
6. Definišite četiri tipa statusa koja se mogu naznačiti u HDLC supervizorskom okvim.
7. Šta je emisiona adresa?
8. Navedite druge protokole slične HDLC-u i organizacije koje ih sponzorišu.
9. Navedite glavne razlike između HDLC i BSC protokola.
10. Šta je SYN karakter?
11. Šta je Data Link Escape karakter?
12. Navedite razliku između transparentnih i netransparentnih podataka.
13. Sta je dopunjavanje bajta?
14. Navedite razlike izmedu lokalnih i WAN mreža.
15. Navedite tipične LAN topologije.
16. Kako fizička topologija zvezde može da funkcioniše kao logička topologija magistrale?
17. Koje su glavne podele sloja veze i koje su njihove najvažnije funkcije?
18. Sta je primopredajnik?

19. Da li su sledeće konstatacije tačne, ili netačne (zašto)?
    a. HDLC je bajtovima orijentisani protokol.
    b. HDLC može da koristi ili selektivnu retransmisiju, ili go-back-n protokol.
    c. Ethernet je protokol sa sedam slojeva, slično OSI modelu.
    d. Polje Pad u Ethernet okviru je opciono.
    e. Postoji samo jedan Ethernet protokol.
    f. Kod token ring mreže uredaji naizmenično šalju okvire preko prstena.
    g. Fast Ethernet i IO Mbps Ethernet koriste iste MAC protokole.
    h. Sva tri LAN protokola omogućavaju definisanje prioriteta za uredaje.
    i. Token ring nema centralizovanu kontrolu.
    j. Svaki uredaj može da povisi, ili snizi prioritet tokena.
    k. Jedina prava razlika izmedu prenosa infracrvenim i radio talasima kod WLAN-a ogleda se u mogućim bitskim brzinama.
20. Opišite sve specifikacije 802.3 kablova.
21. Zašto Ethernet uredaj mora i dalje da prenosi okvir koji se sudario sa nekim drugim okvirom?
22. Gde se koristi T-konektor?
23. Zašto su u nekim mrežama neophodni repetitori?
24. U čemu je razlika izmedu huba i repetitora?
25. Sta je domen kolizije?
26. 10 Mbps Ethernet specifikacija koristi Mančester kodiranje. Zašto Fast Ethernet ne može da se koristi sa UTP kablovima kategorije 5?
27. Zašto se kod Fast Etherneta 4 bita menjaju sa pet bitova, čime se uvećava ukupni broj bitova koje je potrebno preneti?
28. Navedite razlike izmedu lOOBaseTX i 100BaseT4 specifikacija kod Fast Etherneta.
29. Šta je MLT-3 kodiranje?
30. Zašto se MLT-3 kodiranje ne može koristiti za 100BaseT4?
31. Fast Ethernet ima veću bitsku brzinu od originalnog Etherneta, ali segmenti moraju da budu kraći. Zašto?
32. Zašto kod Gigabit Etherneta minimalna veličina okvira mora da bude veća od minimalne veličine okvira kod prethodnih verzija?
33. Sta je navalni okvir?
34. Zašto je kod Gigabit Etherneta sa optičkim fiberom značajno sačuvati približno podjednak broj nula i jedinica za vreme istog prenosa?
35. Sta je hibrid kod Gigabit Etherneta?
36. Koja je svrha ekstenzije nosioca kod Gigabit Ethernet okvira?
37. Kako Gigabit Ethernet može da funkcioniše bez protokola za detektovanje kolizija?
38. Zašto je neophodno definisati maksimalnu i minimalnu veličinu Ethernet okvira?

39. Čemu služi token na token ring mrežama?
40. Prokomentarišite sadržaj i svrhu svakog polja u formatu tokena (za token ring mrežu).
41. Šta su non-data-J i non-data-K signali?
42. Šta je stanica za odlaganje?
43. Šta je stanica za nadgledanje (monitor)?
44. Opišite svrhu sledećih kontrolnih okvira u token ring mrežama:
    a. Active Monitor Present okvir
    b. Beacon okvir
    c. Claim Token okvir
    d. Purge okvir
    e. Standby Monitor Present okvir
    f. Duplicate Address Test okvir
45. Šta je ispušteni okvir?
46. Šta je pristupna tačka kod bežičnog LAN-a?
47. Koje su prednosti korišćenja infracrvenih talasa u odnosu na radio talase u WLAN mrežama? Koji su nedostaci?
48. Šta je tehnologija širokog spektra?
49. Pošto WLAN koristi deljeni medijum kao i originalni Ethernet, zašto ne može da se koristi protokol za detektovanje kolizija?
50. Opišite problem skrivene stanice kod WLAN mreža.
51. U čemu je razlika izmedu FHSS i DSSS?
52. U čemu je razlika između chipping sekvence i chip koda?
53. WLAN okvir sadrži četiri adresna polja. Svako od tih polja sadrži informacije koje zavise od toga kome je okvir namenjen i odakle se prenosi. Kako protokol može ispravno da interpretira sadržaj svih adresnih polja?

Vežbe

1. Uzmite za primer 10 Mbps 802.3 LAN (10Base5 kabl). Koliko maksimalno vremena uredaju može da bude neophodno za detektovanje kolizije ako je dužina segmenta 500 metara?
2. Kako izgleda MLT-3 signal za string 01001100100110? Pretpostavite da je nivo signala na početku 0.
3. Zašto 8B/6T signal izgleda kao dva bajta 42C3?
4. Zašto naizgled komplikujemo token ring protokol rutiranjem svakog bita u okviru, ili tokenu dm se primi? Zašto se ne prime svi bitovi tokena, prouče se, a zatim proslede do sledećeg uredaja?
5. Uzmite za primer 200 metara dugački 4 Mbps token ring sa 20 uredaja, koji prenose podatke sa istim prioritetom. Pretpostavite da ni jednom uredaju nije dozvoljeno da prenese više od 5.000 podataka u jednom ciklusu posedovanja tokena.

Kada uređaj prepusti token, koliko vremena može da protekne (u najgorem slučaju) dok ponovo ne dobije token?

6. U sklopu diskusije o rezervisanju tokena, istakli smo da uređaj koji povisi prioritet tokena mora kasnije i da ga snizi. Zašto uopšte snižavati prioritet tokena? Zašto ga ne ostaviti takvim kakav jeste?
7. Završite tabelu 9.7 do tačke u kojoj su svi uređaji poslali svoje okvire i kroz prsten ponovo cirkuliše token sa prioritetom O.
8. Ponovite primer opisan na slici 9.28 i u tabeli 9.7, uz pretpostavku da se token kreće suprotno od smera kretanja kazaljki na časovniku.
9. Ako su definisani prioriteti na token ringu, koliko najduže uredaj može da čeka pre nego što prisvoji token?
10. U našoj diskusiji o SMP okvirima svaki uređaj je ponavljao primljeni okvir i kasnije slao sopstveni okvir. Zašto svaki uređaj koji primi SMP okvir ne povuče taj okvir sa prstena i odmah pošalje sopstveni okvir?
11. Razmotrite algoritam sa slike 9.27. Diskutujte efekte uklanjanja pratećeg koda pod sledećim uslovima.
    a. Uslov 2
    b. Uslov 4
    c. Uslov 7
12. Napišite pseudokod logike uređaja koji prisvaja token pomoću Claim Token okvira.
13. Kako izgleda preneti binarni string nakon dopunjavanja bitova kod sledećeg stringa (najpre se prenosi krajnji levi bit)?

    `0101111110111101111111101111`
14. Ponovo iscrtajte sliku 9.8b, uz pretpostavku da B šalje SREJ (umesto REJ) kada detektuje grešku.
15. Ponovo iscrtajte sliku 9.8b, uz pretpostavku da je B poslao četiri informaciona okvira, koji su stigli u A pre nego što je A poslao drugu grupu olcvira. Takode pretpostavite da je treći okvir koji je poslat iz B oštećen u toku prenosa.
16. LJ odeljku 9.2 popunjavanje bajta je opisano kao umetanje dodatnih DLE karaktera uvek kada se u podacima javi DLE. Namera je da se izbegne pogrešno tumačenje podatka sa vrednošću DLE kao kontrolnog karaktera DLE. Zašto se isti efekat ne može postići jednostavnim korišćenjem STX karaktera pre bajtova sa podacima, a ako se u podacima javi ETX, zašto jednostavno ne umetnemo još jedan ETX karakter?
17. Napišite program koji vrši popunjavanje bajta u nizu karaktera. Napišite kompletni program koji prihvata dopunjeni niz karaktera i uklanja dopunjene bajtove.
18. Planirate da postavite WLAN u kancelarijsko okruženje koje ima dosta pregrađenih odeljenja na velikom prostoru. Ako je sve ostalo isto, da li bi infracrveni, ili radio talasi bili bolji izbor? Zašto?

19. Planirate da postavite zasebne WLAN mreže u r.izu laboratorija (po jedan WLAN za svaku laboratoriju). Veličine laboratorija su različite, ali sve su blizu jedne dmgih, u istom delu zgrade. Ako je sve ostalo isto, da li bi infracrveni, ili radio talasi bili bolji izbor? Zašto?

20. Pretpostavite da na slici 9.34 uredaj Y šalje okvir do uredaja Z. Prikažite sadržaj adresnih polja okvira dok "putuje" od Y do AP2, preko APl do Z.

21. Uređaj će prenositi bitove podataka 010010 pomoću DSSS-a. Chipping sekvenca je takozvani Barkerov kod 10110111000. Koji se bitovi, u stvari, prenose?

22. Želite da bežični uredaj može da prenosi podatke brzinom od 2 Mbps. Ako koristi Barkerov kod iz prethodnog primera, kolika mora da bude izvorna bitska brzina?

23. Dva WLAN uređaja se "nadmeću" za pristup medijumu. Jedan šalje RTS okvir radi pristupa, a drugi želi potvrdu za prethodno poslati RTS okvir u vidu CTS okvira. Pretpostavimo da oba istovremeno detektuju neaktivnost medijuma. Koji uređaj može da ostvari uspešan prenos? Objasnite.

Reference

[Ar02] Arbaugh, W., N. Shankar, Y.C. Justin Wan, and K. Zhang. "Your 802.11 Wireless Network Has No Clothes." *IEEE Wireless Communications,* vol. 9, no. 6 (December 2002),44-51.

[Be02] Berger, R. *802.11 Unleashed.* Englewood Cliffs, NJ: Prentice-Hall, 2002.

[Du03] Dubendorf, V. *Wireless DaIa Technologies reference Handbook.* New York: Wiley, 2003.

[Fo03] Forouzan, **B. Local Area Networks.** New York: McGraw-Hill, 2003.

[GrOl] Gravano, *S. Introduction to Error Control Codes.* Oxford and New York: Oxford University Press, 2001.

[HaOl] Halsall, F. *Multimedia Communications.* Reading, MA: Addison-Wesley, 2001.

[PrOl] Proakis, J. *Digital Communications.* New York: McGraw-Hill, 2001.

[Sc95] Schneier, B. *Applied Cryptography:* Protocols, Algorithms, and Secure Code in C. 2nd ed. New York: Wiley, 1995.

[Sc97] Schlegel, C. *Trellis Coding.* New York: Wiley, 1997.

[ScOO] Schiller, J. *MoUIe Communications.* Reading, MA: Addison-Wesley, 2000.

[St99] Stallings, W. *ISDN and Broadband ISDN with Frame Relay and ATM,* 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[Sw03] Swaminatha, T.M., and CR. Elden. *Wireless Security and Privacy: Best Practices and Design Techniques.* Reading, MA: Addison-Wesley, 2003.

POGLAVLJE

10

Povezivanje mreža

Informacije mogu da nam kažu sve. Daju odgovore na sva pitanja. AH, reč je o pitanjima koja možda još uvek nisu i nesumnjivo neće ni biti postavljena.

—**Jean Baudrillard,** francuski semiolog

10.1 Uvod

Kako se potreba za komunikacijama povećava, zajedno sa sve većim brojem uređaja, protokoli koje smo opisali u prethodnom poglavlju postaju neefikasni. Veće potrebe obično zahtevaju veću fleksibilnost od one koju te topologije i protokoli mogu da obezbede. Pošto se broj okvira povećava, LAN mreže postaju "zasićenije" i performanse se degradiraju. To je slično gradskim saobraćajnicama. U jednom trenutku može sasvim dobro da zadovolje saobraćajne potrebe, ali kako se povećava broj vozila, uvode se nove trake, izlazne i ulazne rampe, a ponekad i nove saobraćajnice kako bi se regulisao dodatni saobraćaj.

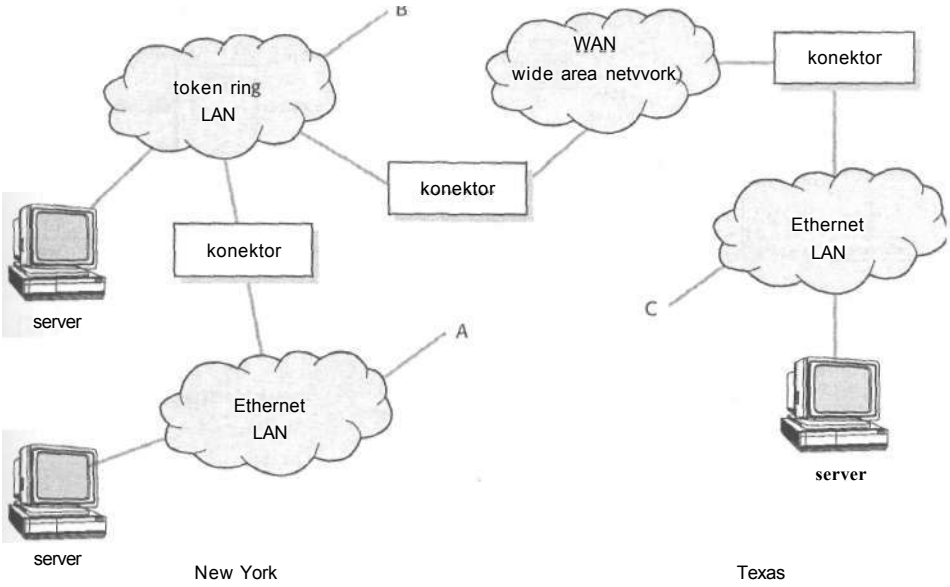
Jedan od načina da se izbegnu ozbiljna "uska grla" je da se LAN podeli na više manjih LAN-ova, čime se redukuje broj uredaja po LAN-u. Ovo rešenje doprinosi očuvanju performansi na prihvatljivom nivou. Međutim, distribuiranje svih uredaja na više LAN-ova nameće pitanje koji su uredaji pridruženi kojem LAN-u. Marfijev zakon kaže da će se, bez obzira kako grupišete uredaje, uvek desiti da dva uređaja iz različitih LAN-ova zahtevaju međusobnu razmenu informacija. Zato je neophodno obezbediti komunikacije između LAN-ova, a, pri tom, zadržati LAN protokol. To je neminovno vodilo ka dizajniranju načina na koje je moguće povezati mreže. Svaka dva uredaja na LAN-u mogu da komuniciraju pomoću LAN protokola. Međutim, ako se nalaze u različitim LAN-ovima, neophodno je razviti nove protokole koji će prevaziđ granice LAN-a.

Različite mreže su dizajnirane za različite ljude za sasvim različite ciljeve. Ipak, bilo bi nerealno pretpostaviti da su oni medusobno izolovani. Na primer, u većoj korporaciji različita odeljenja mogu da imaju specifične ciljeve i različite pristupe u obavljanju svojih poslova.

Postoje značajne razlike izmedu proizvodnog odeljenja, razvojnog i istraživačkog odeljenja i marketinga. Na njihove odluke o instaliranju kompjutera i povezivanju mreža utiču različiti faktori. Zato je moguće da usvoje sasvim različite i nekompatibilne sisteme. Medutim, i pored toga je neophodno obezbediti komunikaciju izmedu tih odeljenja; ako koriste nekompatibilne mreže, dolazi do problema. U takvim situacijama postoje dva moguća rešenja. Prvo je primoravanje na usvajanje jedinstvenog mrežnog standarda. Nažalost, ako izabrana mreža ne ispunjava postavljene ciljeve i potrebe odeljenja, onda je izbor kontraproduktivan. Drugo rešenje podrazumeva pronalaženje načina za povezivanje različitih mreža da mcgu da komuniciraju bez problema. Pošto vecina Ijudi očekuje da kompjuteri i mreže rade za njih, a ne da im se povinuju, drugi izbor je poželjan.

Na primer, razmotrite scenario sa slike 10.1, koji predstavlja mreže koje se koriste u velikoj korporaciji. LAN mreže su instalirane u dva odeljenja u Njujorku, a jedna u odeljenju u Teksasu. Uredaji A i B iz Njujorka pristupaju fajl serverima na svojim LAN-ovima, koristeći protokole o kojima je bilo reči u prethodnim poglavljima. Uredaj C iz Teksasa radi nešto slično. Medutim, sva tri uredaja povremeno zahtevaju pristup drugim LAN-ovima. Dve mreže u Njujorku mogu direktno da se povežu, ali, zbog velike udaijenosti, nije moguća direktna konekcija sa mrežom u Teksasu. Zato se za povezivanje Teksasa i Njujorka koristi WAN (mreža šireg geografskog područja).*

Mogućnost povezivanja mreža definitivno nije ništa novo. Postoje standardi i za softver i za povezivanje uredaja, a svi proizvodači kompjuterske opreme nude razne proizvode za tu svrhu. Pitanje je gde se u softverskoj i hardverskoj hijerarhiji ti uredaji uklapaju.



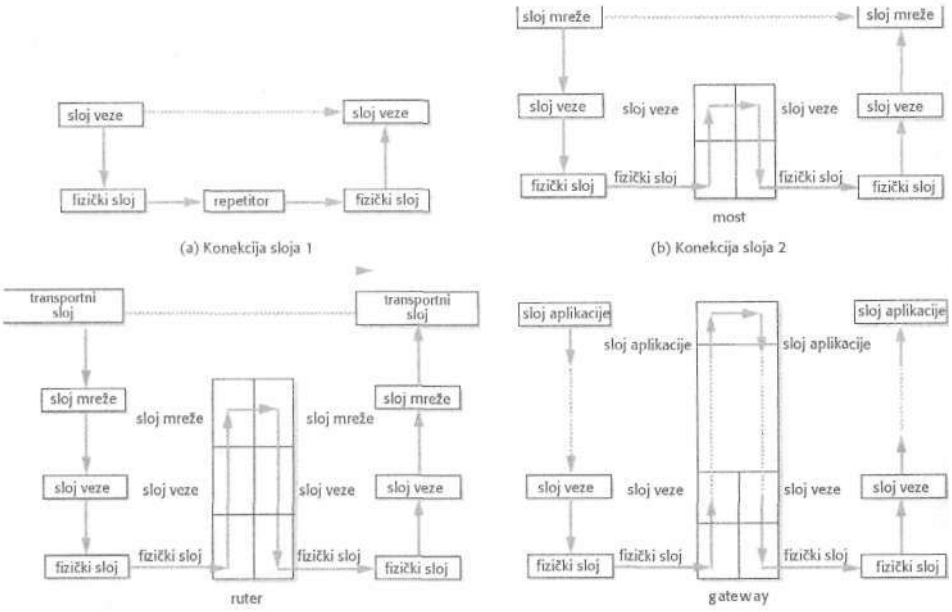
SLIKA 10.1 *Povezivanje mreža*

′ Kasnije ćemo predstaviti WAN mreže. Za sada je dovoljno da ih zamišljate kao mreže koje "pokrivaju" veću geografsku oblasi od LAN-a i koriste drugačije protokole.

Šta ti proizvodi, u stvari, rade? Modeli kao što je OSI obezbeduju nekoliko slojeva protokola za kompjuterske mreže. Na primer, pretpostavimo da se dva LAN-a u Njujorku razlikuju samo na MAC podsloju sloja veze, a da razlike u odnosu na WAN postoje na višim slojevima. Prilikom konekcija sa WAN-om treba rešiti probleme nekompatibilnosti, a to je mnogo složenije od povezivanja dva LAN-a.

Za uspostavljanje konekcija koriste se **konvertori protokola,** koji definišu logiku za prevodenje jednog protokola u drugi. U opštem slučaju, dve identične mreže mogu da se povežu na sloju 1, što uključuje samo elektronske konekcije koje omogućavaju regenerisanje i ponavljanje signala. Takođe je moguće povezati dve potpuno nekompatibilne mreže na najvišem sloju, koristeći uređaj poznat kao *gateuiay.* Takve konekcije zahtevaju kompletno poznavanje oba protokola i mogućnost prevođenja jednog u drugi. Konekcije su moguće i na slojevima izmedu najvišeg i najnižeg, ali nivo zavisi od stepena kompatibilnosti između dve mreže.

Najčešći konvertori protokola postoje na slojevima 1, 2 i 3 (slika 10.2). Ti uredaji i protokoli na kojima se zasnivaju predstavljaju najvažnije teme ovog poglavlja.



SLIKA 10.2 *OSIkonekcije*

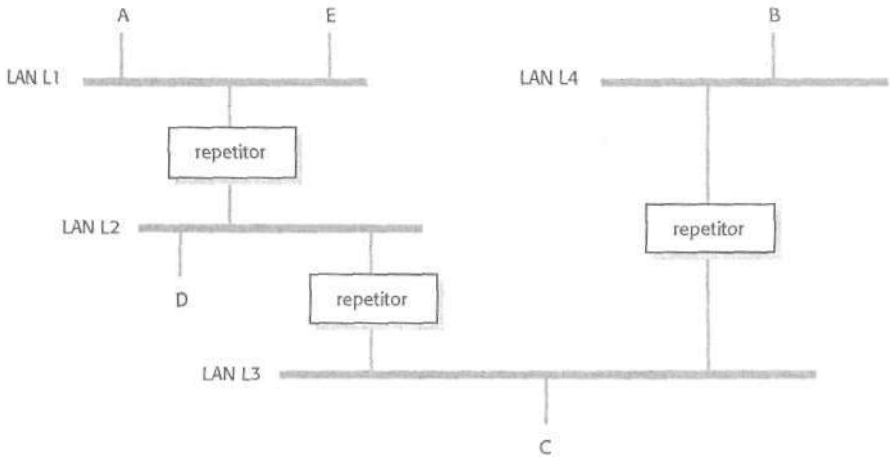
U odeljku 10.2 predstavićemo uređaje sloja 1, hubove i repetitore. U odeljku 10.3 biće reči o mostovima **i** komutatorima, uređajima koji izvršavaju translacije na sloju 2. U tom odeljku ćemo prikazati i konfigurisanje konekcija, rutiranje informacija između različitih LAN-ova i strategije za kreiranje tabela koje ukazuju na uređaje koji postoje na tom LAN-u. Počećemo sa originalnim uređajima sloja 2, mostovima, a zatim prelazimo na tekuće tehnologije, uključujući komutatore, komutirani Ethernet i virtuelne LAN-ove. U odeljku 10.4 ćemo predstaviti mreže koje se prostim preko širokih područja, kao što je Internet. U odeljcima 10.5 do 10.8 opisani su različiti algoritmi koji su neophodni ruterima za utvrdivanje putanja do zahtevanog odredišta i neki problemi koji su izazvani zagušenjem i "otkazima" uređaja, zajedno sa nadnima za oporavak od tih problema. U prethodnim poglavljima bavili smo se individualnim komponentama velikog komunikacionog sistema, a sada je vreme da sastavimo te delove.

10.2 Konekcije sloja 1

Repetitori i hubovi

Na slici 10.3 prikazan je uobičajeni način za povezivanje mreža pomoću **repetitora (repeaters),** uređaja koji funkcionišu na fizičkom sloju (sloj 1). Repetitor prihvata bitove okvira sa LAN-a na koji je povezan i prenosi ga do drugog LAN-a. Pretpostavlja da LAN-ovi na koje je povezan koriste iste protokole **i** iste formate okvira, a ne postavlja nikakve pretpostavke o značenju bitova. Primarna funkcija repetitora je regenerisanje signala, čime se povećava rastojanje koje je moguće "pokriti" IAN protokolima.

Uredaji A, B, C i D su povezani na različite LAN-ove (možda Ethernet segmente). Ipak, svaki uredaj vidi sve okvire koje drugi uredaji pošalju. Na primer, D vidi sve što A pošalje. Okvir će biti prihvaćen u zavisnosti od toea kome je namenien.

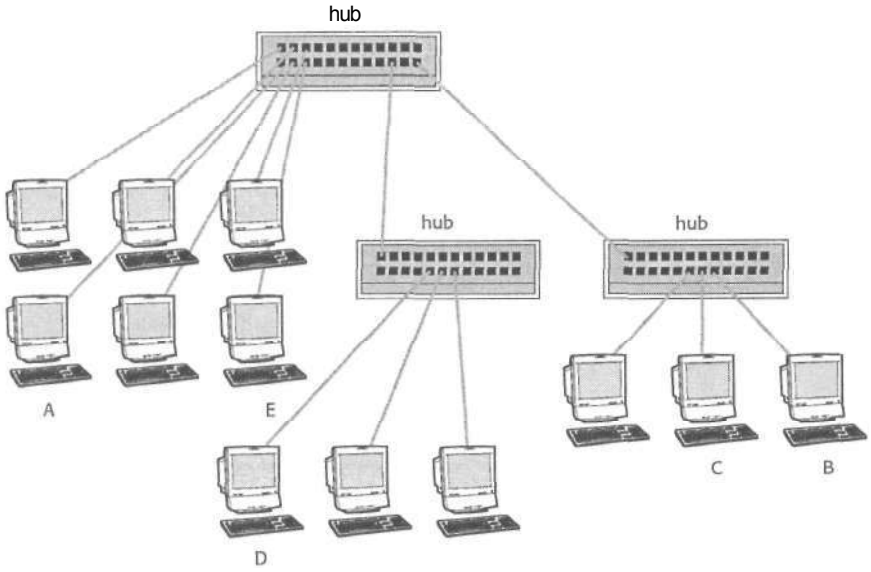


SLIKA 10.3 *LAN-ovi povezani pomoću repetitora*

Uređaji nisu "svesni" postojanja repetitora: što se njih tiče, oni su povezani na jedan, ali mnogo veći LAN. Ako su LAN-ovi Etherneti koji koriste CSMA/CD, svi uređaji se nalaze u istom **domenu kolizije.** Ovo znači da će, ako bilo koja dva od uređaja A, B, C, ili D istovremeno šalju okvir, doći do kolizije.

Kao što je opisano u prethodnom poglavlju, mnogi uređaji se oslanjaju na **hubove** prilikom uspostavljanja konekcija. Na slici 10.4 prikazano je tipično uređenje. Jedan hub može da poveže više uređaja, uključujući radne stanice i druge hubove. Sa stanovišta uredaja, postoji mala razlika izmedu povezanosti na hub, ili na segment. Ako uredaj za nadmetanje za pristup medijumu koristi CSMA/CD, u svakom slučaju funkcioniše na isti način: šalje okvire i "osluškuje" da li je došlo do kolizija. Funkcije huba su prilično slične funkcijama repetitora - on prihvata signal preko jednog porta (konekcija), regeneriše signal i šalje ga do svih ostalih portova. Hub se ponekad naziva **višeportni repetitor (multiport repeater).** Kao i kod repetitora, ako uredaji implementiraju CSMA/CD, svi se nalaze u istom domenu kolizije. Ponavljamo: ako bilo koja dva od uređaja A, B, C, ili D istovremeno pošalju okvir, dolazi do kolizije.

Repetitori i hubovi prvenstveno proširuju domet mreže, ali mogu da stvore i neke probleme. Jedan problem nastaje zato što više uredaja ima pristup medijumu. To povećava saobraćaj i može da degradira performanse LAN-a. Na primer, ako A šalje okvir do E (slika 10.3, ili 10.4), repetitori, ili hubovi prosleduju okvir do svih mogućih lokacija. Nemaju ugradenu logiku koja bi prepoznala da su A i E na istom LAN-u, ili da su povezani na isti hub i da ponavljanje okvira nema smisla. Naravno, rezultat je to da ni jedan uredaj ne može da šalje ništa dok A ne završi prenos. Ako uređaji izvršavaju CSMA/CD, proširivanje dometa LAN-a je ekvivalentno proširivanju domena kolizije.



SLIKA 10.4 *Uspostavljanje konekcija pomoću hubova*

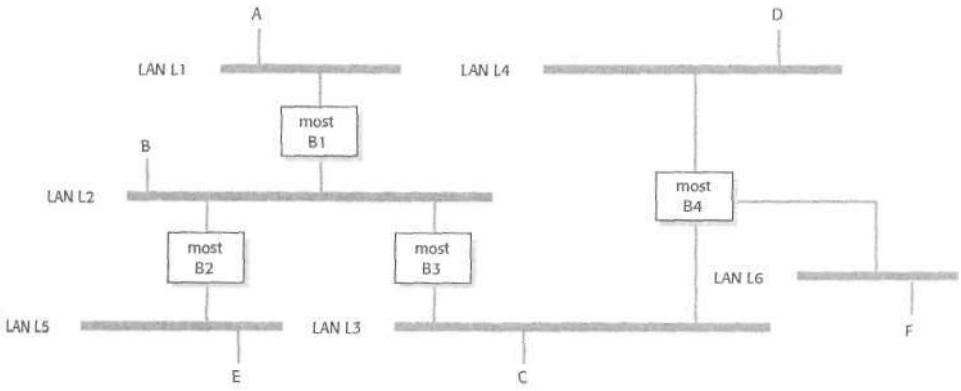
Drugi problem je zaštita. U opštem slučaju, što je više ljudi koji imaju pristup informacijama, zaštita se teže implementira. Ako A i E razmenjuju neke osetljive informacije, okviri prolaze kroz sve ostale uredaje, što povećava potencijalne bezbednosne propuste.

10.3 Konekcije sloja 2

Mostovi

Sledeći način za povezivanje LAN-ova je korišćenje mosta, ili komutatora. Razlika izmedu mosta i komutatora je analogna razlici između repetitora i huba. **Most** (bridge) tradicionalno povezuje dva LAN-a, a **komutator (switch)** povezuje više uredaja. Ovde je bitno to što oba konektora imaju mogućnost izvršavanja protokola sloja 2 i donošenja odluke kada treba proslediti okvire koje prime. U stvari, na slici 10.3 zamenite reč *repetitor* sa *most,* a na slici 10.4 *hub* sa *komutator* i dobićete istu mrežu sa istom fizičkom topologijom, ali sa mnogo većim mogućnostima za kontrolu, upravljanje i izolovanje saobraćaja. Mostovi su često korišćeni kada je bilo neophodno povezati Ethernet uredaje na zajednički segment. Sa razvojem Fast Ethemet protokola i prelaskom na UTP kao fizički medijum, komutatori postaju sve popularniji izbor. Najpre ćemo obraditi tradicionalne funkcije mosta. Tako možemo da predstavimo jednostavniji pogled na "stvari", bez kompromisa u vezi diskusije o nekim značajnim zadadma koje izvršavaju i mostovi i komutatori. Većina onoga što kažemo za mostove važi i za komutatore. Nakon toga, nastavljamo razmatranje o komutatorima i posebno komutiranom Ethernetu i virtuelnim LAN-ovima.

Most je konektor sloja 2. Kao takav, izvršava funkcije veze, kao što su detektovanje grešaka, formatiranje okvira i rutiranje okvira. Na primer, pretpostavimo da uredaj B šalje okvir na LAN L2 u mrežnoj konfiguraciji sa slike 10.5.



SLIKA 10.5 *LAN-ovi povezani mostom*

Most Bl proučava odredišnu adresu i, ako je okvir namenjen bilo kojem uređaju sa LAN L1 (recimo A), prihvata okvir i prosleduje ga korišćenjem protokola za nadmetanje za pristup na L1 (glavno pitanje kojim ćemo se uskoro pozabaviti je kako Bl zna da se nalazi na L1). Ako je okvir namenjen nekom uredaju sa drugog LAN-a, Bl ga ignoriše. Tako se Bl ponaša kao bilo koji uređaj koji selektivno odbacuje, ili prihvata okvire na osnovu njihovog odredišta.

Ako Bl prihvati okvir, izvršava rutine za detekciju grešaka da bi se utvrdilo da li je okvir ispravan. Ako nema grešaka, okvir se šalje preko LAN-a L1. Ako je format okvira na L1 isti kao format na L2, most šalje okvir onakav kakav jeste. Ako se formati razlikuju, most mora da reformatira primljeni okvir u format koji je konzistentan sa standardima koji važe na L1. U nekim slučajevima reformatiranje se svodi na reorganizovanje polja, dodavanje novih neophodnih polja i izbacivanje nepotrebnih. Nažalost, kao što ćemo uskoro videti, u nekim slučajevima reformatiranje stvara probleme - neke od njih protokoli sloja 2 ne mogu da reše.

Pre nego što predstavimo dizajn mosta, razmotrimo koji su razlozi za korišćenje mostova. Jedan razlog je povećanje efikasnosti. Videli smo da kod repetitora svaki okvir prolazi kroz svaki LAN, izazivajući bespotrebni saobraćaj. Da bi se to izbeglo, menadžer mreže može da kreira topologiju u sklopu koje se uredaji koji najčešće komuniciraju nalaze na istom LAN-u. Na primer, svaki LAN sa slike 10.5 može da odgovara različitim odeljenjima u velikoj kompaniji. Uredaji u sklopu odeljenja mogu međusobno da komunidraju, a mostovi prosleđuju samo okvire koji su namenjeni nekom drugom LAN-u. Na primer, dva uredaja mogu da komuniciraju preko L1 istovremeno, dok daiga dva uredaja komuniciraju preko L2. LI opštem slučaju, komunikacija izmedu različitih LAN-ova odvija se paralelno i tako je povećana efikasnost. Ako su LAN-ovi Etherneti, svaki most koristi CSMA/CD protokol prilikom prosleđivanja okvira. Značajan rezultat toga je činjenica da mostovi razdvajaju LAN-ove na različite domene kolizije i tako se redukuje broj kolizija koje bi se u suprotnom javile.

Mostovi po potrebi omogućavaju i komunikaciju između odeljenja. Na primer, ako A pošalje poruku do F, okvir će proći kroz sledeći niz mostova i LAN-ova: L1-B1-L2-B3-L3-B4-L6. Eventualno, okvir će stići do F.

Sledeći razlogza korišćenje mostova je bezbednost. Pošto selektivno prosleduju okvire, oni mogu da spreče prosledivanje odredenih okvira kroz mrežu. Ova procedura poboljšava zaštitu, jer neki uredaji ne mogu da vide šta drugi šalju. Na primer, LAN L1 može da povezuje uređaje koji razmenjuju osetljive informacije o zaposlenima, a na drugim LAN-ovima uredaji nemaju takve potrebe.

Premošćavanje različitih tipova LAN-a

Premošćavanje je još složenije kada je potrebno povezati različite tipove LAN-ova. Jedan od mogućih problema je što različiti LAN-ovi mogu da imaju različite bitske brzine. Na primer, pretpostavimo da most prihvata okvire sa bržeg LAN-a i da ih prosleduje na sporiji LAN, ili na LAN na kome je došlo do kolizije. Okviri mogu da stižu većom brzinom od one kojom se mogu prosledivati. Zato u baferu mosta mora da postoji dovoljno prostora da bi se omogućilo odlaganje svih okvira. Kašnjenja u mostu mogu da stvore probleme kao što su pauze u protokolima kontrole toka.

Tajmeri su postavljeni da bi bio definisan razuman period u kome okvir treba da stigne do svog odredišta i da se pošalje potvrda o uspešnom prijemu. Kašnjenja u mostovima mogu da uzrokuju preterane pauze (time-out) ako ih mrežni softver uredaja ne podešava. Ipak, tajmeri danas zavise od strategija međusobnog povezivanja i gubi se odredena mera transparentnosti topologije.

Formatiranje okvira je sledeđ problem. Sećate se iz Poglavlja 9 da svaki LAN standard ima drugačiji format okvira. Zato, ako most povezuje dva razlidta LAN-a, mora da reformatira sve primljene okvire pre nego što ih pošalje dalje. Površno gledano, reformatiranje ne deluje komplikovano, jer je prvenstveno reč o preuređivanju informacija. Pretpostavite da most povezuje Ethernet i token ring LAN. Okviri na token ringu imaju prioritete; na Ethernetu nemaju. Zato okvir koji odlazi sa token ringa na Ethemet gubi svoj prioritet. Okviru koji stiže sa Etherneta na token ring mora da se dodeli prioritet, ali koji? Obično se dodeljuje podrazumevani prioritet. Ali, šta se dešava ako okvir ide sa token ringa na Ethernet, pa na drugi token ring? Okvir je inicijalno imao prioritet, izgubio ga je kada je prešao na Ethernet, a dobija drugi prioritet kada stigne na drugi token ring. Medutim, nema nikakvih garancija da će inicijalni i konačni prioritet biti isti.

Rutiranje

Povezivanje dva razlidta tipa LAN-a nameće nova pitanja. Međutim, u ostatku odeljka pretpostavićemo da most povezuje dva slična LAN-a. Ovo je mogući scenario, zato što je Ethernet postao dominantan LAN standard. Osim toga, na ovaj način možemo da se fokusiramo na sledeću funkciju mosta - na prosledivanje okvira. Samo po sebi, prosledivanje nije teško. Problem je kako most može da zna kada treba da prihvati i prosledi okvir. Na primer, pretpostavimo da most B3 sa slike 10.5 detektuje okvir na LAN-u L2. Ako je okvir namenjen uredaju A, ili E, B3 ga ignoriše. Medutim, pretpostavimo da je odredište okvira D, C, ili F. Uređaj C je na LAN-u L3 i jedini nadn da se dode do uredaja D, ili F je preko L3. Zato most mora da prihvati okvir i da ga prosledi na LAN L3. Ali, kako B3 može da zna da su D, C, ili F dostupni preko L3? Što je još gore, šta ako neko pomeri F sa L6 na L5? Kako rnost zna da taj uredaj više nije dostupan preko L3?

Ova pitanja možda na prvi pogled deluju trivijalno, jer Vi, kao čitalac, celu mrežu možete da sagledate preko jednog dijagrama. Globalno videnje situacije uvek olakšava problem. Ali, mostovi ne mogu tako da sagledavaju "stvari". Oni su povezani na dva, ili više LAN-ova i vide samo ono što stiže do njih. Osim toga, most sigurno ne može da vidi šta se dešava na LAN-u koji je udaljen na nekoliko mostova od njega. Proces donošenja odluke koji će okviri biti prosleđeni i gde naziva se **rutiranje pomoću mosta.**

Tabele rutiranja

Mostovi donose odluke o rutiranju na osnovu informacija koje se čuvaju u **tabeli rutiranja,** koja se ponekad naziva baza prosledivanja (forvvarding database), ili direktorijum za rutiranje. Gde će most rutirati okvir zavisi od LAN-a na koji okviri stižu. Svaki most ima po jednu tabelu rutiranja za svaki LAN na koji je povezan. Kada okvir stigne sa tog LAN-a, most pronalazi odredišnu adresu u odgovarajućoj tabeli rutiranja. Ulazi tabele defmišu na koji LAN most treba da usmeri pristigli okvir.

Da bismo ovo ilustrovali, na slici 10.6 su predstavljene tabele rutiranja za mostove sa slike 10.5. Most Bl ima dve tabele rutiranja (slika 10.6a), po jednu za LAN-ove L1 i L2. Kada most detektuje okvir na L1, on utvrđuje odredišnu adresu i traži je u tabeli rutiranja za L1. Ako odredišna adresa ukazuje na neki od uredaja B do F, most prosleđuje okvir do L2, jer je to jedini način da se dođe do tih uređaja. Ako je odredište A, most ne prosleduje okvir i on ostaje na L1. Slično tome, ako most detektuje okvir sa L2 namenjen za A, on ga prosleđuje do L1.

Tabele za most B3 su slične. Most će proslediti sve okvire sa L2 na L3 ukoliko su namenjeni uredajima C, D, ili F. Međutim, ako okvir stiže na L2 i namenjen je za A, B, ili E, most ga neće proslediti. Trebalo bi da pogledate i ostale tabele da biste se uverili da ulazi tačno odražavaju topologiju sa slike 10.5.

Izvor LAN L1		Izvor LAN L2	
Odredište	Sledeći LAN	Odredište	Sledeći LAN
A	—	A	L1
B	L2	B	—
C	L2	C	—
D	L2	D	—
E	L2	E	—
F	L2	F	—

(a)MostBI

Izvor LAN L2		Izvor LAN L5	
Odredište	Sledeći LAN	Odredište	Sledeći LAN
A	—	A	L2
B	—	B	L2
C	—	C	L2
D	—	D	L2
E	L5	E	—
F	—	F	L2

(b) Most B2

Izvor LAN L2		Izvor LAN L3	
Odredište	Sledeći LAN	Odredište	Sledeći LAN
A	—	A	L2
B	—	B	L2
C	L3	C	—
D	L3	D	—
E	—	E	L2
F	L3	F	—

(c) Most B3

Izvor LAN L3		Izvor LAN L4		Izvor LAN L6	
Odredište	Sledeći LAN	Odredište	Sledeći LAN	Odredište	Sledeći LAN
A	—	A	L3	A	L3
B	—	B	L3	B	L3
C	—	C	L3	C	L3
D	L4	D	—	D	L4
E	—	E	L3	E	L3
F	L6	F	L6	F	—

(d) Most B4

SLIKA 10.6 *Tabele rutiranja za tnostove sa slike 10.5*

Kako most definiše svoju tabelu rutiranja? Jedan način podrazumeva programiranje svakog mosta sa adresom svakog uredaja i LAN-om na koji okviri mogu da budu prosledeni od tog uredaja. Ovaj pristup nazivamo **fiksno rutiranje,** jer pretpostavljamo da se informacije iz tabele ne menjaju. Ipak, u većini mrežnih okruženja ova pretpostavka je isuviše ograničavajuća. Moguće je da se na mrežu priključuju novi uredaji, stari se uklanjaju, a neki uredaji se premeštaju na nove lokacije. Potrebno nam je tačno rutiranje koje neće zavisiti od promene lokacije uredaja i mrežne topologije. Tako bi bila obezbedena transparentnost koja bi olakša'la korišćenje mreže.

Ako želimo da koristimo tabele rutiranja u dinamičkim okruženjima, imamo dva moguća izbora. Jedan je da se vrši reprogramiranje mostova svaki put kada neko doda, ukloni, ili premesti uredaj. U dinamičkim okruženjima ovaj pristup nije funkcionalan, tako da moramo da se odlučimo za drugu opciju - za utvrdivanje načina za automatsko ažuriranje mostova.

Transparentni mostovi

Mostove koji mogu da kreiraju i ažuriraju svoje tabele mtiranja nazivamo **transparentni mostovi.** Oni imaju sopstveni standard (IEEE 802.Id). Dizajnirani su tako da ih možete priključiti i odmah će funkcionisati, bez obzira na topologiju i lokacije uredaja. Nema potrebe da im ukazujete gde se uredaji nalaze - utvrdiće to automatski i tako inicijalizovati svoje tabele rutiranja, bez potrebe za specijalnim programiranjem. Ako se uredaj premesti iz jednog LAN-a u drugi, svaki most to shvata i ažurira svoju tabelu ažuriranja na odgovarajući način. Ova mogućnost ažuriranja tabela rutiranja naziva se **učenje rute,** ili **učenje adrese.**

Učenje rute Most uči šta treba da postavi u tabelu rutiranja posmatranjem saobraćaja. Uvek kada primi okvir, on proučava njegovu izvornu adresu. Tako zna da je uredaj koji je poslao okvir dostupan na LAN-u preko koga je okvir stigao. Most proučava sve svoje tabele rutiranja, tražeći adresu uređaja. Ako zapis iz tabele ukazuje da je uredaj dostupan preko drugog LAN-a, most menja zapis tako da naznači LAN preko koga je okvir stigao. Moguće je da se uredaji premeštaju na drugi LAN.

Ilustracije radi, uzmimo za primer tabele rutiranja sa slike 10.6 i LAN-ove sa slike 10.5. Pretpostavimo da se uredaj D premešta iz LAN-a L4 na LAN L1. Tabele rutiranja sada nisu tačne i jedini uredaji koji mogu da pošalju okvir do D su oni koji se nalaze na istom LAN-u kao i D. Zatim, pretpostavite da D šalje okvir do E na LAN-u L5. Most Bl rutira okvir na L2. Međutim, proučava i svoje tabele rutiranja (slika 10.6a). Pošto je most primio okvir od D preko L1, on zna da se D nalazi negde u smeru L1.* Zbog toga, menja četvrti zapis u svim tabelama rutiranja i redefiniše ih kao što je prikazano u tabeli 10.1.

″ Napomenimo da Bl ne mora da zna da je D na L1. Sve što zna je da Je okvir m<5gao prode kroz nekoliko mostova dok nije došao do L1. Ovde je važno što most zna u kom smeru treba da usmeri okvir.

Tabela 10.1: Ažurirana tabela rutiranja za most B1 sa slike 10.5

Izvor LAN L1		Izvor LAN L2	
Odredište	Sledeći LAN	Odredište	Sledeći LAN
A		A	L1
B	L2	B	
C	L2	C	
D		D	L1
E	L2	E	
F	L2	F	

Sada most Bl zna da ne treba da prosleđuje okvire sa L1 koji su namenjeni za D i da treba da prosledi sve okvire iz L2 koji su namenjeni za D do L1. Kada most B2 detektuje okvir uređaja D sa L2, on ažurira svoje tabele na sličan način. U ovom slučaju B2 "shvata" da je uredaj D dostupan preko L2. Međutim, iz perspektive mosta B2, ovo se ne razlikuje od mogućnosti da se D nalazi na L4. Zato su "ažurirane" vrednosti iste kao i originalne.

Ipak, ostaje nedoumica. Mostovi Bl i B2 su "naučili" da se D pomerio samo zato što je poslao neki okvir. Mostovi B3 i B4 još uvek ne znaju da je D pomeren. Da li treba da "žive u neznanju" sve dok D ne pošalje nešto do njih? Sta ako D to nikada ne uradi? Da li to znači da nikada neće "saznati" šta se desilo? Ako je tako, okviri poslati ka D sa LAN-ova L3, L4, ili L6 nikada neće stići do D. Šta ako D nije poslao okvir ni do E? Onda ni Bl i B2 ne bi znali da se D pomerio i ništa ne bi moglo da stigne do D.

Do sada smo razmatrali samo promenu informacija. Sledeća pitanja su kako se tabele inicijalizuju i šta mostovi rade na samom početku. Srećom, standard 802.Id daje odgovore na ta pitanja. Svaki most ima tajmer. Svaki put kada tajmer istekne, most čisti sadržaj tabela rutiranja. Most "smatra" da se nakon nekog vremena lokacije uređaja možda mogu promeniti; zato se uklanjaju sve informacije o rutiranju.

Na osnovu ovoga bi se moglo zaključiti da uređaji postaju nedostupni, jer se uklanjaju sve poznate informacije za rutiranje. Medutim, kada most primi okvir za uredaj za koji nema zapis u tabeli rutiranja, koristi **algoritam plavljenja (flooding algorithm).** Naime, šalje okvir preko svih LAN-ova na koje je povezan, osim onog sa koga je okvir stigao. Ovo ima dve namene: okvir zagarantovano stiže do svog odredišta (uz pretpostavku da postoji) i omogućava ostalim mostovima da vide okvir i tako "zaključe" u kojem smeru se nalazi uredaj pošiljalac. Na taj način su obezbedene najnovije informacije u tabeli rutiranja.

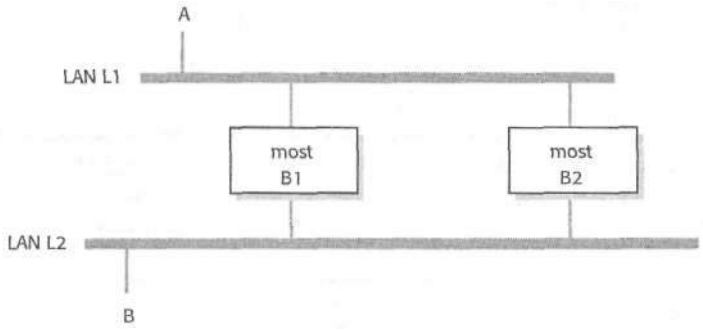
Razmotrimo prethodni primer u kojem D prelazi sa L4 na L1 na slici 10.5, pa šalje okvir do E. Ako E nije ništa poslao već duže vremena, njegovi zapisi u tabeli rutiranja su izbrisani. Kada Bl primi okvir iz D, on registruje da ne postoji zapis za E i automatski prosleduje okvir do L2. Slično tome, B2 i B3 prosleđuju okvir do L5 i L3, respektivno. Konačno, B4 prosleduje okvir do L4 i L6. Krajnji rezultat je da E dobija okvir i da je svaki most prosledio okvir iz D i tako ažurirao svoje tabele rutiranja. Tako će sve što se pošalje iz D biti ispravno prosleđeno (bar dok tajmer sledeći put ne istekne).

Algoritam plavljenja omogućava mostovima i da inicijalizuju svoje tabele rutiranja. Pretpostavimo da je IAN instaliran i da su sve tabele rutiranja u mostovima prazne. Ni jedan most ne zna lokadje uređaja na mreži. Kada most primi svoj prvi okvir sa LAN-a, šalje ga ka svim ostalim LAN-ovima na koje je povezan. Slično tome, kada prime neki okvir, mostovi sa tih LAN-ova takode ga prosleđuju, poštujući isti algoritam. Neće proteći mnogo vremena dok okvir ne stigne do svih mostova i svih LAN-ova. Sigurno će stići do svog odredišta i svaki most će znati pravac ka uredaju koji ga je poslao.

Što veći broj uredaja šalje okvire, mostovi ih prosleđuju koristeći zapise iz tabele rutiranja, ili algoritam plavljenja. Dok okvir napreduje kroz mrežu, mostovi koji nisu imali zapise eventualno "uče" smer iz koga okvir dolazi i mogu da proslede okvire bez algoritma plavljenja.

Propagiranje okvira Prethodni pristup u dizajniranju transparentnih mostova i učenja ruta funkcioniše dobro za konkretni primer. Medutim, određene topologije mogu da izazovu beskonačno propagiranje okvira i da tako prezasite mrežu. Ilustracije radi, pretpostavimo da su dizajneri mreže odlučili da dodaju drugi most koji će povezivati dva LAN-a. Dodavanje redundantnih mostova izmedu dva LAN-a ponekad može da bude korisno u cilju zaštite sistema od eventualnih "otkaza". Ako prvi most "otkaže", drugi je već spreman da "uskoči" i nema nikakvog (ili postoji veoma malo) kašnjenja izazvanog "otkazom". Ovo je izuzetno korisno u real-time sistemima, kod kojih bi "otkazi" opreme imali katastrofalne posledice.

Na primer, jednostavna LAN konekcija sa slike 10.7 prikazuje dva mosta između dva ista LAN-a. Pretpostavimo da su njihove tabele rutiranja prazne i da A šalje okvir do B. Pošto ni jedan most nije "svestan" onog drugog, svaki prihvata okvir i prosleđuje ga na LAN L2. Zatim, Bl vidi okvir iz B2, a B2 vidi okvir iz Bl na L2. Pošto ni jedan ne zna gde se B nalazi, oba mosta prihvataju okvir i prosleđuju ga na LAN L1. Oba mosta vide okvir od onog drugog i ovoga puta ga prosleduju nazad na L2.

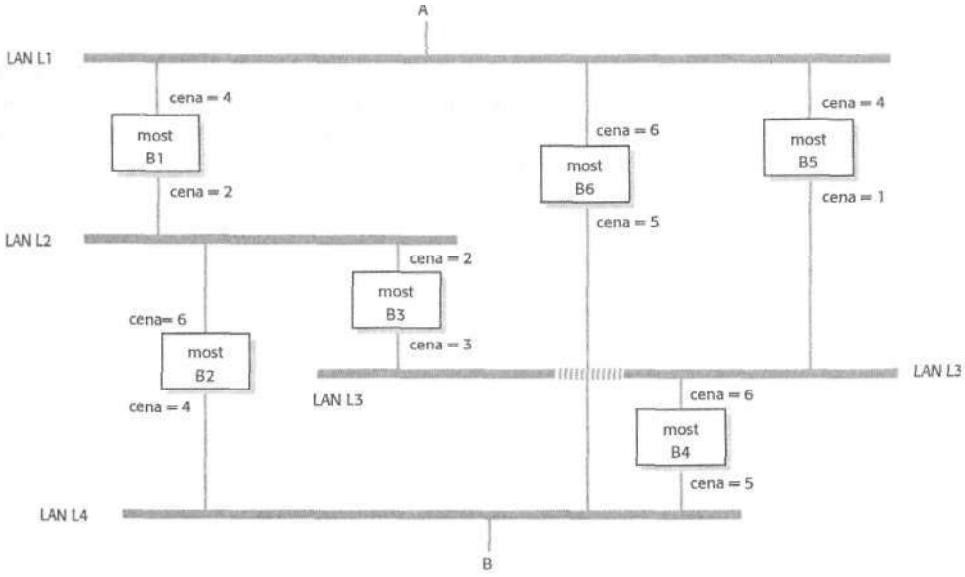


SLIKA 10.7 *Dva mosta povezuju dva LAN-a*

Sve dok B ne potvrdi svoju lokaciju, okviri će se prenositi n'pred-nazad.

Situacija postaje još gora ako se uvede i treći most (B3) koji povezuje ove LAN-ove. Ako A šalje jedan okvir duž L1, svaki most ga prosleduje do L2, šaljući tri okvira na L2. Svaki most vidi druga dva okvira (po jedan iz druga dva mosta) i prosleđuje ih na L1, što stvara šest okvira na L1. Sada mostovi vide po četiri okvira (po dva od druga dva mosta) i prosleduju ih, tako da se nazad na L2 šalje 12 okvira. Proces se nastavlja, izazivajući "eksploziju" okvira, koja eventualno može da zaguši sistem i onemogući normalne komunikacije.

Ilustrovali smo problem pomoću jednostavne topologije, prikazujući samo dva mosta izmedu dva LAN-a. Okviri koji potendjalno beskonačno cirkulišu postoje i kod drugih topologija. Ovo se posebno odnosi na situacije u kojima postoje dve zasebne rute izmedu uredaja koje stvaraju petlje u topologiji. To znači da okvir može da napusti LAN preko jedne rute samo da bi se vratio preko druge. Na slici 10.8 prikazana je topologija sa više različitih ruta izmedu A i B (uskoro ćemo videti šta cene označavaju). U stvari, postoji najmanje sedam ruta, ako ignorišemo prolazak kroz isti most više puta (možete li da pronađete rute?). Da biste videli razmere ovog problema, pretpostavimo da A šalje okvir do B i da su mostovi koristili algoritam plavljenja. Pretpostavimo da okvir prati rutu od L1 do L4 preko Bl i B2. Most B6 prosleđuje okvir nazad do L1. Most B4 će proslediti okvir do L3. Odatle, B5 prosleđuje okvir do L1, a B3 ga prosleđuje do L2.



SLIKA 10.8 *Više LAN-ova sa petljama*

Zatim, Bl prosleđuje okvir do L1, B2 ga prosleđuje nazad do L4 i proces se nastavlja. Kao i ranije, jedan okvir izaziva beskonačno povećanje broja okvira koji se kreću kroz mrežu. Ako uzmemo u obzir i druge rute od L1 do L4, povećanje je još brže. Broj okvira veoma brzo postaje ogroman i mreža pada.

Spanning tree algoritam

Jedan mogući način rešavanja problema propagiranja okvira podrazumeva eliminisanje petlji, tako što se neće koristiti odredeni mostovi. Mostovi neće biti fizički isključeni, ali će se sprečiti prosleđivanje okvira iz njih; koriste se kao rezervni uredaji u slučaju da drugi most "otkaže". Nezgodno je utvrditi koji se mostovi koriste i kada se moraju automatski rekonfigurisati ako dode do "otkaza" rnosta. Kao i obično, želimo da mostovi samostalno rade i da konfiguracija bude transparentna za korisnika. Jedno rešenje podrazumeva izvršavanje **spanning tree algoritma;** *spanning tree (stablo razapinjanja),* termin koji Vam je verovatno poznat sa kurseva o strukturama podataka, uključuje minimalni podskup grana uzetih sa povezanog grafa koji povezuje temena grafa. Podskup je minimalan po tome što ovo stablo nema petiji. * Za više informacija **o** ovoj vrsti stabla pogledajte reference za strukture podataka, kao što su [Dr95], ili [Pa95].

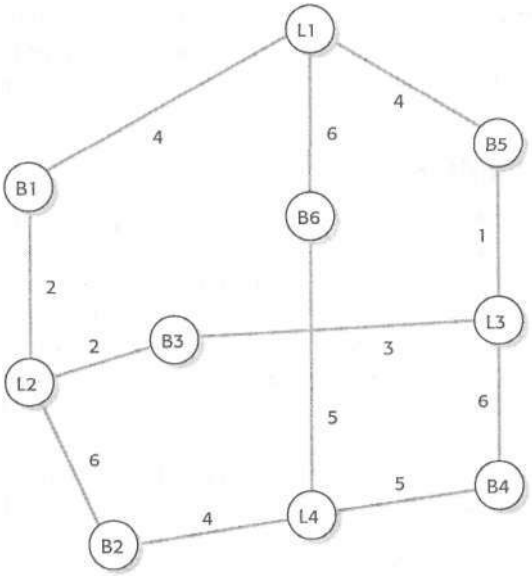
Da bi algoritam funkcionisao, najpre dodeljujemo cenu konekciji izmedu mosta i LAN-a, tj. **portu mosta (bridge port).** Cene mogu da budu definisane na osnovu bitske brzine kojom port mosta može da prenosi okvire na LAN. Tipično, niže bitske brzine znače višu cenu. Cena slanja okvira sa jednog LAN-a na drugi predstavlja sumu cena svih portova na konkretnoj ruti. U nekim slučajevima su sve cene postavljene na 1, tako da je cena rute odredena brojem uključenih mostova. Na slici 10.8 prikazana je cena svakog porta mosta. Cena slanja okvira od L1 **do** L4 preko mostova Bl i B2 iznosi 6; dobijena je kao suma cena prolaska od Bl do L2 (2), plus B2 do L4 (4). Napomenimo da ovde nisu uključene cene portova Bl-do-L1 i B2-do-L2 - oni se koriste za prosledivanje okvira u drugom smeru.

LAN topologiju smo vizuelno predstavili grafom. LAN-ovi i mostovi su temena, a konekcije izmedu LAN-a i mosta su grane. Naslici 10.9 data je grafička reprezentacija topologije sa slike 10.8. Zapamtite da se, dok okvir putuje preko grafa, sabiraju samo cene od čvora mosta do čvora LAN-a.

Spanning tree algoritam utvrduje niz grana koje povezuju sve LAN čvorove sa slike 10.9. Zapamtite da smo prilikom razmatranja algoritma imali priliku da vidimo topologiju mreže; rnostovi koji izvršavaju algoritam nemaju mogućnost ovakvog sagledavanja mreže. Oni znaju sarao za LAN-ove na koje su povezani. To algoritam čini još kompleksnijim nego da ga izvršava procesor koji ima uvid u kompletnu topologiju.

Da bi započeli spanning tree algoritam, mostovi mogu da izaberu jednog od njih da bude **root (koren stabla).** To je, obično, most sa najnižim ID-om, iako je moguće koristiti i prioritete. Koristeći terminologiju struktura podataka, root će biti početak stabla razapinjanja.

* Kod struktura podataka termin *cihlus* se, obično, koristi umesto termina *petlja.*



SLIKA T 0.9 *Grafiđia reprezentacija topologije LAN-a sa slike 10.8*

Mostovi biraju root slanjem niza specijalnih okvira poznatih kao **BPDU (bridge protocol data units)** u pravilnim intervalima. Svaki BPDU sadrži ID mosta, ID porta preko koga je okvir inicijalno poslat i akumulirane cene portova preko kojih je okvir prešao. Akumulirana cena predstavlja cenu putanje od tekuće lokacije BPDU-a do izvora okvira.

Kada most primi BPDU, on poredi ID izvomog mosta sa sopstvenim. Ako most ima viši ID, on zna da neće biti root. Beleži ID mosta pošiljaoca i cenu putanje do njega, inkrementira putanju za cenu prijemnog porta i prosleđuje BPDU kroz sve svoje preostale portove. Osim toga, prestaje da šalje sopstvene BPDU-e. Ako je ID mosta niži od onog koji je poslao BPDU, on neće proslediti okvir. "Smatra" da izvorni most nikada neće biti izabran, tako da nema potrebe da prosleduje njegove okvire.

Na kraju, svaki most, osim onog sa najnižim ID-om, prestaje da šalje okvire, zato što zna da neće biti izabran kao root. Preostali most zaustavlja prosleđivanje svih primljenih okvira i eventualno prestaje da prima nove. Posle određenog vremena kada okviri prestanu da stižu, on smatra da je propisno izabran kao root. Nakon toga, zajedno sa svim ostalim mostovima prelazi na izvršenje sledećeg koraka u sklopu algoritma.

U drugom koraku svaki most utvrduje svoj **root port,** port koji odgovara najjeftinijoj putanji ka root mostu. Pošto svaki most prethodno beleži cene za svaki BPDU primljen preko svakog porta, jednostavno traži najnižu cenu. Tako je omogućena komunikacija sa root mostom preko root porta.

U poslednjem koraku se utvrduje **označeni most (designated bridge)** za svaki IAN. To je most koji eventualno prosleduje okvire sa tog LAN-a. Mostovi biraju označeni most slanjem BPDUa preko svakog LAN-a na koji su povezani.

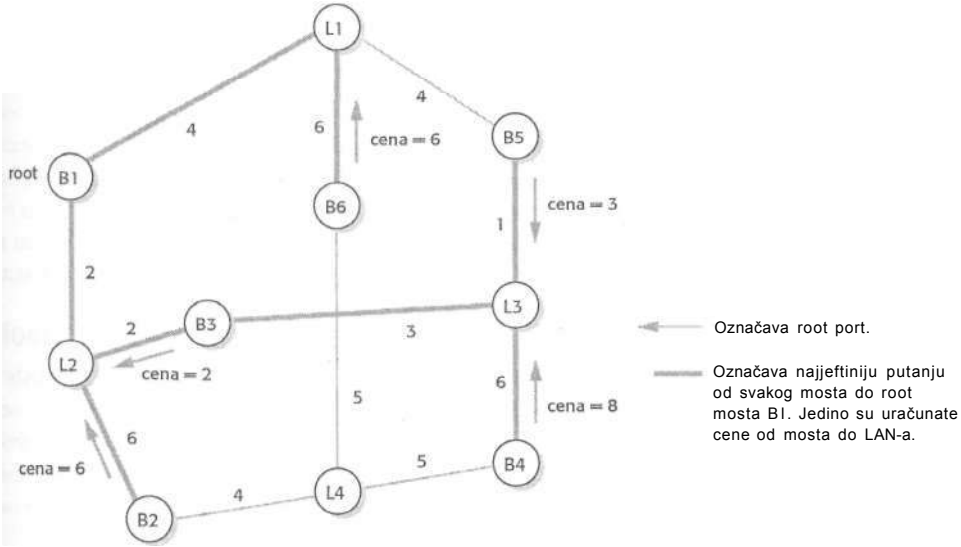
Most neće poslati BPDU na LAN preko ranije utvrdenog root porta. U suštini, root port odreduje LAN u smem root mosta. Algoritam sada mora da utvrdi da li postoje neki LAN-ovi u drugim smerovima.

Proučimo aktivnosti iz perspektive određenog LAN-a. LAN prenosi BPDLJ-e od svojih mostova koji zahtevaju da budu izabrani za označeni most. Svaki BPDU sadrži cenu od mosta koji ga je poslao do root mosta. Kada most primi BPDU, on poredi njegovu cenu sa svojom cenom do root mosta. Ako je njegova cena veća, zna da neće biti izabran i odustaje od svojih zahteva. Na kraju, samo jedan most ostaje sa najnižom cenom i on postaje označeni most tog LAN-a. U slučaju da postoje dva mosta sa najnižom cenom, konačni izbor se donosi na osnovu ID-a. "Pobeduje" niži ID.

Nakon biranja označenih mostova za svaki LAN, spanning tree algoritam je kompletan. Svaki LAN je povezan na svoj označeni most, a svaki most može da komunicira sa root mostom preko svog root porta. Tako je defmisana jedinstvena putanja između bilo koja dva LAN-a i izbegnuto je propagiranje okvira koje nastaje kod algoritama plavljenja.

Sada ćemo ilustrovati spanning tree algoritam pomoću jednog primera. Uzmimo LAN topologiju sa slike 10.8 i pridaiženi graf sa slike 10.9. U prvom koraku se utvrđuje root most. Ako pretpostavimo da su mostovi numerisani rastućim redosledom od Bl do B6, onda je Bl izabran kao root most.

U toku procesa biranja svaki most beleži cenu putanje do root mosta preko svih svojih portova i bira najjeftiniju. Na slici 10.10 prikazani su root portovi (označeni strelicom) i putanje do root mosta (označene obojenim linijama).



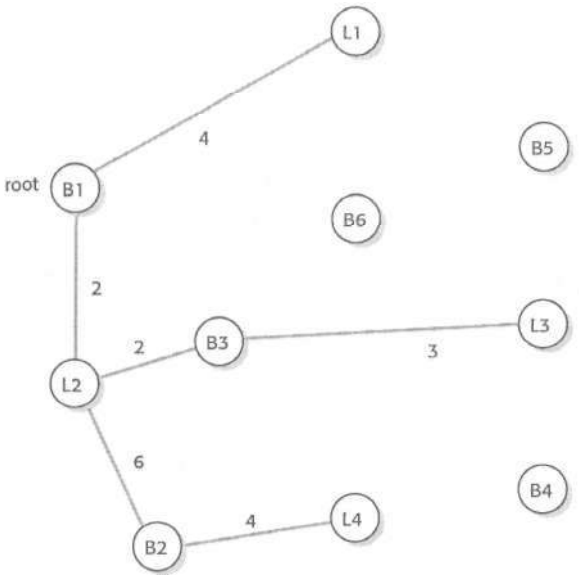
SLIKA 10.10 *Graf nakon utvrđivanja root portova*

Označava najjeftiniju putanju od svakog mosta do root mosta Bl. Jedino su uračunate cene od mosta do LAN-a. (Denotes cheapest path from each bridge to the root bridge Bl. OnIy costs from a bridge to a LAN are accrued.)

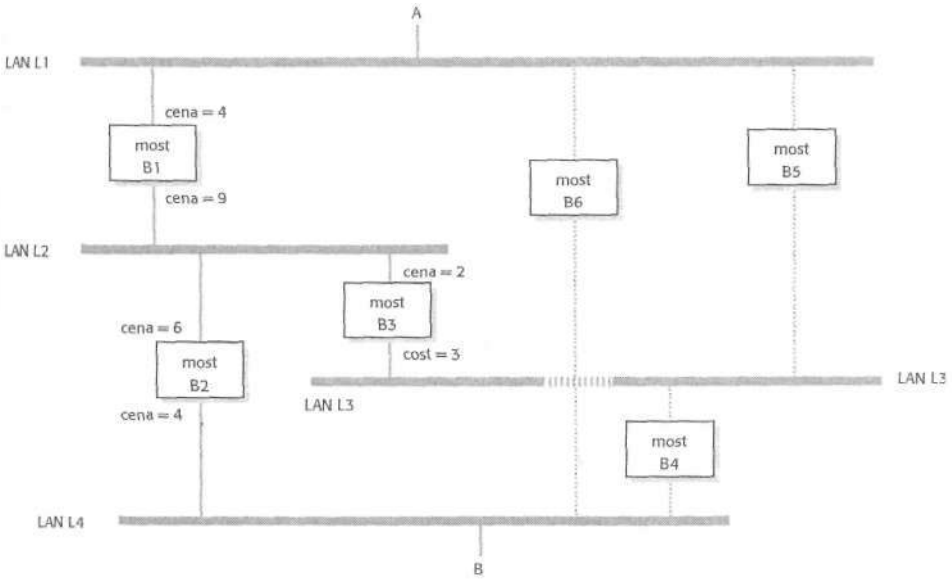
Pored strelica, navedene su i cene putanja. Na primer, root port mosta B2 je povezan na L2. Najjeftinija putanja je B2-L2-B1, sa cenom 6. Zapamtite da su uračunate samo cene od mostova ka LAN-ovima. Root port mosta B4 je povezan na L3. Putanja ka root mostu je B4-L3-B3-L2-B1, sa cenom 8. Kao što slika pokazuje, postoje i druge putanje, ali njihove cene su više.

U poslednjem koraku utvrđujemo izabrani most za svaki LAN. Root portovi povezuju mostove na neke LAN-ove, ali mogu da postoje i drugi LAN-ovi koji nisu deo razvojne šeme povezivanja. Na primer, na slici 10.10 nema root porta povezanog sa L4 i moramo da utvrdimo koji će most prosledivati informacije sa L4. Da bi se to izvelo, mostovi B2, B4 i B6 šalje BPDU-e duž L4, zahtevajući da dobiju ulogu označenog mosta. B4 ističe da je njegova cena do root mosta 8; B2 i B6 ističu da je njihova cena 6. Zato B4 odustaje, jer ima višu cenu, a B6 odustaje jer ima viši ID. Tako B2 postaje označeni most za L4. Nastavljajući na sličan način, B3 postaje označeni most za L3, a Bl je označeni mostza L1 i L2 (slika 10.11).

Na slici 10.11 prikazano je rezultujuće stablo, a na slici 10.12 originalna mrežna topologija. Isprekidane linije označavaju fizičke, ali ne i aktivne konekcije. Mostovi koriste ove konekcije za slanje i prijem BPDU-a, ali ne i za opšte prosleđivanje okvira. Tri povezuju sve LAN-ove, iako se neki mostovi ne koriste. To se moglo i očekivati, jer su mostovi redundantni uredaji koji se postavljaju za slučajeve "otkaza". Sve dok ne dode do "otkaza", LAN-ovi komuniciraju koristeći topologiju prikazanu na slici 10.12.



SLIKA 10.11 *Grafnakon utvrđivanja označenih mostova*



SLIKA 10.12 *Mrežna topologija sa konekcijama ka aktivnim mostovima*

Da bi bio detektovan "otkaz" nekog mosta i da bi bila izvedena rekonfiguracija šeme povezivanja, svaki most održava tajmer označen kao **tajmer starosti poruke (message age timer).** U naznačenom periodu svaki most (čak i oni koji nisu deo stabla razapinjanja) očekuje da "čuje" od root mosta potvrdu njegovog statusa kao root mosta. Kada primi ovu potvrdu, resetuje svoj tajmer. Naravno, root most saraduje periodičnim slanjem BPDU-a, radi potvrde svog statusa. Ako postoji kvar kod raosta, jedan, ili više mostova neće primiti konfiguracioni BPDU i njihovi tajmeri će isteđ. Ako "otkaže" neki drugi most koji nema ulogu roota, mostovi na koje to utiče razmenjuju BPDU-e da bi bio izabran novi označeni most za njihove LAN-ove. Ako dode do "otkaza" root mosta, oni moraju da izaberu novi root most. U svakom slučaju, rekonfiguracija aktivne topologije mora da se izvrši dinamički.

Mostovi koji koriste izvorno rutiranje

Poslednji pristup rutiranju okvira koji ukratko pominjemo prebacuje teret rutiranja na individualne uređaje, umesto na mostove. Logika u pošiljaocu utvrđuje rutu ka odredištu i smešta je u okvir. Ruta se sastoji od niza **oznaka ruta,** koje se sastoje od LAN-a i ID-a odgovarajućeg mosta. Kada most vidi okvir, utvrduje da li se u oznaci nalaze njegov ID i ID LAN-a koji prenosi okvir. Ako se nalaze, most prihvata okvir i prosleđuje ga do LAN-a koji je naveden u narednoj oznađ.

Na primer, pretpostavimo da Ašalje okvir do B (slika 10.8) koji sadrži oznaku rute L1-B5-L3-B4-L4. Oba mosta koja su povezana na L1 vide okvir, ali, pošto u oznaci iza L1 sledi B5, samo B5 prihvata okvir. Nakon toga, B5 prosleduje okvir do L3, gde ga vide mostovi B3 i B4. Slično tome, pošto B4 sledi iza L3, samo B4 prihvata okvir. Konačno, B4 prosleđuje okvir do L4, gde uređaj B prima okvir.

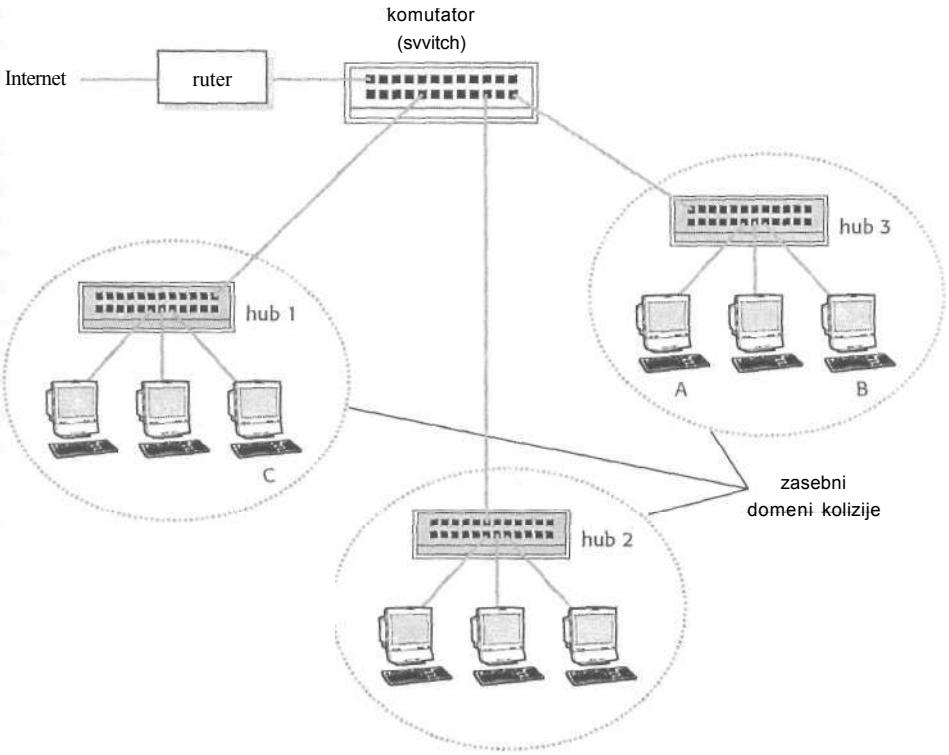
Da bi utvrdio rutu, uređaj šalje okvir do drugog uređaja sa pitanjem: "Gde si?". Prijemni uredaj dobija zahtev i reaguje na njega. Kada pošiljalac dobije odgovor, on utvrduje najbolju rutu. Ali, da li pošiljalac zna gde da pošalje inicijalni zahtev kada ne zna kuda da pošalje okvir? Ne zna. Mostovi moraju biti potpomognuti varijacijom algoritma plavljenja da bi se obezbedio prijem zahteva i odziva. Lltvrdivanje rute do odredišta naziva se otkrivanje rute.

Jedan od načina da se ovo postigne je slanje specijalnog emisionog okvira (all-route broadcast frame) do željenog odredišnog uređaja. U kontrolnom polju okvira naznačen je tip okvira i obavešteni su svi mostovi da treba da proslede okvir do svih LAN-ova, osim onog sa koga okvir potiče. Da bi se izbeglo neželjeno propagiranje okvira, polja oznake rute su prazna, a dužina polja za rutiranje u kontrolnom polju je 0. Kada most primi okvir, umeće sopstveni ID i ID LAN-a u polje rutiranja i inkrementira dužinu polja rutiranja. Da bi se izbeglo prethodno ponaenuto prosledivanje okvira, most proučava postojeće oznake rute. Okvir neće biti prosleđen do LAN-ova čiji je ID već deo oznake rute u dolazećem okviru.

Kada okvir konačno stigne do svog odredišta, polje rutiranja sadrži rutu koju je okvir prešao do odredišta. Odredište postavlja tu aitu u polje rutiranja neemisionog okvira (nonbroadcast frame) i šalje taj okvir nazad do izvornog uredaja. Osim toga, postavlja direkcioni bit u kontrolnom polju da bi mostovi bili obavešteni da oznaku arte treba da interpretiraju obrnutim redosledom. Kada most primi neemisioni okvir, on ga prosleduje, ili ispušta u skladu sa informacijama u polju rutiranja. Kada izvorni uredaj primi sve odzive, bira rutu koju će koristiti za sve naredne prenose do B. Moguće je da će proučiti cene (izračunate u toku emisije) i da će izabrati najjeftiniju. Alternativno, može da izabere rutu sa manje mostova, ili izbor može da izvrši na osnovu ID-ova mostova. Postoje i drugi načini za utvrdivanje ruta, ali taj tip mosta nije mnogo uobičajen, tako da mu ovde nećemo posvećivati pažnju.

Komutatori i komutirani Ethernet

Pošto je Ethernet postao dominantan LAN standard, ubuduće ćemo pretpostavljati da su naša četiri LAN-a Etherneti. To će nam pojednostaviti razmatranje i omogućiti fokusiranje na relevantne detalje. Komutatori (svvitches) izvršavaju iste funkcije kao i mostovi. Primarna razlika je u tome što most obično povezuje samo nekoliko LAN-ova, dok komutator može da ima nekoliko desetina portova, čime je otvorena mogućnost za više konekcija. Na slici 10.13 prikazana je jedna konfiguracija. Jedan komutator ima više portova; svaki od njih je povezan na hub.



SLIKA 10.13 *Konekcije izvedene pomoću hubova i homutatora*

Postoji konekcija i sa ruterom, koja obezbeđuje pristup Internetu. Rutere ćemo predstaviti nešto kasnije u ovom poglavlju. Svaki hub se sastoji iz više uredaja. Svi uredaji koji su vezani na hub izvršavaju Ethernet protokole i za njih važi jedinstveni domen kolizije. Pošto komutatori, kao i mostovi, selektivno prosleduju okvire, svaki port vodi ka drugom domenu kolizije. Zato, u stvari, imamo scenario sličan onome na slici 10.5, osim što, umesto više mostova, koristimo samo jedan komutator.

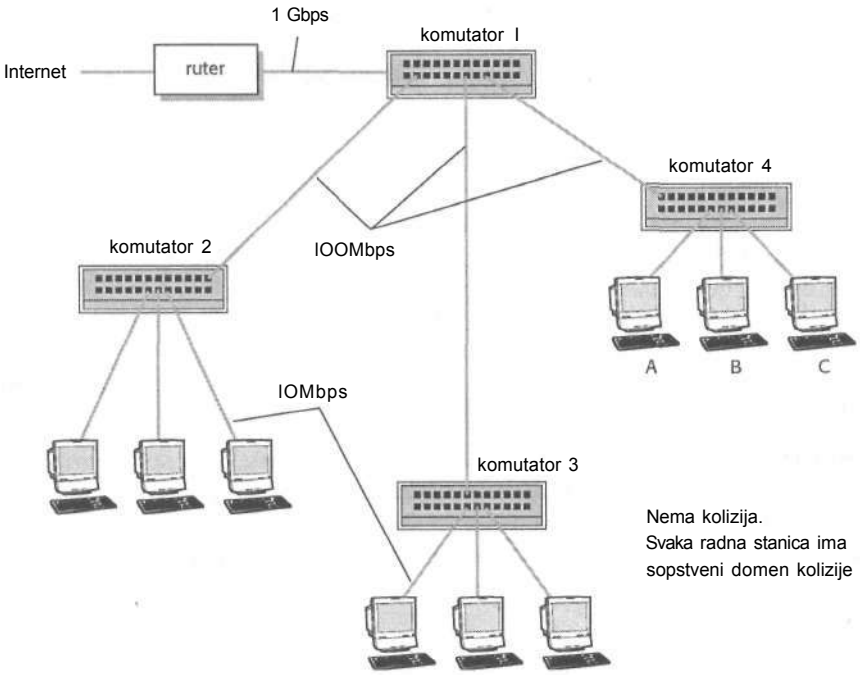
Dva uredaja povezana na isti hub komuniciraju direktno preko tog huba. Komutator ignoriše taj saobraćaj. Međutim, ako jedan uredaj šalje okvir do drugog koji je povezan na drugi hub, komutator prihvata ovaj okvir i prosleduje ga preko porta koji povezuje odgovarajući hub. U našem primeru komutator ignoriše sve okvire koje A šalje za B, ali prihvata i prosleduje okvire koje A šalje do C. Naravno, komutator mora da "nauči" lokacije uredaja na isti način na koji mostovi "uče" njihove lokacije.

Iako na slici 10.13 postoji više domena kolizija, ta topologija defmiše jedinstveni **emisioni domen (broadcast domain).** 48-bitna odredišna MAC adresa u Ethernet okviru može da naznači individualni uređaj, ili da ukaže na emisionu adresu. Ako se navede emisiona adresa, svaki komutator prosleduje emisioni okvir preko svih odlazeđh portova.

Svi uređaji prihvataju emisioni okvir kao da je reč o okviru koji je namenjen tim uređajima. U stvari, emisioni okvir dopire do svih uredaja na LAN-u. Ovo je uobičajeni način koji administrator mreže koristi za obaveštavanje svih korisnika o značajnim informacijama, kao što su problemi, isplanirane satnice, ili šta se juče desilo u prikazanoj epizodi omiljene TV "sapunske" serije.

Značajna činjenica u vezi topologije sa slike 10.13 je da i dalje može da dode do kolizija. Na slici 10.14 prikazana je sledeća konfiguracija, potpuno **komutirani Ethernet,** koji se implementira na nekim mestima. Fizičk,i on izgleda slično onome sa slike 10.13, osim što su svi hubovi zamenjeni komutatorima. Pošto su komutatori značajno pojeftinili (u vreme pripreme ove knjige 24-portni komutator mogao je da se kupi za oko 800 dolara), neki ljudi smatraju da nema potrebe da se koriste hubovi i da su sada zastareli. Glavna prednost komutiranog Etherneta je to što više nema kolizija - nigde! Komutirani portovi vode do zasebnih domena kolizije i svaki uredaj je povezan na zaseban port komutatora, tako da se uređaj nalazi u svom domenu kolizije. Osim toga, ako konekcije između radne stanice i komutatora funkcionišu u full-duplex modu, uređajima više nije potreban CSMA/CD protokol nadmetanja. Svi uredaji i dalje mogu da medusobno komuniciraju bez straha od kolizija i cela topologija i dalje definiše jedan emisioni domen.

Sledeća karakteristika komutatora je da bitske brzine portova mogu da budu različite. Ovo je izuzetno važno kada komutatori kombinuju saobraćaj iz raznih izvora preko jednog linka.



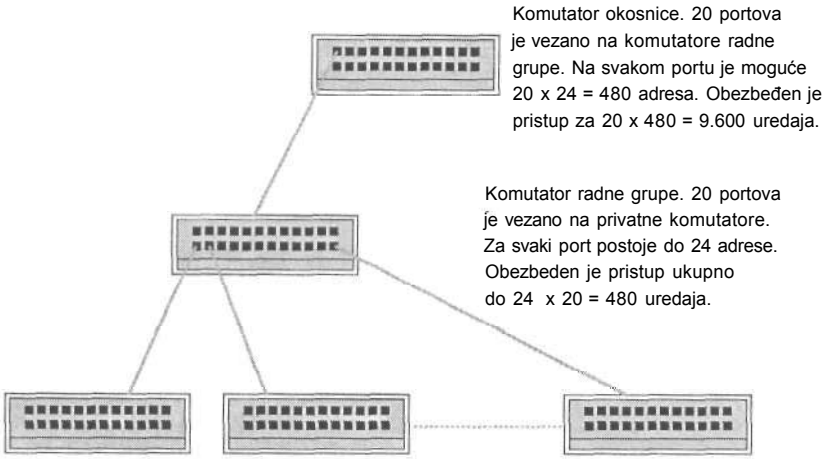
SLIKA 10.14 *Konekcije koriste isključivo komutatore.*

Taj link mora da podrži više bitske brzine od svakog vodećeg linka, ili se rizikuju značajna kašnjenja. Na slici 10.14 prikazan je 10 Mbps link za svaku radnu stanicu, ali linkovi od 100 Mbps između komutatora. Prema ruteru postoji link od 1 Gbps. Ovo nije jedina konfiguradja i moguće je uspostaviti linkove sa većim brzinama izmedu komutatora i prilično korišćenog servera. Moguće su razne konfiguradje.

Topologija sa slike 10.14 može da se proširi na više slojeva komutatora (slika 10.15). Komutator na dnu hijerahije označava se kao *privatni komutator,* jer na njemu nema više od jedne Ethernet adrese koja je dodeljena svakom portu. Naime, svaki port se povezuje direktno na jednu radnu stanicu. *Komutator radne grupe (uiorkgroup sivitch)* je smešten iznad privatnog komutatora u hijerarhiji. Primarna razlika ogleda se u tome što je svakom portu moguće dodeliti više Ethernet adresa. U našem slučaju su adrese za uredaje A, B i C dodeljene istom portu u komutatoru 1. Uređaji koji se povezuju na komutator radne grupe obično su postavljeni veoma blizu jedni drugih.

Poslednji komutator je *komutator okosnice (backbone switch),* koji obezbeduje pristup svim uređajima u domenu emisije. Iako izgleda da ne postoji granica u broju komutatora koje je moguće povezati, ona ipak postoji. Za većinu komutatora postoji maksimalni broj adresa koje je moguće dodeliti portu. Na slici 10.15 prikazano je da komutatori koji se nalaze bliže uredajima imaju manji broj adresa za svaki port. Komutatori koji se nalaze na višem nivou u hijerarhiji imaju ved broj adresa.

Kod full-duplex komutiranog Etherneta više nema kolizije okvira, mada je i dalje moguće da se neki okviri izgube. Na primer, komutator prihvata okvire od više radnih stanica i postavlja ih u red ćekanja za transfer preko odlazedh portova. Ako je saobraćaj sporadičan i ako okviri dolaze isuviše brzo, moguće je da se bafer napuni i tada komutator može da odbaci okvir. Ethernet protokol rešava takve situacije koristeđ MAC kontrolni podsloj, koji leži izmedu originalnih MAC i LLC podslojeva sloja veze.



SLIKA 10.15 *Privatni komutatori, komutatori radne grupe i komutator okosnice*

MAC kontrolni podsloj definiše Pause okvir za kontrolu toka, kao što je prikazano u sledećem primeru:

1. Uređaj A prima niz uredaja od uredaja B. Baferi u A počinju da se pune.
2. A šalje do B Pause okvir koji sadrži pozitivnu vrednost tajmera.
3. B prima Pause okvir i prestaje da šalje nove okvire do A sve dok ne istekne naznačeni period.

Podsloj dopušta i da vrednost tajmera bude 0. Možda na prvi pogled deluje čudno, ali na taj način je moguće ponovo osposobiti prenos okvira. Ako uređaj B čeka, jer je prethodno primljen Pause okvir, i tek kada primi drugi sa tajmerom čija je vrednost 0, može da nastavi slanje okvira.

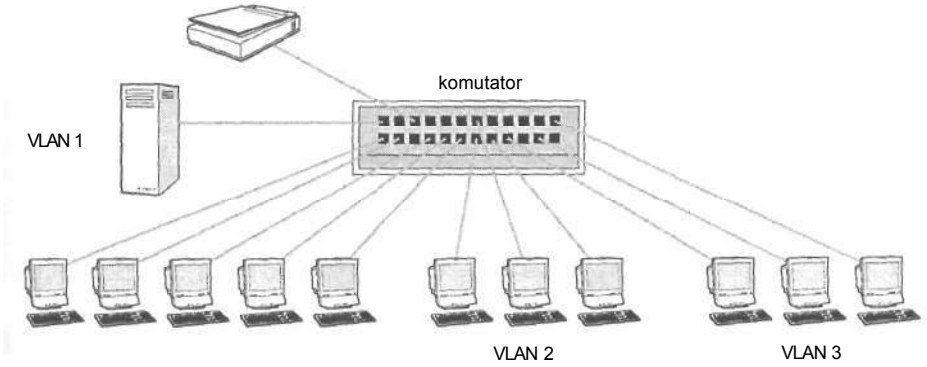
Virtuelni LAN-ovi

Sledeća opcija koju omogućavaju Ethernet komutatori je kreiranje virtuelnog LAN-a (VLAN). Razvoj VLAN-ova je bio iniciran potrebama administratora mreža za boljim upravljanjem saobraćaja na mreži, poboljšanjem zaštite i razdvajanjem korisnika u radne grupe. Na primer, u mnogim organizacijama se praktikuje angažovanje grupa zaposlenih na različitim projektima. Svi zaposleni koji su angažovani na jednom projektu čine radnu gaipu. Bilo bi zaista pogodno ako bi se mreža mogla konfigurisati tako da se personalni kompjuteri iz jedne radne grupe razdvoje od druge radne grupe. Tako bi se saobraćaj odvijao preko više linkova i postigla bi se bolja ravnoteža ukupnog saobračaja na mreži.

Jedna mogućnost je kreiranje LAN-a koji se sastoji od svih uredaja dodeljenih ljudima jedne radne grupe. Svaka radna grupa ima svoj LAN i administrator mreže koristi aitere (uredaje za rutiranje koji se koriste na sloju 3) za razdvajanje svih LAN-ova (kasnije ćemo razmotriti rutere, a za sada smatrajte da su to uredaji koji razdvajaju LAN-ove na različite i autonomne entitete). Osim toga, svaki LAN odgovara jednom domenu emisije. Drugim rečima, ako uređaj šalje emisioni okvir, okvir ostaje u okviru njegovog LAN-a i ne prolazi kroz ruter.

Problem kod ovog pristupa je što se zaposleni u jednog radnoj grupi možda ne nalaze blizu jedni drugih i zato je pomalo čudno što se postavljaju na isti LAN. Grupu mogu da čine zaposleni sa različitim sposobnostima iz različitih odeljenja sa različitih spratova, ili čak iz različitih zgrada. Tražimo način da logički povežemo grupe uređaja koji fizički mogu da budu raspoređeni na većim udaljenostima, ali logički posmatrano pripadaju istoj grupi. VLAN predstavlja grupu uređaja koja je logički grupisana nezavisno od njihovih fizičkih lokacija.

Na slici 10.16 prikazana je ta ideja. Postoje tri grupe uređaja. VIANl sadrži pet personalnih kompjutera, server i skener, a VLAN2 i VLAN3 po tri personalna kompjutera. Logika u komutatoru pridružuje oznaku VLAN-a svakom portu. Tako portovi defmišu koji uredaji sačinjavaju koji VLAN. Osim toga, svaki VLAN definiše nezavisni domen emisije. Drugim rečima, koristeći protokole sloja 2, uređaji u VLANl mogu da komuniciraju samo sa ostalim uredajima u VLANl.



SLIKA 10.16 *Komutator povezuje više LAN-ova.*

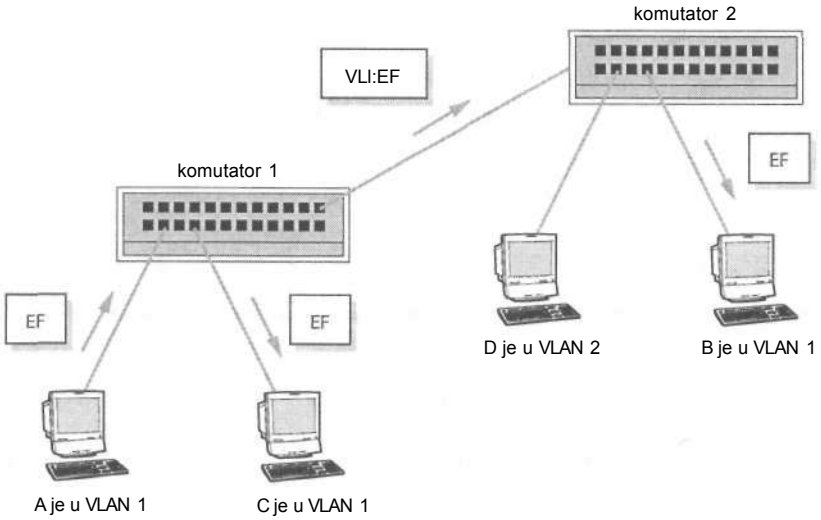
Ako uredaj sa VLANl pošalje emisioni okvir, komutator šalje taj okvir samo do portova sa oznakom VLANl. Slično važi i za VLAN2 i VLAN3.

Ova tehnika je korisna zbog nekoliko razloga. Jedan je činjenica da korisnik može da pošalje emisioni okvir samo za uredaje iz istog VLAN-a, čime je obezbeđena mnogo efikasnija komunikacija izmedu njih i izbegnuto je nepotrebno slanje podataka do uređaja izvan grupe. Sledeći razlog je bezbednost. Protokoli sloja 2 ne rutiraju okvire sa jednog VLAN-a u drugi. Time je ograničen saobraćaj i omogućena je bolja kontrola toka saobraćaja. Osim toga, prednost je i to što VLAN može da se konfiguriše nezavisno od lokacije uredaja. Na slici 10.16 prikazan je samo jedan komutator, a moguće je postaviti više komutatora čiji će portovi imati iste VLAN oznake portova. Tako je moguće povezati uredaje na različite portove na različitim komutatorima (i, samim tim, na druge lokacije) koji mogu da pripadaju istom VLAN-u. Domen emisije sadrži uredaje sa različitih lokacija i uredaje koji se nalaze blizu jedni drugih mogu, ali ne moraju da pripadaju istom domenu emisije.

Mogućnost kreiranja VLAN-a nezavisno od fizičke lokacije uređaja stvara robustnija dinamička okruženja. Pretpostavimo da neki zaposleni dobije premeštaj iz jedne radne grupe u drugu. Potrebno je samo postaviti novu oznaku VLAN-a na port na koji su priključeni uređaji tog zaposlenog, koji ne mora da se premešta, niti je potrebno bilo šta prevezivati. Komutator jednostavno vidi taj port kao deo drugog VLAN-a i omogućava tom zaposlenom da komunicira sa sasvim novom grupom.

Kako komutator donosi odluku kako će rutirati okvir? Sečate se da Ethernet okvir ima odredišnu adresu, ali nije bilo pomena o bilo kakvim VLAN oznakama. Kada komutator dobije Ethernet okvir, kako odlučuje šta će da uradi? Ako se rutiranje zasniva na Ethernet adresama, gde se tu U celu "priču" uklapa identifikatorVLAN-a?

Slika 10.17 će nam pomoći da objasnimo kako to funkcioniše. Pretpostavimo da se A, B i C nalaze u VLANl, a da je D u VLAN2. Osim toga, A i C su povezani na jedan komutator, a B i D na drugi.



SLIKA 10.17 *VLAN-ovi postavljeni između komutatora*

Kada A šalje Ethernet emisioni okvir, dešava se sledeće:

1. Ethemet okvir (označen kao EF na slici) poslat je do komutatora I. Komutator I primećuje da okvir koji stigao preko porta ima VLANl oznaku.
2. Port povezan na C takode ima oznaku VLANl, tako da komutator šalje okvir na taj port.
3. Sledeći port povezuje komutator 1 sa komutatorom 2. Pošto postoje VLANl uredaji u tom smeru, dodeljeni port takode ima VLANl oznaku. Napomenimo da i drugi VLAN-ovi mogu da budu dodeljeni istom portu.
4. Komutator 1 dodaje identifikator VLAN-a u Ethernet okvir (prikazano kao VLl :EF) i šalje ga na port komutatora 2.
5. Komutator 2 prima proširenl okvir. Portu preko koga komutator 2 prima okvir može da bude dodeljeno nekoliko VLAN-ova. Medutim, pošto okvir sadrži VLAN ID, komutator 2 zna kojem VLAN-u pripada. Zbog toga, šalje okvir direktno do B preko odgovarajućeg porta. On *neće* poslati okvir do D, jer se D ne nalazi u VLANl.

Iako je u prethodnom razmatranju pretpostavljeno da Je komutator dodeljen VLAN-u sa ID-om porta, postoje i drugi metodi. Na primer, komutator može da održava tabelu u kojoj je nizu MAC adresa dodeljen ID VLAN-a. U tom slučaju, kada komutator primi Ethemet okvir od uredaja, on traži izvornu adresu okvira, umesto porta preko koga je okvir primljen. Izvoma adresa se koristi za indeksiranje tabele i izvlačenje ID-a VLAN-a. Ako se uređaj premesti, neće biti nikakvih promena u komutatoru, jer MAC adresa zavisi od mrežne kartice u tom uredaju, a ne od njegove lokacije.

VLAN ID može da se dodeli i na osnovu informacija sa sloja 3. Ovo je malo složenije, jer zahteva implementiranje nekih protokola sa sloja 3 u komutatoru, tako da komutator malo podseća na ruter. U sledećem poglavlju predstavićemo Internet Protocol (IP), a za sada ćemo istaći samo da se IP paketi mogu postaviti u polja Data konkretnog okvira. IP paket sadrži elemente kao što je IP izvorna adresa (koja se razlikuje od MAC adrese) i specifikaciju protokola iznad IP-a koji se takode koriste. Tako VLAN može da se definiše u skladu sa IP adresom uredaja, ili, čak, korišćenim protokolima viših slojeva. Sve što komutator (ruter) treba da uradi je da izvuče paket iz okvira, prouči izvornu IP adresu, ili identifikator protokola i da utvrdi pridruženu oznaku VLAN-a. U referenci [Fo03] možete da pronađete više detalja o implementiranju VLAN-ova.

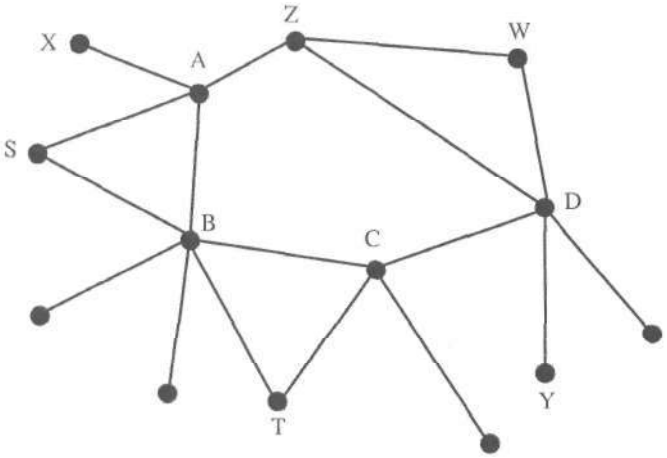
10.4 Konekcije sloja 3

Prethodno predstavljene lokalne mreže (LAN-ovi) i konekcije slojeva 1 i 2 obično "pokrivaju" manje geografske oblasti i ograničene su na jednu zgradu, ili grupu zgrada. Dizajnirane su pomoću relativno jednostavne topologije magistrale, prstena, ili zvezde. Medutim, mreže šireg geografskog područja (WAN - wide area networks) prostiru se preko cele zemaljske kugle i zahtevaju sofisticiranije tehnike.

Uzmimo opet primer sistema gradskih saobraćajnica. U mnogim gradovima postoji jedna glavna saobraćajnica koja prolazi kroz centar (zajednička magistrala), ili postoje kružne zaobilaznice (prsten). Ako grad nije veliki, saobraćaj se obično dobro kontroliše. AJi, šta je sa velikim oblastima, kao što su regioni, ili cele države? Ne bi imalo smisla dizajnirati jedan autoput kroz "srce" Amerike, ili zaobilaznice koje bi se nalazile na granicama sa Kanadom i Meksikom. Umesto toga, dizajnira se složena strateška veza glavnih autoputeva, zaobilaznica i puteva kroz države, regione i gradove. Ovakav tip sistema nije prikladno kategorizovati kao magistralu, prsten, zvezdu, ili neku njihovu jednostavnu kombinaciju. Reč je o mnogo složenijem sistemu.

Poput nacionalnog sistema autoputeva, topologije WAN mreža su složene i obično mogu da se uvrste negde između jednostavne magistrale, prstena, ili zvezde i potpuno povezane topologije. Sa složenijim topologijama dolaze i složeniji protokoli za uspostavljanje konekcija i donošenje odluka o rutiranju. Cinjenica da postoji toliko veliki broj načina za dolazak iz jedne tačke do druge situaciju čini još složenijom. Na primer, ako živite u Winoni (u Minesoti) i želite da idete na odmor u Charleston (u Južnoj Karolini), verovatno ćete neko vreme provesti proučavajući mape da biste utvrdili kuda je najbolje da idete. Slično tome, ako želite da prenesete informacije izmedu kompjutera koji se nalaze na te dve lokacije, a koje su povezane preko WAN-a, i mrežni protokoli moraju da utvrde kuda će se te informacije proslediti.

Da bi cela "priča" bila još komplikovanija, napomenimo da neki linkovi mogu da "otkažu". Sta u tom slučaju mrežni protokol radi sa svim podacima koji su putovali tom rutom? U nekim slučajevima ruta može da postane toliko popularna da se preko nje počne prenositi prevelika količina podataka. Rezultat su zagušenja, a ponekad i "otkazi" linka. Da li mrežni protokoli mogu da izbegnu ovakve situacije? Ako ne mogu, šta mogu da urade da bi se ti efekti sveli na najmanju moguću meru? Poslužimo se opet analogijom sa autoputevima - ovi problemi su slični ograničavanju, ili smanjivanju saobraćaja u vreme velikih praznika.



SLIKA 10.18 *Generalizovana mrežna topologija*

Većina nas poznaje te probleme i zna da se tu ne može uraditi ništa mnogo više nego da se ostane kod kuće.

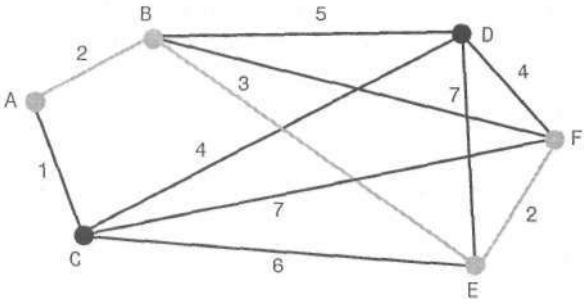
Na slici 10.18 prikazana je generalizovana mrežna topologija. Neki uređaji često komuniciraju direktno sa više uredaja u poredenju sa nekim drugim. Nesumnjivo, oni predstavljaju lokacije u gusto naseljenim oblastima, ili lokacije sa velikim obimom razmene informacija. Uredaji sa manjim brojem konekcija mogu da imaju manje potrebe - mogu da se uporede sa udaljenim područjima. Na primer, čvorovi A i B predstavljaju mesta u velikim gradskim područjuma, a X i Y udaljene delove države.

Pretpostavimo da jedan uredaj treba da komunicira sa dmgim uredajem sa kojim nema direktnu vezu. Mrežni protokoli moraju da pronadu najbolju putanju koja ih povezuje. Na primer, pretpostavimo da na slici 10.18 X šalje nešto za Y. Postoje dve moguće putanje: X-A-B-C-D-Y i X-A-Z-D-Y. Koja putanja je bolja? Odgovor obično zavisi od cena i vremena koje je potrebno za slanje informacija preko svake putanje.

Poređenje nije uvek jednostavno izvodljivo. Gledajući sliku 10.18, možda možete da pomislite da je putanja preko A, Z i D bolja, jer je kraća od druge alternative. Medutim, kraća putanja ne mora da bude uvek i bolja, a to mogu da potvrde svi koji su putovali automobilom. Mnogi ljudi će izabrati put sa većom kilometražom ako to znači da će ići putem sa više traka, u boljim uslovima, ili ako postoji više restorana sa brzom hranom nego u slučaju kraće alternative. Iako će duže voziti, možda će stići pre i sa manje poteškoća.

Tabele rutiranja

Slično prethodno predstavljenim mostovima, mrežni čvorovi mogu da koriste tabele rutiranja. Slično onim tabelama iz sekcije 10.3, tabele rutiranja ne definišu celu rutu. Umesto toga, definišu sledeći čvor u okviru rute do traženog odredišta i cenu puta do tog čvora.



SLIKA 10.19 *Mreža sa dodeljenim cenama konekcija*

Na primer, uzmimo mrežu sa slike 10.19, kod koje su svim konekcijama između susednih čvorova dodeljene cene. Pretpostavimo da želimo da pronademo najjeftiniju rutu,* onu koja ima minimalnu sumu cena konekcja između susednih čvorova na ruti. Na primer, postoji nekoliko ruta između čvorova A i F, ali najjeftinija je ona koja ide od A do B (cena 2), od B do E (cena 3) i od E do F (cena 2), što daje ukupnu cenu 7.

Na slici 10.20 date su parcijalne tabele rutiranja čvorova A, B i E (uskoro ćemo objasniti kako su ove tabele kreirane). Tabela čvora A ukazuje da sve što je namenjeno čvorovima B, E, ili F treba da se pošalje direktno do B, gde će tabela rutiranja čvora B definisati sledeći čvor za najjeftiniju rutu. Slično tome, sve što je namenjeno za C, ili D treba da se pošalje do C. Odatle, tabela rutiranja za C ukazuje na sledeći korak.

Odredište	Sledeći čvor	Cena
B	B	2
C	C	1
D	C	5
E	B	5
F	B	7

(a) Parcijalna tabela rutiranja za čvor A

Odredište	Sledeći čvor	Cena
D	D	5
E	E	3
F	E	5

(b) Parcijalna tabela rutiranja za čvor B

Odredište	Sledeći čvor	Cena
F	F	2

(c) Parcijalna tabela rutiranja za čvor E

SLIKA 10.20 *Parcijalne tabele rutiranja za čvorove A, B i E*

* Moguće je da se svim konekcijama dodeli cena 1. U tom slučaju, najjeftinija putanja postaje najkraća, sa cenom koja odgovara broju direktnih veza na putanji. Ovo se ponekad naziva *broj skokoua (hop count);* uskoro čemo reći nešto više i o tome.

Ilustracije radi, pretpostavimo da aplikacija u čvoru A želi da pošalje podatke do čvora F. Logika u čvoru A traži zapis u tabeli rutiranja sa čvorom F kao odredištem. Zapis "kaže" da je čvor B sledeći čvor u ruti i mrežni protokol šalje podatke do B. Logika u B proučava svoju tabelu rutiranja za zapis koji odgovara ruti ka čvoru F. Treći zapis u tabeli ukazuje da podatke treba poslati do čvora E. Konačno, tabela rutiranja u čvoru E ukazuje da je F sledeći čvor u sklopu rute.

Ko definiše tabele rutiranja i kako? Proces u kojem se definišu tabele rutiranja naziva se **algoritam rutiranja.** Postoji nekoliko osnovnih tipova rutiranja; mi ćemo ovde predstaviti četiri: centralizovano, distribuirano, statičko i adaptivno. Nakon toga, analiziraćemo neke specifične algoritme za rutiranje.

Centralizovano rutiranje

Centralizovano rutiranje podrazumeva da se sve informacije o medusobnim vezama generišu i održavaju na jednoj centralnoj lokaciji. Nakon toga se sa te lokacije informacije emituju do svih mrežnih čvorova, tako da svaki čvor može da definiše svoje tabele rutiranja. Jedan mogući način za održavanje informacija o rutiranju na centralnoj lokaciji je preko **matrice rutiranja.** Ona se sastoji od redova i kolona za svaki čvor u mreži. Redovi odgovaraju izvornim čvorovima, a kolone odredišnim čvorovima. Zapis koji je određen konkretnim redom i kolonom ukazuje na prvi čvor na ruti od izvora do odredišta. Na osnovu tog zapisa, moguće je izvesti celu rutu.

Na slici 10.21 prikazana je matrica rutiranja za mrežu sa slike 10.19. Kao i ranije, selektovane su najjeftinije rute. U slučaju da postoje dve rute sa istom najnižom cenom, ruta se bira proizvoljno. Razmotrimo ponovo rutu od A do F. Prema prvom redu i šestoj koloni matrice, sledeći čvor u najjeftinijoj ruti je B. Naredni se utvrđuje razmatranjem rute od B do F. Na osnovu drugog reda i šeste kolone, dobijamo da je to čvor E. Konačno, čvor koji sledi iza E (čvor F) pronađen je na osnovu petog reda i šeste kolone. Dakle, ruta ide od A do B do E do F.

Prilikom kreiranja tabele rutiranja za mrežni čvor neophodno je da red u matrici odgovara tom čvoru. Na primer, tabela rutiranja za A na slici 10.20 (minus cena)* ista je kao i prvi red matrice. Slično važi i za sve ostale čvorove.

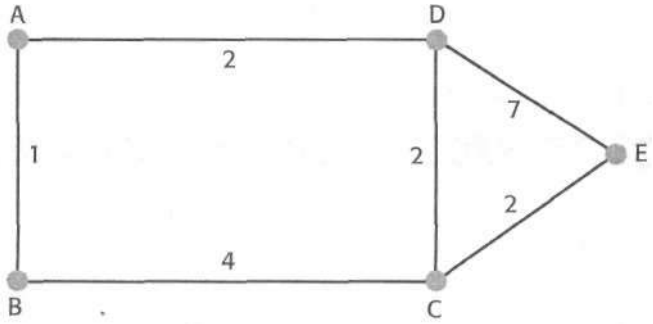
izvorni čvor	A	B	C	D	E	F
A	—	B	C	C	B	B
B	A	—	A	D	E	E
C	A	A	—	D	E	F
D	C	B	C	—	F	F
E	B	B	C	F	—	F
F	E	E	C	D	E	—

SLIKA 10.21 *Matrica rutiranja za vnrežu sa slike 10.19*

Distribuirano rutiranje

Distribuirano rutiranje podrazumeva da nema centralne kontrole. Svaki čvor mora nezavisno da utvrdi i da održava svoje informacije o rutiranju. To obično izvodi tako što zna ko su mu susedi, izračunava cenu puta do svog suseda i utvrduje cenu za onog koji bi poslao podatke do željenog odredišta. Svi susedi rade isto. Na osnovu informacija od svih suseda moguće je izvesti sopstvenu tabelu rutiranja. Ovaj metod je složeniji od centralizovanog rutiranja. Razlog je činjenica da zahteva nezavisno komuniciranje sa svim susedima.

Teško je proceniti složenost ovog pristupa, jer primeri obično daju globalni prikaz mreže sa svim konekcijama i njihovim cenama. Ovaj opšti pregled može da nas stavi u neravnopravan položaj, jer, za razliku od nas, znanje čvora o izgledu cele mreže je veoma ograničeno. Ilustracije radi, razmotrimo mrežu sa slike 10.22. Pretpostavimo da svaki čvor inicijalno zna samo cene puta do svo jih suseda; kasnije ovim informacijama može da doda ono što sazna od svojih suseda. Na primer, Ainicijalno zna samo da može da pošalje nešto do B (cena =1), ili do D (cena = 2). On uopšte ne zna da čvorovi C i E postoje. Isto važi i kod drugih čvorova. Medutim, ako susedni čvorovi mogu da komuniciraju, A saznaje za identitet suseda čvorova B i D, tj. za čvorove C i E. Saznavši cene čvorova B i D za dolazak do tih čvorova, a pošto od ranije zna za cene do B i D, A može da izračuna cene dolaska do C i E. Periodičnom razmenom informacija o susednim čvorovima svaki čvor saznaje identitete drugih čvorova na mreži i otkriva najjeftinije rute do njih. Uskoro ćemo prikazati specifične algoritme koji se koriste u ovom procesu.



SLIKA 10.22 *Primer mreže za distribuirano rutiranje*

″ Ovde u matricu rutiranja nismo uključili cene, ali to se jednostavno izvodi smeštanjem cene uz svaki Čvor u matrid.

Statičko rutiranje

Statičko rutiranje podrazumeva da svaki čvor utvrdi svoju tabelu rutiranja i da je nakon toga više ne menja. Drugim rečima, najjeftinija putanja ne zavisi od vremena. Osnovna pretpostavka je da se uslovi koji odreduju definicije u tabeli ne menjaju. Ovo može ponekad da bude validna pretpostavka, jer cene često zavise od rastojanja i bitske brzine između prelaznih čvorova. Osim kada se vrši nadogradnja glavne opreme, ili prelazak na sasvim drugačiju opremu, ovi parametri se neće menjati.

Adaptivno rutiranje

Statičko rutiranje dobro funkcioniše sve dok se uslovi na mreži ne menjaju. Ipak, u nekim mrežama ovo nije validna pretpostavka. Na primer, ako cena svakog linka zavisi od mrežnog saobraćaja, cena zavisi od vremena. Razmotrimo problem slanja paketa od čvora A do čvora E na slici 10.22. Optimalna ruta je A-D-C-E. Pretpostavimo da nakon što A prenese pakete do D, dolazi do povećanja cene linka D-C, **i** to na 10, zbog velike količine saobraćaja. Najjeftinija ruta od A sada glasi A-B-C-E; ruta koja je inicijalno izabrana sada ima previsoku cenu. U tom slučaju, bilo bi jeftinije poslati paket nazad do A i početi sve iz početka. Strategija adaptivnog **rutiranja** dopušta mrežnom čvoru da reaguje na ovakve promene i da, u skladu sa njima, ažurira svoju tabelu rutiranja.

Postoje neke "zamke" u ovom sistemu. Na primer, pretpostavimo da u našem tekućem primeru D šalje paket nazad do A, a da se, zatim, cena linka A-B povećava na 10. Logičan izbor je slanje paketa nazad do D. Sada možete da shvatite problem: paket će neprestano ići napred-nazad i nikada neće stiđ do svog odredišta. Jedna tehnika koja izbegava ovakve situacije održava brojač u zaglavlju paketa koji se inkrementira sa svakim prenosom. Ako brojač premaši odredenu vrednost, paket se uklanja sa mreže. U takvim situacijama logika za rutiranje ne garantuje isporuku paketa.

Tabela 10.2: Tipovi rutiranja

Tip rutiranja	Prednosti	Nedostaci
Centralizovano rutiranje	Jednostavan metod, zato što se kontrola rutiranja vrši sa jedne lokacije	"Otkaz" na centralnoj lokaciji, ili bilo kom od linkova ima ozbiljan efekat na obezbedivanje informacija o rutiranju za čvorove na mreži.
Distribuirano rutiranje	"Otkaz" čvora, ili linka ima manji efekat na obezbedivanje tačnih informacija o rutiranju.	Razmena informacija je složenija. Čvorovima može da bude potrebno više vremena da "nauče" uslove na udaljenim lokacijama.
Statičko rutiranje	Jednostavan metod, zato što čvorovi ne moraju neprestano da izvršavaju algoritam za rutiranje	Ne uzimaju se u obzir promene uslova. Dobra ruta može da postane veoma loša.
Adaptivno rutiranje	Obezbeduje najnovije informacije sa stanovišta cena linka.	Visoki troškovi, zato što čvorovi moraju da održavaju tekuće informacije. Prenos informacija u skladu sa promenama uslova povećava saobraćai na mreži.

Ipak, obično postoji protokol za kontrolu toka koji se izvršava na višem sloju, koji će naznačiti da je vreme za isporuku paketa isteklo i iniciraće ponovno slanje paketa.

U opštem slučaju adaptivno rutiranje je teško efikasno implementirati. Čvorovi mogu da prate promene uslova samo ako dobijaju izveštaje od ostalih čvorova o cenama linkova. Ovi izveštaji povećavaju saobraćaj na mreži i mogu da doprinesu daljoj promeni uslova. Osim toga, oduzimaju vreme, jer čvor mora da "učl" promenjene uslove, tako da dok čvor to "nauči", taj uslov možda neće više biti validan.

U tabeli 10.2 dati su kraći pregled i pbređenje četiri tipa rutiranja.

10.5 Dijkstrin algoritam

Dijkstrin algoritam (ref. [Di59]), ponekad nazvan algoritam najkraćeg puta, ili agoritam pretraživanja unapred, predstavlja centralizovani statični algoritam, iako može da se učini adaptivnim ako se periodično izvršava. Zahteva da čvor koji ga izvršava ima informacije o cenama linkova izmedu čvorova na mreži.

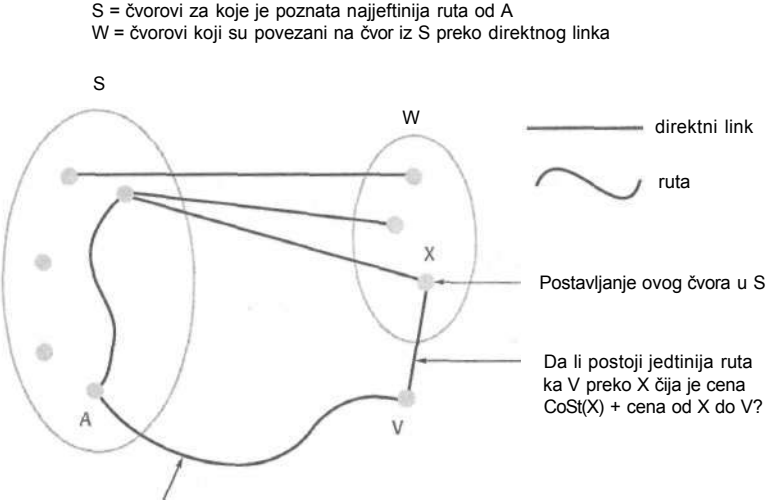
Svaki čvor izvršava Dijsktrin algoritam da bi utvrdio najjeftiniju rutu do svakog drugog čvora na mreži. U slučajevima kada cena rute jednostavno predstavlja broj prelaznih čvorova, najkraća ruta je, ujedno, i najjeftinija. Algoritam je i iterativan; kreira skup čvorova, jedan po jedan, u okviru svake iteracije. Za svaki čvor u skupu je poznata najjeftinija ruta do njega od bilo kog konkretnog čvora.

Na slici 10.23 dat je opšti pregled algoritma. Inicijalno, definiše se skup S koji sadrži samo čvor A, koji izvršava algoritam. Nakon toga se definiše funkcija koja za svaki čvor X definiše Cost(X) = cena najjeftinije rute od A do X, čiji su međučvorovi deo skupa S. Pošto S inicijalno sadrži samo čvor A, Cost(X) je cena direktnog linka od A do X. Ako takav link ne postoji, za Cost(X) je dodeljen proizvoljno veliki broj. Funkdja Prior(X) u algoritmu sadrži čvor koji prethodi čvoru X u sklopu najjeftinije rute.

Algoritam sadrži petlju. Sa svakim prolaskom utvrduje se skup W, koji sadrži sve čvorove iz S, sa direktnim linkom do nečega u S (slika 10.24). Bira se jedan čvor X iz W za koji je Cost(X)<Cost(Y) za bilo koji daigi čvor *Y* iz W.

```
Definisanje S kao skupa cvorova. Inicijalno S sadrzi cvor A.

Definisanje Cost(X) kao cene najjeftinije rute od A do X koriscenjem samo cvorova iz S
   (osim cvora X). Inicijalno, Cost(X) je cena linka od A do X. Ako takav link
   ne postoji, onda za Cost(X) moze biti izabrana proizvoljno velika vrednost (veca od
   cene bilo koje druge moguce rute). Za cvorove koji su povezani sa A definise
   se Prior(X) = A.
do {
   Utvrdjivanje skupa cvorova koji se ne nalaze u S, ali su povezani sa nekim cvorom
   iz S. Nleka se taj skup zove W.
   Biranje cvora X u W za koji je CoSt(X) minimalna. Dodavanje X u skup S.
   Za svaki cvor V koji nije u S, definisanje Cost(V) = minimum{Cost(v), Cost(X)+cena
   linka koji povezuje X i V}. Ako je Cost(V) promenjeno, definise se Prior(V) = X.
}
while svi cvorovi nisu u S.
```

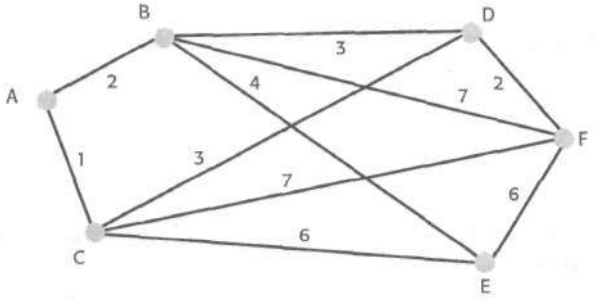
SLIKA 10.23 *Dijkstrin alsoritam*



SLIKA 20.24 *Dodavanje čvorova u S korišćenjem Dijsktrin algoritma*

Nakon toga, X se dodaje u S i ažurura se funkcija cene Cost(V) za svako V koje još uvek nije u S. Poredi se tekuća vrednost Cost(V) sa Cost(X), plus cena direktnog linka od X do V, Ako je druga vrednost manja, algoritam redefmiše Cost(V) tako da ima tu vrednost. Namera je da se utvrdi da li dodavanje X u S omogućava jeftiniju rutu od A do V preko čvorova iz S.

Tačnost ovog algoritma nije odgledna, a njeno dokazivanje prelazi ciljeve ovog teksta. Ako želite formalniju predstavu, pogledajte referencu [Ah83]. Primenićemo ovaj algoritam na primer mreže sa slike 10.25, a u tabeli 10.3 prikazane su vrednosti koje algoritam generiše kada se primeni na ovu mrežu. U koraku 1 skup S sadrži samo izvorni čvor A. Jedini čvorovi koji su povezani na A su B i C i cene tih grana su 2 i 1, respektivno.



SLIKA 10.25 *Mreža sa dodeljenim cenama konekcija*

Tabela 10.3: Vrednosti definisane Dijkstrin algoritmom za mrežu sa slike 10.28

				Funkcija Cost za					Funkcija Prior za				
Korak	**S**	**W**	**X**	**B**	**C**	**D**	**E**	**F**	**B**	**C**	**D**	**E**	**F**
1	{A}	{B, C}	C	2	1	c o	c o	o o	A	A	—	—	—
2	{A, C}	{B, D, E, F}	B	2	1	4	7	8	A	A	C	C	C
3	{A, B, O	{D, E, F}	D	2	1	4	6	8	A	A	C	B	C
4	{A, B, C, D}	{E, F}	E	2	1	4	6	6	A	A	C	B	D
5	{A, B, C, D, E}	{F}	F	2	1	4	6	6	A	A	C	B	D

Tako je Cost(B) = 2 i Cost(C) = 1. Inicijalne vrednosti za Cost(D), Cost(E) i Cost(F) su proizvoljno velike i označene su simbolom za beskonačnost (°o). Osim toga, Prior(B) = A i Prior(C) = A. Pošto algoritam još uvek nije otkrio rute ka D, E i F, funkcija Prior je nedefinisana u tim čvorovima.

Kada udemo u petlju, skup W sadrži čvorove B i C, jer su samo oni povezani sa A. Zatim, pošto je Cost(C) < Cost(B), biramo X = Ci dodajemo ga u S. Poslednja linija u petlji sada zahteva da, proučimo Cost(V) za svaki čvor V koji još uvek nije dodat u S. On se sastoji od čvorova B, D, E i F. Pošto funkcija Cost predstavlja najjeftiniju putanju od A preko čvorova iz S, moramo da se zapitamo da li dodatni čvor u S obezbeduje jeftiniju aitu. Drugim rečima, razmatramo sve čvorove V koji se ne nalaze u S. Ako je Cost(C), plus cena direktnog linka koji povezuje C sa V manja od Cost(V), ruta od A do C koja prati link od C do V predstavlja jeftiniju rutu. Tabela 10.4 sadrži neophodno poređenje za svaki čvor V koji nije u S. Za čvorove D, E i F kasnije vrednosti su manje i C je postavljen kao prethodni čvor. Drugi red u tabeli 10.3 reflektuje te promene.

Drugi prolazak kroz pedju nastavlja u sličnom "maniru". Čvor B se dodaje u S, zato što je Cost(B) najniža cena medu čvorovima koji se ne nalaze u S. Osim toga, uključivanje B u S obezbeduje jeftiniju rutu do E (preko čvorova iz S). Treći red u tabeli 10.3 pokazuje kako se menjaju zapisi ispod E, Cvor B sada prethodi čvoru E, a Cost(E) je sada 6. Vežbe radi, možete da ispratite algoritam i proverite da li su 4. i 5. red iz tabele 10.3 tačni.

Tabela 10.4: Poređenje cena za Dijsktrin algoritam

V	Cost(V)	Cost(C) + cena linka koji povezuje C sa V
B	2	Nema linka od C do V
D	∞	1 + 3 = 4
E	∞	1 + 6 = 7
F	∞	1 + 7 = 8

Kada se algoritam završi, poslednji red u tabeli pokazuje da su cene najjeftinije rute do B, C, D, E, i F 2, 1, 4, 6 i 6, respektivno.

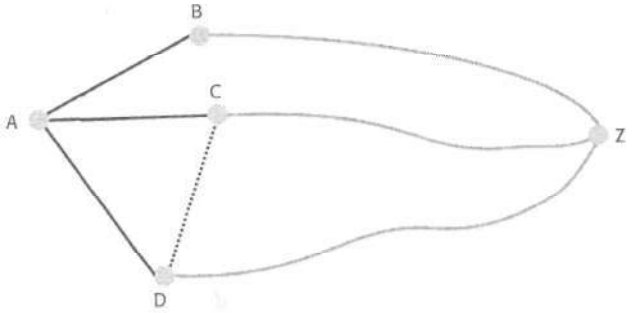
Funkcija Prior može da se koristi za oporavljanje konkretne aite. Na primer, ako želite konkretnu rutu od A do F, Prior(F) = D ukazuje da je D prethodio čvoru F na toj ruti. Prior(D) = C ukazuje da C prethodi čvoru D, a Prior(C) = A znači da je A prethodio C. DaWe, najjeftinija ruta od A do F je A-C-D-F.

10.6 Bellman-Fordov algoritam

Prethodno opisani Dijkstrin algoritam daje najjeftiniju putanju kretanjem unapred iz bilo kog konkretnog izvora. Sledeći pristup podrazumeva kretanje unazad od željenog odredišta. Bellman-Fordov **algoritam** (ref. [Fo62]), poznat i kao **agoritam pretraživanja unazad,** ili **algoritam vektorskog rastojanja,** koristi takav pristup. Zasniva se na sledećem principu. Neka CoSt(A, Z) bude cena najjeftinije rute od čvora A do čvora Z. Pretpostavimo da A ima direktnu konekciju sa čvorovima B, C, ..., D (slika 10.26). Onda je

Cost(A, Z) = najmanja vrednost od
- cena linka od A do B + cena najjeftinije rute od B do Z
- cena linka od A do C + cena najjeftinije rute od C do Z
- cena linka od A do D + cena najjeftinije rute od D do Z

Uskladu sa ovim principom, čvorAmože da utvrdi najjeftiniju rutu do Z sve dok Azna ceneveza do svih svojih suseda i ako svaki sused zna cenu najjeftinije rute do Z. Nakon toga, čvor A može da izvrši ranija izračunavanja i da utvrdi najjeftiniju rutu. Ali, kako ostali susedi znaju najjeftinije rute do Z? Odgovor obezbeduje način na koji algoritam funkcioniše. Postoji i centralizovana i distribuirana verzija algoritma. Pošto smo već obradili centralizovani algoritam, predstavićemo distribuiranu verziju.



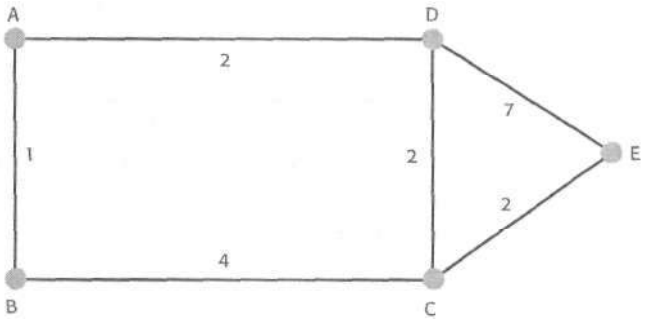
SLIKA 10.26 *Najjeftinija ruta od AdoZ*

Kao što je ranije pomenuto, svaki čvor zna samo cene do svojih suseda i sve informacije koje može da dobije od suseda. Dakle, kod distribuiranog algoritma svi čvorovi emituju ono što znaju o svojim susedima. Svaki čvor prima nove informacije i u skladu sa njima ažurira svoju tabelu rutiranja. Kako susedi nastavljaju periodično da emituju svoje informacije, informacije o svim rarežnim čvorovima i konekcijama na kraju prolaze kroz celu mrežu. Informacije koje dolaze u ćvorove omogučavaju im da otkriju nove čvorove i nove najjeftinije putanje do drugih čvorova.

Ilustracije radi, primenićemo algoritam na mrežu sa slike 10.27. Svaki čvor održava informacije o najjeftinijoj ruti do svakog drugog čvora. Sadrži cenu rute i prvi sledeći čvor na toj rutj. Inidjalno, svaki čvor zna cene do svojih suseda. U tabeli 10.5a prikazano je kako se smeštaju te informacije. Svaki red odgovara izvornom čvoru, a svaka kolona odredišnom čvoru. LI tabeli 10.5a prvi čvor na ruti uvek je isti kao i čvor u zaglavlju kolone, jer još uvek nisu otkrivene druge rute, osim onih ka susedima. Ova situacija će se promeniti kada se algoritam nastavi.

Imajte na umu da ova tabela navodi redove i kolone za sve čvorove i da svaki čvor zna samo šta se nalazi u redu koji mu odgovara. Na primer, A zna samo da su cene do B i D 1 i 2, respektivno; B zna samo da su cene do A i C 1 i 4, respektivno, i tako redom. Neki čvorovi i ne znaju da postoje neki drugi čvorovi. Na primer, A ne zna da postoje čvorovi C i E. B ne zna da postoje čvorovi DiEi tako dalje. To je u tabeli označeno kao *Nepoznato.*

Slika 10.28 sadrži algoritam u vidu pseudokoda.* Inicijalno, čvor u svoju tabelu rutiranja smešta informacije o rutama do svojih suseda. Prva while petlja ukazuje da algoritam nastavlja da prati informacije koje dolaze od suseda.



SLIKA 10.27 *Mreža za Bellman-Fordov algoritam*

* Ovaj algoritam nije kompletan, zato što postoje slučajevi u kojima će doći do greške. Ipak, naša namera je da ilustrujemo koncept učenja ruta unatraške kako bi se pronašle kraće putanje. Kasnije ćemo proučiti nedostatke ove verzije.

Tabela 10.5: Tri iteracije Bellman-Fordovog algoritma

	Odredište				
Izvor	**A**	**B**	**C**	**D**	**E**
			(a) Prva iteracija		
A		(B, 1)	Nepoznato	(D, 2)	Nepoznato
B	(A, 1)		(C, 4)	Nepoznato	Nepoznato
C	Nepoznato	(B, 4)		(D, 2)	(E, 2)
D	(A, 2)	Nepoznato	(C, 2)		(E, 7)
E	Nepoznato	Nepoznato	(C, 2)	(D, 7)	
			(b) Druga iteracija		
A		(B, 1)	(D, 4)	(D, 2)	(D, 9)
B	(A, 1)		(C, 4)	(A, 3)	(C, 6)
C	(D, 4)	(B, 4)		(D, 2)	(E, 2)
D	(A, 2)	(A, 3)	(C, 2)		(C, 4)
E	(D, 9)	(C, 6)	(C, 2)	(C, 4)	
			(c) Treća iteracija		
A		(B, 1)	(D, 4)	(D, 2)	(D, 6)
B	(A, 1)		(C 4)	(A, 3)	(C, 6)
C	(D, 4)	(B, 4)		(D, 2)	(E, 2)
D	(A, 2)	(A, 3)	(C, 2)		(C, 4)
E	(C, 6)	(C.6)	(C, 2)	(C, 4)	

```
Za svakog suseda umetnuti zapis (susedni cvor, cena linka) u tekucu
    tabelu rutiranja.
while mrezni protokoli me zavaravaju do
{
    prijem informacija iz tabele rutiranja cvora N;
    for each cvor Z in tabela rutiranja cvora N do
      if Z nije u tekucoj tabeli rutiranja
    umetnuti par (N, tekuca cena do N + cena rute od N do Z) u tabelu;
    else
      if tekuca cena do N + cena rute od N do Z < tekuca cena do Z
        zameniti tekucu cenu do Z tekucom cenom do N + cena rute od N do Z i
        naznaciti N kao novi prvi cvor duz rute do Z;
}
```

SLIKA 10.28 *Bellman-Fordov algoritam*

U okviru petlje svaki čvor prima informacije od svih svojih suseda. * Na osnovu njih, on uči kojim čvorovima mogu da pristupe njegovi susedi i kolike su cene dodeljene tim rutama. Za svaki čvor Z kome sused N ima pristup, postoje dve mogućnosti:

1. Tekući čvor nema nikakva ranija saznanja o čvoru Z. Tekući čvor umeće zapis (N, tekuća cena do N + cena rute od *N* do Z) u svoju tabelu rutiranja. Tekući čvor sada zna rutu do Z preko *N*.

2. Tekući čvor je već imao rutu do Z. Vrši se poređenje cene poznate rute sa cenom rute do N, plus cena od *N* do *Z*. Ako je druga vrednost manja, tekući čvor je pronašao jeftiniju rutu do Z. On menja svoju tekuću cenu za nižu i naznačava N kao prvi sledeći čvor duž rute do *Z*.

U tabeli 10.5b pokazano je kako se informacije svakog čvora menjaju dok mu njegovi susedi saopštavaju šta znaju. B govori A da ima pristup čvoru C sa cenom 4. Pošto A zna da ima pristup B po ceni 1, sada zaključuje da ima pristup u C sa cenom 5 (cena do B plus cena od B do C). Slično tome, D govori A da ima pristup do C, ali sa cenom 2. Sada A zaključuje da ima pristup u C i preko D, i to sa ukupnom cenom 4. Pošto je to jeftinije nego ruta preko B, on umeće zapis (D, 4) u svoju tabelu rutiranja (tabela 10.5b). D govori A i da ima pristup u E, sa cenom 7. Pošto A zna da ima pristup u D sa cenom 2, zaključuje da ima pristup u E sa cenom 7 + 2 = 9. Nakon toga, u tabelu rutiranja postavlja zapis (D, 9).

Sa naše tačke gledišta, mi znamo da postoji ruta od A do E preko D i C, čija je cena svega 6. Ali, zapamtite da smo mi u prednosti u poređenju sa A, jer možemo da sagledamo celu mrežu. Zbog svoje ograničene perspektive, A još uvek ne zna za tu rutu. Ne dozvolite da Vas naša perspektiva navede na pogrešno tumačenje algoritma.

Nastavljajući na ovaj način, B prima informacije od A i C o rutama do drugih čvorova.t Informacije iz čvora A ukazuju čvoru B da postoji aita do D sa cenom 3. Slično tome, informacije iz C pokazuju da postoji još jedna ruta do D sa cenom 6 i mta do E sa cenom 6. Biranjem najjeftinijih aita, drugi red tabele 10.5b prikazuje najnovije informacije o rutiranju za čvor B.

Nakon što svaki čvor po jednom dobije informacije od svojih suseda, tabela 10.5b prikazuje tabelu rutiranja svakog čvora. U ovoj tački svaki čvor zna najbolji način da dode do svih svojih suseda i svih njihovih suseda. Ipak, to još uvek nisu optimalne rute, jer možda postoje neke koje zahtevaju tri, ili više linkova.

* Iako ovaj algoritam odslikava uredenu sekvencu prijema od svih suseda, skoro je potpuno sigurno da se to neće desiti na ovakav način. Stvarni prijem informacija zavisi od mnogih događaja u realnom vremenu, koji su krajnje nepredvidivi. Naš glavni cilj je da opišemo kako čvorovi reaguju na dobijene informacije, tako da algoritam može da posluži za našu svrhu.

t Ono što B tačno prima od A zavisi od toga da li je A poslao informacije pre, ili nakon što je primio informacije od D. Pošto ovde ne možemo da garantujemo tajming, pretpostavljamo da svaki čvor šalje ono što je imao na početku svakog koraka.

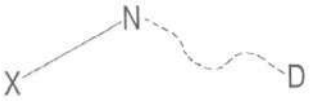
Dakle, svaki čvor prolazi kroz još jednu "rundu" prikupljanja informacija od svih svojih suseda i utvrduje da li postoje bolje rute. Na primer, A zna, na osnovu tabele 10.5b, da je najjeftinija ruta do E preko D sa ukupnom cenom 9. Medutim, kada A ponovo čuje D, saznaće da D može da stigne do E sa cenom 4. Pošto je cena linka od A do D 2, on sada zaključuje da može da dode do E preko D sa ukupnom cenom 6. Trebalo bi da ispratite sve korake algoritma sa slike 10.28 i da proverite da li su zapisi iz tabela 10.5b i 10.5c tačni.

Problemi sa Bellman-Fordovim algoritmom

Sve dok svaki čvor neprestano primenjuje algoritam, dolazi do smanjenja cena negde na mreži. Ipak, u zavisnosti od mrežne topologije, potrebno je određeno vreme da se reaguje na takve promene. Razlog je činjenica da, ako se smanji cena linka između dva čvora, jedino njihovi neposredni susedi znaju za to u okviru jednog prolaska kroz algoritam. Da bi za ovu promenu saznali i čvorovi koji su udaljeni dva linka, potrebna su dva prolaska algoritma, za čvorove udaljene tri linka potrebna su tri prolaska i tako redom. U opštem slučaju vreme potrebno da se prime vesti o promeni cena proporcionalno je broju međučvorova.

Ipak, postoji jedan ozbiljniji problem. Šta se dešava ako se cena linka izmedu dva čvora poveća? Hi, što je još gore, šta se dešava ako "otkaže" link između neka dva čvora? Slučaj "otkaza" može da se posmatra kao beskonačno povećanje cene. Pogledajmo primer. Red 2 u tabeli 10.5c ukazuje da je najjeftinija ruta od B do E preko C sa cenom 6. Ovo je delimično tačno, jer cena linka C-E iznosi 2. Šta će se dogoditi ako se poveća na 22? Algoritam, kao što je opisano, neće promeniti tabelu rutiranja za B, zato što vrši promene samo kada one omogućavaju kraću rutu.

Srećom, za ovaj problem postoji jednostavno rešenje, mada ono ima sporedne efekte u nekim situacijama. U opštem slučaju pretpostavimo da je X čvor i da je N prvi čvor na ruti koju X smatra najjeftinijom do čvora D.



Algoritam već obezbeduje slučaj kada *X* čuje od svojih suseda za jeftiniju rutu. Međutim, pretpostavimo da *X* čuje od N da je cena rute do D povećana. Pošto je *N* bio prvi sledeći čvor u ruti od X do D, X mora da ažurira svoju tabelu i da poveća cenu rute do D preko N. Nije teško modifikovati algoritam da se to izvede.

Na primer (ponovo se pozivamo na tabelu 10.5c), ako se link C-E poveća na 22 (povećanje za 20), C govori B da je došlo do povećanja. B reaguje tako što modifikuje svoju tabelu rutiranja, menjajući zapis koji odgovara ruti do E sa (C, 6) na (C, 26). Cvor B sada smatra da je najjeftinija ruta do E preko C sa cenom 26 (slične promene će se pojaviti i kod ostalih čvorova). Naravno, ovo nije tačno, ali nakon još par iteracija algoritma biće uspostavljena najjeftinija ruta preko linka D-E i u tabelama rutiranja će biti prikazano stvarno stanje cena linkova.

Ovo izgleda kao razumno rešenje, ali razmotrite sledeći scenario:

- Pretpostavite da se podskup sa slike 10.27 sastoji *samo* od linkova A-B, B-C i C-E.
- Nakon nekoliko iteracija algoritma, najjeftinija ruta od C do E ima cenu 2, od B do E 6 i od A do E 7.
- Dolazi do "otkaza" linka C-E (odnosno, cena linka C-E postaje oo).

Čvor C registruje "otkaz" i ažurira svoju tabelu mtiranja tako da naznači da cena dolaska do E iznosi oo. Osim toga, C prosleđuje informacije do B, koji menja svoju tabelu tako da naznači da ruta do E preko C ima cenu oo. Nažalost, B čuje od A da ruta do E ima cenu 7. * B nema pojma da se ruta iz A vraća u B. Pošto je cena linka B-A 1, B sada ažurira svoju tabelu tako da je najjeftinija ruta do E preko A sa cenom 8 (a da se dode do A i 7 za cenu od A do E).

Sada može da se desi sledeće:

1. A čuje od B da najjeftinija ruta do E ima cenu 8. Zato A ažurira svoju tabelu rutiranja tako da naznači da najjeftinija ruta do E ide preko B i da ima cenu 9 (1 da se dode do B i 8 za cenu od B do E).
2. B čuje od A da je došlo do povećanja cene njegove mte preko A na 10.
3. A reaguje i povećava cenu svoje rute preko B na 11.
4. Petlja se nastavlja u beskonačnost.

Kako A i B nastavljaju da razmenjuju informacije, tako se njihove cene neprestano povećavaju, a ni jedan njih ne shvata da je link ka E "otkazao". Iz naše ptičije perspektive na mrežu, ovo je sraešno. Medutim, svaki čvor zna samo ono što mu susedi saopšte, tako da su ovakve situacije moguće.

Ovo se naziva *problem brojanja u beskonačnost* (count-to-infinity problem) i može da se reši definisanjem nekog praga. Ako cena rute do nekog čvora prede vrednost praga, cena se postavlja na co, čime se označava da je čvor nedostupan. Problem je što vrednost praga mora da bude dovoljno velika da ne dode do mešanja sa legitimnim povećanjima cene. Zbog toga, algoritmu treba više vremena da ispravno reaguje na nastali problem.

Možda mislite da se ovaj problem mogao izbeći u potpunosti da A jednostavno nije slao informacije do B za rute koje prolaze kroz B. Naime, zašto bi A govorio B bilo šta o ruti do E koja prolazi kroz B? Nema smisla. Ovo nas dovodi do sledeće varijacije algoritma koja je poznata kao **podeljeni horizont (split horizon);** koristi se i termin *zatrovane rute (poisoned reverse)*. Reč je o jednostavnoj modifikaciji informacija koje se šalju do susednog čvora. U opštem slučaju pretpostavljamo da čvorX zna za rutu do *D* preko suseda *N*. Pravilo podeljenog horizonta kaže da *X* govori *N* da cena do D iznosi co. Na ovaj način, N neće pokušavati da ide do D preko X i da sledi rutu koja bi vodila nazad do njega.

* Ovo zavisi od tajminga. Ako A prvo čuje od B da je cena do E postala °o, onda će A videti cenu do E kao °o. Poenta je u tome da ne znamo šta će prvo da se desi i zato moramo da uzmemo u obzir sve mogućnosti.

Ako bi se ovo pravilo implementiralo u prethodnom primeru, A bi rekao B da je cena do E posta-Ia QO (jer ruta koju A koristi ide preko B). Pošto B već zna da je cena do E preko C postala oo, B će shvatiti da E nije dostupan. U sledećem koraku A shvata isto.

Iako ovo rešava ovde opisani problem, i dalje nema nikakvih garancija da će ovaj metod funkcionisati u svim slučajevima. Na primer, razmotrimo ponovo mrežu sa slike 10.27, ali pretpostavimo da nije postojao link D-E. Sve rute do E idu preko linka C-E i ta je činjenica prikazana u tabelama rutiranja. Osim toga,

- Ruta od D do E ide najpre do C.
- Ruta od B do E ide najpre do C.
- Ruta od A do E ide najpre do C.

Pretpostavimo da u određenom trenutku link C-E "otkaže". C će videti da je cena do E postala ». Osim toga, C neće pokušavati da ide preko B, ili D, jer mu oba kažu da je cena do E postala co (pravilo podeljenog horizonta). Nažalost, A i B mogu da razmene poruke (pravilo podeljenog horizonta ih ne sprečava da razmenjuju informacije), tako da A misli da se do E može stići preko B. B misli da se do E može stići preko A. Sada više ne postoji pravilo podeljenog horizonta koje bi zabranilo prenos informacija u C preko B. Zbog toga, B govori C za rutu do E. C sada misli da može da dode do E preko B.

Sada "stvari" postaju krajnje "zbrkane", jer isuviše toga zavisi od tajminga. Krajnji zaključak je da petlja u mreži izaziva da A, B, C i D razmenjuju poruke, čime se stvara utisak da postoje različite putanje do E, i otvorene i zatvorene. Postaju "senilne", neprestano menjajući odluku kuda treba da se stigne do E. Ova nestabilnost se nastavlja, što izaziva eskalaciju cena. Ako se koristi vrednost praga, petlja će se na kraju zaustaviti. Ipak, ovo će oduzeti neko vreme.

10.7 Dodatni metodi za rutiranje

Rutiranje na osnovu stanja linka

Problemi koje smo istakli pojavili su se zato što su čvorovi pokušavali da razmene informacije kako da se stigne do specifičnih odredišta. Dobra okolnost kod Dijkstrin algoritma je da svaki čvor izvršava algoritam lokalno kako bi doneo sopstvene odluke. Nedostatak je potreba čvora da ima informacije o mrežnoj topologiji i cenama linkova. Ako može da pribavi ove informacije, može da koristi Dijkstrin (ili bilo koji drugi) algoritam. Protokol za rutiranje na osnovu stanja linka je dizajniran da obavi upravo to.

Rutiranje na osnovu stanja linka je slično Bellman-Fordovom rutiranju sa jednog aspekta: svaki čvor prenosi do svojih suseda ono što zna. Razlika se ogleda u vrsti informacija koje se razmenjuju. Rutiranje na osnovu stanja linka je dizajnirano u skladu sa sledećim idejama:

- Cvor prikuplja informacije o statusu svakog linka od svih njegovih suseda. Na primer, medu značajne informacije mogu da se uvrste bitska brzina tog linka, kašnjenje prilikom slanja paketa preko tog linka, broj paketa koji se postavlja u red čekanja i pouzdanost tog linka. Sve ovo ima uticaja prilikom utvrdivanja cene tog linka.

- Čvor formira *paket stanja linka* (link state packet) za svaki link. Paket identifikuje dva čvora koja su povezana tim linkom i sadrži prikupljene informacije. Čvor šalje svaki paket do svakog suseda.
- Cvor koji primi paket stanja linka prosleđuje taj paket do svih svojih suseda (izuzimajući suseda od koga je paket stigao).
- Kako se paketi stanja linka razmenjuju izmedu čvorova, svaki čvor uči topologiju mreže i cene i status linkova izmedu mrežnih čvorova. Tako može da izvrši algoritam pronalaženja najjeftinije rute, kao što je Dijkstrin algoritam (koristeći sebe kao izvorni čvor), da bi utvrdio sopstvenu rutu do konkretnog odredišta (ili da bi bar utvrdio prvi sledeći čvor na toj ruti). Nakon toga, može da formira tabelu rutiranja u skladu sa prikupljenim informacijama.

Poput Bellman-Fordovog rutiranja, ovaj pristup može da iskoristi prednost i smanjenja i povećanja cena sve dok čvor periodično kreira i šalje pakete stanja linka, koji sadrže najnovije informacije.

Ipak, ovo ne znači da protokol nema problema. Na primer, ako mreža ima petlje, paketi stanja linka mogu beskonačno dugo da cirkulišu kroz mrežu, jer ih čvorovi neprestano razmenjuju. Ne samo da se na taj način povećava saobraćaj na mreži, već se javlja problem razlikovanja starog paketa stanja linka od onih novijih. Rešenje ovog problema podrazumeva instaliranje brojača u svaki paket. Brojač je pozitivan broj koji se postavlja prilikom kreiranja paketa. Uvek kada čvor prosledi paket do svog suseda brojač se dekrementira. Na kraju, brojač dolazi do vrednosti O i prijemni čvor ga odbacuje.

Sledeći problem je to što može da protekne odredeno vreme od trenutka kada dode do "otkaza" linka do trenutka kada udaljeni čvor sazna za to. U tom periodu paketi mogu neispravno da se rutiraju, što može da izazove kašnjenja (ili, čak,"otkaze") paketa koji stižu u njihova odredišta.

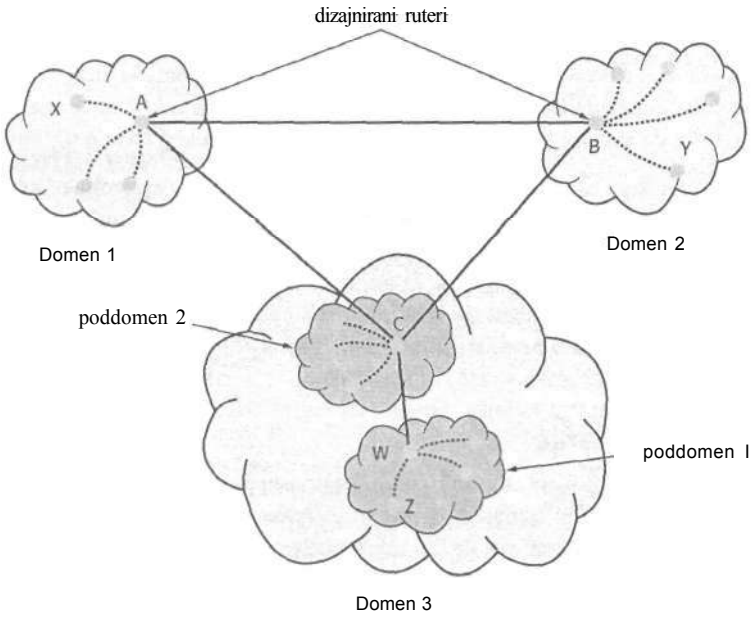
Rutiranje u dinamičkom okruženju može da bude problematično. Nepovoljne "stvari" koje smo prethodno opisali verovatno neće da se dese, ali loš tajming može da ih inicira. Zato su pristupi koje koristite dobri većim delom vremena, ali povremeno mogu da budu problematični. Svako ko je nekada debagirao neki program zna za najteže i najnezgodnije probleme na koje može da naide. U stvari, kasnije ćemo videti da kod Internet rutiranja ne postoje garancije za isporuku paketa. Ako izgleda da paketi ne napreduju, Internet Protocol ih jednostavno izbacuje. Internet Protocol ćemo prikazati Poglavlju 11.

Hijerarhijsko rutiranje

Svi do sada razmotreni pristupi rutiranju imaju nešto zajedničko - dizajnirani su tako da svaki ćvor dobije odgovarajuće informacije o rutiranju. Ipak, ponekad postoji isuviše veliki broj čvorova da bi se svima efikasno prosledile informacije o rutiranju. Tretiranje svih čvorova kao ravnopravnih učesnika u velikim mrežama dovodi do generisanja isuviše velike količine informacija koje treba poslati preko mreže. Alternativa je da postoje neki čvorovi koji će obaviti rutiranje za ostale. Uobičajeni pristup je hijerarhijsko rutiranje. Ima sledeće karakteristike:

- Svi čvorovi su podeljeni u grupe koje su nazvane **domeni.** Domen možemo da smatramo zasebnom i nezavisnom mrežom koju održavaju neka kompanija, ili organizacija. Koristi se i termin **autonomni sistem (AS).**
- Rute između dva čvora u zajedničkom domenu određene su pomoću protokola domena, ili mrežnih protokola.
- Svaki domen ima jedan, ili više specijalno dizajniranih rutera koji utvrduju rute između domena. Efektivno, ti ruteri formiraju mrežu.
- Ako je domen veliki, može da sadrži više poddomena, od kojih svaki može da sadrži sopstveni naznačeni ruter. Ovi ruteri određuju rute izmedu poddomena u samom domenu.

Pretpostavimo da čvor X treba da pošalje paket čvora Y. Ako se nalaze na istom domenu, rute mogu da se utvrde pomoću prethodno predstavljenih tehnika. Sa druge strane, pretpostavimo da se nalaze u različitim domenima (slika 10.29). Čvor X šalje paket do rutera A u okviru njegovog domena. Cvor A je odgovoran za utvrdivanje najbolje rute do domena čvora Y (domen 2) i slanje paketa. Pošto je čvor B izabrani ruter za domen 2, on prima paket, a zatim ga šalje do čvora Y. Ovaj pristup se primenjuje za svaki par čvorova sa domena 1 i 2. A izvršava neophodno rutiranje u ime bilo kog čvora na svom domenu, čime se redukuje ukupan broj čvorova koji bi morali da izvrše takve zadatke. Cvor X i drugi čvorovi na domenu I treba da vode računa samo o tome kako da dodu do A.



SLIKA 10.29 *Domeni u hijerarhijskom rutiranju*

Koncept domena možemo da se predstavi pomoću hijerarhijske strukture (slika 10.30). Svi domeni odgovaraju čvorovima na drugom nivou stabla ispod zajedničkog korena.* Svi mrežni čvorovi u okviru domena odgovaraju čvorovima trećeg nivoa na domenu. Ako domen sadrži poddomene, a oni imaju čvorove trećeg nivoa u stablu, i svi mrežni čvorovi u njima su na četvrtom nivou u okviru odgovarajućeg poddomena.

U opštem slučaju mrežni ruteri su delimično definisani mestom u hijerarhiji. Pretpostavimo da Z sa slike 10.29 treba da pošalje paket do X. Pošto je Z u poddomenu 1, šalje paket do W, izabranog rutera za poddomen čvora Z. W rutira paket preko poddomena do C, izabranog rutera za domen 3. Nakon toga, C rutira paket preko domena do A, koji konačno šalje paket do X.

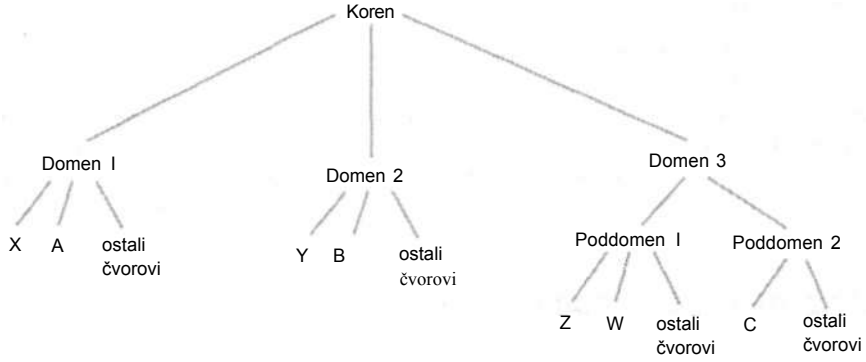
Do sada smo izostavljali jedan detalj. Da bi hijerarhijsko rutiranje funkcionisalo, čvor koji šalje pakete defmiše odredišnu adresu, uključujucl svoj domen i sve njegove poddomene. Na osnovu uključene adrese, svaki ruter utvrduje da li se odredište nalazi na tekućem domenu (ili poddomenu). Ako se nalazi, ruter može da ispomči paket. Ako nije na tom domenu, ruter mora da utvrdi domen na koji treba poslati paket. Ovo je slično slanju pisama preko regulame poštanske službe sa tipičnim formatom adresiranja.

Ime

Ulica i broj

Grad, država, poštanski broj

Radnici u poštama u Hardfordu, Connecticut ne moraju da brinu da li je pismo adresirano na Jane Smith koja živi u 123 Main Street, osim ako kao grad i država nije naveden Hardford, Connecticut.



SLIKA 10.30 *Hijerarhijsko uređenje čvorova na domenu*

′ Ne smatrajte koren slvarnim mrežnim čvorom. On jednostavno označava da su svi njegovi zavisnici (domeni) povezani.

Ako se grad i država razlikuju, poštanski službenici u Hartfordu brinu samo o tome da se pismo pošalje do odgovarajućeg grada, a u tom gradu poštanski službenici vode računa o konkretnoj adresi i imenu osobe na koju je pismo adresirano. Slično tome, mrežni čvor naznačava adresu kao sekvencu identifikatora domena i poddomena. Na primer, čvor X sa slike 10.29 može da pošalje paket do čvora Z tako što navodi adresu Z.poddomen-1.domen-3.

Internet je mreža koja koristi hijerarhije u okviru svoje šeme adresiranja. Hijerarhija reflektuje činjenicu da je Internet, u stvari, kolekcija mreža, koje imaju sopstvene protokole. Univerziteti, kompanije, ili vladine agencije obično imaju mrežu u okviru koje su povezani svi njihovi kompjuteri. Te mreže takode mogu da budu deo Interneta.

Internet adresa je 32-bitni broj* predstavljen kao sekvenca četiri 8-bitna broja koja su razdvojena tačkama. Na primer, adresa može da bude 143.200.128.3, gde svaka četvorka ima 8-bitnu reprezentaciju. Svaka Internet adresa može da se interpretira sa dva dela: Internet Protocol (IP) mrežna adresa pridružena tom mestu na mreži i adresa lokalnog uređaja (tj. personalnog kompjutera, ili servera) na toj mreži. Prikazani primer adrese pripada Klasi B,t što znači da prvih 16 bitova (143.200) označava IP mrežnu adresu, a drugih 16 bitova (128.3) označavaju specifični uredaj.

Za vreme rutiranja paketa, ruter najpre proučava IP mrežnu adresu. Ako je paket namenjen nekom drugom mestu, njegovo rutiranje je zasnovano isključivo na toj IP mrežnoj adresi. U suprotnom, ruter proučava lokalni deo adrese i izvlači, ako je potrebno, identifikator fizičke adrese. Nakon toga, on postavlja paket u red čekanja za isporučivanje do odredišta preko odgovarajuće fizičke mreže. Odatle protokoli nižeg sloja mreže upravljaju isporukom u skladu sa tipom mreže. Ovaj proces predstavlja, u stvari, tehniku hijerarhijskog rutiranja na dva nivoa. Postoji par "stvari" koje su neophodne za dalji opis; bavićemo se njima u sledećem poglavlju kada budemo predstavljali Internet.

Nekim sajtovima može da bude dodeljena jedna IP mrežna adresa, a da, u stvari, imaju više fizičkih mreža na tom mestu koje su označene kao **podmreže (subnets).** Jedna IP mrežna adresa omogućava proširivanje sajta i razvoj mrežnih aplikacija nezavisno od konekcije sa Internetom. Na primer, lokalna uprava može da koristi deo 16-bitnog lokalnog ID-a (možda nekoliko prvih bitova) kako bi bila označena odredena podmreža. Tako se kreira hijerarhija adresa na tri nivoa, gde prva dva okteta označavaju IP mrežu, sledećih par bitova označavaju podmrežu, a preostali bitovi ukazuju na konkretno odredište na toj podmreži.

Routing Information protokol (RIP)

Kao što smo već istakli, velike mreže, kao što je Internet, u opštem slučaju mogu da se posmatraju kao kolekcije domena. Neki više vole termin *autonomni sistem (AS),* umesto termina *domen,* da bi se ukazalo na činjenicu da mogu nezavisno da funkcionišu.

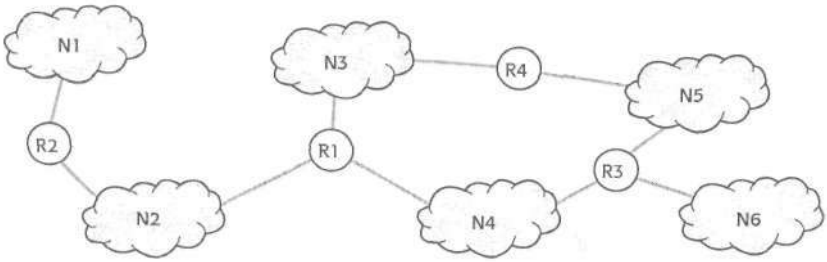
* Neke adrese imaju više bitova, ali njih obrađujemo u sledećem poglavlju.

t Internet Protocol defmiše i adrese Klase A (8-bitna mrežna adresa i 24-bitni lokalni identifikator) i Klase C (24-bitna mrežna adresa i 8-bitni lokalni identifikator). Postoje i klase D i E adrese i grupe besklasnih adresa, ali njih cemo prikazati u odeljku 11.2.

U takvim mrežama možemo da definišemo dve glavne kategorije strategija za rutiranje: unutrašnje i spoljašnje protokole.* **Protokoli za unutrašnje rutiranje (exterior routing protocols)** kontrolišu rutiranje između rutera u okviru autonomnog sistema. **Protokoli za spoljašnje rutiranje (interior routing protocols)** kontrolišu rutiranje izmedu rutera različitih autonomnih sistema. Na primer, unutrašnji protokol se koristi za utvrđivanje rute od X do A (slika 10.29). Eksterni protokoli se koriste da bi se došlo od A do B, ili C.

Ovde je bitno istaći razliku, jer različiti AS-ovi mogu da koriste različite metrike za izračunavanje najbolje rute u okviru AS-a. Neki mogu da zasnivaju svoja izračunavanja na osnovu bitskih brzina, a drugi mogu u potpunosti da se oslone na broj rutera koji moraju da proslede paket. Broj koji se koristi za predstavljanje cene može da se izračuna na različite načine. Ruta sa cenom 100 u jednoj mreži može da bude sporija od rute sa cenom 200 u nekoj drugoj mreži. Zato pronalaženje "najbolje" mte može da bude teško, jer se metrike koje su korišćene za merenje najbolje rute možda ne mogu primeniti na sve sisteme. Na primer, možda će biti teško definisati najbolju rutu od X do Z na slici 10.29, jer se definicija onoga što je "najbolje" može razlikovati od domena do domena. Uskoro ćemo videti da ovo može da utiče na izbor protokola koji će biti korišćeni.

Jedan primer protokola za unutrašnje rutiranje koji je dugo korišćen na Internet ruterima u okviru AS-a je **Routing Information Protocol (RIP).** RIP je protokol koga je koristio rutiranit program razvijen u University of California, Berkeley, koji je bio zadužen za rutiranje u lokalnim mrežama. Ruteri koji povezuju više mreža koriste RIP kako bi svi znali najkraće rute do specifične mreže. Obično koriste **brojače skokova (hop count),** broj medurutera i rastojanje. Na slici 10.31 prikazano je nekoliko mreža koje su povezane ruterima. Broj "skokova" od mreže Nl do N2 je 1, a od Nl do N4 je 2, kada se koriste najkraće rute. Ipak, to još uvek ne zna ni jedan ruter. Algoritam se startuje kada svaki ruter pošalje poruku do svih svojih mreža. Ova poruka ukazuje da sve mreže na koje je ruter povezan mogu da se dostignu u jednom "skoku". Kada ruteri na tim mrežama dobiju poruku, oni znaju koje se mreže mogu dostići sa dva "skoka".



SLIKA 10.31 *Mreže povezane ruterima*

* Za ovo se koriste razlidti nazivi u odnosu na to koga slušate. Na primer, unutrašnji protokoli mogu da se nazivaju i protokoli unutrašnjeg gatewaya (interior gateway protocols), protokoli unutrašnjih rutera (interior router protocols), ili protokoli unutar domena (intradomain protocols). Spoljašnji protokoli mogu da se nazivaju i protokoli spoljašnjeg gatewaya (exterior gateway protocols), protokoli spoljašnjih rutera (eXterior router protocols), ili protokoli između domena (interdomain protocols).

t U okviru engleskog izgovora rutirani programi se izgovaraju kao rut-di (route-dee) i razlog za to su bik UNIX-OVe konvencije imenovanja.

Ove informacije se smeštaju u tabele rutiranja i periodično se emituju preko mreža. Periodičnim primanjem, smeštanjem i emitovanjem informacija svaki ruter zna za najmanji broj "skokova" na konkretnoj mreži.

Pogledajmo kako ovo funkcioniše na mreži sa slike 10.31. Korišćenjem RIP protokola moguće je da se desi sledeće:

1. R2 šalje poruku duž N2 koja može da stigne do Nl u jednom "skoku" (takode šalje poruku i duž Nl koja može da stigne do N2 u jednom "skoku").
2. Pošto je Rl povezan na N2, on zna da može da stigne do Nl u dva "skoka" i beleži tu činjenicu u svoju tabelu mtiranja. Zbog toga, Rl emituje sledeće preko N4: Može da stigne do N2 i N3 u jednom "skoku", i do Nl u dva "skoka" (slične informacije emituje preko N3 i N2).
3. R3 prima i smešta informacije primljene preko N4 i emituje sledeće preko N5: Može da stigne do N4 i N6 u jednom "skoku", do N2 i N3 u dva "skoka" i do Nl u tri "skoka". Slične informacije emituje preko N4 i N6.

U ovoj tački neke od emitovanih informacija postaju redundantne. Na primer, pošto R3 emituje informacije preko N5, R4 uči da može da stigne do N2 u tri "skoka". Naravno, ako je prethodno primio poruku od Rl preko N3, on već zna da do N2 može da stigne u dva "skoka". U tom slučaju neće beležiti najskorije primljene informacije.

Sta se dešava ako ruter, ili mreža na odredenoj ruti "otkažu"? Na primer, R4 zna da može da stigne do N2 u dva "skoka", ali šta ako mreža N3 "otkaže"? Kada ruter zapamti informacije o rutiranju, on, ujedno, startuje i tajmer. Kada tajmer istekne, ruter označava rutu kao nevalidnu. Ponovno uspostavljanje rute zavisi od novih informacija o rutiranju. Naravno, ruteri moraju da sarađuju priliko slanja informacija o rutiranju na regularnoj osnovi (obično na svakih 30 sekundi). Ako dode do "otkaza" na N3, dogadaji 1 i 3 i dalje mogu da se dese (osim onih delova koji uključuju N3). Ovoga puta R4 saznaje da može da stigne do N2 u tri "skoka" i beleži tu činjenicu, jer nema drugu alternativu.

RIP funkcioniše razmenom *RIP paketa.* Svaki paket predstavlja zahtev susedu za informacijama, ili sadrži broj "skokova" do čvora. Svaki paket sadrži sledeća polja:

- Polje Command dužine 1 bajt, koje ukazuje da li paket predstavlja zahtev, ili odziv. Paket Request traži informacije iz susedove tabele rutiranja. Paket Response ukazuje da se u paketu nalaze informacije o rutiranju.
- Broj RIP verzije dužine 1 bajt
- Polje Address dužine 4 bajta, koje sadrži adresu odredišnog čvora koji se oglašava
- Polje Address Family Indicator dužine 2 bajta, koje definiše adresu sa kojom protokol radi - na primer, da je reč o IP adresi, ili adresi koju koristi neki drugi protokol
- Polje Hop Count dužine 4 bajta. Ova vrednost je uvek između 1 i 16. RIP pretpostavlja da do svakog čvora treba najviše 15 "skokova". Ako čvor ne može da se dostigne u 15 "skokova", u paketu stoji 16, što je za RIP ekvivalentno beskonačnosti.
- Nekoliko polja koja se ne koriste (rezervisana su za neke buduće verzije)

Sledeća verzija ovog protokola RIP verzija 2 (RIP-2) funkcioniše uglavnom na isti način kao i RIP (verzija 1), ali ima neke dodatne moguchosti, prvenstveno za uzimanje u obzir različitih AS sistema. Koristi isti format paketa kao i RIP, osim što se u ranije neiskorišćenim poljima sada prenose neke korisne informacije. RIP-2 paketi sadrže sledeća polja:

- Sva naznačena polja iz ranije verzije RlP-a, Naravno, ovoga puta je broj verzije 2. Address Family Indicator može da bude specijalna vrednost sa svim jedinicama, što znači da se paket koristi za autentifikaciju. U ovom slučaju ostatak paketa sadrži informacije o autentifikaciji, iako je, u vreme pisanja ove knjige, jedina opcija bila lozinka autentičnosti.
- Maska podmreže dužine 4 bajta za naznačenu adresu. Ranije smo istakli da lokalni identifikator IP adrese može da sadrži ID podmreže i drugi ID koji je lokalni za tu podmrežu. Maska podmreže pomaže utvrđivanje bitova u adresi koji se koriste za naznačavanje ID-a podmreže. Podmreže i maske podmreža dalje obrađujemo u Poglavlju 11, ali u osnovi, ruter mora da zna masku podmreže kako bi utvrdio rutu.
- Polje Route Tag dužine 2 bajta. RIP ne prepoznaje postojanje zasebnih AS sistema. RIP-2 pokušava da adresira ovaj nedostatak korišćenjem polja Route Tag koje ukazuje na razliku izmedu unutrašnjih i spoljašnjih ruta. Ne defmiše se sadržaj polja, ali pošiljalac paketa može da ukaže na tip rute koja se naznačava. Prijemni ruter tada može da reaguje na odgovarajući način. Na primer, ako vrednost 0 označava unutrašnji protokol sa brojanjem skokova, paket može da se razmeni sa drugim ruterima u AS-u, sa inkrementiranjem broja "skokova" na prethodno opisani način. Nenulta vrednost može da ukaže na neki spoljašnji protokol, a ruteri moraju da dele te informacije onakve kakve jesu. Naravno, ruteri moraju da podrže spoljašnji protokol koji se oglašava.
- Polje Next Hop dužine 4 bajta. RIP (verzija 1) paket oglašava domen i pošiljalac paketa može da koristi ovo polje za sledeći "skok" do njega. Polje Next Hop eksplicitno defmiše adresu sledećeg čvora (koja se može razlikovati od pošiljaoca paketa) duž te mte. Na primer, pretpostavimo da su A, B i C ruteri i da svi znaju da najbolja ruta do mreže X ide preko B i da najbolja ruta ide do mreže Y preko C. Da bi četvrti ruter D pronašao najbolje rute do X i Y pomoću RlP-a, on zavisi od izvršenja protokola u A, B i C. Ako se koristi RIP-2, samo A treba da pošalje RIP-2 pakete do D, ukazujud na odgovarajući sledeći "skok" do mreža X i Y.

Algoritam Open Shortest Path First

Sleded primer šeme unutrašnjeg rutiranja je **Open Shortest Path First (OSPF).** Oznaka otvorenosti u nazivu ukazuje da algoritam nije vlasnički, tj. da ga ne poseduje ni jedna kompanija. U suštini, OSPF je oblik mtiranja na osnovu stanja linka sa nekoliko dodatnih polja:

- OSPF uključuje mogućnost autentifikacije poaika.
- Obezbeduje dodatne hijerarhije. Kao što je ranije opisano, domen može da sadrži više poddomena.

- Za dolazak do željenog odredišta može da se koristi više ruta. Ovo je posebno korisno za rute koje se često koriste. Slanje svih paketa preko iste rute, čak i ako je "najbolja", može da dovede do zagušenja. Korišćenje rezervnih ruta može da poboljša ukupne performanse. Ovo se naziva *balansiranje opterećenja,* a pomalo podseća na upozorenja na velike zastoje u saobraćaju na medunarodnim autoputevima (možete da se odlučite za putovanje nekim manje frekventnim saobraćajnicama).
- Za definisanje stanja linkova može da se koristi nekoliko faktora. Faktori mogu da budu dužina linka, bitska brzina, kašnjenje i cena u dolarima.
- Postoje mogućnosti za bolje reagovanje na potrebe korisnika. Na primer, korisnik možda želi da brže šalje manje poruke. Pošto je reč o malom paketu, visoka bitska brzina nije od ključnog značaja, ali je bitno kašnjenje koje postoji na linkovima kojima paket putuje. Sa druge strane, kada korisnik želi da prenese veliki fajl, najbolji transfer može da se ostvari preko linkova koji omogućavaju prenos na veoma velikim bitskim brzinama.

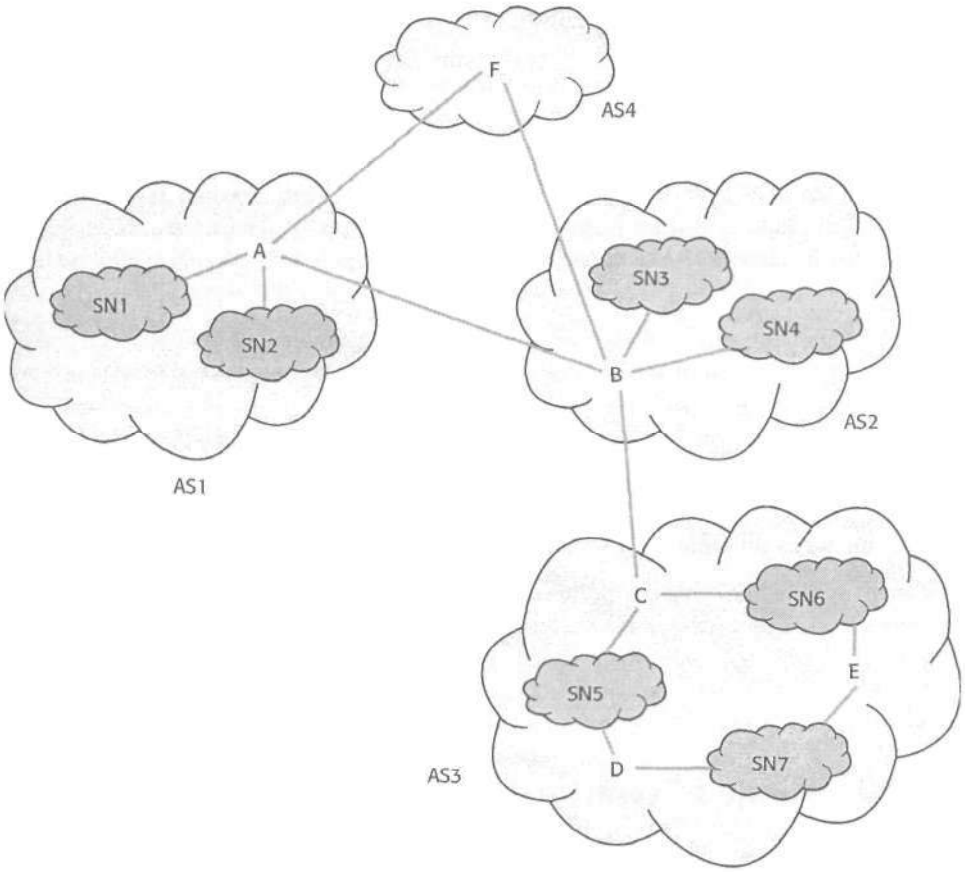
Border Gateway protokol

Border Gateway Protocol (BGP) je primer spoljašnjeg protokola. Tekuća verzija je BGP-4 i obično se koristi na Internetu za uspostavljanje putanja između rutera koji se nalaze u različitim autonomnim sistemima. Ima neke zajedničke karakteristike sa rutiranjem zasnovanom na vektorskom rastojanju. Glavna razlika se ogleda u tome što se, umesto da se razmenjuju cene rute do odredenog odredišta, razmenjuju informacije o konkretnoj ruti, definisane kao sekvenca uključenih AS sistema. Postoji nekoliko razloga za navodenje konkretnih ruta umesto samo prvog sledećeg linka na najjeftinijoj ruti. Jedan je taj što protokoli kao što je BGP-4 često nastoje samo da dodu do odredišta, za razliku od protokola koji traže najbolju, ili najjeftiniju rutu. Pošto rute neizbežno prolaze preko jednog, ili više AS-a, a svaki AS može da definiše cene linkova na različite načine, poredenje cena ruta u različitim sistemima može da bude besmisleno. Zbog toga, koncept "najbolje rute" ponekad nema značenja kod spoljašnjih protokola.

Sledeći razlog za razmenu konkretnih ruta je to što neki drugi faktori, osim cena, mogu da budu uključeni u pronalaženje rute. Svaki ruter može da implementira specifične polise, ili ograničenja koja svaka ruta mora da ispuni. Ovo je posebno značajno kod globalnih konekcija, kao što su one koje se koriste za povezivanje na Internet. Na primer, ruter u nekoj zemlji može da ima eksplicitnu polisu o izbegavanju svih ruta koje prolaze kroz neprijateljske zemlje. Slično tome, ruter u kompaniji može da ima polisu da se izbegavaju sve rute koje prolaze kroz resurse konkurenta.

Na slici 10.32 ilustrovan je ovaj proces. AS3 je AS sistem koji se sastoji od tri podmreže (SN5, SN6 i SN7) i tri rutera (C, D i E). U okviru AS3 ruteri mogu da koriste unutrašnji protokol, kao što je RIP. Međutim, C izvršava i BGP-4, koji mu omogućava komunikaciju sa B iz AS2. Ruter C uspostavlja TCP konekciju (biće prikazana u Poglavlju 11) sa B i počinje slanje Open poruke do B. Tako C može da se identifikuje kao BGP-4 ruter. Ruter B može da odgovori sa Keep Alive porukom koja potvrduje prijem poruke iz C. Na osnovu te poruke, C može da zaključi da B pristaje da učestvuje u razmeni informacija o rutama.

Da bi se ažurirale informacije o ruti, ruter šalje Update poruku. Ova poruka može da obavesti suseda o novim rutama, ili da ukaže na činjenicu da neke rute više nisu validne.



SLIKA 10.32 *Autonomni sistemi sa podmrežama*

U tom slučaju C šalje Update poruku do B, u kojoj naznačava SN5, SN6 i SN7 kao odredišne mreže, a AS3 kao jedini AS u okviru te rute. Takode naznačava adresu rutera C kao način za dolazak do AS3. Kada B dobije te informacije, može da ih podeli sa A. On može da pošalje Update poruku do A, u kojoj ukazuje da su SN5, SN6 i SN7 odredišne mreže, a AS2 i AS3 autonomni sistemi preko kojih mora da se prode da bi se stiglo do tih podmreža. Ako je A povezan na neki drugi AS, može da nastavi proces slanja; sa svakom novom Update porukom povećava se niz uključenih AS sistema. Kada ruter dobije Update poruku, može tačno da utvrdi preko kojih autonomnih sistema informacije mogu da putuju.

U ovom primeru je pretpostavljeno da Update poruka automatski ažurira sve tabele rutiranja. To ne mora da bude tako. Na primer, pretpostavimo da ruter na drugoj strani ASl dobija Update poruku koja naznačava AS2 kao jedan od uključenih autonomnih sistema. Takode pretpostavimo da polisa višeg nivoa na toj lokaciji brani slanje bilo kakvih informacija preko putanje koja uključuje AS2. U tom slučaju ruter ne prihvata oglašenu rutu kao validnu i koristiće ili ono što ima od ranije, ili će čekati Update poruku u kojoj se naznačava neki drugi niz autonomnih sistema.

Prednost ovog pristupa je što se petlje na putanjama lako eliminišu. Na primer, B na slici 10.32 može da dobije Update poruku od F sa naznačenim odredištima SN5, SN6 i SN7 i autonomnim sistemima AS4, ASl, AS2 i AS3. Pošto je B deo AS2, a AS2 je već naznačen u sekvenci, B prepoznaje da ruta sadrži petlju i može da je ignoriše.

Sledeća mogućnost se odnosi na situaciju u kojoj naznačena ruta više nije validna. Na primer, ruter A zna da može da dođe do SN5 preko AS2 i AS3. Međutim, pretpostavimo da je link između B i C "otkazao". Ruter B šalje još jednu Update poruku do A, u kojoj naznačava da putanje do SN5, SN6 i SN7 preko AS2 i AS3 više nisu validne. Nakon toga, ruter A uklanja te informacije iz svojih tabela.

U okviru zaključka trebalo bi da napomenemo da svi ruteri ne moraju da donose ovako složene odluke. Na primer, AS3 sa slike 10.32 može da se nalazi na geografskoj lokaciji koja omogućava pristup Internetu samo preko AS2. U tom slučaju C koristi statičko rutiranje, tehniku koja podrazumeva slanje svih odlazećih paketa na isto mesto, na ruter B. Tek od rutera B počinje usmeravanje na različite rute. Naravno, dolazeći paketi se drugačije rutiraju ka C, u skladu sa korišćenim unutrašnjim protokolima.

Rezime tehnika za rutiranje

U tabeli 10.6 dat je kraći pregled opštih strategija rutiranja koje su prikazane u ovom poglavlju, Ovo definitivno nisu jedine mogućnosti, ali, zajedno sa strategijama rutiranja koje uključuju LAN mostove i komutatore, predstavljaju značajan broj strategija koje se primenjuju u današnje vreme. Za više informacija o različitim strategijama pogledajte referencu [Ta03]. Osim toga, u referencama [Co99] i [CoOO] posvećeno je dosta pažnje rutiranju na Internetu.

10.8 Zagušenje i "samrtni zagrljaj"

Zagušenje

Kako mreže rastu i obuhvataju više čvorova, strategije rutiranja moraju da kontrolišu sve veći broj paketa. Sve veći zahtevi mogu da ugroze mrežne operacije. Sta se dešava kada dođe do "otkaza" na jednom, ili na više mrežnih linkova? Sta se sešava ako broj paketa koji se mora isporučiti premaši mogućnosti mreže za isporuku? Potencijalno opasna posledica takvih događaja je zagušenje, ili preterano nagomilavanje paketa u jednom, ili više mrežnih čvorova.

Ponovo možemo da napravimo analogiju sa kontrolom saobraćaja u urbanim područjima. Autoputevi i gradske saobraćajnice moraju da se dizajniraju tako da mogu da prihvate predvidenu količinu saobraćaja. Svako ko se vozio kroz urbana područja zna u čemu je problem. Za vreme špica saobraćaj je veoma gust i tada se mogudiosti za kontrolu saobraćaja umanjuju. Nesreće, ili radovi na putu mogu da isključe nekoliko traka, ili da u potpunosti "odseku" saobraćaj preko te deonice. U talcvim slučajevima se saobraćaj usporava (ili potpunosti zaustavlja) i nastaju strahovita zagušenja. Ljudi koji se nalaze u svojim automobilima ne mogu da stignu do željenih odredišta. Transportni sistem privremeno gubi svoju upotrebljivost.

Tabela 10.6: Rezime strategija rutiranja

Metod	Komentari
Dijkstrin algoritam	Algoritam učenja unapred koji može da se implementira kao strategija centralnog rutiranja. Može da se koristi i sa rutiranjem zasnovanim na stanju linka.
Bellman-Fordov algoritam	Algoritam učenja unazad. Čvorovi uče najjeftinije rute do čvora od svojih suseda, zajedno sa prvim čvorom na toj ruti. Koristi se na Internetu, ili na svakoj velikoj mreži na kojoj promene uslova na linkovima zahtevaju ažuriranje tabela rutiranja u čvorovima.
Rutiranje na osnovu stanja linka	Čvorovi sarađuju, razmenjujući pakete stanja linka u kojima se nalaze statusne informacije o susednim linkovima. Čvor može da sakupi sve pakete koje primi i utvrđuje mrežnu topologiju. Nakon toga, može da izvrši svoj algoritam za pronalaženje najkraće rute.
Hijerarhijsko rutiranje	Metod deljenja čvorova na domene, ili autonomne sisteme. Unutrašnji protokoli su zaduženi za rutiranje u okviru sistema, a spoljašnji za pronalaženje ruta izmedu različitih sistema. Koristi se na Internetu i svim mrežama sa većim brojem čvorova i nije praktičan izbor u situacijama kada svi koriste istu strategiju rutiranja.
RIP	Protokol za unutrašnje rutiranje koji se koristi sa manjim brojem "skokova" unutar konkretne mreže. Razvijen je na UC-Berkeley za potrebe lokalne mreže i korišćen je na Internetu.
OSPF	Protokol za unutrašnje rutiranje sličan rutiranju zasnovanom na stanju linka, ali obezbeduje dodatne karakteristike koje omogućavaju bolje performanse i veću fleksibilnost
BGP	Protokol za spoljašnje rutiranje koji ruterima omogućava implementiranje specifičnih polisa, ili ograničenja koja rute moraju da ispoštuju. Ruteri razmenjuju informacije o konkretnim rutama do odredišta, umesto da se razmenjuju samo cene i prvi link na konkretnoj ruti.

Slično tome, zagušenje redukuje korist od mreže. Paketi više kasne, a korisnici mreže se suočavaju sa slabim odzivom i nemogućnošću za ispunjavanje njihovih zahteva. Šta mrežni protokoli mogu da urade u takvim situacijama? Jedna opcija podrazumeva da se ne preduzima ništa, već da se sačeka da zagušenje prirodno nestane. Cak i u vreme saobraćajnog špica ljudi stižu kući i zagušenje prestaje. Međutim, ovo nije praktično rešenje kada je reč o mrežama (mnogi će se složiti da ovo nije praktično rešenje ni kada je reč o putevima). Kao prvo, korisnici očekuju bolju uslugu i trebalo bi da je dobiju. Drugo, zagušenja možda neće prestati zbog efekta nagomilavanja. Zbog zagušenja, čvor ne može da prima pakete od ostalih čvorova. Zato ti čvorovi ne mogu brzo da se reše svojih paketa, a dolazeđ paketi se brzo akumuliraju. Ovo može da ima lančani efekat u okviru koga će se zagušenje javiti na svim čvorovima, tako da se problem samo dalje pogoršava.

Postoji nekoliko načina za rešavanje zagušenja.

- **Eliminisanje** paketa Ako se u čvoru nagomila veliki broj paketa, neki od njih se eliminišu. Tako se redukuje broj nerešenih paketa koji čekaju na prenos i redukuje se opterećenje mreže. Naravno, nedostatak je to što uništeni paketi nikada neće stići do svojih odredišta (problem izgubljenih paketa smo razmatrali u odeljcima 8.4 i 8.5). Moguće je da će na kraju protokol koji se izvršava u čvoru pošiljaocu utvrditi da paket nikada nije stigao do svog odredišta i iniciraće njegovo ponovno slanje. Ako je zagušenje izazvano naglim povećanjem saobraćaja, možda će opasti kada paket bude slat sledećeg puta. Uništavanje paketa deluje kao drastično rešenje, ali, ako su zagušenja sporadična, mrežni protokoli mogu dobro da ih kontrolišu i neprijatnosti kojima će korisnik biti izložen zbog uništavanja paketa su svedene na minimum.

- **Kontrola toka** Kao što je istaknuto u Poglavlju 8, protokoli za kontrolu toka su dizajnirani tako da kontrolišu broj poslatih paketa. Ipak, oni ne mogu da se koriste kao prava rešenja za kontrolu zagušenja na mreži. Problem je što kontrola toka ograničava broj poslatih paketa između dve tačke, a zagušenja obično uključuju pakete koji dolaze iz različitih izvora. Tako je moguće da čak i čvorovi koji regulišu broj poslatih paketa i dalje budu predmet zagušenja ako isuviše veliki broj čvorova pošalje pakete.

  Možda ćete predložiti da svaki čvor reguliše svoj saobraćaj - ako svaki čvor šalje pakete, ukupan broj paketa i dalje može da se kontroliše. Problem je što, ako veliki broj čvorova ništa ne šalje, mreža nije dovoljno iskorišćena. Možda ćete predložiti da svaki čvor redukuje broj odlazećih paketa ako detektuje da drugi čvorovi više šalju. Međutim, na velikim mrežama ovo nije praktično. Mnogi čvorovi i ne vide da ostali šalju pakete. Osim toga, uspostavljanje nekog komunikacionog protokola izmedu njih samo povećava saobraćaj na mreži i komplikuje problem koji pokušavamo da rešimo.

- **Dodela bafera** Ovaj pristup može da se koristi kod virtuelnih kola. Sećate se iz odeljka **1**.4 da je **virtuelno kolo** uspostavljena ruta između mrežnih čvorova koja je utvrđena pre nego što su paketi sa podacima poslati. Kada se ruta uspostavi, protokoli u čvoru na toj ruti mogu da rezervišu bafere za virtuelno kolo. Uspostavljeno virtuelno kolo obaveštava čvorove učesnike o predstojećim paketima i podseća ih da treba da isplaniraju njihovo prihvatanje. Ako drugi zahtevi za uspostavljanje virtuelnog kola dolaze do tog čvora, može da ih odbaci ako nema dovoljno raspoloživog prostora u baferu. Nakon toga, mrežni protokoli mogu da pronadu drugu rutu za kolo, ili da obaveste izvor da je zahtev za virtuelnim kolom odbijen. U odeljcima 13.3 i 13.4 detaljnije ćemo predstaviti protokole virtuelnih kola.

- **Prigušivanje paketa** Ovaj pristup obezbeduje dinamičniji način za rešavanje zagušenja. Svaki čvor nadgleda aktivnost na svojim odlazećim linkovima, prateći iskorišćenje svakog od njih. Ako je iskorišćenje linije suviše malo, opasnost od zagušenja je niska. Međutim, iskorišćenje se povećava ako se šalje veći broj paketa. Ako iskorišćenost neke linije premaši neki definisani kriterijum, protokol u konkretnom čvoru reaguje, postavljajući čvor u specijalno stanje upozorenja.

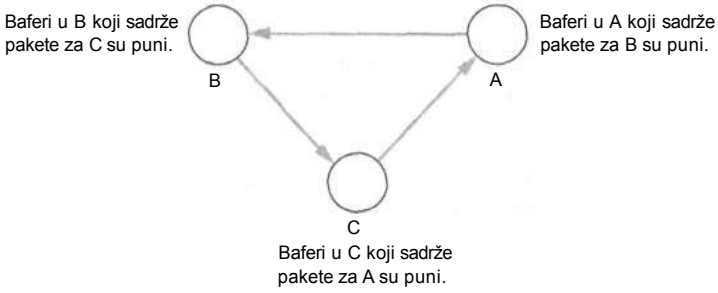
Kada se nalazi u tom stanju, čvor reaguje slanjem specijalnog Choke paketa kao odgovora na sve dolazeće pakete koji treba da se proslede preko njegovih odlazećih linija. Choke paket ide do izvora dolazećeg paketa. Kada izvor primi Choke paket, reaguje redukovanjem broja paketa koje šalje za određeni period.

Nakon što istekne zadati period, moguće **su** dve "stvari". Ako ne stignu dodatni Choke paketi, čvor može da poveća učestalost slanja paketa na njenu originalnu vrednost. Ukoliko su u toku perioda ćekanja Choke paketi i dalje stizali do izvora, dolazi do daljeg redukovanja učestalosti slanja. Redukovanjem dolazećeg saobraćaja originalni čvor dobija šansu da obori stepen iskorišćenja svojih odlazećih linija do prihvatljivog nivoa.

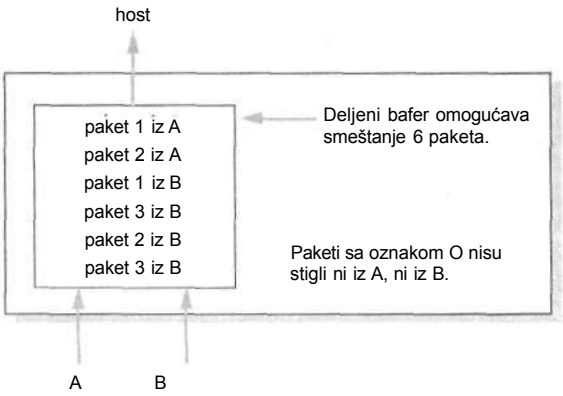
"Samrtni zagrljaj"

U najgorem slučaju, zagušenje može da postane toliko ozbiljno da ništa ne može da se pomeri. Na slici 10.33 ilustrovan je taj problem. Tri čvora A, B i C su stigla do tačke u kojoj su njihovi baferi puni i ne mogu da prihvataju nove pakete. Paketi iz A su namenjeni za B, koji ne može da primi nove pakete, jer su mu baferi puni. Tako A ne može ništa da pošalje do B sve dok B ne pošalje svoje pakete i oslobodi mesto u baferima. Paketi iz B su namenjeni za C_1 čiji su baferi takode puni. Paketi iz C su namenjeni za A, čiji su baferi puni. Drugim rečima, A čeka da se isprazne baferi u B, B čeka da se isprazne baferi u C, a C čeka da se isprazne baferi u A. U ovoj situaciji svi čvorovi čekaju na dogadaje koji se neće desiti, pa kažemo da je došlo do **samrtnog zagrljaja (deadly embrace,** mada se u literaturi na engleskom jeziku sreću i termini **dead-Iock,** ili **lock-up).**

Upravo opisani slučaj je primer **"samrtnog zagrljaja"** tipa **"smesti i prosledi" (store-and-forward),** koji se tako opisuje zato što čvorovi smeštaju pakete dok čekaju da ih proslede dalje. Na slici 10.34 ilustrovan je drugi mogući tip - **"samrtni zagrljaj" koji nastaje zbog ponovnog sastavljanja paketa (reassembly deadlock).** U ovom primeru čvorovi koriste zajedničke bafere za dolazeće pakete iz različitih izvora (A i B). Takođe se koristi protokol klizajucih prozora sa selektivnom retransmisijorn za prijem paketa iz A i B koji su namenjeni hostu. Sećate se da protokol selektivne retransmisije dopušta dolazak paketa van redosleda.



SLIKA 10.33 *"Samrtni zagdjaj" tipa "smesti i prosledi"*



SLIKA 10.34 *"Samrtni zagrljaj" nastao zbog ponovnog sastavljanja paketa*

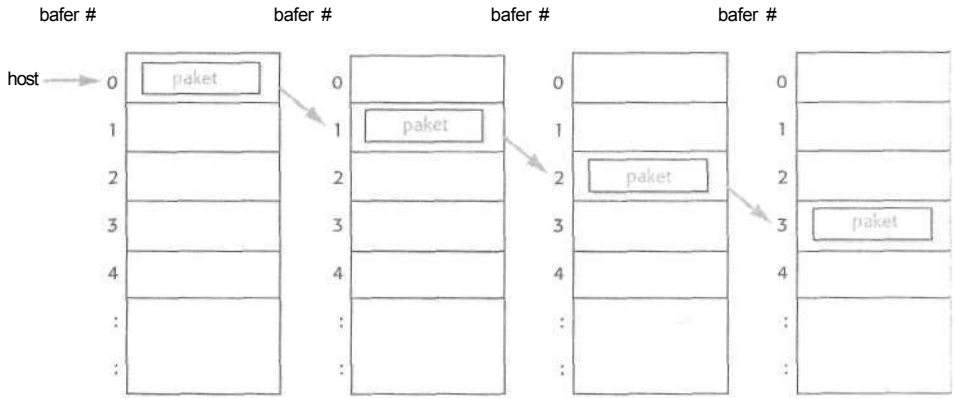
Nakon toga, u primopredajniku se vrši ponovno sastavljanje, pre nego što se proslede do hosta.

U našem primeru i A i B su poslali pakete sa oznakom 0, ali oni nisu stigli. Medutim, naredni paketi numerisani od 1 do 3 iz A i B stižu. Ako pretpostavimo da su baferi ispunjeni, čvor ne može da prihvata nove pakete. Pošto paketi O i dalje nedostaju, čvor ne može ponovo da ih sastavi i da ih isporuči hostu. Osim toga, čvor više ne može da prihvati pakete 0, čak i ako bi kasnije stigli. Zato se čvor postavlja u stanje u kome niti može da preduzme neku akciju, niti može da reaguje na dogadaj koji bi stvorio uslove za akciju. Došlo je do "samrtnog zagrljaja".

"Samrtni zagrljaj" ovog tipa može da se spreči usaglašenim uspostavljanjem konekcije izmedu predajnih i prijemnih čvorova. Utvrđuje se veličina prozora i primalac može da rezerviše dovoljno prostora i koristi taj prostor samo za tu konekdju. "Samrtni zagrijaj" tipa "smesti i prosledi" se teže rešava. Može da se redukuje, ili eliminiše korišćenjem većih bafera, ali problem je kako utvrditi koliko mesta u baferu treba rezervisati, posebno kod servisa datagrama kod kojih paketi mogu da pristižu i odlaze nasumice.

Jedno moguće rešenje podrazumeva dopuštanje situacije u kojoj dolazi do "samrtnog zagrljaja", nakon čega se pristupa otklanjanju problema. Kada dode do "samrtnog zagljaja", obično se neki paketi odbacuju da bi se oslobodio prostor u baferu. Naravno, odbačeni paketi nikada ne stižu do svojih odredišta. To je cena koja se plaća za otklanjanje "samrtnog zagrljaja". Moguće je da će komunikacioni protokoli utvrditi da paketi nisu stigli do odredišta i kasnije će inicirati njihovo ponovno slanje. Ako se "samrtni zagrljaji" retko javljaju, ovo može da bude najbolje rešenje.

Sa druge strane, ako se ovo često dešava, više se isplati preduzeti korake da do "samrtnog zagrljaja" ne dode, ili bar smanjiti verovatnoću da se on desi. Sve prethodno predstavljene tehnike za kontrolu zagušenja smanjuju šanse za "samrtni zagrljaj", ali i dalje nema garancija da do njega neće doći. Sledeći pristup (ref. [Me80]) koristi broj "skokova" (čvorova preko kojih paket putuje) za svaki paket.



SLIKA 10.35 *Smeltanje paketa u zavisnosti od broja "skokova"*

Kada host postavi paket na mrežu, broj "skokova" je postavljen na O. Kako paket putuje preko mreže, broj "skokova" se u svakom čvoru inkrementira za 1. Osim toga, svaki čvor deli svoje bafere na različite grupe, tako da te grupe odgovaraju različitim brojevima "skokova", od O do maksimalnog očekivanog broja. Nakon toga, čvor smešta dolazeći paket u odgovarajući bafer, u zavisnosti od broja skokova koji je zabeležen u konkretnom paketu, ali samo ako je bafer dostupan. Ako bafer nije dostupan, paket čeka u prethodnom čvoru sve dok se bafer ne oslobodi. Na slici 10.35 prikazano je kako to funkcioniše. Host inicijalno predaje paket i čvor hosta postavlja paket u bafer 0. Kada sledeći čvor primi paket, on ga postavlja u bafer I, sledeći čvor u bafer 2 i tako redom.

Ovaj metod sprečava "samrtni zagrljaj", jer paket uvek ide u bafer sa višim rednim brojem (ili čeka). Drugi način da se to opiše je da se kaže da jedan bafer nikada neće čekati na bafer sa nižim, ili istim rednim brojem. Tako su izbegnuta kružna čekanja, koja postoje na slici 10.33. Argument protiv ovog pristupa je moguća neiskorišćenost bafera. Dodeljivanje paketa baferu unapred sprečava njegov prenos svaki put kada je taj bafer zauzet; međutim, ako je samo jedan bafer zauzet, ostali ostaju neiskorišćeni.

10.9 Zaključak

U ovom poglavlju su obradeni načini za povezivanje mreža i neki uređaji i protokoli koji to omogućavaju. Poglavlje smo organizovali u skladu sa slojem na kome je mreža povezana, a fokusirali smo se na konekcije slojeva 1, 2 i 3.

Uređaji sloja 1 - repetitori i hubovi - primarno proširuju geografska rastojanja na kojima je moguće implementirati LAN protokol. Glavna razlika izmedu repetitora i huba je u činjenici da hub ima više portova od repetitora. Glavno zaduženje oba uredaja je prihvatanje signala preko jednog porta, regenerisanje tog signala i prosleđivanje preko svih ostalih portova.

Ako LAN protokol koristi CSMA/CD, svi uređaji koji su povezani repetitorima i hubovima pripadaju istom domenu kolizije.

Mostovi i komutatori funkcionišu na sloju 2 i obično dele uredaje u zasebne domene kolizije. Međutim, svi uredaji koji se povezuju na ovakav način pripadaju istom domenu emisije. Glavna razlika između ova dva uređaja proističe iz činjenice da komutator ima veći broj portova od mosta. Oba su zadužena za rutiranje na sloju 2, selektivno prosledivanje, ili ignorisanje okvira na osnovu njegove MAC adrese i koriste interne tabele rutiranja. Mostovi se dele na različite tipove, u skladu sa tim kako donose odluke o rutiranju:

- Mostovi sa fiksnim rutiranjem imaju programirane informacije o rutiranju.
- Transparentni mostovi kreiraju svoje tabele aitiranja proučavanjem saobraćaja i beleženjem lokacija uredaja na osnovu izvornih adresa u okvirima.
- Mostovi sa izvornim rutiranjem donose odluke o rutiranju na osnovu informacija iz okvira.

Komutatori su uobičajeni izbor kada je potrebno povezati Ethernet uredaje, što je dovelo do razvoja full-duplex komutiranog Etherneta, Ethernet protokola kod koga su kolizije eliminisane, tako da nema potrebe za CSMA/CD protokolom. Komutatori mogu da se koriste i za kreiranje virtuelnih LAN-ova (VLAN), logičkih grupa uređaja koji mogu da funkcionišu kao zaseban LAN. Grupe ne zavise od fizičkih lokacija svojih uređaja, a definisani su i zasebni domeni emisije.

Uređaji sloja 3 (ruteri) povezuju mreže koje se prostiru na mnogo većim geografskim područjima. Koriste mnogo složenije topologije i zahtevaju neke sofisticirane algoritme za rutiranje. Predstavili smo četiri osnovna tipa rutiranja:

- *Centralizovano:* Informacije o rutiranju se održavaju na centralnoj lokaciji.
- *Distribuirano:* Informacije o rutiranju su distribuirane do čvorova.
- *Statičko:* Informacije o rutiranju se ne menjaju, iako dolazi do promene uslova na mreži.
- *Adaptivno:* Promena uslova na mreži dovodi do promene informacija o rutiranju.

Jedan način za upravljanje informacijama o rutiranju je korišćenje tabela rutiranja koje naznačavaju kuda treba proslediti dolazeće pakete. Održavanje ovakvih informacija zavisi od korišćenog algoritma za rutiranje. Prikazali smo nekoliko mogućih pristupa, medu kojima su:

- Dijkstrin algoritam, centralizovani algoritam dizajniran radi utvrdivanja najjeftinije putanje izmedu dva čvora
- Bellman-Fordov algoritam, distribuirani pristup kod kojeg svaki čvor sa svim svojim susedima razmenjuje informacije o dostupnim čvorovima.
- Rutiranje zasnovano na stanju linka, kod koga čvorovi razmenjuju pakete stanja linka, tako da se obezbede informacije o statusu susednih linkova. Cvor može da sakupi sve pakete i da, na osnovu njih, utvrdi mrežnu topologiju. Nakon toga, može da izvrši sopstveni algoritam za pronalaženje najkraće rute.
- Hijerarhijsko rutiranje, kod kojeg se čvorovi dele na grupe (domene), koje izvršavaju sopstvene protokole za rutiranje

- Routing Information Protocol, unutrašnji protokol koji je dizajniran za razmenu informacija izmedu rutera o dostupnim mrežama i broju neophodnih "skokova" za dolazak do tih mreža.
- Protokol za unutrašnje rutiranje Open Shortest Path First liči na rutiranje zasnovano na stanju linka, ali obezbeđuje dodatne karakteristike koje daju bolje performanse i veću fleksibilnost.
- Border Gateway Protocol, spoljašnji protokol koji omogućava implementiranje specifičnih polisa, ili ograničenja u ruterima koja odreduju moguće rute. Ruteri razmenjuju informacije o konkretnim rutama, umesto da se šalju samo cena i prvi sledeći link u okviru rute.

Predstavili smo dva moguća problema u toku rutiranja. Zagušenje se javlja kada čvor primi više paketa nego što može efikasno da kontroliše, što produžava vreme potrebno za njihovo prosledivanje. "Samrtni zagrljaj" se javlja kada postoji ciklična lista čvorova, u kojoj svi čekaju da proslede pakete do sledećeg čvora u listi.

Pitanja i zadaci za proveru

1. Koje su razlike između repetitora, mosta i rutera?
2. Koje su razlike izmedu repetitora i huba?
3. Koje su razlike izmedu mosta i komutatora?
4. Sta je domen kolizije?
5. Sta je emisioni domen?
6. Navedite glavne razloge za korišćenje mostova, ili komutatora izmedu dva LAN-a.
7. Šta je tabela rutiranja?
8. U čemu je razlika izmedu mosta sa fiksnim rutiranjem i transparentnog mosta?
9. Sta je algoritam plavljenja i koja je njegova svrha?
10. Kakav problem može da stvori plavljenje ako postoji petlja u LAN topologiji?
11. Šta je stablo razapinjanja (spanning tree) u kontekstu povezivanja LAN-ova?
12. Sta je root most?
13. Šta je root port mosta?
14. Koja je svrha označenog mosta u LAN-u?
15. Šta je most sa izvornim rutiranjem?
16. ŠtajeBPDU?
17. Šta je konvertor protokola?
18. Na kojem sloju OSI modela se nalaze konvertori? Navedite najčešće implementirane i njihove nazive.
19. Šta je virtuelni LAN i zašto se kreira?
20. U čemu je razlika izmedu privatnog komutatora i komutatora radne grupe?
21. Kako komutator utvrduje kojem VLAN-u uredaj pripada?

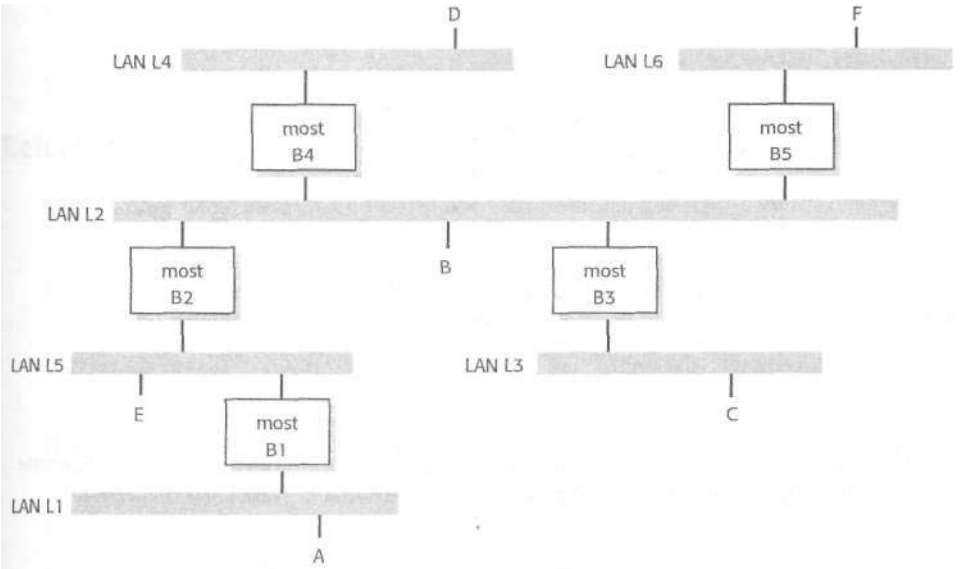
22. Navedite četiri glavne klasifikacije rutiranja i istaknite karakteristike svake od njih.
23. Da li su sledeće tvrdnje tačne, ili netačne (zašto)?
    a. Povećanje broja hubova koji se koriste za povezivanje Ethemet uređaja uvek ima prednosti, jer se tako povećavaju udaljenost "pokrivena" mrežom i broj povezanir uredaja.
    b. Komutator razdvaja uredaje na različite domene emisije.
    c. Spanning tree algoritam emiliniše petlje koje stvaraju redundantni mostovi.
    d. Komutirani Ethernet eliminiše potrebu za fizičkim kablovima za povezivanje uredaja.
    e. Distribuirano rutiranje zahteva postojanje tabela rutiranja u svim čvorovima.
    f. Adaptivno rutiranje omogućava čvorovima da ažuriraju svoje tabele rutiranja.
    g. Najkraća putanja i najjeftinija putanja u opštem slučaju imaju isto značenje.
    h. Hijerarhijsko rutiranje organizuje sve mrežne čvorove u strukturu stabla.
    i. Zagušenje znači da je na svim mrežnim putanjama saobraćaj zakrčen.
    j. Zagušenje ne vodi obavezno do "samrtnog zagrljaja".
24. Navedite razlike izmedu algoritama traženja unapred i traženja unazad.
25. Sta je hijerarhijsko rutiranje?
26. Zašto rutiranje zasnovano na stanju linka rešava problem koji je karakterističan za Bellman-Fordov algoritam?
27. Navedite razlike između unutrašnjih i spoljašnjih protokola.
28. Šta rade čvorovi koji koriste Routing Information Protocol za održavanje svojih tabela rutiranja?
29. U čemu je razlika između RIP i RIP-2 protokola?
30. Zašto je teško utvrditi najjeftiniju, ili najbolju putanju na velikim udaljenostima?
31. Ruter šalje paket samo do prvog sledećeg čvora na putanji. Zašto ne bi znao kompletnu sekvencu autonomnih sistema kroz koje paket može da prođe?
32. Navedite razlike između zagušenja i "samrtnog zagrljaja".
33. Navedite neke načine za otklanjanje zagušenja.
34. Koja su dva moguća tipa "samrtnog zagrljaja"?
35. Zašto dodeljivanje paketa specifičnim baferima, onako kako je prikazano na slici 10.35, sprečava "samrtne zagrljaje" tipa "smesti i prosledi"?

Vežbe

1. Uzmite za primer strategiju medusobnog povezivanja LAN-ova, kod koje se oni koji najčešće komuniciraju povezuju sa najmanjim mogućim brojem mostova. Pretpostavite da sledeći parovi LAN-ova ne smeju da budu rastavljeni sa više od naznačenog broja mostova. Dizajnirajte medusobne konekcije koje koriste najmanji broj mostova:

- Jedan most: L1 i L5; L2 i L3; L2 i L4.
- Dva (ili manje) mostova: L1 i L3; L2 i L5; L1 i L2

2. Kreirajte tabele rutiranja za sledeće mostove.



3. Uzmite za primer LAN-ove sa slike 10.5 i pretpostavite da su sve tabele rutiranja inicijalno prazne. Pretpostavite da se događaji dešavaju u skladu sa narednom navedenom listom. Preko kojih LAN-ova se prenose naznačeni okviri? Prikažite zapise tabele rutiranja u svakom mostu nakon što se pošalju svi okviri.

   a. A šalje okvir do F.

   b. E šalje okvir do A.

   c. D šalje okvir do E.

   d. C šalje okvir do B.

4. Navedite sve rute izmedu A i B sa slike 10.8. Ignorišite sve rute kod kojih okvir prolazi preko istog mosta dva puta. Ako se ne ignorišu te rute, koliko ukupno ima ruta?

5. Pretpostavimo da most Bl sa slike 10.8 "otkaže". Izvršite spanning tree algoritam i pokažite novi root most, najjeftinije portove mostova, izabrane mostove i rezultujuće osposobljene konekcije mostova.

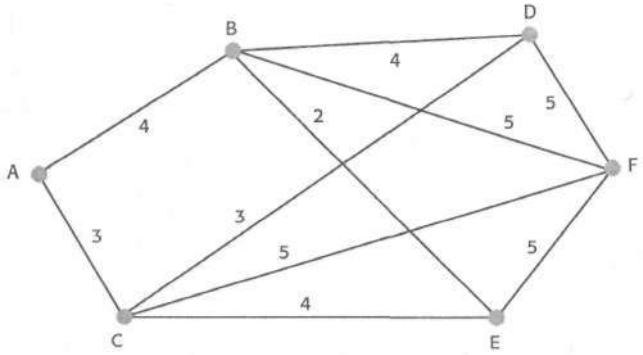
6. Pretpostavimo da su unutrašnje mreže na slici 10.8 promenjene, tako da su sve navedene cene jednake. Uz pretpostavku da je cena prenosa izmedu dva različita LAN-a sada jednaka broju mostova preko kojih okviri moraju da predu, utvrdite spanning tree.

7. Pretpostavimo da A šalje emisioni okvir do B sa slike 10.8. Koliko kopija stiže u B?

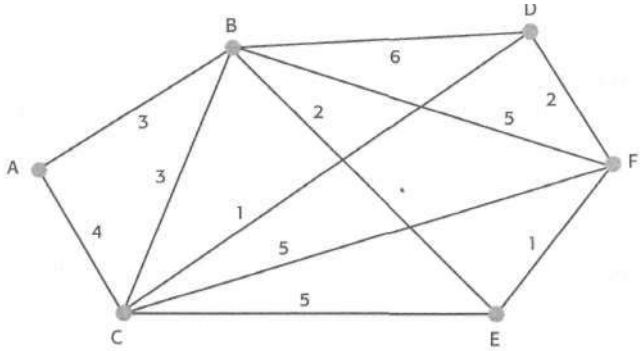
8. Koliko različitih ruta postoji od X do Y na mreži sa slike 10.8?

9. Definišite tabele aitiranja za sve čvorove na mreži sa slike 10.19.

10. Kreirajte matricu rutiranja za sledeću mrežu. U slučaju da postoje dve rute sa najnižom cenom, izaberite onu sa manjim brojem čvorova. Ako su oba kriterijuma ista, ruta se bira proizvoljno.



11. Primenite Dijkstrin algoritam za pronalaženje najkraćeg puta za sledeću mrežu. Kreirajte tabelu sličnu tabeli 10.3, sa odgovarajućim vrednostima za svaki korak algoritma.



12. Dizajnirajte algoritam koji, kada se izvrši nakon Dijkstrin algoritma, daje listu svih čvorova sa najjeftinijim putanjama do datog odredišta.

13. Koristeđ terminologiju Dijkstrin algoritma, dokažite da svaki put kada se funkcija Cost(V) promeni, i dalje predstavlja najjeftiniju aitu od A do V preko čvorova iz S.

14. Premisa Bellman-Fordovog algoritma glasi da ako je Cost(A, Z) cena najjeftinije rute od čvora A do čvora Z i ako A ima direktnu konekciju sa čvorovima B, C i D, onda je

Cost(A, Z) = najmanja vrednost od
- cena linka od A do B + cena najjeftinije rute od B do Z
- cena linka od A do C + cena najjeftinije rute od C do Z
- cena linka od A do D + cena najjeftinije mte od D do Z

Zašto je ova premisa validna? Dokažite tu tvrdnju.

15. Kreirajte tabele slične tabelama 10.5a do 10.5c primenom Bellman-Fordovog algoritma na sledećoj mreži.

16. Po čemu je Routing Information Protocol sličan Bellman-Fordovom algoritmu?

Reference

[Ah83] Aho, A., J. Hopcroft, and J. Ullman. *Data Structures and Algorithms.* Reading, MA: Addison-WesIey, 1983.

[Co99] Corner, D. E., and D. Stevens. *Internetworking with TCP/IP. VoI. II. ANSI C Version: Design, Implementation, and Internals.* 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1999.

[CoOO] Corner, D. E. *Internetivorking with TCP/IP. VoI. 1. Principles, Protocols, and Architecture.* 4th ed.Englewood Cliffs, NJ: Prentice-Hall, 2000.

[Di59] Dijkstra, E. "A Note on Two Problems in Connection with Graphs." *Numerical Mathematics,* October 1959, 269-271.

[Dr95] Drozdek, A., and D. Simon. *DaU Structures in C.* Boston: PWS, 1995.

[Fo62] Ford, L., and D. Fulkerson. *Flows in Networks.* Princeton, NJ: Princeton University Press, 1962.

[Fo03] Forouzan, B. *Local Area Netivorks.* New York: McGraw-Hill, 2003.

[Me80] Merlin, P.M., and PJ. Schweitzer. "Deadlock Avoidance in Store-and-Forward NetWorks I: Store-and-Forward Deadlock." *IEEE Transactions on Communications,* vol. COM-28 (March 1980), 345-354.

[Pa95] Parsons, T. *Introduction to Algorithms in Pascal.* Nevv York: Wiley, 1995.

[Ta03 JTanenbaum, A. S. *Computer Netuiorks,* 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2003.

POGLAVLJE

11

Internet protokoli i aplikacije

Od svetskih naroda, osim ako možda izuzemo neka beduinska plemena, Amerikanci su najskhniji dezinformacijama. Ovo nije posledica bilo kakve posebne sklonosti ka laganju, iako je ta tendencija u vrhovima državne administracije dostigla impresivne razmere. To je više posledica činjenice da, uglavnom, veruju u pogrešne "stvari".

—**John Kenneth Galbraith,** američki ekonomista

11.1 Uvod

Osim ako poslednjih 10 godina niste proveli u udaljenim oblastima Himalaja, sigurno ste svesni uticaja koji je Intemet imao i na naš posao i na slobodno vreme. Skoro sve vrste biznisa su postavile svoje Web sajtove sa opisima onoga šta rade, ili za oglašavanje svojih proizvoda. I individualci postavljaju svoje Web sajtove, na kojima može da se nađe skoro sve što možete da zamislite. Izmišljeni igrači fudbala i bejzbola redovno koriste Internet za razmene i nadmetanja. Mnogi koriste Internet za preuzimanje muzike i video fajlova, osim za ažuriranje softvera i ostalih tipova programa. Igrači kompjuterskih igara mogu da se takmiče preko Interneta. Pre 10-ak godina niko nije mogao da predvidi da će se ovo desiti.

Ali, šta je, u stvari, Internet? Mnogi su svesni njegovog postojanja, ali mali broj ljudi stvamo razume kako on funkcioniše.

U ovom poglavlju se bavimo Internetom, šta je, kako funkcioniše i koje su najčešće aplikacije zasnovane na njemu. Odeljak 11.2 startuje sa **Internet Protocolom (IP).** U stvari, postoje dva Internet protokola. Jedan je ISO standard i deo je sloja mreže u okviru OSI modela.

Ovde nećemo razmatrati taj protokol. Drugi je zasnovan na Internet protokolu koji je razvijen u U.S. Defense Advanced Research Projects Agency (DARPA). Korišćen je na sloju mreže ARPANET-a i najčešće se implementira zajedno sa **Transmission Control Protocolom (TCP).**

Zajedno su poznati kao TCP/IP, a formiraju protokole sloja 3 i 4 koji se koriste za povezivanje komercijalnih, istraživačkih, vojnih i obrazovnih mreža. Ovi protokoli su razvijeni u ono što danas nazivamo Internet.

Većina sajtova koristi verziju 4 Internet Protocola, poznatu pod nazivom Ipv4. Dizajniran je u vreme kada su transfer fajlova i email bile dominantne aplikacije za koje je bilo potrebno obezbediti podršku. Ko je u to vreme mogao da predvidi globalnu mrežu preko koje će se prenositi video zapisi (i to u realnom vremenu), na kojoj će postojati lični Web sajtovi, postojati pretnje po bezbednost informacija i gde će biti neophodna implementacija autentifikacije? Zbog svega toga, Ipv4 se razvijao i sve veći broj sajtova je prešao na verziju 6, ili Ipv6. U odeljku 11.3 razmatramo Ipv6 i kako se on "bori" sa problemima koje Ipv4 nije uspeo da reši. U tom odeljku i o koegzistenciji tih verzija protokola, koja mora da postoji dok se ne zamene sve implementirane verzije Ipv4.

U odeljku 11.4 biće reči o nekim protokolima sloja 4 implementiranim na Internetu. Najvećim delom odeljak je posvećen TCP protokolu, vezi orijentisanom protokolu izmedu krajnjih tačaka. Videćemo da obe strane učestvuju u usaglašavanju za uspostavljanje logičke konekcije pre izvršavanja protokola koji razmenjuju informacije. Već smo obradili mnoge koncepte i ponovo ćemo se vratiti na njih, ali u kontekstu TCP protokola.

Osim toga, obradićemo i User Datagram Protocol (UDP), protokol sloja 4 koji se izvršava bez uspostavljanja konekcija, i Real-Time Transport Protocol (RTP), protokol koji je razvijen za potrebe real-time aplikacija. Konačno, u odeljku 11.5 predstavljamo neke aplikacije koje se mogu izvršavati preko Interneta.

Proučićemo Telnet i neke druge sigurnije protokole za daljinsko logovanje. Osim toga, pogledaćemo i FTP (File Transfer Protocol), protokole za transfer elektronske pošte i protokole za upravljanje. Primetićete da su aplikacije zasnovane na Webu izostavljene. To je toliko značajna tema da nismo hteli da je obrađujemo zajedno sa ostalim temama, već smo joj posvetili posebno poglavlje.

Opisivanje načina na koji Internet funkcioniše je izuzetno obimno: u ovom poglavlju prikazujemo samo najznačajnije aspekte protokola. O TCP/IP protokolu su napisane cele knjige. Čitaoci treba da razumeju da mi ovde obrađujemo samo osnove protokola.

11.2 lnternet protokol

Verovatno najpoznatija WAN mreža, koja se, u stvari, sastoji od ogromnog broja mreža, poznata je kao lnternet. Njen istorijat datira još sa kraja 60-ih godina prošlog veka, kada je Advanced Research Projects agencija (ARPA) američkog ministarstva odbrane (DoD-U.S. Department of Defense) počela da prikuplja sredstva od univerziteta i privatnih organizacija za razvoj komunikacionih sistema. Istraživanja su na kraju dovela do razvoja ARPANET-a, male eksperimentalne mreže koja je demonstrirala mogućnost povezivanja različitih kompjutera pomoću mreže sa komutacijom paketa u međuvremenu je značajno porasla i razvila se u lnternet, koji danas povezuje hiljade univerziteta, privatnih institucija i vladinih agencija širom sveta.

Mnogi koriste termin *lnternet* kako bi se ukazalo na bilo koju kolekciju povezanih mreža. Mreža koja je nastala na osnovu ARPA projekta obično se označava kao lnternet. Ona se sastoji od više hiljada zasebnih mreža. Teško je proceniti koliko ljudi danas koristi lnternet, ali sa mrežama koje

postoje bukvalno u svim pnvatmm i javnim orgamzacijama i sa sve vecim brojem Internet provajdera, broj korisnika dostiže desetine miliona, a sasvim je moguće da je reč i o nekoliko milijardi korisnika širom sveta. U ovom i sledećem odeljku dali smo opšti pregled IP i TCP protokola. Medutim, postoje knjige posvećene isključivo tim temama, a zainteresovani čitaoci mogu da pogledaju reference [Co99], [CoOO], i [Mi99],

Pregled TCP/IP protokola

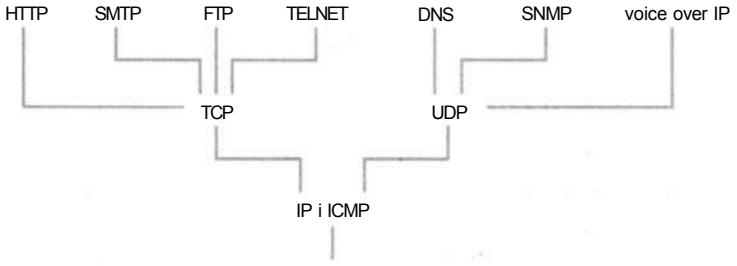
Intemet se sastoji od velikog broja mreža, koje izvršavaju protokol poznat kao TCP/IP. TCP (Transmission Control Protocol) i IP (Intemet Protocol) su protokoli sloja 4 i 3, respektivno. Razvijeni su u okviru ARPA projekta i kasnije su postali DoD standardi. TCP/IP je verovatno najčešće implementirani protokol u svetu; izvršava se na skoro svim konfiguracijama, od personalnih kompjutera do super kompjutera.

Par protokola TCP/IP je deo kolekcije protokola poznatih pod nazivom *TCP/IP skup protokola* (slika 11.1). TCP obezbeduje konekciji orijentisane servise za Intemet aplikacije sloja S i oslanja se na IP za rutiranje paketa preko mreže. Ove aplikacije obezbeduju specifične servise za korisnike Interneta.

Dve strane koje implementiraju TCP najpre se usaglašavaju tako da se omogući uspostavljanje logičke konekcije između njih. Nakon toga, svaka strana izvršava protokole za kontrolu toka, za potvrde segmenata i za reagovanje na situacije u kojima se javljaju oštećeni paketi, tako da se obezbede pouzdane komunikacije.

Prethodnik TCP protokola u originalnom ARPANET-u bio je NCP (Netvvork Control Protocol), koji je dizajniran za pokretanje na vrhu pouzdane mreže. ARPANET je bio dovoljno pouzdan, ali je sa razvojem u kolekciju povezanih mreža izgubio na pouzdanosti. Zato je i transportni protokol morao dalje da se razvija. NCP koji je redizajniran za izvršavanje preko nepouzdanih mreža postao je poznat kao TCP.

Nekoliko aplikacija se oslanja na TCP. Na primer, SMTP (Simple Mail Transfer Protocol) definiše protokol koji se koristi za isporuku email poruka preko Interneta.



SLIKA 11.1 *Internet protokoli*

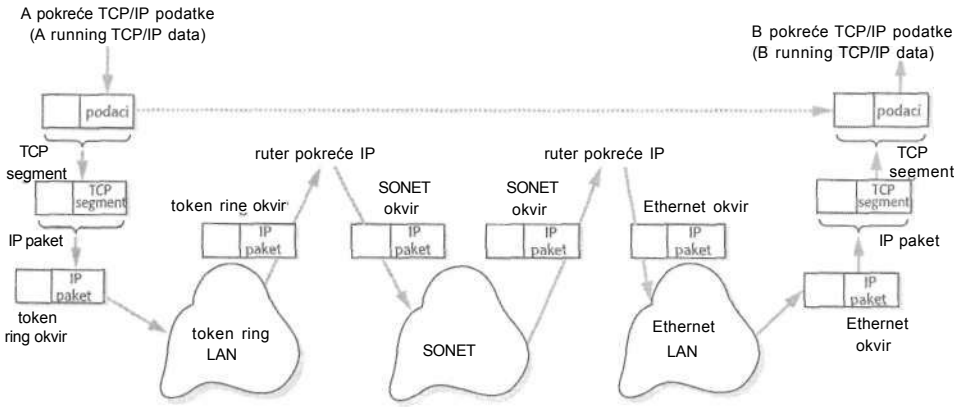
Telnet protokol korisnicima omogućava logovanje na udaljene kompjutere preko Interneta. FTP (File Transfer Protocol) omogućava korisnicima Interneta da prenose fajlove sa udaljenih kompjutera. HTTP (Hypertm Transfer Protocol) se koristi kao podrška Web pretraživačima. Ove protokole ćemo razmotriti kasnije.

UDP (User Datagram Protocol) je alternativni protokol sloja 4. Razlika u odnosu na TCP je to što UDP obezbeduje komunikacije između različitih mreža bez uspostavljanja konekcije. Eliminisani su troškovi uspostavljanja konekcije; zato je omogućen brži prenos podataka. Sporedni efekat je što nema kontrole toka i UDP ne ozbezbeduje negativne potvrde za oštećene segmente. Naime, nema garancija za pouzdanu isporuku. Nekoliko aplikacija je dizajnirano za pokretanje preko UDP-a. Na primer, DNS (Domain Name System) obezbeđuje mapiranje naziva hosta u adrese. SNMP (Simple Netvvork Management Protocol) je protokol za upravljanje dizajniran za obezbedivanje funkcionalnosti protokola za upravljanje i uredaja. Voice over **IP (prenos** glasa preko IP-ja) je termin za Internet telefoniju, koji se odnosi na protokole dizajnirane za isporuku paketa sa govornim informacijama.

UDP i TCP omogućavaju dva tipična moda za komuniciranje. 1 TCP i UDP čemo detaljnije razmotriti u narednom poglavlju.

Internet Protocol je protokol sloja 3 dizajniran za obezbedivanje isporuke paketa izmedu dva sajta. Uobičajeno je da se koristi sa TCP protokolom. Na slici 11.2 prikazano je kako funkcioniše saTCP protokolom. Pretpostavimo da dva sajta (A i B) zahtevaju konekciji orijentisani servis za prenos odredene količine podataka.

Najčešći primeri su email, transfer fajlova i većina aplikacija zasnovanih na Webu. TCP obezbeđuje pouzdane konekcije nezavisne od mrežnih arhitektura na stranama koje se povezuju, a IP je zadužen za rutiranje paketa preko različitih mreža. To je unekoliko slično telefonskim pozivima. Na jednom nivou jednostavno okrenete nečiji broj i neko na drugom kraju odgovori. Uspostavili ste konekciju. Vi ne morate da znate kako je konekcija izvedena, ili kroz koliko je telefonskih komutatora poziv morao da prođe.



SLIKA 11.1 *Prenos IP paketa preko različitih mreža*

Telefonske kompanije koje učestvuju u uspostavljanju poziva zadužene su za sve detalje upravljanja. Na slici 11.2 prikazane su moguće Internet konekcije izmedu A i B.

Za početak, TCP na sajtu A kreira TCP segment* u kome se nalaze korisnički podaci i šalje segment do sajta B. Ako sve protekne u najboljem redu, B će poslati poruku za ono što primi. Sa stanovišta TCP-ja, uspostavljena je direktna konekcija sa B (isprekidana linija).

Međutim, IP prekida segment i kreira *IP paket* (čiji format ćemo uskoro objasniti), u sklopu koga će se naći i TCP segment. Možda A predstavlja kompjuter kompanije i paketi moraju da se rutiraju preko token ring LAN-a. U tom slučaju protokoli veze kreiraju token ring okvir, postavljaju IP paket u polje Data novog okvira i šalju okvir do rutera preko token ring mreže. Protokoli nižeg sloja ne znaju da, u stvari, prenose IP paket. Oni jednostavno izvršavaju zadatke koji su naznačeni u prethodnom poglavlju, isporučujući sve pristigle podatke.

Kada okvir stigne do rutera, njegov sloj veze izvlači IP paket iz token ring okvira i daje ga IP-ju rutera. IP proučava adresu paketa i, na osnovu tabela rutiranja, utvrđuje da treba da prosledi paket do drugog rutera preko SONET nosioca. Niži slojevi rutera ugraduju IP paket u SONET okvir i šalju taj okvir do drugog rutera.

Drugi ruter takode mora da donese odluku o rutiranju. U ovom slučaju on zaključuje da je željeni primalac povezan na Ethernet mrežu kojoj ima pristup. Zbog toga, sloj veze rutera na tom LANlI-u kreira Ethernet okvir, smešta IP paket u njega i šalje ga preko Etherneta.

Na kraju, Ethemet okvir stiže do odredišta, gde protokoli sloja veze Etherneta izvlače podatke (IP paket) i daju ga IP-ju u B. IP interpretira paket i predaje TCP segment TCP-ju, koji eventualno izvlači podatke i daje ih B. Ovaj jednostavni opis služi kao ilustracija uloge IP-a u rutiranju paketa preko različitih vrsta mreža.

Adresiranje na Internetu

Pošto je rutiranje značajan deo Internet protokola, važno je da razumete kako izgleda Internet adresa i kako je ruter interpretira. Za korisnike Intemeta Internet adresa ima sledeći oblik

server.institucija.domen

Ova forma može da bude sastavni deo email adrese, na primer, korisnik@server.institucija.domen, ili može da bude deo Web adrese, http://server.institucija.domen.

Međutim, ovo nije stvarna Internet adresa, već tekstualna reprezentacija koja identifikuje host kompjuter, ili server neke institucije koja je povezana na Internet.

U slučaju emaila karakteri pre simbola @ predstavljaju korisnika koji je autorizovan za pristup email serveru. U slučaju Web pretraživača obično se oznakom "www" predstavlja podrazumevani server na naznačenoj lokaciji.

* TCP segmenc, slično paketu, sadrži podatke i neke dodatne informacije. Njegov format predstavićemo u odeljku 11.4.

Medutim, može se koristiti i stvarni naziv servera umesto www (uz pretpostavku da je server postavljen tako da može da mu se pristupa sa Weba).

Tačke u tekstualnoj adresi služe za razdvajanje komponenata adrese. Krajnja desna komponenta se odnosi na Internet domen. Nije suvišno istaći da je u ovoin kontekstu domen kolekcija sajtova određenog tipa. Razlikuje se od domena koje smo opisivali u prethodnom poglavlju i u ovom kontekstu nema nikakve naznake geografske lokacije. Znači, dva sajta na istom domenu mogu da budu u istom gradu, ali mogu da budu udaljeni i hiljadama kilometara jedni od drugih. Ovi domeni se koriste prvenstveno za administriranje i organizovanje tekstualnih adresa, radi eventualnog prevodenja Internet adresa. Kao takvi, ne koriste se za rutiranje. U tabeli 11.1 navedeni su neki mogući nazivi domena i vrste sajtova kojima odgovaraju. Neki od tih domena su Vara verovatno već poznati, a neki drugi možda i nisu - razlog može da bude činjenica da su dodati u listu u skorije vreme i još nisu postali opštepoznati.

U ostalim delovima tekstualne adrese može da se nađe bilo šta; to isključivo zavisi od konkretnog sajta. Nazivi obično identifikuju organizaciju, odeljenje, ili specifični server (kasnije ćemo prikazati kako se izvodi prevodenje u IP adresu). Na primer, icsa.uwgb.edu ukazuje na Linux server u University of Wisconsin - Green Bay u obrazovnom domenu, dok www.nasa.gov ukazuje na podrazumevani Web server u National Aeronautics and Space Administration u Vladinom domenu. Ako je institucija dovoljno velika, može da se podeli po odeljenjima (na više poddomena) i host kompjutera (servera) unutar svakog odeljenja. Na primer, puna email adresa autora ove knjige je shayw@msa.uwgb.edu.

Tabela 11.1: Internet domeni

Domen	Značenje
com	Komercijalna institucija
edu	Obrazovna institucija
int	Medunarodna organizacija
gov	Vladina agencija
kids	Sajt sa sadržajem prikladnim za decu
mil	Vojni sajt
net	Provajder mrežnih usluga
org	Neprofitna organizacija
biz	Poslovni sajtovi
info	Opšte namene
pro	Profesionalni
museum	Muzeji
coop	Poslovna saradnja
aero	Avionska industrija
name	Za individue
kod države	Na primer, jp za Japan, ili nl za Holandiju

Ako je univerzitet dovoljno veliki da želi da koristi različite servere u različitim odeljenjima, odeljenja mogu da se identifikuju umetanjem oznaka izmedu "msa" i "uwgb". Kada je sajt dovoljno mali, može da postoji jedan podrazumevani server koji se koristi za aplikacije, kao što je email koji prima sve dolazeće poruke. Ovo definitivno zahteva navođenje konkretnog kompjutera. U našem slučaju server nazvan "msa" je podrazumevani; on omogućava navođenje autorove email adrese kao shayw@uwgb.edu, umesto dugačke verzije shayw@msa.uwgb.edu.

Primer tekstualne adrese sa više komponenata može da bude www.legis.state.wi.us. U ovom slučaju najviši domen je *us* (za zemlju). Podeljen je na poddomene; jedan od njih je *wi* (za Wisconsin). Poddomen *wi* je dalje podeljen, tako da se *state* koristi za predstavljanje državnih službi. Preostale komponente ukazuju na zakonodavne ogranke i na podrazumevani Web server. Ako su potrebne zakonodavne ustanove u drugim saveznim američkim državama, dovoljno je samo zameniti kod za state za odgovarajuću ustanovu. Primeri su www.legis.state.ia.us (Iowa), www.legis.state.wv.us (West Virginia), www.legis.state.ak.us (Alaska) i www.legis.state.de.us (Delaware). Moguće je zameniti i poddomen *legis* nekim drugim poddomenom. Na primer, www.dnr.state.wi.us (Department of Natural Resources) i www.dpi.state.wi.us (Department of Public Instruction).

Kako je ovo povezano sa stvarnom Internet adresom? U prethodnom odeljku smo pomenuli da je Internet kolekcija nezavisnih mreža i da svaka od tih mreža sadrži brojne kompjutere. Zbog toga, svaki kompjuter može da se identifikuje navođenjem mreže kojoj pripada i lokalne adrese po kojoj se razlikuje od ostalih kompjutera na toj mreži. Ove dve oznake zajedno definišu 32-bitnu adresu* tog kompjutera, koja se često zapisuje kao sekvenca četiri 8-bitna broja, razdvojena tačkama. Na primer, prethodna tekstualna adresa msa.uwgb.edu predstavlja, u stvari, adresu 143.200.128.162.

Sledeće pitanje je koji bitovi predstavljaju broj mreže, a koji lokalni ID. Odgovor zavisi od tipa adrese. IP je odavno prepoznavao nekoliko klasifikacija Internet adresa, koje uglavnom zavise od veličine mreže konkretne organizacije (tabela 11.2). Mreža Klase A koristi 8-bitni broj mreže čiji je prvi bit uvek 0.

Tabela 11.2: Klasifikacije Internet adresa

	32-bitna adresa					
Klasifikacija	**bajt 1**	**bajt 2**	**bajt 3**	**bajt 4**	**Broj mogućih mreža**	**Maksimalni broj mrežnih čvorova**
Klasa A	0nnnnnnn	xxxxxxxx	xxxxxxxx	xxxxxxxx	$2^7 = 128$	$2^{24} = 16,777,216$
Klasa B	10nnnnnn	nnnnnnnn	xxxxxxxx	xxxxxxxx	$2^{14} = 16,384$	$2^{16} = 65,536$
Klasa C	110nnnnn	nnnnnnnn	nnnnnnnn	xxxxxxxx	$2^{21} = 2,097,152$	$2^8 = 256$
Klasa D	1110, iza čega sledi 28-bitna adresa ka više odredišta					
Klasa E	1111; rezervisano					

Napomena: Sa n su predstavljeni bitovi broja mreže, a sa x bitovi lokalnog identifikatora.

* Ovo važi za IFV4. IPv6 koristi 128 bitova.

Preostala 24 bita dodeljuju se lokalno. Adrese Klase A se koriste samo za veoma velike mreže. Na primer, originalni ARPANET je imao broj mreže 10 (binarno 0000 1010). Sa 24-bitnim lokalnim identifikatorom mreža Klase A može da podrži do 2^{24} = 16.777.216 različitih čvorova. Međutim, sa sedam promenljivih bitova u broju mreže ne može da postoji više od 2^7 = 128 različitih mreža Klase A. Adrese **Klase B** koriste 16 bitova i za broj mreže (prva dva bita su uvek 10) i za lokalni identifikator. Te mreže su takođe prilično velike (do 65.536 ćvorova), ali to nije ni izbliza ravno kapacitetu mreža Klase A. Osim toga, moguće je do 2^{14} = 16.384 različitih B mreža. Konačno, adrese Klase C se koriste za reletivno male mreže. One koriste 24 bita za broj mreže (prva tri bita su uvek 110) i osam bitova za lokalni identifikator. Može da postoji do 2^{21} = 2.097.152 mreža Klase C.

Na primer, 64.37.246.3 je Internet adresa za www.nasa.gov. Pošto je 64 binarno 0100 0000, ovo je primer adrese Klase A (prvi bit je 0). Sa druge strane, server msa.uwgb.edu ima Internet adresu 143.200.128. 162, što je adresa Klase B jer je 143 =1000 1111 (osnova 2). Primer adrese Klase C je adresa 198.133.219.25, što predstavlja adresu za www.cisco.com.

Adrese **Klase D** koriste se za istovremeno slanje na više **različitih odredišta** (multicasting). Znači, jedna adresa, u stvari, defmiše grupu host kompjutera. Na primer, ako se fajl prenosi na ovakvu adresu, svaki host kompjuter u toj grupi dobija fajl. Ovo je lakše nego da se fajl eksplicitno šalje do svih članova grupe posebno. Primeri kod kojih je ovo praktično uključuju grupe kojima je neophodno ažuriranje instaliranog softvera, ili prikazivanje živog videa izabranoj grupi korisnika. Rutiranje ka više odredišta obradićemo nešto kasnije u ovom odeljku.

Besklasne adrese

IPv4 je dizajniran pre više godina i počeo je da pokazuje neke znake "starenja". Jedan problem je trošenje adresa, tj. nedovoljan broj adresa za opsluživanje globalnih zahteva. Pošto su Internet adrese 32-bitne, postoji konačan broj raspoloživih adresa. Vaša prva reakcija može da bude sledeća: 32 bita omogućavaju ukupno 2^{32}, ili oko 4,3 milijarde različitih adresa. Da, u svetu ima više ljudi od tog broja, ali u globalnoj razmeri, vedna i dalje nisu korisnici Interneta. Zato problem nije ozbiljan. Takvo razmišljanje nije ispravno, a evo zašto.

Kao što smo ranije opisali, postoje različite klase Internet adresa. Pretpostavimo da se organizacija srednje veličine prijavljuje i dobija mrežu Klase B. Možda u organizaciji radi 1.000 ljudi, tako da će biti iskorišćeno 1.000 različitih adresa.

Definisanjem mreže Klase B lokalna uprava mora da ima mogućnost dodele 2^{16} = 65.536 različitih lokalnih identifikatora. Pošto postoji samo 1.000 korisnika, oko 64.500 Internet adresa ostaje neiskorišćeno. Pošto sve imaju isti broj mreže Klase B, ne mogu da se prebace nekoj drugoj organizaciji. Ono što se ne iskoristi u ovoj konkretnoj organizaciji biva izgubljeno za opštu Internet populaciju.

Logična reakcija može da bude predlog da se koristi nekoliko mreža Klase C, umesto jedne mreže Klase B. Pošto mreže Klase C imaju 256 lokalnih identifikatora za svaku mrežu, bilo bi mnogo efikasnije dodeliti mreže Klase C. U prethodnom primeru bile bi dovoljne četiri mreže Klase C za 1.000 Internet korisnika; samo bi nekoliko adresa ostalo neiskorišćeno.

Postoji nekoliko problema kod ovakvog pristupa. Jedan je planiranje rasta. Nema menadžera mreže koji ne planira širenje mreže u budućnosti. Pretpostavimo da je kompanija tražila adrese i da je dobila jednu mrežu Klase C. Ako jednog dana na mreži bude više od 256 korisnika, moraće da se prijave za još jednu mrežu Klase C. U stvari, za svakih novih 256 korisnika mora se tražiti nova mreža. Dodatna "papirologija", "glavobolje" i kašnjenja sigumo su kontraproduktivni. Sa mrežom Klase B, moguće je povećati broj korisnika do 65.536 bez formalnih zahteva i odobrenja novih adresa (naravno, osim inicijalnog zahteva). Iz perspektive menadžera mreže, ovo je daleko jednostavniji scenario.

Sledeći problem su dodatne informacije o rutiranju. Sećate se da smo u Poglavlju 10, u odeljku o hijerarhijskom rutiranju, razmotrili razliku između unutrašnjeg i spoljašnjeg rutiranja. Na Internetu spoljašnje rutiranje se zasniva na broju mreža članica. Korišćenje mreža Klase C znači dodelu novih mreža. Više mreža podrazumeva više članova koje ruteri moraju da prate. Zato ova povećanja utiču na performanse protokola za spoljšnje rutiranje. Korišćenje mreža Klase B redukuje broj mreža koje ruteri moraju da kontrolišu, ali dobijamo neiskorišćene adrese.

Sasvim je sigurno da mora da postoji neko rešenje koje nema ozbiljne sporedne efekte. U stvari, ima ih nekoliko! Jedno je nova verzija IP-ja koja koristi više od 32 bita za adresu; predstavićemo je u narednom odeljku. Drugo rešenje je poznato kao besklasno rutiranje između domena (dassless interdomain routing), ili CIDR; standardizovao ga je IETF 1993 godine, a trenutno ga podržava BGP-4. Definiše grupu adresa koje ne pripadaju ni jednoj kategoriji unapred defmisanih klasa, već svaka adresa iz grupe može da se interpretira kao broj mreže, iza koga sledi lokalni idemifikator. U stvari, broj bitova koji definišu broj mreže može da bude različit da bi bilo omoguceno adresiranje mreža različite veličine.

CIDR se, obično, koristi za dodelu više mreža Klase C. Na primer, pretpostavimo da je za projekte neke organizacije neophodno do 1.000 različitih radnih stanica sa IP adresama. CIDR pristup nalaže dodeljivanje četiri uzastopne mreže Klase C. Adrese koje čine ove četiri mreže su susedne. Inicijalno, ovo možda ne izgleda ništa drugačije nego dodela proizvoljnih mreža Klase C, ali razmotrite sledeće moguće adrese.

Mreža Klase C	Bitska reprezentacija	Opseg adresa
211.195.8.0	11010011-11000011-00001000-xxxxxxxx	211.195.8.0 do 211.195.8.255
211.195.9.0	11010011-11000011-00001001-XXXXXXXX	211.195.9.0 do 211.195.9.255
211.195.10.0	11010011-11000011-00001010-xxxxxxxx	211.195.10.0 do 211.195.10.255
211.195.11.0	11010011-11000011-00001011-xxxxxxxx	211.195.11.0 do 211.195.11.255

U opštem slučaju ove mreže Klase C odgovaraju susednim adresama iz opsega 211.195.8.0 do 211.195.11.255. Ipak, proučite adresu malo pažljivije i videćete da su prva 22 bita ista u svim adresama. Zato bilo koju adresu iz ovih mreža klase C možemo da posmatramo kao 22-bitni broj mreže, iza koga sledi 10-bitni lokalni identifikator. Osim toga, ruter može da izvuče broj mreže (u ovom slučaju 211.195.8.0) pomoću logičke operacije I između 22-bitne maske podmreže (255.255.252.0) i IP adrese.

11010011-11000011 -000010XX-XXXXXXXX (IPadresa)

11111111-11111111-11 111100-00000000 (22-bitna maska podmreže)

1 101001 1 -1 101001 1 -00001000-00000000 (brojmreže)

211 195 8 0

Da smo koristili osam uzastopnih adresa Klase C, onda bi prvih 21 bitova u svakoj adresi bilo jednako. Šesnaest uzastopnih mreža Klase C deli prvih 20 bitova i tako redom.

Ovde smo grupisali nekoliko manjih mreža I, radi rutiranja, vizuelno ih zamislili kao jednu veću mrežu. Ovo je poznato kao formiranje supermreža (supernetting). Prednost je što, umesto da se u svakom ruteru pamti više brojeva mreža Klase C, možemo da smestimo broj samo jedne mreže. Međutim, i dalje postoje problemi rukovanja ovakvim mrežama, Kasnije ćete videti da se u svakom IP paketu nalazi IP adresa odredišta. Proučavanjem prva tri bita ruter može da utvrdi da li adresa pripada klasi A, B, ili C. Nakon toga, izvlači 8-bitni, 16-bitni, ili 24-bitni broj mreže i traži ga u tabeli rutiranja. Kako ruter utvrđuje koliko se bitova koristi za broj mreže ako je moguće koristiti različit broj bitova?

Da bi dao odgovor na to pitanje, ruter mora da zna broj bitova koji se koristi za ID mreže. Zato se uobičajena reprezentacija mrežne adrese *w.x.y.z* menja sa *w.x.y.z/m,* gde m predstavlja broj bitova u ID-u mreže. Na primer, ruter može da predstavi četiri prethodno pomenute mreže sa jednim zapisom 211.195.8.0/22, gde "/22" ukazuje da se za broj mreže koriste 22 bita. Kada stigne IP paket, ruter pokušava da uporedi odgovarajuci broj bitova iz tabele rutiranja sa istim brojern bitova u odredišnoj adresi paketa. Ako se poklapaju, paket se prosleđuje do odgovarajuće lokacije.

Dobijanje adrese

Sledeće logično pitanje je kako dobiti IP adresu. Postoji nekoliko mogućih odgovora. Na primer, ako imate personalni kompjuter povezan na LAN Vaše kompanije, ili Vašeg univerziteta, ili ako imate kompjuter kod kuće koji na Internet povezujete preko ISP-ja, već imate IP adresu. ISP, ili menadžer LAN-a održavaju listu IP adresa koje se dodeljuju klijentskim mašinama koje povezuju. Svaki put kada se ulogujete na LAN, ili ISP, obično klijent mašina zahteva IP adresu od servera. Server izvršava protokol poznat pod nazivom Dynamic Host Configuration Protocol (DHCP), koji dodeljuje IP adresu iz liste koju održava. Postoji nekoliko načina da utvrdite IP adresu koju Vaš kompjuter koristi. Ako koristite operativni sistem Windows, možete da predete u MS-DOS prompt iz menija Programs i unesite komandu ipconfig. Dobićete rezultat sličan sledećem:

1 Ethernet Adapter:
IPAddress.....24.163.137.90
Subnet Mask.....255.255.252.0
Default Gateway.....24.163.136.1

Osim toga, možete da selektujete komandu Windows Run i unesite winipcfg. Dobićete sličan odziv.

Sledeće pitanje je kako menadžer LAN-a, ili ISP dobija te adrese. Odgovor je malo kompliko-vaniji. Kao što ISP, ili menadžer LAN-a održavaju blokove adresa, tako mora da postoji sličan menadžer koji vodi računa o svim mogućim IP adresama. Pre 1999. godine, **Internet** Assigned Numbers **Authority (IANA),** organizacija koju sponzoriše država i koja je pridružena Information Sciences institutu, u University of Southern California, bila je odgovoma za upravljanje i dodelu IP adresa. Tu i neke druge tehničke funkcije koje se tiču DNS-a (biće opisan kasnije) preuzela je 1999. godine neprofitna organizacija **Internet Corporation for** Assigned Names **and** Numbers (ICANN). Njen upravni odbor uključuje ljude iz različitih kompanija i akademskih institucija širom sveta. Više informacija o ovoj organizaciji možete da pronadete na Web sajtu www.icann.org.

Pošto većina ljudi ne voli da pamti IP adrese, sledecl logičan korak je dobijanje registrovanog naziva hosta. To znači da se naziv hosta pamti u distribuiranom direktorijumu koji može da se referencira pomoću klijent programa. Na primer, mnogi ljudi znaju da mogu da pristupe pretraživačkoj Web mašini pomoću http://www.google.com. Manji broj njih zna da je pristup mogući sa adresom http://216.239.53.99. Većina će se složiti da je mnogo lakše zapamtiti naziv hosta nego niz brojeva.

Dovoljno je to što moramo da pamtimo brojeve telefona, čekovnih knjižica, računa u bankama, automobilskih tablica i matične brojeve. Kome je potrebno još brojeva za pamćenje? Stvami proces registracije naziva domena odvija se preko registratora. Obično se selektuje registrator, popune se formulari i uplati se potrebni iznos. Sve ostalo završava registrator.

Uz pretpostavku da ste izabrali naziv koji još uvek niko drugi nije zakupio, ostali korisnici Interneta će moći da pristupe Vašem sajtu. Ipak, morate da budete pažljivi kada birate nazive, jer je bilo slučajeva kada su podnošene tužbe protiv ljudi koji su koristili iste (ili čak samo slične) nazive, kao što su zaštićene marke. Postoje različiti registratori koji izvršavaju registraciju naziva hosta. Lista ICANN-akreditovanih registratora može da se pronade na www.icann.org/registrars/.

Domain Name System

Pošto smo opisali i tekstualni oblik adresa i 32-bitne adrese, sledeće logićno pitanje je kako se vrši prevođenje iz jednog oblika u drugi. Svaki sajt izvršava protokol koji pristupa distribuiranoj bazi podataka poznatoj pod nazivom Domain Name System (DNS). Operativne reči su *distribuirana* i *baza podataka.* Kao baza podataka, DNS sadrži kopije tekstualnih adresa i njihove 32-bitne reprezentacije. Na primer, sadrži tekstualnu adresu msa.uwgb.edu sa pridruženom Intemet adresom 143.200.128.162.

DNS se ne čuva na bilo kojoj lokaciji. Kad bi se to radilo, postojala bi potencijalna opasnost od katastrofalnih posledica ako bi došlo do kvara na toj lokaciji. Osim toga, ne bi bilo moguće kontrolisati velike količine zahteva iz celog sveta na jednoj lokaciji. Umesto toga, DNS se distribuira između kolekcije *DNS servera* koji su razbacani na Internetu. Kada je host kompjuteru potrebno prevođenje adrese, on poziva jedan, ili više tih servera da pronadu specifičnu tekstualnu adresu i da vrate njenu Internet adresu. U stvari, ako imate pristup UNIX, ili Linux sistemu, možete da testirate DNS prevođenje pomoću komande nslookup. Na primer,

unošenjem komande nslookup msa.uwgb. edu* dobijate 32-bitnu adresu 143.200.128.162. Isprobajte i neke druge tekstualne adrese.

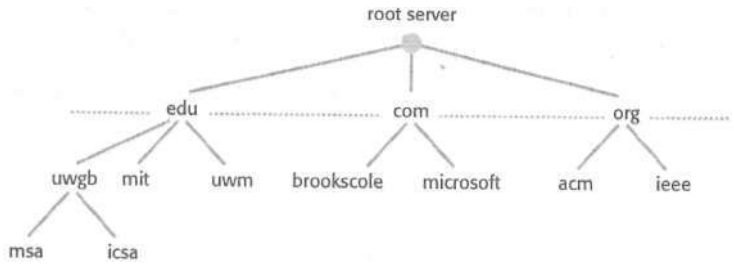
Izgleda jednostavno; mi smo pojednostavili celu situaciju. Ipak, osnovna ideja traženja adrese je tačna. Teži deo je implementacija - upravljanje milionima adresa na različitim serverima i obezbeđivanje brze translacije tekstualne adrese. Ovde nastupa koncept domena. DNS možemo da smatramo hijerarhijskim uređenjem tekstualnih adresa koje su uređene najpre po nazivima domena. Na slici 11.3 prikazano je kako su neke tekstualne adrese organizovane.

Na primer, grupisane su sve "edu" institucije, kao i "com" institucije i tako redom. Ipak, edu domen je i dalje veliki, tako da se organizuje u podgrupe na drugom nivou hijerarhije. Univerziteti kao što su UWGB (University of Wisconsin - Green Bay), UWM (University of Wisconsin - Milwaukee) i MIT grupisani su ispod edu domena. Slično tome, postoje podgrupe i ispod com, org i drugih domena. Postoje dodatne hijerarhije ukoliko je reč o velikim institucijama koje se dalje dele na odeljenja.

Istakli smo da su tekstualne adrese organizovane hijerarhijski, ali to ne znači i da se smeštaju na ovaj način. Već smo pomenuli postojanje servera širom Interneta i istakli da ni jedan server ne sadrži sve informacije iz DNS-a. Informacije koje su predstavljene u hijerarhiji na slici 11.3 dele se na *zone,* koje predstavljaju hijerarhije sa jednim, ili više čvorova.

Osim toga, dve zone ne mogu da se preklapaju. Svaka zona uključuje najmanje dva servera (primarni i rezervni) - svaki je odgovoran za upravljanje informacijama iz tog dela hijerarhije. Zato je možda tačnije reć da se hijerarhija primenjuje na organizaciju servera.

Na vrhu hijerarhije je root server. Iako je na slici 11.3 prikazan samo jedan, postoji, u stvari, nekoliko root servera smeštenih širom sveta.



SLIKA 11.3 *DNS hijerarhija*

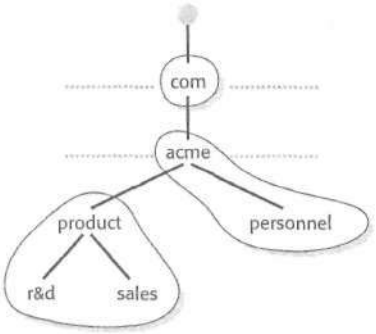
* U najskorijoj verziji Red Hat Linuxa nslookup se ne odobrava. Znači, korisnici se ohrabruju da koriste komande dig, ili host, koje imaju sličnu funkciju.

Svaki root server zna lokaciju ostalih DNS servera koji su dodeljeni specifičnim domenima. Ti serveri lociraju neke naredne DNS servere. Dok napredujemo kroz hijerarhiju, pojavljuje se sve više servera koji su specifični za određenu lokaciju. Teško je predvideti kako će se rukovati zahtevima, jer mnogo čega zavisi od onoga šta serveri znaju, ko im postavlja zahtev, gde se traženi sajt nalazi i gde se serveri nalaze u okviru hijerarhije. Ipak, postoji jedna mogućnost.

Kada host zahteva translaciju adrese, šalje zahtev do lokalnog servera (na istom mestu kao i host). Nazovimo ga A. Ako A može da obezbedi translađju, on to i radi i proces je završen. Ako ne može, onda šalje zahtev do servera B, koji se nalazi na prvom sledećem nivou iznad njega u hijerarhiji. Ako B zna adresu, šalje je nazad do A, koji je, zatim, prenosi do hosta. Translacija je završena. Medutim, možda ni B ne zna adresu. U tom slučaju, on može da nastavi dalje na dva načina. Može da obavesti A da nije uspeo da izvrši translaciju i da uputi A na server C, koji možda zna adresu. Druga opcija, koju ćemo uvek koristiti u narednom razmatranju, podrazumeva da B šalje zahtev do C u ime A. C obično odgovara serveru na sledećem višem nivou u hijerarhiji. Server C dalje nastavlja na isti način kao i B.

Na kraju, biće pronaden server koji zna adresu, ili se zahtev prosleduje sve do root servera. Ipak, ne postoji način da se obezbedi način na koji bi root server mogao da zna sve moguće adrese. Ako dobije zahtev za adresu koju ne može da obezbedi, on ne zna za neki drugi server koji bi mogao da mu pomogne. U zavisnosti od domena sa koga je potekao originalni zahtev, root server prosleđuje zahtev do jednog od nekoliko DNS servera koji su postavljeni na sledećem nivou hijerarhije - nazvaćemo ga server D. Na primer, tekstualne adrese koje se završavaju na *com* i *edu* idu na različite servere. Ako D ima pristup informacijama koje su predstavljene naznačenom tekstualnom adresom, Internet adresa se vraća do root servera. Na kraju se informacije vraćaju nazad do hosta preko C, B i A. Ako nema adrese, D zna za neki drugi server koji bi trebalo da obezbedi dodatne informacije. Ovo prosleđivanje zahteva se nastavlja sve dok se ne stigne do servera koji može da izvrši translaciju. Kada se adresa pronađe, vraća se nazad do originalnog hosta preko svih servera (suprotnim redosledom) koji su dobili i prosledili taj zahtev.

Na slici 11.4 prikazano je kako ovo funkcioniše u jednom mogućem slučaju; prikazan je samo manji ogranak com domena.



SLIKA 11.4 *Zone u DNS hijerarhiji*

Kompanija ACME održava dve mreže - po jednu za proizvodno (product) i personalno (personnel) odeljenje. Proizvodno odeljenje se sastoji iz dva dela - istraživačkog i razvojnog (research and development) i prodaje (sales). Krugovi oko čvorova definišu zone.

Pretpostavimo da neko spolja naznači tekstualnu adresu hercules.products.acme.com; "hercules" je kompjuter u istraživačkom i razvojnom delu proizvodnog odeljenja kompanije ACME. Pretpostavimo da adresa ne može da se prevede lokalno i da zahtev ide do root servera, pa do servera domena com. Server domena com ne može da zna punu adresu svih host kompjutera u svim komercijalnim mrežama (postoji ogroman broj takvih mreža). Medutim, "acme" deo tekstualne adrese ukazuje serveru da drugi server u ACME može da obezbedi više informacija. Tekstualna adresa se zato šalje do ACME sajta.

Moguće je da je personalno odeljenje prilično malo, a da je proizvodno odeljenje znatno veće; zato su nadležni u upravi odlučili da se za proizvodnu mrežu koristi zaseban server. Možda su menadžeri mreže u proizvodnom odeljenju jednostavno hteli da budu potpuno nezavisni od personalne mreže. U svakom slučaju, server u ACME zoni i dalje ne može da razreši punu adresu, jer tekstualna ukazuje na zonu za koju nije odgovoran. Napomenimo da bi server, ako bi u adresi stajalo "personnel.acme.com", mogao da je razreši, ali samo zato što se i ACME i personalni čvorovi nalaze u istoj zoni. Međutim, pošto se proizvodni čvor nalazi u drugoj zoni, zahtev je poslat do drugog servera u proizvodnoj zoni. Od te tačke preostali deo tekstualne adrese se nalazi u zoni odgovornosti tog servera, koji pristupa, pa vrača odgovarajuću Internet adresu.

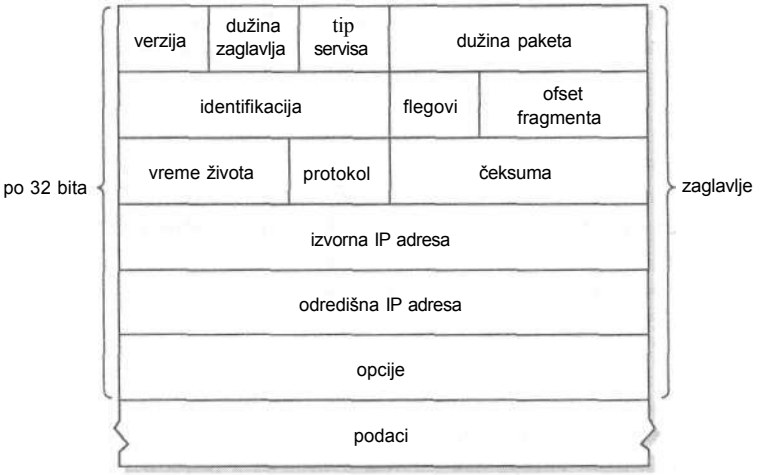
Da li ovaj složeni proces mora da se izvede prilikom svake translacije? Srečom, nije nužno. Kada se adrese prevedu i rezultati vrate nazad preko imenovanih servera, svaki može da zabeleži adresu u svom lokalnom kešu. Zahvaljujući tome, u slučaju nekog budućeg zahteva, server može da obezbedi translaciju iz keša.

IP paketi

Pošto ste upoznali osnove lnternet terminologije i operacija, predimo na razmatranje formata paketa i nekih specifičnih karakteristika lnternet Protocola. IP paket je po mnogo čemu sličan ranije predstavljenim formatima paketa i okvira. Ipak, postoje neke jedinstvene karakteristike. Na slici 11.5 prikazan je sadržaj IP paketa.

Sledi lista objašnjenja polja u IP paketu.

- Version (Verzija) Definiše verziju IP-ja koji je kreirao paket. Na ovaj način je omogućeno zajedničko funkcionisanje različitih verzija IP-ja.
- Header Length (Dužina zaglavlja) Definiše broj 32-bitnih reči u zaglavlju paketa (polja koja prethode podacima).
- Type of Service (Tip servisa) Ovo 8-bitno polje je dizajnirano da bi defmisalo zahteve transportnog sloja u vezi načina rukovanja paketom. Dopušta različite opcije zahteva, kao što su Precedence (Prioritet), Lovv delay (MaIo kašnjenje), High throughput (Visoka propusnost), i High reliability (Visoka pouzdanost). 3-bitno polje Precedence omogućava defmisanje prioriteta za pakete (O za niski i 7 za visoki prioritet); posebno je korisno kada je potrebno ispuniti određeni kvalitet servisa (QoS).



SLIKA 11.5 *Intemet paket*

Transportni protokol je mogao da zahteva prenos sa niskim kašnjenjem *(low-delay*), što je korisno ako se transportni korisnik ulogovao na udaljeni kompjuter i želi brz odziv. Zahtev za visoku propusnost *(high-throughput)* je koristan kod prenosa velikih fajlova, a zahtev sa visoku pouzdanost *(high-reliability)* traži mreže koje je zapamtio kao mreže koje nude pouzdanije servise.

Skoriji QoS problemi su rešeni pomoću složenih protokola, kao što su RSVP i RTP. Osim toga, IETF je definisao **Differentiated Services** arhitekturu (poznata je i kao **Diffserv)** za diferenciranje različitih tipova saobraćaja koji se odvijaju preko Interneta. Ruteri koji implementiraju Diffserv arhitekturu koriste 8-bitno Differentiated Services (DS) polje, koje služi za označavanje paketa sa različitim QoS tretmanom. Da bi arhitektura bila konzistentna sa tekućim IP protokolom, koristi polje Type of Service kao DS polje.

Šest krajnjih levih bitova definišu kodnu tačku koja, teorijski, može da obezbedi 64 različite klase saobraćaja. Trenutno se ne koriste dva krajnja desna bita. Osnovna ideja je da se paket, kada se postavi na Internet, označava odredenom DS vrednošdu. Ruteri koji dobiju paket proučavaju DS polje i mogu da tretiraju paket na osnovu polisa koje je proizvodač implementirao u svojim ruterima.

- **Packet Length (Dužina paketa)** Definiše dužinu celog IP paketa. To je 16-bitno polje i zato obezbeđuje maksimalnu dužinu od 65.535 bajtova.
- **Identification, Flags, Fragment Offset (Identifikacija, flegovi i ofset fragmenta)** Ova tri polja se koriste za fragmentaciju.
- **Time to Live (Vreme života)** Stanica koja šalje paket na Internet prvi put može da postavi polje Time to Live tako da se definiše maksimalno vreme koje paket može da provede na Internetu. Kada drugi ruter primi paket, dekrementira polje Time to Live za period koji je paket proveo u ruteru i šalje paket do sledećeg rutera. Ako vrednost polja Time to Live postane O, ili manja, ruter odbacuje paket i šalje poruku o grešci do stanice koja ga je poslala. Ovaj korak garantuje da problemi sa rutiranjem, ili zagušenjem neće izazvati beskonačno kruženje paketa po Internetu.

- **Protocol (Protokol)** Definiše protokol višeg sloja koji koristi IP. Omogućava prenos podataka odredišnog IP-ja do odgovarajućeg entiteta na toj strani. Na primer, ako IP paket sadrži TCP segment, vrednost polja Protocol iznosi 6. Paketi koji sadrže UDP, ili ICMP segmente u polju Protocol imaju vrednosti 17, ili 1, respektivno. IGMP (protokol koji se koristi za slanje paketa na više adresa istovremeno) koristi vrednost 2, a RSVP (protokol koji garantuje određeni kvalitet servisa) 46.

- **Checksum** (Čeksuma) Koristi se za detekciju grešaka u zaglavljima paketa. Pošto podaci odgovaraju segmentu TCP-ja, ili drugog protokola, postoji sopstvena detekcija grešaka, koja se izvodi na višem sloju. Tako IP treba da vodi računa samo o detektovanju grešaka u zaglavlju paketa. Prednost ovoga je u tome što se sa proverom manjeg broja bitova u svakom ruteru paketi brže obrađuju. Da bi se izračunala čeksuma, zaglavlje se interpretira kao sekvenca 16-bitnih celih brojeva. Vrednosti se sabiraju korišćenjem aritmetike komplementa 1, a rezultat se komplementira i smešta u polje Checksum. Na prijemnoj strani vrši se ponovno izračunavanje čeksume na osnovu pristiglih informacija. Ako se dobijeni rezultat razlikuje od vrednosti iz polja Checksum, primalac zna da je došlo do greške u zaglavlju. Napomenimo da čeksuma mora da se izračunava prilikom svakog prenosa, jer se zaglavlje menja (menja se vrednost polja Time to Live).

- **Source IP Address** (Izvorna **IP** adresa) **i Destination IP Address (Odredišna IP adresa)** Ova polja sadrže adrese predajne i prijemne strane.

- **Options (Opcije)** Ovo polje nije obavezno u svakom paketu, ali može da se koristi za slanje zahteva za specijalno tretiranje paketa. Sadrži niz zapisa, od kojih svaki odgovara zahtevanoj opciji.

  Opcija *Record Route* prati rutu kojom paket putuje. Stanica koja šalje paket rezerviše prostor za listu IP adresa u polju Options. Svaki ruter koji rutira paket umeće svoju adresu u ovu listu, na osnovu čega prijemna stanica može da utvrdi ko je upravljao paketom.

  Opcija *Timestamp* je slična opciji Record Route. Osim što smešta svoju adresu, svaki ruter beleži i vreme kada je rutirao paket.

  Opcija *Source Route* omogućava pošiljaocu da naznači rutu kojom treba proslediti paket navođenjem sekvence IP adresa u polju Options. Svi ruteri koristi ove informacije umesto sopstvenih tabela rutiranja. Ovo nije normalan način funkcionisanja prilikom rutiranja, zato što je neophodno poznavanje fizičke topologije. Može da bude korisno ako administratori mreže sumnjaju da postoji problem sa nekim ruterom i ako žele da testiraju određenu rutu.

  Opcija *Loose Source Route* ne obezbeđuje tačnu rutu, ali daje listu rutera preko kojih paket mora da prođe.

  Opcija *Security* može da navede određene rutere, ili lokacije koje bi trebalo izbeći u toku rutiranja.

  Više detalja o ovim opcijama i njihovom izvršenju možete da nađete u referenci [CoOO].

- **Data (Podaci)** Sadrži podatke koje je neophodno obezbediti sledećem višem sloju.

Fragmentacija

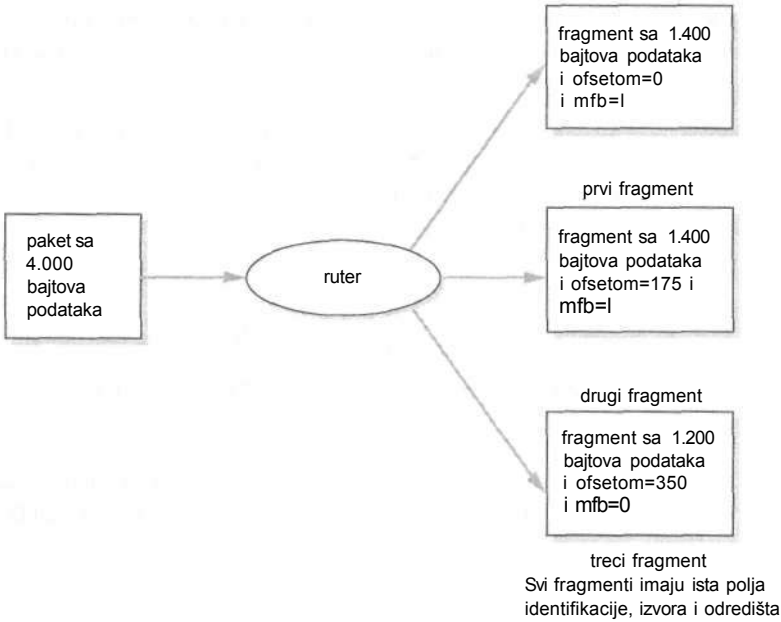
Jedan od problema sa kojim se Internet Protocol suočava je da različite mrežne arhiteklure dopuštaju razlidte veličine okvira (još se nazivaju i maksimalne jedinice prenosa—maximum transfer units, ili MTU). Ako je dužina IP paketa manja od svakog MTU-a na koji se nailazi na putanji, nema problema, ali, ako je MTU manji, paket se deli na manje jedinice, koje se nazivaju i fragmenti. Fragmenti putuju do svojih konačnih odredišta (moguće je preko različitih ruta), gde se moraju ponovo sastaviti. Da bi fragmentacija funkcionisala, odredišni IP mora da bude sposoban da razlikuje fragmente od nefragmentisanih paketa i da prepozna koji fragmenti odgovaraju islom paketu, kojim se redosledom moraju sastaviti i koliko se fragmenata nalazi u svakom paketu. Polja Identification, Flags i Fragment Offset obezbeduju te informacije.

Pretpostavimo da ruter primi paket i da utvrdi da mora da putuje preko mreže sa MTU-om manjim od dužine paketa. Paket se deli na fragmente; svaki od njih sadrži deo podataka iz paketa. Osim toga, svaki fragment ima zaglavlje fragmenta (fragment header), koje je skoro identično zaglavlju paketa i omogućava rutiranje fragmenta. Mnoga polja u zaglavlju fragmenta imaju istu ulogu kao i odgovarajuća polja u zaglavlju paketa. Sledeća polja su relevantna za naše razmatranje:

- Ruter postavlja vrednost identifikacije paketa u polje *Identification* svakog fragmenta.
- Polje *Flag* sadrži *More Fragments bit* (mfb). Ruter postavlja mfb na 1 u svakom fragmentu, osim u poslednjem. Postoji i *Do Not Fragment* bit, koji, ako je postavljen, ne dopušta fragmentaciju. Ako ruter primi takav paket, odbacuje ga i šalje poruku o grešci do stanice koja ga je poslala. Stanica koja je poslala paket koristi tu poruku kako bi utvrdila granične vrednosti kod kojih dolazi do fragmentacije. Ako je tekući paket suviše veliki, pošiljalac može da ponovi prenos sa manjim paketima da bi se eventualno utvrdilo kada dolazi do fragmentacije.
- Pošto se u fragmentu nalaze delovi podataka iz paketa, ruter utvrđuje i ofset u polju Data paketa iz koga su podaci izvučeni i smešta ga u polje *Fragment Offset.* Ofsete meri u jedinicama od po osam bajtova. Tako ofset I odgovara bajtu pod brojem 8, ofset 2 bajtu 16 i tako redom.

Na slici 11.6 prikazan je paket koji je podeljen na tri fragmenta. Pretpostavljeno je da mreža ima MTU-e koji dopuštaju najviše 1.400 bajtova podataka. Zato ruter deli dolazeći paket sa 4.000 bajtova podataka na tri fragmenta. Prva dva fragmenta imaju po 1.400 bajtova podataka. Polje Fragment Offset u prvom fragmentu je 0, čime se ukazuje da podaci počinju sa ofsetom 0 u originalnom paketu. Polje Fragment Offset u drugom fragmentu uma vrednost 175, čime se ukazuje da njegovi podaci počinju od bajta 1.400 (8x175) u paketu. Treći fragment ima 1.200 bajtova podataka i ofset je *350.* More Fragments bitovi u prva dva fragmenta su I, čime se ukazuje da iza svakog od njih postoji još fragmenata. Mfb u poslednjem fragmentu je 0, što znači da je to poslednji fragment. Zaključak da je reč o fragmentu izveden je na osnovu vrednosti polja Offset.

Kada odredišni IP vidi dva različita fragmenta sa istom identifikacijom, izvornom i odredišnom adresom, on zna da potiču iz istog paketa. Ponovo sastavlja takve pakete u skladu sa redosledom koji odreduju vrednosti polja Offset.



SLIKA 11.6 *Fragmentacija paketa*

Poslednji fragment prepoznaje po tome što ima mfb jednako 0 i nenulti ofset. U okviru procesa sastavljanja postavlja i **tajmer** ponovnog sastavljanja (reassembly timer) čim stigne prvi fragment. Ako se svi fragmenti ne prime pre nego što tajmer istekne, pretpostavlja se da je jedan, ili više fragmenata izgubljeno. U tom slučaju se odbacuju svi zadržani fragmenti i šalje se poruka o grešci do stanice koja je poslala fragmente. Dakle, prijem fragmenata se vrši po principu "sve, ili ništa".

IP rutiranje

Razmatranju rutiranja na Internetu je prethodilo detaljno objašnjavanje osnova. U stvari, većina ideja je već bila predstavljena u prethodnom poglavlju. IP rutiranje se zasniva na tabelama rutiranja koje se čuvaju na ruterima i na interpretaciji IP adresa, a umnogome se oslanja na RIP-2 i BGP protokole. U prethodnim odeljcima su već predstavljene te teme, tako da nema potrebe da ih ponovo navodimo. Ipak, postoji nekoliko detalja u vezi rutiranja koji se tiču Interneta. Sećate se da smo rekli da email adresa korisnika Interneta može da bude

korisnik@host.odeljenje.institucija.domen

DNS serveri prevode karaktere iza simbola @ u 32-bitni broj, koji se, obično, izražava kao sekvenca četiri 8-bitna broja, razdvojena decimalnim tačkama (tačkasta notacija). Na primer, IP adresa host kompjutera naznačena u shayw@msa.uwgb.edu odgovara 32-bitnom broju 10001111-11001000-10000000-00111110, ili, koristeći ekvivalent tačkaste notacije, 143.200.128.62.

Problem kojim se još uvek nismo bavili jeste razlika između IP adrese i fizičke adrese. jedinstvena 32-bitna IP adresa se dodeljuje svakom kompjuteru kada pristupi Internetu. Njegova *fizička adresa* je adresa koja se koristi za osnovnu fizičku mrežu.

Na primer, uredaji koji su povezani na Ethernet "osluškuju" adrese smeštene u IEEE 802.3 okvirima da bi utvrdili koji je okvir namenjen za njih. Međutim, to su Ethernet adrese (48-bitni brojevi koji su dodeljeni na osnovu mrežne kartice). One imaju lokalno značenje i nemaju smisla na globalnoj IP skali. Kako će takav uređaj prepoznati paket sa IP adresom?

Odgovor je da neće prepoznati. Sećate se sa slike 11.2 da se IP paketi (IP adresa i sve ostalo) smeštaju u okvire ako "putuju" preko LAN-a. Okviri sadrže adrese koje zavise od protokola za kontrolu veze između podataka. Unutar LAN-a adresa okvira definiše njegovo odredište.

Ako okvir "putuje" preko rutera, IP u ruteru rastavlja paket, proučava adresu i utvrduje gde treba poslati paket. Sa druge strane, ako ruter treba da pošalje paket do personalnog kompjutera koji je povezan na LAN, on mora da postavi paket u LAN okvir, a zatim ga šalje. Ako se paket ugraduje u LAN okvir, koju fizičku adresu koristi? Drugim rećima, kako ruter utvrduje fizičku adresu konkretne IP adrese? Postoji nekoliko načina da se to izvede, u zavisnosti od specifičnosti nižih slojeva.

Kada ruter primi IP paket, postoje dve mogućnosti: ili se odredište paketa pridružuje mreži na koju je ruter priključen, ili se ne pridružuje. Ruter prepoznaje prvu opciju, jer prvi deo svake IP adrese defmiše mrežu na kojoj se nalazi odredišni uređaj. Ako prepozna mrežu na koju je i sam priključen, zna da može da pošalje paket direktno do njegovog odredišta. Ovo se naziva *direktno rutiranje.* Postavlja fizičku adresu odredišta u okvir i šalje ga do odredišta. Ponovo se postavlja pitanje kako ruter utvrduje fizičku adresu na osnovu konkretne IP adrese.

Kod jednog mogućeg pristupa, dinamičkog povezivanja (još je poznato i kao Address Resolution Protocol), ruter prenosi emisioni Request okvir sa IP adresom do svih stanica na LAN-u. Kada primi emisioni zahtev, uređaj sa konkretnom IP adresom reaguje slanjem svoje fizičke adrese nazad do rutera, a ruter je beleži zajedno sa njenom IP adresom u svoj lokalni keš. Nakon toga, šalje okvir do odgovarajućeg uređaja, koristeći upravo pribavljenu fizićku adresu. Smeštanjem IP i fizičke adrese u lokalni keš ruter može da izbegne ponovno slanje emisionog Request okvira ako ponovo bude morao da pošalje neki paket do istog uredaja.

Ovo važi bar za određeno vreme, zato što se keš periodično prazni. Razlog za pražnjenje keša je što se mora obezbediti tačnost tekućih informacija u kešu. Naravno, nameće se logično pitanje kako informacije mogu da postanu netačne.

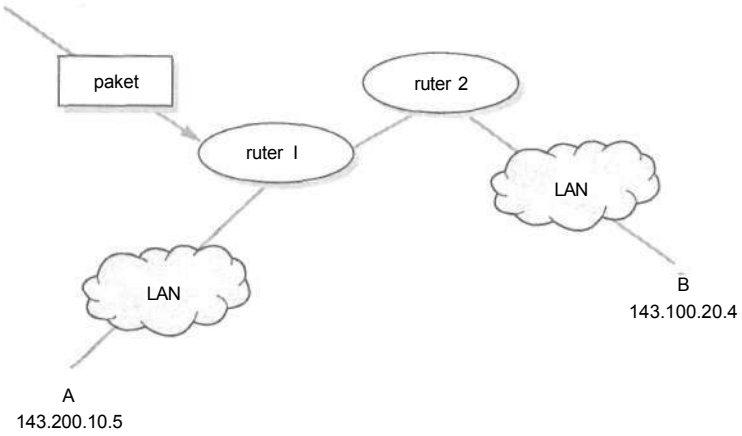
Zar fizička adresa kompjutera nije konstantna? Odgovor je negativan. Ako se Ethernet kartica u personalnom kompjuteru pokvari, neophodno je instalirati novu. Pošto svaka fizička kartica ima jedinstvenu adresu, sve informacije koje se tiču IP adrese kompjutera i stare fizičke adrese više nisu validne.

Zatim, pretpostavimo da odredište više nije dostupno preko neke mreže na koju je mter priključen. U tom slučaju ruter koristi hijerarhijsko rutiranje kako bi utvrdio do kojeg drugog rutera može da pošalje paket. Paket, nakon toga, "putuje" do drugog rutera sve dok ne dođe do rutera koji je priključen na odredišnu mrežu i proces se nastavlja kao što je ranije opisano.

Pogledajmo jedan primer kako bismo razjasnili ceo proces. Slede koraci rutera:

1. Prijem IP paketa i izvlačenje IP adrese
2. Ako je opcija paketa Source Route označena, paket se rutira u skladu sa naznačenom rutom. U tom slučaju "preskočite" naredne korake.
3. Utvrđuje se broj mreže na osnovu IP adrese.
4. Da li broj mreže odgovara bilo kojoj mreži na koju je ruter povezan?
5. Ako odgovara, utvrduje se fizička adresa odredišta (bilo traženjem, ili dinamičkim povezivanjem) i do odredišta se šalje okvir koji sadrži IP paket.
6. Ako ne odgovara, broj mreže se traži u tabeli rutiranja i paket se prosleduje do naznačenog rutera.
7. Ako zbog nečega mreža nije navedena u tabeli rutiranja, paket se prosleduje do podrazumevanog rutera.

Uzmimo za primer topologiju sa slike 11.7. Ruter I je povezan na mrežu pod brojem 143.200 (svaki čvor te mreže ima IP adresu 143.200.x.y). Slično tome, ruter 2 je povezan na mrežu pod brojem 143.100. Pretpostavimo da ruter I primi paket sa IP adresom 143.200.10.5. Proučava broj mreže 143.200 i zna da može direktno da isporud paket do odredišta. Šalje emisioni Request okvir sa brojem 143.200.10.5. Pošto je reč o emisionom okviru, svi uređaji na LAN-u dobijaju taj okvir, ali samo uređaj A reaguje vraćanjem okvira do rutera sa svojom fizičkom adresom.



Nakon toga, ruter smešta IP paket u okvir u kome se nalazi fizička adresa uređaja A i šalje taj okvir.

Sa druge strane, pretpostavimo da ruter 1 primi paket sa adresom 143.100.20.4. Ruter I zna da nije povezan na mrežu 143.100 i da mora da rutira paket do drugog rutera. Proučava svoju tabelu rutiranja i otkriva da paket treba proslediti do rutera 2. Kada dobije paket, ruter 2 izvršava zadatke slične onima koje je izvršio ruter 1. Naravno, paket može da prode kroz nekoliko rutera, ali koraci su slični u svim ruterima.

Da li ste se nekada zapitali kojom rutom "putuje" email poruka koju pošaljete. Postoji način da se utvrdi ruta koja se koristi do odredenog odredišta.

Na UNIX i Linux sistemima komanda `traceroute` prikazuje rutu (prelazne rutere) izmedu tog sistema i naznačenog odredišta. Na Red Hat Linuxu možete da unesete sledeću komandu

```
/usr/sbin/traceroute www.alaska.org
```

Naravno, za odredište možete da unesete bilo šta drugo. U odzivu se prikazuju IP adrese rutera koje je paket prošao i vreme koje je bilo potrebno da se paket pošalje do odredišta i vrati nazad. Izdavanjem ove komande sa Linux servera u Green Bay, Wisconsin, dobija se sledeći rezultat.

```
1  143.200.128.1 (143.200.128.1) 0.492 ms 0.459 HLS 0.426 IHS
2  fwallal (143.200.208.11) 19.270 ms 30.757 ms 31.306 ms
3  C7204vxr (143.200.225.4) 62.370 ms 62.225 IHS 62.236 ms
4  uwgreenbay-atm6-0-252.core.wiscnet.net (216.56.24.17)
   46.954 ms 52.289 ms 56.635 ms
5  uwmilwaukee-atml-0-4.core.wiscnet.net (140.189.8.217)
   52.952 ms 55.939 ms 53.458 ms
6  p7-2.chcgil1-cr4.bbnplanet.net (4.24.164.101) 55.849 ms
   54.659 ms 54.359 ms
7  p6-0.chcgil1-br2.bbnplanet.net (4.24.5.245) 53.241 ms
   55.618 ms 62.713 ms
8  so-3-0-0.chcgil2-br2.bbnplanet.net (4.0.1.197) 77.895 ms
   62.004 ms 54.161 ms
9  so-7-0-0.chcgil3-hcr1.bbnplanet.net (4.0.1.185) 55.233ms
   62.002 ms 46.912 ms
10 pl-3.xchcgil4-uunet.bbnplanet.net (4.24.95.10) 68.787 ms
   71 .272 IHS 63.828 ms
11 0.so-5-0-0.x12.chil3.alter.net (152.63.73.21) 76.797 ms
   69.014 ms 71.139 ms
12 0.so-2-2-0.x12.chi2.alter.net (152.63.70.106) 62.488ms
   70.069 ms 54.447 IHS
13 0.SO-1-0-0.t12.chi2.alter.net (152.63.67.121) 70.295 ms
   54.236 ms 62.613 ins
14 0.so-5-3-0.t12.sea1 .alter.net (152.63.136.62) 109.052ms
   109.310 ms 109.175 ms
15 0.so-1-0-0.x12.seal.alter.net (152.63.2.133) 109.536ms
   108.712 ms 109.495 ms
```

```
16 pos5-0.xr2.seal.alter.net (152.63.106.234) 93.678 ms
   108.724 ms 118.475 ms
17 194.atm7-0.gw8.seal.alter.net (152.63.105.221) 100.296 ms
   115.843 ms 102.321 ms
18 gci-gw.customer.alter.net (157.130.182.6) 102.076 ms
   116.190 ms 125.023 ms
19 inetseasdcgw-2.gci .net (209.165.129.34) 117.285 ms 116.636
   ms 124.990 ms
20 208.155.87.121 (208.155.87.121) 156.240 ms 140.107 ms
   157.259 ms
21 209.165.128.1 (209.165.128.1) 155.268 ms 154.801 ms
   156.259 ms
22 209.165.128.82 (209.165.128.82) 156.261 ms 152.754 ms
   156.600 ms
23 205.140.73.143 (205.140.73.143) 187.115 ms 155.779 ms
   131.571 ms
```

Verovatno pogađate lokacije nekih od ovih sajtova na osnovu naziva koje prikazuju. U ovom konkretnom slučaju paket izgleda "putuje" od Green Baya, preko Milvokija i Čikaga, do Sijetla. Nazivi hostova koji se završavaju sa *bbnplanet.net* odgovaraju Genuity, Inc, provajderu IP mrežnih servisa. Naziv *alter.net* odgovara UUNET, WorldCom kompaniji, a *gci.net* odgovara Alaska-based kompaniji, koja obezbeduje servise za razmenu podataka za različite potrošače. Ponekad se nazivi ne navode zato što neke mreže ne obezbeđuju prevođenje adresa u nazive. Ipak, dobijate neku predstavn o tome kako ruta izgleda.

Ako nemate pristup Linux mašini, možete da isprobate DOS komandu `tracert`. Kao sledeći primer, nakon unošenja komande `tracert` www.alaska.org sa druge radne stanice u Green Bayu, Wisconsin, dobili smo sledeće:

```
1   8 ms     10 ms     7 ms     10.42.32.1
2   9 ms     8 ms      14 ms    srp-3-0.applwi2-rtr1.new.rr.com
                                 [24.164.224.24]
3   11 ms    8 ms      8 ms     srp-0-0.applwil-rtr1.new.rr.com
                                 [24.164.224.17]
4   9 ms     13 ms     9 ms     srp-4-0-0.applwi3.rtr1.new.rr.com
                                 [24.164.224.91]
5   12 ms    15 ms     17 ms    pop2-chi-P8-0.atdn.net
                                 [66.185.141.109]
6   17 ms    13 ms     13 ms    bb1-chi-P0-2.atdn.net
                                 [66.185.148.70]
7   29 ms    36 ms     26 ms    bb1-kcy-P7-0.atdn.net
                                 [66.185.152.125]
8   26 ms    25 ms     27 ms    pop1-kcy-P0-0.atdn.net
                                 [66.185.137.225]
9   26 ms    49 ms     27 ms    sl-gw16-kc-9-0.sprintllnk.net
                                 [144.232.131.65]
```

```
10  30 ms   '  30 ms       49 ms     sl-bb20-kc-8-0.sprintlink.net
                                      [144.232.23.53]
11  80 ms     75 ms        76 ms     sl-bb21-sea-8-3.sprintlink.net
                                      [144.232.18.98]
12  75 ms     80 ms        74 ms     sl-bb21-sea-15-0.spnintlink.net
                                      [144.232.6.89]
13  75 ms     77 ms        18 ms     sl-gwll-sea-7-0.sprintlink.net
                                      [144.232.6.126]
14  76 ms     88 ms        75 ms     sl-gcomm-2-0.sprintlink.net
                                      [144.228.93.234]
15  80 ms     77 ms        77 ms     InetSeaSDCgw-2.gci.net
                                      [209.165.129.34]
16  118 ms    115 ms       111 ms    209.165.170.157
17  114 ms    112 ms       119 ms    209.165.128.1
18  113 ms    109 ms       119 ms    209.165.128.82
19  112 ms    113 ms       110 ms    www.alaska.org  [205.140.73.143]
```

Neki od ovih sajtova su slični, ali neki se znatno razlikuju. Ovo je interesantno zato što se dva kompjutera sa kojih smo izdavali ove komande nalaze na rastojanju od nekoliko milja. Različite rute se koriste prvenstveno zbog ISP-ja čije usluge koriste. Linux server je deo univerzitetske mreže, koja je, takode, povezana na mrežu zasnovanu na SONET-u pod nazivom Wiscnet. DOS komanda je uneta na mašini koju opslužuje Roadrunner, širokopojasni ISP. Jednostavno rečeno, svaki server koristi druge organizacije i linije za uspostavljanje konekcija.

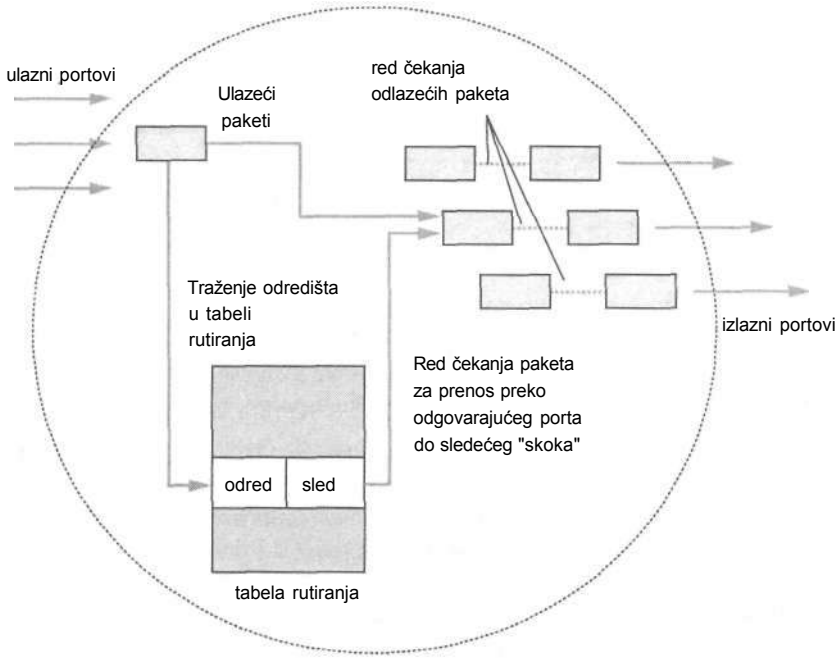
Ruten

Sve do sada smo krajnje uprošćavali način funkcionisanja rutera. Na slici 11.8 su prikazane primarne funkcije rutera. Paket stiže preko ulaznog porta i logika u ruteru preduzima sledeće akcije:

- izvlačenje odredišne adrese iz paketa
- pronalaženje te adrese u tabeli rutiranja
- pristup vrednosti sledećeg "skoka" i utvrdivanje odgovarajućeg odlazećeg porta
- premeštanje paketa u red čekanja za taj port
- prenos paketa

Iako je ovo sigurno tačno, pravo pitanje je kako ruter funkcioniše. Dizajn rutera je izuzetno složen i postoji mnogo pitanja u vezi tajminga, postavljanja u red čekanja, planiranja i intemog funkcionisanja koje pomera paket sa dolazećeg na odlazeći port. Ovde ne možemo detaljno da opišemo dizajn rutera, ali ćemo predstaviti nekoliko značajnih ideja. Više detalja o ruterima sadrže reference [Pi03] i [CaOl]. Ukratko ćemo se pozabaviti problemima pronalaženja vrednosti sledećeg skoka, premeštanja paketa sa ulaznog na izlazni port i planiranja transmisije paketa.

Pronalaženje sledećeg "skoka" do odredišta je relativno lako - ruter proverava svoju tabelu. Problem je što traženje mora da se izvede veoma brzo. Zamislite da su ulazni portovi optičke fiber konekcije i da paketi stižu brzinom od 100 Mbps. Ako ruter ne uspe da pošalje te pakete dovoljno brzo pre nego što stignu novi, paketi će se nagomilati i doći će do kašnjenja. U ekstremnim slučajevima može da dođe do prekoračenja bufera i tada se paketi izbacuju.



SLIKA 11.8 *Funkcije rutera*

Pretraživanje je jedna od važnijih tema brojnih kurseva o programiranju i strukturi podataka. Među poznatije metode ubrajaju se linearna pretraživanja, binarna pretraživanja, pretraživanja B-stabla, ostali metodi zasnovani na stablima i hash strukture. Normalno, linearna i binarna pretraživanja su suviše spora za high-speed rutere i ne mogu dovoljno brzo da obrade pakete. Korišćenje hash strukture, koja je još poznata i kao **memorija sa adresabilnim sadržajem (content addressable memory),** verovatno je najbrži i najsofisticiraniji metod. Ideja je krajnje jednostavna. Kada ruter kreira zapis u tabeli za neko odredište, primenjuje hash funkciju na vrednost odredišta (slika 11.9). Funkcija generiše lokaciju na kojoj se smešta vrednost sledećeg "skoka". Kasnije, kada stigne paket sa tom odredišnom adresom, koristi se hash funkcija kako bi se utvrdilo gde se tačno u tabeli nalazi vrednost sledečeg skoka. U ovom slučaju nema pretraživanja, jer se tačna lokacija izračunava i ruter veoma brzo pronalazi vrednost sledećeg skoka. Ipak, korišćenje i kreiranje hash funkcija prate brojni problemi (oni su najbolje objašnjeni na predavanjima o strukturama podataka i algoritmima za njihovo rešavanje). Za naše potrebe dovoljno je da znate da se sledeći "skok" brzo pronalazi.

Kada je sledeći skok poznat, ruter i dalje treba da pomeri paket sa ulaznog na izlazni port. Ovo se ne razlikuje od načina na koji više procesora pristupa memoriji. Jedan pristup nalaže praćenje dizajna više minikompjutera i korišćenje *sistema transportne magistfale (bus transport systetn),* kod koga se svaki paket koji stigne preko ulaznog porta šalje do odgovarajućeg izlaznog porta preko magistrale.

SLIKA 11.9 *Hash funkcija za ažuriranje labele nitiranja*

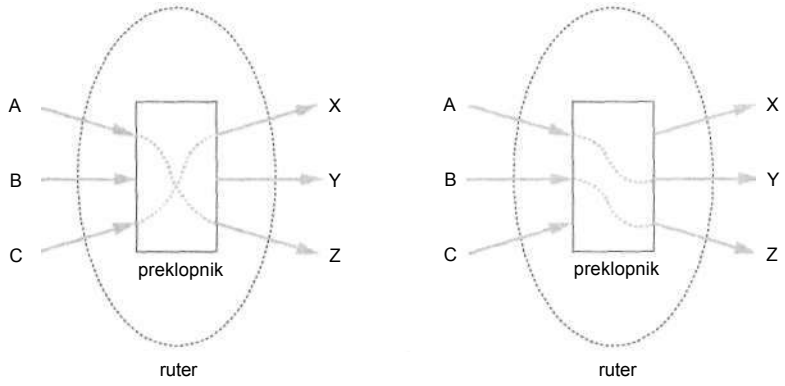
Jednostavan je, ali glavni nedostatak je isti kao i kod mreža zasnovanih na magistralama: jedino jedan paket može da se premešta u jednom trenutku, što usporava ceo proces.

Sledeći pristup podrazumeva korišćenje *deljene memorije.* To znači da više procesora može da pristupi istoj memoriji. Redovi čekanja na svim izlaznim portovima odgovaraju naznačenoj oblasti deljene memorije. Svaki ulazni port je kartica na kojoj se nalazi procesor koji pronalazi vrednost sledećeg skoka. Taj procesor može da prenese paket iz jedne oblasti memorije u drugu, u zavisnosti od reda čekanja kome je dodeljen. Pošto postoje zasebni procesori (po jedan za svaku ulaznu karticu), postoji viši nivo konkurentnosti, koji omogućava brže procesiranje. Naravno, i dalje postoje neka pitanja na koja se čekaju odgovori; na primer, šta se dešava ako dva zasebna procesora treba da pomere paket u isti red čekanja i kako procesori komuniciraju sa memorijom.

Sledeći pristup podrazumeva dizajniranje hardverskog preklopnika koji povezuje svaki ulazni port sa svakim izlaznim portom (slika 11.10). Preklopnik *(sivitch)* je kolo koje uspostavlja konekcije izmedu dve krajnje tačke. Na slici 11.10a paket koji stižepreko ulaznog porta A prenosi se do izlaznog porta Z, a paket koji stigne preko ulaznog porta C prenosi se na izlazni port X. Na slici 11.10b "stvari" se menjaju. Preklopnik povezuje A sa Y i B sa Z. Nameću se nova pitanja. Šta se dešava ako sva tri ulazna porta imaju pakete namenjene istom izlaznom portu? Kako da dizajnirati preklopnike? I to je dobra tema za naprednije studije.

Poslednje pitanje je planiranje paketa na izlaznim portovima. Opravdano je pretpostaviti da paketi koji stižu na različite ulazne portove moraju da produ preko istog izlaznog porta. Kako ruter treba da ih isplanira? Kojim redosledom se prenose? Jedan pristup nalaže smeštanje paketa u bafere izlaznog porta istim redosledom kojim pristižu. Naime, baferi su, u stvari, redovi čekanja. Prenose se paketi koji su najduže u redu čekanja. Ovo je jednostavan pristup i funkcioniše dobro kod brojnih aplikacija. Medutim, **kvalitet servisa (QoS - quality of service)** je postao značajan u poslednjih nekoliko godina. U osnovi, on definiše tip servisa koji se očekuje od mreže.

Kao što smo ranije istakli, Intemet je bio dizajniran za upravljanje uglavnom prenosom fajlova i emailom. Pojava kraćih čekanja dok se paketi nalaze u izlaznim redovima čekanja kod takvih aplikacija ne stvara ozbiljnije probleme i teško da će ih iko i primetiti.



(a) Povezivanje A sa Z i C sa X

(b) Povezivanje A sa Y i B sa Z

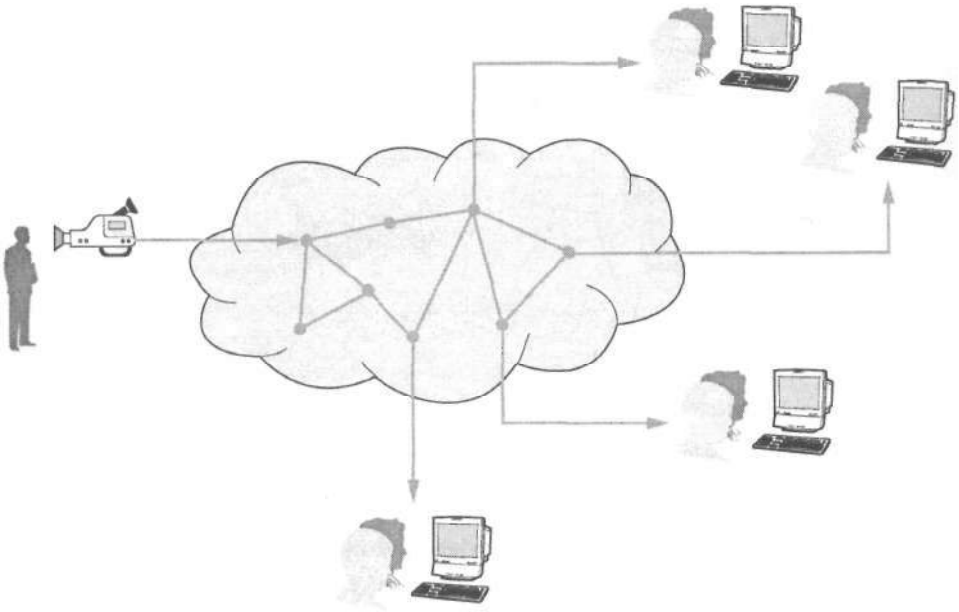
SLIKA 11.10 *Ruter zasnovan na preklopnicima*

Međutim, nije dizajniran za real-time audio, ili video zapise. Kod takvih aplikacija QoS očekivanja nalažu pravovremenu isporuku paketa bez kašnjenja. U takvim slučajevima standardno definisanje redova čekanja odlazećih paketa može da ima suprotan efekat od kvaliteta koji se očekuje kod takvih aplikacija. Zato su nam potrebne drugačije opcije, kao što su prioritetni redovi, struktura u kojoj se paketi organizuju po prioritetu koji može da se defmiše na osnovu vfednosti polja Type of Service u IP paketu. Najviši prioritet imaju paketi koji su blizu početka reda i oni se brzo prenose. Negde između aplikacije i kreiranja IP paketa protokoli su morali da postave polja za odgovarajući tip aplikacije. Ovde je značajno istači da su neke opcije koje su na raspolaganju za defmisanje prioriteta i obradu paketa brže od ostalih. Vratićemo se na ovu temu kasnije, kada budemo razmatrali Resource Reservation Protocol.

Rutiranje paketa ka više odredišta

Naše dosadašnje predstavljanje rutiranja bilo je zasnovano na pretpostavci da svaki paket ima jednu unicast adresu, što znači da sledi specifičnu rutu od odredišta do jedinstvenog odredišta. Sledeća raspoloživa opdja za slanje paketa uključuje *multicast adresu.* Kao što smo ranije objasnili, IP adresa Klase D definiše prenos ka više odredišta (tzv. multicasting). To znači da jedna multicast adresa predstavlja, u stvari, niz odredišta; svako od njih mora da primi jednu kopiju paketa. Ovo je unekoliko slično postavljanju distributivne liste za email. U Vašem adresam postoji zapis distributivne liste koji sadrži više email adresa. Kada šaljete email, dovoljno je sarno da selektujete jedan zapis iz distributivne liste. Ipak, do svake osobe se šalje zasebna email poruka. Slično se dešava i prilikom rutiranja ka više odredišta, samo što sve funkcioniše na drugom sloju.

)edan primer gde je ovakvo rutiranje korisno jeste emitovanje video sadržaja za selektovane korisnike na mreži. Podaci izživog video prenosa (slika 11.11) dele se u pakete i prenose se preko mreže.



SLIKA 11 .T1 *Emitovanje video zapisa na vile odredišta*

Medutim, adresa Klase D u svakom paketu definiše grupu ljudi koji su se prijavili za gledanje videa. Kopija svakog paketa mora da pronade način da dođe do svake osobe iz multicast grupe. U svakom sajtu podaci se izvlače iz paketa i predstavljaju u vidljivoj formi.

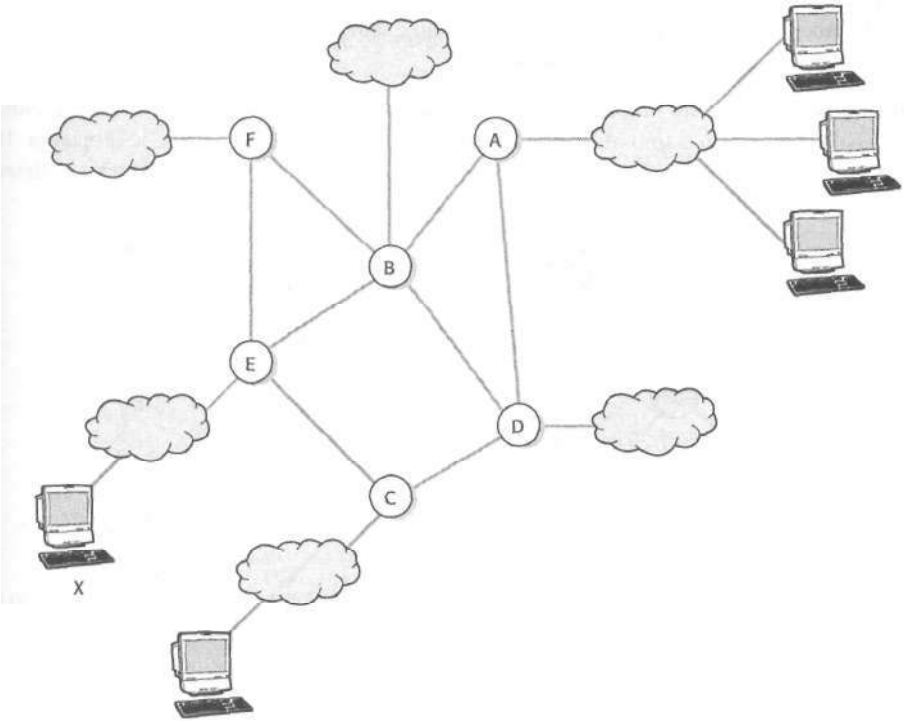
Multicasting ima nekoliko različitih komponenata. Prva je **Internet Group Management Protocol (IGMP),** protokol koji funkcioniše između hosta i lokalnog rutera i omogućava hostu da se pridružuje, ili napušta različite multicast grupe. Hi host, ili lokalni ruter mogu da pošalju IGMP poruku ugradenu u IP paket kod koga je vrednost polja Protocol postavljena na 2. Slede primeri takvih razmena.

- Ruter šalje upit do jednog, ili više hostova da bi saznao ko pripada određenoj multicast grupi. Poruka sadrži adresu Klase D za tu grupu.
- Host šalje poruku do lokalnog rutera, ukazujući mu da pripada odredenoj multicast grupi. Kao i ranije, poruka sadrži odgovarajuću adresu.
- Host šalje poruku do lokalnog rutera, ukazujući mu da više ne pripada odredenoj multicast grupi. Poruka sadrži odgovarajuću adresu.

Ovaj deo je relativno jednostavan, jer se IGMP koristi jednostavno da bi se ukazalo na pripadnost određenoj grupi. Nije defmisano kako multicasting funkcioniše. Teži deo je poslati ove informacije do svih rutera na mreži i implementirati na njima neki tip algoritma za rutiranje ka više odredišta.

Na slici 11.12 prikazan je primer. Svaka ikonica personalnog kompjutera predstavlja host koji se pridružio multicast grupi; grupa se sastoji isključivo od tih hostova. Zato ruter A zna da se u grupi nalaze tri hosta, a ruteri C i F znaju za po jedan host u grupi. Ostali ruteri su povezani na mreže kod kojih hostovi nisu pridruženi grupi. U stvari, u kontekstu definisanja algoritma za rutiranje ka više odredišta nema razlike ako A zna za tri, ili 300 hostova u grupi. Ruter A mora da bude angažovan u svakom slučaju. Slično tome, ruteri C i E moraju da budu angažovani. Medutim, iako ruteri B, D i F nemaju lokalne hostove u grupi, bar jedan od rutera B, ili D takođe mora da bude angažovan. U suprotnom, ne bi postojao način da se stigne od A do C, ili E, U ovom slučaju nema potrebe angažovati ruter F.

Pošto smo definisali dijagram, ovo je možda dobar trenutak da se postavi pitanje zašto koristiti multicasting umesto slanja zasebnih unicast paketa do svih hostova u grupi. Pretpostavimo da hostX izabere takav pristup. Zato bi na slici 11.12 morao da pošalje četiri kopije svakog paketa koji potiče iz X. Svaka od te četiri kopije mora da se rutira preko Interneta. Naravno, ako postoji *n* članova grupe, onda bi bilo potrebno poslati *n* -1 kopija svakog paketa. U slučaju multicastinga X šalje samo jednu kopiju svakog paketa. Kada ruter F dobije paket, šalje samo jednu kopiju do C i još jednu kopiju možda do B (u zavisnosti od korišćenog algoritma za rutiranje) i na kraju doA.



SLIKA 11.12 *Ruteri uključeni u multicasting*

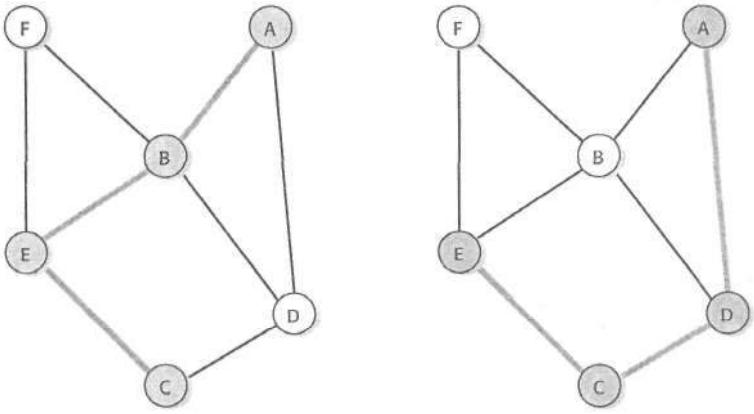
Multicasting podrazumeva manji broj paketa koji "putuju" preko Interneta; zato je mnogo efikasniji od slanja više unicast paketa. Ovde razlika nije mnogo velika, ali postaje značajna u situaciji u kojoj postoje stotine hostova u grupi, umesto samo nekoliko.

Sada dolazimo do najznačajnijeg pitanja - kako svaki ruter zna šta da radi sa multicast paketom. Pojednostavljeno rešenje podrazumeva da se kreira spanning tree od rutera koji dopiru do svih hostova u grupi. Za grupu sa slike 11.12 to stablo može da bude A-B-E-C (slika 11.13a), ili A-D-C-E (slika 11.13b). U ovom slučaju spanning tree mora da sadrži A, E i C i najmanje jedan čvor koji omogućava da se iz A dode do C, ili do E. Ako nešto od ovoga zvuči poznato, znači da ste pažljivo pročitali deo Poglavlja 9 u kome je spanning tree algoritam korišćen za eliminisanje petlji prilikom konekcija koje se kreiraju pomoću mostova. L1 ovom kontekstu stabla se obično nazivaju multicast stabla, umesto stabla razapinjanja (spanning trees).

Problem potiče iz teorije grafova - za konkretni skup temena i grana u grafu treba kreirati stablo razapinjanja koje sadrži podskup temena grafa. Postoji nekoliko različitih pristupa; većina može da se svrsta u jednu od dve kategorije.

Prvi način je kreiranje jednog multicast stabla koje povezuje odgovarajuće čvorove. Drugi podrazumeva kreiranje jednog multicast stabla za sve moguće izvore u grupi. Kada se multicast stablo kreira, svaki ruter u njemu jednostavno prenosi sve primljene multicast pakete preko svih odlazećih linkova u stablu. Naravno, ovo se primenjuje samo na jednu multicast adresu. Za svaku adresu je neophodno kreirati zasebno stablo.

Problem je što je kreiranje multicast stabala, posebno u mrežama sa velikim brojem čvorova, teško. Ovde opisujemo samo distribuirani algoritam za kreiranje stabala iz Poglavlja 10; zamislite kako to izgleda u globalnim razmerama! Zbog toga, većina IP rutera ne podržava multicasting.



(a) A-B-E-C (b) A-D-C-E

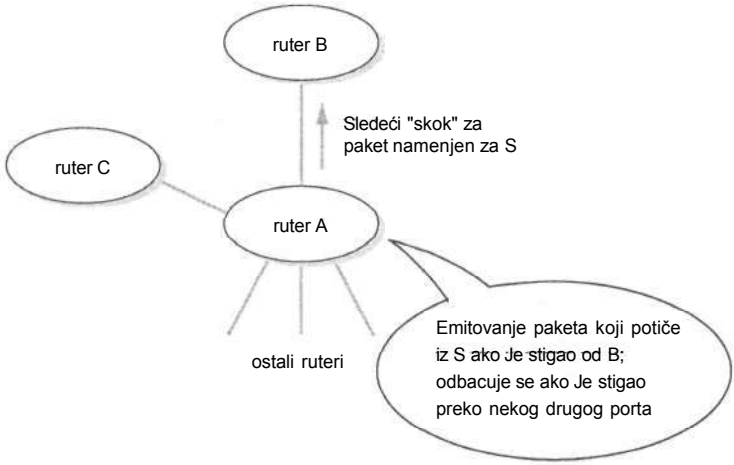
SLIKA 11.13 *Moguće multicast stablo za mrežu sa slike 11.12*

Manji* broj IP rutera formira Mbone, mrežu u okviru Intemeta koja podržava rutiranje adresa Klase D.

Način funkcionisanja Mbone mreže je složen, a ideja se zasniva na nekim mehanizmima za rutiranje koje smo predstavili prilikom kreiranja multicast stabla. Metod predstavlja **Distance Vector Multicast Routing Protocol (DVMRP)** i koristi nekoliko komponenata. Jedna je **emitovanje obrnutom putanjom (RPB - reverse-path broadcasting).** Pod *emitovanjem* se podrazumeva da paket treba da ode na sva moguća odredišta. Ruter obično može da implementira algoritam za emitovanje prosleđivanjem paketa na sve izlazne portove. Međutim, kao što ste videli u prethodnom poglavlju, paketi "putuju" **u** petljama i količina saobraćaja postaje preterana. RPB pretpostavlja da ruter zna sledeći link duž najkraće putanje do konkretnog čvora. Zbog toga, kada ruter dobije paket, preduzima sledeće:

- Utvrduje izvor dolazećeg paketa i port preko koga je primljen.
- Traži izvor u tabeli rutiranja i utvrđuje sledeći "skok" na putanji nazad do tog izvora.
- Ako sledeći "skok" odgovara portu preko koga je paket stigao, paket se šalje do svih ostalih portova.
- Ako prethodni uslov nije ispunjen, paket se odbacuje.

Na primer, ako ruter A primi paket od izvora S, preko rutera B na slici 11.14, paket će biti emitovan do svih ostalih portova.



SLIKA 11.14 *Ruter izvršava protokol inverznog emitovanja.*

Emitovanje paketa koji potiče iz S ako je stigao od B; odbacuje se ako je stigao preko nekog drugog porta. (Broadcast packet originating at S if it arrives from B; drop it if it arrives over any other port.)

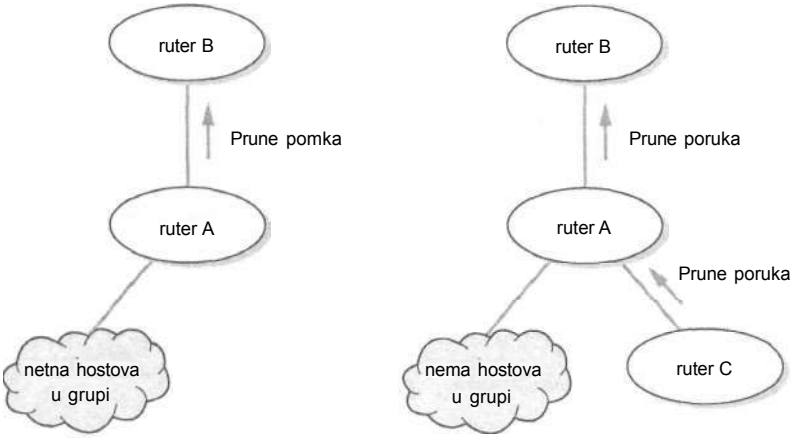
' *Uanji* ne znači *mali.* I dalje mogu da postoje hiljade rulera koji pokreću Mbone softver.

Ako primi paket preko bilo kog drugog porta, paket će biti odbačen. Ovo je slično defmisanju emisionog stabla sa izvorom S kao korenom stabla i prosleđivanju paketa samo duž putanja koje vode od S. Dok svi ostali ruteri rade isto, paketi neće "putovati" preko petlji.

Ipak, ovaj metod, sam po sebi, ne funkcioniše dobro kod multicastinga. Paket može da bude poslat u smerovima u kojima nema hostova iz multicast grupe. Zbog toga, emisiono stablo sa izvorom S kao korenom stabla mora da se "potkreše" da bi bile eliminisane takve grane. Zato je druga komponenta u DVMRP algoritam za skraćivanje koji postavlja granice kuda ruter može da prosleduje poruke.

Pretpostavimo da ruter dobije multicast paket. Startuje korišćenjem RPB-a i inicijalno multicast paketi mogu da idu do lokacija na koje nije neophodno da idu. Medutim, pretpostavimo da ruter dobija multicast paket, ali da je povezan na mrežu koja nema hostove u toj multicast grupi. Sećate se da IGMP protokol neprestano obezbeđuje najnovije informacije za rutera ko je u toj grupi. U ovom slučaju on šalje Prune poruku do rutera od koga je dobio multicast poruku. Kada ruter dobije Prune poruku pridruženu njegovoj multicast adresi, on zaustavlja multicasting u tom smeru (primer je prikazan na slici 11.15a). Ruter A dobija multicast poruku od rutera B, ali mreža na koju je A povezan nema hostove u toj grupi. A šalje do B Prune poruku i B više ne šalje do A multicast poruke.

Ponovo smo isuviše pojednostavili ceo proces! Šta bi bilo da smo imali situaciju sa slike 11.15b? Možda postoje hostovi koji su prošli ruter C, a *nalaze* se u grupi. U tom slučaju ruter B mora i dalje da pošalje multicast pakete do A. Ipak, pretpostavimo da A primi Prune paket od C i Prune pakete od svih narednih rutera. U tom slučaju nema razloga da se šalju multicast paketi u bilo kom od tih smerova i A može da pošalje Prune paket do B, tražeći od B da prestane da šalje takve pakete.



(a) Nema rutera iza A

(b) Iza A postoje ruteri

SLIKA 11.15 *Skraćivanje stabla*

Šta se dešava ako A naknadno dobije IGMP poruku od hosta koji ukazuje da se pridružio grupi? Pošto je A izbačen iz stabla, neće dobiti multicast pakete. Ipak, A može da pošalje Graft poruku do B, ukazujuđ da bi B trebalo da nastavi da šalje multicast pakete do A.

Sasvim je sigurno da ima i drugih pitanja, ali, kao i obično, ovde smo se bavili samo osnovama. Više detalja o rutiranju ka više odredišta možete da pronađete u referencama [Ko98], [Gr02], i [HaOl]. U stvari, rutiranje ka više odredišta je razvijeno i za IP preko LEO satelita (ref. [Ek02]).

Resource Reservation protokol (RSVP)

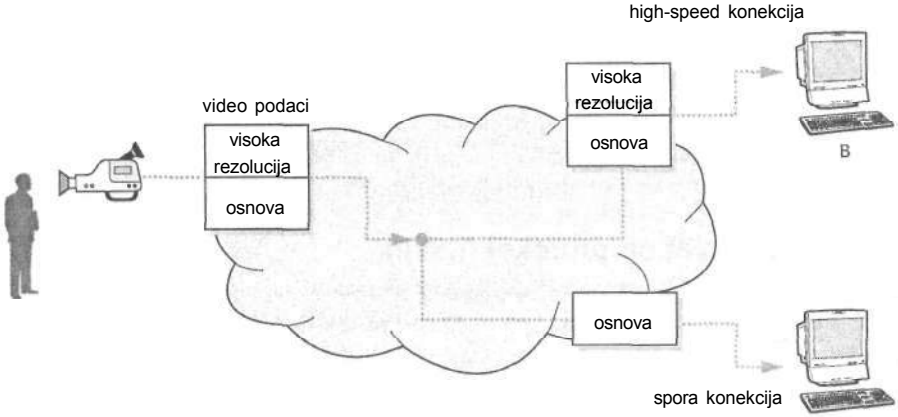
Duži niz godina kritikovan je IP da nije ispunio QoS zahteve za mnoge novije aplikacije. Proces ažuriranja tabela rutiranja i nezavisnog prosledivanja paketa sigurno omogućava dolazak paketa sa različitim brzinama. Za prenos fajlova, email i^uifovanje Webom to ne predstavlja problem. Međutim, aplikacije kao što su živi video prenosi na više odredišta imaju QoS zahtev koji određuje minimalnu bitsku brzinu. To znači da paketi moraju da stižu minimalnom brzinom koja je određena potrebama krajnjeg korisnika.

Mreže poput telefonskog sistema oslanjaju se na fizička kola za prenos govornih podataka u realnom vremenu. U Poglavlju 13 predstavićemo Asynchronous Transfer Mode (ATM), protokol koji prenosi podatke pomoću virutelnih kola (videti Poglavlje 1). Sličnost se ogleda u tome što kod obe mreže, pre ikakve razlike podataka, postoji proces signalizacije koji uspostavlja kolo koje će biti korišćeno. Kada se kolo defmiše, logika u svakom komutatoru rezerviše ono što je potrebno za garantovanje prijema i prenosa podataka dovoljno velikim brzinama za ispunjavanje QoS zahteva. Ovo nije moguće kod IP-a.

Zato je IETF razvio **Resource Reservation Protocol (RSVP),** protokol koji ispunjava QoS zahteve preko Interneta. Ovaj protokol ugrađuje poruke u IP pakete. Te poruke sadrži informacije o određenom toku podataka i zahtevima za dovoljnom količinom resursa koji se moraju rezervisati da bi bili ispunjeni QoS zahtevi. Može da se koristi sa različitim rešenjima, ali ovde ga opisujemo u kontekstu rutiranja ka više krajnjih korisnika za različitim QoS zahtevima.

Na slici 11.16 definisan je naš kontekst. Preko mreže je uspostavljena video linija i video podaci će biti rutirani ka više krajnjih korisnika. Međutim, različiti korisnici imaju razlidte mrežne konekcije koje ograničavaju brzinu kojom mogu da primaju podatke. A ima sporu konekciju koja mu ne dopušta gledanje videa u visokoj rezoluciji, dok B ima high-speed konekciju koja to omogućava. Kako pošiljalac može da pošalje podatke za oba korisnika?

Jedan način za rešavanje ovog problema je transfer video podataka po slojevima. Tako pošiljalac može da funkcioniše nezavisno od primaočevih potreba i mogućnosti. Osnovni sloj može da obezbeduje jednostavnu video sliku u niskoj rezoluciji. Drugi sloj visoke rezolucije može da se koristi za poboljšavanje slike kreirane na osnovnom sloju. Naravno, za to je neophodno imati veče bitske brzine. Korisnik A može da primi samo podatke sa osnovnog sloja na nižim brzinama, dok korisnik B može da prima podatke sa oba sloja na većim bitskim brzinama. Oba korisnika imaju određene QoS zahteve, a RSVP im omogućava da postavljaju zahteve posredničkim ruterima koji če ispuniti te zahteve.



SLIKA 11.16 *Prijem živog video prenosa na različitim brzinama*

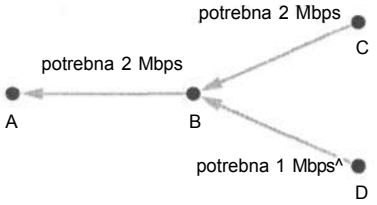
RSVP definiše dva tipa poruka koje ovo rade. Te poruke se ugrađuju u IP pakete koja u polju Protocol imaju vrednost 46. Ruteri koji znaju RSVP mogu da izvuku te poruke i da izvrše odgovarajuće akcije. Prva poruka je Path poruka - pošiljalac je šalje preko multicast stabla do svakog primaoca. Kada Path poruka stigne u ruter, pamti se adresa rutera sa koje je stigla. Ovo se kasnije koristi kada se postave QoS zahtevi. Path poruka sadrži i Tspec identifikator pošiljaoca (identifikator saobraćaja), parametar koji opisuje saobraćaj (na primer, bitsku brzinu kojom će pošiljalac prenositi podatke).

Kada primalac dobije Path poruku, inicira QoS zahtev slanjem Reserve poruke. Reserve poruka prvo ide do rutera koji je isporučio Path poruku. Svaki ruter šalje Reserve poruku do rutera koji je poslao prethodnu Path poruku (ova informacija je zapamćena kada je stigla Path poruka). Tako Reserve poruka ide naviše kroz multicast stablo.

Reserve poruka sadrži informacije o QoS zahtevima koje primalac očekuje - na primer, o zahtevanoj bitskoj brzini. Ruter izvlači ove informacije i zna da mora da prosledi pakete dovoljno brzo da bi se generisala tražena bitska brzina. Zato može da rezerviše resurse koji će biti neophodni (možda prostor u baferu) i da modifikuje svoje parametre za planiranje tako da paketi mogu da se proslede traženom brzinom.

Ovakvi paketi mogu da se identifikuju na osnovu izvorne i odredišne adrese i vrednosti koja je smeštena u polju Type of Service u zaglavlju paketa. RSVP ne defmiše kako ruter to treba da izvede; obezbeduje samo tražene QoS zahteve. Ruter radi ono što je potrebno, u zavisnosti od svog dizajna. Nakon toga, prosleđuje Reserve poruku do narednog višeg rutera u multicast stablu,

Međutim, pretpostavimo da ruter primi nekoliko Reserve poruka od rutera koji se nalaze ispod njega u stablu. Ako svaki od njih ima drugačije QoS zahteve u istom nizu, ruter treba da pošalje samo jednu Reserve poruku do svog prethodnika, sa naznakom maksimalne tražene brzine. Na primer, pretpostavimo da C i D (slika 11.17) zahtevaju QoS od 1 Mbps i 2 Mbps.



SLIKA 11.17 *Mešanje Reserve poruka*

B mora da obezbedi maksimalnu brzinu medu postavljenim zahtevima, tako da šalje Reserve poruku do A sa naznačenom potrebom za 2 Mbps.

Informacije u Reserve poruci definišu *meko stanje (soft state)* rutera. Za razliku od strogog (hard) stanja (ono se definiše za virtuelne putanje), koje mora eksplicitno da se promeni, meko stanje ostaje na snazi u kraćem periodu. Nakon toga, ruter se vraća na svoje podrazumevane procedure. Da bi održao meko stanje, ruter mora periodično da prima Reserve poruke.

Meko stanje postoji jer se ruteri na Internetu mogu menjati. "Otkaz", ili zagušenje rutera mogu da iniciraju ažuriranje tabela rutiranja i preusmeravanje saobraćaja na neku drugu rutu. Kada se ovo desi, novi ruteri moraju da implementiraju primljene QoS zahteve primaoca, a stari ruteri više ne bi trebalo da rezervišu zahtevane resurse.

Ovde je pretpostavljeno da postoji samo jedan multicasting, ali u praksi može da postoji više multicastinga iz različitih izvora. Osim toga, RSVP nije ograničen isključivo na ovakvu vrstu rutiranja. U referenci [Fo03] možete da pronadete dodatne detalje o ovom protokolu.

Internet Control Message protokol (ICMP)

Pošto IP ne garantuje pouzdan servis, Internet **Control** Message **Protocol (ICMP)** koristi se za podnošenje izveštaja o greškama i za obezbedivanje najnovijih informacija za rutere o uslivima koji su stvoreni na Internetu. ICMP šalje poruke, ugradujući ih u IP pakete i postavljajući polje Protocol u zaglavlju na I.

Sledeća lista sadrži neke tipične kontrolne poruke koje ICMP šalje.

- **Destination** Unreachable Kao što smo ranije istakli, IP ne može da garantuje isporuku paketa. Odredište možda ne postoji, ili je trenutno nefunkcionalno, pošiljalac je možda postavio zahtev za rutom koju je nemoguće izvesti, ili je paket sa postavljenim flegom Do Not Fragment isuviše veliki da bi se enkapsulirao u okvir. U takvim situacijama ruter detektuje grešku i šalje ICMP paket do originalnog pošiljaoca. Može da sadrži celo IP zaglavlje neisporučenog paketa i prva 64 bita njegovih podataka, tako da pošiljalac može da prepozna paket koji nije mogao biti isporučen.
- Echo Request ICMP koristi ovaj paket kako bi utvrdio da li je odredeno odredište dostupno. Na primer, ako A želi da zna da je B dostupan, on šalje Echo Request paket adresiran na B. Ako B primi paket, on reaguje slanjem Echo Reply paketa nazad do A. Echo Reply paket vraća sve podatke koji su bili postavljeni u Echo Request paket. Ovo

je možda jednostavnije od pokretanja protokola sa celim nizovima paketa koji se šalju do odredišta samo da bi se na kraju utvrdilo da nije dostupno.

- **Echo Reply** Šalje se kao odgovor na Echo Request paket. Korisnici možda od ranije znaju za komandu ping, koja koristi kombinaciju Echo Request/Reply paketa. Korisnici mogu da izdaju komandu `ping` nazivhosta na UNIX-U, ili iz DOS prompta kako bi se utvrdila dostupnost hosta. Na primer, unošenjem komande `ping www.uwgb.edu -c 3` iz Linuxovog prompta generiše se sledeđ izlaz:

```
PING weba.uwgb.edu (143.200.128.158) from 143.200.128.235
: 56(84) bytes of data.
64 bytes from weba.uwgb.edu (143.200.128.158): icmp_seq=1
tt1 = 128 time =0.198 ms
64 bytes from weba.uwgb.edu (143.200.128.158): icmp_seq=2
tt1 = 128 time =0.190 ms
64 bytes from weba.uwgb.edu (143.200.128.158): icmp_seq=3
tt1 =128 time=0.180 ms
```

Unošenje `ping` komande inicira ICMP da pošalje sekvencu Echo Request paketa do naznačenog odredišta. Opcija **-c** 3 definiše broj paketa koji će biti poslati. Prethodno prikazane linije izlaza predstavljaju vraćene Echo Reply pakete. Poslednja vrednost u svakom odzivu je vreme punog kruga.

- **Parameter Problem** Pretpostavimo da IP paket sadrži grešku, ili nedozvoljenu vrednost u nekom polju zaglavlja. Ruter otkriva grešku i šalje Parameter Problem paket nazad do izvora. Ovaj paket sadrži problematično IP zaglavlje i pokazivač na polje zaglavlja u kome postoji greška.

- **Redirect** Pretpostavimo da host stanica šalje paket do rutera i da ruter zna da paket može da se brže isporuči preko nekog drugog rutera. Da bi se olakšalo buduće rutiranje, ruter šalje Redirect paket nazad do hosta. On obaveštava host stanicu gde se drugi ruter nalazi i da bi ubuduće sve pakete do istog odredišta trebalo slati njemu. Ovako je omogućeno dinamičko ažuriranje tabela rutiranja i na taj način je moguće povoljno iskoristiti neke primene uslova na mreži. Redirect paket se ne koristi za ažuriranje rute između rutera, jer IP paket sadrži izvornu adresu, a ne adresu rutera koji je prethodno imao paket. Kada ruter primi paket od drugog rutera, on ne može da zna tačno od koga je paket stigao.

- **Source Quench** Ako ruter primi isuviše veliki broj paket od hosta, može da pošalje poruku kojom zahteva redukovanje učestalosti kojom se paketi šalju.

- **Time Exceeded** Time Exceeded paket se šalje kada je vrednost polja Time to Live u IP paketu dostigla O, ili kada je tajmer za ponovno sastavljanje paketa (postavljen prilikom prijema prvog fragmenta paketa) istekao. U svakom slučaju, paket, ili neki nesastavljeni fragmenti se odbacuju sa mreže. Ruter koji je kriv za njihovo odbacivanje šalje Time Exceeded paket do izvora kako bi se ukazalo da paketi nisu isporučeni.

- **Timestamp Request i Timestamp Reply** Vremenska oznaka paketa (timestamp) omogućava hostu da proceni koliko je vremena potrebno za krug do drugog hosta i nazad. Host kreira i šalje Timestamp Request paket u kome se nalazi vreme prenosa (originalna vremenska oznaka). Kada prijemni host dobije paket, on kreira Timestamp Reply paket koji sadrži originalnu vremensku oznaku - trenutak u kome je prijemni host dobio paket (vremenska oznaka prijemne strane) i vreme kada je prijemni host poslao odgovor (vremenska oznaka prenosa). Kada primi odgovor, originalni pošiljalac beleži vreme dolaska. Razlika između vremena dolaska i originalne vremenske oznake predstavlja vreme kruženja. Izračunavanjem razlike izmedu vremenske oznake prijema i prenosa host može da utvrdi koliko je drugom hostu trebalo vremena da generiše odziv na zahtev. Oduzimanjem tog vremena od vremena kruženja host može da proceni vreme prenosa i zahteva i odgovora. Vremenske oznake paketa omogućavaju hostu da proceni efikasnost mreže u isporuci paketa.

- **Address** Mask **Request and Reply** Ranije smo pominjali podmreže kao jedan moguči način za dodeljivanje istog broja IP mreže za više fizičkih mreža. Korišćenjem nekoliko dodatnih bitova iz dela lokalnog ID-ja u IP adresi brojevi podmreže mogu da se dodele različitim fizičkim mrežama. Na primer, sajt Klase B može da upravlja sa osam zasebnih fizičkih mreža pomoću tri bita iz lokalnog ID-a za naznačavanje fizičke mreže. Ti bitovi, zajedno sa 16-bitnim brojem IP mreže, formiraju broj podmreže.

  Da bi izvukao broj podmreže iz IP adrese, interni ruter koristi masku adrese **podmreže (subnet address mask)** i izvršava logičku operaciju I na nivou bitova između maske i konkretne IP adrese. Na primer, pretpostavimo da je prethodno pomenuti sajt Klase B imao IP adresu 143.200.123.78. Koji je broj podmreže? U ovom slučaju maska adrese je 255.255.224.0, ili

  11111111.11111111.11100000.00000000

  (primečujete da postoje 19 jedinica zbog 16-bitnog broja IP mreže i tri bita iz lokalnog ID-a). Zatim, primenjuje se logičko I izmedu bitova maske i IP adrese. Rezultat je 19-bitni broj podmreže (iza koga sledi 13 nula) - **u** ovom slučaju 143.200.96.0. Host može da pošalje Address Mask Request paket do rutera da bi utvrdio masku adrese za mrežu na koju je priključen. Ruter može da pošalje odgovor.

- **Information Request i Information Reply** Ove poruke su originalno dizajnirane kako bi host utvrdio svoju IP adresu kada se startuje. To sada obično rade drugi protokoli, tako da su ove dve poruke postale zastarele.

11.3 IPv6

Kao što smo istakli na početku ovog odeljka, Internet protokol datira još sa kraja 60-ih prošlog veka (na polju računarstva to može da se uporedi sa kamenim dobom). Ranije smo rekli da Je Internet protokol počeo da pokazuje neke znake "starenja"; u ovom odeljku prestavljamo IPv6, koji je naslednik tekuće verzije Internet protokola (IPv4). Mnogi će možda pitati da li je postojao IPv5. Jeste postojao, u izvesnom smislu. Protokol poznat kao ST2 (Stream Protocol version 2) bio je razvijen kao vezi orijentisani protokol koji je funkcionisao na sloju 3. Namera je bila da se razvije protokol za real-time podatke čija bi isporuka bila naznačena krajnjim rokovima.

Neki su videli ST2 kao naslednika IPv4 i nazivali su ga 1PV5. Medutim, pošto je razvojem RSVP-a (koji je ranije opisan) rešen problem real-time QoS zahteva, 1PV5 je izbačen.

Nedostaci IP-ja

Originalna namena Interneta je bila povezivanje kompjutera i razmena podataka. Zato su bili razvijeni protokoli koji su omogućili postizanje tog osnovnog cilja. Problem je što povezivanje kompjutera neće biti jedini cilj globalne mreže u budućnosti i zato moraju da se pojave novi protokoli koji će omogućiti ostvarivanje novih ciljeva. Kao što su personalni kompjuteri bili fenomen 80-ih godina prošlog veka, tako su multimedijalne i video aplikacije postale fenomen deceniju kasnije i početkom ovog veka. Zabava i digitalna tehnologija nastavljaju da se spajaju, namečući nove zahteve koje globalne mreže moraju da ispune. Servis plaćanja po gledanju je već dostupan kod kablovskih sistema, a dostupno postaje i prikazivanje videa na zahtev. Sve veći broj ljudi već uživa u real-time video igrama preko Interneta.

Mobilnost je sledeći aspekat razvoja. U trenutnoj Internet zajednici najveći deo host kompjutera nikada ne menja lokacije. Premeštaju se iz jedne kancelarije u drugu, ali to je problem lokalnog upravljanja. Iz perspektive Internet protokola, oni zadržavaju fiksne lokacije. Medutim, i to se menja. Mobilni kompjuteri i satelitske tehnologije obezbeđuju sredstva za komunikaciju dva uredaja u svim delovima sveta. Protokoli moraju da se razvijaju tako da omoguće povezivanje miliona parova uredaja sa proizvoljnih lokacija.

Neki ljudi vide tekuće tehnologije, kao što su mobilni telefoni, pejdžeri, PDA i prenosivi kompjuteri, kao uredaje koji će eventualno prerasti od uredaja za lične potrebe u uređaje koji mogu da ispune različite vrste zahteva. Iniciranje telefonskih poziva i prihvatanje poruka preko pejdžera predstavljaju sasvim uobičajene funkcije, ali mnogi smatraju da će doći dan kada će ti uređaji omogućavati mnogo više.

U kućama će možda postojati kompjuterizovani uređaji sa kojima će moći da se komunicira. Ako se vracate svom domu kasnije nego što ste planirali, mogli biste da iskoristite personalni komunikacioni uređaj za uključivanje svetla u kući, ili za uključivanje pećnice. Senzorski sistem može da pošalje signal ako ste zaboravili da zaključate sva vrata, ili da ugasite svetla, tako da biste mogli da to ispravite opet korišćenjem istog sistema.

Možda ćete moći da uključite uredaj u prenosivi kompjuter i da preuzimate fajlove sa bilo kog udaljenog kompjutera kojem imate pristup, bez korišćenja telefona. Sistem biste mogli iskoristiti i da nahranite svog sajber ljubimca koga ste ostavili kod kuće. Sve ove aplikacije bi zahtevale komunikacije nezavisne od mesta.

Bezbednost je sledeći problem. Internet protokol nije preterano siguran. Zbog toga su za većinu aplikacija veoma bitne lozinke, tehnike autentifikacije i firewalli. Ljudi prepoznaju da paketi koji stižu na Internet mogu da stignu doslovno iz bilo kog dela sveta i zato su za zaštitu resursa neophodne veoma jake mere zaštite. U poslednjih nekoliko godina svedoci smo raznih prevara i upada na privatne kompjutere, tako da je svima jasno da zaštita uvek mora da bude prioritet.

I pored svega toga, najvažniji cilj kod razvoja svakog novog protokola je mogućnost koegzistencije sa tekućim sistemima. Najveća prepreka za uvodenje novih tehnologija u računarstvo je osiguravanje konkurentnog izvršavanja sa postojećim tehnologijama. Kasnije bi, tokom godina,

aplikacije postepeno prelazile sa starih tehnologija na nove. Na globalnoj skali jednostavno ne postoji ni jedan drugi način da se to izvede.

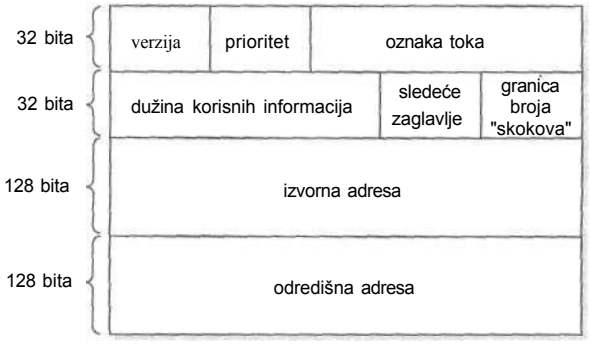
Ljudi razmišljaju o ovome većgodinama. Internet Engineering Task Force (IETF) počinje 1991. godine da razmatra problem promene postojećeg IP-ja i kreiranje nove generacije IP-ja, neformalno nazvane IPng (IP Next Generation - IP nove generacije). U pokušaju da se uključi kompjuterska zajednica, pozvani su razni profesionalci (istraživači, proizvodači, prodavci, programeri, itd) da daju svoje predloge.

Oformljeni komitet nazvan IPng Directorate razmotrio je predloge i mnoge odbacio, jer su bili namenjeni specijalnim zahtevima, ili su bili isuviše kompleksni. Međutim, jedan predlog koji je uključivao dizajn pod nazivom Simple Internet Protocol - SIP (ref. [De93]) bio je proširen tako da se iskoriste ideje opisane u drugim predlozima. Rezultujući protokol je dobio naziv Simple Internet Protocol Plus (SIPP).

IETF se 1994. godine sastao u Torontu i, na osnovu preporuka IPng Directorate, izabrao je SIPP kao osnovu za sledeću generaciju Internet protokola, koja je formalno trebalo da bude poznata kao IPv6 (IP verzija 6). Internet Engineering Steering Group nešto kasnije te godine daje odobrenje protokola i uvodi protokole u proces standardizacije u toku naredne godine. Daćemo opšti pregled IPv6 protokola; zainteresovani čitaoci mogu da u referencama [Hi96], [St96], [Br95], i [CoOO] pronađu više detalja.

Zaglavlja paketa

Razmatranje o IPv6 počinjemo proučavanjem formata koji se koristi za zaglavlje paketa. Tako dobijamo radni okvir u kome možemo da objasnimo neke opcije koje IPv6 obezbeduje i da ukažemo na razlike u odnosu na tekuću verziju Internet protokola. Na slici 11.18 prikazano je zaglavlje 1PV6 paketa. U poređenju sa IPv4 paketom sa slike 11.5, odmah se uočavaju dve razlike. IPv6 adrese imaju više bitova, a u zaglavlju postoji manje opcija. To možda deluje kontradiktorno sa ciljem da se obezbede dodatne mogućnosti, ali to nije tako, kao što ćete uskoro videti.



SLIKA 11.18 *Zaglavlje lPv6 paketa*

Polje *Version* je dužine četiri bita i identifikuje verziju IP-ja koju ovaj paket predstavlja (vrednost 4 za tekuću verziju IP-ja, a 6 za novu).

Polje *Priority* takode ima četiri bita i izuzetno je korisno za kontrolu zagušenja. Koncept prioriteta je jednostavan: više vrednosti ukazuju na značajnije pakete. Bitno je kako se prioriteti koriste. Već smo objasnili zagušenje i neke načine za njegovo rešavanje. IPv6 prepoznaje da su kašnjenja u nekim aplikacijama, kao što je email, često i neprimetna, dok kašnjenja u nekim daigim aplikacijama, kao što su multimedijalne, čine gledanje skoro nemogućim. Trik je u tome da se identifikuje koji paketi odgovaraju kojim aplikacijama.

Sajt sa koga se šalju IP paketi može da iskoristi ovo polje za definisanje značaja paketa u odnosu na ostale pakete koji se šalju sa istog mesta. Vrednosti prioriteta se nalaze izmedu 0 i 7 i odgovaraju paketima koji se mogu zadržavati malo duže radi rešavanja zagušenja. IPv6 preporučuje vrednosti u zavisnosti od aplikacije. Na primer, email ima prioritet 2, FTP i HTTP 4, Telnet 6, a SNMP 7. Kao što možete da vidite, više vrednosti odgovaraju aplikacijama kod kojih su kašnjenja primetna. Vrednosti iznad 7 odgovaraju real-time, ili multimedijalnim aplikacijama, slučajevima kada kašnjenja mogu da budu veoma neprikladna, do krajnje neprihvatljivosti. Na primer, prezuimanje zvučnih, ili video fajlova radi prikazivanja u realnom vremenu zahteva izuzetno mala, ili nikakva kašnjenja. Naravno, ako kašnjenja mogu nekako da se sinhronizuju sa potrebama gledaoca da predahnu, onda ipak mogu da se tolerišu.

24-bitno polje *Flow Label* koristi se zajedno sa poljem Priority. Ideja je da se identifikuju paketi koji zahtevaju "specijalni tretman" u ruterima. Normalno rukovanje zahteva od rutera da pretraže svoje tabele rutiranja pre nego što proslede pakete. Pošto se te tabele menjaju vremenom, paketi sa istim odredištem mogu da "putuju" preko različitih ruta.

IPvG definiše tok *(floiv)* kao sekvencu paketa koji se šalju od izvora do jednog odredišta, kao odziv na neku aplikaciju. Ako su ti paketi dizajnirani za prikazivanje u realnom vremenu na odredištu, specijalni tretman može da podrazumeva njihovo rutiranje na isti način (i brzo) kako bi se garantovao dolazak u ispravnom redosledu. U stvari, ovo prilično podseća na virtuelno kolo. Da bi se uspostavio tok, izvor generiše nasumično izabrani nenulti broj i smešta ga u polje Flow Label svih paketa u tom toku. Ruter primenjuje hash funkciju na broj toka kako bi se izračunala lokacija na kojoj se nalaze informacije o tome kako paket treba tretirati. Instrukcije o specijalnom tretmanu su postavljene pre slanja paketa. Ruter koristi hash funkciju zato što ona u opštem slučaju predstavlja najbrži način da se nešto pronade. Nasumični brojevi su korišćeni zato što dovode do manjeg broja kolizija kada se na njih primene hash funkcije. Ipak, kao krajnji zaključak može da se naglasi da ruter treba da pošalje pakete što pre.

16-bitno polje *Payload Length* predstavlja broj bajtova u paketu minus 40. Pošto je zaglavlje dugačko 40 bajtova, ovo polje defmiše koliko značajnih bitova sledi iza njega. Nemojte da mislite da dužina polja sa korisnim informacijama označava broj bajtova sa podacima. Ako postoji samo jedno zaglavlje, onda je takav zaključak ispravan, ali iza prvog može da sledi više drugih zaglavlja.

Polje *Hop Limit* u suštini ima istu funkciju kao i polje Time to Live kod IPv4 paketa.

Ovo nas dovodi do 8-bitnog polja *Next Header,* koje predstavlja značajnu razliku u poređenju sa IPv4 paketima. Tekuće zaglavlje IPv4 paketa sadrži polja Options i Protocol, pomoću kojih se naznačava kada ruter treba da preduzme odredene akcije. Pošto se u polje sa tim nazivom

ugraćtuju različite opcije, svaki ruter mora da parsira zaglavlje paketa (specijalno polje Option) kako bi utvrdio da li postoje opcije koje mogu da utiču na njegove odluke. To zahteva dodatnu logiku i vreme koje ruter mora da izdvoji, što usporava ceo proces rutiranja.

Da bi se omogućilo navođenje različitih opcija, IPv6 ima *zaglavlje proširenja (extension header),* dodatno zaglavlje izmedu zaglavlja sa slike 11.18 i korisnih informacija paketa (podataka). Svako dodatno zaglavlje ima i polje Next Header, koje defmiše tip dodatnog zaglavlja koje sledi (ako postoji). Ovo omogućava nekoliko zaglavlja proširenja koja se postavljaju izmedu originalnog zaglavlja i korisnih informacija paketa; svako od njih ukazuje na različitu opciju. Ako nema zaglavlja proširenja, onda, poput polja Protocol u IPv4 zaglavlju, polje Next Header defmiše transportni protokol (na primer, TCP, ili UDP) koga IPv6 koristi. Možda se najznačajnijim a spektom ovog uredenja može smatrati to što će neka dodatna zaglavlja ruteri ignorisati. Tako će ruteri moći brže da prosleduju pakete.

Sadržaj i format svakog dodatnog zaglavlja zavisi od njegovog tipa. Ovde predstavljamo šest tipova:

- **Destination options header (Zaglavlje sa opcijama** za **odredište)** Ovo zaglavlje obezbeduje informacije za odredište. Ne koristi se za vreme rutiranja.
- **Fragmentation header (Zaglavlje fragmentacije)** Ovo zaglavlje obezbeduje informacije za slučaj da je neophodno ponovo sastaviti fragmente paketa. Kao takvo, ono sadrži stavke kao što su ofset fragmenta, bit Last Fragment i identifikator koji je jedinstven za originalni paket. Ovo je umnogome slično fragmentaciji i procesu ponovnog sastavljanja kod IPv4. Ipak, postoji jedan značajna razlika. Prelazni IPv4 ruteri su mogli da fragmentiraju dolazeće pakete ako su bili suviše veliki. IPv6 ne dopušta fragmentiranje paketa u prelaznim ruterima. Ovo je značajno zbog toga što uprošćava logiku U ruteru i doprinosi efikasnijem i bržem rutiranju. Naravno, nameće se logično pitanje šta se dešava ako ruter dobije paket koji je suviše veliki da bi bio poslat preko mreže. U tom slučaju ruter jednostavno odbacuje paket i šalje poruku (preko ICMP-a) nazad do izvora. Ta poruka ukazuje da je paket bio suviše veliki i naznačava se maksimalna dopuštena veličina. Izvor će nakon toga fragmentirati paket i poslati fragmente, koji sadrže zaglavlje fragmentacije. Fragmenti se ponovo sastavljaju na odredištu.
- **Hop-by-hop header (Zaglavlje za pojedinačne "skokove")** Ovo zaglavlje, ako postoji, mora da se prouči u svakom ruteru. Ideja je da se navedu sve informacije koje moraju da imaju svi ruteri. Postoji nekoliko mogućih opcija. Pošto je dužina polja Payload Length 16 bitova, maksimalna velidna paketa je 64 KB. Ovo zaglavlje dopušta *džambo pakete,* pakete veće od 64 KB. Ovo je korisno prilikom prenosa velikih količina podataka, kao u slučaju video zapisa. Sledeca opcija je olakšavanje RSVP protokola, gde paketi sadrže informacije o rezervisanom propusnom opsegu koji mora da se obezbedi u svakom ruteru.
- **Routing header (Zaglavlje za rutiranje)** Ovo zaglavlje obezbeduje dodatne informacije o rutiranju, kao što je opcija Loose Route kod IPv4. Odnosno, sadržaće 128-bitne adrese rutera preko kojih ovaj paket mora da pređe.

- Security header (Bezbednosno zaglavlje) Ovo zaglavlje ukazuje na činjenicu da su korisne informacije paketa šifrovane.
- Authentication header (Zaglavlje autentiflkacije) Ovo zaglavlje služi za autentifikaciju paketa koja se koristi sa IPSec, bezbednosnim protokolom na nivou paketa.

IPSec

Dugogodišnje kritike na račun Interneta odnosile su se na nedostatak zaštite. Sve bezbednosne mere su morale da se implementiraju na višim slojevima i da se ugovore pre bilo kakve razmene informacija. Zato je IETF razvio IPSec, protokol dizajniran za obezbeđivanje bezbednih prenosa na nivou paketa. IPSec se izvršava direktno iznad IP-ja, ali ispod bilo kog transportnog protokola sloja 4, kao što je TCP. Zato su sve IPSec bezbednosne mere transparentne za protokole sloja 4 i viših slojeva.

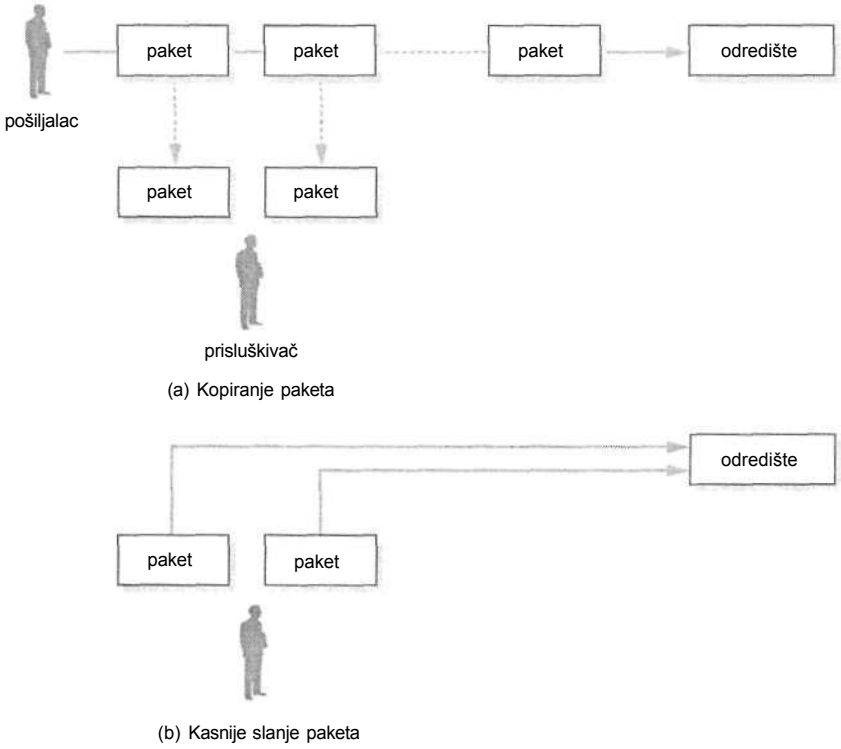
Postoji prednost implementiranja zaštite i autentifikacije na ovom nivou. Na primer, pretpostavimo da se aplikacija oslanja na SSL, ili TLS, radi autentifikacije. Pretpostavimo da na sloju 4 TCP razmenjuje segmente koristeći mehanizme za kontrolu toka (to jeste tako, kao što ćete uskoro videti). "Uljez" može da pošalje lažni TCP segment u niz segmenata. Ako TCP segment ima validan broj sekvence, TCP ga prihvata i prosleduje do SSL, ili TLS, radi verifikacije. Ako je segment bio falsifikovan, bezbednosni sloj ga detektuje. Ipak, TCP sada traži sledeći TCP segment. Kada pravi segment stigne, TCP ga odbacuje. Da se autentifikacija desila na nižem sloju, ovo se ne bi dogodilo.

IPSec ima tri glavne komponente: zaglavlje autentifikacije (AH) koje obezbeđuje autentifikaciju paketa, Encapsulating Security Payload (ESP) za šifrovanje paketa i autentifikaciju i protokol za razmenu ključa. Tako su moguće različite kombinacije autentifiakcije, šifrovanja i servisa za razmenu ključa. Specifična kombinacija zajedno sa odredišnom adresom defmiše *bezbednosnu asocijaciju (security association)* i način na koji se bezbedne informacije prenose i kome se prenose.

AH može da postoji kao jedno od dodatnih IPv6 zaglavlja, ili može da sledi iza IPv4 zaglavlja. U poslednjoj varijanti polje Protocol u IPv4 zaglavlju ima vrednost Sl kako bi se ukazalo na prisustvo AH. AH sadrži sledeća polja:

- Security Parameter Index (SPI) Ovo polje sadrži kod koji, zajedno sa odredišnom adresom, definiše bezbednosnu asocijaciju. Korišćenje različitih vrednosti polja SPI omogućava biranje između nekoliko bezbednosnih opcija i opcija za autentifikaciju.
- Sequence number Svaki paket ima jedinstveni broj sekvence za određeno SPI polje. Njegova svrha je slična onoj koja je opisana u Poglavlju 8 - ne prihvatati redundantne pakete. Ovo je, u suštini, kontrola toka za pakete sa konkretnom vrednošću polja SPI. Ipak, koristi se zbog različitih razloga da se spreče napadi odgovorima (replay attacks). Oni se dešavaju kada prisluškivač nadgleda saobraćaj i kopira pakete (packet sniffing) koji su u tranzitu (slika 11.19a). Ako su ti paketi poslati radi uspostavljanja konekcije, ili obezbedivanja pristupa nekom servisu, ponovno slanje (slika 11.19b) omogućava prisluškivaču da uradi isto, zaobilazeći normalne protokole. Na primer, paketi mogu da sadrže korisničko ime i lozinku koji se šalju radi logovanja na

udaljeni sistem, ili radi pristupa ličnim podacima. Prisluškivač može da pokuša da imitira istu akciju kasnije ponovnim slanjem tih paketa kako bi dobio pristup.



SLIKA 11.19 *Napad slanja odgovora*

Ako su paketi sekvencirani u okviru SPI-a, svi ponovo poslati paketi sadrže duplikate brojeva sekvenci i odbacuju se. Naravno, kriminalac može jednostavno da pokuša sa brojevima druge sekvence, ali to bi se onda detektovalo pomoću mehanizama za autentifikaciju.

- **Authentication Data** Ovo polje sadrži hash vrednost koja se izračunava u zavisnosti od zaglavlja i sadržaja paketa. Od ovog izračunavanja se izuzimaju sva polja zaglavlja čiji se sadržaj može menjati u toku rutiranja. Na primer, to se kod IPv4 odnosi na polje Time to Live, čeksume i oznake toka.

ESP izvršava isti servis kao i AH, ali obezbeđuje poverljivost šifrovanjem podataka. Kao i AH, može da postoji u vidu dodatnog IPv6 zaglavlja, ili može da sledi iza IPv4 zaglavlja. Takode sadrži SPI, broj sekvence i polje Authentication Data. Osim toga, sadrži polje Payload Data u kome se nalaze šifrovani podaci. Metod šifrovanja i ključ su definisani bezbednosnom asocijacijom i, samim tim, vrednošću SPI. Postoji i polje Pad, jer mnogi algoritmi za šifrovanje primenjuju šifrovanje na nivou blokova i zahtevaju ceo broj blokova. Polje Pad se koristi ako poslednji podaci ne popunjavaju ceo blok.

IPSec upravljanje ključem se sastoji iz nekoliko delova, a nastalo je iz originalne inkarnacije. Prvi deo je **Internet Security Association and Key Management Protocol (ISAKMP).** On ne definiše konkretne metode za razmenu ključa, već formate paketa i pravila za razmenu paketa u kojima se nalaze informacije o ključu. Jedan algoritam za razmenu ključa koji je razvijen za korišćenje sa ISAKMP je Oaldey Key Management Protocol.

Oakley je zasnovan na Diffie-Hellman algoritmu, ali uključuje neke protokole za autentifikaciju koji se izvršavaju pre razmene ključa. U skorije vreme on je prerastao u *Internet Key Exchange* (IKE) mehanizam. IKE koristi javni i privatni ključ za autentifikaciju i generisanje ključeva sesije. Nakon toga, ključevi sesije se koriste za šifrovanje sadržaja paketa.

Poglavlje 7 je posvećeno većim delom takvim aktivnostima i zato ih ovde nećemo ponavljati. Zainteresovani čitaoci mogu da potraže više detalja i analizu IKE u referenci [PeOO].

IPv6 adresiranje

Poslednja dva polja u zaglavlju IPv6 paketa su izvoma i odredišna adresa, koje su jasne same po sebi. Ono što nije jasno na prvi pogled jesu tipovi adresa koje IPv6 definiše i neki problemi u vezi kompatibilnosti sa IPv4 adresiranjem. Najočiglednija razlika u odnosu na IPv4 je to što su IPv6 adrese 128-bitne, četiri puta duže od IPv4 adresa. Teorijski je omogućeno 2^{128} *s* 10^{40} različitih adresa. Jedan od problema sa eksponencijalnom notacijom je to što je često teže razumeti koliko je to veliko, nego zapisati broj. Da biste lakše razumeli koliko je veliko 2^{128}, u referenci [Hi96]izvršena su neka izračunavanja i pokazano je da ako bi se sve adrese rasporedile ravnomerno po površini cele zemaljske kugle, postojale bi 1.024 adrese na svakom kvadratnom metru, što je više nego dovoljno za svaku osobu, glistu i insekta na planeti.

Adrese se svrstavaju u tri opšte kategorije: unicast, anycast i multicast. *Unicast adresa* definiše jedinstveni interfejs. Postoji nekoliko njenih tipova, koje ćemo predstaviti ubrzo. *Anycast adresa* defmiše grupu interfejsa. Paket sa anycast odredišnom adresom može da se isporuči jednom interfejsu u bilo kojoj grupi (obično u onoj koja je najbliža izvoru). *Multicast adresa* definiše grupu, ali u ovom slučaju paket prolazi kroz svaki interfejs u grupi.

Notacija 128-bitnih adresa se razlikuje od one koja se koristi za IPv4. Korišćenje tekuće notacije u kojoj se tačkama razdvajaju trocifreni brojevi dalo bi notaciju koja sadrži 16 trocifrenih brojeva razdvojenih tačkama. Naravno, ovo postaje pomalo nezgrapno. Umesto toga, tačke se menjaju dvotačkama i svakih 16 bitova u adresi predstavlja heksadecimalnu notaciju četvorocifrenog broja. Primer IPv6 adrese ima sledeći oblik

7477:0000:0000:0000:0000:0AFF: 1BDF:7FFF

Svaka heksadecimalna cifra u ovoj reprezentaciji ima jedinstveni 4-bitni ekvivalent. Rezultat je i dalje nezgrapan, ali je bolji od alternative.

Za adrese koje sadrže mnogo nula (a sa 2^{128} adresa biće ih dosta) koristi se skraćena notacija. U suštini, nule se ne navode, već se na njihovo prisustvo ukazuje sa dve dvotačke (::). Stvarni broj nula koje nedostaju izračunava se oduzimanjem broja heksadecimalnih cifara u notaciji od 32, broja heksadecimalnih cifara koje su potrebne za punu 128-bitnu reprezentaciju.

Na primer, prethodna adresa bi bila zapisana kao

7477::0AFF:1BDF:7FFF

Pošto ova notacija sadrži 16 cifara, znamo da mora da nedostaje 16 nula. U slučajevima kada adresa počinje sa 0, notacija započinje dvotačkom. Drugim rečima, adresa

0000:0000:0000:0000:0AFF: 1BDF:000F:0077

može da se zapiše i kao

OAFF: 1BDF:000F:0077

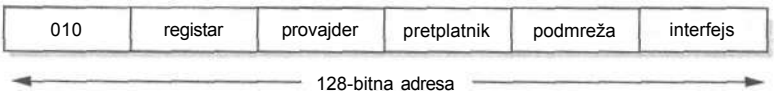
Da bi se adrese dalje pojednostavile, vodeće nule u okviru četvorocifrene grupe ne moraju da se navode. Ovo omogućava uprošćavanje notacije na sledeći način:

::AFF: 1BDF:F:77

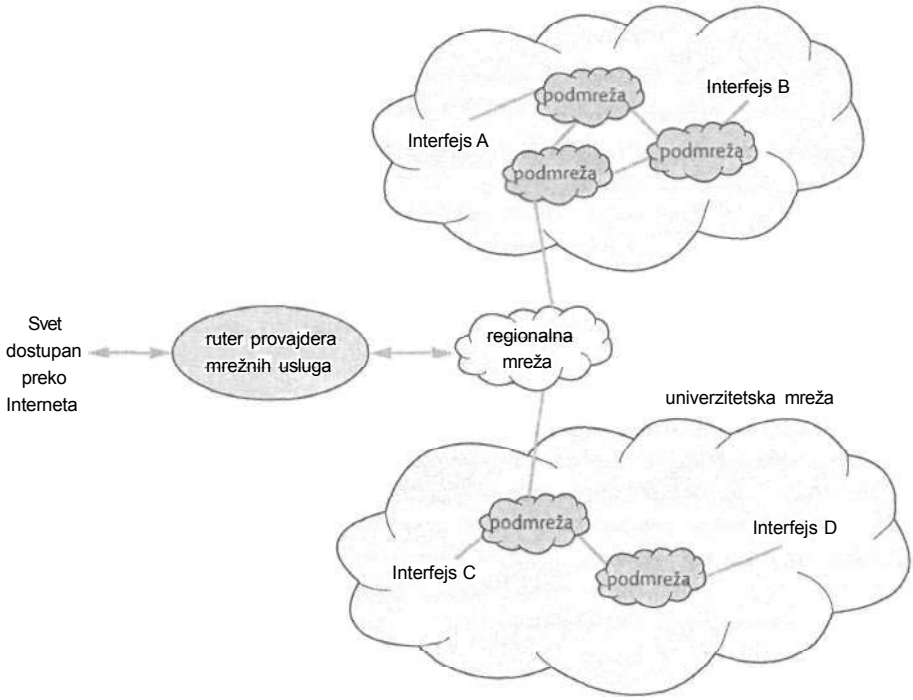
Kao što IPv4 deli svoje adrese na različite klase u zavisnosti od vodećih bitova, to radi i IPv6. Trenutno postoje 22 različita tipa adresa; svaki ima jedinstven bitski prefiks. Prefiksi mogu da sadrže od tri do 10 bitova. Na primer, adresa koja počinje sa osam nula odgovara IPv4 adresi (u stvari, IPv4 adrese počinju sa mnogo više nula, ali o tome će biti više reči nešto kasnije). Adrese koje počinju sa osam nula su multicast adrese. Adrese koje počinju sa 0000 010 kompatibilne su sa Novellovim IPX protokolom. Većina preddefinisanih prefiksa još nije dodeljena i rezervisana je za dalji rast. Ovo ostavlja oko 85 odsto adresa za buduće namene.

Posebno su interesantne unicast adrese, koje su najsličnije IPv4 adresama. Na slici 11.20 prikazan je format unicast adrese. Kao i kod IPv4 adrese, postoji hijerarhija; samo je još složenija. Prva tri bita unicast adrese su 010. Preostali bitovi defmišu hijerarhiju sa pet nivoa. Primer topologije sa slike 11.21 ilustruje moguću organizaciju hijerarhije.

U opštem slučaju **Internet provajder (ISP - Internet service provider)** sklapa ugovore sa korisnicima za obezbeđivanje pristupa Internetu. Korisnici odredenog provajdera mogu da se povezuju preko neke regionalne mreže koja obuhvata grad, okrug, ili nekoliko okruga, u zavisnosti od stepena naseljenosti. U nekim slučajevima korisnici mogu da budu velike kompanije, ili univerziteti koji poseduju sopstvene hijerarhijeske strukture. Kao što je istaknuto u odeljku 10.7, mogu da podele svoju mrežu na podmreže, kojima su dodeljeni jedinstveni identifikatori.



SLIKA 11.20 *Format unicast adrese*



SLIKA 11.21 *IPv6 organizacija*

Te podmreže će obezbediti eventualni pristup ljudima koji koriste servise.

Pošto postoji ogroman broj ISP-a, deo adrese sa slike 11.20 identifikuje provajdera. Tako je rutiranje koje se izvodi da bi se pronašao odgovarajući provajder nezavisno od provajderovih korisnika. Provajder svojim korisnicima dodeljuje pretplatničke ID-ove. Neke velike kompanije mogu da defmišu sopstvene podmreže. Svaka će imati svoj broj podmreže, a pune IPv6 adrese svih podmreža sadrže isti pretplatnički broj. Konačno, podmreže povezuju korisnike. Svaki korisnik na istoj podmreži ima drugačiji ID interfejsa, ali svi imaju isti broj podmreže, pretplatnički broj i ID provajdera. Sve ovo je, u stvari, proširenje strategija hijerarhijskog rutiranja objašnjenih u odeljku 10.7.

Konačno, JD *registra* na slici 11.20 odnosi se na medunarodne i kontinentalne granice. Kanadski regio održava listu autorizovanih provajdera u Kanadi. Slični registri postoje i u Sjedinjenim Američkim Državama, Evropi i tako dalje. Zato se ID registra nalazi na vrhu hijerarhije i omogućava donošenje odluka o rutiranju na osnovu geografskih lokacija. Ovo pomaže paketima poslatim iz Kanade u Sjedinjene Američke Države da izbegnu rutere u Evropi.

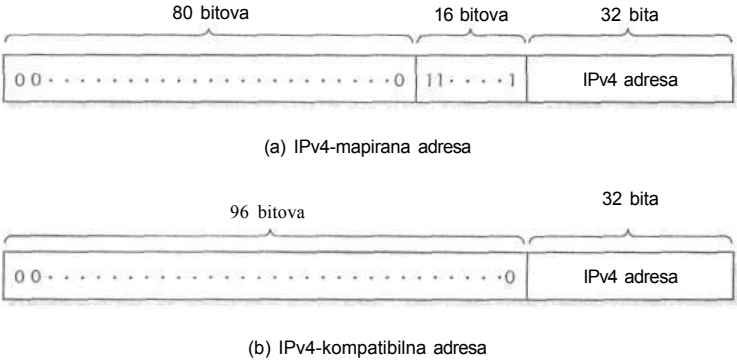
Kompatibilnost sa IPv4

Više miliona kompjutera komunicira pomoću IPv4. Ovako veliki broj onemogućava konvertovanje u IPv6 preko noći, ili u toku vikenđa. Poteškoće koordinacije, neizbežni problemi i cene čine ovaj scenario nemogućim. Da bi se prešlo na IPv6, biće potrebno više godina, zato što individualni sajtovi moraju da defmišu vreme koje će im biti potrebno za implementaciju. Zato i IPv4 i IPv6 ruteri moraju da koegzistiraju i da održavaju sve neophodne konekcije. IPv6 protokoli su dizajnirani tako da prepoznaju IPv4 protokole. Sa druge strane, IPv4 protokoli su dizajnirani pre IPv6 i ne znaju ništa o njemu. Ovo stvara neke probleme koji moraju da se reše.

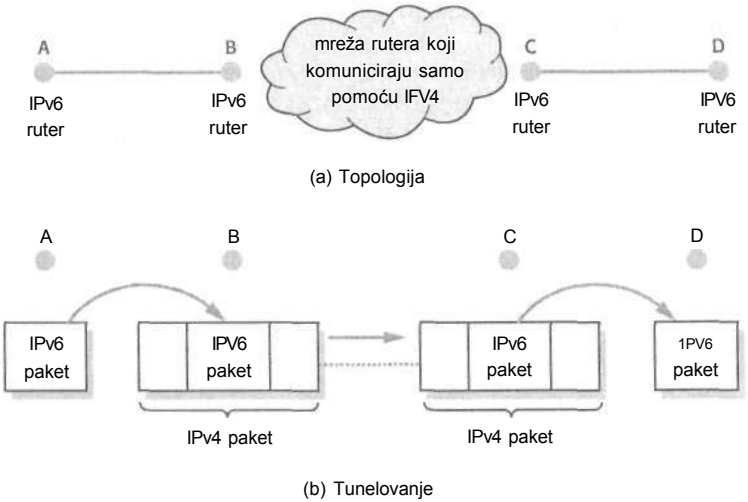
Da bi bili rešeni ti problemi, struktura IPv6 adresa dopušta tipove IPv4 adresa. Na slici 11.22 prikazana su dva načina za postavljanje JPv4 adrese u strukturu IPv6 adrese. Pošto su 1PV6 adrese veče od IPv4 adresa, veoma je teško pronaći način da se smeste IPv4 adrese. Teži deo je navesti dva protokola da zajedno rade. Na slici 11.22a prikazana je IPv4-mapirana adresa. Pretpostavimo da paket mora da prođe preko mreže IPv6 rutera da bi stigao do IPv4 odredišta. Jedna opcija je da se ruteri prebace na IPv4 kada je u odredištu JPv4 čvor. Pošto je namena IPv6 da poboljša)Pv4, to nije dobra opcija.

Ako ruteri pokreću 1PV6, kako znaju da je adresa za 1PV4 čvor i kako da se ne interpretira u skladu sa IPv6 hijerarhijom? Ovde ponovo nastupaju tipovi adresa. Osam nula i 16 jedinica označavaju IPv4 adresu. Zato ruteri interpretiraju adrese u skladu sa IPv4 pravilima, što je suprotno od onoga što se predlaže na slici 11.20.

Šta je sa suprotnim problemom? Pretpostavimo da dva IPv6 rutera moraju da razmene pakete, ali jedina ruta vodi preko IPv4 rutera (slika 11.23a). Pošto 1PV6 adrese sadrže informacije koje se ne mogu smestiti u IPv4 paket, izgleda da imamo ozbiljan problem. Srećom, postoji rešenje - tunelovanje. Na primer, pretpostavimo da A, B, C i D sa slike 11.23a koriste IPv6 i da A treba da pošalje paket do D.



SLIKA 11.22 *IPv4 adrese u Wv6 formatu*



SLIKA 11.23 *Rutiranje IPv6 paketa preko IPv4 mreža*

Nažalost, jedini način da se stigne od B do C je preko mreže čiji čvorovi razumeju samo IPv4. Pošto je adresa za D 128-bitna, kako da propustimo taj paket preko mreže koja nema koncept 128-bitnih adresa?

Kada B dobije IPv6 paket od A, on ga ugrađuje u IPv4 paket (slika 11.23b) koji je namenjen za C. IPv4 protokoli rade ono što je neophodno da se IPv4 paket prenese od B do C. Kada C dobije paket, izvlači IPv6 paket iz IPv4 paketa. IPv6 rutiranje se efektivno nastavlja kada C pošalje paket do D. Iz perspektive IPv4 rutiranje na ovoj mreži se ne razlikuje od onog koje smo ranije prikazali. Iz perspektive IPv6 postojao je jedan link od B do C (tj. *tunel).*

Sada se možda pitate, ako je C IPv6 ruter, kako da se njegova adresa postavi u IPv4 paket. Ovo je deo prelaza. Svaki čvor na bilo kom kraju tunela ima dodeljenu IPv4-kompatibilnu adresu, kao na slici 11.22b, pored IPv6 adrese. Zato se lako smešta u zaglavlje IPv4 paketa.

Zaključak

IPv6 izvršava iste primarne funkcije kao i IPv4, ali pruža mogućnost rutiranja bez uspostavljanja konekcije. Ipak, ima dodatne mogućnosti, kao što su autentifikacija i šifrovanje, kojih nije bilo u IPv4. Osim toga, značajno povećava adresni prostor i uprošćava zaglavlja kako bi rutiranje bilo što efikasnije. I drugi faktori doprinose efikasnijem rutiranju sa IPv6. Fragmentiranje i ponovo sastavljanje paketa više se ne izvodi u međučvorovima. Kod IPv4 nema mogućnosti za detekciju grešaka (nema čeksume), za razliku od IPv4, čime su ruteri oslobođeni ovog vremenski zahtevnog posla. Uklanjanje čeksume ne predstavlja nikakav značajan gubitak. Većina mreža je veoma pouzdana. Osim toga, protokoli nižeg sloja veze i, kao što ćete videti, protokoli višeg sloja imaju metode za detekciju grešaka.

11.4 Transportni protokoli

Do sada smo se bavili prvenstveno mrežnim operacijama. Formati okvira, rutiranje, kontrola zagušenja i adresiranje imali su suštinski značaj za omogućavanje komunikacije izmedu dva uređaja. Međutim, podjednako je značajno *kako* oni "razgovaraju". *Transportni protokol* je protokol najnižeg sloja koji defmiše šta jedan uredaj može da "kaže" drugom u ime korisnika. Niža tri sloja definišu kako mreža funkcioniše; transportni sloj je prvi sloj koji defmiše protokol krajnjeg korisnika.

Postoje sličnosti sa ranije prikazanim protokolima veze po tome što je fokus postavljen na način na koji se informacije razmenjuju izmedu dva entiteta. Medutim, protokoli veze definišu komunikacije medu uredajima izmedu kojih postoji fizička konekcija. Konekciji orijentisani transportni protokoli defmišu komunikacije između sajtova sa logičkim konekcijama, često na većim rastojanjima. Postoje i transportni protokoli koji se izvršavaju bez prethodnog uspostavljanja veze, ali za sada ćemo razmotriti samo konekciji orijentisani protokol.

Transportni sloj obezbeduje i "konekciju" koju korisnik opaža. Na primer, korisnici mogu da se uloguju na kompjutere koji su smešteni na udaljenim lokacijama, tako da imaju utisak da su se povezali. Ali, konekcija nije uspostavljena u fizičkom smislu kao kada se koriste kablovi, ili kada se iniciraju telefonski pozivi. Nije neophodno da postoji kolo koje je isključivo namenjeno za prenos informacija između korisnika i kompjutera.

Transportni sloj može da obezbedi utisak konekcije, tako što se ponaša kao interfejs između korisnika i mrežnih protokola. Ovo je slično sekretarici koja u ime svog direktora poziva klijente. Sekretarica dobija zahtev od direktora, poziva traženog klijenta, pronalazi tu osobu i tako uspostavlja konekciju.

Nakon toga, direktor može da razgovara bez problema koje je sekretarica eventualno imala dok jetražila tu osobu, koja je možda u tom trenutku možda bila na nekom značajnom sastanku, na pauzi za ručak, ili igrala tenis. Kada direktor završi razgovor, sekretarica može da okonča konekciju pribavljanjem dodatnih bitnih informacija, kao što su klijentova adresa, broj telefona, ili lokacija teniskog terena.

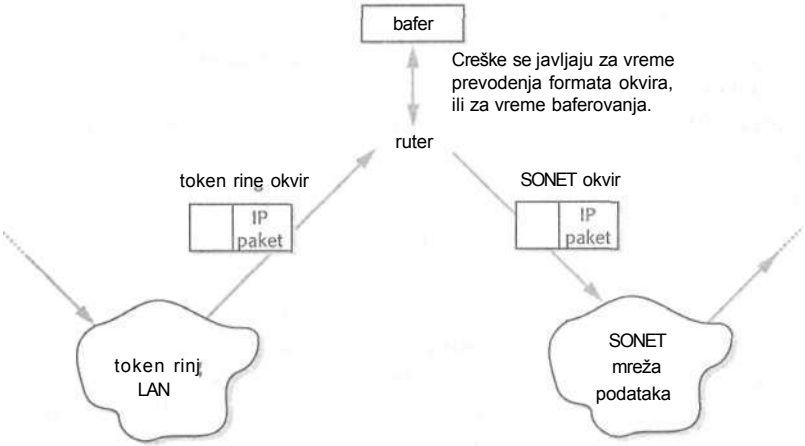
Transportni protokol obavlja mnogo više od uspostavljanja i raskidanja konekcija. Najniža tri sloja obezbeđuju sredstva za povezivanje različitih uredaja, ali transportni sloj je najniži sloj, koji, u stvari, omogućava efikasnu i bezbednu komunikaciju između korisnika. Medu važnije funkcije transportnog sloja ubrajaju se:

- Upravljanje konekcijom Ova funkcija definiše pravila koja omogućavaju korisniku da započne razgovor sa drugim korisnikom ako su direktno povezani. Defmisanje i postavljanje konekcije poznato je i kao **usaglašavanje (handshaking).**
- **Kontrola toka** Transportni sloj ograničava količinu informacija koju jedna stanica može da pošalje do druge bez prijema neke potvrde. Ako Vam ovo zvuči poznato, dobro je! Sećate se nekih informacija iz prethodnih poglavlja. U Poglavlju 8 smo objasnili kontrolu toka i njen značaj za sloj veze. Možda izgleda čudno što se kontrola toka ponovo javlja na transportnom sloju, ali zapamtite da taj sloj mora da funkcioniše nezavisno od nižih slojeva. Niži slojevi mogu da omoguće kontrolu

toka u većoj, Oi manjoj meri (ili nimalo). Da bi se očuvala nezavisnost, transportni sloj može da koristi sopstvenu kontrolu toka. Ovo može da deluje redundantno, ali nezavisnost često uslovljava redundantnost. Osim toga, transportni sloj defmiše kontrolu toka između krajnjih korisnika. Protokoli sloja veze defmišu kontrolu toka između prelaznih, ali susednih entiteta.

- Detekcija grešaka Ovo je sledeći slučaj u kome deluje kao da se dupliraju karakteristike nižeg sloja. Neke greške ipak uspevaju da prođu detekciju na nižem sloju. Ovo zvuči neuobičajeno, jer to znači da, iako detekcija grešaka na sloju veze obezbeđuje pouzdan prenos preko svakog linka, i dalje nema garancija za prenose bez grešaka izmedu izvora i odredišta. Kako je ovo moguće? Razmotrite ruter sa slike 11.24 (uzet sa slike 11.2). Pretpostavimo da prima netaknuti IP paket, ali u vreme reformatiranja okvira koji sadrži paket javlja se greška koja utiče na sadržaj paketa. Bilo kakve softverske, ili hardverske greške mogu da stvore probleme. Pošto se čeksuma izračunava nakon što se novi okvir kreira, uključiće pogrešne podatke. Strogo govoreći, to nije greška u prenosu, zato što se nije desila dok je paket bio u vlasništvu rutera. Ali, pokušajte da kažete to transportnom korisniku, koji vidi da su njegovi podaci promenjeni u tranzitu. Mehanizam detekcije grešaka na transportnom sloju detektuje ovakvu grešku.
- Odziv na korisničke zahteve Primeri uključuju slanje i prijem podataka, kao i specifične zahteve. Na primer, korisnik rnože da zahteva veliku propusnost, mala kašnjenja, ili pouzdan servis. IP može dobro da se "izbori" sa tim zahtevima. Transportni sloj prosleduje korisnikove zahteve do IP-a.

Da rezimiramo: transportni protokol mora da obezbedi pouzdane komunikacije između krajnjih korisnika. Ovo je izuzetno značajno, zato što IP ne garantuje pouzdan servis. Transportni protokoli moraju da obezbede potvrde i tajmere koji će osigurati slanje i prijem svih korisničkih podataka. Kao i u slučaju detekcije grešaka, protokoli nižeg sloja mogu da utvrde kada su okviri izgubljeni u tranzitu.



SLIKA 11.24 *Greška koju nisu detektovale tehnike za detekciju na nižem sloju*

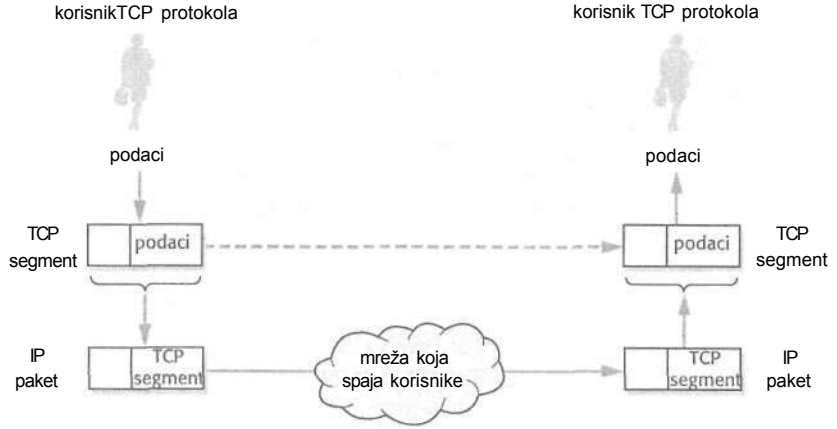
Ponovo želimo da se pouzdanost obezbedi nezavisno od nižih slojeva. Osim toga, pretpostavimo da je prelazni mrežni čvor izgubio okvir nakon što ga je primio i potvrdio, ali pre nego što je dalje preneo okvir. Kao što smo ranije pomenuli, ovo mogu da izazovu razne greške koje se javljaju na sajtu. Pošto nije bilo nikakve greške u bilo kojem point-to-point linku, na kraju će grešku detektovati protokol između krajnjih tačaka.

j

Transmission Control protokol (TCP)

Dva protokola transportnog sloja koja je DoD dizajnirao specijalno za pokretanje sa ARPANET IP protokolom su Transmission Control Protocol (TCP) i User Datagram Protocol (UDP). TCP je konekciji orijentisani protokol koji omogućava upravljanje konekcijama na Internetu. To je najšire korišćeni protokol transportnog sloja u celom svetu. UDP je protokol transportnog sloja koji funkcioniše bez uspostavljanja veze. Verovatno se ne koristi u istoj meri kao TCP, ali obezbeđuje transportne olakšice za značajne aplikacije, kao što su DNS i SNMP. U ostatku odeljka fokusiraćemo se na TCP, a zatim ćemo prikazati UDP. Da bismo kompletirali "priču", moramo da napomenemo da je ISO takođe definisao sopstveni transportni protokol za sloj 4. lntemet i TCP su toliko dominantni za defmisanje konekcija da će se DoD TCP verovatno još dugo koristiti.

TCP obezbeđuje konekciji orijentisani servis prenosa bajtova između dva korisnika. To znači da obezbeđuje logičku konekciju izmedu dva sajta i da može da prenosi sekvence bajtova između njih. Niz bajtova se deli na sekvencu segmenata, a zatim se šalje do odredišta preko varijacije protokola klizajućih prozora za kontrolu toka. Obezbeđuje inicijalno usaglašavanje uspostavljanjem, održavanjem i oslobadanjem konekcija. Rukuje zahtevima za pouzdanu isporuku informacija do odredišta, što je veoma bitno, zato što niži sloj (obično IP) ne ispomku paketa. TCP prima podatke, ili zahteve od svojih korisnika (slika 11.25), smešta ih u formatu TCP *sennenta* i predaje ih IP-u.



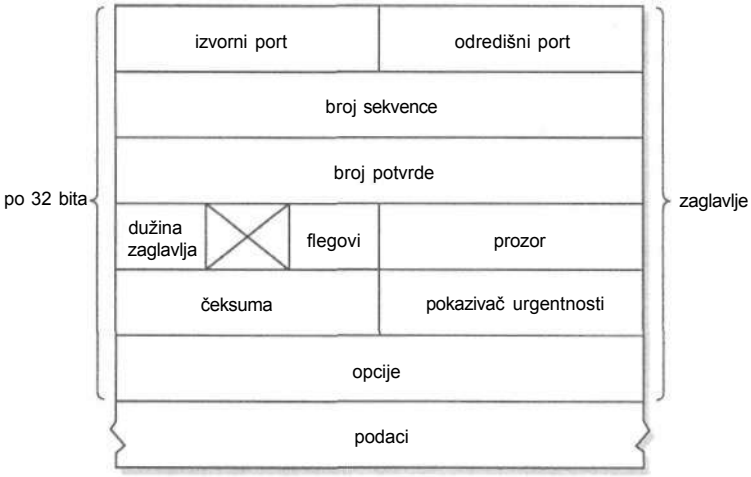
SLIKA 11.25 *TCP kao servis između krajnjih korisnika*

Ne igra nikakvu ulogu u rutiranju i transferu informacija. TCP na prijemnoj strani koji dobija segment reaguje na primljene informacije, izvlači podatke i daje ih korisniku. Iz perspektive TCP protokola nema mreže.

TCP segment

Na slici 11.26 prikazan je sadržaj TCP segmenta. U sledećoj listi defmišemo polja.

- **Destination port (odredišni port)** (16 bitova) Identifikuje aplikaciju kojoj je segment poslat. Ovo se razlikuje od IP adrese, koja defmiše Internet adresu. Pošto **se** u istom Internet čvoru istovremeno može pokretati više aplikacija, ovo polje identifikuje željenu aplikaciju. U opštem slučaju TCP proučava sve segmente koji stižu i razdvaja ih (demultipleksira) u skladu sa brojem porta. Informacije u segmentima sa istim brojem porta prenose se do iste aplikacije. Na strani pošiljaoca TCP se ponaša kao multiplekser, jer uzima informacije od razlidtih aplikacija, kreira segmente za svaku (sa odgovarajućim brojevima porta) i šalje ih do IP-ja radi eventualnog rutiranja. Brojevi portova ispod 1024 nazivaju se **opštepoznati portovi.** Dodelila ih je IANA i odgovaraju najčešće korišćenim aplikacijama. Na primer, port 53 odgovara DNS serveru, a portovi 21 i 23 su dodeljeni FTP-u i Telnetu, respektivno. IANA navodi portove numerisane od 1024 do 49151 kao registrovane portove - mogu da se koriste za programe koje pokreću obični korisnici. Da bi članovima Internet zajednice bilo što lakše, IANA je već dodelila mnoge portove iz ovog opsega; ipak, i dalje postoji veliki broj slobodnih portova koji se mogu slobodno koristiti.



SLIKA 11.26 *TCP segment*

U Poglavlju 12 ćemo pokazati kako možete da definišete sopstvene portove pomoću kojih je moguće definisati komunikacije izmedu aplikacija koje se pokreću na različitim sajtovima. Kompletna lista brojeva porta i njihovih dodela može da se pronade na Web sajtu www.iana.org/assignments/port-numbers.

- **Source port (izvorni port)** (16 bitova) Definiše aplikaciju koja šalje segment.
- **Sequence number (broj** sekvence) (32 bita) Svaki bajt u nizu koji TCP šalje je numerisan. Na primer, ako svaki segment podataka sadrži 1.000 bajtova podataka, brojevi sekvenci prvih bajtova u sukcesivnim segmentima bili bi x, x + 1000, x + 2000 i tako dalje, gde je x broj sekvence prvog bajta. * Ako segment sadrži podatke, ovo polje sadrži broj sekvence prvog bajta podataka u segmentu. Za razliku od drugih protokola koji sukcesivno numerišu svaki paket, ili okvir, TCP numeriše bajtove. TCP takode koristi ovo polje i kao deo inidjalne strategije za uspostavljanje konekcije (ubrzo ćemo i to objasniti). Korišćenjem 32 bita dobija se oko četiri milijarde brojeva sekvenci. Ovo je značajno zato što su šanse za dupliranje brojeva sekvenci eliminisane, što, ujedho, uklanja potrebu ograničenja veličine prozora, o kojoj smo govorili u Poglavlju 8, kada je bilo neophodno izbeći greške protokola.
- **Acknowledgment number (broj potvrde)** (32 bita) Sadrži broj sekvence bajta koga TCP na prijemnoj strani očekuje. Efektivno, on potvrduje sve primljene bajtove pre onog koji je naznačen. Ovo je slično potvrdama koje su prikazane u Poglavlju 8, osim što ovde protokol daje potvrde za bajtove, a ne za pakete.
- **Header length (dužina zaglavlja)** (četiri bita) Definiše veličinu TCP zaglavlja kao umnožak od po četiri bajta. Dužina zaglavlja može da bude različita, zbog promenljive dužine polja Options; ovo polje omogućava TCP protokolu na prijemnoj strani da zna odakle počinju podaci.
- Flags (flegovi) (šest bitova) Polje Flag naznačava kada druga polja sadrže smisaone podatke, ili defmišu neke kontrolne funkcije. Na primer, flegovi ACK i URG defmišu da li se **u** poljima Acknowledgment i Urgent Pointer nalaze smisaoni podaci. Slede opisi ostalih flegova:

  FIN (fmish - kraj) Ukazuje na poslednji TCP segment podataka.

  PSH (push - guranje) TCP obično donosi odluku kada segment sadrži dovoljno podataka da bi bio dopušten prenos. U nekim slučajevima aplikacija može da primora TCP da pošalje segment ranije izdavanjem komande Push. Na primer, interaktivna aplikacija može da izda komandu Push nakon što korisnik unese liniju sa tastature. Ovo daje bolji i "glatkiji" odziv od onoga kada TCP baferuje nekoliko linija ulaznih podataka pre nego što ih pošalje. Kada primi komandu Push, TCP postavlja polje PSH u segmentu. Ako je polje PSH postavljeno u dolazećem segmentu, prijemni TCP entitet čini sadržaj segmenta odmah raspoloživim za aplikaciju. Ako aplikacija omogućava prikazivanje dolazećih podataka na video ekranu, ovaj mehanizam obezbeđuje brz i "gladak" prikaz.

* Inicijalni broj sekvence ne mora nužno da bude 1, već se o njemu pregovara kada se konekcija uspostavi. To ćemo objasniti u odeljku o upravljanju konekcijom.

- RST (reset) Ovo je indikacija od entiteta na strani pošiljaoca da entitet na prijemnoj strani treba da raskine transportnu konekciju. Koristi se u slučaju nekih nenormalnih uslova i omogućava okončavanje konekcije od strane oba entiteta, zaustavljanje toka podataka i oslobadanje prostora u baferu koji je bio dodeljen konekciji.

  SYN (sinhronizacija) Koristi se kod inicijalnog uspostavljanja konekcije i omogučava sinhronizaciju oba entiteta (dogovor) oko inicijalnih brojeva sekvence.

- Window (prozor) (16 bitova) Ovo polje "govori" TCP entitetu koji prima ovaj segment koliko još bajtova podataka može da pošalje, osim onih koji su već potvrđeni. Kao što ćete uskoro videti, ovo otprilike odgovara veličini prozora kod protokola klizajućih prozora. Razlikuje se od ranije prikazanih protokola po tome što TCP entitet koji prima niz bajtova može da koristi ovo polje za promenu veličine prozora na strani pošiljaoca.

- Checksum (čeksuma) (16 bitova) Koristi se za detekciju grešaka na transportnom sloju. Algoritam čeksume interpretira sadržaj TCP segmenta kao sekvencu 16-bitnih celih brojeva i sumira ih. Ovo nije toliko strogo kao ranije prikazani algoritmi za detekciju grešaka iz Poglavlja 6 i zato su postojale neke kritike na račun ove detekcije.

- Urgent Pointer (pokazivač urgentnosti) (16 bitova) Ako je URG bit postavljen, segment sadrži urgentne podatke, što znači da TCP entitet na prijemnoj strani mora da isporuči podatke višim slojevima što pre. Urgentni podaci se nalaze na početku segmenta, a pokazivač urgentnosti ukazuje na prvi bajt koji sledi iza urgentnih podataka. Ovo omogućava prijemnoj strani da razlikuje urgentne od neurgentnih podataka. Neki su kritikovali polje Urgent Pointer tvrdeći da predstavlja slabu implementaciju urgentnosti. Istina, polje Urgent generiše specijalni tretman kada se segment primi, ali ne utiče na kontrolu toka koja je odgovorna za isporuku segmenta do prijemne strane. Ovo je pomalo slično situaciji u kojoj čekate na kraju dugačkog reda ispred šaltera, a, kada konačno dodete na red, zahtevate da odmah vidite šefa službe.

- Options (opcije) (Promenljiva veličina) Jedna opcija omogućava TCP entitetu da definiše maksimalnu veličinu segmenata koje prima od drugog entiteta. Ova vrednost se obično definiše za vreme inicijalnog uspostavljanja konekcije. Ovo je važna opcija i zbog toga što TCP može da povezuje dva kompjutera sa veoma različitim karakteristikama. Da bi se ostvarila uspešna komunikacija, moraju se uzeti ograničenja svih strana (kao što je, na primer, veličina bafera). Na primer, ne bi trebalo da veliki mainframe kompjuter "zatrpa" personalni kompjuter prevelikim segmentima. Personalni kompjuter će definisati maksimalnu velidnu segmenta koji može da prihvati kada se konekcija uspostavi. Pošto je veličina zaglavlja umnožak četiri bajta, polje Options se dopunjava tako da se zaglavlje završava na 32-bitnoj granici.

  Sledeća opcija omogućava TCP entitetima da se slože za 32-bitno polje prozora, umesto da se koristi 16-bitno. Ovo je izuzetno korisno kada se za prenos velikih fajlova koriste linije sa velikim propusnim opsegom. 16-bitni prozor nikada ne omogućava protokolu na strani pošiljaoca da koristi prozor veći od 2^{16} = 64 KB.

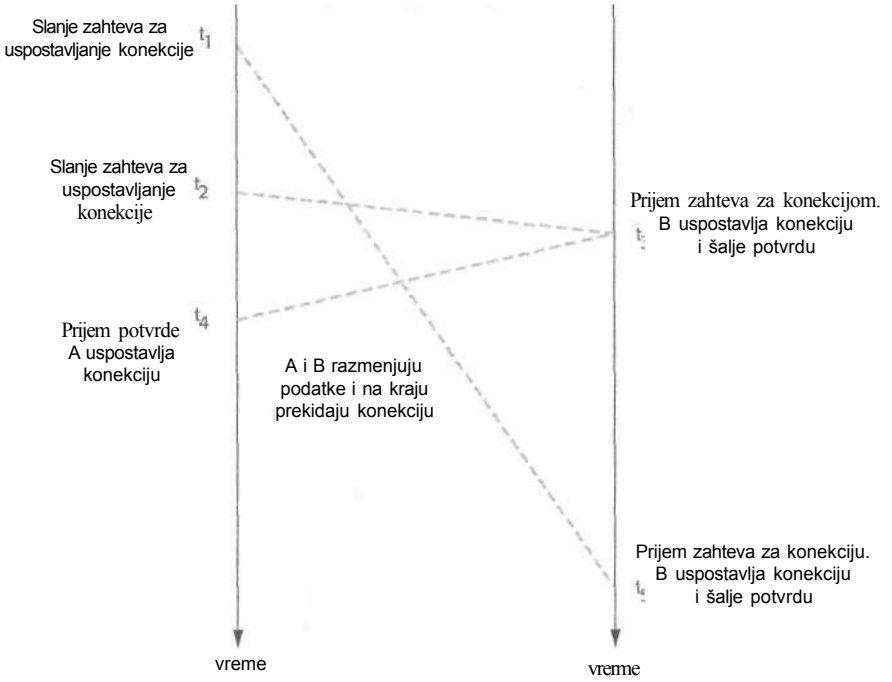
- Data (podaci) (promenljiva veličina) To su podaci koje obezbeđuje korisnik.

Upravljanje konekcijom

Upravljanje konekcijom uključuje proces uspostavljanja, održavanja i okončavanja konekcije. Ali šta, je u stvari, konekcija? Kao što je ranije istaknuto, ona je više virtuelna, nego fizička veza. U osnovi, dva TCP entiteta se dogovaraju da razmene TCP segmente i da uspostave neke parametre koji opisuju razmenu segmenata. Parametri obično sadrže brojeve sekvenci koji se koriste za bajtove i broj bajtova koje entitet može da prihvati. Entiteti mogu da razmenjuju segmente, izvode proveru grešaka, šalju potvrde i vrše kontrolu toka kao da izmedu njih postoji direktna konekcija, a detalji prenosa se prepuštaju nižim slojevima.

Za početak, dva entiteta moraju da se dogovore o uspostavljanju konekcije. Ovo na prvi pogled deluje jednostavno: jedan entitet šalje zahtev za uspostavljanje konekcije, a drugi odgovara potvrdno. Ovo je dvosmerno usaglašavanje (two-way handshake). Ipak, mogući su problemi ako prvi zahtev zakasni i pojavi se mnogo kasnije, nenamerno izazivajući drugu konekciju.

Na primer, razmotrimo situaciju sa slike 11.27. LJ trenutku t,, korisnik A zahteva konekciju. Medutim, zbog nečega, možda zbog zagušenja na mreži, ili problema u nekom prelaznom sajtu, zahtev kasni. Korisnik A, misleći da je poruka izgubljena, šalje još jedan zahtev za konekciju u trenutku t_2. Korisnik B prima drugi zahtev u trenutku t, i odmah šalje potvrdu za njega.

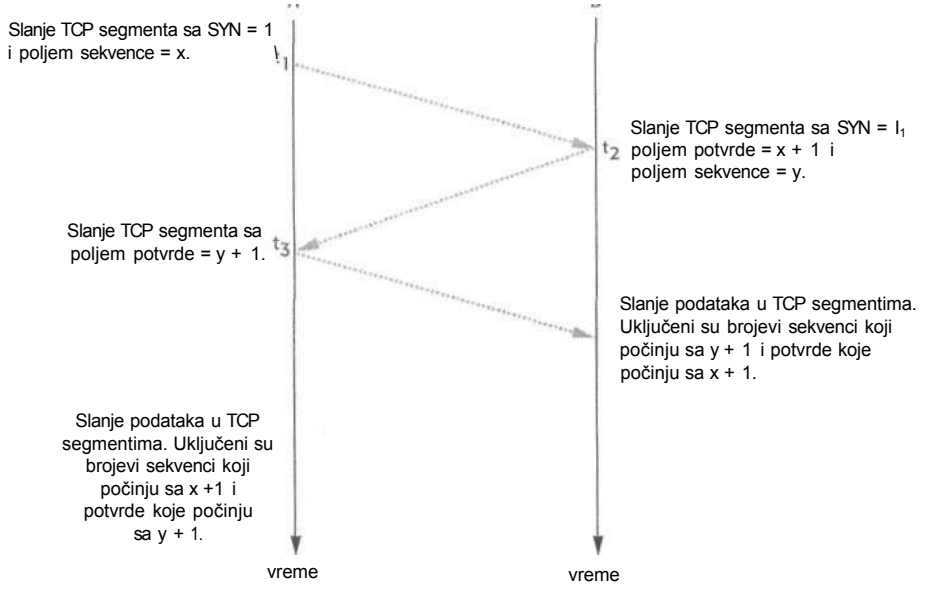


SLIKA 11.27 *Greška kod protokola sa dvosmemim usaglašavanjem*

Korisnik A prima potvrdu u trenutku t_4, i uspostavlja se konekcija. Za sada nema problema.

Korisnici A i B rade ono što bi trebalo da rade i na kraju raskidaju konekciju. Međutim, prvi zahtev za uspostavljanje konekcije i dalje postoji negde na mreži. Pretpostavimo da konačno stiže do B u trenutku t_5. Korisnik B misli da je to novi zahtev i šalje potvrdu za njega. Sto se tiče B, postoji još jedna konekcija, ali ovoga puta neplanirana. Situacija može da bude još gora: zamislite šta bi moglo da se desi ako je korisnik A poslao neke podatke za vreme prve konekcije koja je ozbiljno zakasnila. Ako ne primi potvrdu, korisnik A bi inicirao retransmisiju. Ali, prvi zahtev je i dalje negde na mreži. Šta se dešava ako ga B konačno dobije nakon trenutka t_5? On misli da je to drugi segment podataka i reaguje na njega. Koliko je ovaj problem ozbiljan? Zamislite da je reč o privatnim konekcijama sa švajcarskom bankom i da su se u segmentima nalazili podaci sa zahevima o depozitima od po pet miliona dolara. Službenici banke sigurno ne bi hteli da plate kamatu na lažni depozit u tolikom iznosu.

Ovde postoji mnogo više od običnog slanja zahteva za uspostavljanje konekcije i potvrda. Ranije smo pomenuli da TCP tretira podatke kao sekvence bajtova podeljene i poslate po segmentima. Umesto da se numeriše svaki segment, TCP pamti broj sekvence prvog bajta podataka u polju Sequence konkretnog segmenta. Polje Sequence je 32-bitno, tako da je moguće više od jedne milijarde različitih brojeva sekvenci. Da bi se izbegli problemi koji prate dvosmemo usaglašavanje, **trosmerno usaglašavanje (three-way handshake)** uspostavlja inicijalne brojeve sekvenci koje koristi svaki TCP entitet. Koraci trosmernog usaglašavanja ilustrovani su na slici 11.28.



SLIKA 11.28 *Protokol trosmernog usaglašavanja*

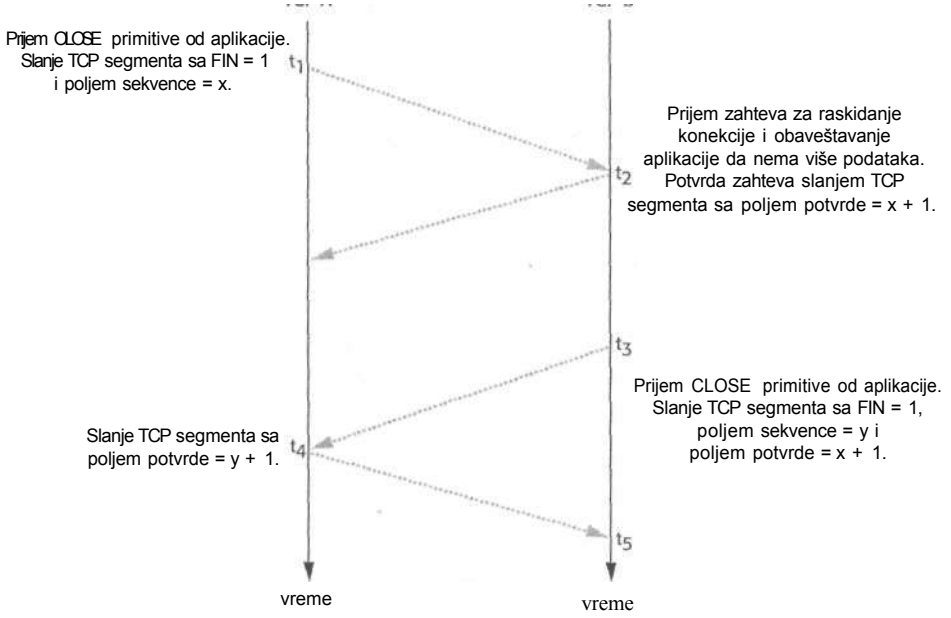
1. TCP entitet A prenosi TCP segment, zahtevajući konekciju (trenutak t_1). Postavlja fleg SYN kako bi ukazao da segment predstavlja zahtev za konekciju i defmiše polje Sequence da ima vrednost x. Može da utvrdi x pomocfu tajmera, ili brojača. Svaki novi zahtev prate drugi i veći (po moduly 2^{32}) inicijalni brojevi sekvence.

2. TCP entitet B prenosi TCP segment, potvrdujući i zahtev i broj sekvence (trenutak t_2). To izvodi postavljanjem flega SYN i definisanjem polja Acknowledgment = x + 1 i polja Sequence = y. B utvrđuje y na isti način na koji A utvrđuje x.

3. TCP entitet A potvrduje potvrdu (trenutak t_3). Sledeći segment koji šalje ima vrednost polja Sequence = x + 1 i Acknowledgment = y + 1.

Nakon što se ova tri segmenta pošalju i prime, svaki entitet zna koji inicijalni broj sekvence ovaj drugi koristi i pomoću polja Acknovvledgment ukazuju jedan drugom šta očekuju. U narednim segmentima sa podacima nalaze se uvećani brojevi sekvenci (po modulu 2^{32}), gde je svaki inkrement jednak broju bajtova podataka u prethodnom segmentu. Uskoro ćemo ovo razmotriti kada budemo govorili o kontroli toka.

Okončavanje konekcija umnogome podseća na uspostavljanje. TCP obezbeduje full-duplex komunikacije - ako jedan entitet želi da raskine konekciju, to ne znači da je drugi spreman. U osnovi, obe strane moraju da se dogovore o raskidu konekcije pre nego što to i izvedu. Zato se za okončavanje konekcije koristi sledeći protokol trosmernog usaglašavanja (slika 11.29):



SLIKA 11.29 *TCP protokol za raskidanje konekcije*

1. TCP entitet A dobija CLOSE zahtev od svoje aplikacije (trenutak t,). Reaguje slanjem TCP segmenta sa postavljenim flegom FIN i poljem Sequence = x. Fleg FIN ukazuje da nema više podataka i da tekući segment predstavlja zahtev za raskidanje konekcije. Parametar x predstavlja tekući brojač sekvence koji je TCP entitet održavao.

2. TCP entitet B prima segment (trenutak t_2). Reaguje obaveštavanjem svoje aplikacije o zahtevu za raskidanje konekcije, dme se ujedno kaže da više nema podataka. Osim toga, šalje TCP segment nazad do A, potvrđujući prijem zahteva. U međuvremenu aplikacija u B može da nastavi slanje podataka, ili može jednostavno da se pripremi za izdavanje sopstvenog zahteva za raskidanje konekcije.

3. TCP entitet B dobija CLOSE zahtev od svoje aplikacije (trenutak t_3). Salje TCP segment do A sa postavljenim flegom FIN i poljem Acknowledgment = x + 1.

4. Kada primi potvrdu (trenutak t_4), TCP entitet A šalje potvrdu i raskida konekciju. Kada potvrda stigne u B (trenutak t_5), i on prekida konekciju.

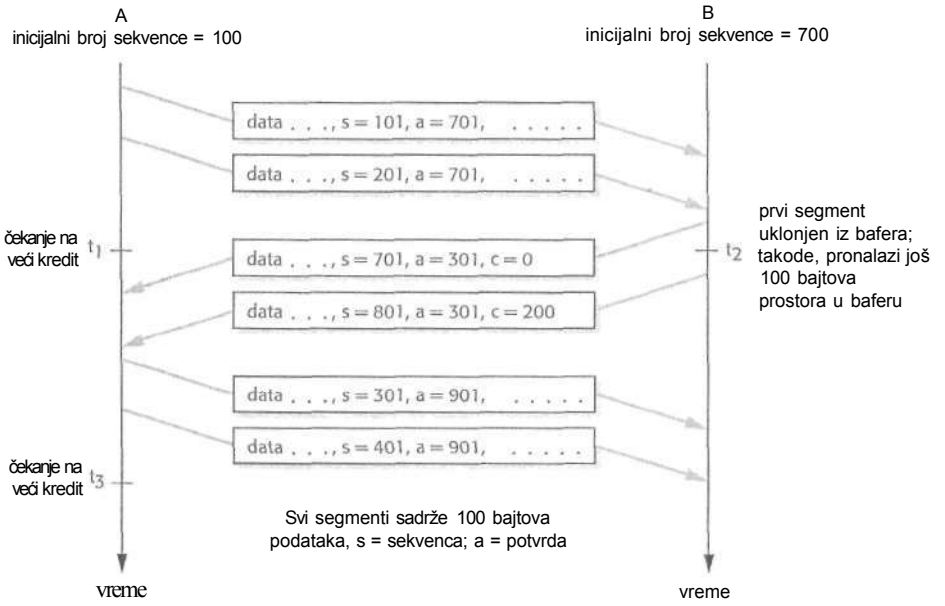
Kontrola toka

Kada se inicijalna konekcija uspostavi, dva TCP entiteta mogu da razmenjuju segmente koristeći full-duplex komunikacije, sa baferovanjem segmenata koje šalju i koje primaju. TCP entitet baferuje segmente koje šalje, jer nema garancija da će segment stići. Zato može da postoji potreba za retransmisijom. Bafemju se i segmenti koje primi, jer nema garancija da su stigli ispravnim redosledom. Zapamtite da je ovo logička konekcija, za razliku od fizičkih konekcija, i bilo kakvi problemi sa nižeg sloja mogu da izazovu probleme sa isporukom. Entiteti mogu da razmenjuju segmente koristeći varijaciju protokola klizajućih prozora koji su prikazani u Poglavlju 8. Ovde nećemo ponovo objašnjavati kontrolu toka, ali ćemo se fokusirati na nekoliko razlika između kontrole toka kod TCP protokola i kontrole toka koja je prikazana u Poglavlju 8.

- Kod TCP kontrole toka broj sekvence ukazuje na sekvencu bajtova, umesto na sekvence paketa (ili segmenata).
- Svaki entitet može dinamički da menja veličinu prozora na strani pošiljaoca, koristeći polje Window u segmentu.

Svaki entitet implementira kontrolu toka, koristeći *mehanizam kredita (credit mechanism),* poznat i kao *oglašavanje prozora (window advertisement).* Kredit, koji se smešta u polju Window konkretnog segmenta, definiše maksimalni broj bajtova koji entitet koji šalje taj segment može da primi i baferuje od drugog entiteta. Ovaj broj se zbraja sa brojem bajtova koji su već primljeni i baferovani (ali još uvek nisu preneti do višeg sloja). TCP entitet koji uzima ovaj segment koristi kredit kako bi utvrdio koliko bajtova može da pošalje pre nego što mora da čeka na potvrdu, ili na povećanje kredira.

Na slici 11.30 prikazan je primer ovog mehanizma. Pretpostavljamo da su dva entiteta već pregovarala o inicijalnoj konekciji, inicijalnim brojevima sekvenci i kreditima, koristeći trosmemo usaglašavanje. TCP entiteti A i B imaju inicijalne brojeve sekvenci 100 i 700, respektivno.



SLIKA 11.30 *Kontrola toka izvedena pomoću mehanizma kredira*

Osim toga, pretpostavljamo da svaki segment sadrži 100 bajtova podataka i da svaki entitet može da baferuje do 200 bajtova - kredit svakog entiteta je 200. Entitet A startuje slanjem dva segmenta, jednog sa brojem sekvence (s) 101 i drugog sa brojem sekvence 201. Potvrde (a) ukazuju šta A očekuje od B. Da bismo uprostili ovaj primer, pokazaćemo samo vrednosti kredita u paketima koji se šalju od B ka A.

Nakon slanja drugog okvira, A je iskoristio kredit i mora da čeka (trenutak tj). Kasnije prima segment od B u kome je broj sekvence 701, a potvrda 301. To znači da je B primio bajtove sekvenci do 300 i trenutno očekuje bajt 301 kao sledeći bajt. Segment sadrži i kredit (c) 0, jer su dva segmenta koja je B primio još uvek u baferima.

Ukratko, B nema mesta za nove segmente i ukazuje na to korišćenjem kredita 0. Zato A i dalje mora da čeka. U trenutku t_2 B isporučuje prvi primljeni segment višem sloju, tako da se oslobada 100 bajtova u baferu. Drugi segment je i dalje u baferu, tako da prostor bafera još nije slobodan. Ipak, pretpostavimo da u ovoj tački B može da poveća ukupnu količinu prostora u baferu na 300 bajtova.

Pošto se drugi segment koji sadrži 100 bajtova i dalje nalazi u baferu, B sada ima slobodnih 200 bajtova u baferu. U sledećem segmentu koji se šalje do A B naznačava potvrdu sa vrednošću 301 i kredit sa vrednošću 200.

Nakon nekog vremena A prima taj segment od B. Nakon toga, A šalje još dva segmenta, ali u trenutku t_3 mora ponovo da čeka na sledeću potvrdu. Da bi se primer pojednostavio, prikazani su samo krediti koje B šalje do A i prikazano je kako to utiče na A. Ipak, morate da znate da se ograničenja kredita primenjuju i na B.

Glavna prednost mehanizma kredita je što se omogućava robustniji protokol. Umesto da se koriste prozori fiksne veličine, TCP entiteti mogu da iskoriste prednost promene uslova. Ako na određenom sajtu ima malo aktivnosti, TCP entitet može da potvrdi segment sa povećanim kreditom kako bi bio bolje iskorišćen slobodni prostor u baferu.

Sa dmge strane, ako postoji dosta aktivnosti sa drugim konekcijama, može da se pošalje segment sa redukovanim kreditom kako bi se dolazeće informacije zadržale na nivou koji može da se kontroliše.

Kontrola zagušenja

Metod kontrole toka iz prethodnog odeljka deluje kao rešenje koje može da obezbedi efikasnu i "glatku" razmenu segmenata. Medutim, mogući su i neki drugi problemi koje ovaj metod ne može da reši. Ilustracije radi, prethodno razmatranje je pokazalo da se velidna prozora (krediti) podešava na osnovu količine informacija koju A, ili B mogu da kontrolišu. Nije uzeto u obzir šta može da se desi između njih. Na primer, pretpostavimo da su i A i B povezani na IO Mbps linkove i da mogu veoma brzo da šalju segmente. Šta se dešava ako jedina ruta koja povezuje A i B ide preko 1 Mbps linka? Ovo pomalo liči na povezivanje dva autoputa sa tri trake pomoću lokalnog puta, uz ograničenje brzine od 30 milja na sat. Autoputevi mogu da podnesu veliku količinu saobraćaja, ali to ne važi i za lokalni put. Rezultat su zagušenje i dugački redovi na izlaznim rampama.

Važno je da se razume da samo zato što prijemni entitet može da kontroliše odredeni broj paketa ne znači da i ruta može da kontroliše isti broj. Možda se sećate da smo u odeljku o IP-ju analizirali neke pristupe za rešavanje zagušenja, ali oni u opštem slučaju regulišu IP pakete koji se već nalaze na mreži.

Logično poboljšanje rešenja za zagušenje je dorada TCP protokola tako da se ograniči količina saobraćaja koja odlazi na mrežu na samom početku. U određenoj meri to se postiže kontrolom toka, ali, se, u stvari, odnosi samo na ono što primalac može da prihvati, a ne i na ono što se dešava u međuvremenu.

Da bi se mehanizmi kontrole zagušenja poboljšali, Jacobson (ref. [Ja88]) opisuje doradu TCP protokola koja omogućava entitetu pošiljaocu da reaguje na zagušene linkove i da promeni broj segmenata koje može da pošalje. Trebalo bi da sve TCP implementacije trenutno podržavaju ovu tehniku.

Da bismo pojednostavili ovu analizu, ponovo ćemo koristiti primer u kome je TCP entitet A uspostavio konekciju sa TCP entitetom B. Fokusiraćemo se na prenos od A do B, ali verujte da sve isto funkcioniše i u suprotnom smeru.

Pa bi implementirao ovu tehniku, A održava *prozor zagušenja (congestion ivindoui)* koji definiše broj bajtova koje A misli da može da pošalje bez izazivanja, ili pojačavanja zagušenja (uskoro ćemo videti kako utvrđuje taj broj). Ako je kapacitet prozora zagušenja veći od kredita u A (kapacitet B za prihvatanje segmenata), A neće slati više nego što mu kredit dopušta.

Međutim, ako je kapacitet prozora zagušenja manji, A koristi tu vrednost umesto kredita kako bi utvrdio koliko segmenata može da pošalje pre nego što bude morao da čeka na potvrdu. U svim trenucima A definiše svoj prozor za slanje tako da bude manji od prozora zagušenja i kredita koji je B naznačio. Nakon toga, koristi protokole za kontrolu toka za slanje segmenata i prijem potvrda.

U ovoj tački se nameću dva logična pitanja. Kako A može da utvrdi da li je došlo do zagušenja? Kako A reaguje na zagušenje? Odgovor na prvo pitanje daje mehanizam time-outa. Pretpostavimo da A šalje onoliko segmenata koliko mu predajni prozor dopušta i da se time-out javlja pre nego što stigne potvrda. Teorijski, time-out može da se desi ako je segment bio oštećen i nikada nije stigao do svog odredišta. U praksi, na mnogim mrežama linkovi se izvode pomoću optičkog fibera, koji je veoma pouzdan; na takvim mrežama se retko javljaju oštećeni paketi. Zato time-out obično znači da segment kasni zbog zagušenja u nekom delu mreže. Dakle, A interpretira time-out kao postojanje zagušenja.

Normalni odgovor na time-oute je ponovno slanje segmenata, ali to će samo pogoršati situaciju, jer pojačava zagušenje. Zato se preduzima mera redukovanja veličine prozora zagušenja za polovinu, ponovo se izračunava ptedajni prozor na prethodno opisani način i ponovo se šalje samo onoliko segmenata koliko to predajni prozor dopušta. Ako dođe do novog time-outa, veličina prozora zagušenja se dalje smanjuje za polovinu. Prozor zagušenja se neprestano redukuje za polovinu minimalne vrednosti sve dok veličina ne bude jedan segment. Ako u nekom trenutku stigne potvrda za sve segmente u redukovanom prozoru, protokol prestaje da redukuje veličinu prozora zagušenja, Poštovanjem ovog protokola A brzo redukuje količinu informacija koju šalje na mrežu. Tako Internet protokoli imaju vremena da reaguju i da ublaže zagušenje.

Zatim, pretpostavimo da je veličina predajnog prozora kod A mnogo manja od kredita koji šalje B. Ako je zagušenje smanjeno, A će imati predajni prozor čija se veličina utvrđuje na osnovu uslova koji više ne važe. Protokol može da reaguje povećanjem veličine prozora tako da se iskoriste rasterećeni linkovi.

Da bi se to izvelo, svaki put kada stigne potvrda za sve segmente iz predajnog prozora i ne javi se ni jedan time-out A povećava prozor zagušenja za ekvivalent jednog segmenta i ponovo preračunava veličinu predajnog prozora. Zato u odsustvu time-outa A može da pošalje povećani broj segmenata do one mere koju B može da prihvati i na taj način maksimalno koristi kapacitet mreže. Ovaj šablon redukovanja i povećavanja prozora zagušenja se nastavlja sve dok se konekdja održava.

Možda ste primetili da će A češće redukovati prozor zagušenja nego što će imati prilike da ga poveća. Ovo umnogome reflektuje filozofiju koja diktira da se rešenja za zagušenja moraju brže implementirati jer korisnici trpe zbog njih. Sa druge strane, izostanak time-outa ne mora da znad da je zagušenje eliminisano. Pre odražava činjenicu da je zagušenje postalo manje ozbiljno. Zbog toga, protokol implementira konzervativniji pristup kada se pokušava povećanje saobraćaja. Nema smisla brzo povečavati broj segmenata samo da bi se ponovo javilo zagušenje.

Jedino preostaje da se još opiše kako protokol definiše inicijalni prozor zagušenja. U stvari, funkcioniše slično periodu oporavka nakon zagušenja, sa nekim minornim promenama. Inicijalno, veličina prozora zagušenja je definisana kao ekvivalent jednog segmenta. Ako A pošalje jedan segment i dobije potvrdu, protokol udvostručava veličinu prozora zagušenja na dva

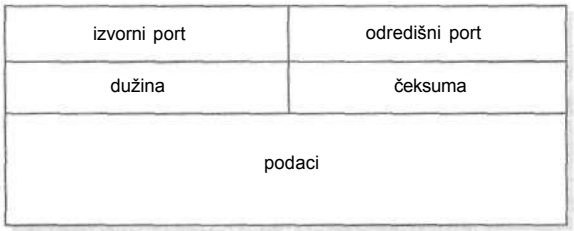
segmenta. Ako stignu potvrde i za te segmente, protokol ponovo udvostručava veličinu na četiri segmenta. U odsustvu time-outa, veličina prozora zagušenja se neprestano udvostručava, sve dok se ne dostigne vrednost kredita. Ako se u nekom trenutku registruje time-out, protokol uspostavlja veličinu koja je korišćena kada se time-out desio.

Kao što možete da vidite, u ovom slučaju protokol pokušava da poveća prozor zagušenja brže nego u slučaju kada je prethodno detektovano zagušenje. Ovo je logično, jer protokol pretpostavlja da nema zagušenja i pokušava što brže da uspostavi maksimalnu brzinu razmene. Zato je pomalo čudno što se ovakva procedura startovanja naziva lagani start (slow start). Naziv nema veze sa procedurom koja sledi nakon što se zagušenje detektuje; umesto toga, reč je o laganom startu u odnosu na standardnu kontrolu toka, kod koje nema prozora zagušenja. Kod obične kontrole toka A bi jednostavno slao do B onoliko paketa koliko on može da prihvati, bez uzimanja u obzir načina na koji posrednički linkovi mogu da regulišu naglo povećanje saobraćaja.

User Datagram protokol (UDP)

Veći deo ovog odeljka smo posvetili TCP protokolu. Iako je to najčešće korišćeni protokol, postoje i mnogi drugi protokoli. Jedan od njih je User Datagram Protocol (UDP), protokol transportnog sloja koji funkcioniše bez uspostavljanja veze. Jednostavniji je od TCP protokola; to može da se zaključi i na osnovu formata UDP segmenta (slika 11.31). Ima malo dodataka, što ukazuje na ograničene mogućnosti. Segment uključuje uobičajenu izvomu i odredišnu adresu, dužinu segmenta i čeksumu koja služi za detekciju grešaka. Pošto nije orijentisan konekciji, nema usaglašavanja za uspostavljanje konekcije (i, naravno, nema protokola za raskidanje konekcije). Kada UDP ima podatke za slanje, on kreira UDP segment i daje ga IP-ju radi isporuke. Na prijemnoj strani UDP dobija podatke od IP-ja i vrši proveru grešaka. Ako nema grešaka, UDP prosleđuje podatke do svog korisnika; ako postoji greška, UDP odbacuje podatke. Ne postoji formalni mehanizam za potvrdu grešaka, niti odredbe za kontrolu toka, ili sekvenciranje segmenata. Predstavlja nešto više od interfejsa između višeg sloja i IP-ja. Referenca [CoOO] sadrži detaljniji prikaz protokola.

Možda je najinteresantnije napomenuti da smo već opisali komandu koja koristi UDP. Ranije smo u ovom poglavlju predstavili komandu `traceroute`, koja prikazuje lokacije duž rute na Internetu.



SLIKA 11.31 *UDPsegment*

Kada korisnik u izvoru unese komandu traceroute, dešavaju se sledeće akcije:

1. Izvor kreira UDP Probe paket sa brojem neiskorišćenog porta. Postavlja ga u IP paket sa vrednošću polja Time to Live (TTL) postavljenom na 1 i paket se šalje do odredišta.
2. Prvi ruter umanjuje vrednost polja TTL i, pošto vidi da je 0, odgovara slanjem ICMP Time Exceeded poruke.
3. Izvor dobija ovaj odgovor i beleži adresu rutera i vreme potrebno za pun krug.
4. Izvor kreira sličan UDP Probe paket i postavlja ga u IP paket, ali ovoga puta sa TTL postavljenim na 2.
5. Ovoga puta paket prolazi prvi ruter, ali drugi ruter reaguje kao u koraku 2.
6. Izvor reaguje kao u koraku 3.
7. Izvor neprestano šalje probne IP pakete sa sve većim vrednostima u polju TTL. Svakog puta ruter dospeva do jednog rutera dalje pre nego što bude odbačen. U svakom slučaju, taj ruter reaguje kao u koraku 2 i izvor beleži te informacije.
8. Na kraju, TTL vrednost je dovoljno velika da paket stigne do svog odredišta (ako postoji i ako je u granicama maksimalne vrednosti skokova). Pošto UDP Probe paket naznačava neiskorišćeni port, odredište odgovara Port Unreachable porukom. Izvor prepoznaje ovo kao poslednji odziv koji će videti. Beleži informacije i završava.

Real-Time Transfer protokol (RTP)

Većsmo predstavili probleme koji prate QoS zahteve aplikacije, kao što su real-time audio, ili video. Objasnili smo neke pristupe nižeg nivoa, kao što su mreže sa komutacijom kola i rezervacijom propusnog opsega, i rekli smo da ćemo u Poglavlju 13 prikazati ATM, sledeću tehnologiju zasnovanu na virtuelnim kolima. Sve ove tehnologije imaju jednu zajedničku karakteristiku - rezervišu resurse tako da protokoli nižeg nivoa mogu da garantuju QoS koji je zahtevan tim aplikacijama. Postavlja se logično pitanje ako mreža ne obezbeđuje ovakve protokole nižeg sloja da li su real-time aplikacije nemoguće, ili nepraktične. Odgovor je: nije neminovo.

Protokol transportnog sloja pod nazivom Real-Time Transport Protocol (RTP) dizajniran je za podršku real-time aplikacija, kao što su audio, ili video aplikacije. Ovo se pomalo razlikuje od dragih protokola koje smo opisivali, jer je dizajniran za rad sa određenom aplikacijom i funkcioniše samo na jednoj strani konekcije. Zato neki ovaj protokol smatraju protokolom sloja aplikacije. Sa druge strane, RTP se obično izvršava iznad UDP-a, iako može da se izvršava i iznad drugih transportnih protokola. Zbog toga, neki ga smatraju podslojem transportnog sloja. Međutim, za naše razmatranje nije bitno kako se posmatra. Ono što je značajno je to što se obično nalazi između real-time aplikacije i UDP-a i predstavlja protokol izmedu krajnjih tačaka. Na ovom sloju nalazi se iznad svih strategija rutiranja i ne obezbeđuje nikakve QoS garancije. Međutim, to i nije nieeova svrha.

SLIKA 11.32 *Pomeranje paketa*

Iako protokoli nižeg nivoa na Internetu nisu dizajnirani za real-time aplikacije, poboljšanja u brzini linije i dizajnu rutera defmitivno su omogućila prenos ogromne količine informacija preko Interneta. Uvodenje nekih zaista sofisticiranih algoritama za kompresiju i mogućnost propuštanja podataka na dovoljno velikim brzinama da se podrže real-time aplikacije postaju realnost, iako nedostaju protokoli koji bi ih garantovali.

Medutim, i dalje postoje neki problemi kod protokola koji se zasnivaju na prenosu paketa za podršku real-time aplikacija. Jedan je **pomeranje paketa (packet jitter)** (slika 11.32). Pretpostavimo da govorite u mikrofon koji je povezan na personalni kompjuter.

Lokalni softver moduliše Vaš glas, koristeći metod kao što je PCM i smešta govorne podatke u sukcesivno generisanim paketima. Nakon toga, softver šalje pakete na mrežu. Pošto je reč o real-time aplikaciji, lokalni softver sempluje govorne podatke u fiksnim intervalima i generiše pakete na fiksnim brzinama. Međutim, dok paketi "putuju" kroz mrežu, kod nekih mogu da se jave neka manja kašnjenja u prelaznim ruterima.

Zbog toga, paketi (i govorni podaci) mogu da stignu u odredište promenljivim brzinama. Krajnji rezultat može da se uporedi sa puštanjem audio trake kod koje se brzina okretanja periodično ubrzava i usporava.

Na slici 11.33 prikazan je sledeći problem koji može da se javi kada se prenose i audio i video podaci. Pretpostavimo da video kamera snima jedno predavanje. Neki protokoli zasebno kodiraju audio i video komponente - postoje dva niza podataka: jedan za audio i drugi za video. Kao i ranije, i audio i video podaci se, nakon toga, modulišu, smeštaju u pakete i šalju na mrežu. Zbog kašnjenja, ne postoje garancije da će video i audio paketi koji su istovremeno generisani istovremeno i stići do odredišta i biti pušteni u isto vreme.

SLIKA 11.33 *Nesinhronizovani podaci za glas i sliku*

Ovaj nedostatak sinhronizacije **(sinhronizacija između medija - intermedia synchronization)** uzrokuje to da plejer na odredištu prikazuje video i pušta odgovarajuđ audio zapis u neznatno različitim trenucima. Ovo je kao kada gledate film koji je sinhronizvan na neki drugi jezik. Reči koje ćujete ne odgovaraju pokretima usta glavnih likova.

Ako izvršavamo real-time aplikacije nad TCP, ili UDP protokolom, ti problemi mogu da se pojave zato što ni jedan od ovih protokola nije dizajniran za njihovo rešavanje. Ovde "nastupa" RTP. Ne može da garantuje pravovremenu isporuku, ali može da "izgladi" neke probleme koji su izazvani ne mnogo ozbiljnim kašnjenjima u prenosu podataka.

U stvari, postoje još dva protokola koja moramo da predstavimo. Jedan je RTP, a drugi je RTP Control Protocol (RTCP). RIP se bavi isporukom podataka, dok RTCP izvršava neke kontrolne funkcije. Ova dva protokola obično odgovaraju brojevima sukcesivnih portova, što im omogućava nezavisno funkcionisanje. Najpre ćemo razmotriti RIP.

Tipični pristup postavlja modulisane podatke u RTP paket. Taj paket se, nakon toga, ugrađuje u UDP paket, gde se formira red čekanja za prenos. Zato je možda logično početi opisom RTP paketa, koji sadrži sledeće stavke: *

- **Version number (Broj** verzije) Ovo polje ukazuje na tekuću verziju RlP-a.
- **Payload type (Tip korisnih informacija)** Ovo je 7-bitni ceo broj koji definiše način na koji su podaci modulisani i sa kojom frekvencijom semplovanja. Postoje identifikatori za PCM, MPEG, JPEG i za još nekoliko drugih šema modulacije. Tako prijemna strana može da koristi odgovarajuću šemu dekodiranja za konvertovanje nazad u audio, ili video.
- **Sequence number (broj sekvence)** Paketi su sekvencirani tako da se sačuva njihov inicijalni raspored i da se utvrdi da li su neki paketi izgubljeni.
- **Timestamp (vremenska oznaka)** Ovo se koristi za ranije opisano pomeranje paketa. Kada izvor generiše RIP paket, koristi signal takta za smeštanje vrednosti vremenske oznake u zaglavlju RIP paketa. Ova oznaka odgovara trenutku kada je kreiran prvi sempl. Vrednost vremenske oznake nije kritična, već je kritična razlika između vremenskih oznaka dva sukcesivna RIP okvira. Na primer, pretpostavimo da izvor koristi PCM i da uzima 8-bitne semplove učestalošću od 8.000 semplova u sekundi i inkrementira vrednost vremenske oznake za 1 za svaki sempl. Svaki bajt predstavlja sempl uzet na svakih 125 psec (ili jednom za svaku vrednost vremenske oznake). Pretpostavimo da primalac dobija dva sukcesivna RTP paketa čije se vremenske oznake razlikuju za 100. Na osnovu polja Payload Type primalac zna da PCM generiše semplove u intervalima od 125-msec. Pošto je razlika vremenskih oznaka 100, on zna i da mora da konvertuje 100 semplova u audio na odgovarajućoj brzini.t Osim toga, zna da mora da počne puštanje paketa 12,5 milisekundi (125 (isec x 100) nakon što pusti prvi paket. Ako drugi paket stigne pre nego što treba da bude pušten, primalac može da odloži njegovo puštanje kako bi se obezbedio ravnomeran konzistentan zvuk.

' Postoji još nekoliko stavki, ali one nisu relevantne za naše razmatranje.

t Ovo nije isto kao da se kaže da paket sadrži 100 bajtova. U stvari, možda postoji manje bajtova zbog šema za kompresiju.

Međutim, šta se dešava ako RTP paket stiže isuviše kasno da bi bio pušten? Jedna opcija nalaže jednostavno odbacivanje paketa. Slušalac može odmah da primeti prekid zvuka, ali... Sofisticiraniji softver može eventualno da pokuša da ekstrapolira sadržaje na osnovu onoga što je primio u prethodnim paketima. Odbačeni paketi nisu problem ako se retko dešavaju. Kasnije ćemo objasniti protokol koji se bavi češćim gubljenjem paketa.

- **Synchronization source identifier (SSRC - identifikator sinhronizacije izvora)** Ovo polje identifikuje izvor niza podataka. Ako postoje zasebni audio i video nizovi, svaki može da ima drugačiju SSRC vrednost. Ako na istom sajtu postoji više kamera, svaka ima drugačiji SSRC. Različite SSRC vrednosti omogućavaju primaocu da se uskladi sa različitim izvorima. Kompjuter na prijemnoj strani može da ima jedan prozor koji prikazuje predavanje, a u drugom je crtani film Nickelodeon u periodima kada predavanje postaje dosadno.

- **Contributing** source **identifier** (CSRC - **identifikator** svih izvora **koji** doprinose **komunikaciji)** Neki prenosi po principu multicastinga mogu da imaju više aktivnih učesnika. Drugim rečima, umesto da postoji jedan govomik i da ga svi ostali slušaju, sledeća opcija može da bude obezbedivanje mogućnosti da svi govore i intereaguju. Ovo je slično konferencijskim pozivima. U tom slučaju više nizova mogu da se kombinuju (multipleksiraju) u jedan niz.* Ako RlP paket sadrži blokove podataka iz više izvora, CSRC ih identifikuje.

- Number of CSRC entries (broj CSRC zapisa) Ovo polje je jasno samo po sebi.

- **Data** (podaci) Modulisani audio, ili video podaci

Za razliku od RIP protokola, koji se bavi prvenstveno prenosom podataka, RTCP obezbeduje neke kontrolne informacije. RTCP niz postoji za svaki RIP niz. On prenosi informacije koje i pošiljaocu i primaocu omogućavaju da obezbede značajne informacije za nizove podataka. Pošiljalac i primalac nakon toga mogu da koriste te informacije za izvršavanje odredenih funkcija, ili radi adaptiranja na odredene promene uslova. RTCP ne defmiše kako treba da reaguju na informacije. Kao što je ranije istaknuto, RTCP se izvršava na drugom broju porta i to konkurentno sa RlP protokolom. Jedna "stvar" koju RTCP obezbeđuje jeste sinhronizacija izmedu medijuma.

Prethodno smo istakli da primalac ponekad mora da sinhronizuje podatke od dva različita medijuma (na primer, audio i video). RTP ovo ne može da izvede zbog nekoliko razloga. Prvi je to što dva različita niza mogu da imaju različite SSRC vrednosti. Ne postoji ništa što bi ih vezalo. Ipak, svaki izvor ima jedinstveni identifikator pod nazivom CNAME (obično se daje u formi *korisnickoime@adresahosta)* i pošiljalac može da pošalje RTCP paket do svakog primaoca sa pridruženim CNAME uz jednu, ili više SSRC vrednosti. Primalac dobija ove informacije i može da utvrdi kada dva niza podataka potiču od istog izvora.

Nažalost, to i dalje nije dovoljno za sinhronizaciju. Iako svaki RIP paket ima vremensku oznaku, ta vrednost zavisi od frekvencije semplovanja i ne odgovara realnom vremenu. Osim toga, različiti audio i video nizovi mogu da imaju različite frekvencije semplovanja; zbog toga, poređenje brojeva za dva niza možda nema smisla.

■ Uređaj koji ovo izvodi naziva se *mikser*.

U izveštaju pošiljaoca od RTCP-a može da se uključe vrednost vremenske oznake i odgovarajuće stvarno vreme. Kada primalac dobije ove informacije, može da utvrdi koji RTP nizovi potiču od istog izvora, a zatim koristi vremensku oznaku i vrednosti realnog vremena za sinhronizaciju izvođenja.

Sledeče što RTCP omogućava jeste *adaptacija brzine (rate adaption).* Primalac može da "izađe na kraj" sa pomeranjem paketa korišćenjem vremenskih oznaka za utvrđivanje odgovarajućih brzina izvođenja. Medutim, ako paket stigne suviše kasno da bi se izveo, moguće je da će biti odbačen. Ovo nije problem ako se ne dešava suviše često. Ipak, ako dođe do zagušenja na mreži, stvoriće se veći broj paketa koji kasne za izvođenje. Možda na mreži nema zagušenja, ali pošiljalac isuviše brzo generiše i prosleđuje pakete na mrežu.

U svakom slučaju, problem velikog broja izgubljenih paketa može da se reši tako što će ih pošiljalac sporije generisati, ili će se koristiti bolji algoritam za kompresiju podataka. Međutim, kako pošiljalac može da zna šta se desilo? Primalac može da koristi RTCP kako bi obezbedio periodične izveštaje za pošiljaoca, u kojima će naznačiti procenat izgubljenih paketa. Ako je procenat suviše visok, pošiljalac se adaptira i koristi nižu frekvenciju semplovanja, ili bolji kompresije za kodiranje podataka. Ako izveštaj ukazuje da nije došlo do gubljenja podataka, pošiljalac može čak da pokuša i da poveća frekvenciju semplovanja kako bi se obezbedio kvalitetniji tok informacija. Moguće je funkcionisanje na oba načina.

0 rešavanju problema sa real-time audio i video aplikacijama može da se kaže još mnogo štošta; mi smo ovde samo "zagrebali" po površini. Ako ste zainteresovani za detalje multimedijalnih komunikacija, možete ih pronaći u referencama [KuOl], [Mi02] i [HaOl].

11.5 Internet aplikacije

Pošto je TCP/IP široko korišćen protokol, deluje logično da razmotrimo i neke aplikacije koje se pokreću zajedno sa TCP/IP mrežama. Verovatno su najpoznatiji Telnet, FTP, email i, naravno, World Wide Web. Postoje i drugi, ali ograničeni prostor u ovoj knjizi nalaže da napravimo izbor šta čemo predstaviti. lnternet aplikacije uključuju interakciju između programa koji se izvršavaju na različitim kompjuterima povezanim na lnternet (klijent/server model) i lako ćete zapamtiti da sve aplikacije o kojima ćemo govoriti zahtevaju saradnju izmedu dva programa na različitim mašinama pokretanjem zajedničkog protokola.

Protokoli virtuelnog terminala

Mreže obezbeđuju komunikaciju izmedu više različitih tipova opreme i softvera. Ranije aplikacije su često od korisnika zahtevale da se uloguju na udaljene kompjutere preko nemog terminala (uredaji koji su imali malo, ili nimalo mogućnosti za lokalnu obradu). Značajan problem u to vreme bila je činjenica da je softver često bio pisan imajući na umu specifičnu opremu. Na primer, uzmimo editor koji je prikazivao tekst na ekranu i omogućavao pomeranje kursora i izvođenje promena. Ali, prikazani broj redova i kolona razlikovao se u zavisnosti od tipa terminala. Komande za pomeranje kursora i brisanje i umetanje teksta zahtevale su kontrolne sekvence koje su ponekad bile odredene tipom terminala. Neki terminali su imali, čak, i različite tastature. Standardizacija je bila izvedena u mnogo manjoj meri nego u današnje vreme.

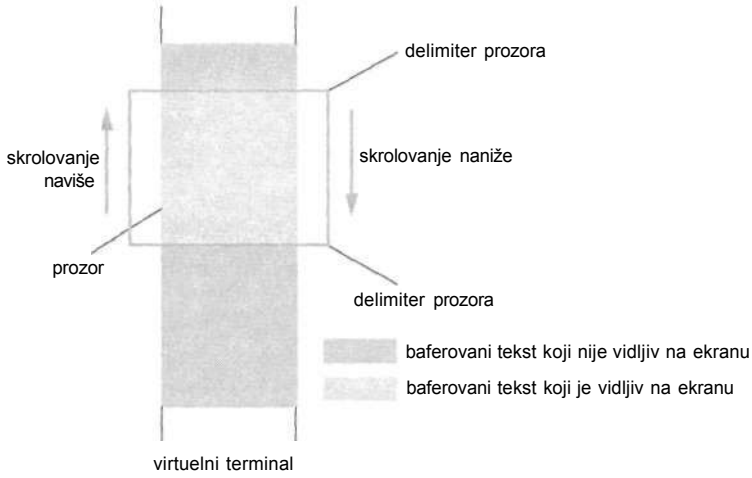
Danas su nemi terminali zamenjeni personalnim kompjuterima, ali mnogi imaju softver koji im omogućava emulaciju različitih terminala i logovanje na udaljene mašine. Udaljene mašine pokreću aplikaciju (server program) koja obezbeduje informacije iz fajla za korisnika. Nakon toga, korisnik može da vidi informacije i da lokalno radi sa njima (pomoću klijent softvera). Ova interakcija može da uključi umetanje, ili brisanje podataka, pomeranje kursora, ili pronalaženje odredenog teksta. Ako iskoristimo primer tekstualnih editora koji koriste kompletni ekran, postoji značajan problem zbog toga što različiti editori zahtevaju različite sekvence sa tastature za izvršavanje odredenih akcija. Svi koji poznaju UNIX, ili Linux platformu mogu da razmotre mogućnost rada sa vi, emacs i pico editorima i primetiće značajne razlike. Kada aktivirate editor daljinskim logovanjem, Vaš lokalni softver mora da razume mapiranje pritisaka na tastere tastature koje editor zahteva. U suprotnom, unošenje komandi izaziva nepredviđene efekte na fajlu koji se edituje. Primera radi, kada se neko preko telneta poveže na Linux server i pozove vi editor, kursorski tasteri možda neće funkcionisati kao što se moglo očekivati. Umesto da se kursor pomeri na susednu poziciju, u stvari se u fajl umeću neki čudni karakteri. Razlog je to što se kodovi koje kursorski tasteri generišu lokalno ne prevode u odgovarajuće kodove na udaljenom kraju koji pomera kursor. Programi za emulaciju terminala omogućavaju mapiranje pritisaka na tastere na različite kodove kako bi "stvari" ispravno funkcionisale. Neophodno je samo proučiti opcije koje su dostupne iz menija.

Emulacija terminala je primer šire klasifikacije protokola, poznate kao **protokoli virtuelnog terminala.** *Virtuelni terminal* je struktura podataka koju održavaju softverska aplikacija, ili lokalni sajt. Njegov sadržaj predstavlja stanje lokalnog terminala. Na primer, struktura može da uključuje tekuću poziciju kursora, indikator obrnutog videa, oblik kursora, broj redova i kolona i boju. I korisnik i aplikacija mogu da se referenciraju na ovu strukturu. Aplikacija koja upisuje podatke u virtuelni terminal i softver virtuelnog terminala zahtevaju prevođenje ulaza sa tastature. Kada korisnik unese podatke, proces funkcioniše u suprotnom smeru. Protokoli Virtuelnog terminala defmišu format strukture podataka, softver konvertuje korisnički ulaz u standardnu formu, a, nakon toga, aplikacija čita standardni "ekran".

Virtuelni terminali mogu da sadrže više podataka nego što ih je moguće prikazati na ekranu. Ovo je izuzetno korisno kada je neophodno skrolovanje. Na primer, pretpostavimo da virtuelni terminal može da smesti 200 linija u bafer, ali je u jednom trenutku na ekranu moguće prikazati samo 24 linije. Informacije u virtuelnom terminalu defmišu prvu i poslednju liniju prikazanih podataka (slika 11.34). Prikazani podaci predstavljaju *prozor* i označeni su njegovim delimiterima (graničnicima). Ako korisnik koristi linije za skrolovanje sa strane, softver virtuelnog terminala jednostavno menja delimitere prozora. Rezultat je to da se na terminalu prikazuju različite linije teksta.

Telnet Jedan primer mrežnog protokola za virtuelni terminal je **Telnet.** Bio je dizajniran za ARPANET i predstavlja jedan od protokola iz TCP/IP skupa. Možda većina ljudi Telnet zna kao aplikaciju koja omogućava daljinsko logovanje.

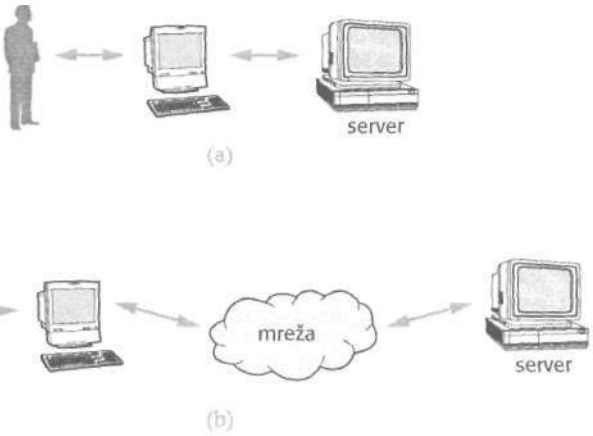
Iz perspektive korisnika **daljinsko logovanje** se ne razlikuje od logovanja na lokalni kompjuter (slika 11.35a). Medutim, slika 11.35b daje verniji prikaz situacije. Korisnik koji radi za personalnim kompjuterom pokreće protokole za povezivanje na mrežu. Protokoli uspostavljaju konekciju preko mreže sa udaljenim kompjuterom. Korisnik i udaljeni kompjuter razmenjuju komande i podatke koristeći mrežne protokole.



SLIKA 11.34 *Prikazivanje baferovanog teksta po prozorima*

Korisnik radi na višem sloju, tako da je sve ovo za njega transparentno, a personalni kompjuter ima "osećaj" da su tastatura i monitor direktno povezani na udaljeni kompjuter. Jedina razlika može da bude neznatno zakašnjenje izmedu odziva, posebno ako je udaljeni kompjuter veoma daleko i ako je mreža opterećena velikom količinom saobraćaja.

Telnet funkcioniše u klijent/server modu (slika 11.36). Odnosno, personalni komjputer (klijent) lokalno pokrece Telnet i prenosi podatke između korisnika i mrežnih protokola. Može da formatira i šalje specifične komande, a neke od njih ćemo uskoro i prikazati. Udaljeni kompjuter (server) pokreće svoju verziju Telneta.



SLIKA 11.35 *Daljinska konekcija*

SLIKA 11.36 *Relacija Telnel klijenta/servera*

Izvršava slične funkcije, razmenjuje podatke izmedu mrežnih protokola i operativnog sistema i interpretira komande koje je korisnik izdao.

Korisnik može da iskoristi Telnet na nekoliko načina, u zavisnosti od klijenta. Jedan način je da se uloguje na lokalni kompjuter, kao što su UNIX, ili Linux sistem, čeka na sistemski prompt ("$" u našem primeru) i unosi komandu

```
$ Telnet tekstualna-adresa
```

Tekstualna adresa defmiše host kompjuter na koji korisnik želi da se poveže. Telnet nakon toga poziva transportni protokol pregovara i uspostavlja konekciju sa udaljenim sajtom. Kada se poveže, korisnik mora da se uloguje na udaljeni sajt naznačavanjem broja naloga i lozinke. Takode je moguće uneti komandu Telnet bez tekstualne adrese.

Lokalni sistem reaguje prikazivanjem `Telnet` prompta (`telnet>`). Na udaljene sajtove možete da se povežete unošenjem komandi connect, ili open (u zavisnosti od lokalnog sistema) i naznačavanjem tekstualne adrese.

Sledeća mogućnost je kupovina Telnet programa i njegovo instaliranje na personalni kompjuter. Program se obično startuje duplim klikom na ikonicu aplikacije, ili selektovanjem naziva aplikacije iz menija, ćime se otvara prozor sličan onom na slici 11.37. Kada unesete neophodne elemente, konekcija je uspostavljena i potrebno je uneti korisničko ime i lozinku.

Postoji još jedan način - unošenje *telnet://tekstualna-adresa* u adresno polje Internet Explorera, ili Netscape Navigatora. Kako se možete povezati trenutno nije bitno; fokusiračemo se na ono što se dešava nakon što se povežete.

Kada se povežete, Telnet radi u pozadini i potpuno je transparentan za korisnika. Sve što unesete ide na server. Medutim, ponekad će Vam biti potrebna interakcija sa Telnetom. Ako pokrećete Telnet program na svom kompjuteru, to možete da izvedete pomoću padajućih menija koji se nalaze na vrhu ekrana.

Ako pokrećete Telnet sa Linux sistema, možete da izbegnete daljinsko logovanje unošenjem kontrolne sekvence kao što je control-]. Ovo korisniku vraća Telnet prompt, ali ne prekida daljinsku konekciju. Nakon toga, možete da unosite komande direktno za Telnet.

Na primer, pretpostavimo da se korisnik povezuje preko komande linije u Linuxu, loguje se na udaljeni kompjuter i suočava se sa nekim kašnjenjima. Možda se pita da li kasni odziv udaljenog servera, ili je server prestao da funkcioniše. Tada može da unese control-] i Telnetu izda komandu `send ayt`.

Komanda ispituje udaljeni server i pita ga da li je funkcionalan (ayt je skraćenica od "are you there" - "da li si tu"). Ako jeste, korisnik vidi odziv [remote server identifier: yes].

SLIKA 11.37 *Telnet protokol zahteva naziv hosta i sesije.*

Ako korisnik pokreće program na udaljenom kompjuteru i taj program ude u beskonačnu petlju, korisnik može da prede u Telnet prompt i tamo unese komande send abort, ili send susp kako bi pokrenuti proces bio "ubijen", ili suspendovan.

Postoji veliki broj različitih Telnet opcija; one zavise od softvera i hardvera koji koristite. Obično postoji pomoćno sredstvo koje obezbeđuje informacije o njima. Zato ovde nećemo pokušavati da ih opisujemo. Najbolji način da se detaljno nauči kako se Telnet koristi omogućava dokumentacija koja se dobija uz sistem.

Secure Shell Telnet je možda najpoznatiji protokol za daljinsko logovanje, ali svakako nije i jedini. Primeri su rlogin, rsh i ssh. Pomoćni program rlogin (Remote Login) podseća na Telnet, ali je manje fleksibilan i dizajniran je za UNIX sisteme koji pokreću rlogind demon.* Pomoćni program rsh slično obezbeduje daljinsko logovanje, ali može da dopusti korisniku da naznači jednu komandu koju želi da izvrši na udaljenoj mašini. Ovde ih nećemo predstaviti, jer je ssh (Secure Shell) mnogo moćnija i fleksibilnija alatka za sve namene, tako da su ostali napušteni. Navešćemo nekoliko primera za korišćenje ssh.

'Demon proces se pokreće u pozadini i čeka na neku vrstu zahteva za odredenim servisom. Kada zahtev stigne, demon obično kreira drugi proces koji upravlja lim zahlevom. Originalni proces nastavlja normalno funkcionisanje, čekajući na sledeći zahtev.

Testirani su na serveru koji pokreće Red Hat Linux verzija 7.3. Neke starije verzije ne podržavaju ssh i komande mogu da funkcionišu malo dmgačije nego na ostalim varijantama UNIX-a.

Jedna razlika izmedu ssh i Telneta je da ssh omogućava korisniku da lako izvrši jednu komandu na udaljenom hostu. Sa Telnetom korisnik mora da se uloguje, unese komandu, pa da se izloguje. Kod ssh korisnik može da unese komandu

```
ssh korisnik@naziv_udaljenog_hosta Is -1
```

ssh traži od korisnika da unese tačnu lozinku. Nakon toga, udaljeni sistem izvršava komandu Is -1 za izlistavanje fajlova i atributa, a ssh vraća rezultate do korisnika; korisnik nije ulogovan na udaljeni host. Konekcija je uspostavljena samo radi izvršavanja jedne komande i okončana je čim je generisan izlaz komande.

Ovo je jednostavan primer, a najznačajnija razlika između ssh i Telneta je to da ssh obezbeđuje autentifikaciju i šifrovanje razmenjenih podataka između lokalnog i udaljenih hostova. Ovo je izuzetno značajno kada se inicijalno ulogujete, jer morate da unesete lozinku. Ako lozinke nisu šifrovane, izlažete se velikom riziku. To bi bilo isto kao da ostavite kreditnu karticu na vidno mesto gde svako može da iskopira broj Vašeg računa. Različite verzije ssh obezbeđuju šifrovanje i autentifikaciju korišćenjem različitih metoda. Ovde se nećemo baviti tim razlikama, već ćemo samo ukratko opisati neke najznačajnije karakteristike. Zainteresovani čitaoci mogu da pogledaju reference [Sc02], [MaOO] i [BaOl], a za više informacija treba da posete Web sajt www.openssh.org.

Korisnik može da inicira ssh na isti način kao i Telnet - jednostavno unosi odgovarajuću komandu, ili selektuje odgovarajuće dugme na grafičkom korisničkom interfejsu (GUI) i započinje protokol za uspostavljanje konekcije. Razlika je u onome što sledi nakon toga. Na primer, jedna karakteristika koju ssh obezbeđuje je autentifikacija servera. Da biste razumeli značaj ovoga, pretpostavite da korisnik želi da uspostavi konekciju sa udaljenim hostom. Medutim, neko presreće taj zahtev i, lažno se predstavljajući kao željeno odredište, vraća odziv validnog izgleda. Ovo se naziva *lažno predstavljanje (spoofing).* Kada korisnik unese lozinku, ona se šalje na pogrešan sajt bez korisnikovog znanja i dospeva u ruke kriminalca.

ssh sprečava ovakve prevare autentifikacijom servera na koji se korisnik povezuje. Ovaj proces autentifikacije se pomalo razlikuje od onoga koji je korišćen za SSL (videti Poglavlje 7). Ilustracije radi, pretpostavimo da korisnik unese komandu ssh *udaljeni_host.* Ako je ovo prvi put da ssh vidi ovaj naziv hosta, ne može da potvrdi da je to host koji bi trebalo da bude. Zato ssh generiše poruku sličnu sledećoj:

```
The authenticity of host 'remote_host (100.200.200.2g0) ' can't
be established.
RSA key fingerprint is
a5:42:fe:ce:45:a3:7c:c4:ff:52:5f:f9:65:16:9b:4d.
Are you sure you want to continue connecting (yes/no)?
```

ssh govori korisniku da ne može da potvrdi autentičnost hosta i pita ga šta želi da uradi. Osim toga, obezbeđuje otisak (vrednost digitalnog sažetka) ključa koji korisniku omogućava da utvrdi autentičnost servera (i ključa).

Na primer, korisnik može da kontaktira sa administratorom hosta i da proveri otisak i ključ. Ako selektuje negativan odgovor, konekcija se neće uspostaviti. Ako izabere potvrdan odgovor (a to treba da uradi samo ako je siguran da je udaljeni host validan), konekcija se uspostavlja. U tom slučaju se kreira fajl pod nazivom known_hosts (ili se ažurira ako fajl već postoji) u poddirektorijumu .ssh, koji je smešten u korisnikovom matičnom direktorijumu. Taj fajl sadrži javni ključ udaljenog hosta.

Sledećeg puta kada korisnik pokuša da uspostavi ssh konekciju sa tim hostom, ssh klijent dobija javni ključ od udaljenog hosta i proverava da li postoji u fajlu sa poznatim hostovima. Ako se pronađe podudaran ključ, klijent generiše nasumični broj, šifruje ga pomoću javnog ključa, a zatim ga vraća nazad do udaljenog hosta. Ako host može da dešifruje broj, onda je autentičan, jer samo udaljeni host zna privatni ključ. Ako nema podudarnog ključa, onda ssh generiše poruku sličnu prethodno prikazanoj. Ovo može da se desi ako host promeni javni ključ. U tom slučaju korisnik mora ponovo da verifikuje ključ. Ako postoji poklapanje, ali udaljeni host ne može da dešifruje broj, autentičnost nije potvrdena i nema konekcije.

Kada se konekcija uspostavi, klijent i server pregovaraju o ključu sesije koji se koristi za šifrovanje narednih podataka koji će biti razmenjeni. Utvrdivanje ključa sesije razlikuje se u zavisnosti od verzije ssh. Na primer, to može da bude nasumičan broj koji klijent generiše. Pošto klijent šifruje nasumični broj sa javnim ključem udaljenog hosta, jedino host može da ga dešifruje. Klijent i server mogu da izvršavaju i DiffieHellman razmenu ključa.

Program ssh obezbeduje i autentifikaciju klijenta. U zavisnosti od verzije, postoji nekoliko načina da se to izvede. Jedan način je da se od korisnika zahteva da unese privatnu lozinku. Sledeći način omogućava korisniku da se bezbedno uloguje bez unošenja lozinke. Sledeći koraci, pokrenuti na Linux sistemu, pokazuju mogući pristup za postavljanje autentičnosti klijenta.

1. Korisnik izdaje komandu `ssh - keygen - t dsa`. Ova komanda kreira javni i privatni ključ koji se smeštaju u fajlovima id^dsa.pub i id_dsa, respektivno. Ti fajlovi su smešteni u poddirektorijumu .ssh u korisnikovom lokalnom matičnom direktorijumu.

2. Korisnik kopira fajl id_dsa.pub u drugi fajl authorized_keys u poddirektorijumu .ssh korisnikovog matičnog direktorijuma na udaljenoj mašini. Napomenimo da se javni ključ sada nalazi na oba hosta.

3. Korisnik inicira konekciju pomoću komande `ssh` *naziv_udaljenog_hosta.* Korisnik se povezuje bez unošenja lozinke.

Kako se ovo desilo? Umesto da se za autentifikaciju korisnika koriste lozinke, ssh je upotrebio par ključeva koji je korisnik generisao. Slede opšti koraci razmene:

1. U toku inicijalnog setupa ssh server dobija javni ključ od korisnika iz fajla authorized__keys. Zatim, generiše nasumični broj i šifruje ga pomoću korisnikovog javnog ključa.

2. ssh server šalje šifrovani ključ do ssh klijenta kako bi ga ovaj dešifrovao pomoću korisnikovog privatnog ključa.

3. ssh klijent dešifruje broj, koristeći privatni ključ smešten u id_dsa, i rezultat šalje nazad do servera.

4. Server utvrđuje da je klijent uspešno dešifrovao broj, čime je potvrdena njegova autentičnost (privatni ključ bi trebalo da zna samo korisnik).
5. Korisnik se ulogovao.

Pomoćni program ssh je veoma moćan i mnogo štošta se dešava od trenutka kada korisnik unese komandu ssh dok se ne uloguje. Postoje mnoge druge opcije za autentifikaciju i razmenu ključa. Tu su uključeni algoritmi za digitalne potpise, RSA, Blowfish, trostruki DES, AES i Diffie-Hellman razmena. U stvari, na Linuxu korisnik može da unese ssh *naziv_udaljenog_hosta* - v. Opcija - v postavlja ssh protokol u takozvani "brbljivi" (verbose) mod. On obaveštava korisnika o svim interakcijama koje se dešavaju dok pokušava da se poveže i proveri autentičnost. Ovo je interesantno pogledati, jer su objašnjenja veoma detaljna, ali je neophodno predznanje. Zainteresovani čitaoci treba da pogledaju prethodne reference, ili stranice sa uputstvom za Linux.

Dostupna je i freeware verzija ssh nazvana PuTTy. Pretraživanje izvedeno pomoću Internet pretraživačke mašine treba da pokaže lokacije sa kojih možete da preuzmete program. Održava se log fajl u kome se beleže sve razmene podataka koje korisnik inicira, slično verbose modu u Linuxu.

Transfer fajlova

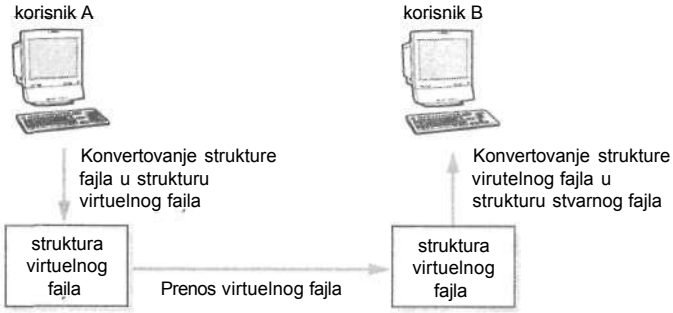
Jedna od najčešdh mrežnih aplikacija je transfer fajlova. Ima brojne primene. Ljudi koji su angažovani na grupnim projektima, ili izvode razna istraživanja često moraju da dele fajlove. Mogućnost pristupa i transfera fajlova ima suštinski značaj za deljenje informacija. Predavači kreiraju fajlove za studente koji ih koriste za polaganje praktičnih delova ispita. Korisnici mogu da prenose fajlove sa kućnog kompjutera na fajl server na poslu kako bi imali rezervne kopije onoga što rade.

U većini slučajeva svi fajlovi se nalaze na jednom mestu i kontroliše ih fajl server. Kada korisnik želi da prenese fajl, postavlja zahtev odgovarajućoj aplikaciji. ZaUm, proučava mrežne protokole za transfer fajlova. Međutim, postoje problemi koje je neophodno rešiti.

Jedan je struktura fajlova, koja može da bude razlidta. Na primer, neki fajlovi se sastoje od jednostavnih sekvenci bajtova. Drugi su takozvani *ravni fajlovi (flatfiles),* koji se sastoje iz linearne sekvence zapisa. *Hašovani fajlovi (hashed files)* omogućavaju nasumićni pristup zapisima pomoću vrednosti ključa. *Hijerarhijski fajlovi (hierarchial files)* mogu da organizuju sve pojave ključnog polja u strukturi stabla.* Ove razlike mogu da izazovu probleme ako odredišni sistem ne podržava struktum izvornog fajla.

Jedan od mogućih načina uprošćava transfere između nekompatibilnih sistema definisanjem strukture virutelnog fajla, koju podržava samo mreža, radi transfera fajlova. Na slici 11.38 prikazan je proces transfera. Korisnik A želi da prenese fajl do korisnika B, ali oni rade na kompjuterima koji podržavaju različite fajl sisteme. Fajl korisnika A mora da se prevede u strukturu definisanu mrežom izmedu njih. Nakon toga, stvarnim transferom upravlja protokol za transfer fajlova i fajl se na kraju konvertuje u strukturu koju podržava kompjuter korisnika B.

* Morate da razlikujele strukturu fajla od njegove implementacije. Na primer, faj! može da bude hijerarhijski, ali postoje razne implementacije hijerarhijske strukture.



SLIKA 11.38 *Transfer fajla izmedu kompjutera koji podržavaju različite strukture fajlova*

Struktura virtuelnog fajla mora da saćuva suštinske gradivne elemente fajla. Na primer, mora da sadrži naziv fajla, atribute i kodirane informacije o stvarnoj strukturi kako bi se omogućilo ispravno prevođenje. Naravno, mora da sadrži i podatke.

Sledeći problem transfera fajlova je dostupnost. Sistem za transfer fajlova ne treba da odobri svaki zahtev. Mora da sadrži zaštitu. Mogu li se fajlovi prenositi u oba smera? Da li se onome ko zahteva fajl uopšte dopušta pristup? Službe zadužene za čuvanje reda i zakona definitivno ne bi htele da njihov sistem dopusti pristup svim njihovim fajlovima. Mraju se uzeti u obzir i višestruki pristupi istim fajlovima. Da li su dopušteni? Za zaštitu fajla i kontrolu konkurentnosti koriste se lozinke, zabrane i ključevi.

FTP Jedan od najčešće korišćenih protokola za transfer fajlova naziva se upravo tako - *protokol za transfer fajlova (File Transfer Protocol),* ili FTP (kreatori se nisu "proslavili" maštovitošću). To je još jedan protokol iz TCP/IP skupa i formira se na istoj klijent/server paradigmi kao Telnet. Korisnik koji radi sa lokalnim FTP programom usposatvlja vezu sa udaljenim sajtom koji takode pokreće FTP. Kao i kod Telneta, ovo može da se izvede na nekoliko načina. Jedan način podrazumeva jednostavno unošenje sledeće komande na Linuxu

```
ftp tekstualna-adresa
```

čime se uspostavlja konekcija sa naznačenim udaljenim kompjuterom, slično što radi i Telnet.* Drugi način podrazumeva unošenje

```
ftp
```

i čekanje da se pojavi prompt `ftp>`. Nakon toga, korisnik unosi

```
open tekstualna-adresa
```

radi uspostavljanja konekcije. U zavisnosti od lokalnog sistema, ponekad se, umesto open, koristi komanda `connect`. Kada se poveže, od korisnika se zahteva identifikacija sa lozinkom.

' Kao i Telnet, FrP može da se pokreće iz GUI-a, ili iz Web pretraživača. Nastavljamo kao da ste uneli sopstvene komande (funkcioniše isto u svakom slučaju).

Tabela 11.3: FTP komande

Komanda	Značenje
`Cd`	Menja radni direktorijum na udaljenom hostu.
`close`	Zatvara FTP konekciju.
`dir ili Is`	Obezbeduje listu direktorijuma koji se nalaze u tekućem direktorijumu.
`get`	Kopira naznačeni fajl sa udaljenog hosta na lokalni sistem. Lokalni fajl dobija isti naizv kao i fajl na udaljenom hostu. U slučaju da postoji nekompatibilnost u konvenciji imenovanja, ili ako korisnik jednostavno želi da preneti fajl drugačije nazove, lokalni naziv fajla može da se naznači kao drugi parametar komande.
`giob`	Ponaša se kao preklopnik koji dopušta, ili zabranjuje upotrebu džoker znakova. Na primer, ako Je * džoker znak i njena upotreba je dopuštena, onda se sa mget * .TXT dobijaju svi fajlovi sa ekstenzijom .txt.
`help`	Prikazuje listu svih klijentskih FTP komandi.
`mget`	Kopira više fajlova sa udaljenog hosta na lokalni sistem.
`mput`	Kopira više fajlova sa lokalnog sistema na udaljeni host, na kome je dopušteno kreiranje novih fajlova.
`put`	Kopira naznačeni fajl sa lokalnog sistema na udaljeni host ako to udaljeni host dopušta.
`pwd`	Prikazuje tekući radni direktorijum na udaljenom hostu.
`quit`	Završava FTP.
`remotehelp`	Prikazuje listu svih serverskih FTP komandi.

Nakon unošenja odgovarajuće identifikacije i lozinke, korisnik dobija na korišćenje podirektorijume, listu direktorijuma i kopije fajlova.

Mnogi sajtovi postavljaju svoje fajlove da budu dostupni opštoj Internet zajednici. Ovo znači da korisnik može da im pristupi bez potrebe da ima nalog na toj mašini. Kada se korisnik povezuje na sajt, obično se prijavljuje sa korisničkim imenom "anonymous", ili "guest", ili sa svojom email adresom kao lozinkom. Email adresa se koristi za praćenje pristupa. Ova aplikacija je obično poznata kao **anonimni FTP.** *

Na površini, FTP izgleda isto kao Telnet: obe aplikacije korisniku omogućavaju uspostavljanje konekcije sa udaljenim mašinama. Razlika je u tome što Telnet zahteva legitimno logovanje, dok FTP prvenstveno obezbeđuje pristup određenim fajlovima i direktorijumima.

Kada se FTP konekcija uspostavi, korisnik vidi prompt `ftp>` . U tom trenutku on ima više raspoloživih opcija. Kao i ranije, nećemo pokušavati da predstavimo sve komande; objasnićemo samo neke najčešće korišćene iz tabele 11.3. Korisnici mogu da dobiju informacije o komandama pomoću komande `help`, ili unošenjem znaka pitanja (?). Verovatno su najčešće korišćene komande cd, koja omogućava prelazak u radni direktorijum na udaljenom hostu, i `get`, koja kopira fajlove sa udaljenog na lokalni sajt.

* Zbog ogromnog broja sajtova na Web serverima, anonimni FTP se manje koristi nego ranije, jer korisnici mogu da pristupe istim fajlovima i pomoću pretraživača i HTTP protokola. U sledećem poglavlju ih detaljnije predstavljamo.

Anonimni FTP je činio fajlove i tehničke izveštaje dostupnim celoj lnternet zajednici pre nego što su Web serveri postali uobičajeni. Dostupni fajlovi su zavisili od toga šta su administratori udaljenog sajta postavili na anonimni FTP nalog. Anonimni FTP sajtovi i dalje postoje; pokazaćemo jedan primerza korišćenje FTP-a kako bi se dobio pristup širokom opsegu informacija o mrežama i komunikacijama preko niza dokumenata poznatih kao Request for Comments (RFC) serije. To su beleške o istraživanjima koje su dostupne u elektronskoj i štampanoj formi. Obuhvataju široki dijapazon tema, lao što su lnternet protokoli, upravljanje mrežama, administriranje, email i mrežni standardi.

Jedan repozitorijum za RFC dokumente nalazi se na ftp.ietf.org. RFC serije su numerisane i objavljuju se u formi RFCxxxx, gde je oznaka xxxx četvorocifrena. Ovde nećemo ni pokušavati da dajemo listu RFC-a, jer bi nam za to bilo potrebno oko 40 stranica. U tabeli 11.4 navedeno je nekoliko RFC-a koji su "vezani" za teme obrađene u ovoj knjizi. Za konkretne teme postoji još mnogo dokumenata.

Tabela 11.4: RFC dokumenti

Tema	RFC broj
Border Gateway Protokol (BGP)	1105, 1266, 1265, 1771
Domain Name System (DNS)	1591, 1101
File Transfer Protocol (FTP)	114, 172, 265, 354, 542, 765, 959
Hypertext Transfer Protocol (HTTP)	2068,2616
Internet Control Message Protocol (ICMP)	777, 792
Internet Protocol (IP)	791,760
Internet Protocol verzija 6 (IPv6)	2460
Internet Protocol preko ATM-a	1577
IPSec	2401
Open Shortest Path First (OSPF) protokol verzija 2	2328
Routing Information Protocol (RIP)	1387, 1388, 1721, 1723, 1058
Resource Reservation Protocol (RSVP)	2205
Simple MaN Transfer Protocol (SMTP)	788,821
Simple Netvvork Management Protocol (SNMP) verzija 2	1901-1907
Simple Netvvork Management Protocol (SNMP) verzija 3	2570-2575
specifikacija Telnet protokola	764, 854
Transmission Control Protocol (TCP)	793,761,675
Preno IP datagrama preko IEEE 802 mreža	1042
Prenos IP datagrama preko javnih mreža za prenos podataka	877
Uniform Resource Locators (URL) na Webu	1738
User Datagram Protocol (UDP)	768
kritike X.25	874
X.400 email standard	822,987, 1616
X.509 sertifikati	2459, 2560

```
/home/shayw-$>ftp ftp.ietf.oro
Connected to 4.17.168.6 (4:17.168.6).
228 www.ietf.org NcFTPd Server (licensed copy) ready.
Name (ftp.ietf.org:shayw): anonymous
331 Guest login ok, send your complete e-mail address as password.
Password;  shaywBuwgb.edu
238-You are user #3 of 50 simultaneous users allowed.
230-
230 Logged in anonymously.
Remote system type is UNIX1
Using binary mode to transfer files.
ftp> Is
227 Entering Passive Mode (4,17,168,6,206,14)
150 Data connection accepted from 143.200.128.235:34548; transfer starting.
lrwxrwxrwx  1    ftpuser  ftpusers  7       Jun  21  2001   bin-> usr/bin
drwxrwxr-x  250  ftpuser  ftpusers  4608    May  3   2001   concluded wg-ietf-mail-archive
dr-xr-xr-x  2    ftpuser  ftpusers  512     JuI  19  1996   dev
dr-xr-xr-x  2    ftpuser  ftpusers  512     Aug  13  1998   etc
drwxr-xr-x  3    ftpuser  ftpusers  6656    Sep  6   15:06  iesg
drwxrwxr-x  487  ftpuser  ftpusers  9216    Sep  13  16:07  ietf
drwxr-xr-x  6    ftpuser  ftpusers  512     Feb  2   1999   ietf-gopher
drwxrwxr-x  212  ftpuser  ftpusers  4096    Sep  15  15:33  ietf-mail archive
drwxr-xr-x  16   ftpuser  ftpusers  512     May  30  2001   ietf-online-proceedings
drwxrwxr-x  2    ftpuser  ftpusers  184832  Oct  1   13:54  internet-drafts
dr-xr-xr-x  2    ftpuser  ftpusers  512     Aug  29  1996   lib
drwxr-xr-x  2    ftpuser  ftpusers  512     Apr  17  1996   lost+found
drwxr-xr-x  3    ftpuser  ftpusers  512     Sep  13  09:47  pub
drwxrwxr-x  9    ftpuser  ftpusers  74752   Oct  1   19:29  rfc
dr-xr-xr-x  5    ftpuser  ftpusers  512     Apr  18  1996   usr
226 Listing completed.
ftp> Cd rfo
250- "/rfc" is new cwd.
250-
250-* . I . . . . . = . . . . . = . . . . = . . . . . = .*
250-*
250-* This directory is maintained by the RFC Editor. If you experience*
250-* any problems, please report thera to rfc-editor9rfc-editor.org
250-*
250-* . « » . . • . « » « « . . . . . » . . « . . = • » . » « » » « « *
250
ftp> get rfc-index.txt
local: rfc-index.txt remote: rfc-index.txt
227 Entering Passive Mode (4,17,168,6.218,101)
150 Data connection accepted from 143.200.128.235:34549;       transfer starting
for rfc-index.txt (524437 bytes).
226 Transfer completed.
524437 bytes received in 4.42 sees (1.2e-i-02 Kbytes/sec)
ftp> quit
221 Goodbye.
 /home/shayw-$>
```

SLIKA 11.39 *Primer korišćenja FTP-a za prenos fajla*

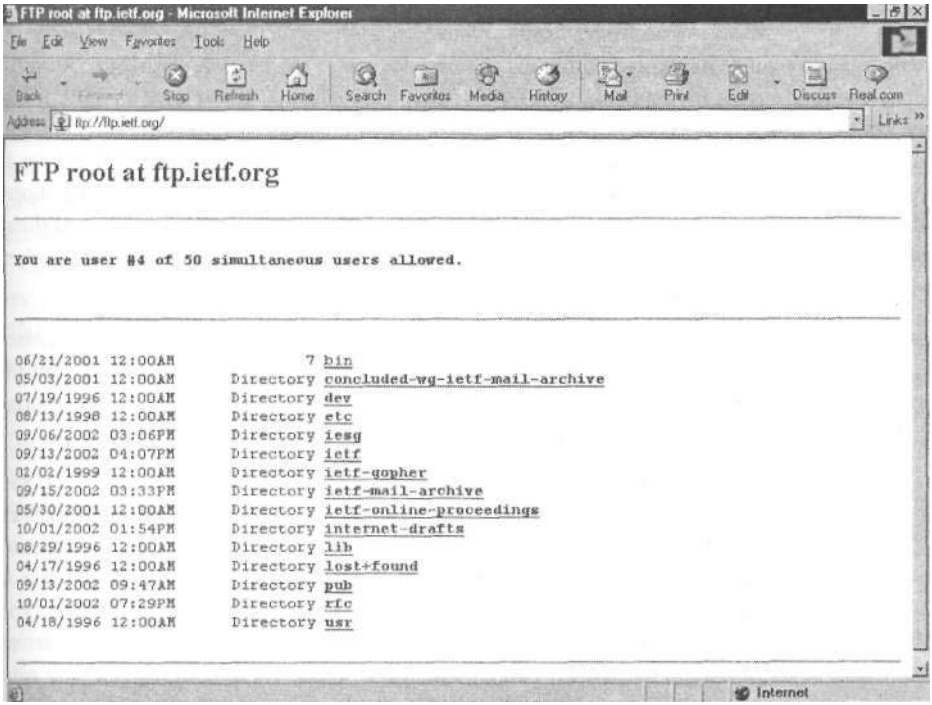
Sajt repozitorijuma RFC dokumenata održava RFC listu u fajlu kome može da se pristupi pomoću FTP-a. Na slici 11.39 prikazano je kako je fajl pribavljen preko Linux sistema. Boldirani karakteri predstavljaju one koje je uneo korisnik, a obični karakteri odgovaraju FTP odzivima. Linija 1 prikazuje FTP komandu za povezivanje na udaljeni sajt ftp.ietf.org. Linija 4 odgovara zahtevu za logovanje i ulazu za anonimno logovanje. Linija 6 zahteva lozinku, što je u ovom slučaju autorova email adresa. Kao i kod većine sistema, na lozinku nisu dobijene nikakve povratne informacije, većsu ovde prikazane samo radi ilustrovanja interakcije. Linija 12 prikazuje sadržaj tekućeg direktorijuma na FTP sajtu. Linija 31 menja radni direktorijum na poddirektorijum rfc tekućeg direktorijuma. U ovom poddirektorijumu se nalaze RFC dokumenti. Konačno, linija 41 zahteva prenos kopije fajla pod nazivom rfc-index.txt na lokalni sajt. U ovom fajlu se nalazi lista svih RFC dokumenata raspoloživih utom repozitorijumu i obuhvaćenih tema. Kopirani fajl će takođe nositi naziv rfc-index.txt. Preostale linije ukazuju na status transfera dok je toku. FTP se završava u liniji 48.

Ako želite kopiju određenog RFC dokumenta, unesite

```
get rfcxxxx.txt
```

gde je *xxxx* četvorocifreni RFC broj.

Naravno, ovo nije jedini način da se pristupi takvim dokumentima. Ako imate pretraživač i Internet konekciju, možete ih koristiti za pristup FTP sajtovima. Jednostavno, navedite *ftp://ftp-sajt* kao LIRL. Naslici 11.40 prikazan je rezultat unošenja ftp://ftp.ietf.org u adresno polje Internet Explorera.



SLIKA 11.40 *Korišćenje FTP-a pomoću pretraživača*

Uporedite prikazane direktorijume sa onima koji su prikazar.i u linijama 16 do 29 na slici 11.39. Direktorijumu pristupite jednostavnim klikom, a kada kliknete naziv fajla možete da vidite sadržaj fajla.

Secure Copy Kao i Telnet, FTP nije bezbedan. Nema ni šifrovanja, ni autentifikacije. Medutim, program Secure Copy (scp) obezbeduje te funkcije. Funkcioniše slično FTP-u, ali koristi istu autentifikaciju i zaštitu kao ssh. Jedan način za kopiranje fajla pomoću scp je unošenje sledeće komande (ponovo pretpostavljamo da se koristi Linux):

```
scp korisnickoime@host:nazivfajla1 nazivfajla2
```

U stvari, sasvim je jednostavno - kopira se fajl *(nazivfajlal)* sa naloga *(korisnickoime)* na naznačeni host. Fajl se na lokalni host smešta kao *nazivfajla2.* Nakon što korisnik unese ovu komandu, scp traži od korisnika da unese lozinku za nalog na udaljenom hostu. Naravno, postoje različite opcije koje korisniku omogućavaju naznačavanje ovih "stvari" i razni metodi koji se koriste za šifrovanje i proveru autentičnosti. Ako korisnik pokrene scp u "brbljivom" modu (pomoću opcije ■ v), razmena između lokalnog i udaljenog hosta prilično podseća na onu kod ssh.

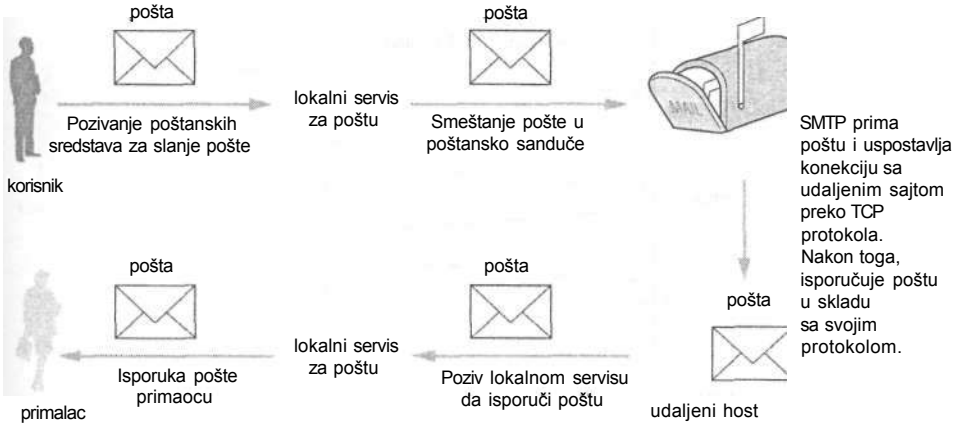
Simple Mail Transfer protokol (SMTP)

Jedna od najčešće korišćenih aplikacija na mreži sigurno je elektronska pošta, tj. mogućnost slanja poruke, ili fajla do specifičnog korisnika na lokalnom, ili udaljenom sajtu. Poruku obično šaljete tako što naznačite email adresu primaoca. Uobičajeni format je *ime@iekstualna_adresa_hosta,* Poruka se baferuje na odredišnom sajtu i dostupna je samo korisniku kome je namenjena.

Email ima neke sličnosti sa protokolima za transfer fajlova. Na primer, u oba slučaja se koriste klijent i server za pregovaranje o transferu podataka. Međutim, email obično šalje fajl do specifičnog korisnika, po čijem se nalogu poruka baferuje. Email klijent i server rade u pozadini. Na primer, ako uzimate fajl pomoću FTP-a, obično morate da sačekate dok ne stigne pre nego što dobijete mogučnost da uradite nešto drugo. Ako šaljete (ili primate) fajl pomoću emaila, možete da radite nešto drugo dok klijent i server izvršavaju isporuku email poruke u pozadini. U slučaju prijema elektronske pošte, ne morate čak da budete ni ulogovani. Kada se sledeći put ulogujete, bićete obavešteni o novoj pošti.

Standardni email protokol u TCP/IP skupu protokola je Mail Transfer Protocol (SMTP). Pokreće se iznad TCP/IP protokola, a ispod bilo kog lokalnog email servisa. Njegova primarna odgovornost je da se osigura da se poruka prenese između različitih hostova. Sa druge strane, lokalni servis je odgovoran za distribuiranje emaila do specifičnih primaoca.

Na slici 11.41 prikazana je interakcija između lokalnog emaila, SMTP i TCP protokola. Kada korisnik pošalje email, lokalni kapaciteti za email utvrduju da li je adresa lokalna, ili zahteva povezivanje na udaljeni sajt. Ukoliko je potrebno povezivanje na udaljeni sajt, lokalni kapaciteti za email skladište poštu (slično kao kada ubacite pismo u poštansko sanduče), gde čeka na klijentov SMTP. Kada klijentov SMTP isporuči email, najpre poziva TCP da uspostavi konekciju sa udaljenim sajtom. Kada je konekcija uspostavljena, SMTP protokoli na klijentu i serveru razmenjuju pakete i na kraju isporučuju email. Na udaljenom kraju lokalni kapaciteti za email uzimaju poštu i isporučuju je namenjenom primaocu.

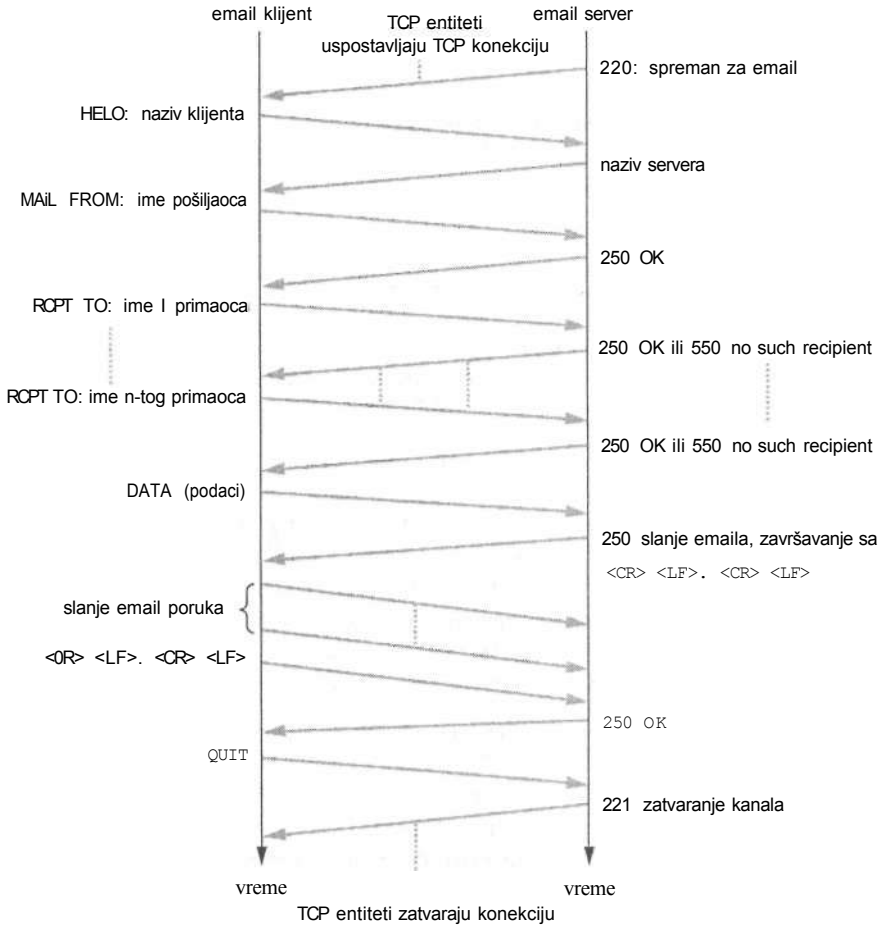


SLIKA 11.41 *SMTP u mterakaji sa loltalnim servisom za postu i TCP protokolom*

Na slici 11.42 prikazana je razmena paketa između klijenta i servera. Paketi se još nazivaju i *jedinice podataka SMTP protokola* (SMTP *protokol dala units* - PDU), ili, jednostavno, *komande.* Kada se uspostavi TCP konekcija, server šalje 220 PDU, ukazujući da je spreman za prijem pošte. Broj 220 služi za identifikovanje tipa paketa. Nakon toga, klijent i server razmenjuju identitete svojih sajtova. Zatim, klijent šalje MAIL FROM PDU,ukazujući da je stigla pošta i identifikuje pošiljaoca. Ako je server spreman da prihvati poštu od pošiljaoca, odgovara sa 250 OK PDU.

Server tada šalje jedan, ili više RCPT TO PDU-a, naznačavajući primaoce kako bi se utvrdilo da li oni postoje pre nego što se pošta pošalje. Za svakog primaoca server odgovara sa 250 OK PDU (znači da primalac postoji), ili 550 Recipient Not Here PDU (znači da primalac ne postoji). Nakon što se identifikuju svi primaoci, klijent šalje DATA PDU, ukazujući da će početi prenos pošle. Server odgovara sa 354 Start Mail PDU, čime daje "zeleno svetlo" za početak slanja, i definiše sekvencu koju klijent treba da koristi za označavanje kraja pošte. U ovom slučaju sekvenca je <CR><LF>.<CR><LF>. Klijent šalje poštu sa PDU-ima fiksne veličine, postavljajući ovu sekvencu na kraj pošte. Kada server dobije poslednji PDU, šalje potvrdu primaocu sa sledećim 250 OK PDU. Konačno, klijent i server razmenjuju PDU-e, ukazujući da se isporuka pošte završava, i TCP otpušta konekciju.

Ovaj opis je obuhvatio osnovne crte funkcionalnosti SMTP protokola; nismo zalazili u detalje PDU formata, ili teme kao što su prosleđivanje poruka, ili reagovanje na nekonzistentne adrese. Više informacija o SMTP protokolu može da se nade u referencama [CoOO] i [Ru89] i u RFC dokumemima 788 i 821.

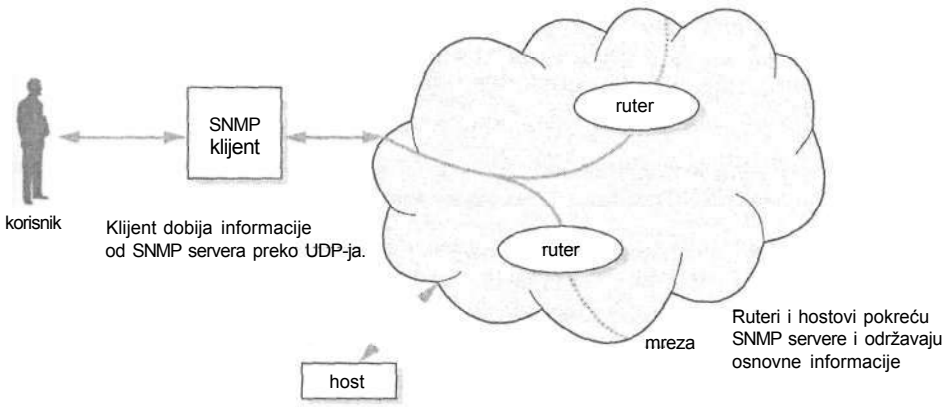


SLIKA 11.42 *Slanje emaila pomoću SMTP-a*

Simple Netvvork Management protokol (SNMP)

Simple Netvvork Management Protocol (SNMP) je protokol za upravljanje koji je dizajniran da bi se osiguralo ispravno funkcionisanje mrežnih protokola i uređaja. Omogućava menadžerima da lociraju probleme i izvedu podešavanja razmenom sekvence komandi izmedu klijenta i servera. Za razliku od prethodne aplikacije, pokreće se iznad UDP-ja umesto TCP-ja. Pošto je reč **o** značajnom delu upravljanja Internetom, zavreduje posebnu pažnju.

Na slici 11.43 prikazane su SNMP arhitekture. Menadžer mreže pokreće klijentski program za upravljanje na sajtu koji komunicira sa programom za upravljanje na serveru na drugom sajtu. Serverski programi se, obično, pokreću na udaljenim hostovima, a posebno na mrežnim ruterima.



SLIKA 11.43 *SNMP arhitektura*

Oba programa za upravljanje koriste komande definisane SNMP protokolom. Komande prvenstveno definišu kako se zahtevaju informacije od servera i kako se informacije šalju do servera, ili klijenta.

SNMP ima nekoliko ciljeva (opisanih u dokumentu RFC 1157). **Prvi** je uprošćavanje funkcija upravljanja kako bi se redukovale cene podrške i kako bi se SNMP lakše koristio. Drugo, mora da bude proširiv da bi bile omogućene buduće izmene karakteristika mrežnih operacija i upravljanja. Treće, protokol mora da bude nezavisan od specifičnosti dizajna hosta, ili rutera.

Rezultat je protokol sloja aplikacije koji saraduje sa transportnim servisima.

Pošto je SNMP aplikacija za upravljanje, mora da zna kojim procesima treba da upravlja i kako da se referencira na njih. Ruteri i hostovi kojima SNMP upravlja nazivaju je *objekti.* Objekat ima formalnu definiciju u skladu sa **ASN. 1 (Abstract Syntax Notation 1), formalnim** jezikom dizajniranim isključivo za definisanje PDU formata i objekata. O formalnom tretmanu objekata i ASN.1 dodatne informacije možete da pronadete u referencama [SuOO], [CoOO] i [Co99[.

Standard Management Information Base Server svakog objekta održava bazu podataka sa informadjama koje opisuju njegove karakteristike i aktivnosti. Pošto postoje različiti tipovi objekata, standard precizno definiše šta treba da se održava. Ovaj standard **(Management Information Base - MIB)** bio je definisan grupom koju je predložio SNMP. Osam kategorija informacija je defmisano sa MIB. Kompletan opis svake kategorije možete da pronadete u referencama [SuOO], [CoOO], [St99] i [MiOO]. Ovde ćemo navesti svaku kategoriju sa nekim primerima informacija koje sadrže.

- **System (Sistem)** Opisuje operativni sistem hosta, ili rutera i sadrži informacije kao što su vreme kada se server startovao, opis svakog uredaja na kome se pokrece, lokadja uredaja i informacije o osobi za kontakt.

- **Interface (Interfejs)** Opisuje svaki mrežni interfejs i sadrži stavke kao što su veličina MTU (videti sekviju 11.2), brzina prenos, broj paketa odbačenih zbog različitih razloga, broj prenetih i primljenih bajtova, broj interfejsa i opis interfejsa.
- Address **translation** (Prevodenje adresa) Sadrži tabelu koja se koristi za promenu IP adrese u adresu specifičnu za konkretnu mrežu.
- **IP** Opisuje informacije specifične za Internet protokol. Medu informacije ubrajaju se vrednost Time to Live za IP pakete, broj datagrama eliminisanih zbog različitih razloga, broj prosleđenih i isporučenih datagrama do transportnog protokola i broj primljenih datagrama od protokola veze, broj kreiranih fragmenata, broj ponovo sastavljenih datagrama i tabele rutiranja.
- **ICMP** Opisuje informacije specifične za ICMP. Sadrži prvenstveno više brojača koji prate broj kontrolnih poruka svih tipova (videti odeljak 11.2) koje šalje ICMP.
- **TCP** Medu stavkama koje pamti nalaze se dužina time-outa, broj konekcija, broj poslatih i primljenih segmenata, maksimalni broj simultanih konekcija, IP adresa svakog entiteta koji koristi TCP, IP adresa daljinske konekcije i broj neuspešnih pokušaja uspostavljanja konekcije.
- **UDP** Između ostalog, sadrži broj isporučenih, odbačenih, ili primljenih datagrama i IP adrese entiteta koji koriste UDP.
- **EGP (Exterior Gateway Protocol)** Ovo je protokol za razmenu informacija o izmedu dve autonomne mreže u mreži sa više manjih nezavisnih mreža. Kao i kod drugih kategorija, MIB sadrži brojače za pračenje broja poslatih i primljenih EGP poruka.

SNMP komande Programi za upravljanje koji koriste SNMP izvršavaju se asinhrono - šalju zahteve, ali mogu da rade i nešto drugo dok čekaju na odzive. U opštem slučaju, zahtevi, ili PDU-i zahtevaju informacije sa servera, šalju informacije programu za upravljanje na udaljenom sajtu, ili reaguju na specijalne uslove. SNMP definiše pet PDU formata:

1. **GetRequest** Ova komanda izaziva slanje GetRequest PDU-a* koji sadrži kod komande, naziv objekta i specifikaciju MIB promenljive. Prijemni entitet reaguje slanjem GetResponse PDU-a u kome se nalaze vrednost tražene promenljive i kod greške ako postoji greška.

* Specifični PDU formal i mehanizam identifikovanja MIB promenljivih je složen. Više detalja o njima sadrže reference [MiOO] i [St99].

2. GetNextRequest Ova komanda je slična komandi GetRequest, osim što se zahtevaju vrednosti promenljivih koje "slede" one koje su specificirane u PDU-u. Notacija je zasnovana na leksikografskom redosledu koji MIB dizajn određuje. Ovo je posebno korisno za prolazak kroz tabele koje održava server za upravljanje.

3. GetResponse PDU poslat kao odgovor na prethodno primljeni GetRequest PDU sadrži tražene vrednosti, ili kodove grešaka.

4. SetRequest Ova komanda omogućava menadžeru da ažurira vrednosti MIB promenljivih koje se održavaju programima za upravljanje na udaljenom sajtu i za daljinsku izmenu karakteristika određenog objekta, što može da utiče na mrežne operacije. Format ne narušava nikakve bezbednosne mere koje sprečavaju neautorizovana ažuriranja.

5. Trap Ovaj PDU se šalje sa servera do menadžera kada se ispune specifični uslovi, ili se dese odredeni dogadaji. Omogučava menadžeru da upozna sve promene u operativnom okruženju. Sledi lista nekih Trap PDLl-a i njihovih događaja:

    a. Coldstart trap Program za upravljanje je reinicijalizovan, sa potencijalnim promenama u karakteristikama objekta.

    b. Warmstart trap Reinicijalizacija se desila, ali nije došlo do promene bilo kojih karakteristika.

    c. Linkdown trap Komunikacioni link je "otkazao".

    d. Linkup trap Ponovo je uspostavljena ranije "otkazana" komunikacija.

    e. EgpNeighborLoss trap Stanica je izgubila kontakt sa ravnopravnim EGP susedom.

    f. Authentication failure trap Primljen je SNMP PDU koji nije prošao proveru autentičnosti.

SNMPv2 je dizajniran tako da prevaziđe neke slabosti SNMP-a. Na primer, SNMP je kritikovan zbog toga što je imao jednostavan format komande i što je bio neophodan veći broj paketa. SNMPv2 ima više opcija za ramenu poruka, tako da je obezbeđena efikasnija komunikacija izmedu klijenata i hostova. Primer je komanda CetBulkRequest, koja pribavlja informacije koje su ranije sakupljene pomoću više zahteva PDU-ima. Broj kodova greške je takode proširen, tako da su obezbeđene opširnije informacije o uzrocima grešaka. Osim toga, povečana je fleksibilnost koja omogućava pokretanje SNMPv2 iznad više protokola, kao što su AppleTalk, IPX i OSI.

Postoji i SNMPv3, koji uključuje tri nivoa zaštite. Najniži nivo ne obezbeduje ni autentifikaciju, ni zaštitu, tako da je kompatibilan sa prethodnim verzijama. Sledeci nivo koristi autentifikaciju zasnovanu na SHA, ili MD5 algoritmima kako bi se osiguralo da se PDU nije promenio, ili da bi se verifikovao njegov izvor. Najviši nivo obezbeđuje šifrovanje pomoću CBC moda DES šifrovanja.

Kao i u slučaju bilo kojeg drugog protokola, verzije SNMP-a nisu jedini raspoloživi protokoli za upravljanje. ISO protokol za upravljanje je Common Management Information Protocol (CMIP). Kada se koristi preko TCP konekcije, CMIP je poznat kao CMOT (CMIP over TCP). CMIP je složeniji od SNMP-a i smatra se da ie prikladniji protokol za veće mreže. Remote

Monitoring (RMON) je sledeći protokol za upravljanje. Ima dve zasebne komponente: *agentsku, Ui ispitnu stanicu (agent station,* ili *probe)* i *upmvljačku stanicu (management station).* Ispitna stanica može da bude specifični uređaj namenjen za prikupljanje informacija, ili može da bude softverski agent koji se pokreće u postojećem mrežnom čvoru, kao što su radna stanica, server, ruter, ili komutator. U poslednjoj varijanti softver se pokreće u pozadini, dok uređaj izvršava svoje normalne dužnosti, što je sporije od normalnog tempa. Svrha ispitne stanice je prikupljanje informacija o aktivnostima na mreži, kao što su broj poslatih i ispuštenih paketa, broj multicast paketa, izvori koji šalju pakete i greške LAN segmenta. U stvari, IETF defmiše različite grupe statističkih podataka i preporuke za njihovo korišćenje. Nakon toga, ispitna stanica šalje prikupljene informacije do upravljačke stanice koja te informacije smešta u MIB. Zatim, proizvođač softvera analizira podatke kako bi obezbedio informacije o performansama mreže i za praćenje problema na mreži. Čitaoci koji su zainteresovani za kompletnija objašnjenja upravljanja mrežama mogu da pogledaju reference [SuOO], [MiOO] ili [St99].

11.6 Zaključak

U ovom poglavlju smo se bavili prvenstveno protokolima koji defmišu šta je Internet i neke aplikacije koje se pokreću na njemu. U opštem slučaju, Internet je kolekcija uređaja koji podržavaju TCP/IP. IP je protokol sloja 3, aTCP protokol sloja 4. Sledi pregled nekih najvažnijih karakteristika i problema "vezanih" za IP protokol:

- Omogućava transfer podataka preko nesrodnih mreža i razlike čini nevidljivim za protokole viših slojeva.
- Defmiše format paketa, opcije za rutiranje, tipove servisa i načine za rešavanje paketa koji su suviše veliki da bi "putovali" preko nekih mreža.
- Koristi hijerarhijsko rutiranje, tako da se adrese interpretiraju kao broj mreže iza kojeg sledi lokalni identifikator.
- Oslanja se na DNS (Domain Name Service), koji koristi distribuiranu bazu podataka za konvertovanje tekstualnih adresa u numeričke IP adrese. Obezbeduje multicasting preko različitih protokola. Internet Group Management Protocol (IGMP) funkcioniše izmedu hosta i lokalnog rutera i omogućava hostu da se pridruži različitim multicast grupama, ili da ih napusti. Zatim, oslanja se na Mbone, kolekciju rutera koja implementira rutiranje paketa Klase D. Funkcioniše usvajanjem protokola za rutiranje koji kreiraju multicast stablo preko koga paketi Klase D "putuju".
- Pošto IP ne obezbeđuje zahtevani kvalitet servisa za real-time aplikacije, razvijen je Resource Reservation Protocol (RSVP) - on ugrađuje poaike u IP pakete sa informacijama o odredenom toku podataka, koje zahtevaju rezervaciju dovoljnog broja resursa. Specifični detalji o tome kako se ovo izvodi zavise od rutera koje paketi prelaze na putu od izvora do odredišta.
- IP ne garantuje isporuku paketa. Zbog toga, Internet Control Message Protocol (ICMP) podnosi izveštaje o greškama i obezbeđuje ruterima najnovije informacije o uslovima na Internetu. Definiše različite kontrolne poruke i poruke o greškama i prenosi ih preko IP-a.

- Pošto IP nije bezbedan, razvijen je IPSec. Obezbeduje zaglavlje autentifikacije, kojim se proverava autentičnost paketa, i stavku Encapsulating Security Payload, koja obezbeduje šifrovanje i autentifikaciju paketa i protokol za razmenu ključa.
- Zbog neverovatnog porasta broja korisnika Intemeta i pojave novih tehnologija, razvijen je IPv6. On obezbeduje mnogo veći adresni prostor nego IPv4; teorijski, do 2^{128} jedinstvenih adresa. Osim toga, koristi višestruka zaglavlja paketa i ubrzava proces rutiranja. Proči će dosta vremena dok se IPv6 ne implementira na celom Internetu, tako da je za sada omogućeno uporedno funkcionisanje sa IPv4.

Transportni protokoli su protokoli najnižeg sloja koji se bave komunikacijama izmedu krajnjih korisnika, nezavisno od mrežnih operacija. Najčešće korišćeni protokol na Internetu je Transmission Control Protocol (TCP), konekciji orijentisani protokol. Neke od njegovih osnovnih funkcija su upravljanje konekcijom, kontrola toka i detekcija grešaka. Efektivno, TCP garantuje pouzdanu razmenu informacija. Neki značajni aspekti TCP-a su:

- defmicija formata jednog segmenta i za podatke i za kontrolu
- trosmerno usaglašavanje koje zahteva ne samo potvrdu zahteva za uspostavljanje konekcije, već i potvrdu potvrde
- mehanizam kredita za kontrolu toka sličan protokolima sa klizajućim prozorima
- upravljanje pomoću prozora zagušenja koji reaguje na promene uslova na mreži koje su izazvane velikom količinom saobraćaja

Postoje i drugi transportni protokoli. Jedan je UDP, protokol bez uspostavljanja veze koji ima manje troškove od TCP-a, ali i manje mogućnosti. Sledeći je Real-Time Transport Protocol (RTP), koji je dizajniran za podršku real-time aplikacija, kao što su audio, ili video aplikacije. Iako ne garantuje specifične QoS, pomaže redukovanje pomeranja paketa, tako da je omogućeno konzistentnije izvođenje multimedijalnih sadržaja; takođe kontroliše više nizova podataka kako bi se obezbedila sinhronizacija video i audio zapisa.

Postoji veliki broj aplikacija koje se izvršavaju na Internetu; mi smo obuhvatili samo nekoliko.

- Telnet omogučava korisnicima da se loguju na udaljene host kompjutere. Koordinira razmenu između lokalnih i udaljenih hostova.
- Program Secure Shell (ssh) izvršava iste funkcije kao i Telnet (i još neke dodatne), ali obezbeduje i šifrovanje i autentifikaciju hosta i korisnika.
- FTP korisnicima omogućava povezivanje na udaljene hostove, radi prikazivanja, ili preuzimanja fajlova.
- Secure Copy izvršava iste funkcije kao i FTP, ali obezbeduje šifrovanje i autentifikaciju.
- SMTP (Simple Mail Transfer Protocol) defmiše PDU-e i sekvencu razmena neophodnih za email servis.
- SNMP (Simple Netvvork Management Protocol) prikuplja informacije o mrežnom saobraćaju i statistikama. Informacije mogu da se analiziraju softverski, tako da se dobijaju statistički podaci za performanse mreže i za problematične oblasti.

Pitanja i zadaci za proveru

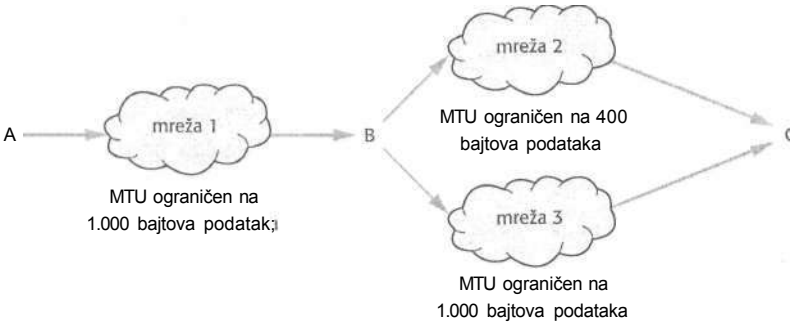
1. Šta radi Dynamic Host Configuration Protocol (DHCP)?
2. Adrese kao što su 143.200.128.162 i msa.uwgb.edu predstavljaju isti čvor Internetu. U čemu je razlika izmedu njih?
3. Navedite razlike između IP adresa Klase A, B, C i D.
4. Po čemu se besklasne adrese razlikuju od adresa Klase A, B, ili C?
5. U čemu je razlika između multicast i unicast adresa?
6. Šta je ICANN (Internet Corporation for Assigned Names and Numbers)?
7. Šta radi DNS?
8. Da li su sledeće tvrdnje tačne, ili netačne (zašto)?
    a. IP garantuje da će sve informacije stići do svojih odredišta.
    b. IP paketi pridaiženi konkretnoj aplikaciji, kao što je email, mogu da koriste razlidte rute do istog odredišta.
    c. Suštinska komponenta dizajna rutera je utvrđivanje kako ruter locira IP adresu u svojoj tabeli rutiranja.
    d. TCP i UDP funkcionišu slično kao protokoli sloja 4.
    e. Real-Time Transport Protocol garantuje kvalitet servisa za real-time aplikacije na transportnom sloju.
    f. Telnet dopušta logovanje na bilo koji nalog na udaljenoj mašini sve dok je mašina dostupna preko bilo koje mrežne konekdje.
    g. Program Secure Shell omogućava logovanje korisnika na udaljeni host (na kome korisnik ima nalog), bez unošenja lozinke.
    h. TCP i IP su protokoli OSI modela na slojevima 3 i 4, respektivno.
    i. Slanje emaila preko Interneta ne zahteva konekciju.
9. Koja je svrha deljenja DNS hijerarhije na zone?
10. Koji je razlog za postojanje polja Type of Service u IP paketu?
11. Zašto je fragmentiranje IP paketa ponekad neophodno?
12. Navedite giavne funkcije IP-ja.
13. Cemu služi polje Time to Live u IP paketu?
14. Sta je maksimalna jedinica transfera I kako ona utiče na Internet Protocol?
15. Navedite razlike izmedu Internet adrese i fizičke adrese.
16. Šta je dinamičko povezivanje?
17. Šta se podrazumeva pod kvalitetom servisa?

18. Šta je lnternet Management Group Protocol?
19. Šta je multicast stablo?
20. Šta je Mbone?
21. Koja je svrha Prune paketa kod Distance Vector Multicast Routing protokola?
22. Šta je Resource Reservation Protocol?
23. U čemu je razlika izmedu "mekog" t strogog stanja rutera?
24. Za šta se koristi lnternet Control Message Protocol?
25. Navedite lipične kontrolne poruke koje defmiše ICMP.
26. 1PV4 obezbeđuje 32-bitnu IP adresu, a 32-bitni broj može da ima oko četiri milijarde različitih vrednosti, Pošto predstavlja više od polovine stanovništva zemaljske kugle a (na globalnoj skali) većina ljudi nema lnternet konekcije, zašto ponestaje IPv4 adresa?
27. Po čemu se IPv4 fragmentacije razlikuje od 1PV6 fragmentacije?
28. Šta je tunelovanje?
29. Navedite nekoliko faktora koji doprinose mogućnosti IPv6 protokola za brže rutiranje.
30. Zašto je kod IPv6 eliminisana čeksuma iz zaglavlja paketa?
31. Cemu služe različita zaglavlja IPv6 paketa?
32. ŠtajelPSec?
33. Kako TCP koristi broj porta?
34. Šta predstavlja zahtev za vremensku oznaku?
35. U čemu je razlika između dvosmernog i trosmernog usaglašavanja?
36. Šta je pokazivač urgentnosti?
37. Šta je TCP kredit?
38. Šta je User Datagram Protocol?
39. Kako Real-Time Transport Protocol potpomaže podršku real-time aplikacija kada nema kontrolu nad rutiranjem u okviru mreže?
40. Šta je pomeranje paketa?
41. Šta predstavlja potencijalni problem kod razdvajanja audio i video niza podataka iz jednog izvora?
42. Šta je RTCP adaptiranje brzine?
43. ŠtajeTelnet?
44. Po čemu se ssh razlikuje od Telneta?
45. Program ssh može da autentifikuje korisnika pomoću lozinki, ili mehanizma javnog/privatnog ključa. Opišite razliku.

46. Šta je stuktura virtuelnog fajla?
47. Po čemu se anonimni FTP razlikuje od običnog?
48. ŠtajeRFC?
49. ŠtajeSMTP?
50. ŠtajeSNMP?
51. Šta predstavlja SNMP-ova Management Information Base?

Vežbe

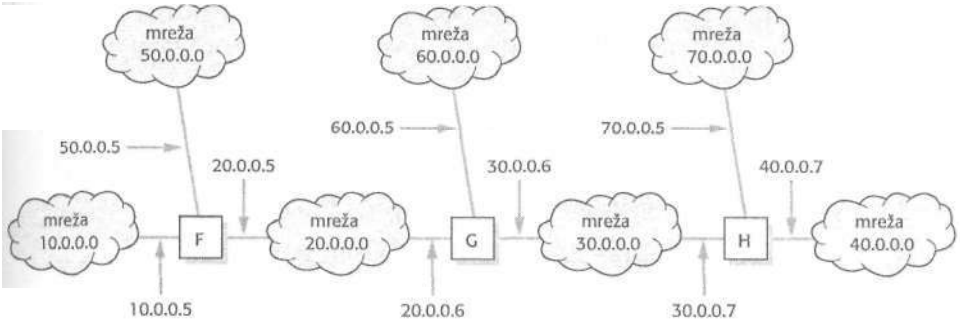
1. Klasifikujte sledeće adrese kao adrese Klase A, B, C, ili D.
   a. 183.104.200.32
   b. 230.4200.104.32
   c. 210.20.34.100
   d. 115.193.23.32
2. Šta je broj mreže u IP adresi 140.100.120.02 ako je maska podmreže 255.255.224.0?
3. Da li ima smisla posedovati masku podmreže 255.255.224.7? Zašto ima, ili nema?
4. Pretpostavimo da je nekoj organizaciji potrebno 8.000 IP adresa. Koliko je adresa Klase C potrebno? Ako su sukcesivne, opišite CIDR šemu adresiranja.
5. Ako koristite usluge ISP-ja, ili je Vaš kompjuter povezan na LAN, utvrdite IP adresu Vašeg kompjutera i masku podmreže. Zatim, utvrdite klasu adrese i podmrežu i lokalne ID-ove u IP adresi.
6. Pretpostavimo da ruter A sa slike primi IP paket u kome se nalazi 4.000 bajtova podataka, fragmentira paket i rutira fragmente do B preko mreže 1. B rutira sve fragmente, osim drugog do C, preko mreže 3. Međutim, on fragmentira drugi fragment i šalje nove fragmente do C preko mreže 2. Pokažite fragmente koje C prima i naznačite relevantne vrednosti u zaglavljima fragmenta.



7. Zašto je prilikom IP fragmentacije neophodno polje Identification u zaglavlju fragmenta? Zašto odredište jednostavno ne koristi izvornu adresu kako bi se utvrdilo koji su

paketi povezani i ne izvrši njihovo ponovno sastavljanje u skladu sa vrednostima polja Offset?

8. Kako izgleda tabela rutiranja za ruter G za međumrežu na pratećoj slici? Za svaku mrežu naznačite adresu rutera na koju se paket mora poslati. U slučaju direktne konekcije, naznačite da se paket mora isporučiti direktno na svoje odredište.



9. Kako izgledaju tabele rutiranja za rutere F i H iz prethodne vežbe?

10. Izaberite omiljeni Web sajt i utvrdite rutu do njega od Vaše tekuće lokacije.

11. Koje od sledećih aplikacija zahtevaju real-time kvalitet servisa?

    a. preuzimanje audio fajlova

    b. pristup udaljenom hostu pomoću Telneta, ili ssh

    c. gledanje uživo sesije za obučavanje na personalnom kompjuteru

    d. preuzimanje video fajlova

    e. gledanje najnovijih vesti

    f. korišćenje FTP-ja za preuzimanje veoma velikih fajlova

    g. korišćenje FTP-ja za preuzimanje izuzetno malih fajlova

12. Zašto se u TCP segmentima ne nalazi broj bajtova podataka koji segment sadrži? Kako prijemni TCP entitet može da zna koliko bajtova podataka treba da izvuče iz segmenta?

13. Pretpostavite sledeće:

    - TCP entiteti A i B imaju inicijalne brojeve sekvenci 400 i 900, respektivno.
    - Svaki segment sadrži 100 bajtova podataka i svaki ima inicijalni kredit od 200 bajtova.
    - Svaki entitet isporučuje segment čim ga primi i tako oslobađa prostor u baferu.
    - A može (to mu dopušta kontrola toka) da šalje TCP segmente u vremenskim intervalima T (počevši od T = 0); B može da šalje svoje segmente u intervalima 3T (počevši od 1.57).

Uz pretpostavku da je vreme prenosa između A i B zanemarljivo, skicirajte dijagram sličan onome na slici 11.30, koji prikazuje razmenu segmenata do trenutka 12T.

14. Ponovite sledeću vežbu uz pretpostavku da se vrednost kredita u A poveća na 300 nakon što primi drugi segment od B.

15. Razmotrite logiku kontrole toka za TCP, prikazanu u odeljku 11.4. Pošto entitet koristi polje Credit za utvrđivanje kada može da pošalje nove segmente, u čemu je svrha potvrde? Drugim redma, šta može da se desi ako eliminišemo polje Acknowledgment iz segmenta?

16. Pretpostavimo da ruta od čvora A do čvora Z ide preko rutera P, Q, R i X. Opišile pakete (tipove i relevantne sadržaje) koje A šalje i prima reagujud na komandu `traceroute`.

17. Ulogujte se daljinski na neki nalog i eksperimentišite sa Telnet komandama. Na primer, možete da uradite sledeće:

    a. Unesite veliki fajl, predite na Telnet, pa prekinite izlaz.

    b. Utvrdite odziv na komandu `send ayt`.

    c. Pokrenite dugački program, pređite na Telnet i prekinite proces.

    d. Predite na Telnet i unesite `help` da biste proučili ostale Telnet komande.

    Ako imate samo jedan nalog na serveru, možda ćete moći da se ulogujete na njega, a zatim da drugi put pristupite nalogu preko Telneta. Ovo je pomalo slično situaciji u kojoj sami sebe pozivate telefonom, ali funkcioniše i omogućava da upoznate Telnet (moraćete da utvrdite da li Vaš sajt dopušta višestruko logovanje na jedan nalog).

18. Povežite se na udaljeni sajt preko anonimnog FTP-a, prenesite fajl na svoj nalog i napišite kraći rezime sadržaja fajla.

19. Razmotrite sliku 11.12 i pretpostavite da svaka mreža predstavljena oblakom može da ima 50 hostova u multicast grupi. Ako host X pripada toj gmpi, koliko kopija svakog paketa može da "putuje" preko Interneta ako se koristi slanje do jedinstvenog odredišta? Ako se koristi multicasting, koliko kopija svakog paketa "putuje" preko Interneta?

20. Ako imate nalog na serveru, postavite ssh tako da, kada se ulogujete, server proverava Vašu autentičnost pomoću sistema javnog/privatnog ključa.

Reference

[BaOl] Barrett, D., and R. Silverman. *SSH: The Seaire Shell.* Sau Francisco: O'Reilh/, 2001.

[Br95] Bradner, B., and A. Mankin. "The Recommendation for the IP Next Generation Protocol". RFC 1752, Ianuary 1995.

[CaOl] Caslovv, A., and V. Pavlichenko. *Cisco Certificalion. Bridges, Routers and Sivitches for CClEs₁* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2001.

[Co99] Corner, D. E., and D. Stevens. *Internetivorking with TCP/IP. VoI. IL ANSI C Version: Design, Implementation, and Internals,* 3rd ed. Englevvood Cliffs, NJ: Prentice-Hall, 1999.

[CoOO] Corner, D. E. *lnternetWOrking with TCP/IP. VoI. 1. Principles, Protocoh, and Architecture,* 4th ed. Englewood Cliffs, NJ: Prentice-Hall, 2000.

[De93] Deering, S. "SlP: Simple Internet Protocol". *IEEE Netivork Magazine,* vol. 7, no. 3 (May/June 1993), 16—28.

[Ek02] Ekici, E., I. Akyildiz, and M. Bender. "A Mullicast Routing Algorithm for LEO Satellite IP Netvvorks". *IEEE/ACM Transactions on Nelivorking,* vol 10, no. 2 (April 2002), 183-192.

[Fo03] Forouzan, B. *TCP/IP Protocol Suite,* 2nd ed. New York: McGraw-Hill, 2003.

[Gr02] Gralla, P. *How lhe Inlernet Works,* 6th ed. Engkwood Cliffs, NJ: Prentice-Hall, 2002.

[HaOl] Halsall, F. *Multimedia Communicatiorts.* Reading, MA: Addison-Wesley, 2001.

[Hi96] llinden, R. "IP Next Generation Overvievv". *Communications ofthe ACM,* vol. 39, no. 6 (June 1996), 61-71.

[Ja88] Jacobson, V. "Congeslion Avoidance and Control". *Proceedings of SIGCOMM Symposium,* August 1988, 314-329.

[Ko98] Kosiur, D. IP Multicasting: The Complete Guide to Interactive Corporate Netvvorks. NewYork: Wiley, 1998.

[KuOlJ Kurose, 1., and K. Ross. *Computer Netivorking.* Reading, MA: Addison-WesIey, 2001. [MaOO] Mann, S., and E. Mitchell. *Linux Systems Security: The Administrator's Gnide to Open Sonrce Security Tools,* 2nd ed. Englewood Cliffs, NJ: Prentice-Hall, 2000.

[M199] Miller, M. *Troubleshooting TCP/IP,* 3rd ed. New York: M&T Books, 1999.

[MiOO] Miller, M. *Managing Internetivorks ivith SNMP,* 3rd ed. New York: Wiley, 2000.

[Mi02] Miller, M. *Voice over IP Technologies: Building the Converged Netivork,* 2nd ed. New York: Wiley, 2002.

[PeOOj Perlman, R., and C. Kaufman. "Key Exchange in IPSec: Analysis of IDE". *IEEE Intemel Compuling,* vol. 4, no. 6 (November/December 2000), 50-56

[Pi03] Pink, S. *High Speed Roulers and Fireivalls.* Reading, MA: Addison-Wesley, 2003.

[Ru89] Russel, D. *The Principles of Computer Networking.* Cambridge, England: Cambridge University Press, 1989.

[Sc02[Schedna, E., K. Green, and J. Carlson. *Internet Sile Security.* Reading, MA: Addison-Wesley, 2002.

[St96] Stallings, W. "IPv6: The New lnternet Protocol". *IEEE Communications Magazine,* vol. 34, no. 7 (1996), 96-108

[St99] Stallings, W. *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2,* 3rd ed. Reading, MA: Addison-Wesley, 1999.

[SuOO] Subramanian, M. *Nelivork Management Principles and Praclice.* Reading, MA: Addison-Wesley, 2000.

POGLAVLJE

12

Internet programiranje

Obično smo zatrpani tolikim količinama informacija i krećemo se toliko brzo da često sve gledamo površno i brzo prelazimo preko nekih "stvari" koje nam se saopštavaju. Ovo je suprotno otvorenosti i sposobnosti opažanja koje morate Aa imate da biste imali sklonosti ka poeziji. . . Poezija kao da postoji u univerzumu koji je paralelan današnjem životu u Americi.

—**Rita Dove,** američki pesnik

12.1 Uvod

Da li ste nekada koristili pretraživačku mašinu da biste pronašli nešto na Internetu? Šta je sa formularima za naručivanje nekih proizvoda? Možda ste koristili Instant Messenger za konverzaciju sa svojim prijateljima. Da li ste ikada zastali i zapitali se, ako već postoje ljudi koji pišu softver koji funkcioniše u Internet okruženju, da li to možete i Vi? Pošto ste sada razumeli mrežna okruženja, preostaje samo da primenite programerske veštine i otkrijete kako intereaguju sa mrežnim protokolima. Iako je možda čudno, mnogi tehnički detalji nisu preterano teški, jer možemo da pretpostavimo da mrežni protokoli implementiraju sve te "zastrašujuće" detalje. Ako pretpostavite da nešto većpostoji, sve što treba da uradite je da kreirate još jedan sloj na vrhu.

Namera nam je da u ovom poglavlju pokažemo neke načine na koje je to moguće izvesti - razvijamo neke aplikacije koje se oslanjaju na Intemet protokole. Napisaćemo neke "stvari" na strani klijenta, aneke na strani servera. O programiranju u Internet okruženju napisani su čitavi tomovi, a mi smo odlučili da predstavimo samo par različitih pristupa. Kao i u slučaju ostalih tema u ovoj knjizi, o svakom našem primeru mogu da se napišu celi udžbenici. Ovde nameravamo da predstavimo onoliko detalja koliko je potrebno da razumete funkcionisanje nekih osnovnih Internet programa. Svako može da iskoristi ovde predstavljene primere kao osnovu na kojoj će proširivati svoja znanja u raznim pravcima.

U odeljku 12.2 počinjemo sa primerom koji prikazuje kako klijent i server mogu da razmenjuju pakete radi preuzimanja fajla. Koristićemo UNIX soket interfejs za transportni sloj i opisaćemo neke osnovne funkcionalnosti.

Završićemo dizajniranjem koda koji, u stvari, preuzima fajl sa jednog hosta na drugi. Kada razumete koncept soketa, otvoriće Vam se "vrata" ka svim ostalim vrstama aplikacija, kao što su programi za razmenu poruka i pretraživanje klijent/server baze podataka.

U odeljku 12.3 skrećemo pažnju na Web okruženja. Obezbedićemo osnovne informacije o HTTP-u (Hypertext Transfer Protocol) i HTML (Hypertext Markup Language) jeziku i opisaćemo kako se kreiraju Web sajtovi koji mogu da izvršavaju različite funkcije. Ovaj odeljak je fokusiran na stranu klijenta i obezbeduje uvod u JavaScript, jezik koji može da "radi" sa Web formama radi izvršavanja različitih akcija. U odeljku 12.4 prelazimo na stranu servera i fokusiramo se na CGI (Common GateWay Interface) programiranje pomoću programskog jezika C. Reč je o načinu pisanja programa na udaljenom hostu koji reaguju na zahteve koje šalje klijent. Primera radi, ilustrovaćemo kako se može postaviti jednostavna pretraživačka mašina na Internetu.

U poslednjem odeljku opisani su pomalo ograničeni, ali potpuno funkcionalni sistem za naručivanje pica i nešto dmgačiji pristup programiranju na strani servera. Obuhvatićemo skript jezik* pod nazivom Perl i pokazati kako se piše Perl skript koji prima narudžbine, verifikuje brojeve telefona, izračunava cene narudžbina i vraća te informacije klijentu. Uzevši zajedno, ovi primeri obezbeduju dobai osnovu za učenje drugih programskih alatki za Intemet okruženje i za kreiranje sofisticiranijih primera.

12.2 Soket programiranje

Kada su prvi put kreirane mrežne aplikacije, mnoge od njih su pokretane pod operativnim sistemom UNIX, i to Berkeley UNIX-om. Da bi se olakšali slanje i prijem podataka preko mrežnih konekcija, dizajniran je niz TCP primitiva koje UNIX aplikacije koriste. Pomoću ovih primitiva UNIX aplikacije, koje se pokreću na različitim kompjuterima povezanim na Internet, mogu da komuniciraju, koristeći mehanizam poznat pod nazivom *soket.* Kako su se mreže razvijale i jezid i operativni sistemi razradivali, soket je napredovao tako da omogućava komunikaciju mreža sa različitim jezicima i operativnim sistemima. Osim programa zasnovanih na soketima na UNIX-u, ovakvi programi su postali uobičajeni i na personalnim kompjuterima i u Java i C++ okruženjima.

Primarni cilj u ovom odeljku je obezbediti minimalni, ali veoma funkcionalni klijent/server model koji se pokreće na Linuxu (varijanta UNIX-a). To znači da ćemo pokazati kako da razvijete dva različita programa koji će moći medusobno da komuniciraju. Jedina pretpostavka je da se pokreću na dva različita Linux sistema, koja imaju pristup Internetu. Konkretno, razvićemo dva programa koja implementiraju protokol za transfer fajlova. Jedan program će čitati tekst iz fajla, podeliti ga na pakete i poslati do drugog programa pomoću soket konekcije. Drugi program će prihvatiti pakete, jedan po jedan, i kreirati novi fajl na osnovu sadržaja paketa.

* *Sbiptjezik* je jezik dji se izvorni kod interpretira u vreme izvršenja. Ovo je supromo *kompajliranom jeziku,* koji se prevosi u mašinski kod, a zatim ga centraini procesor izvršava direktno.

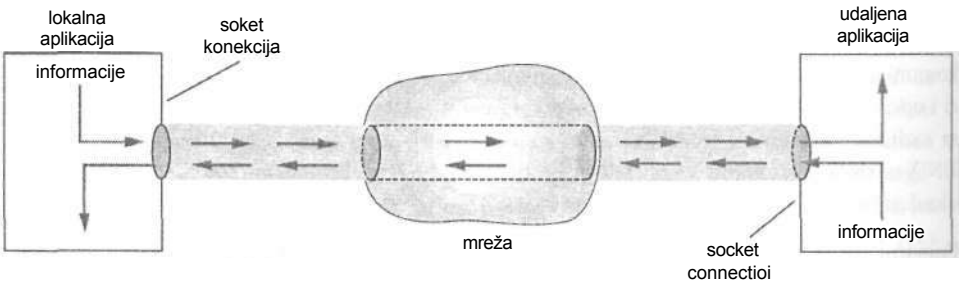
Osim toga, ovi programi mogu da se izvršavaju konkurentno i asinhrono na različitim kompjuterima. Na kraju odeljka napomenućemo moguća proširenja ovog projekta koja omogućavaju klijentu, ili serveru (ili i jednom i drugom) da budu napisani ili na Linux sistemu, ili u Javi. Osim toga, ukazujemo na kodni primer koji to obavlja.

Da bi ovaj odeljak ostao koncizan i fokusiran, obezbedujemo samo neophodne detalje soketa za ovu aplikaciju i nemamo nameru da ugradujemo mnošivo opcija u programe. Osim toga, nećemo pokušavati da navodimo iscrpljujuća objašnjenja svih soket komandi i svih okruženja u kojima mogu da se koriste. Na ovom nivou ti detalji nisu mnogo bitni, jer je naš cilj da obezbedimo funkcionalnost. Zato ćemo razviti radnu klijent/server aplikaciju koju možete da iskoristite kao polaznu osnovu.

Ako imate sistem pod Linuxom, trebalo bi da bez problema, ili uz neke manje modifikacije, pokrenete ovde predstavljene programe. Pretpostavljamo da imate osnovna znanja o Linuxu i programskom jeziku C. Postoje brojni načini da proširite ono što ćemo mi uraditi, a svi oni koji SU zainteresovani za detaljnija objašnjenja soketa, njihove komande i različite opcije trebalo bi da pogledaju reference [Co99] i [CoOO].

Soketi

Soket se konstruiše kao podrška mrežnom ulazu/izlazu (1/0). Aplikacija kreira soket kada Je potrebna komunikacija sa mrežom. Nakon toga, uspostavlja konekciju* sa udaljenom aplikacijom preko soketa i komunicira sa njom čitajući podatke sa soketa i upisujući podatke na njega.



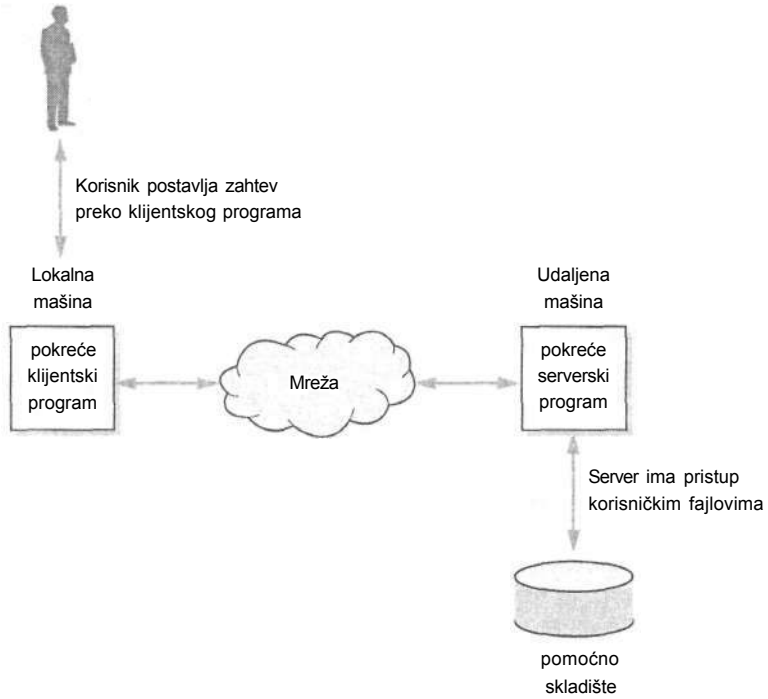
SLIKA 12.1 *Soket konekcija sa mrežom*

* Pretpostavljamo da soketi koje kreiramo koriste TCP i da su orijentisani konekciji. Postoje i sokcti koji funkcionišu bez uspostavljanja veze i oslanjaju se na UDP. Zbogograničenogprostora, fokusiraćemo sesamo na konekciji orijenlisanesokete, a za više informacija o UDP soketima čitaoce upućujemo na [Ku03].

Na slici 12.1 ilustrovana je ova ideja. Lokalni program može da usmeri informacije kroz soket na mrežu. Kada se informacije nadu na mreži, mrežni protokoli ih vode kroz mrežu, gde pristupaju udaljenom programu. Slično tome, udaljeni program može da poslavi informacije u svoj soket. Odatle, one idu preko mreže i završavaju nazad u lokalnom programu. Definisanjem pravila preko kojih lokalni i udaljeni programi razmenjuju informacije u stvari možemo da definišemo sopstvene protokole.

Model klijent/server

Pre nego što pokušamo da definišemo bilo koju vrstu protokola, počinjemo opisivanjem osnovnog modela na kome se zasniva većina mrežnih protokola - **klijent/server modela.** Suština klijent/server modela (slika 12.2) uključuje dva različita programa, koji se obično pokreću na različitim mašinama smeštenim na različitirn lokacijama. Mašine imaju mrežne konekcije. Za naše razmalranje nije bitno da li je mreža lokalna, ili "pokriva" širu geografsku oblast, ili nešto izmedu te dve varijanle. Osnovni koncept je isti u svakom slučaju.



SLIKA 12.2 *Klijent/server model*

U suštini, i klijent i server imaju odredene uloge. **Server** mora da obezbedi servise i da reaguje na zahteve koji stižu od klijentskog programa. Tipičan primer je obezbedivanje pristupa fajlovima koji su smešteni na udaljenoj mašini. Korisnik pokreće **klijentski** program na lokalnoj mašini i preko njega postavlja različite zahteve. Sledeći naš primer, korisnik može da zahteva pristup jednom, ili više fajlova smeštenih na drugoj (udaljenoj) mašini. Tako tipična razmena može da izgleda slično sledećoj:

1. Korisnik zahteva fajl.
2. Klijent šalje zahtev do servera u ime korisnika.
3. Server prima zahtev od klijenta i analizira ga.
4. Server kopira fajl iz svog pomoćnog skladišta.
5. Server prenosi sadržaj fajla nazad do klijenta.
6. Klijent dobija sadržaj fajla od servera i čini ga dostupnim korisniku.

U opštem slučaju, jedan server može da obezbedi servis za više klijenata. Zato se postavlja pitanje kako server može efikasno da upravlja sa više klijentskih zahteva. O tome će biti više reči nešto asnije; za sada treba da vidimo kako server rukuje zahtevima od jednog klijenta.

Strukture podataka koje soketi koriste

Kada na kraju budemo razmatrali neke od poziva soketa, moraćemo da opišemo parametre koji im se prenose. Neki od tih parametara imaju tipove dizajnirane specifično za mrežne komunikacije. Zato započinjemo predstavljanjem različitih struktura podataka koje soketi pozivaju. Svaka od tih struktura je smeštena u header fajlu Linuxa. Kasnije ćemo naznačiti koji su header fajlovi neophodni.

Prva struktura koja je neophodna je

```
struct sockaddr in {
        u_short sin family;
        u_short sin port;
        struct in_addr sin_addr;
        char sin_zero[8]
}
```

gdeje

```
struct in_addr {
    u_int s_addr
}
```

U suštini, ovo je 16-bajtna struktura koja sadrži adresu soketa (kombinacija IP adrese i broja porta; pretpostavljamo da se koristi TCP/IP mreža). U tabeli 12.1 dat je pregled prisutnih polja. Sledeća struktura je

```
struct hostent {
        char *h_name;
        char **h_alias;
        int haddrtype;
        int h_length;
        char **h_addr_list
}
```

Tabela 12.1: Polja u strukturi sockaddr_in

Polje	Značenje
sin_family	Ovaj 16-bitni ceo broj definiše protokole koji će biti korišćeni za implementaciju soket konekcije. U opštem slučaju, soketi mogu da se koriste i sa drugim mrežama osim TCP/IP mreža, ali to ovde nećemo analizirati.
sin_port	Ovo 16-bitno polje definiše broj porta koji identifikuje aplikaciju (sećate se definicije broja porta iz razmatranja o TCP zaglavlju).
s_addr	32-bitna Internet adresa (pretpostavlja da sin_family naznačava TCP/IP protokol).
sin_zero	Ne koristi se. Sadrži sve nule.

Tabela 12.2: Polja u strukturi hostent

Polje	Značenie
h_name	Niz karaktera okončan Null karakterom za tekstualnu adresu host kompjutera
h_alias	Lista alternativnih naziva hosta (ovde nije značajno)
h_addrtype	Tip adrese (u našem primeru definiše Internet adresu)
h_length	Dužina adrese
h_addr_list	Lista dodatnih adresa hosta (ovde nije značajno)

U tabeli 12.2 definisana su polja strukture hostent.

Komande soketa

Postoje različite komande "vezane" za sokete. U tabeli 12.3 date su komande koje su nam bile potrebne za naš primer, zajedno sa kraćim opisom svake od njih.*

Primer klijent/server modela

Sada smo spremni da napišemo klijent/server protokol koji se koristi za transfer fajla.' Prvi korak je opisivanje poziva "vezanih" za sokete u klijentskom i serverskom programu. Ovo ce pomoći da razumete njihovu svrhu, bez potrebe da se previše bavite detaljima. Na slici 12.3 prikazani su i klijent i server. Imajte na umu da se oba izvršavaju konkurentno na različitim mašinama.

*Nećemo navoditi kompletne opise, zato što je naš osnovni cilj da obezbedimo samo ono što je neophodno za našu aplikaciju.

t I klijent i server su testirani i pokretani na Red Hat LinuXU, verzija 7.3.

Tabela 12.3: Pregled potrebnih komandi soketa

Komanda	Značenje
`socket (int domain, int mode, int protocol)`	Kreira soket. Prvi parametar defmise domen koji se konsti. Na pnmer, simbolička konstanta AF_INET definiše Internet domen.AFjJNix definiše UNIX domen. Drugi parameter definiše mod komunikacije. Na primer, simbolička konstanta SOC_STREAM ukazuje da ćemo koristiti konekciji orijentisane nizove bajtova. SOCDGRAM označava mod sa datagramima, bez uspostavljanja konekcije. Treći parametar definiše protokol, ili, ako je 0, omogućava sistemu da se osloni na podrazumevani protokol (u ovom slučaju to je TCP/IP). Ako Je ova komanda uspešno izvršena, vraća ceo broj (deskriptor) soketa koji će se koristiti u narednim komandama. U ovoj tački soket je nešto više od indeksa u tabeli deskriptora kernela. Za uspostavljanje stvarne konekcije potrebno je nešto više.
`gethostbyname( char *hostname)`	Vraća pokazivač na hostent strukturu u kojoj se nalaze relevantne informacije o hostu naznačenom tekstualnom adresom u hostname.
`gethostname( char "hostname, int length)`	Postavlja string za tekstualnu adresu tekućeg hosta u promenljivu hostname. Drugi parametar predstavlja broj raspoloživih bajtova u hostname.
`connect (int s, struct sockaddr_in *sa, int size)`	Zahteva konekciju sa udaljenim soketom. Poziv mora da naznači identifikator soketa (s) i strukturu u kojoj se nalazi adresa udaljenog soketa (*sa). Ovaj poziv je neophodan kada se zahteva konekciji orijentisani servis. Treći parametar definiše veličinu *sa strukture. Kao funkcija, vraća status zahteva.
`bind[1] (int s, struct sockaddr *sa, socklen_t length)`	Dodeljuje adresu i broj porta (unutar *sa) za soket S. Server izdaje ovu komandu kako bi se učinio dostupnim udaljenim klijentima koji se povezuju na ovaj soket.
`listenfint s, int n)`	Koristi Je konekciji orijentisani server koji reaguje na zahteve udaljenih klijenata. Ukazuje da je server spreman za zahteve za konekciju i da ih "osluškuje" preko soketa S. Nakon toga, zahteve postavlja u red čekanja za dalje procesiranje. Drugi parametar (n) definiše broj zahteva koji se mogu postaviti u red čekanja. Ako zahtev stigne dok je red čekanja pun, server ga odbacuje.
`accept(int s, struct sockaddr *sa, socklen_t length)`	Omogućava serveru da prihvati zahtev za konekciju preko soketa S. Kada se prihvati, IP adresa klijenta koji je postavio zahtev smešta se u *sa strukturu. Osim toga, ova funkcija, u stvari, kreira i vraća soket sa svim istim svojstvima kao i S. Razlog za to je činjenica da, nakon što server prihvati poziv, obično se račva na "decu" potprocese, koji koriste novi soket za komunikaciju sa klijentom. U meduvremenu, proces "roditelj" nastavlja da "osluškuje" nove zahteve preko starog soketa.
`send( int s, char *buffer, int length, int flags)`	Šalje podatke preko soketa S. Lokacija i dužina podataka odgovaraju parametrima buffer i length, respektivno. Odredište nije naznačeno, jer prethodne komande connect, ili accept definišu ko se nalazi na drugoj strani soketa. Parametar f lags ovde nije značajan.
`recv(int s, char *buffer, int length, int flags)`	Prima podatke od soketa i smešta ih u naznačeni baf er. Parametar length definiše dužinu bafera. Kao i kod komande send, parametar f lags nije značajan za naše razmatranje.
`close (int s)`	Zatvara naznačeni soket.

[1] Obratite pažnju na drugi parametar. Prethodno opisani struct sockaddr_in predstavlja generalizovaniju slrukturu (struct sockaddt-) i koristi se sa Internet aplikacijama. Medutim, neke verzije Linuxa zahtevaju da se adresa generalizovanije strukture koristi sa odredenim komandama soketa. U kodnom primcru koji ćemo kasnije opisati izdavaćemo struct sockaddr_in adresu kao struct sockaddr adresu.

Klijent	Server
socket (kreira soket)	**socket** (kreira soket)
gethostbyname (mapira naziv udaljenog hosta na IP adresu)	**gethostname** (uzima naziv lokalnog hosta)
	gethostbyname (mapira naziv na IP adresu)
connect (izdaje zahtev za konekciju do naznačenog servera na udaljenom hostu)	
	bind (definiše IP adresu i broj porta)
	listen (postavlja soket u pasivni mod; server je spreman za prijem zahteva)
	accept (prihvata zahtev za konekciju)
razmena informacija pomoću komandi send i recv	
dose (okončavanje konekcije)	**razmena** informacija pomoću komandi send i **recv**
	close (okončavanje konekcije)

SLIKA 12.3 Opsfi' *pregled klijenata i servera koji koriste komande "vezane" za sokete*

Jedina pretpostavka koju smo napravili je da oba kompjutera imaju Internet konekcije.

Na slici je pokazano da je prvo što moraju da urade i klijent i server kreiranje soketa pomoću komande socket. Nakon toga, klijent poziva komandu gethostbyname kako bi utvrdio IP adresu dodeljenu tekstualnoj adresi udaljenog hosta na kome je pokrenut server. Kada je IP adresa poznata, klijent se povezuje na server. U ovoj tački klijent može da razmenjuje poruke sa serverom pomoću komandi send i recv. Potrebno je definisati pravila za razmenu i za utvrdivanje kada je ona završena. Poslednje što klijent radi je zatvaranje soketa i završavanje.

Server, nakon kreiranja soketa, mora da pribavi naziv hosta na kome se pokreće (poziv komande gethostname). Nakon toga, koristi naziv hosta i poziva komandu gethostbyname da bi pribavio relevantne informacije (tj. svoju IP adresu) i smešta ih u strukturu hostent. Zatim, poziva komandu bind za pridruživanje IP adrese i broja porta sa soketom i počinje "osluškivanje" svih dolazećih poziva. Kada poziv stigne, server ga prihvata i uspostavljena je soket konektija sa klijentom. U ovoj tački server poštuje isti protokol koji klijent koristi za razmenu niza poruka. Kada se razmena obavi, zatvara soket i završava.

Svakako postoji još detalja koji bi se mogli pomenuti, ali ovakav način predstavljanja pomaže razumevanje radnog okvira u kome ih opisujemo. Zato bi, pre nego što nastavimo, trebalo da razumete bar razloge za pozive soketa i osnovnu organizaciju klijentskih i serverskih programa.

Sledeći korak je obezbedivanje detalja za radne klijentske i serverske programe. Slike 12.4 i 12.5 sadrže izvorne kodove tih programa. Možete ih iskopirati i pokrenuti u skladu sa sledećim pravilima (uz pretpostavku da se pokreću pod Linuxom):

- Kompajlirate server i pokrenite ga prvog. Server mora da se pokrene tako da klijent ima koga da pozove.
- Koristili smo port pod brojem 6250, tako da ne dode do konflikta sa bilo kojim drugim brojevima portova. Slobodno možete da koristite i druge raspoložive brojeve portova, ali možda ne bi bilo loše da proverite sa lokalnim administratorom da li fireWall blokira neke komunikacije na osnovu broja porta.
- Kompajlirajte klijenta i pretpostavite da izvršni fajl nosi naziv myftp. Pokrenite klijenta unošenjem komande myftp tekstualna-adresa-hosta. Na primer, ako je tekstualna adresa host kompjutera hercules.uWgb.edu, pokretanje vršite unošenjem myftp hercules.uWgb.edu.
- Server će preneti unapred odredeni fajl do klijenta. Odnosno, server bira naziv fajla.
- Klijent će pitati pod kojim nazivom želite da snimite fajl kada se pribavi od servera.
- Server čita tekstualni fajl, deli ga na pakete i šalje ih do klijenta u grupama od po pet. Nakon svakog petog paketa, čeka na potvrdu od klijenta.
- Klijent reaguje na odgovarajući način - prihvatanjem paketa i izvlačenjem podataka iz njih. Podatke iz svakog paketa smešta u tekstualni fajl. Nakon što stigne poslednji paket, transfer fajla je kompletiran. Klijent šalje potvrdu posle svakog petog primljenog paketa.
- Ne postoji detekcija grešaka, a kontrola toka je ograničena na ono što smo upravo opisali.

Sada prelazimo na opise ldijenta i servera. Većim delom ćemo se ograničavati na logiku "vezanu" za sokete, a Vama prepuštamo da proučite detalje "vezane" za programski jezik C. Komentari u izvornom kodu treba da služe kao pomoć i zato navodimo neke opšte opise bitne za zadatke koji se tiču prvenstveno C-a.

Izvorni kod klijenta Početak klijentskog programa (slika 12.4) sadrži header fajlove* koje morate da uključite. U njima se nalaze definicije za pozive upućene soketima i definicije opisanih struktura podataka. Definisali smo prilično jednostavan protokol za prenos paketa. Prvo što sadrži je polje servicetype, koje ukazuje na različite tipove paketa.

* Ovaj program je testiran pod Red Hat LinuXom. Neki header fajlovi mogu da se razlikuju u zavisnosti od toga koja se varijanta UNIX-a koristi.

```
/*              KLIJENT             */
/*              KLIJENT             */
/*              KLIJENT             */

/* Ovaj program je dizajniran kao klijent koji ce pozivati server pokrenut na
   drugoj masini. Pretpostavlja se da se koristi TCP/IP protokol i da su
        konekcije uspostavljene preko Interneta */

#define PACKETSIZE 20
tfinclude <sys/types.h>
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

/define PORTNUM 6250

typedef enum {data, ackl servicetype;

/* jedinica prenosa za protokol ovog programa */
typedef struct
{
 servicetype service;          /* tip paketa */
 int sequence;                 /* broj sekvence paketa */
 int datasize;                 /* kolicina podataka (u bajtovima) u paketu  */
 int last:                     /* indikator poslednjeg paketa */
 char data[PACKETSIZE];        /* sadrzi podatke */
 int checksum;                 /* sluzi za detekciju gresaka */
} PACKET;


/* otvara fajl i vraca njegov identifikator */

FILE * openfile( )

 char * filename;
 FILE * fid;

 printff'This program WIII copy a file from a remote server\n");
 printf("Enter the name under Which the file should be saved>");
 filename = (ohar *) malloc(30);
 scanf("%s", filename);
 if ((fid = fopen(filename. "W")) == NULL)
 {
  printf("error opening file\n");
  exit (1);
 }
 free(filename);
 return fid;
}
/ **********************

/* kreiranje soketa */

int opensocket( )
{
 int s;
 if ((S = socket (AF_INET, SOCK_STREAM, 0)) < 0)
 {

  perror ("socket error");
  exit (1)
```

```
 }
 return s;
}

/* dobijanje informacija od udaljenog hosta i povezivanje na udaljeni server */

void makeconnection(int argc, char *argv[ J1 int s)
{
 struct hostnet   *ph;          /*sadrzi naziv udaljenog hosta i adresne informacije*/
 struct sockaddr in sa;         /*sadrzi IP adresu i port protokola */

memset(&sa,0,sizeof(sa));       /*nuliranje strukture *I
if (argc !=2)                   /*Provera da li komanda za pokretanje na klijentu
                                  sadrzi tekstualnu adresu udaljenog hosta */
{
 printf ("Error it command line\n");
 exit (1);
}
if ((ph = gethostbyname (argv[l]))== NULL     /* uzimanje relevantnih informacija
                                                 od udaljenog hosta */
{
 printf("error in gethostbyname\n');
 exit(1);
}
                              /* Smestanje IP adrese udaljenog hosta, broja port
                                 na serveru i tipa protokola u strukturu */
 memcpy((char*) &sa.sin_addr, ph- > h_addr, ph -> h_length);
 sa.sin_port = htons ((u_short) PORTNUM); /*definise broj porta na udaljenom serveru
 sa.sin_family = ph- >h_addrtype;
 if (connect (s, &sa, sizeof (sa)) < 0) /* povezivanje na udaljeni server */
 {
  perror ("connect error");
  exit(1);
 }
}

/* uzimanje fajla od udaljenog servera u paketlma fiksne velicine */
I*****************************************************************************/
void getfile(FILE * fid, int s)
{
 PACKET * packet;              /* paket protokola */
 int i;
 packet = (PACKET *) malloc (sizeof(*packet));
 do

  if (recv(s, packet, sizeof(*packet), 0) <= 0) /*uzimanje paketa sa udaljenog
                                                  servera */
{
  printf("error reading\n");
  exit(1);
}
  printf("Received packet %4d: %s\n", packet->sequence, packet->data);
```

```
   for (i=0; i < packet->datasize; i++)     /*Smestanje sadrzaja paketa u
                                            tekstualni fajl */
     putc (packet->data[i], fid);
   if (packet->sequence % 5 == 4)      /*Ako je ovo bio peti paket, salje se potvrda*/
   {
   printf('acknoWledging 5th packet-Press enter to continuen");
   getchar();
   packet->service = ack;
   if (send(s, packet, sizeof(*packet), 0) <= 0)    /* Slanje potvrde */
    {
     printf("ERROR in send\n');
     exit(1);
    }
   }
   While (!packet-> last);            /*Nastavlja se prethodno predstavljeni proces
                                       dok se ne primi poslednji paket */
   fclose(fid);
  }
 void main (int argc,char *argv[])
 {
  int s;
  FILE * fid;
  fid = openfile();
  s = opensocket();
  makeconnection(argc, argv, s);
  getfile(fid, s);
  close(s);
 }
```

SLIKA 12.4 *Klijentski program*

```
/*              SERVER          */
/*              SERVER          */
/*              SERVER          */

/*0vaj program je dizajniran tako da se ponasa kao server koji ce prihvatati pozive
od klijenta napisane na drugoj masini. Pretpostavlja se da se koristi TCP/IP
protokol i da su konekcije izvedene preko Interneta */

#define PACKETSI2E 20
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>
#include <stdio.h>
#include <sys/param.h>

#define PORTNUM 6250
typedef enum {data, ack} servicetype;
/* jedinica prenosa za protokol ovog programa */
typedef struct
{
  servicetype service;           /* tip paketa */
  int seguence;                  /* broj sekvence paketa */
```

```
  int datasize;                 /* kolicina podataka (u bajtoviraa) u paketu */
  int last;                     /* indikator poslednjeg paketa */
  char data[PACKETSIZE];        /* sadrzi podatke */
  int checksum;                 /* za detekciju gresaka */
} PACKET;

/*uzimanje informacija o hostu za eventualnu soket konekciju */

struct sookaddr_in gethoststuff()
{
  struct sockaddrrin sa;        /* sadrzi IP adresu i port protokola */
  struct hostent * ph;          /* sadrzi naziv hosta i informacije o adresi */
  char myname[MAXHOSTNAMELEN+l]; /* naziv hosta */
  gethostname(myname, MAXHOSTNAMELEN); /* Uzimanje naziva host na kome je server
                                          pokrenut */
  printf("host name is %s \n", myname);
  if ((ph = gethostbyname(myname)) == NULL)/*Uzimanje relevantnih informacija o hostu*/
  {
  printf("gethostbyname failed\n");
  exit(1);
  }
  Memset(&sa, 0, sizeof(struct sookaddr_in));
              /*Postavljanje tipa "familije" protokola i broja porta u strukturu*/
  sa.sin_family = ph->n_addrtype;
  sa.sin_port = htons(PORTNUM);
  return sa;
}

/*Crtvaranje fajla i vracanje njegovog identifikatora */
|*****************************************************/
FiLE * openfile()
{
  FILE * fid;
  char * filename = "test.dat";

  if ((fid=fopen(filename,"r")) == NULL)
  {
   printf("error opening file\n");
   exit(1);
  }
  return fid;
}

/* Kreiranje soketa */

int opensocket()
{
  int s;

  if ((s = socket (AF_INET, SOCK_STREAM, 0)) < 0) /* Kreiranje soketa */
  {
    perror("socet error");
    exit(1);
  {
```

```
  return s;
}
/                ********************************************j
/* Povezivanje i "osluskivanje' soket konekcije */
/ ********************************************/
void bindnlisten(int s, struct sockaddr_in sa)
{
  if (bind (s. (struct sockaddr*) &sa, sizeof (sa)) < 0) /* Dodela IP adrese i
                                                            broja porta soketu s */
  {
    perror("bind error'");
    exit (1);
  }
  listen (S, 5);                  /* "Osluskivanje* dolazecih poziva od klijenta */
  }
t*****************************j
/* Prihvatanje soket konekcije */

int acceptconn(int s)
{
  int sd;
  struct sockaddr_in sa;
  unsigned int sasize;

  sasize = sizeof(sa);
  if ((sd = accept (s, (struct sockaddr*) &sa, Ssasize)) < 0) /* Prihvatanje
                                                                 konekcije */
  {
    perror("accept error");
    exit(1);
  }
  printf ("Connection accepted from: %u\n", sa.sin__addr.s_addr);
  return sd:
}

/* Slanje sadrzaja fajla pomocu paketa fiksne velicine */
/* Cekanje na potvrdu nakon svakog petog paketa        */
I*****************************************************/
void sendfile(FILE * fid, int sd)
{
  PACKET packet; /* paket protokola */
  int i,
  count = P; /* brojac paketa */
  char c;
  packet = (PACKET *) malloc (sizeof *packet);
  packet - >last=0;
  c=getc(fid);
  do
  {
    for (i=0; i<PACKETSIZE && c != EOF; i++, c=getc(fid)) /* Postavljanje podataka
                                                             iz fajla u paket */
      packet->data[i] = c;
    packet->datasize = i;
    if (c==EOF)
```

```
    packet-> last=1;
    packet->sequence = count++;
    packet->service = data:
    printf("Sending packet %4d: %s\n", packet->sequence, packet->data);
    if (send(sd, packet, sizeof (*packet), 0 <= 0)  /* Slanje paketa do klijenta */
   {
     printf("error in Writing to socket\n");
     exit(1);
   }
   if (count % 5 = 0)                    /* Nakon petog paketa ceka se na potvrdu */
   {
   if (recv(sd, packet, sizeof *packet, 0) < 0)
   {
     printf("socket read failed \n");
     exit(1);
   }
   printf("Receiving acknoWledgment \n");
  }
 } While (c I=EOF);                    /* Zaustavljanje kada se dodje do kraja fajla */
 printf("file transfer done\n");
 free(packet);
}
int main (int argc, char *argv[])
  {
  int s;                               /* identifikuje soket */
  int sd;                              /* identifikuje konekciju sa soketora */
  struct sockaddrin sa;                /* sadrzi IP adresu i port protokola */
  int i;                               /* prlvremena promenljiva */
  FILE * fid;                          /* identifikator fajla */
  sa = gethoststuff();
  s = opensocket();
  bindnlisten(s, sa);
  While (1)       /* Ovaj test server ponavlja za prihvatanje poziva od klijenata */
  {
    sd=acceptconn(s);    /*poziv funkcije za prihvatanje poziva. Novi soket je sd */
    if (fork()==0)       /*Kreiranje "dete' procesa za obavljanje transfera fajla */
    {
    close(s);            /*Zatvaranje soketa s. "Dete* procesu nije potreban, jer on
                           koristi soket sd */
    printf("beginning file transferAn");
    fid=openfile();
    sendfile(fid, sd);
    fcloseffid);
    close(sd);
    exit (1):
  }
  else
    close(sd);            /* Zatvaranje soketa sd, "Roditelju" nije potreban. */
    }
  printf("press enter to quit\n");
  getchar();
  close(s);
}
```

SLIKA 12.5 *Serverski program*

U ovora primeru postoje samo dva tipa paketa: Data i Ack (za potvrdu). Takođe, sadrži broj sekvence, polje koje ukazuje na broj bajtova podataka u paketu, polje koje ukazuje na poslednji paket u nizu i polje Checksum (koje ovde nećemo koristiti). Jednostavnosti radi, pretpostavili smo da su ova polja tipa `int`. Konačno, preostalo polje je niz sa kapacitetom za smeštanje 20 bajtova; ono predstavlja podatke u paketu.

Letimični pogled na klijenta pokazuje da postoje četiri C funkcije: `openfile`, opensocket, makeconnection i `getfile`. Funkcija `openfile` traži od korisnika da unese naziv pod kojim će klijent smestiti preneti fajl. Kada korisnik unese naziv, funkcija otvara fajl u modu za upis. Ako sve prode kako treba, funkcija vraća identifikator fajla. Funkcija opensocket kreira soket. Sadrži naredbu za izlazak iz programa ako dode do greške u pozivn soketa. Ako je nspešno izvršena, funkcija vraća identifikator soketa.

Funkcija makeconnection uzima informacije od udaljenog hosta i povezuje se na server koji je pokrenut na tom hostu sa naznačenim brojem porta. Pretpostavljamo da korisnik unosi tekstualnu adresu udaljenog hosta kada unosi naziv izvršnog klijenta (na primer, myftp hercules.uWgb.edu). Kao takav, tekstualni string `hercules. uWgb. edu` je smešten u parametru glavne funkcije argv [1], a u drugom parametru glavne funkcije argc nalazi se vrednost 2. Funkcija poziva host po nazivu kako bi se relevantne informacije o udaljenom hostu postavile u strukturu `hostent`, koja se locira pomoću pokazivačke promenljive ph. Nakon toga, prenosi druge relevantne informacije, kao što je serverov broj porta, u strukturu adrese soketa, naznačenu promenljivom sa*. Kada promenljiva bude sadržala neophodne informacije, funkcija poziva funkciju connect, koja šalje zahtev za konekciju do servera.

Poslednja funkcija je najsloženija, jer sadrži pravila koja odreduju transfer fajla. Ipak je jednostavnija od mnogih drugih protokola koje smo opisivali. Kao što smo ranije pomenuli, protokol zahteva od klijenta da primi pet paketa, izvuče njihov sadržaj i upiše ga u fajl. Nakon što primi peti paket, klijent kreira paket tipa Ack i šalje ga nazad do servera. Nastavlja tako sve dok ne primi paket kod koga je `packet->last` jednako 1. Funkcija ima nekoliko `printf` i `getchar` komandi koje nemaju nikakvu drugu svrhu osim da osobi koja pokreće program pokažu šta se dešava. Kada stigne poslednji paket, klijent zatvara fajl. Zatim se vraća u glavni program, gde zatvara soket i završava program. Fajl je prenet.

Izvorni kod servera Serverski program (slika 12.5) je malo složeniji, jer treba više toga da uradi. Kao i klijent, mora da uključi neophodne header fajlove i da definiše kompatibilnu strukturu paketa. Sadrži šest funkcija: `gethoststuff`, `openfile`, opensocket, `bindnlisten`, `acceptconn` i `sendfile`. Njegov glavni deo je, takode, nešto složeniji i zato polazimo odatle. Pošto ovaj server služi iskljudvo za testiranje, dizajnirali smo ga tako da prihvati tačno tri poziva i da svaki put uradi isto. Ovako može da se testira njegova sposobnost da reaguje na više konkurentnih klijenata.

* Kod sadrži rcferencu na funkciju btons (skraćeno od "host to netWork"). Kada se radi sa razlidtim mašinama, moguće je da postoji nekompatibilnost u načinu smeštanja celih brojeva. Na nekim mašinama najznačajniji bitovi se smeštaju u bajtovima sa većim adresama (arhitektura "little endian"), dok se kod drugih najznačajniji bitovi smeštaju u bajtovima sa manjom adresom (arhitcklura "big cndian"). Operator htons osigurava ispravno interpetiranje podataka definisanih u programu.

Uvek kada prihvati zahtev za konekcijom vraća novi soket (sd), koji ima ista svojstva kao i soket s. Nakon toga, server prelazi na novi "dete" proces koji upravlja detaljima transfera fajla, koristeći soket sd. Veoma je važno da se razume Linuxova komanda fork da biste videli kako ovo funkcioniše; trebalo bi da pogledate neku literaturu za UNIX, ili Linux ako ne znate za ovu komandu.

Komanda fork kreira zaseban "dete" proces koji izvršava kod odmah nakon linije "if (fork ( ) ==0)". Ova sekcija koda zatvara soket s (zato što ovaj soket koristi "roditeljski" proces i "detetu" nije potreban), poziva funkcije za otvaranje fajla i vraća njegov identiflkator; kada završi transfer fajla, završava se. Zapamtite: završava se "dete" proces, a "roditeljski" proces je i dalje pokrenut i može da prihvata nove pozive. U ovom slučaju "roditelj" izvršava liniju nakon else, koja zatvara soket vraćen pozivom accept (promenljiva sd). Pošto je ovo soket koji "dete" proces koristi, "rodilelju" nije potreban.

Prethodno pomenute funkcije izvršavaju zadatke u skladu sa svojim nazivima. Prva funkcija gethoststuff uzima informacije koje su relevantne za hosta na kome je server pokrenut (zbog prenosivosti, kod servera ne treba da sadrži bilo kakve pretpostavke o hostu na kome se pokreće). Takođe, smešta neophodne informacije u strukturu adrese soketa s i vraća kopiju. Funkcija openfile otvara naznačeni tekstualni fajl (pretpostavljamo da je naziv fajla test .dat) za pristup radi čitanja i vraća identifikator fajla. Ovo je, kao što verovatno pogadate, fajl koji će server preneti. Funkcija opensocket kreira soket i vraća njegov identiflkator. Ima komandu za izlazak ako zbog nečega soket ne može da se kreira. Funkcija bindnlisten soketu dodeljuje adresu i broj porta i postavlja server u mod osluškivanja.

U nekoj tački stiže zahtev za konekciju (kao rezultat klijentovog poziva za povezivanje). Tada server prihvata konekciju. Poziv accept kreira novi soket i vraća njegov identifikator, koji se dodeljuje promenljivoj sd. Kao što je prethodno pomenuto, ovaj soket ima ista svojstva kao i soket s. Kada se kontrola vrati na glavni program, server se račva, kreirajući "dete" proces koji koristi soket sd i zatvara s. "Roditelj" zatvara sd i vraća se na početak petlje kako bi prihvatio sledeći poziv kada stigne preko s.

Konačno, funkcija sendfile u jednom trenutku čita 20 karaktera iz tekstualnog fajla i postavlja ih u pakete. Salje pet paketa odjednom, pa čeka na potvrdu od klijenta. Kao i kod klijenta, postoje komande printf koje omogućavaju da vidite šta se dešava dok se program izvršava. Konačno, stiže se do kraja fajla i funkcija prestaje da kreira pakete. U toj tački transfer fajla je kompletiran.

Soketi su veoma moćan i fleksibilan interfejs ka mreži. Iako su tradicionalno kreirani za UNIX okruženja i obično se koriste u programima pisanim u programskom jeziku C, više nisu neophodni. Na primer, programski jezik Java podržava sokete i može da se implementira u WindoWs okruženjima. Naravno, mnoge ideje su zadržane, ali se detalji razlikuju, zbog toga što je Java objektno-orijentisana.

Da bi bili ilustrovani fleksibilnost, proširivost, i platforma i jezik nezavisni od soket konekcija, u referenci [Sh02] projekat iz ovog odeljka je proširen, tako da uključuje Java klijent i server. Isti Java klijent može da preuzima fajl sa Java servera, ili C servera koji je ovde opisan.

Slično tome, isti Java server može da reaguje na zahteve za preuzimanje sa Java klijenta, ili ovde opisanog C klijenta. Specifično, ovaj proširenI projekat pokazuje da klijent i server:

- mogu da se napišu na istom, ili različitim jezicima (C, ili Java)
- mogu da se napišu na istoj, ili razlidtim platformama (WindoWs, ili Linux)
- mogu, ali ne moraju, da koriste različite jezičke paradigme (objektno-orijentisano, ili struktumo programiranje)
- mogu da se pišu za "big endian", ili "little endian" okruženja
- mogu da za reprezentaciju podataka koriste bajtove, ili Unicode

Kompletna kopija Java i C koda za klijent i server može da se preuzime sa adrese WWW.uWgb.edu/shayW/courses/files358.htm.

12.3 World WideWeb

Skoro je sasvim sigurno da je najznačajnije ostvarenje u poslednjoj deceniji **World Wide Web (WWW).** Protokoli kao što su Telnet, FTP i email ujedinili su Internet zajednicu i učinili dostupnom ogromnu količinu informacija. Ali, razvoj Weba je, izmedu ostalog, učinio da informacije postanu dostupnije. Bilo za potrebe ozbiljnih istraživanja, ili, jednostavno, radi zabave i igara, milioni ljudi su otkrili koliko čega Web ima da ponudi. Ipak, sve to su pratile i frustracije zbog kašnjenja, jer se broj Web sajtova povećavao eksponencijalno. Ovaj rast je prouzrokovao ne samo ogromnu količinu saobraćaja preko Interneta, već i sve teže pronalaženje korisnih informacija u tolikoj količini široko dostupnih informacija.

Kao i u slučaju ranije obradenih tema, postoje brojne knjige posvećene isključivo Webu. U tim knjigama možete pronaći mnogo štošta, od listi razlidtih Web sajtova do diskusije o Web protokolima, koji mogu pomoći da kreirate sopstvene interaktivne Web stranice. Bez sumnje, Web Je značajna tema na ovom polju i treba ga razmotriti. Verujemo da je značajno predstaviti kombinaciju teorije i praktične primene. Tako se proširuju osnovna znanja i stiču neke nove veštine. Odavde pa do kraja ovog odeljka naš cilj je predstavljanje:

- opšteg pregleda Weba i njegovog osnovnog operativnog koncepta
- opšteg pregleda načina za kreiranje Web stranice (predstavljamo glavne korake za generisanje minimalno funkcionalne Web stranice)
- integrisanja programiranja i razvoja Web stranica (ovo je ono čemu, u stvari, težimo, a prethodne stavke su samo kratkoročni ciljevi neophodni da se stigne do ovog). Ponovo se vraćamo na klijent/server model i opisujemo kako da se uključi programiranje i na strani klijentskih i serverskih protokola.

Poslednja stavka predstavlja pravi izazov, prvenstveno zbog toga što postoji nekoliko različitih opcija, ali ovde ih ne možemo sve obuhvatiti. Čak i kada se izabere samo jedna opcija, postoji ogromna količina materijala i zato je neophodna selekcija.

Sledi lista nekih alata koji se koriste u Web okruženjima:

- **HTML (Hypertext Markup Language)** HTML ima oznake, ili tagove koji omogućavaju kreiranje i formatiranje Web stranice.
- **JavaScript** To je skript jezik čiji se kod ugrađuje u HTML dokument. Klijent izvršava kod kako bi prihvatio korisnički unos i odgovorio na njega.
- **VBScript** Izvršava sličnu funkciju kao i JavaScript, osim što se zasniva na programskom jeziku Visual Basic.
- **Java applet** Omogućava klijentu da izvršava rutine napisane u programskom jeziku Java.
- **XML (Extensible Markup Language)** XML je sličan HTML-u, osim što je HTML uglavnom zadužen za prikazivanje informacije na Web stranici, dok XML defmiše standarde za opisivanje onoga što podaci, u stvari, predstavljaju.
- **ASP (Active Server Pages)** HTML stranice sadrže skriptove napisane, na primer, u JavaScriptu, ili VBScriptu. Skriptovi prihvataju informacije od klijenta i opisuju akciju koju Microsoft Web server preduzima pre nego što vrati informacije nazad do klijenta.
- **Perl** To je skript jezik koji se često koristi, na primer, u LINIX, ili Linux okruženjima za CGI programiranje (biće opisano kasnije) za smeštanje informacija i pribavljanje informacija iz tekstualnih fajlova sa servera.
- **Programski jezik C** To je programski jezik čiji se kompajlirani kod može pozivati u Web okruženju radi izvršavanja akcija kao što je pretraživanje baze podataka.

Ovde nemamo dovoljno prostora za detaljno opisivanje svih ovih alata. Ipak, predstavićemo u osnovnim crtama HTML i JavaScript za programiranje na strani klijenta i navešćemo nekoliko primera Perl i C programa za CGI programiranje na strani servera i nekoliko radnih primera koji predstavljaju razne aplikacije na Webu. Iako ovde nećemo navoditi sve detalje, možemo predstaviti dovoljno informacija da i sami poćnete da pišete programe za Web.

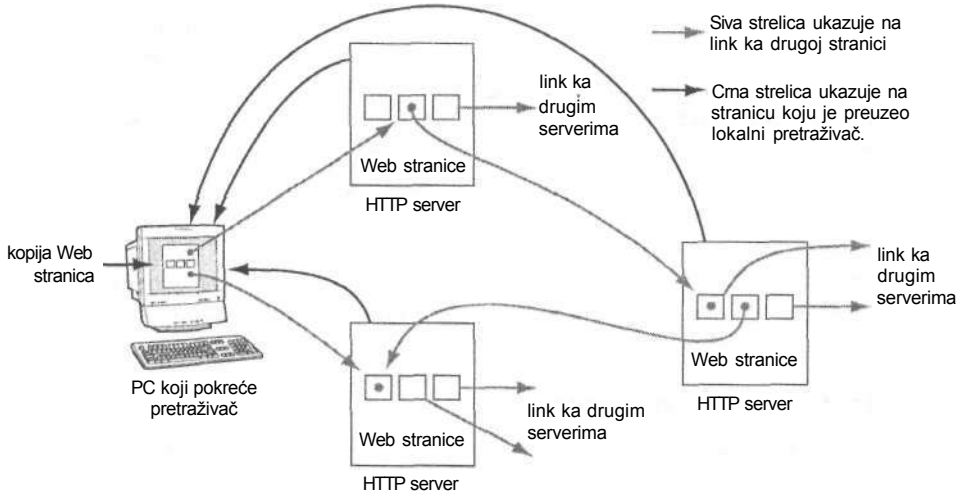
Pristup Web stranicama

Pretpostavićemo da čitaoci već imaju iskustva sa Webom, bar u vidu pristupa raznim sajtovima i praćenju (kliktanju) linkova ka drugim sajtovima. Lakoća kojom Web može da se koristi predstavlja i ogromnu prednost i (prema nekima) nedostatak. Počinjemo opisivanjem osnovnih akcija koje se dešavaju kada surfujete Webom.

Osnova svih Web operacija je **HTTP (Hypertext Transfer Protocol),** klijent/server protokol dizajniran tako da omogući razmenn informacija preko Weba. HTFP definiše tipove zahteva koje pretraživač može da postavi i tipove odziva koje server vraća. Preko HTIP-ja korisnik može da pribavlja Web stranice sa udaljenih servera i da smešta stranice na server ako ima odgovarajuće pravo pristupa. Osim toga, obezbeduje i mogućnost dodavanja novih informacija na postojeće stranice, Oi potpunog brisanja stranice. Ovde se fokusiramo strogo na aspekat pribavljanja.

Na slici 12.6 prikazani su osnovni koncepti uključeni u ceo proces. Korisnik na svom personalnom kompjuteru pokreće pretraživač (browser). Kao pretraživači najčešće se koriste Netscape Navigator i Internet Explorer. Pretraživač se pokreće na strani klijenta i može da prikazuje sadržaj Web stranice*. Ubrzo ćete videti kako izgleda tipična Web stranica, ali za sada ćemo samo pretpostaviti da je reč o dokumentu (fajlu) koji predstavlja informacije za korisnika. Osim toga, Web stranica ima *linkove* (reference) ka drugim Web stranicama. Ove stranice se smeštaju u pomoćnom skladištu kojem pristupaju udaljeni HTTP serveri.

Korisnik može da prati link postavljajući kursor miša preko naznačenog teksta, ili slike i klikom na taster miša. U toj tački lokalni HTIP klijent šalje zahtev udaljenom HTTP serveru za naznačenu Web stranicu. Server reaguje prenosom stranice nazad do klijenta, gde eventualno zamenjuje staru stranicu na ekranu (iako se ona obično baferuje za slučaj da korisnik želi da se vrati na nju). Korisnik može ponovo da klikne neki link na toj novoj stranici i na taj način poziva sledeću Web stranicu. Sledeći udaljeni HTTP server reaguje na isti način i prenosi zahtevanu Web stranicu nazad do pretraživača. Korisnik ovaj proces može da ponavlja sve dok postoji potreba. Linkovi mogu da povezuju dokumente smeštene na HlTP serverima koji su smešteni širom zemaljske kugle, a samo jednim klikom na taster miša stranica se prenosi do Vašeg lokalnog kompjutera.



SLIKA 12.6 *OpUi pregled Weba*

*Web stranice je možda lačnije nazivati *HTML dokumenli,* ali to ćemo razmotriti nešto kasnije.

Naravno, pojednostavili smo "stvari". Možda bi bilo ispravnije reći da smo uzeli u obzir postojanje TCP, IP i protokola veze koji su zaduženi za detalje konkretnog transfera. Ponovo dolazi do izražaja prednost slojevite strukture, jer su zadaci viših slojeva jednostavniji za opisivanje ukoliko pretpostavimo da postoje niži slojevi,

U suštini svega ovoga nalazi se odgovor na pitanje kako pretraživač zna gde da traži zahtevanu stranicu. Svaka stranica koja je dostupna preko Weba mora da ima jednistveni naziv kako bi se izbegle zabune. Da bi to bilo moguće, neophodni su lokacija stranice, jedinstveni naziv te lokacije i protokol za pristup toj stranici. Kolektivno, ove tri stavke defmišu **URL (Uniform Resource Locator)** stranice, koji ima oblik

protokol: //adresa-sajta/naziv

Na primer, autorovoj Web stranici (još se naziva i *polazna, Ui matična stranica - home page)* možete da pristupite preko URL-a

http://WWW. uWgb.edu/shayW

Protokol je HTTP*, a adresa sajta je WWW.uWgb.edu. Poslednji deo shayW predstavlja indirektnu referencu na specifični fajl. String "shayW" predstavlja nalog na host kompjuteru WWW.uWgb.edu. Nalog sadrži fajl pod nazivom index.htm. Referenciranje navedenog URL-a inicira prikazivanje sadržaja fajla u pretraživaču.t Ovo je konvencionalni način pristupa matičnim stranicama bez potrebe da se vodi računa o konvencijama imenovanja na host kompjuteru. Naravno, ako želite da pristupite nekom drugom fajlu, morate da ga naznačite eksplicitno. Na primer, syllabus za predavanja o mrežama je u fajlu syll358.htm u autorovom Web direktorijumu. Toj stranici možete da pristupite sa

http://WWW.uWgb.edu/shayW/syll358.htm

Ne bi imalo smisla kreirati liste svih stranica kojima ćete možda pristupiti sa eksplicitnim referencama na te stranice. Alternativa direktnom naznačavanju stranice je praćenje linka ka njoj radi pristupa sa druge Web stranice. Ovo se realtivno jednostavno izvodi, ali najpre moramo da objasnimo način na koji se sadržaj Web stranice predstavlja u nečijem pretraživaču.

Hypertext Markup Language

Sledeći logičan korak je opisivanje načina na koji se Web stranica postavlja i kako se u nju uključuju linkovi ka drugim stranicama. Jezik koji to omogućava je **Hypertext Markup Language (HTML).** U osnovi, korisnik kreira fajl (obično sa ekstenzijom .htm, ili .html) sa HTML elementima. Ovaj fajl (poznat i kao **HTML dokument)** definiše kako će Web stranica biti prikazana u pretraživaču (ovde ćemo ignorisati isključivo tekstualne pretraživače). Ipak, pre nego što predstavimo opšti pregled HTML-a, potrebno je razumeti šta može da se nade u HTML dokumentu.

* HT11' protokol nije jedini mogući prefiks koji se koristi u URL-u. Mogud su i drugi protokoli, kao što su FTP, Telnet i email.

t U zavisnosti od sajta, postoje neke manje razlike. Neki sajtovi zahtevaju tilda karakter (-) pre imena korisnika, neki inicijalno koriste fajl pod nazivom index.html (umesto index.hlm), dok neki zahtevaju da se fajlovi nalaze u odredenom poddireklorijumu konkretnog korisničkog naloga. Proverite dokumentaciju Vašeg sajta da bistc videli šta se inicijalno koristi.

Kada se Web stranica prikazuje u pretraživaču, korisnik može da vidi sledeće tipove sadržaja:

- obični tekst
- grafičke elemente, ili animirane slike
- linkove ka drugim HTML dokumentima. Linkovi se obično predstavljaju podvučenim tekstom u drugoj boji, ili pomoću slike. Postavljanjem kursora preko teksta, ili slike i klikom na taster miša preuzima se novi HTML dokument, koji se prikazuje u prozoru pretraživača.

U opštem slučaju, HTML može da obezbedi stranice prilično sofisticiranog izgleda, sa interesantnim formatima i šarenim pozadinama i slikama. Pošto autor nije preterano "vešt sa bojama", koncentrisaćemo se samo na osnovnu funkcionalnost. Generisanje kreativnih Web stranica prepuštamo čitaocima sa većom maštom. Takode nečemo pokušavati da kompletiramo razmatranje o HTML-u, ali ćemo opisati neke najčešće korišćene elemente. Postoji bezbroj njiga o HTML-u, u kojima možete da pronađete sve potrebne informacije.

Tagovi HTML koristi *tagove* da bi ukazao šta želi da uključi u HTML dokument i kako želi da se ti podaci prikažu. Obično se zapisuju na sledeći način:

<*tag opcije*> ... neke stvari između... </*tag*>

Neki tagovi zahtevaju i početni i završni delimiter. Oba uključuju identifikator taga, ali završni delimiter sadrži i kosu crtu /. Ovi delimiteri ukazuju pretraživaču šta da radi sa "stvarima" između delimitera. U nekim slučajevima opcije navedene u početnom delimiteru obezbeduju dodatne informacije.

Na slici 12.7 prikazani su primer HTML dokumenta i neki tipični tagovi koji posmatraču omogućavaju da vidi grafičke elemente, neki osnovni tekst i linkove ka dmgim dokumentima. Na slici 12.8 data je stvarna prikazana stranica. Prva i poslednja linija slike 12.7 odgovaraju HTML tagu (<html> i </html>), koji ukazuje da sve što je navedeno izmedu treba da se interpretira u skladu sa HTML pravilima. Svaki HTML dokument ima zaglavlje *[head)* i telo *(body).* Tagovi <head> i </head> ograničavaju zaglavlje, a tagovi <body> i </body> ograničavaju telo. U zaglavlju može da se nade nekoliko "stvari", od kojih je jedna naslovna linija (delimiteri <title> i </title>). Pretraživač uzima tekst izmedu naslovnih tagova i prikazuje ga na vrhu prozora. Vidite il frazu "The Title Goes Here" na slici 12.8?

U telo se smeštaju sve tekstualne informacije, grafički elementi i, eventualno, tekst hiperlinkova. Na primer, da bi bila prikazana slika, najpre morate da je kreirate i snimite u nekom fajlu. HTML tag za slike (<img>) naznačava taj fajl i slika će biti prikazana kada se dokument prikazuje u pretraživaču. Opcija u tagu slike na slici 12.7 naznačava fajl kao "logo.gif". Na slici 12.8 je kao slika upotrebljen logo znak jednog univerziteta, koji je ranije kreiran i snimljen u tom fajlu.

```
<html>
<head>
<title> The Title Goes Here </title>
</head>
<body>
center> <img src = "logo.gif"></center>
<hr>
<center> <h1>Links to Courses</h1> </center>
This is a short paragraph containing links to three courses. Each can be accessed
by clicking on the cource number. The first one is titled<B> Numerical
Analysis</B> and has course number of <A HREF = "http:/WWW.uWgb.edu/shayW
/syll350.htm">266-350</A>. The second course is titled <B>Data Structures</B> and
has a course number <A HREF = "http://WWW.uWgb.edu/shayW/syll351.htm">
266-351</A>. Finally the third course is titled <B> Data Communications and
Computer netWorks</B> and has a course number <A HREF = "http:/ WWW.uWgb.edu/
shayW/syll358.htm">266■358</A>.
<hr>
If you nave any comments or questions you can send them to <A HREF = "mailto:
shayW@uWgb.edu"> Blll Shay</A> at the University of Wisconsin-Green Bay
<hr>
</body)
</html>
```

SLIKA 12.7 *Primer HTML dokumenta*

SLIKA 12.8 *Izgled HTML dokumenta prikazanog u slici 12,7*

Delirniteri taga za centriranje (<centei> i) postavljeni sa obe strane taga za sliku izazivaju poslavljanje slike na sredini ekrana. Ako delimiteri taga za centriranje nisu prisutni, logo znak sa slike 12.8 bi se našao na levoj strani ekrana. LJ telu dokumenta nalazi se nekoliko tagova (<hr>) koji umeću horizontalnu liniju. Na slici 12.8 prikazane su tri tamne horizontalne linije koje se koriste kao separatori. Ovo je uobičajena vizuelna pomoćza razdvajanje različitih delova prikaza.

U većem delu tela dokumenta nalazi se tekst, koji je prikazan na slici 12.8. Postoje neke razlike u načinu prikazivanja teksta - odredeni delovi su veći, neki su boldirani, a neki su podvučeni (oni, u stvari, odgovaraju linkovima). HTML tagovi omogućavaju ove razlike. Delimiteri taga zaglavlja <hl> i </hl> definišu zaglavlje 1 nivoa. Sav tekst koji se nalazi između njih prikazuju većim fontom. Postoje i tagovi za <h2>, <h3>, <h4>, <h5> i <h6>, koji imaju sličan efekat, ali koriste različite veličine slova. Tag <hl> defmiše najveće, a tag <h6> najmanje zaglavlje. Sav tekst između tagova za boldiranje (<B> i </B>) prikazuje se boldirano.

Za nas su ovde najznačajniji *uporišni tagovi (anchor tags)* (<A> i </A>), koji definišu akcije koje korisnik može da izabere jednostavnim klikom na taster miša. Delimiteri uporišnog taga imaju nekoliko oblika i namena. Jedan mogući oblik je

<A HREF=url> *tekst na koji se može kliknuti* </A>

Ovi delimiteri imaju dvostruku funkciju. Prva je prikazivanje teksta koji može da se klikne*. Ovo znači da možete da postavite kursor preko tog teksta i klikom na taster miša pribavljate udaljeni doktiment. Druga je definisanje URL-a tog dokumenta. Na primer, na slici 12.7 u telo dokumenta je ugraden sledeđ tag:

<A HREF = "http://WWW.uWgb.edu/shayW/syll350.htm">266-350</A>

Opcija HREF definiše URL dokumenta smeštenog u autorovom Web direktorijumu. Tekst koji je moguće kliknuti u ovom slučaju je "266-350". Na slici 12.8 ovaj tekst je prikazan kao podvučen. Korisnik može da postavi kursor preko tog teksta i nakon klika na taster pribavlja željeni dokument - u ovom slučaju fajl pod nazivom syll350.htm iz autorovog Web direktorijuma.

Uporišni tagovi mogu da se koriste i za iniciranje drugih akcija, osim onih koje pribavljaju HTML dokumente. Na dnu slike 12.7 postoji još jedan uporišni tag

<A HREF= "mailto:shayW@uWgb.edu">Bill Shay</A>

U ovom slučaju tekst odgovara autorovom imenu. Opcija HREF identifikuje program mail servera. Kada se aktivira, prikazuje se forma koja korisniku omogućava kreiranje email poruke koja će biti poslata na naznačenu email adresu (u ovom slučaju shayW@uWgb.edu). Na formi se nalazi i dugme Send, koje, kada se klikne, šalje poruku.

Tabela 12.4 sadrži neke druge često korišćene HTML tagove sa kratkim opisima njihovih funkcija. Postoji još mnogo drugih tagova i opcija, pored liste tagova koju smo naveli u tabeli. Zainteresovani čitaoci mogu da pogledaju bilo koju knjigu o HTML-u.

* Tag za sliku može da se koristi umesto teksta, tako da se dobija slika koju možete da kliknete.

Tabela 12.4: Neki HTML tagovi

Tag	Značenje
</K>... </A>	Uporišni tag. Prikazuje tekst koji se nalazi izmedu delimitera. Ako se izrneđu delimitera nalazi tag slike, onda se prikazuje slika. Uporišni tag ima i opciju za navodenje URL-a za koji se kreira link, ako korisnik postavi kursor preko teksta, ili slike i klikne tasterom miša.
<B> ... </B>	Boldiranje. Tekst koji se nalazi između delimitera prikazan je boldiranim stilom.
<body> ... </body>	Ograničava telo HTML dokumenta.
 	Prekid linije. Umeće prekid linije na Web stranici. Sve što se nade iza prikazuje se na početku sledeće linije.
...	Centrira tekst smešten izmedu delimitera.
 ... 	Opcije utiču na boju i veličinu teksta smeštenog izmedu delimitera.
<form>... </form>	Kreira i prikazuje formu u koju korisnik može da unosi informacije na naznačenim mestima. Forma može, nakon toga, da se usmeri na izvršavanje neke akcije. U nekim slučajevima ona se koristi i za prikazivanje rezultata koji se dobijaju na osnovu unetih informacija.
<hl>... </hl>	Prikazuje tekst smešten između delimitera sa većim slovima. Postoje i tagovi <h2>, <h3>, <h4>, <h5> i <h6>, koji definišu različite veličine slova.
<head>... </head>	Ograničava zaglavlje HTML dokumenta.
<HR>	Prikazuje horizontalnu liniju, koja se koristi za vizuelno razdvajanje sadržaja Web stranice.
<html>... </html>	Ukazuje da sve izmedu ovih delimitera treba tretirati u skladu sa pravilima HTML jezika.
<I> ... </I>	Tekst koji se nalazi izmedu delimitera prikazuje kurzivom (italikom).
<img>	Naznačava fajl u kome se nalaze slika, ili animacija koja se prikazuje kada se Web stranica učita.
<input>	Koristi se sa tagom za formu, a omogućava korisniku da unese informacije.
	Ukazuje na element u listi.
<option>... </option>	Koristi za obezbedivanje opcija pop-up menija.
<P>	Označava start novog paragrafa na Web stranici.
<script>... </script>	Ograničava kod skript jezika za program na strani klijenta.
<select>... </select>	Omogućava korisniku da selektuje opciju iz liste u pop-up meniju.
<table>... </table>	Definiše tabelu koja će biti prikazana na Web stranici.
<title> ... </title>	Definiše naslov HTML dokumenta.
...	Definiše neuredenu listu elemenata.

HTML forme

Kao što smo ranije istakli, glavni cilj u ovom odeljku je da čitaoci upoznaju Web programiranje, ili mogućnost klijenta, ili servera da preduzmu odredene akcije, pored onih koje su naznačene pomoću HTML-a. Počinjemo razmatranjem HTML *formi.* Njih možete da smatrate i nekom vrstom šablona za unošenje i prikazivanje informacija. Forme su prilično česte, posebno kada se vrši pretraživanje Interneta pomoću pretraživačkih mašina, kao što je Google, ili Yahoo. Pre nego što zahteva pretraživanje, korisnik u formu unosi ključne reči, ili fraze. Korisnik može da klikne dugme Clear za brisanje onoga što je prethodno uneto, ili dugmad Submit, ili Search za predaju informacija pretraživačkoj mašini. U zavisnosti od pretraživačke mašine, korisnik možda može da selektuje i različite opcije koje utiču na rezultate pretraživanja, kao što je traženje isključivo zvučnih, ili grafičkih fajlova.

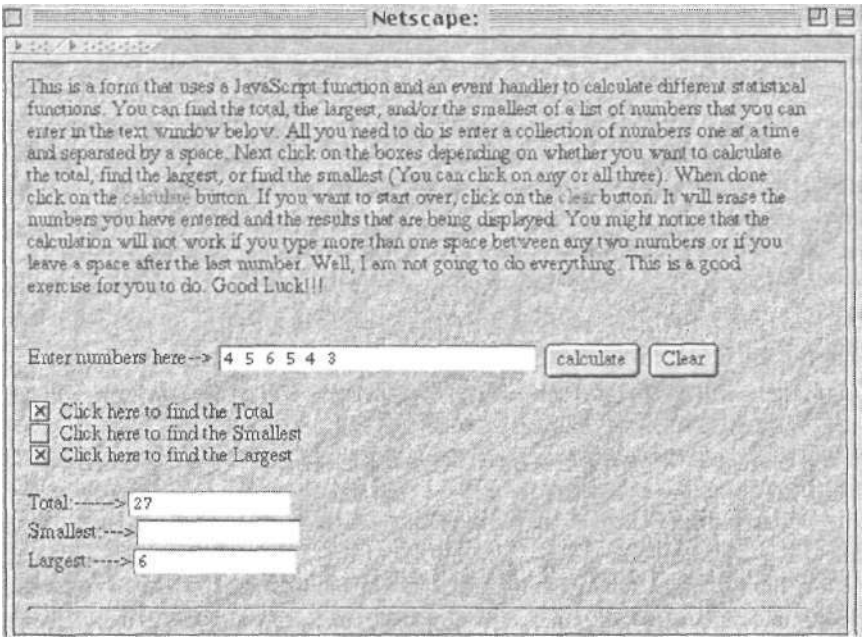
Predaja informacija iz formi često zahteva akcije i od klijenta i od servera. Na primer, možda Vaša lokalna picerija ima svoju Web stranicu preko koje možete da naručite picu. Možete da popunite formu sa svojim imenom, adresom, brojem telefona, sastojcima pice i tako dalje. Kada se informacije iz forme predaju, server treba da reaguje prikazivanjem cene i možda procenjenog vremena potrebnog za isporuku. Klijent može da preduzme i neke druge akcije, kao što je verifikovanje odredenih informacija iz forme. Na primer, ako neko namči veliku picu sa pečurkama, ljutim papričicama i kišnim glistama, klijent može da prepozna da jedan od ovih priloga nije adekvatan (nikada mi se nisu dopadale pečurke). Ideja je da se određeni zahtevi identifikuju kao besmisleni pre nego što zapadnete u nevolju nakon predaje informacija sa forme serveru. IJ opštern slučaju, ovo je mnogo efikasniji način, jer štiti Web softver od predaje zahteva koji se ne mogu ispuniti.

Razmatranje formi započinjemo primerom koji se u potpunosti može upravljati od klijenta. Na slici 12.9 prikazana je forma koja korisniku omogućava da unese niz brojeva, a zatim izračunava najmanji i najveći broj i sumu unetih brojeva. Postoje neka ograničenja, kao što je objašnjeno u instrukcijama forme. Forma sadrži sledeće elemente:

- tekst koji obezbeduje instrukcije za ono što treba da se uradi i kako treba da se uradi
- prazni okviri (tekstualni okviri) koji korisniku omogućavaju da unese nizove brojeva
- prostor za prikazivanje najvećeg i najmanjeg broja i sume unetih brojeva
- polja za potvrdu koja korisniku omogućavaju da izabere izračunavanja koja želi da se prikažu
- dugmad koja iniciraju izvršenje akcija. Moguće akcije su izvodenje željenih izračunavanja, ili čišćenje (brisanje) svih unetih brojeva. Druga opcija postoji u slučaju da ste napravili grešku i potrebno je da unesete novi niz brojeva.

Inicijalno, okviri na formi mogu da budu prazni. Na slici 12.9 prikazani su rezultati nakon što je korisnik uradio sledeće:

1. Uneo je brojeve 4, *5, 6,* 5, 4 i 3.
2. KHknuo je polja za potvrdu koja iniciraju izračunavanje sume i najvećeg unetog broja.
3. Kliknuo je dugme Calculate.



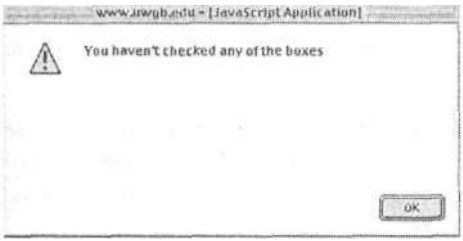
SLIKA 12.9 *HTML forma za statistička izračunavanja*

Nakon izvršavanja ovih akcija, u poljima Total i Largest prikazuju se brojevi 27 i 6, respektivno. Mo korisnik klikne dugme Clear, svi brojevi sa slike 12.9 nestaju. Korisnik nakon toga može da unese novu kolekciju brojeva i tako dobije nove rezultate.

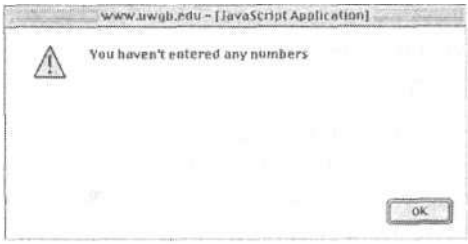
Ako korisnik poštuje naznačene instrukcije, dobiće rezultate. Ali, šta ako uradi nešto što *nije* u skladu sa instrukcijama? Svako ko je ikada programirao bilo šta zna da je jedan od najtežih aspekata programiranja reagovanje na ono što korisnik ne bi smeo da uradi. Na primer, šta ako korisnik unese brojeve, ali ne selektuje ni jedno polje za potvrdu pre klika na dugme Calculate? Sta ako korisnik ldikne dugme Calculate pre nego što unese bilo koji broj?

Na slici 12.10 ukazujemo na moguće odzive na takve akcije. Na slici 12.10a prikazan je prozor koji se otvara ako korisnik klikne dugme Calculate pre nego što potvrdi bilo koje polje za potvrdu, a na slici 12.1 Ob prozor koji se otvara ako korisnik klikne dugme Calculate pre nego što unese bilo koji broj. U svakom slučaju, korisnik bi kliknio dugme OK i pokušao ponovo da skoristi formu na ispravan način.

Sve ovde opisane akcije morala je da isprogramira osoba koja je postavila Web stranicu i te akcije su mogle da se kontrolišu preko skriptova, ili skript jezika. **Skript** je, u suštini, program koji se automatski izvršava kao odziv na neku drugu akciju. **Skript jezik** je, naravno, jezik koji se koristi za pisanje skriptova.



(a) Korisnik nije potvrdio ni jedno polje za potvrdu



(b) Korisnik nije uneo ni jedan broj

SLIKA 12.10 *Odzivi na neočekivane akcije horisniha*

Sada je potrebno da opišemo sve eletnente u primeru koji može da se postavi u okviru HTML dokumenta.

Prvi korak je definisanje načina na koji može da se definiše forma i kako se koristi za prihvatanje korisničkogulaza iza prikazivanje izlaza. Na slici 12.11 prikazanesu HTMLkomande koje definišu formu sa slike 12.9, Prva polovina forme jednostavno sadrži tekst koji obezbeduje instfukcije za korisnika, kao šlo je prikazano na slici 12.9. U okviai teksta postoje dva taga sa opcijama za bojenje teksta, koji naznačavaju 24-bitnu vrednost boje (po osam bitova za crvenu, zelenu i plavu boju) delimitiranog teksta u okviru tih tagova.

```
<form>
This is a form that uses a JavaScript function and an event handler to calculate
different statistical functions. You can find one total, the largest, and/or the
smallest of a list of numbers that you can enter in the text WindoW belloW. AIl
you need to do is enter a collection of numbers one at a time and separated by a
space. Next click on the boxes depending on Whether you Want to calculate the
total, find the largest, or find the smallest. (You can click on any or all
three).
When done click on the <F0NT COLOR="ff0000">calculate </FONT>button. If you Want
to start over, click on the <F0NT COLOR="ff0000">clear </FONT>button. It Will
erase the numbers you have entered and the results that are being displayed. You
might notice that the calculation Will not Work if you type more than one space
betWeen any tWo numbers or if you leave a space after the last number. WeIl, I am
not going to do everything. This is a good exercise for you to do. Gooo
LucK!!!<BR><BR><BR>
<left> Enter numbers here-> <Input type="text" Name="expr" size=30>
<Input Type="button" Value="calculate" Onclick="compute(this.form)">
<Input Type="reset" Value="Clear"><BR><BR</left>
<Input Type="checkbox" Name="gettotal">Click here to find the TotakBR>
<Input Type="checkbox" Name'="getmin">Click here to find the Smallest<BR>
<Input Type="checkbox" Name="getmax">Click here to find the Largest<BR><BR>
Total :—><Input Type="text" name="total" size=15><BR>
Smallest:— -><Input Type="text" name="min" size=15><BR>
Largest:—><Input Type="text" name="max" size=15><BR>
</Form>
```

SLIKA 12.11 *HTMLkomande za definisanje forme sa slike 12.9*

U ovom slučaju vrednost boje #FF0000 znači da postoji samo crvena komponenta, tako da se delimitirani tekst (u ovom primeru reči *calculate* i *clear)* u Web pretraživaču prikazuje crvenom bojom.

Naredne linije sa instrukcijama sadrže *ulazne (input) tagove,* koji defmišu način na koji se informacije unose na formu. Svaki okvir, dugme, ili polje za potvrdu sa slike 12.9 definisani su jednim ulaznim tagom. Primećujete da svaki ulazni tag ima nekoliko različitih opcija, ili tipova. Sledi pregled opcija:

- Type = "text" Ova opcija je namenjena slobodnom unosu i prikazivanju rezultata. Prikazuje tekstualno polje u koje korisnik može da unese bilo šta po sopstvenom izboru. Kao što ćete videti, skript klijenta može da koristi ovaj tip polja i za prikazivanje informacija. Na slici 12.9 brojevi koje korisnik unese i eventualni rezultati prikazuju se pomoću ovog tipa.
- **Type = "checkbox"** Ovaj tip omogućava korisniku da postavi kursor iznad polja i da selektuje polje ldikom na taster miša. Kada se selektuje, prikazuje se simbol X (videti sliku 12.9). Korisnik može da selektuje više polja za potvrdu po sopstvenom izbom.
- **Type = "reset"** Ova opcija izaziva prikazivanje dugmeta koje, kada se klikne, briše sve informacije koje je korisnik eventualno uneo **u** formu. Ulazni okviri se ili brišu, ili **se** vraćaju na podrazumevane vrednosti koje su postavljene kada je forma definisana.
- **Type = "button"** Ova opcija kreira dugme koje, kada se klikne, inicira određenu akciju. Uskoro ćete videti kako se definiše željena akcija.
- **Name = *"ne\d_rumv"*** Ova opcija dodeljuje naziv nekoj ulaznoj oblasti (okvir, dugme, ili polje za potvrdu). Naziv nije vidljiv za korisnika, ali, kao što ćete kasnije videti, skript jezik može da iskoristi ove nazive za proučavanje korisničkog unosa.
- **Value = *"neka_yrednosf*** U opštem slučaju, opcija vrednosti zavisi od tipa ulaza. Ovde smo kao ulazne tipove koristili samo "button" i "reset". Kao što je prikazano na slici 12.9, string koji je dodeljen vrednosti prikazuje se na dugmetu na formi.
- Size = ***broj*** Kada se koristi sa tekstualnim okvirom, ova opcija definiše veličinu okvira (mereno u brojevima karaktera).
- **OnClick = *"neka_aktija"*** Ovo je primer kontrolisanja dogadaja *(event handler),* ili definisanja neke akcije koja se dešava ako dođe do odredenog dogadaja. Kada se primenjuje za dugme, ova opcija definiše skript koji se automatski izvršava kada se klikne dugme (dogadaj). Nesumnjivo, skript hvata formu i sve što ona sadrži i izvršava neku akciju.

Programiranje na strani klijenta i JavaScript

Finalni korak je objašnjenje kako se piše skript koji odgovara formi sa slike 12.9. Potreban nara je skript koji, kada korisnik klikne dugme Calculate, obavlja sledeće:

- utvrdivanje vrednosti svakog polja za potvrdu koje je selektovano i postavljanje rezultata u odgovarajući tekstualni okvir

- generisanje okvira upozorenja (poruka greške) ako korisnik nije selektovao ni jedno polje za potvrdu
- generisanje okvira upozorenja ako korisnik nije uneo ni jedan broj

Slika 12.12 sadrži JavaScript kod koji ovo obavlja. Nismo pretpostavili da već znate JavaScript, već da poznajete programski jezik C i da razumete bar osnove objektno-orijentisanog programiranja i hijerarhije objekata.

```
<script language = javascript>

function MakeArray(form)
{
 var ind1=0;
 var ind1=0;
 var blank="  ";
 var i = 0;

 While ((ind2=form.expr.value.index0f(blank, indl)) != -1)
   {
     this[++i]=parseInt(form.expr.value.substring(ind1,   ind2));
     ind1=ind2+1
   }
  this[++i]=parseInt(form.expr.value.substring(ind1, form.expr.value.length));
  this.length=i;
}

function dototal(myarray, form)
{
  var sum=0;
  for (var i=1; i<=myarray.length; i++)
   um = sum + myarray[i];
  form.total.value=sum;
}
function domin(myarray, form)
{
  var temp=myarray[1];

  for (var i=2; i<=myarray.length; i++)
   if (myarray[i] < temp)
    temp = myarray[i];
  form.min.value=temp;
}
function domax(myarray, form)
{
  var temp=myarray[l];

  for (var i=2; i<=myarray.length; i++)
  if (myarray[i] > temp)
    temp = myarray[i];
  form.max.value=temp;
}

function compute(form)
{
  var myarray;
```

```
  if  (form.expr.value.lengtti==0)
    alert("You haven't entered any numbers")
  else
  if  (!form.gettotal.checked &&  !form.getmin.checked &&  !form.getmax.checked)
    alert("You haven't checked any of the boxes")
  else
  {
    my rray = neW MakeArray(form);
    if  (form.gettotal.checked)
     dototal(myarray,  form);
    if  (form.getmin.checked)
     domin(myarray,  form);
    if  (form.getmax.checked)
     domax(myarray,  form);
  }
}

</script>
```

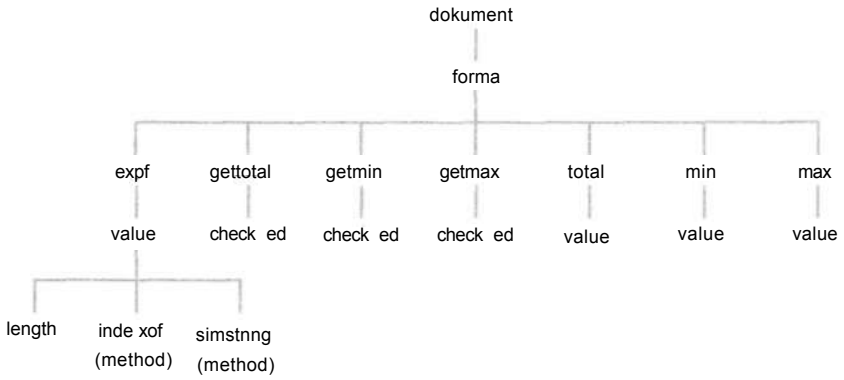
SLIKA 12.12 *JavaScript kod za pivgramiranje na strani klijenta*

Osim toga, nećemo pokušavati da obezbedimo nikakve detalje o JavaScriptu, osim onih koji su nam ovde neophodni. LI referencama [FlOl] i [GoOl] možete da pronađete informacije o JavaScriptu.

Pre nego što počnemo, postoji nekoliko "stvari" koje treba da napomenemo u vezi koda sa slike 12.12:

- JavaScript kod je delimitiran tagovima <script> i </script>.
- Kod se nalazi u zaglavlju MTML dokumenta.
- Kod sadrži pet funkcija: compute, MakeArray, dototal, domin i domax. Kao što smo naznačili OnClick event handlerom sa slike 12.11, kontrola se najpre prebacuje na funkciju compute, koja upravlja akcijama koje skript mora da izvede.
- Brojevi koje korisnik unese postoje u vidu stringa. JavaScript kod mora da parsira ovaj string, pronađe brojeve i smesti ih u niz.

JavaScript je objektno-orijentisani jezik sa složenom hijerarhijom objekata. Ovde nećemo opisivati celu hijerarhiju, ali je važno da razumete bar onaj njen deo koji je relevantan za kod sa slike 12.12. Na slici 12.13 prikazana je hijerarhija koja nam je polrebna. Formu sa kojom treba da radimo JavaScript vidi kao objekat. Forma je, ujedno, i svojstvo objekta *document* (smatrajte da je to naš HTML dokument). LI opštem slučaju, dokument može da ima više formi. Sa druge strane, forma ima svoja svojstva, koja su navedena na trećem nivou hijerarhije prikazane na slici 12.13. Svako svojstvo odgovara jednom ulaznom tagu sa slike 12.11. LI stvari, ako uporedite nazive opcija iz tih tagova i nazive sa slike 12.13, videćete da su isti. Kao što ćete uskoro otkriti, ovo omogućava pristup skripta objektima koji su pridruženi ovim tagovima. Slika 12.13 nije potpuna, ali ćemo dodatne napomene davati kako se bude ukazivala polreba.



SLIKA 12.13 *Hijerarhija JavaScript objekata relevantnih za sliku 12.12*

Sećate se sa slike 12.11 da ulazni tag tipa "button" ima opciju Onclick= "compute(this.form)". Kada korisnik klikne to dugme, kontrola se prenosi na JavaScript funkciju pod nazivom compute. Osim toga, tekući objekat forme, zajeđno sa svim svojim svojstvima i metodima, prenosi se do te funkcije. Ključna reč this obično se koristi u JavaScriptu za referenciranje tekućeg objekta. Naravno, šta se smatra *tekućim* zavisi od konteksta u kome se koristi. U ovom slučaju this.form se odnosi na tekuću formu (onu sa slike 12.11) u tekućem HTML dokumentu.

Kada se kontrola prenese na funkciju compute, njen parametar omogućava JavaScripl kodu da pristupi svim informacijama u vezi forme preko objekta pod nazivom f orm. Ova funkcija prvo proverava da li je korisnik uneo bilo koji broj u formu. Sećate se sa slike 12.11 da forma sadrži ulazni tag tipa "text" pod nazivom expr. Tu ide ono što korisnik unese, a expr je ujedno i svojstvo objekta forme (videti sliku 12.13). Dalje, hijerarhija objekata pokazuje da expr ima svojstvo value. Predstavlja konkretni string unet preko tekstualnog okvira expr. Na kraju, pošto svaki string ima odredenu dužinu koja predstavlja broj karaktera u stringu, value ima svojstvo length. Ako korisnik ne unese ni jedan karakter, dužina stringa je 0. Zato JavaScript proverava ovuopciju poredenjem form.expr.value.lengthsa 0 (primećujete da izraz sa tačkama odgovara sekvenci svojstava definisanoj hijerarhijom objekata sa slike 12.13). Ako su vrednosti identične, JavaScript poziva funkciju alert, prenoseći joj string "you haven't entered any numbers". Funkcija alert otvara prozor poput onog na slici 12.10b. Naravno, ako dužina stringa nije 0, taj korak se "preskače" i ne prikazuju se nikakvi okviri upozorenja. Napomenimo da nenulta dužina stringa ne mora da znači da je korisnik stvarno uneo brojeve. To samo znači da je uneo nešto. Ovaj skript ne proverava da li se u stringu nalaze nenumerićki karakteri. Ostavljamo Vam da izvežbate provern grešaka u skriptu.

Zatim, JavaScript mora da utvrdi da li su selektovana neka polja za potvrdu. Ponovo za to koristi hijerarhiju objekata. Postoje tri polja za potvrdu, pod nazivima gettotal, getmin i getmax - svako predstavlja svojstvo objekta forma.

Pošto je svako polje defmisano tipom "checkbox", svako ima svojstvo pod nazivom checked. Ako korisnik postavi kursor preko polja za potvrdu i klikne tasterom miša, odgovarajuće svojstvo checked se postavlja na TRUE. Dakle, svako svojstvo checked je ili TRUE, ili FALSE, u skladu sa tim da li je korisnik selektovao polje, ili nije. Druga if naredba u funkciji compute poredi svojstvo checked svakog polja za potvrdu; ako su sva svojstva FALSE, prikazuje okvir upozorenja.

Ako je selektovano bar jedno polje za potvrdu, kontrola se prosleđuje na poslednju klauzulu else. Prva linija,

```
myarray=neW  MakeArray(form)
```

poziva funkciju koja parsira string koji je korisnik uneo i smešta sve brojeve u niz pod nazivom myarray (videćete ubrzo kako ovo funkcioniše.) Preostale tri if naredbe ponovo utvrđuju koja su polja za potvrdu selektovana. Za svako selektovano polje JavaScript poziva funkciju da izvede naznačeni zadatak. Na primer, funkcija dototal sumira vrednosti iz niza myarray pomoću promenljive sum. Poslednja linija u toj funkciji smešta sumu u form.total.value. Ovaj izrazprati hijerarhiju objekata i dodeljuje promenljivu sum kao svojstvo value tekstualnog okvira pod nazivom total. Zbog toga, stvarna suma se prikazuje u tom okviru na formi, kao što je prikazano na slici 12.9.

Funkcije domin i domax utvrđuju najmanju i najveću vrednost u myarray, respektivno. Na sličan način se dodeljuju i rezultati za dototal.

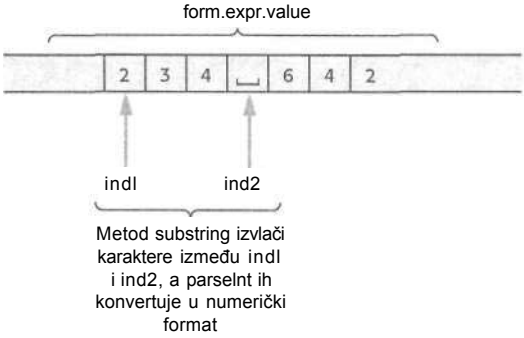
Poslednji deo slagalice je opisivanje naćina na koji JavaScript izvlači brojeve iz stringa koji unese korisnik. Potpuno razumevanje ovog procesa verovatno nije moguće bez kompletnijeg opisa JavaScripta i njegovih objekata i tipova. Ovde predstavljamo samo glavne ideje, a čitaocima prepuštamo da pročitaju neku knjigu o JavaScriptu radi detalja o implementaciji.

U suštini, linija

```
myarray=neW  MakeArray(  form)
```

(iz funkcije compute) poziva funkciju MakeArray, koja kreira i vraća objekat niza sa potrebnim brojevima. U okviru te funkcije JavaScript kod pretpostavlja da su brojevi smešteni u tekstualnom stringu sa razmakom od jednog blanko znaka izmedu svaka dva broja. Kao i ranije, JavaScript referencira string pomoću notacije form.expr.value. Nakon toga, poziva metod form.expr.value.indexOf (blank, indl), koji vraća prvi indeks (ili subskript) prvog blanko karaktera smeštenog na poziciji iza indl. Inicijalno, indl je 0, tako da prvi poziv ovog metoda pronalazi indeks prvog blanko znaka u stringu. Na slici 12.14 prikazano je kako su indl i ind2 povezani sa tekstualnim stringom. U opštem slučaju, oni ograničavaju karaktere koji predstavljaju brojeve koje je korisnik uneo. Dve vrednosti indeksa su korišćene u metodu substring, koji izvlači podstring sa karakterima izmedu blanko znakova. Konačno, JavaScript koristi funkciju parselnt za konvertovanje podstringa u numerićki format.

Dok prolazi kroz pedju, JavaScript smešta konvertovane podstringove u objekat niza pod nazivom this. Kao što smo ranije pomenuli, this je ključna reč koja se koristi za referenciranje objekta. U ovom slučaju funkcija MakeArray se ponaša slično kao konstruktor objekta niza myarray. Zato se this u ovom kontekstu odnosi na objekat niza myarray.



SLIKA 12.14 *Izvlačenje brojeva iz tekstualnog stringa*

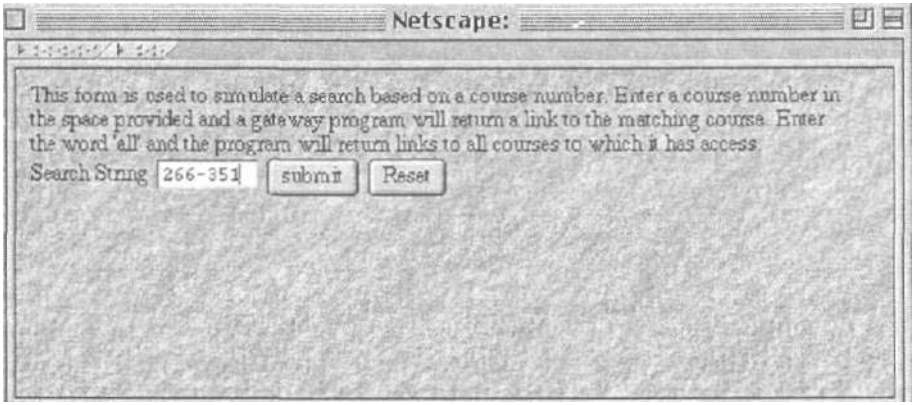
12.4 CGI i programiranje na strani servera: Postavljanje pretraživačke mašine

Sledeća tema ovog poglavlja tiče se programiranja na strani servera. Na primer, dok surfujete Webom, možda ste se zapitali kako pretraživačka mašina funkcioniše. Znate da prihvata jednu, ili više ključnih reči koje navedete i da na dmgom sajtu traži moguće reference i da ih šalje nazad do Vas. Naš zadatak je da opišemo kako ovo funkcioniše i da napišemo ogranak pretraživačke mašine. Kao i u slučaju prethodno obradenih tema, postoji nekoliko načina da se izvede programiranje na strani servera. Ipak, ponovo ćemo se fokusirati na jednu konkretnu, ali prilično standardizovanu tehniku. Oni koji su zainteresovani za alternativna rešenja mogu da konsultuju neku knjigu o HTML programiranju, ili razvoju Weba.

Već znamo da HTTP server može da obavlja neke jednostavne poslove, kao što je slanje zahtevanih dokumenata nazad do HTTP klijenta. Medutim, neke aktivnosti, kao što su one koje postoje kod pretraživačkih mašina, složenije su od onih koje HTTP serveri mogu da izvršavaju. U takvim slučajevima HTfP server se oslanja na *gateWay program* koji upravlja dodatnom logikom. GateWay programi mogu da budu napisani jezicima kao što su C, ili Perl (videti reference [Fe97], [GuOO], [MeOl]Hi [ScOl]), ili, čak, shell skriptovima kao što je Bourne shell skript u UNIX-U. Treba da upoznate Common GateWay Interface (CGI), koji serveru omogućava komunikaciju sa gateWay programom. Kada to uradite, deli Vas samo jedan mali korak od sposobnosti za pisanje sopstvenih gateWay programa.

Postoje tri pitanja na koja je potrebno dati odgovore:

1. Kako gateWay program dobija informacije od klijenta?
2. Kako gateWay program obavlja svoj posao?
3. Kako gateWay program šalje informacije nazad do klijenta?



SLIKA 12.15 *Inlerfejs forme za pretraživaćku mašinu*

Forme

Ođgovor na prvo pitanje nalazi se u HTML formi. Na slici 12.15 prikazan je primer interfejsa z; pretraživačku mašinu koju ćemo simulirati. Sadrži samo funkcionalne delove koji su neophodni z; postavljanje zahteva za pretraživanje: neke jednostavne instrukcije, tekstualni okvir u koji se unosi reči koje se traže, dugme za brisanje onoga što je uneto i dugme za iniciranje pretraživanja.

Na slici 12.16 prikazan je HTML kod koji generiše ovu formu. Liči na HTML kod sa slike 12.11 osim što postoje dve "stvari" koje ranije nismo predstavljali. Prvo, tip "submit'Ti ulaznom tagi izaziva prikazivanje dugmeta na formi. Klikom na to dugme forma i informacije koje su unete n; nju prenose se serveru. Drugo, opcije u tagu <form> ukazuju na to kako se informacije prenosi (method = "get")* i

```
<form method "get" action="http://icsc.uWgb.edu/~shayW/search.cgi">
This form is used to simulate a search based on a course number. Enter a
course number in the space provided and a gateWay program Will return a
link to the matching course syllabus. Enter the Word all and the program
Will return links to all courses to Which it has access.
<BR><BR>
Search String <Input Type="Text" Name="srch" size=8>
<Input Type="submit" Value= "submit">
<Input Type="reset">
</form>
```

SLIKA 12.16 *HTML kod za formu sa slike 12.15*

* Metod get je način da se pošalju informacije do gateWay programa. Ahernative potražitc u nekom tekstu o HTML, ili CGl programiranju.

URL izvršnog fajla koji predstavlja gateWay program (action ="http://icsc.uWgb.edu/search.cgi).* Druga opcija ukazuje da je gateWay program fajl pod nazivom search.cgi smešten u poddirektorijumu shayW na Red Hat Linux serveru.

Stringovi upita

Kada se informacije sa forme prenose do servera pomoću metoda get, HTTP klijent prenosi informacije u vidu stringa. String se sastoji od referenciranog URL-a i liste parova naziv/vrednost, u obliku *"naziv vrednost"*. Obično postoji po jedan par naziv/vrednost za svaku ulaznu stavku koja se koristi za unošenje informacija. Naziv je definisan u HTML kodu, a vrednost je ono što korisnik unese. Znak pitanja (?) razdvaja parove naziv/vrednost međusobno i od HRL-a. Na primer, pretpostavimo da je korisnik u formi sa slike 12.15 uneo "266-351". HTTP klijent šalje string u obliku

http://icsc.uWgb.edu/-shayW/search.cgi?srch=266-351

Pošto postoji samo jedno ulazno polje u koje se unose informacije, string sadrži URL i samo jedan par naziv/vrednost, srch= 266-351.

Pošto opcija action naznačava gateWay program pod nazivom search.cgi, HTTP server prenosi string do njega. GateWay program mora da pristupi tom stringu i da izvuče neophodne informacije. Da bi se to izvelo, CGI koristi promenljive okruženja koje su dostupne gateWay programu. Postoji nekoliko različitih promenljivih okruženja, ali za nas je trenutno bitna promenljiva QUERY_STRING.

Na slici 12.17 prikazan je gateWay program pisan u programskom jeziku C, koji traži ono što je korisnik uneo u formu. Proučavanje gateWay programa pokazuje da je ovo samo ogranak koji je dizajniran tako da obezbedi odgovor na dva osnovna pitanja:

1. Da li mogu da se dobiju tačne informacije od promenljive okruženja?
2. Da li se odgovarajuće informacije mogu vratiti do korisnika?

Primer pretraživačke mašine

Program sadrži niz hard-kodiranih stmktura sa informacijama koje su pronadene (linije 16—20). Svaka struktura sadrži broj kursa (na primer, "266-351"), LIRL i naziv kursa. Ovaj program izvlači broj kursa iz promenljive okruženja QUERY_STRING (linija 26), traži niz sa podudarnim brojem (linije 56—63) i vraća klijentu i naziv kursa i URL. Preko vraćenog URL-a korisnik dobija referencu za selektovani kurs.

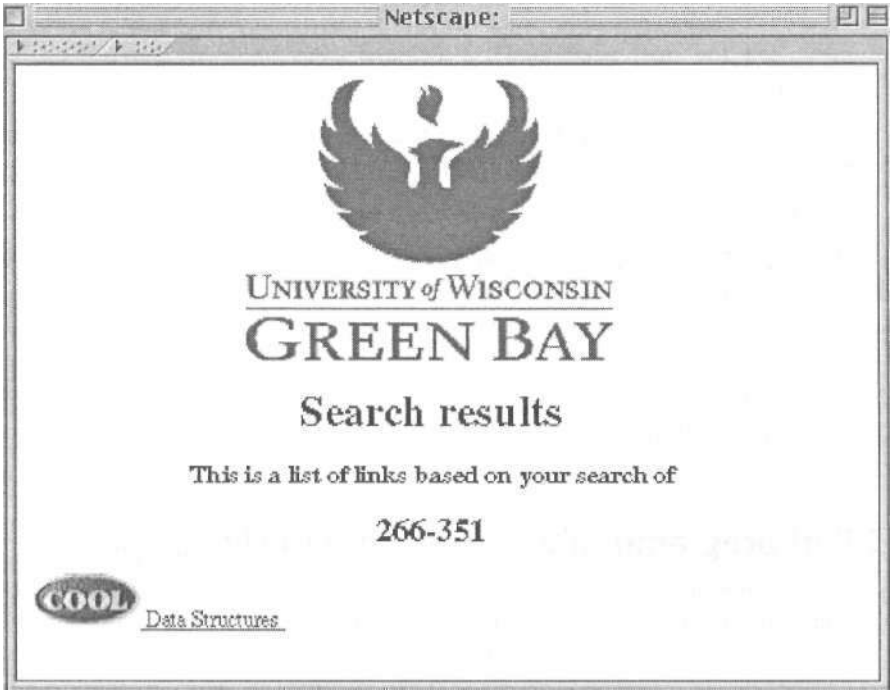
Na primer, ako je korisnik uneo "266-351", kao što je prikazano na slici 12.15, onda će vraćeni odziv izgledati kao na slici 12.18. Forma sa rezultatima pretraživanja prikazuje string koji je korisnik uneo, a ispod njega daje link predstavljen nazivom kursa. Pomoću ove forme korisnik može da klikne taj naziv da bi ispratio link. Imitirali smo sve što pretraživačka mašina radi. U našem gateWay progrnarnu postoji način za vraćanje svih LIRL-ova ako korisnik unese string "all" umesto broja kursa.

* Ovaj URL referencira gateWay program na autorovom sajtu i često se menja. Pokušaje postavljanja sopstvenih gateWay programa mora da odobri personal zadužen za održavanje mreže na željenom sajtu.

```
1    #include <string.h>
2    #include <stdio.h>
3    #include <stdlib.h>
4
5    typeset struct {
6        char cnum[32];
7        char url[50];
8        char cname[32];
9    } entry;
10
11   void  printheading(char  * ) ;
12   void listmatches(entry [], char * ) ;
13
14   int main(int argc,  char *argv[])  {
15
16    entry mystuff[25]=
17     { "266-350", "http://WWW.uWgb.edu/shayW/syll350.htm", "Numerical Analysis",
18       "266-351", "http://WWW.uWgb.edu/shayW/syll351.htm", "Data Structures",
19       "266-358", "http://WWW.uWgb.edu/shayW/syll358.htm", "Computer NetWorks",
20       "home",    "http:/WWW.uWgb.edu/shayW/",  "My Home Page");
21
22    int i;
23    int status;
24    char * srchstring; // string za pretrazivanje koji se prenosi preko html forme
25
26    srchstring = getenv(QUERY_STRING"); // uzima string za pretrazivanje iz okruzenja
27    printheading(srchstring);
28
29    if (srchstring = strchr(srchstring,  '='))
30    srchstring++;
31
32    listmatches(mystuff,  srchstring);
33     return 0;
34   }
35
36   void printheading(char * srchstring)
37   {
38     printf("Content-type; text/html\n\n");
39    printf("<html>\n");
40    printf("<body>\n");
41     printf("<center> <img src=http://icsc.uWgb.edu/-shayW/logo200.gif>\n");
42     printf("<h1>Search results<h1>\n");
43   }
44   void listmatches(entry mystuff[25], char * srchstring)
45   {
46     int  i;
47
48     if (strlen(srchstring) == 0)
49     {
50       printf("No search string is specified\n");
51       exit(0);
52     }
53     printf("<H3>This is a list of links based on your search of</H3>");
54     printf("<h2>%s<h2></Center>\n\n\n", srchstring);
55
56     for  (i=0;i<=3;i++)
57     {
58       if strstr(mystuff[i].cnum, srchstring |] strstr(srchstring, "all"))
59       {
60         printf("<img  src=http://icsc.uWgb.edu/-shayW/cool_fli.gif>");
61         printf("<A href=%s> %s </A>\n<BR>", mystuff[i].url , mystuff[i],cname);
```

```
62          }
63        }
64       printf("</body>\n");
65       printf("</html>\n");
66     }
```

SLIKA 12.17 *C kod za CCI programiranje na strani seivera*



SLIKA 12.18 *Rezultati pretraživanja*

Ovo je izvedeno da bi se testiralo da li gateWay program može da vrati više linkova.

Preostaje samo još da opišemo kako gateWay program sa slike 12.17 generiše rezultate koji su prikazani na slici 12.18. Veći deo programa je jasan svima koji znaju C, tako da ćemo se fokusirati samo na delove koji se tiču razmene informacija sa klijentom. Kao što je ranije istaknuto, gateWay program prvo mora da pribavi string upita. To radi pomoću naredbe iz linije 26:

```
srchstring = getenv("QUERY_STRING");
```

koja postavlja string u C promenljivu `srchstring`. U ovom primeru smo pretpostavili da forma dopušta samo jedno ulazno polje i da je korisnik uneo samo jednu frazu za pretraživanje (broj kursa, ili reč *all).* Ovo uprošćava logiku programa; ostavljamo Vam da vežbate proširenje ovog programa. Pošto string za pretraživanje ima oblik

```
url?srch=fraza
```

program treba samo da locira "=" kako bi pronašao frazu koju mora da pronađe. Koristi C funkciju za rad sa stringovima `strchr` kako bi se locirala "=" i redefinisala promenljiva `srchstring` za lociranje fraze (linije 29-30). Konačno, poziva funkciju `listmatches` (linija 32), prenoseći joj frazu.

Ova funkcija (linije 44-66) koristi standardno linearno pretraživanje niza kako bi se pronašla fraza i prikazali rezultati. Ipak, postoji jedna značajna razlika. Normalno, naredba `printf` šalje izlaz do standardnog izlaznog uredaja. Kada se pokreće iz komandne linije, obično se podrazumeva da se rezultati prikazuju na ekranu. Kada se pokrene kao CGI program, izlaz se šalje od dokumenta koji će se interpretirati kao HTML dokument. Proučavanje ovih naredbi otkriva da daju HTML kod.

Unutar petlje program traži podudarnost između fraze i broja kurseva u svakom elementu niza (linija 58). Kada se pronade poklapanje, naredba

```
printf("<A href=%s> %s </A>\n<BR>", mystuff[i J.url, mystuff[i].cname);
```

šalje i URL i naziv pridružen odgovarajućem zapisu do HTML dokumenta. Primetićete da se nalaze unutar uporišnog taga. Kada se program završi, HTTP server šalje ovaj dokument nazad do klijenta, gde se dokument prikazuje u skladu sa pravilima HTML-a. Kao rezultat, korisniku se prikazuje odgovarajući naziv, koji predstavlja referencu na vraćeni URL, koju korisnik može sa klikne. Pretraživanje je kompletno i korisnik može da klikne taj naziv kako bi ispratio referencu.

Ako je korisnik uneo "all", umesto broja kursa, videće reference ka svim URL-ovima navedenim u strukturi niza.

12.5 Perl programiranje: Sistem za naručivanje pice

Ovo poglavlje završavamo još jednim primerom koji prikazuje tipičnu primenu CGI skriptova. Ovoga puta koristimo programski jezik Pert i pokazujemo kako se postavlja program za naručivanje pica zasnovan na Webu. Perl (Practical Extension and Report Language) se obično pokreće na UNIX, ili Linux sistemirna. Njegova sintaksa podseća na C, ali on se obićno interpretira, umesto da se kompajlira. Perl programi mogu da se pokreću iz UNIX shella, ili im se može pristupiti iz forme na Web stranici. Proces je isti kao kada je korišćen C program u prethodnom odeljku. Jedina razlika je što URL naznačen u tagu forme navodi fajl Perl skripta, umesto kompajliranog C programa.

Pretpostavljamo je da je O´Malley's Pizzeria and Irish Pub postavila Web sajt koji omogućava naaidvanje pica preko Interneta. Ovaj Web sajt sadrži Perl skriptove koji obavljaju neke akcije na strani servera. Ovi skriptovi kreiraju HTML dokumente koji se vraćaju klijentu. Pretpostavljamo da čitaoci već poznaju HTML, JavaScript, C, interakciju sa formama i opšte principe klijent/server računarstva. Ovde nećemo kompletno predstaviti Perlu, ali ćemo dati dovoljno informacija da bi se objasnilo kako skriptovi funkcionišu. Dodatni materijal o Perlu može da se pronade u referencama [ScOl], [MeOl], i [GuOO].

Interakcija sa korisnikom

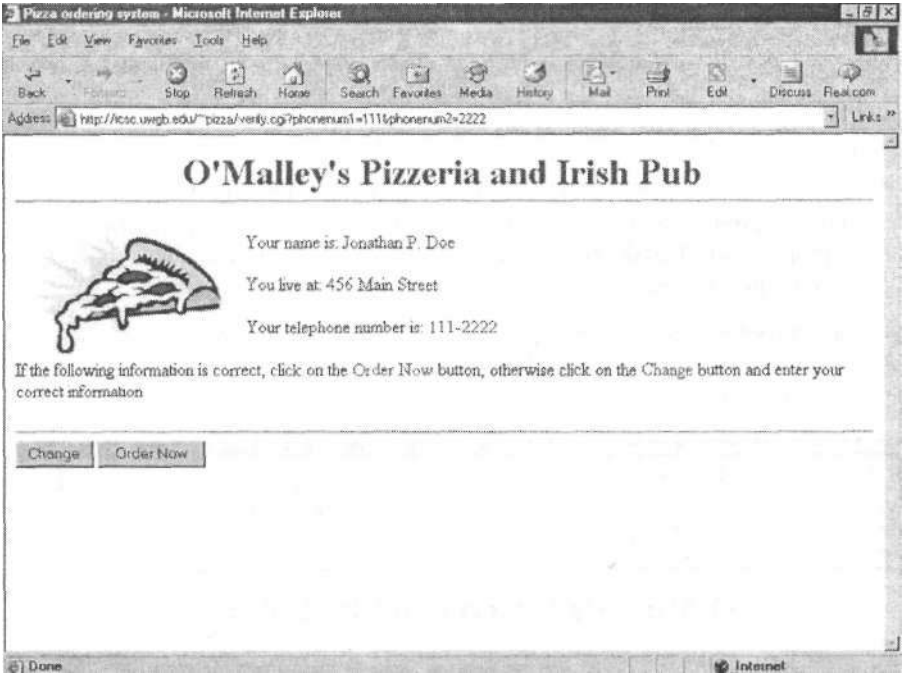
Kada se korisnik poveže na Web sajt 0'Malley's Pizzeria and Irish Pub, moguća je sledeća sekvenca interakcija:*

1. Korisnik unosi svoj broj telefona u formu (slika 12.19).
2. Korisnik klikne dugme Order. Perl skript na serveru traži broj telefona u fajlu. Ako skript ne pronađe broj, od korisnika se traži da ponovo unese svoj broj telefona. Ako se broj pronade, Perl skript kreira formu za verifikaciju korisnika, u kojoj prikazuje informacije o korisniku (slika 12.20).'
3. Kada se prikaže forma za verifikaciju, korisnik ima dve opcije. Može da promeni svoje informacije klikom na dugme Change. Ovo daje formu sa slike 12.21, u kojoj korisnik može da promeni svoje ime, ili adresu.

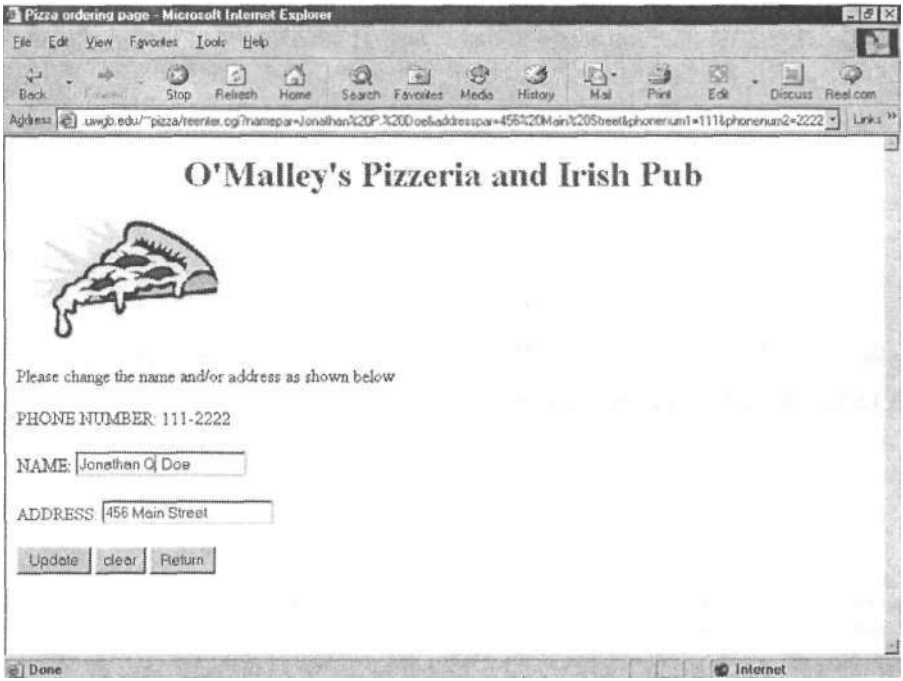
SLIKA 12.19 *Ulazna forma sa brojem telefona*

* Čitaoci mogu da pokrenu ovaj primer i da pristupe kodu prcko Web sajta ove knjige na WWW.uWgb.edu/shayW/udcn3. Perl skriptovi se pokreću na Linux serveru (Red Hat Linux verzija 7.3).

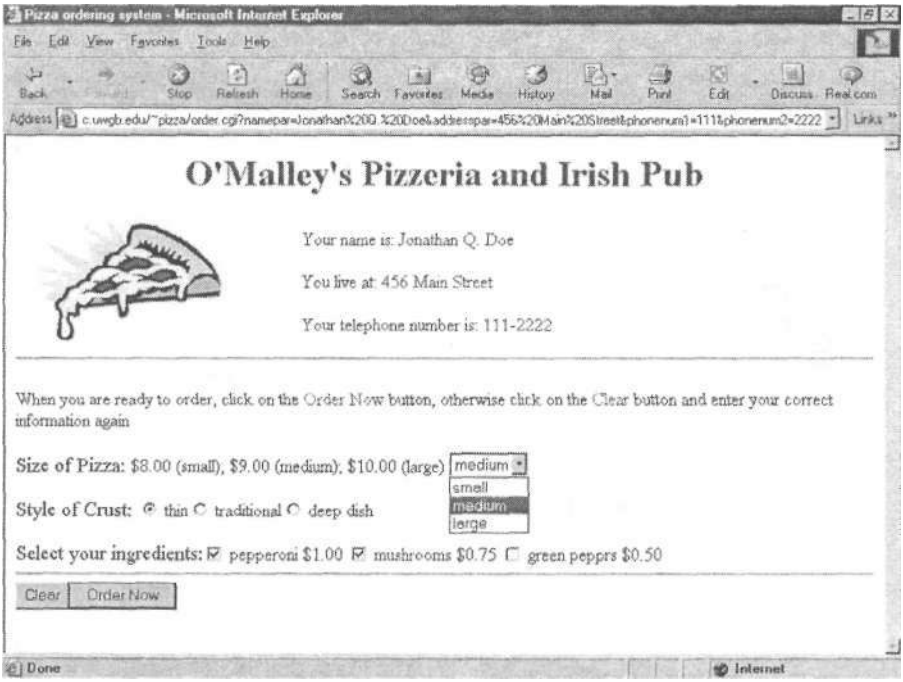
[Ako eksperimentišele sa ovim primerom, u fajlu se nalaze sledeći brojevi telefona: 111-2222, 333-4444, 555-6666 i 777-8888.



SLIKA 12.20 *Forma za verifikaciju korisnika*



SLIKA 12.21 *Forma za ažuriranje informacija o korisniku*



SLIKA 12.22 *Forma za narudžbine*

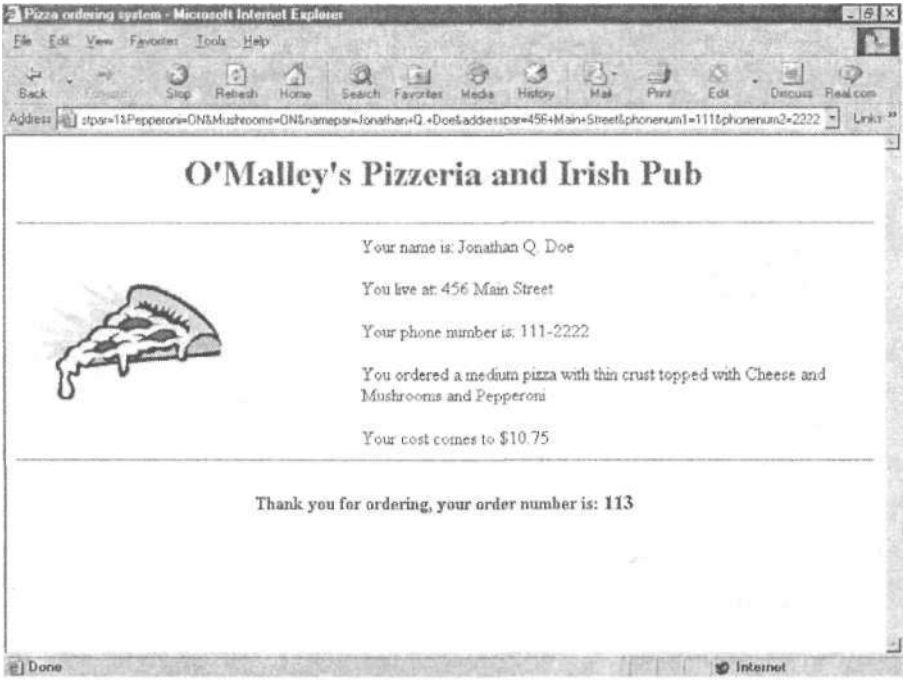
Nakon toga, može da klikne dugme Update kako bi promene bile zapamćene u fajlu na serveru*. Korisnik može da klikne i dugme Order NoW sa slike 12.20, koje generiše formu za naaidžbine sa slike 12.22.

4. Kada korisnik dode do forme za narudžbine, može da izabere veličinu pice, vrstu testa i nadeve. Nakon toga, može da klikne dugme Order NoW kako bi se narudžbina obradila.

5. Nakon što se narudžbina obradi, korisnik vidi formu sa pregledom informacija, prikazanu na slici 12.23, u kojoj su predstavljene informacije o korisniku i namčenoj pici. Osim toga, prikazuje ukupnu cenu i redni broj narudžbine. Broj narudžbine se povećava za 1 nakon svake narudžbine.

Ove forme i njihovi skriptovi implementiraju osnovnu funkdonalnost; sigurno je da mogu još više da se kompletiraju funkcionalnosti i da se kreira robustan Web sajt. Vežbe na kraju ovog poglavlja koristite kao sugestije za neka poboljšanja, ali najpre treba da razumete kako skriptovi izvršavaju željene zadatke.

" Za one koji budu pristupali ovom Web sajtu napominjemo da smo onemogućili izvršavanje promena u fajlu.



SLIKA 12.23 *Forma za pregled informacija*

Verifikovanje broja telefona

Značajno je da u potpunosti razumete ranije opisane korisničke interakcije pre nego što predemo na Perl skriptove koji ih implementiraju. Ako to još uvek ne razumete, preporučujemo da pristupite ranije navedenom Web sajtu i eksperimentišite sa primerom. Pretpostavimo da korisnik unese broj telefona 111-2222 u formu sa slike 12.19. HTML dokument (index.html u direktorijumu 0'Malley picerije) koji kreira ovu formu sadrži elemente slične onima u formi iz prethodne sekcije, pa ih ovde ih nećemo ponavljati. Medutim, kada korisnik klikne dugme Order, forma poziva CGI skript pod nazivom verify.cgi preko URL-a sličnog sledećem:

http://icsc.uWgb.edu/~pizza/verify.cgi?phonenum1=111&phonenum2 = 2222

Značajne "stvari" su parovi naziv/vrednost koji predstavljaju podatke iz ulazne forme. Slika 12.24 sadrži Perl skript verifyxgi koji pristupa parovima naziv/vrednost i reaguje na njih. Obratite pažnju da postoje neke značajne, ali neznatne aktivnosti koje se ovde dešavaju; pažljivo pročitajte njihova objašnjenja. Perl skript se pokreće na serveru i pretražuje fajl na njemu kako bi utvrdio da li postoji broj telefona koji je korisnik uneo. On ujedno kreira HTML formu u zavisnosti od rezultata pretraživanja.

```
1     #!/usr/bin/perl -W
2     use CGI qW(:standard);
3     useCGI::Carp "fatalsToBroWser";
4     # Ovaj cgi skript vrsi validaciju podataka poslatih serveru i kreira forme,
5     # u zavisnosti od prenetih podataka., za klijentovog korisnika
6
7     Sphonel = param("phonenuml");
8     $phone2 = param("phonenum2");
9     $phone = $phone1.$phone2;
18    $address = "";
11    $name = "";
12
13    print header, start_html("Pizza ordering system"),
14      "<h1 align=center> <Font color=RED> 0'Malley's Pizzeria and Irish Pub", hr,
15      "</font></h1><img SRC=\"pizza.jpg\" align=left Width=200 depth=200></img>";
16
17    print «END_of__text;
18
19    <script language="JavaScript">
20
21
22    // Poziva cgi skript za verifikaciju validnosti brojeva
23    // telefona unetih na formi.
24
25      function reverify(form)
26      {
27       var url; // lokacija cgi skripta za verifikovanje podataka
28
29       url = "verify.cgi?phonenum1="+form.phone1.value+
30         "&phonenum2="+form.phone2.value;
31      WindoW.location.href=url;
32     }
33
34
35    // metod izvlaci informacije o korisniku iz forme
36    // i referencira url koji korisniku omogucava da unese nove
37    //  informacije.
38
39     function editForm(form)
40     {
41      var url; // url za cgi program za unosenje novih podataka
42
43      url = "reenter.cgi?namepar=" + form.name.value;
44      url = url + "&addresspar=" + form.address.value;
45      WindoW.loaction.href=url + "&phonenum1=&phone1&phonenum2=&phone2"
46     } // editForm
47
48
49    // metod izvlaci informacije o korisniku iz forme
50    // i referencira url koji korisniku omogucava da naruci picu
51
52    function placeOrder(form)
53    {
54      var url; // url za cgi program za postavljanje narudzbine
55
56      url = "order.cgi?namepar=" + form.name.value;
57      url = url + "&addresspar=" + form.address.value;
58      WindoW.loaction.href=url + "&phonenum1=&phone1&phonenum2=&phone2"
```

```
59      } //plaoeOrder
60
61    </script>
62    END_of_text
63
64    # Ovde pocinje izvrsenje
65    # Ako se podaoi nalaze u oba polja za broj telefona, trazi se broj telefona
66    if ($phone1 && $phone2)
67    {
68    search();
69    }
70    else # jedno, ili oba polja za broj telefona su prazna
71    {
72    createInputForm('Please enter all required fields.[1]');
73    }
74
yc      Jj.....
76    # Poziva se kada korisnik nije uneo brojeve u oba
77    # polja za broj telefona, ili ako uneti broj telefona
78    # ne odgovara ni jednom broju iz baze podataka. Ovaj metod kreira formu
79    # sa porukom koja ukazuje na slucaj koji se desio, prikazujuci
80    # zvezdicu pored svakog polja koje je ostavljeno prazno.
81    # Forma takodje omogucava klijentovom korisniku da ponovo unese broj telefona.
QO     #*****************************************************************/
83    sub createlnputForm
84    {
85      print '<form inethod="get" ACTION="index.html">',
86      p("$_[0] Enter your phone number beloW"),
87       'PHONE NUMBER:';
88      if($phone1 == "")
89      {
90     print font( {color=>RED}, " * " ) ;
91      }
92      print "<input NAME='phonel' VALUE='$phone1' size=3 maxlength=3>-";
93      if($phone2 == "")
94      {
95     print font( {color=>RED}, " * " ) ;
96      }
97      print «END_of_text;
98
99      <input NAME='phone2' VALUE='$phone2' size=4 maxlength=4><p>
100     <input type="button" value="Order" onclick="reverify(this.form)">
101     <input type="subrait" value="Reset">
102     end_form()
103     </body>
104     </html>
105
106   END_of_text
107   } # createlnputForm
108
10g    w*********************************************************
110   # Pretrazivanje fajla, trazenje poklapanja sa brojem
111   # telefona.
112    #*********************************************************/
113   sub search
114   {
115     $found = 0;
116
```

```
117  open(CUSTFILE, "<cutomers.txt");
118  @data = <CUSTFILE>;
119  $i = 0;
120  While ((Si<$#data) && ($found==0))
121  {
122   if($phone == $data[$i]
123   {
124    $name = Sdata[$i+1];
125    Saddress = $data[$i+2];
126    $found = 1;
127   } #if
128    $i=$i+3
129   } # za petlju
130   close (CUSTFILE);
131
132   if($found==0)
133   {
134   createInputForm('The number you entered Was not found in the database.');
135   }
136   Else
137   {
138   createSummaryForm();
139   }
140  } # search
141
142    a**************************************************************************
143  # Ovaj metod kreira formu, koristeci korisnikove informacije koje
144  # su pronadjene u fajlu na serveru. Forma omogucava klijentovom
145  # korisniku da selektuje dugme za promenu svojih informaci]a, ili
146  # drugo dugme za nastavak postavljanja narudzbine.
147    a**************************************************************************/
148  sub createSummaryForm
149  {
150    print «END__of_text;
151
152    <p> Your name is: <font color=BLUE> Sname </font>
153    <p> You live at: <font color=BLUE> Saddress </font>
154    <p> Your telephone number is: <font color=BLUE> Sphonel-Sphone2 </font>
155    <FORM method="get">
156    <p><p><p> If the followIng information is correct, click on the
157    <font color=RED> Order NoW </font> button, otherWise click
158    on the <font color=RED> Change </font> button and enter
159    your correct information <p><hr>
160    <input type=hidden NAME=name value="$name">
161    <input type=hidden NAME=address value="$address">
162    <input type=hidden NAME=phone1 value="$phone1">
163    <input type=hidden NAME=phone2 value="$phone2">
164    <input type=button value="Change" onclick=editForm(this.form)>
165    <input type=button value="Grder NoW" onclick=placeOrder(this.form)>
166    </form>
167    </body>
168    </html>
169  END_of_text
170  } # createSummaryForm
```

SLIKA 12.24 *CGI skript (verify.cgi) za verifikovanje podataka sa forme*

Ta forma sadrži JavaScript (JS) skript koji se pokreće na klijentu kada korisnik klikne drugo dugme. Dakle, Perl skript izvršava ne samo pretraživanje fajla na serveru, već kreira forme koje se prikazuju u korisnikovom pretraživaču. Osim toga, kreira JS skript koji pretraživač izvršava kao odziv na korisnikove akcije. Nepovoljno je to što se koristi skript (Perl) za generisanje drugog skripta (JavaScript) koji će biti daljinski izvršen. Važno je da se prati koji se skript gde izvršava.

Linije 1 do 3 su tipični Perl CGI skriptovi. Linija 1 definiše lokaciju Perl interpretatora, a linije 2 i 3 su slične naredbama #include iz programskog jezika C. One Perl skriptu omogućavaju pristup odredenim promenljivim i funkcijama koje olakšavaju CGI programiranje i pomažu rešavanje problema ako se Perl skript prekine zbog bilo kog razloga. Za naše potrebe samo ih uključujemo, ali pogledajte reference za više detalja o tome. Linije 7 do Il deklarišu naše Perl promenljive. Linije 7 do 8 obezbeduju inicijalne vrednosti uzete iz ranije prikazanog stringa upita, linija 9 nadovezuje inicijalne vrednosti u jedan string, a linije 10 do Il inicijalizuju sve stringove adresa i imena na prazne stringove.

Linije 13 do 62 kreiraju HTML i JS skript koji će biti poslat nazad do korisnikovog pretraživača. One definišu šta će korisnik videti i kako će pretraživač reagovati na klik na dugme. Perl kreira HTML dokumente slično kao što to rade C programi iz prethodne sekcije - koristi komandu `print`. Ako se ovaj skript pokrene iz Linuxove komandne linije, komande `print` će prikazati podatke na ekranu. Pokretanjem iz HTML forme izlaz iz naredbi `print` ide nazad do pretraživača. Nakon toga, pretraživač interpretira izlaz na isti način kao što interpretira bilo koji HTML dokument.

Linija 13 referencira makroe koji se razvijaju u HTML naredbe. Na primer, komanda `print header` daje

Content-Type: text/html

A `print startjrtml ("Pizza order system")` daje

<HTML> <HEAD> <TlTLE>Pizza ordering

system</TiTLE> </HEAD> <BODY>

Linija 13 štampa oboje sa jednorn naredbom `print`. Ideja je da se grupišu neki uobičajeni HTML tagovi i da se defmiše makro za njih. Ovim se redukuje količina linija koje je neophodno otkucati. Linije 14 i 15 daju više HTML tagova.

Linija 17 se koristi kada Perl skript mora da odštampa veći broj linija u HTML dokumentu. Umesto da postoji komanda `print` u svim linijama, ta linija ukazuje Perl interpretatoru da treba da ih tretira kao sukcesivne linije izlaza sve dok ne dode do one u kojoj postoji indikator END_of_text (Hnija 62). Napomenimo da Perl zahteva da linija nakon komande print bude prazna i indikator END_of_text mora da počne u prvoj koloni.

Sve od linije 19 do 61 je JavaScript i daje izlaz za HTML dokument koji Perl skript kreira. Postoje tri JS funkcije, koje odgovaraju pojedinačnim hendlerima događaja OnClick. U svakom slučaju, JS funkcija formira IJRL u kome se nalazi naziv CGI skripta, iza koga sledi string upita sa parovima naziv/vrednost generisanim na osnovu podataka sa forme. Nakon toga, pretraživač poziva taj CGI skript i prosleduje string upita kada korisnik klikne dugme pridruženo hendleru dogadaja. llskoro ćete videti i primer.

Perl skript započinje analiziranje informacija izvučenih iz stringa upita

Phonenuml = 111&phonenum2=2222

u liniji 66, Pretpostavimo najpre da korisnik nije uneo informacije u jedno, ili u oba polja sa slike 12.19. U tom slučaju jeđna, ili obe promenljive $phone1 i $phone2 su prazni stringovi i naredba if u liniji 66 vraća FALSE. Tada skript poziva Perl funkciju za kreiranje nove ulazne forme (linije 83-107). Ta funkcija većim delom sadrži naredbe `print` za kreiranje HTML dokumenta u kome se nalazi forma skoro identična onoj sa slike 12.19. Medutim, postoji jedna razlika.

Tipičan odziv mnogih skriptova u situacijama kada korisnik ne popuni sva tekstualna polja na formi je obezbedivanje neke vrste oznake pored polja koje korisnik nije popunio. U ovom slučaju, ako je promenljiva Sphonel bila prazan string (linija 88), skript će na formi odštampati crvenu zvezdicu pored prvog polja za broj telefona (linije 90-92). Skript nastavlja na sličan način i za promenljivu $phone2 (linije 93-96). Kada se ova funkcija završi, Perl skript je kompletiran i u korisnikovom pretraživaču se prikazuje nova forma. Korisnik dobija novu šansu da unese ispravan broj telefona. Ako klikne dugme Order (definisano u liniji 100), poziva se JS funkcija `reverify` (linije 25-32); sada se vracamo na stranu klijenta. Ova funkcija formira LlRL koji poziva skript pod nazivom verify.cgi i prosleduje vrednosti Sphonel i $phone2. Pošto je skript sa slike 12.24 u stvari skript `verify.cgi`, server iznova startuje ceo proces.

Pretpostavimo sada da je korisnik uneo broj telefona u formu sa slike 12.19. Naredba if iz linije 66 vraća TRLIE i skript poziva Perl funkciju search za utvrdivanje da li broj telefona postoji u fajlu na serveru (linije 113-140). Linija 117 otvara fajl pod nazivom `customers.txt`. Karakter "<" pre naziva fajla ukazuje da se fajl otvara radi čitanja. Linija 118 učitava ceo fajl u niz stringova*, gde svaka linija odgovara jednom stringu u nizu. Ako je fajl veoma veliki, ovo može da stvori neke probleme, ali mi smo za naš primer koristili kraći fajl i možemo bez problema da iskoristimo ovaj pristup. Takode smo redukovali količinu predstavljenih detalja, zato što smo više zainteresovani za to kako Perl reaguje na klijentove zahteve. U ovom slučaju svaki element niza Ždata sadrži jednu liniju iz fajla. U ovom primeru u prve tri linije se nalaze broj telefona, ime korisnika i adresa korisnika. Sledeće tri linije sadrže iste informacije za drugog korisnika. Svaki skup od tri linije sadrži broj telefona, ime i adresu sledećeg korisnika.

Zato petlja definisana linijama 119 do 129 poredi $data[0], $data[3], $data[6] i tako dalje sa unetim brojem telefona, tražeći poklapanje. Ako se pronade poklapanje, linije 124 i 125 definišu ime i adresu korisnika, a linija 126 postavlja promenljivu Sfound na I. Time se usloviz linije 120 postavlja na FALSE i petlja se zaustavlja.

* Simbol @ iz linije 118 je Perl oznaka promenljive niza. Individualni elemenli niza predstavljaju se pomocu simbola S.

Ako skript ne pronade poklapanje, onda se subskript i inkrementira na veću vrednost od veličine niza naznačene promenljivom $#data iz linije 120. U ovom slučaju uslov iz linije 120 postaje FALSE.

Kada se petlja završi, fajl se zatvara. Sledeći korak zavisi od toga da li je broj telefona pronađen. Ako nije pronađen, skript poziva Perl funkciju iz linije 134 koja kreira novu ulaznu formu. Ovo je ista akcija koja se preduzima ako je korisnik ostavio neko tekstualno polje sa slike 12.19 prazno. Ako je broj pronaden, skript poziva Perl funkciju za kreiranje forme za rezimiranje (linije 148-170) slične onoj sa slike 12.20.

Ažuriranje informacija o korisniku

Nakon prikazivanja forme sa slike 12.20, korisnik ima dve opcije: ažuriranje svojih informacija, ili postavljanje narudžbine. Ako korisnik klikne dugme Change (definisano u liniji 164 na slici 12.24), pretraživač poziva JS funkciju editForm (linije 39-46 na slici 12.24). Ta funkcija formira URL string u kome se nalazi CGI skript pod nazivom reenter.cgi. URL je sličan sledećem:

http://icsc.uWgb.edu/~pizza/reenter.cgi7namepar= Jonathan%20P. %20Doe&
addresspar= 456%20Main%20Street&phonenuml=l 1 l&phonenum2=2222

Perl skript (reenter.cgi) kreira formu sa slike 12.21 i šalje je nazad do korisnikovog pretraživača. Na slici 12.25 prikazan je sadržaj skripta reenter.cgi. Ovo je prilično složen skript koji izvlači podatke o korisniku iz stringa upita (linije 3-6) i prikazuje ih. Takođe defmiše JS funkciju za svako dugme. Dugme Clear (linija 51) odgovara funkciji clearForm (linije 14-18), koja briše tekstualna polja. Dugme Return (linija 52) odgovara funkciji goback (linije 20-26), koja formira URL koji referencira skript verify.cgi.

Međutim, pretpostavimo da korisnik promeni tekstualna polja i da klikne dugme Update (linija 50), čime se aktivira JS funkcija update (linije 28-34) i formira se URL sličan sledećem

http://icsc.uWgb.edu/-pizza/update.cgi?namepar=Jonathan%20Q.%20Doe&
addresspar= 456%20Main%20Street«&phonenuml=lll&phonenum2=2222

Na slici 12.26 prikazan je CGI skript update.cgi. On poziva dve Perl funkcije iz linija 11 i 12. Funkcija makeform (linije 14-36) kreira jednostavnu formu (nije prikazana) koja samo ističe da su izvršene promene i uključuje dugme OK koje, u stvari, ponovo prikazuje formu sa slike 12.20 u korisnikovom pretraživaču. Funl«dja update (linije 38-60) ažurira informacije o korisniku u tekstualnom fajlu na serveru. Učitava sadržaj fajla u niz stringova, locira odgovarajućeg korisnika, izvodi promene i upisuje rezultate nazad u fajl. Sofisticiraniji pristup podrazumeva rad sa bazom podataka, ali to bi većzahtevalo opširnija objašnjenja.

Linije 40-43 inicijalizuju promenljive, otvaraju fajl i učitavaju njegov sadržaj u niz stringova. Opcija die u liniji 41 omogućava prikazivanje definisane poruke u pretraživaču ako fajl ne može da se otvori. Pretraživanje niza stringova je slično ranije opisanom pretraživanju. Razlika je u tome što se podaci pronadeni na lokacijama $data[$i], onda $data[$i+1] i $data[$i+2] menjaju u skladu sa novim imenom i adresom iz stringa upita.

```
#!/usr/bin/perl -W
use CGI qW(:standard);
my $name = param("namepar");
my Saddress = param("addresspar");
my $phone1 = param("phonenuml");
my $phone2 = param("phonenum2");

print header, start_html("Pizza ordering page"),
print "<h1 align=center> <Font color=RED> 0'Malley's Pizzeria and Irish Pub";
print "</font></h1><img SRC=N"pizza.jpgV' align=left Width=200 depth=200>";
print «END_of_text;

<script language="JavaScript">
function clearForm(form)
{
  form.name.value="";
  fomr.address.value="";
}

function goback(form)
{
  var url;
  url = "verify.cgi?namepar="+ form.name.value;
  url = url + "Saddresspar=" + form.address.value;
   + "&phonenum1=&phone1&phonenum2=&phone2"
}

function update(form)
{
  var url;
  url = "update.cgi?namepar="+ form.name.value;
  url = url + "Saddresspar=" + form.address.value;
  WindoW.loaction.href=url + "&phonenum1=&phone1&phonenum2=&phone2"
}

</script>
END_of_text

CreateEditForm();

sub createEditForm
{
  print «END_of_text;

  <form method="get" ACTION="update.cgi">
  <p>Please change the name and/or address as shoWn beloW </p>
  <p>PH0NE NUMBER: $phone1-$phone2
  <p>NAME: <input NAME="name" VALUE="Sname" </p>
  <p>ADDRESS: <input NAME="address" VALUE="$address" size="20"> </p>
  <p><input TYPE="button" Value="Update" onclick="update(this.form)">
  <input TYPE="button" Value="clear" onclick="clearForm(this.form)">
  <input TYPE="button" Value="Return" onclick="goback(this.form)">
END_of_text
print end_form();
}
```

SLIKA 12.25 *CGl skript (reenter.cgi) koji kreira formu sa slike 12.21*

```
#!/usr/bin/perl -W
use CGI qW(:standard);
use CGI::Carp "fatalsToBroWser";

my $name = param("namepar");
my Saddress = param( "addresspar");
my $phone1 = param("phonenuml");
my $phone2 = paramf"phonenum2");


update();
makeform();

sub makeform
{
  print header, start_html("Pizza ordering system"),
  "<h1 align=CENTER> <font color=RED> 0'Malley's Pizzeria and Irish Pub";
  "</font></h1><hr>";
  "<form method='get' ACTION='verify.cgi'>",
  "<input type='hidden' NAME='namepar' value=$name>",
  "<input type='hidden ' NAME='addresspar' value=$address>",
  "<input type='hidden' NAME='phonenuml' value=$phone1>",
  "<input type='hidden' NAME='ptionenum2' value=$phone2>",
  "<table border=0> <tr>",
  "<td>",
  "</fontx/h1><img SRC=\"pizza.jpg\" align=CENTER Width=200 depth=200>";
  "</td>",
  "<td>",
  "Information has been updated. Click on OK to return",
  "<input type='submit' value='0K'>",
  "</td>",
  end_form();
  "</body>",
  "</html>",
  end_html;
}

sub update
{
  Sfound = 0;
  open (CUSTFILE, "<customer.txt") |j die "Cannot read from file\n";
  @data » <CUSTFILE>;
  $i = 0;
  While (($i<$#data) && ($found==0))
  {
   if($phone == $data[$i]
   {
   $data[$i+1]=$name."\n";
   $data[$i+2]=$addness."\n";
   $found = 1;
  } # if
  $i=$i+3;
  } # za petlju
  close (CUSTFILE);
# obezbedjivanje da fajl ima pravo upisa, koristi se komanda chmod ako je potrebno
  open (CUSTFILE, ">customers.txt") || die "cannot open file for Write\n;
  flock(CUSTFILE, 2);
  print CUSTFILE ©data;
  close (CUSTFILE);
}
```

SLIKA 12.26 *CCl skript (update.cgi) za ažuriranje fajla o korisnicima*

Sledeći korak je smeštanje novih podataka u fajl. Da bi se to uradilo, skript zatvara fajl (linija 54) i ponovo ga otvara (linija 56). Medulim, ovoga puta prisustvo karaktera ″>″ ukazuje da je fajl olvoren radi upisa. Linija 57 zahteva vrši zaključavanje fajla. Ovo znači da dok je tekuči skript zaključan, ni jedan drugi skript neće moći da dobije pravo na zaključavanje. Zaključavanje fajla sprečava dva korisnika da istovremeno pokušaju da ažuriraju fajl. Preostale dve linije upisuju niz stringova nazad u faj] i zatvaraju ga.

Postavljanje narudžbine

Uz pretpostavku da je korisnik zadovoljan informacijama sa slike 12.20, može da postavi konačnu narudžbinu klikom na dugme Order NoW na toj formi. Na taj način, poziva CGI skript pod nazivom order.cgi preko URL-a sličnog sledećem:

> http://icsc.uWgb.edu/~pJzza/order.cgi?namepar=Jonathan%20Q.%20Doe&
> addresspar=456%20Main%20Street&phonenuml=l 1 l&phonenum2=2222

Ovaj skript (slika 12.27) pristupa informacijama o korisniku iz stringa upita (linije 5-8) i poziva Peri funkciju (linije 20-40) za prikazivanje forme sa slike 12.22. Nakon toga, prikazuje opcije za naručivanje u vidu pop-up menija (linije 56-57), radio dugmadi (linije 58-61) i polja za potvrdu (linije 62-68). Forma definiše i CGI skript checkout.cgi (linija 49) za akciju koja se preduzima kada korisnik klilcne dusme Order NoW.

```
1    #!/usr/bin/perl -W
2    use CGI qW(:standard) ;
3    use CGI::Carp "fatalsToBroWser";
4
5    $name = param("namepar");
6    Saddress = param("addresspar");
7    $phone1 = param("phonenuml");
8    $phone2 = param("phonenum2");
9
10   displayCustomerData();
11   displayOrderForm();
12
13   print end_html;
14
HC     JJ.....
16   # Prikazivanje forme sa podacima o korisniku koja omogucava
17   # proveru korisnikovog imena, adrese i broja
18   # telefona
IQ     JJ.....
20   sub displayCustomerData
21   {
22     print header, start_html("Pizza ordering system");
23     print «END_of_text;
24
```

```
25      <h1 align=CENTER> <font color=RED> 0'Malley's Pizzeria and Irish PUb
26      </font></h1>
27    <table border=0 Width=100%> <tr>
28      <td Width=33%>
29      </font></h1><img SRC=\"pizza.jpg\" align=CENTER Width=200 depth=200>
30      </td>
31      <td Width=67%>
32      <p> Your name is: <font color=BLUE> $name </font>
33      <p> You live at: <font color=BLUE> $address </font>
34      <p> Your telephone number is: <font color-=BLUE> $phone1-$phone2 </font>
35
36      </td> </tr> </table>
37
38      <hr>
39    END_of_text;
40    }
41
A^      JJ.....
43    # Prikazivanje forme za narudzbine koja korisnku omogucava da
44    # izabere velicinu pice, testo i nadeve i da preda
45    # narudzbinu
46      jj.....
47    sub displayOrderForm
48    {
49      print  ('<FORM method="get" ACTION='checkout.cgi">';
50      print p, "When you are ready to order, click on the",
51        font( {color=>RED}, "Order NoW"), "button, otherWise click",
52        "on the", font( {color=>RED}, "Clear"), "button and enter",
53        "your correct information again", p;
54      print start_form();
55      print p, font( {size=>4}1 "Size of pizza: " ) ;
56      print "\$8.00 (small); \$9.00 (medium); \$10.00 (large)",
57        popup_menu (sizepar,('small', 'medium', 'large']),
58      print p, font( {size=>4}, "Style of Crust: " ) ;
59      print "<input Type='radio' Name='crustpar' value='1' checked> thin";
60      print "<input Type='radio' Name='orustpar' value='2' > tr-aditional";
61      print "<input Type='radio' Name='crustpar' value='3' > deep dish<P>";
62      print p, font( {size=>4}, "Select your ingredients: " ) ;
63      print "<input Type='checkbox' Name='Pepperoni' value='ON'> pepperoni";
64      print "\$1.00";
65      print "<input Type='checkbox' Name='Mushrooms' value='ON'> mushrooms";
66      print "\$0.75";
67      print "<input Type='checkbox' Name='Green peppers' value='ON'> green peppers";
68      print "\$0.50";
69      print hr;
70      print "<input type=\"hidden\" NAME=namepar value=\"$name\">";
71      print "<input type=\"hidden\" NAME=addresspan value=\"$address\">";
72      print "<input type=\"hidden\" NAME=phonenum1 value=\"$phone1\">";
73      print "<input type=\"hidden\" NAME=phonenum2 value=\"$phone2\ ">";
74      print "<input type="reset" value="Clear">');
75      print "<input type="submit" value="Order NoW">');
76      print end_form();
77    }
```

SLIKA 12.27 *CCI skript (order.cgi) za naručivanje pice*

Verifikovanje narudžbine

Pretpostavimo da korisnik izabere veličinu pice, vrstu testa i sastojke (slika 12.22), pa klikne dugme Order NoW. Pretraživač poziva skript checkout.cgi preko URL-a sličnog sledećem:

http://icsc.uWgb.edu/~pizza/checkout.cgi?sizepar=medium&crustpar=l&
Pepperoni=ON&Mushrooms=ON<&namepar=Jonathan+Q.+Doe&
addresspar=456+Main+Street&phonenuml=lll&tphonenum2=2222

Primećujete da, osim uobičajenih informacija o korisniku, string upita sadrži i

sizepar=medium&crustpar= 1 &Pepperoni=ON&Mushrooms=ON

čime se definiše šta je korisnik selektovao na slici 12.22. Skript checkout.cgi sa slike 12.28 pristupa ovim informacijama, izračunava cenu pice i kreira formu sličnu onoj na slici 12.23 kako bi bila potvrđena korisnikova narudžbina i prikazana cena.

Linije 4 do 13 izvlače informacije iz stringa upita. Perl skript generiše redni broj narudžbine sa slike 12.23, koristeći brojač koji se pamti u tekstualnom fajlu na serveru. Koristi se isti koncept kao i na mnogim Web sajtovima kada vidite poruku kao što je

You are visitor number ######

koja prati broj posetilaca sajta u jednom tekstualnom fajlu. Funkcija update-Counter, definisana u linijama 25 do 35, pokazuje kako se to izvodi. Linije 27 do 30 otvaraju fajl radi čitanja i upisa, zaključava fajl (kako bi se spredlo postavljanje namdžbina sa istim brojem od dva korisnika), učitava redni broj narudžbine u promenljivu $ordernumber i uvećava njenu vrednost za 1. Preostale linije repozicioniraju pokazivač fajla na početak fajla (tako da se novi podaci ne dodaju na postojeće), upisuju novi brojač u fajl i zatvaraju fajl.

Linije 42 do 107 Perl funkcije utvrduju šta je korisnik naručio, izračunavaju cenu i prikazuju informacije za korisnika u formi za rezimiranje sa slike 12.23. Prvo što funkcija obavlja nakon generisanja nekog izlaza je utvrđivanje vrste testa (linije 59-70). Pošto to ne utiče na cenu, sve što skript treba da uradi je da definiše string promenljivu $Style koja se koristi za izlaz forme. Ova promenljiva zavisi od Scrust, vrednosti koja je uzeta iz stringa upita u liniji 8. Pošto korisnik bira vrstu testa selektovanjem jednog radio dugmeta, vrednost `crustpar` u stringu upita je 1, 2, ili 3, u zavisnosti od toga koje je radio dugme korisnik selektovao. U ovom slučaju korisnik bira tanko testo (prvo radio dugme) i prethodno navedeni string upita sadrži `"crustpar=1"`. Biranjem opcije traditional, ili deep dish, vrednost `crustpar` bi bila 2, ili 3.

Nakon toga, skript mora da utvrdi koje je veličine pica koju je korisnik naručio. Promenljiva `$size` je utvrdena iz stringa upita iz linije 9. Njena vrednost je jednaka stringu u pop-up meniju koji je korisnik izabrao. U našem slučaju korisnik je hteo picu srednje veličine (`medium`), tako da je vrednost pridružena sa `sizepar` i, samim tim, `$size` je `"medium"`. Linije 71 do 82 izračunavaju osnovnu cenu na osnovu izabrane veličine.

Konačno, skript mora da utvrdi nadeve. U ovom slučaju oni su predstavljeni poljima za potvrdu i njihovi pridruženi parametri imaju vrednost ON, ili OFF.

```
1     #!/usr/bin/perl -W
2     use CGI qW(:standard);
3
4     $name = param("namepar");
5     $address = param("addresspar");
6     Sphonel = param("phonenuml");
7     $phone2 = param("phonenum2") ;
8     Scrust = param("crustpar");
9     $size = param("sizepar");
10
11    $topping1 = paramf"Mushrooms");
12    $topping2 = param("Green peppers");
13    $topping3 = param("Pepperoni");
14
15    $cost = 0;
16    $ordernumber = 0;
17
18    updateCounter();
19    displayCustomerOrder();
20
Q1     w**************************************************************************
22    # Metod azurira i prikazuje brojac koji definise                        «•
23    # broj posetilaca koji su narucivali pice

25    sub updateCounter
26    {
27      open (FILE, "+<counter.txt")| jdie "Cannot read from the counter file.\n";
28      flock (FILE, 2);
29      Sordernumber=<FILE>;
30      $ordernumber=$ordernumber+1;
31      seek (FILE, 0, 0);
32      print FILE Sordernumber;
33      flock (FILE, 8);
34      close (FILE)
35    } #updateCounter
36


38    # Prikazivanje forme za narudzbine koja korisnku omogucava da
39    # izabere velicinu pice, testo i nadeve i da preda
40    # narudzbinu
Aj     w**************************************************************************
42    sub displayCustomerCounter
43    {
44      print header, start_html("Pizza ordering system");
45      print «END_of_text;
46
47      <h1 align=CENTER> <font color=RED> 0'Malley's Pizzeria and Irish Pub
48      </font></h1><hr>
49      <table border=0> <tr>
50      <td Widtb=40%>
51      </font></h1><img SRC="pizza.jpg" align=CENTER Width=200 depth=200>
52      </td>
53      <td>
54      <p> Your name is: <font color=BLUE> $name </font>
55      <p> You live at: <font color=BLUE> $address </font>
56    <p> Your telephone number is: <font color=BLUE> $phone1-$phone2 </font>
57
58    END_of_text
```

```
59    if ($crust == 1)
60    {
61      $style = "thin";
62    }
63    if ($crust == 2)
64    {
65      Sstyle = "traditional";
66    }
67    if ($crust == 3)
68    {
69      Sstyle = "deep dish";
70    }
71    if ($size eq "small")
72    {
73      $cost = $cost + 8.00;
74    }
75    if (Ssize eq "medium")
76    {
77      Scost = Scost + 9.00;
78    }
79    if ($size eq "large")
80    {
81      Scost = Scost + 10.00;
82    }
83    print "<p> You ordered a <font color=BLUE> Ssize </font>
84         pizza With <font color=BLUE> $Style </font> crust
85         topped With Cheese";
86    if ($topping1 eq "ON")
87    {
88      print "and <font color=BLUE> Mushrooms </font>";
89      $cost = $cost + 0.75;
90    }
91    if ($topping2 eq "ON")
92    {
93      print "and <font color=BLUE> Green Pappers </font>";
94      $cost = Scost + 0.50;
95    }
96    if ($topping1 eq "ON")
97    {
98      print "and <font color=BLUE> Pepperoni </font>";
99      $cost = Scost + 1.00;
100   }
101
102   print p, "Your cost comes to \$$cost </td> </tr> </table>";
103   print end_html;
104   print hr, "<h4 align=CENTER>
105        Thank you for ordering, your order number is:
106        <font color = BLUE size = 4> Sordernumber </font> </h4>";
107   }
108
109
110
111
```

SLIKA !2.28 *Skript* checkout.cgi

Vrednost ON znači da je korisnik selektovao polje za potvrdu. U ovom slučaju korisnik je selektovao dva nadeva: pepperoni i mushrooms. Zbog toga, string upita sadrži komponentu

Pepperoni=ON&Mushrooms=ON

Skript izvlači vrednosti ON/OFF iz stringa upita u linijama 11 do 13 i ispituje te vrednosti u linijama 86 do 100. Llvek kada pronade vrednost ON, uvećava cenu i štampa (na formi) string koji predstavlja izabrani nadev.

Linije 102 do 106 završavaju prikaz predstavljanjem ukupne cene, broja narudžbine i poruke zahvalnosti, sa nadom da će korisnik i ubuduće naručivati.

12.6 Zaključak

Ovo pogiavlje se fokusirano na razvoj aplikacija koje se oslanjaju na Internet protokole. Sve su bile zasnovane na klijent/server modelu, ali u različitim kontekstima, tipu interakcije između klijenta i servera i korišćenim alatkama. Odeljak 12.2 je fokusiran na soket programiranje i razvoj programa na strani klijenta i servera koji omogućavaju transfer fajlova. Sledi pregled nekih značajnijih koncepata iz tog odeljka:

- Soket omogućava programu da se poveže na mrežu, radi slanja, ili prijema podataka.
- Klijent može da se poveže na program na serveru naznačavanjem njegove IP adrese i broja porta u okvim odgovarajućih soket komandi.
- Klijent i server su obično saglasni o formatu paketa za razmenu podataka. Kada se dogovore i soketi se uspostave, klijent i server treba samo da čitaju i upisuju na soket radi razmene podataka. Ipak, neophodno je voditi računa o tome da je, kada jedan čita, drugi nešto upisao (ili će upisati).

U ostatku poglavlja težište je stavljeno na Web okruženje. U odeljku 12.3 bilo je reči o Webu, HTTP-ju i načinu za kreiranje jednostavnih HTML dokumenata. Značajna tema iz perspektive umrežavanja je mogućnost korišćenja JavaScripta za programiranje na strani klijenta. Sledi pregled nekih značajnih karakteristika JavaScripta:

- Forma može da se koristi kao šablon za unošenje i prikazivanje informacija. Može da uključuje tekstualna polja, polja za potvrdu i druge tipove.
- Forma može da uključuje i dugmad koja iniciraju akcije kada korisnik klikne dugme,
- Dugmetu sa forme može da se pristupi pomoću JavaScript funkcije, tako da, kada korisnik klikne dugme, funkcija izvršava naredbe koje analiziraju podatke sa forme i generišu odgovarajući izlaz. Ovo se izvodi na strani klijenta i može da se koristi za pregled korisničkih podataka pre nego što se pošalju do servera.

Programiranje može da se izvede i na strani servera; postoji nekoliko mogućih opcija. Jedna je CGI (Common GateWay Interface) programiranje. Uključuje kreiranje URL-a koji naznačava izvršni fajl na serveru. URL sadrži i string upita koji se koristi za prenos informacija sa forme do pozvanog CGI programa. CGI program izvlači informacije iz stringa upita i može da izvrši neophodne zadatke. Sledi lista značajnijih karakteristika CGI programa:

- Mogu se pisati u C-u, ili Perlu. C programi su kompajlirani, dok se Perl programi obično interpretiraju.
- Mogu da kreiraju forme koje se prikazuju u korisnikovom pretraživaču.
- Mogu da se koriste za implementiranje pretraživačkih mašina. U odeljku 12.4 predstavili smo primitivnu pretraživačku mašinu koja je pokazala kako C program može da prihvati reč, ili frazu sa forme i da vrati jedan, ili više linkova u klijentov pretraživač.
- Mogu da se koriste za namčivanje preko Interneta. U odeljku 12.5 objašnjen je sistem za naručivanje pica koji sadrži nekoliko Perl skriptova - oni verifikuju broj telefona, ažuriraju informacije o korisniku, omogućavaju naručivanje pice i daju pregled informacija za konkretnu narudžbinu.

Pitanja i zadaci za proveru

1. Zašto se koristi klijent/server model umesto da se programiranje izvede samo na jednom kompjuteru?
2. Šta je soket?
3. Koja je svrha soket komande bind? Šta bi se desilo kada ne bi bila izvršena?
4. Da li klijent, ili server, ili oba obično izvršavaju svaku od skdećih soket komandi?
   a. `socket()`
   b. `connectf)`
   c. `bind()`
   d. `accept()`
   e. `listen()`
   f. `send()`
   g. `recv()`
   h. `close()`
5. Program na serveru iz odeljka 12.2 sadrži komandu f ork() za kreiranje procesa koji upravlja zahtevom klijenta. Zašto se za to ne bi pobrinuo proces koji je i primio zahtev?
6. Šta je Hypertext Markup Language?
7. Šta je Hypertext Transfer Protocol?
8. Šta je Uniform Resource Locator?
9. Šta je HTML tag?
10. Koja je svrha HTML forme?

11. Šta je skript?
12. Sta je hendler dogadaja u kontekstu HTML forme i dugmeta?
13. Šta je CGI skript?
14. Sta je string upita koji se koristi prilikom CGI programiranja?
15. Šta je primenljiva okruženja?
16. Koji je razlog postojanja programskih mogaćnosti i na strani klijenta i na strani servera kod razvoja Web stranice?

Vežbe

1. Modifikujte protokol za transfer fajla iz odeljka 12.2 implementiranjem mehanizma za detekciju grešaka, kao što su čeksume. Greške možete da simulirate tako što će na serveru periodično biti oštećen paket neposredno pre slanja preko soketa. Na primer, može da se pozove generator slučajnih brojeva i, na osnovu vraćenog broja, mogu da se nuliraju bajtovi paketa. Ako Vam se više dopadaju greške nasumičnog izgleda, server može da dodaje nasumično izabrani broj nasumično izabranom bajtu u paketu.
2. Koristite sokete za implementiranje protokola za komunikaciju izmedu klient i server programa. Protokol dopušta komunikacije u balf-duplex modu kod kojih klijent i server razmenjuju poruke definisane onim što korisnik unese preko tastature. Morate da osmislite ko započinje "razgovor" i kako se konverzacija završava.
3. Kreirajte tekstualni fajl na serveru u kome svaka linija ima ključnu reč iza koje sledi URL. Napišite klijentski program koji prihvata ključnu reć kao ulaz od korisnika i šalje je do servera. Napišite serverski program koji uzima ključnu reč od klijenta, pretražuje fajl kako bi pronašao sve instance te ključne reči i vraća sve URL-ove pridružene toj ključnoj reči. Klijent treba da prikaže URL-ove.
4. Modifikujte protokol iz odeljka 12.2 dodavanjem svih sledećih mogućnosti, ili bilo koje od njih.
    a. Klijent treba da traži od korisnika da unese naziv fajla na udaljenom hostu. Nakon toga, klijent šalje taj naziv do servera pomoću soket konekcije.
    b. Kreirajte Disconnect paket i neka kijjent i server prekidaju konekciju nakon što klijent pošalje serveru Disconnect paket i server ga potvrdi.
    c. Kreirajte opciju menija na strani klijenta koja korisniku omogućava da izabere da li klijent treba da zahteva transfer fajla, ili protokol za konverzaciju kao u Vežbi 2.
    d. Neka i klijent i server konkurentno šalju fajlove jedan drugome.
    e. Promenite server tako da primi jedan paket nakon prihvatanja konekcije. Paket će defmisati da li server izvršava transfer fajla, ili protokol za konverzaciju sa klijentom. Naravno, klijent mora da pošalje takav paket.
    f. Dizajnirajte server da može da prenosi fajl do jednog klijenta dok istovremeno razgovara sa drugim. Moraćete da kreirate "decu" procese koji će reagovati na zahteve.

5. Ako već niste, kreirajte Web stranicu na kojoj će se nalaziti bar neki tekst i linkovi ka udaljenim URL-ovima. Sami odlučite šta ćete postaviti na stranicu.

6. Modifikujte JavaScript kod sa slike 12.12 tako da korisnik rnože da umetne proizvoljan broj blanko znakova izmedu brojeva koje unese.

7. HTML formu koja korisniku omogućava da unese dimenzije četvorougla, a zatim koristi skript za vraćanje oblasti četvorougla. Skript treba da utvrdi da li su dimenzije pozilivne; ako nisu, treba da generiše okvir upozorenja.

8. Ponovite Vežbu 7, ali uključite polja za potvrdu pomoću kojih će korisnik naznačiti da li želi da zna oblast, ili obim četvorougla (ili oboje). Skript mora da proveri da li je korisnik selektovao bar jednu opciju.

9. Modifikujte gateWay program sa slike 12.17 da dopusti više unosa na formi sa slike 12.15. GateWay program treba da vrati reference koje odgovaraju bilo kojem od brojeva koji su uneti u tekstualno polje.

10. Perl skript sa slike 12.24 smešten je u fajl verify.cgi i sadrži JavaScript funkciju pod nazivom reverif y. Ova funkcija formira URL koji referencira CGI skript u fajlu verif y. cgi. Da li je ovo primer rekurzivnog poziva? Zašto jeste, ili zašto nije?

11. Za sistem za naručivanje pice komentarišite prildadnost radio dugmadi, polja za potvrdu, ili pop-up menija pomoću kojih korisnik može da selektuje veličinu, vrstu testa i nadeve.

12. Ovaj zadatak podrazumeva da imate pristup UNIX, ili Linux serveru na kome možete da preuzmete i pokrenete Perl skriptove iz odeljka 12.5. Izvedite sledeće izmene u programu za naručivanje pice (izvedite svaku nezavisno).

    a. Dodajte veličinu "extra large" sa cenom 11.00 dolara.

    b. Dodajte Chicago Style u listu vrsta testa.

    c. Dodajte Onions u listu nadeva sa dodatnom cenom 0,50 dolara.

    d. Lfklonite radio dugmad za vrste testa i, umesto njih, koristite pop-up meni.

    e. Uklonite pop-up meni za veličinu pice i, umesto njega, koristite radio dugmad.

    f. Pretpostavite da cene nadeva važe samo za male pice. Za svaku sledeću veličinu pice cena svih nadeva se povećava za 0,25 dolara. Kkorisnik treba obavezno da dobije pregled tih cena na fomi.

    g. Dodajte fajl na server koji akumulira cene svih postavljenih narudžbina.

    h. Kreirajte formu i skript koji korisniku omogućavaju unošenje njegovog broja telefona, imena i adrese kada se prvi put prijavi serveru.

    i. Tekuće uređenje zahteva od korisnika da posebno naručuje pice; ne može odjednom da naruči više pica. Kako biste dizajnirali formu da korisnik može da naruči više pica?

13. Sistem za naručivanje pica sadrži Perl skript koji izračunava cenu pice na serveru. Skript bi mogao da generiše JavaScript funkciju koja bi izračunala cenu pice na strani klijenta. Da li postoji neka prednost ovakvog pristupa? Da li postoji nedostatak?

Reference

[Co99] Corner, D. E., and D. Stevens. *InternetWorking With TCPfIP. VoI. II. ANSI C Version: Design, Implementation, and Internals,* 3rd ed. EngleWood Cliffs, NJ: Prentice-Hall, 1999.

[CoOO] Corner, D. E. *InternetWorking With TCP/IP. VoI. 1. Principles, Protocoh, and Architecture,* 4th ed. EngleWood Cliffs, NJ: Prentice-Hall, 2000.

[Fe97] Felton, M. *CGI: Internet Programming in C+ + and C.* EngleWood Cliffs, NJ: Prentice-Hall, 1997.

[FlOl] Flanagan, D. *JavaScript: The Definitive Guide,* 4th ed. Sebastopol, CA: CReilly & Associates, 2001.

[GoOl] Goodman, D. *JavaScript Bible,* 4th ed. NeW York: Wiley, 2001.

[GuOO] Guelich, S., S. Gundavararn, and G. Birznieks. *CGI Programming With Perl,* 2nd ed. Sebastopol, CA: CReilly & Associates, 2000.

[Ku03] Kurose, *J.,* and K. Ross. *Computer NetWorking: A Top-DoWn Approach Featuring the Internet.* Reading, MA: Addison-Wesley, 2003.

[MeOl] Meltzer, K., and B. Michalski. *Writing CGl Applications With Perl.* Reading, MA: Addison-Wesley, 2001.

[ScOl] SchWartZ, R., and T. Christiansen. *Learning Perl,* 3rd ed. Sebastopol, CA: CReilly & Associates, 2001.

[Sh02] Shay, W. "A Multiplatform/Multilanguage Client/Server Project". In *Proceedings of the 33rd ACM SIGCSE Technical Symposium on Computer Science Education.* NeW York: ACM Press, 2002, pp. 401-405.

POGLAVLJE 13

Tehnologije sa komutacijom kola

Kompjuteri su dobri za brza i tačna izmčunavanja i za smeštanje ogromne količine informacija. Sa druge strane, mozak nije toliko efikasan za računske operacije i njegova memorija često "zakaže", što znači da je osnovna netačnost ugradana u njegov "dizajn". Jača strana mozga je njegova fleksibilnost. Neprevaziđen je u preciznim pogađanjima i poimanju kompletnog značenja predstavljenih informacija.

—**Jeremy Campbell,** britanski novinar

13.1 Uvod

Poslednje poglavlje ove knjige je posvećeno nekim dodatnim komunikacionim protokolima koji imaju, ili su imali značajnu ulogu u razvoju komunikacionih sistema na većim rastojanjima. Obično su razvijeni za implementiranje digitalnih telefonskih sistema, defmisanje interfejsa za mreže sa komutacijom paketa, ili ispunjavanje zahteva za kvalitet servisa (QoS) za različite vrste aplikacija.

U odeljku 13.2 predstavićemo ISDN (Integrated Services Digital Network), koji je dizajniran u vreme kada personalni kompjuteri i radne stanice nisu bili uobičajeni i dok Intemet još nije postojao. Trebalo je da zameni postojeđ telefonski sistem. Mnogi ljudi su videli potencijal u spajanju komunikacija i računarstva i prednosti konvertovanja u kompletno digitalni telefonski sistem. Ideja na kojoj je ISDN zasnovan bila je zamena svih telefona i njihovih kola digitalnim uredajima i komunikadonim linijama. Problem je predstavljala konkurencija sa postojećim telefonskim kompanijama koje su već implementirale digitalne komutaore na svojim mrežama i to što se tehnologija teško prodavala, jer mnogi nisu uviđali prednosti digitalnih telefona u poređenju sa analognim. Zaista, za nekoga ko treba samo da razgovara telefonom prednost je minorna, pa, čak, i nepostojeća.

U odeljcima 13.3 i 13.4 predstavićemo tri različite tehnologije virutelnih kola. X.25 je takode bio razvijen pre Interneta, ali je postao popularan i dizajniran je za komunikacione sisteme koji su prelazili medunarodne granice u Evropi.

Osim toga, pretpostavljeno je da je veći deo osnovnog komunikacionog sistema i dalje analogan i imao je relativno veliku učestalost pojave grešaka. Defmisao je interfejs između DTE (podsetite se definicije za DTE iz Poglavlja 4) i javne mreže za prenos podataka i omogućavao je uspostavljanje virutelnih kola sa udaljenim DTE-ovima. Frame relay je sledeći protokol za virtuelna kola. Dizajniran je za sporadični saobraćaj i mnogi ga vide kao naslednika X.25 protokola. Funkcioniše samo na dva sloja, za razliku od X.25 protokola, koji funkciomše na tri sloja i ne uključuje ni kontrolu grešaka, ni kontrolu toka. Ovim je pojednostavljena logika i omogućeno je korišćenje komutatora za brže prebacivanje okvira.

Poslednji protokol za virtuelna kola je Asynchronous Transfer Mode (ATM). Defmiše ćelije fiksne veličine koje komutatori mogu brzo da obrade. ATM obezbeduje i razlićite QoS zahteve, što ga čini prikladnim ne samo za standardne transfere podataka, već i za real-time aplikacije, kao što su audio, ili video aplikacije.

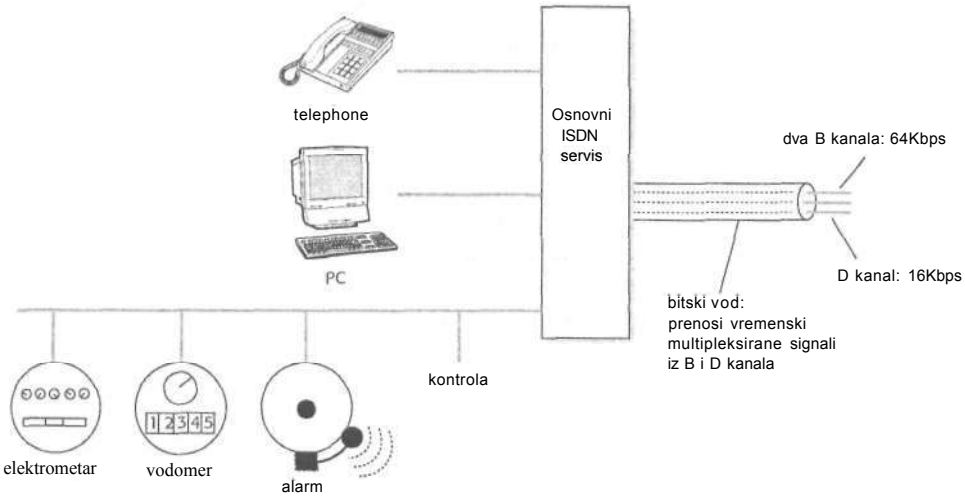
13.2 Digitalna mreža sa integrisanim servisima (ISDN)

Još pre 100 godina ljudi su počeli da razvlače kablove između kuća i gradova da bi mogli da komuniciraju telefonom. Otada je telefonska mreža prerasla u globalni komunikacioni sistem, koji koristi sve komunikacione medijume opisane u ovoj knjizi. Postoji još jedna karakteristika po kojoj se telefonski sistem razlikuje od ostalih mreža koje smo do sada predstavili: ima veliku analognu komponentu. Nismo zaboravili da su optički fiberi i digitalni uredaji za komutaciju uveli značajnu količinu digitalne tehnologije na mrežu. Ipak, telefoni su i dalje analogni uređaji, koji prenose analogne signale do lokalne telefonske centrale. Ovaj deo telefonske mreže se često naziva lokalna petlja (local Ioop), ili poslednja milja (last mile); drugi naziv ukazuje na najveću prepreku za kompletno digitalni sistem.

U početku je analogni sistem bio logičan izbor, zato što je telefon bio dizajniran za prenos glasa. U meduvremenu su se polja komunikadja i kompjuterske nauke sjedinila. Kompjuteri imaju suštinski značaj za komunikacione sisteme, a komunikacioni sistemi se široko koriste za povezivanje kompjutera. Zato je ITU-T razvio standard za globalni digitalni komunikacioni sistem pod nazivom Integrated Services Digital Netvvork (ISDN - digitalna mreža sa integrisanim servisima). Ako bi se u potpunosti implementirao, omogucio bi kompletnu integraciju prenosa glasa i negovornih signala (na primer, podaci, faks, video) u okviru jednog sistema. Uskoro ćemo objasniti ogromne prednosti ovakvog sistema.

Na slici 13.1 prikazana je funkcionalnost ISDN-ove *osnovne brzine (basic rate).* Obezbeduje tri zasebna kanala: dva B kanala prenose podatke brzinom od 64 Kbps i jedan D kanal prenosi podatke brzinom od 16 Kbps. Često se referencira kao 2B+D. B kanal prenosi "čiste" podatke, kao što su govorni podaci na koje je primenjena impulsna kodna modulacija (PCM), ili podaci koje generišu neki drugi uređaji, kao što je personalni kompjuter. D kanal se koristi za kontrolu i neke sporije aplikacije, kao što je telemetrija (daljinsko merenje), ili alarmni sistemi. Tri kanala se multipleksiraju sa podelom vremena na bitskom vodu (bit pipe) koji obezbeđuje konkretni prenos bitova.

ITU-T je razvio i Sevemoamerički standard (North American) za primarnu brzinu prenosa 23B+D (23 B kanala i jedan D kanal), koji se lepo uklapa u sisteme sa Tl nosiocem.



SLIKA 13.1 *Osnovni ISDN servis*

Evropski standard od 30B+D uklapa se u 2.048 Mbps kanal. Dodatni kanali za podatke obezbeduju kapacitet za veću količinu podataka iz različitih izvora.

Možda je čudno što najveća prepreka eventualnom implementiranju globalnog digitalnog komunikacionog sistema nisu tehnički problemi (mada ih ne potcenjujemo). Glavni problemi su logističke i ekonomske prirode. Značajan problem predstavlja i to kako ubediti korisnike telefona da će imati koristi od ISDN-a i da će cena konverzacije viti opravdana. I pored sve više ISP-a i ljudi koji imaju personalne kompjutere, većina ljudi i dalje koristi telefonski sistem za jednu namenu - za razgovor sa prijeteljima i susedima. Nju nije briga da li se glas prenosi preko analognih, ili digitalnih signala. Sa druge strane, prednosti obezbeđivanja dva kanala za podatke u osnovnom ISDN servisu su sasvim jasne. Na primer, roditelji više neće morati da čekaju da njihovo dete konačno oslobodi telefonsku liniju, jer automatski koriste drugu liniju.

Servisi

U ovom odeljku dat je pregled nekih servisa koje obezbeduje potpuno digitalni sistem. Mnogi od njih su dostupni, što nije iznenadujuće, jer je veći deo telefonske mreže već digitalizovan. Ovi servisi postoje zbog digitalne prirode komunikacionog sistema, ne radi samog ISDN-a. ISDN obezbeduje drugi način za njihovo implementiranje. Osim toga, ako se digitalne komponente prenesu i na stranu korisnika, servisi mogu da se implementiraju bez potrebe za složenim metodima modulacije i potencijalnim gubicima zbog konverzija digitalnih u analogne signale.

- Jedan od B kanala može da se koristi za elektronsko slanje poruka, koje se usmeravaju u zavisnosti od primaočevog broja telefona i smeštaju u lokalnom repozitorijumu blizu primaoca radi eventualnog pristupa. Ovo je slično prenosu faksom, ali bez konverzije analognih u digitalne podatke, koja je obavezna kod današnjih faksova.
- Telefonski brojevi dolazećih poziva mogu da se prikažu pre nego što se odgovori na poziv (identifikacija pozivaoca - caller ID). Digitalni sistemi podrazumevaju da su i odlazeći podaci i dolazeći signali digitalizovani. Broj izvora je kodiran u primljenim signalima. Ova karakteristika odvraća maliciozne, ili telemarketinške pozive. Osim toga, olakšava identifikaciju poziva u sistemu za hitne slučajeve (911 u SAD-u), što je veoma bitno ako osoba koja zove ne govori razgovetno, ili je reč o malom detetu. Pošto su dolazeći signali digitalni, brojevi telefona sa kojih je poziv iniciran mogu da se koriste i kao ključevi za zapise u bazi podataka. Ovo koriste profesionalci koji rade sa klijentima, ili pacijentima (lekari, advokati, brokeri, agenti osiguravajućih društava i tako dalje). Softver može da koristi dolazeći broj izvora radi pristupa zapisima o klijentu i za prikazivanje odgovarajućih informacija na ekranu. Ova karakteristika olakšava brzo i efikasno davanje odgovora na postavljena pitanja.
- Servisi za govornu poštu su slični već postojećim automatskim sekretaricama, koje omogućavaju ostavljanje poruka. Razlika je u tome što se poruke snimaju i smeštaju u lokalnom repozitorijumu.
- Svakog meseca uslužne kompanije šalju svoje radnike da po kućama očitavaju stanje na strujomerima, meračima potrošnje gasa i vodomerima. ISDN telemetrijski servis omogucava povezivanje merača na kompaniju i mesečno očitavanje pomoću jednostavnog poziva. Osim toga, senzori mogu da se postave u kućama kako bi detektovali požare, ili provale. Kada se aktiviraju, telefonski pozivi mogu da se pošalju automatski najbližoj vatrogasnoj, ili policijskoj stanici. U tim stanicama se ispisuje broj telefona sa koga se poziv inicira, ili se koristi pristup bazi podataka u kojoj se čuvaju adrese sa kojih je poziv iniciran.
- Videoteks - interaktivni pristup udaljenim bazama podataka omogućava, na primer, pristup bazama podataka sa telefonskim imenicima poput onih koji su danas dostupni u štampanim verzijama. Korisnici mogu da pristupe biblioteci baze podataka i postave upit za dobijanje podataka iz njenih kolekcija i pristupa enciklopedijama, ili javnim zapisima radi dobijanje informacija o određenim temama.
- Korisnici mogu da vrši prenos novca između bankovnih računa, da kupuju unošenjem koda određenog proizvoda i broja svoje kreditne kartice i da plaćaju rate kredita prenosom novca sa svojih bankovnih računa - sve telefonom.
- Više B kanala omogućava da se neke od tih aktivnosti izvode istovremeno. Ako neki član porodice trenutno razgovara telefonom sa svojim prijeteljom, drugi može da iskoristi drugi kanal za izvršavanje neke druge aktivnosti.

Nažalost, postoji prostor i za eventualne zloupotrebe. Raspoloživost tolike količine informacija preko jednostavnog telefonskog poziva zahteva ogromne bezbednosne mere.

Takode se nameću značajna sociološka i etička pitanja. U kojoj meri informacije treba da budu dostupne? Kako spredti da informacije ne dospeju u pogrešne ruke? Da li se merenja na daljinu mogu proširiti tako da se prate (i možda kontrolišu) neki drugi događaji u kući? Već postoje mogućnosti koje dopuštaju elektrodistributivnim preduzećima da daljinski isključuju klimatizaciju (uz saglasnost potrošača) u periodima kada se struja najviše troši. Da li bi ovo moglo (trebalo) da se proširi na kontrolu trošenja struje u toku energetskih kriza (ko uopšte i definiše kada dolazi do krize)?

Arhitektura

ISDN bi trebalo da radi sa različitim korisnicima i opremom, posebno ako treba da se integriše u kancelarijska okruženja. Ovo uključuje i dizajniranu opremu, imajući na umu ISDN i tekuću opremu koja datira iz vremena pre ISDN-a i ima malo čega zajedničkog sa ovom tehnologijom. Da bili olakšani dizajn strategija za povezivanje i standardizacija interfejsa, ITU-T je podelio opremu na *funkcionalne grupe.* Uređaji u okviru grupe obezbeđuju specifične mogućnosti. ITU-T je defmisao i referentne tačke za razdvajanje tih grupa, što je bio koristan dodatak u standrdizaciji interfejsa. Funkcionalne grupe i referentne tačke olakšavaju kategorizaciju osnovnih strategija povezivanja i obezbeđuju osnovu za dizajniranje složenijih arhitektura. U sledećoj listi opisane su oznake primarnih funkcionalnih grupa.

- NTl (network termination 1) "Neinteligentni" uređaji koji su zaduženi za fizičke i električne karakteristike signala - Prvenstveno izvršavaju funkcije sloja 1 u OSI modelu, kao što su sinhronizacija i tajming. NTl uređaji obično formiraju granicu između korisnikovog sajta i ISDN centrale. Sa druge strane, centrala funkcioniše slično centralama telefonskog sistema, a njen primarni zadatak je obezbeđivanje pristupa drugim sajtovima.

- NT2 (netvvork termination 2) "Inteligentni" uređaji koji mogu da izvršavaju funkcije naznačene na slojevima 2 i 3 OSI modela - U ovu grupu funkcija ubrajaju se komutacija, koncentracija i multipleksiranje. Uobićajeni NT2 uredaj je digitalna privatna centrala (PBX). Može da se koristi za povezivanje korisnikove opreme zajedno, ili sa NTl kako bi bio obezbeđen pristup ISDN centrali.

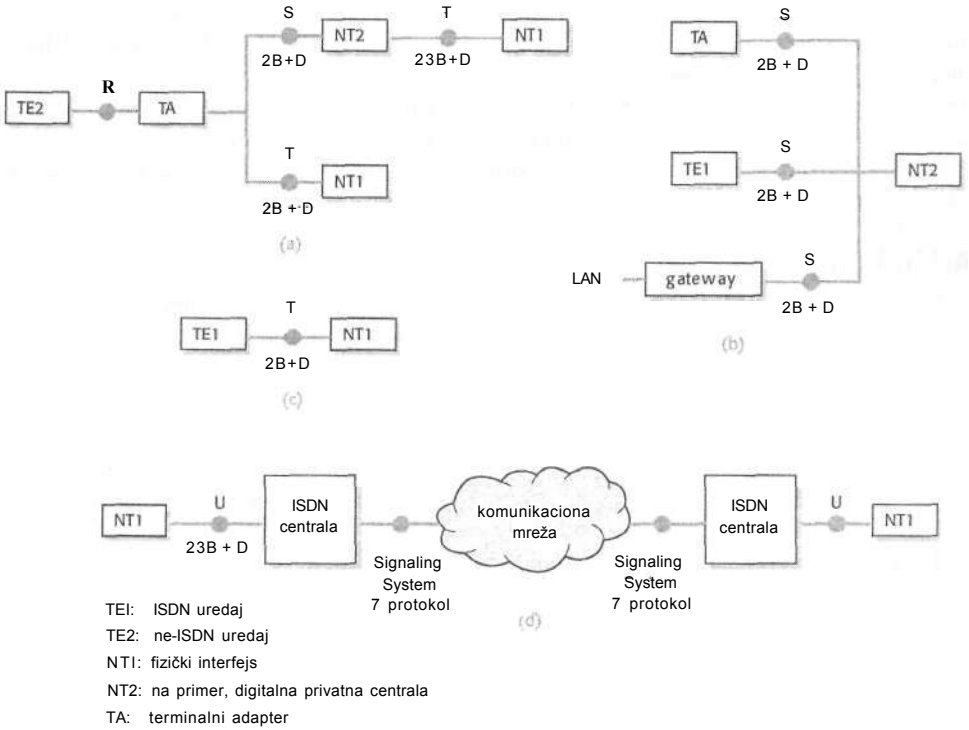
- NT12 Kombinacija NTl i NT2 u jedan uređaj

- TEl (terminal equipment 1) ISDN uredaji, kao što su ISDN terminal, digitalni telefon, ili kompjuter sa ISDN-kompatibilnim interfejsom - Oni se, obično, povezuju direktno na terminator mreže.

- TE2 (terminal equipment 2) Ne-ISDN uređaji, uključujua štampače, personalne kompjtere, analogne telefone, ili bilo šta što ima ne-ISDN interfejs, kao što su EIA.232, iliX.21.

- TA (terminalni adapter) Uredaji dizajnirani za korišćenje sa TE2 opremom za konvertovanje signala u ISDN-kompatibilni format - Svrha je integracija ne-ISDN uredaja u ISDN mrežu.

Na slici 13.2 prikazane su tipične funkcionalne grupe i objašnjeno jekako se mogu povezivati. Da bi interfejsi bili standardizovani, ITU-T je definisao referentne tačke između grupa.



SLIKA 13.2 *ISDN funkcionalne grupe i referentne Lačke*

Iako su ovde prikazane konekcije jednostavne, mogu da se kombinuju u mnogo veće i složenije. Inače, referentne tačke uvek dele funkcionalne grupe onako kako je ovde prikazano. Postoje četiri referentne tačke:

- **Referentna tačka R** Razdvaja TE2 opremu od TA (slika 13.2a). Tačka R može da korenspondira sa nekoliko različitih interfejsa u skladu sa TE2 standardom.
- **Referentna tačka S** Razdvaja NT2 opremu od ISDN uredaja (slika 13.2a,b). Podržava 2B+D kanal i ima bitsku brzinu od 192 Kbps*. Efektivno, razdvaja uredaje rezervisane za korisničke funkcije od uređaja koji su rezervisani za komunikacione funkcije.
- **Referentna tačka T** Pristupna tačka korisnikovom sajtu (slika 13.2a, c) - U opštem slučaju, razdvaja opremu korisnika od opreme mrežnog provajdera.

* Bitske brzine od dva B kanala i D kanal daju do 144 Kbps. Dodatni bitovi pomeraju bitsku brzinu do 192 Kbps, što ćemo uskoro objasnili.

Obično, ako je T interfejs između NTl i terminalne opreme, ili adaptera, odgovara 2B + D kanalu. Ako se nalazi između dva NT uređaja, odgovara 23B+D kanalu.

- Referentna tačka U Definiše konekciju izmedu NTl i ISDN centrale (slika 13.2d). Komunikacija izmedu različitih sajtova može da ide preko jedne, ili više *tačaka za signaliziranje transfera* (tip uredaja za rutiranje), a upravlja se pomoću protokola poznatog pod nazivom Signaling System 7 (uskoro ćemo ga predstaviti).

Protokoli

ISDN je sličan tekućem telefonskom sistemu. Da bi bila uspostavljena konekcija sa drugim sajtom, korisnik izvršava iste kontrolne funkcije. Kod konvencionalnog telefonskog sistema to podrazumeva biranje broja, a kod ISDN-a to znači slanje kontrolnih paketa. Telefon koristi signaliziranje u okviru opsega (in-band signaling) - tonovi generisani pritiskom na dugmad šalju se preko istog kanala preko koga će kasnije biti prenošen glas. ISDN kontrolne informacije se šalju preko D kanala. Pošto je to zaseban kanal u odnosu na onaj koji se koristi za prenos glasa, ili podataka, naziva se signaliziranje van opsega (out-of-band signaling). Značajan aspekat signaliziranja izvan opsega je što, nakon što se konekcija uspostavi za B kanal, D kanal može da koristi za druge namene. Aktivnosti kao što je telemetrija, ili zahtev za drugi poziv može da se izvede paralelno sa prenosima koji se odvijaju preko B kanala. Sledeći značajan aspekat je činjenica da B kanali prenose samo podatke (ili digitalizovani glas) - ISDN ne definiše sadržaj B kanala i tretka sve bitove kao "čiste" podatke. Ako dva korisnika komuniciraju preko B kanala pomoću određenog protokola, kao što je komutacija paketa, moraju da naznače formate paketa koji se prenose preko B kanala. Međutim, stvarni format paketa, uključujući zaglavlja i kontrolne informacije, transparentan je za ISDN.

Preko B kanala je moguće uspostaviti nekoliko tipova konekcija. Prva je konekcija sa komutacijom kola (circuit-switched connection) slična konekcijama koje se izvode u okviru telefonskog sistema. Celokupno signaliziranje i razmena kontrolnih informacija odvijaju se preko D kanala. Druga konekcija je virtuelno kolo, koje se uspostavlja na mrežama sa komutacijom paketa. Sve kontrolne informacije za uspostavljanje poziva i definisanje virtuelnog kola izvode se preko D kanala. Treći tip konekcije je sličan iznajmljenim linijama. Konekcija je stalno prisutna i pre slanja podataka nije neophodno uspostavljanje poziva.

D kanal je "posebna priča". On prenosi kontrolne informacije, kao što su uspostavljanje, ili okončavanje poziva, tip poziva i B kanal dodeljen tom pozivu. Zato je neophodno definisati kontrolu prenosa preko tog kanala.

Kompletan opis protokola koji formiraju osnovu ISDN-a može da bude tema posebne knjige (videti reference [St99], [Bl97b], [Me03] i [Gr98]). Naša namera je da obezbedimo uvod u neke značajne protokole, a čitaocima ostavljamo da pogledaju ove reference ako su zainteresovani za više detalja. Što se tiče ISDN-a, ITU-T je razvio dve zasebne serije preporuka, poznate kao I i Q serije dokumenata. I serije (od 1.100 do 1.605), prvi put objavljene 1984, a ažurirane 1988. i 1992. godine, sadrže više od 60 zasebnih dokumenata i opisuju teme kao što su arhitektura ISDN mreže, referentne konfiguracije, principi za rutiranje i interfejs korisnik-mreža. Neke od tih protokola ćemo uskoro opisati.

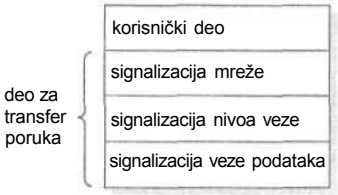
Q serije (od Q.700 do Q.795) opisuju slojeviti protokol poznat kao **Signaling System** 7 **(SS7),** koji je prvi put objavljen 1980. godine, a defmiše standard koji obezbeduje funkcionalnost u *integrisanoj digitalnoj mreži* (IDN). Primećujete da, umesto ISDN, koristimo termin IDN. IDN je posledica starog analognog telefonskog sistema, kod koga se prenos signala i komutacija izvode zasebno. Sa mogućnošću postavljanja svih prenosa u digitalnu formu, ove dve funkcije su integrisane. ISDN koristi IDN, ali uključuje i mogućnost integrisanja digitalizovanog glasa sa raznim drugim tipovima digitalnih podataka na digitalnim linkovima.

Signaling System 7 SS7 je četvoroslojni protokol (slika 13.3). Donja tri sloja čine **deo** za **prenos poruke (MTP - message transfer part)** i izvršavaju funkcije slične X.25 protokolu (biće predstavljen u narednom odeljku). Za razliku od X.25 protokola, SS7 je zadužen za mrežne funkcije kao što su rutiranje **i** pouzdanost. Na primer, obezbeduje pouzdani transport poruka u modu prenosa bez uspostavljanja konekcije. Četvrti sloj *korisnički deo (user parl)* sadrži specifikacije za kontrolu poziva, formate poruka, različite aplikacije i održavanje.

Najniži sloj **signalizacija** veze (signaling data link, ITU-T dokument Q.702) obezbeđuje sve fizičke i električne specifikacije i obezbeđuje 64 Kbps full-duplex prenos. Drugi sloj signalizacija **sloja veze** (signaling link layer, Q.703) obezbeđuje pouzdane komunikacije između dve sukcesivne tačke na mreži. Kao i kod drugih protokola sloja 2, defmiše format okvira i obezbeđuje proveru grešaka i kontrolu toka. Ovo je slično HDLC-u.

Treći sloj **signalizacija sloja** mreže (signaling netvvork layer, Q.704) obezbeduje pouzdan transfer poruka izmedu dve signalne tačke (krajnje tačke). Izvršava dve glavne funkcije: rutiranje i upravljanje. Na primer, utvrđuje da li treba preneti poruku od sledećeg mrežnog čvora, ili je isporučiti četvrtom sloju (korisnički deo). U slučaju da se poruka prenosi do drugog čvora, mora da se utvrdi koji se sledeći čvor. U slučaju da se prenosi do četvrtog sloja, mora da se utvrdi koji korisnički deo dobija poruku. Funkcije za upravljanje uključuju razmenu informacija izmedu čvorova u skladu sa rutama, otkrivanje grešaka i zagušenja i promenu rasporeda rutiranja.

Korisnički deo se sastoji od *telefonskog korisničkog dela* (TUP—*telephone nser pari)* i *ISDN korisničkog AeIa* (ISUP). TUP (dokumenti Q.721 do Q.725) opisuje uspostavljanje konekcija sa komutacijom kola za telefonske pozive, uključujući kontrolne poruke i njihov format. Primer poruka obuhvata one koje naznačavaju troškove poziva, ukazuju da je odgovoreno na poziv, ili ukazuju da je kolo oslobodeno zbog greške.



SLIKA 13.3 *Signaling System 7 protokol*

ISUP (dokumenti Q.761 do Q.766) izvršava slične funkdje, ali je označen kao servis za ISDN korisnike, za razliku od telefonskih korisnika. Sledi lista nekih primera ISUP poruka.*

- **Initial Address** Message (IAM) Poruka koja se prosleđuje napred za iniciranje korišćenje odlazećeg kola i za prenos broja i dmgih informacija koje se tiču rutiranja i kontrole poziva.
- **Subsequent Address** Message (SAM) Poruka koja može da se prosledi napred nakon IAM poruke kako bi bile prenete dodatne informacije o broju pozvane strane.
- **Information Request poruka (INR)** Poruka koja se šalje u okviru razmene nakon zahteva za informacije koje prate poziv
- **Information poruka (INF)** Poruka koja se šalje kako bi bil prikupljene informadje pridružene pozivu zahtevanom INR porukom
- **Address Complete** Message (ACM) Poruka u povratnom smeru koja ukazuje da su primljeni svi adresni signali koji su zahtevani za rutiranje poziva ka pozvanoj strani.
- **CaIl Progress poruka (CPG)** Poruka u povratnom smeru koja ukazuje da se desio događaj za vreme postavljanja poziva koji bi trebalo preneti pozivnoj strani.
- Ansvver Message (ANM) Poruka poslata u povratnom smeru koja ukazuje da je druga strana odgovorila na poziv - koristi se zajedno sa informacijama o troškovima poziva kako bi startovalo zaračunavanje troškova za onoga ko poziva i računanje trajanja poziva radi naplate međunarodnih usluga
- **Facility Request poruka (FAR)** Poruka koja se šalje za postavljanje zahteva za aktiviranje uređaja
- **Facility Accepted poruka (FAA)** Poruka koja se šalje kao odgovor na Facility Request poruku kako bi se ukazalo da je zahtevani uredaj pozvan
- **Facility Reject poruka (FRJ)** Poruka koja se šalje kao odgovor na Facility Request poruku kako bi se ukazalo da je zahtev za uredaj odbačen.
- **User-to-user information poruka (USR)** Poruka koja se koristi za prenos informacija od korisnika do korisnika, nezavisno od poruka za kontrolu poziva
- **CaIl Modification Request poruka (CMR)** Poruka koja se šalje u bilo kom smeru kako bi se ukazalo da pozivna, ili pozvana strana zahtevaju modifikovanje karakteristika uspostavljenog poziva (na primer, promena sa podataka na glas)
- **CaIl Modiflcation Completed** message (CMC) Poruka koja se šalje kao odgovor na CMR poruku kako bi se ukazalo da je zahtevana modifikacija poziva (na primer, sa glasa na podatke) kompletirala.

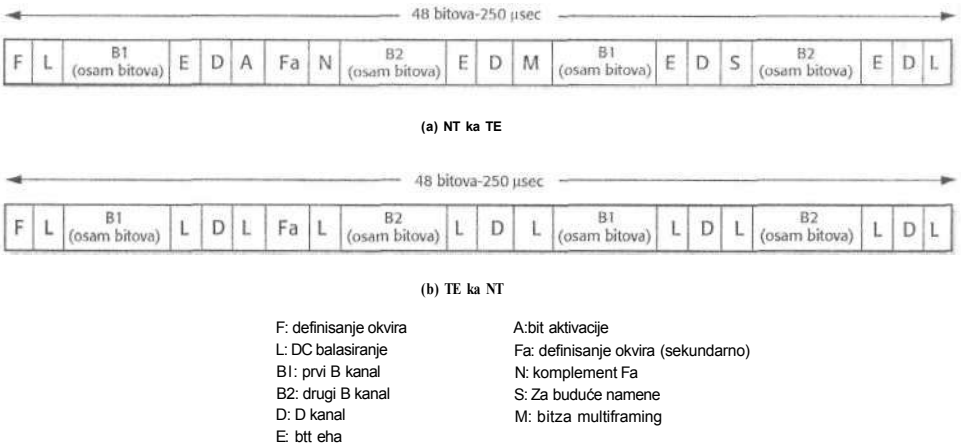
* Iz J. Griffith, *ISDN Explained,* © 1990, pp. 34—35 (ponovo štampano uz dozvolu John Wiley & Sons, New York)

- Call Modification Reject poruka (CMRJ) Poruka koja se šalje kao odgovor na CMR poruku kako bi se ukazalo da je zahtev odbijen.
- Release poruka (REL) Poruka koja se šalje u bilo kom smeru kako bi se ukazalo da je kolo otpušteno zbog osnovanog razloga (uzroka) i da Je spremno za postavljanje u IDLE stanje po prijemu Release Complete poruke; u slučaju da je poziv bio prosleden, ili reaitiran, u poruci je prenet odgovarajući indikator sa adresom redirekcije i preusmeravajućom adresom
- Release Complete poruka (RLC) Poruka koja se šalje u bilo kom smeru kao odgovor na prijem Release poruke, ili, ako je prikladno, na Reset Circuit poruku, kada je konkretno kolo postavljeno u IDLE stanje

Ovom listom smo samo "zagrebali" po površini SS7 protokola. Ako želite da pročitate nešto više o SS7, proučite bilo koju od referenci [Gr98], [B197a], [B197b], [Sc86] i [Ap86].

Protokoli osnovnih servisa Postoji više od 60 dokumenata u ITU-T 1 serijama koji opisuju ISDN standarde. Naša namera je da predstavimo preporuke za interfejs korisnik-mreža pomoću troslojnog protokola. Prvi sloj (1.430) opisuje fizički niz bitova za osnovni ISDN servis. Sličan opis (1.431) postoji i za primarni servis (ref. [Gr98]).

Niz bitova fizičkog sloja za osnovni servis odgovara onom u referentnim tačkama S i T (videti sliku 13.2). Na slici 13.4 prikazano je kako se 2B+D kanal osnovnog servisa multipleksira preko bitskog voda.



SLIKA 13.4 *Format fizičkog ISDN okvira*

Pre nego što ga opišemo, napomenimo nekoliko značajnih činjenica:

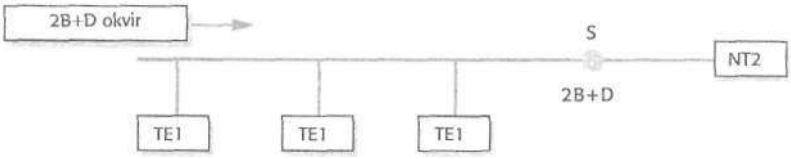
- Format okvira koji idu od TE do NT razlikuje se od formata okvira koji idu u suprotnom smeru.
- Komunikacije izmedu TE i NT su fijll duplex, tako da nema kolizija između okvira koji idu u suprotnim smerovima.
- Reč *okvir* ima neznatno drugačije značenje u poredenju sa ranijim kontekstima. Njegov format nije definisan protokolom sloja 2 (ili višeg sloja). Umesto loga, okvir jednostavno definiše kako se bitovi iz dva logička B i D kanala multipleksiraju u jedan fizički prenosni niz. *

Svaki okvir sadrži 48 bitova i šalje se na svakih 250 msec, što daje bitsku brzinu od 192 Kbps. Svaki okvir sadrži dva 8-bitna polja za svaki B kanal (označena kao Bl i B2). Servis koji koristi B kanal polaže 16 bitova u odgovarajuča polja u svakom okviru. 16 bitova se šalje na svakih 250 msec, što daje ukupnu brzinu prenosa podataka od 64 Kbps za svaki B kanal. Četiri D bita se smeštaju zasebno u svakom okviru (označeni sa D), tako da je brzina prenosa podataka za D kanal 16 Kbps; da kada TE ima paket sa informacijama za slanje preko D kanala, prenose se po četiri bita u jednom trenutku.

Preostali bitovi služe za neke kontrolne funkcije nižeg nivoa. Treba napomenuti da ISDN koristi **pseudoternamo kodiranje,** ili **tehniku inverznog naizmeničnog označavanja (alternate mark inversion technique)** kod koje se bit sa vrednošću 1 predstavlja nultim naponom, a bit sa vrednošću 0 pozitivnim, ili negativnim signalom. Osim toga, svaki 0 bit ima polaritet suprotan prethodnoj nuli. Ovo uzrokuje naizmeničnu promenu pozitivnih i negativnih signala u stringu sa nulama.

F bit je bit za definisanje okvira, pozitivni signal koji ukazuje na početak okvira. *L bit* je bit za DC balansiranje. Nakon F *bita* sledi negativni signal. Zajedno obezbeduju tajming i sinhronizaciju dolazećeg okvira. Preostali L bitovi su postavljeni na 0 ako je broj prethodnih nula neparan, ili na 1 ako je broj nula bio paran. Ovo obezbeđuje električno balansirane signale.

F bit je bit eha; postoji jedan za svaki D bit. U opštem slučaju, NT koristi eho vraćen od svakog D bita koji primi. U slučajevima kada je nekoliko TE-a povezano na NT preko jedne fizičke magistrale (slika 13.5) koristi se i E bit kao primitivni oblik "nadmetanja" za D kanal.



SLIKA 13.5 *Više TE-a povezanih na NTpreko jedne magistrale*

* Za 2B+D kanal postoji analogni fonnat.

Kada TE nema ništa za slanje preko D kanala, prenosi ravnomerni niz jedinica. Zbog toga, ako ni jedan TE ne koristi D kanal, NT prima sve jedinice na pozicijama D bita i vraća njihov eho. Tako proverom vraćenih E bitova TE može da detektuje neaktivni D kanal.

Kada TE treba da pošalje nešto na D kanal, on nadgleda vraćene E bitove. Ako sadrže nule, TE zna da neki drugi TE koristi D kanal i zato čeka. Ako su svi vraćeni E bitovi jedinice, ili je D kanal neaktivan, ili neki drugi TE prenosi niz podataka koji se sastoji isključivo od jedinica. Medutim, protokol višeg sloja izvršava dopunjavanje bitova, čime se ograničava broj uzastopnih jedinica koje je moguće poslati; ako TE detektuje broj koji premašuje ovu vrednost, on zaključuje da je D kanal neaktivan i startuje slanje.

Problem je što drugi TE može istovremeno da počne slanje preko D kanala. Ipak, korišćeni mehanizam i činjenica da se "nadmeću" za prostor u istom fizičkom okviru garantuju da će prenosi biti uspešni. Da biste videli kako ovo funkcioniše, pretpostavimo da je krajnji levi TE sa slike 13.5 poslao D bit sa vrednošću 1 (nema signala). Pretpostavimo da fizički okvir dopire do sledećeg TE-a, koji onda postavlja 0 (visoki, ili niski signal) na poziciju D bita. Efekat je isti kao da je 1 iz prvog TE zamenjena sa O iz drugog TE. Oba TE-a (i bilo koji drugi koji mogu, takođe, da iniciraju slanje) "osluškuju" vraćene E bitove. Ako TE detektuje E bit koji se razlikuje od poslatog D bita, zaključuje da je neki drugi TE zahvatio D kanal i privremeno obustavlja pokušaje da dobije D kanal. Ako TE vidi da su se njegovi D bitovi vratili kao eho preko E bitova, nastavlja slanje duž D kanala. Sledeći faktor koji ima uticaja je postojanje kašnjenje od IO bitova između slanja D bita i prijema odgovarajućeg E bita. Pošto su D bitovi u fizičkom okviru razdvojeni sa više od 10 bitova, odluka se donosi pre nego što se pošalje drugi D bit. Ako su prvi i naredni D bitovi od dvaTE-a isti, obaTE-a nastavljaju da šalju bitove sve dok se ne pojavi razlika. U toj tački neuspešni TE prestaje da šalje bitove.

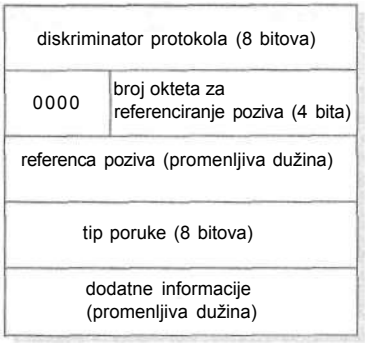
Među preostalim bitovima u fizičkom okviru nalazi se A *bit* koji predstavlja bit aktivacije i može da se koristi za aktiviranje TE-a. *Fa* i *M bitovi* se koriste za multiframing, koji omogućava dodavanje još jednog kanala (Q kanal). S i *N bitovi* su rezervisani za buduće namene.

ISDN protokol sloja 2, defmisan u dokumentima 1.440 i 1.441, poznat je kao Link Access Protocol for channel D (LAP-D). Možda se sećate da smo ga pomenuli u odeljku 9.2. Veoma je sličan HDLC-u i nema mnogo čega što bi moglo da se kaže o njemu, a da već nije istaknuto u Poglavlju 9.

Protokol sloja 3, definisan u dokumentima 1.450 i 1.451, uključuje tipove i formate ISDN poruka koje se šalju preko D kanala, protokole za uspostavljanje i poništavanje poziva, upravljačke funkcije i podršku uredajima. Ovde ćemo predstaviti ISDN poruke i kontrolu poziva.

Na slici 13.6 prikazan je format ISDN poruke. *Diskriminator protokola* omogućava D kanalu da šalje poruke od više protokola tako da se svaki od njih identifikuje pomoću odgovarajuće poruke. Može da definiše X.25 poruke, ili poruke kontrole poziva na relaciji korisnik-mreža, koje ćemo uskoro opisati. Ipak, postoje mogućnosti za uključivanje drugih protokola sloja 3 u doglednoj budućnosti.

Polje *CaIl Reference* defmiše poziv na koji se poruka odnosi. Ovo je neophodno zbog toga što se D kanal koristi za postavljanje i poništavanje poziva iz mnogih drugih kanala. Bez njega, ne bi bilo moguće definisati na koji se poziv kontrola odnosi.



SLIKA 13.6 *Format ISDN poruke na sloju 3*

4-bitno polje koje prethodi polju Call Reference definiše broj bajtova u polju Call Reference. Ovo je neophodno, zato što osnovni i primarni servisi imaju različitu dužinu polja Call Reference. Polje tipa poruke (Message Type) je jasno samo po sebi. Dokument 1.451 definiše oko 30 različitih tipova poruka - neki od njih su navedeni u tabeli 13.1.

Sadržaj i format preostalih polja zavise od tipa poruke i obezbeduju dodatne informacije. Sadrže informacije slične onima koje smo videli u drugim tipovima poruka, kao što su izvorna i odredišna adresa. Osim toga, definišu B kanal, adrese za redirekciju, razloge specifičnih poruka i status poziva.

Postavljanje poziva

Postavljanje poziva se ne razlikuje preterano (na ovom nivou rasprave) od ostalih procedura za inicijalizaciju koje smo do sada predstavili u ovoj knjizi. Na slici 13.7 prikazana je razmena poruka za vreme tipičnog postavljanja poziva. Imajte na umu da svaki TE funkcioniše u ime korisnika. Možda će Vam biti lakše da korisnikom smatrate nekoga ko inicira telefonski poziv preko mreže, sa komutacijom kola.

Kada želi da postavi poziv, korisnik kreira zahtev za TE. Ako napravimo analogiju sa telefonima, to bi bilo isto kao pritiskanje odgovarajućih tastera za defmisanje željenog broja. TE reaguje slanjem Setup poruke do NT-a. Setup poruka sadrži razne informacije (izvornu i odredišnu adresu, kanal, da li izvorna adresa treba da se prosleđuje i ko će da plati poziv). Kada Setup poruka stigne na mrežu, rutira se u skladu sa SS7 protokolima, tako da se utvrđuje ruta do drugog kraja. Osim toga, šalje Setup Acknowledge poruku nazad do TE-a. U toj poruci se nalazi obaveštenje da je zahtev za poziv prosleđen i zahteva se još informacija od TE-a ako u Setup poruci nije bilo dovoljno informacija.

Kada mreža dobije sve potrebne informacije, šalje se Call Proceeding poruka nazad do TE-a. U meduvremenu, ako sve prođe kako treba, Setup poruka putuje preko mreže i stiže do odredišnog TE-a, koji, onda, šalje Alert poruku nazad do pozivaoca (kako bi se označilo da je primio Setup poruku) i obaveštava korisnika o dolazećem pozivu.

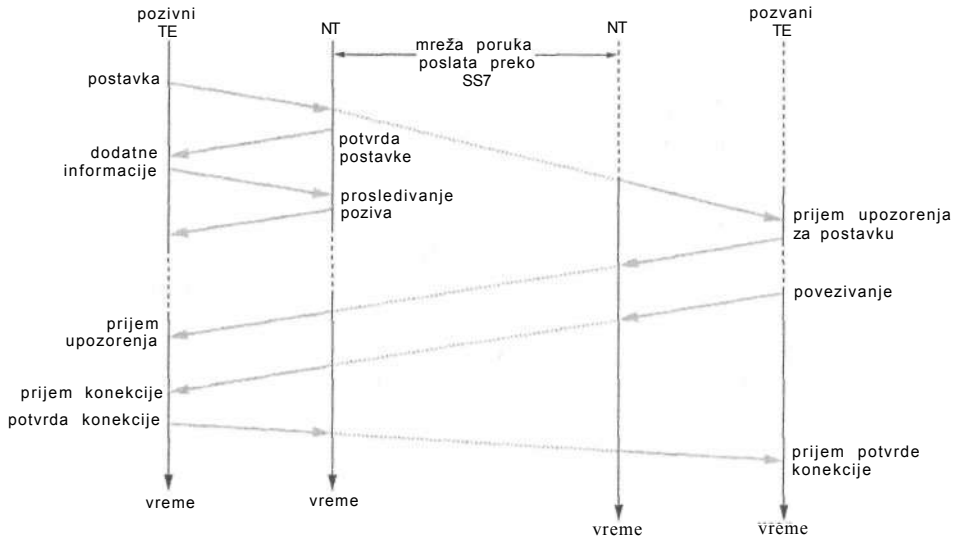
Tabela 13.1: Neke ISDN poruke sloja 3

Poziv	Značenje
Uspostavljanje poziva	
Alert	Šalje se do TE-a koji je inicirao poziv da bi se ukazalo da je pozvani TE upozorio svog korisnika na dolazeći poziv.
Call Proceeding	Šalje ga mreža da bi ukazala da je u toku zahtev za poziv.
Connect	Šalje se do TE-a koji je inicirao poziv da bi se ukazalo da je poziv prihvaćen.
Connect Acknovvledgment	Šalje ga TE koji je inicirao poziv da bi ukazao na prijem Connect poruke.
Setup	Šalje ga TE koji je inicirao poziv da bi zahtevao uspostavljanje poziva.
Setup Acknovvledgment	Šalje ga mreža do TE-a koji je inicirao poziv, ukazujući da je prethodna Setup poruka poslata. Osim toga, zahteva od TE-a koji je inicirao poziv da pošalje dodatne informacije radi obrade zahteva za poziv.
Informacije o pozivu	
Resume	Nastavlja suspendovani poziv.
Resume Acknovvledgment	Potvrda da Je prethodno suspendovani poziv nastavljen
Resume Reject	Prethodno suspendovani poziv nije mogao da se nastavi.
Suspend	Zahtev za suspendovanje poziva
Suspend Acknovvledgment	Poziv je suspendovan.
Suspend Reject	Poziv nije suspendovan.
User Information	Koristi se za prenos informacija izmedu dva TE-a.
Poništavanje poziva	
Disconnect	Zahtev za raskidanje poziva - ovu poruku bi trebalo da prati zaht(za oslobadanje B kanala
Release	Zahtev za oslobadanje B kanala - izdaje se nakon što korisnik spi slušalicu
Release Complete	Ukazuje da ie B kanal osloboden.

U slučaju telefonskog poziva ovo se izvodi generisanjem poznatog zvuka zvona. Kada se Alert poruka vrati do TE-a koji je inicirao poziv, i pozivač čuje zvuk zvona.

Kada korisnik odgovori na poziv, pozvani TE šalje Connect poruku nazad do pozivaoca. Kada je primi TE koji je inicirao poziv, zvono na njegovoj strani prestaje i TE odgovara Connect Acknowledgment porukom. Razmena Connect i Connect Acknovvledgment poruka identifikuje i potvrđuje B kanal koji će biti korišćen i započinje period u kome će se računati cena konverzacije. Svaki TE rutira PCM-kodirani glas na odgovarajući B kanai, kao što je opisano komentaru slike 13.4, i konverzacija počinje.

Poziv se završava kada jedan TE pošalje Disconnect poruku na mrežu, obično nakon što korisnik spusti slušalicu. Mreža rutira poruku i šalje Release poruku nazad do TE.



SLIKA 13.7 *Uspostavljanje ISDN poziva*

Nakon toga, TE šalje Release Complete poaiku nazad na mrežu, oslobađajući B kanal na tom kraju. Kada na drugi kraj stigne Disconnect poruka, mreža i TE razmenjuju i Release i Release Complete poruke. Ova procedura oslobađa B kanal i na tom kraju.

Širokopojasni ISDN

Digitalna priroda ISDN-a je atraktivna, ali neki ljudi su kritikovali uspostavljanje brzine prenosa podataka od 64 Kbps za B kanale. Zaista, u eri gigabitskih brzina na LAN-ovima i optičkog fibera, ISDN-ovih 64 Kbps gubi prednost. Zato je ITU-T razvio niz dokumenata u kojima je opisan **širokopojasni ISDN (B-ISDN - broadband ISDN).** Namera je da se iskoriste tekuće širokopojasne tehnologije da bi bili obezbeđeni servisi neophodni za visoke brzine prenosa podataka.

Pomoću B-ISDN-a moguće je obezbedili nekoliko servisa. Video konferenciranje i video tekfonija su dva primera. Tekuči ISDN standardi mogu da obezbede servis video telefonije, ali samo na manjim ekranima. Za veće ekrane sa visokim kvalitetom video slike neophodne su još veće brzine prenosa podataka. Sledeći primer je prikazivanje televizije sa izabranim sadržajem, koja već postoji u mnogim oblastima. Pomoću svog televizora možete da izaberete šta ćete gledati iz biblioteke filmova. Na kraju perioda za naplatu ispostavlja Vam se račun, u zavisnosti od Vašeg izbora programa.

Jedan od problema kod B-ISDN-a predstavlja izbor metoda za transfer. ISDN servisi koriste unapred dodeljene pozicije u okvirima sloja 1 za kanale (sećate se slike 13.4). Ovaj pristup, koji se još naziva i *tnod sinhronog prenosa* (STM - *synchronous transfer mode),* u suštini predstavlja multipleksiranje sa podelom vremena. Nedostatak je što nedodeljeni kanali dovode do "traćenja" propusnog opsega. Da li drugi kanali mogu da koriste nedodeljeni propusni opseg da bi bile povećane brzine prenosa?

Sledeći pristup koristi mod asinhronog prenosa (ATM - Asynchronous Transfer Mode), proverenu i široko primenjenu tehnologiju. ATM je beoma brz protokol sa komutacijom paketa koji koriste male pakete fiksne veličine (nazivaju se *ćelije),* optimizovane za multimediju. Uspostavlja logičke konekcije slične X.25 i Frame Relay protokolima (predstavljeni su u sledećem odeljku), ali mala fiksna veličina paketa omogućava kreiranje i rutiranje pomoću osnovnog hardvera. Osim toga, umesto da se unapred vrši dodela slotova za ćelije kanala, slotovi se dodeljuju aplikacijama kojima su potrebni. Prednost je što se prazni slotovi mogu iskoristiti. Nedostaci su dodatna složenost i činjenica da svaki slol zahteva zaglavlje. Na primer, u zaglavlju se definiše virutelno kolo, tako da se ATM pakeli mogu brzo rutirati. ATM ćemo predstaviti nešto kasnije u ovom poglavlju.

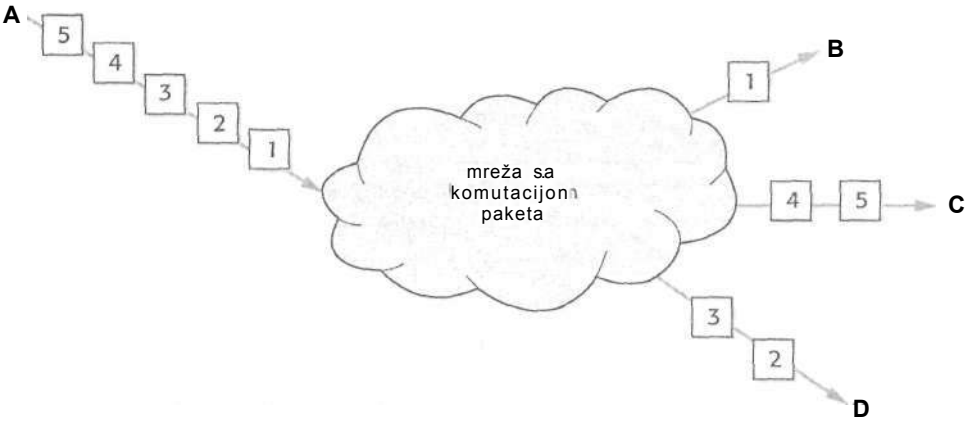
13.3 Protokoli za virtuelna kola: X.25 i Frame Relay

Mnoge evropske zemlje su 70-ih godina prošlog veka počele da razvijaju javne mreže za prenos podataka, dostupne svima kojima su neophodni mrežni servisi. One su se suočile sa drugačijim problemima u poredenju sa onim koji su postojali u Sjedinjenim Američkim Državama, gde su javne mreže razvijene delimično preko iznajmljenih postojeclh telefonskih linija. U Evropi je postojao problem postavljanja komunikacionih sistema preko medunarodnih granica. Tako SU, umesto da razvijaju zasebne i nekompatibilne standarde, evropske zemlje, pod pokroviteljstvom ITU-a razvijale jedan standard. Rezultat je označen kao X *serija* protokola, od kojih jedan (X.25) predstavljamo u ovom odeljku.

X.25 protokol je dizajniran u ranim danima kompjuterskih mreža i njegov razvoj je prouzrokovalo nekoliko faktora. Na primer, personalni kompjuteri nisu bili uobičajeni i većina ljudi je koristila neme terminale (bez centralne procesorske jedinice) kao interfejse ka mreži. Osim toga, mnoge komunikacione linije su bile analogne i imale su visoku učestalost pojave grešaka.

Naravno, ti uslovi više danas ne važe, tako da se X.2S više ne koristi u širokoj meri. Ovo nameće logično pitanje zašto treba proučavati stari protokol. Postoje brojni razlozi, a jedan je što je poslužio kao osnova za drugi, šire korišćeni protokol - za Frame Relay. U stvari, mnogi Frame Relay smatraju zamenom za X.25. Zbog toga, najpre dajemo opšti pregled X.25 protokola, a zatim ćemo preći na Frame Relay.

Mreže su obično mreže sa komutacijom paketa (packet-switched netvvorks), što je predstavljeno oblakom na slici 13.8. Funkcionišu tako što prenose pakete koji su im predati na jednom delu oblaka i rutiraju ih ka njihovim odredištima. Odluke o rutiranju obično donosi logika za komutaciju (kola) u čvorovima mreže. Mi posmatramo sve ovo iz perspektive nekoga ko ostvaruje interakciju sa mrežom. Paketi ulaze na mrežu preko tačke A, a izlaze u tačkama B, C, ili D. Ne moramo nužno da znamo (ili da brinemo) kako su tu stigli. Naš glavni cilj je da definišemo logičku konekciju između izvora i odredišta.

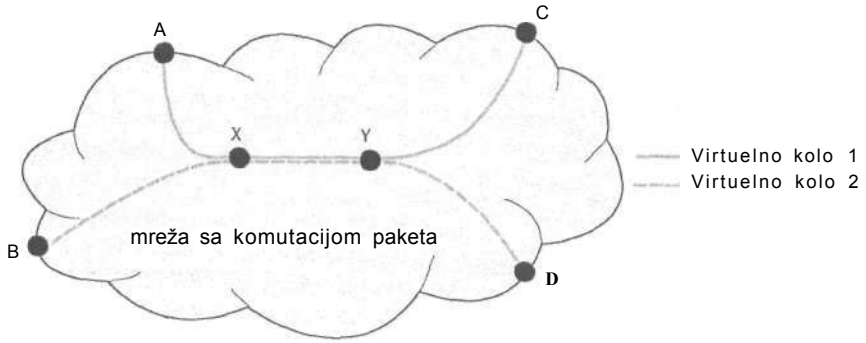


SLIKA 13.8 *Mreža sa komutacijom paketa*

Modovi mreža sa komutacijom paketa

Virtuelna kola Mreže sa komutacijom paketa obično funkcionišu u jednom od dva moguća moda. Prvi mod podrazumeva uspostavljanje virtuelnog kola između dve taćke. Ovo je donekle analogno kreiranju telefonske veze između dva čoveka. Uređaj povezan na mrežu zahteva konekciju sa uredajem na nekoj drugoj lokaciji. Ovaj zahtev se aitira preko mrežnih čvorova, uspostavljajući putanju između pozivaoca i odredišta. Svi naredni paketi koje pozivalac šalje slede istu putanju.

Ipak, konekcija nije samo fizička. Konekcije izmedu čvorova nisu rezervisane isključivo za jedno virtuelno kolo. U stvari, čvor i njegova konekcija ka susedu mogu da ućestvuju u nekoliko virtuelnih kola. Na slici 13.9 prikazana su dva preklopljena virtuelna kola. I A i B su zahtevali uspostavljanje konekcije sa C i D, respektivno.



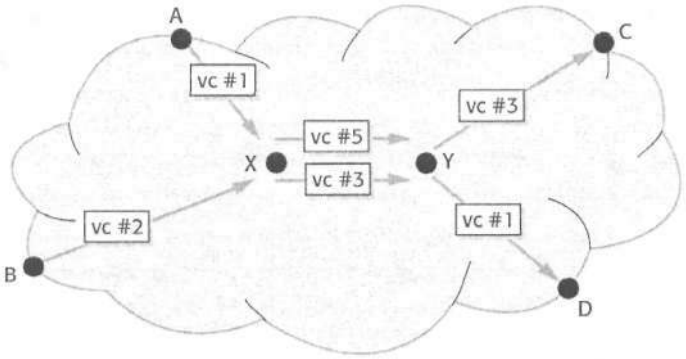
SLIKA 13.9 *Preklopljena virtuelna kola*

Putanje započinju na različitim lokacijama, ali se preklapaju u čvorovima X i Y. X upravlja paketima koji odgovaraju bilo kojem virtuelnom kolu i rutira ih do Y. Y rutira pakete u zavisnosti od toga iz kojeg virtuelnog kola dolaze.

Pošto se putanje virtuelnih kola mogu preklapati, svaki čvor mora da bude sposoban da utvrdi kojem virtuelnom kolu pripada dolazeći paket. Kako inicijalni zahtev za konekciju prolazi kroz svaki čvor, čvor dodeljuje broj svakom virtuelnom kolu, utvrdujući sledeći čvor kome treba poslati zahtev i tako se kreiraju zapisi u tabeli rutiranja. Tabela rutiranja sadrži broj svakog virtuelnog kola i sledeći čvor na odgovarajućoj putanji.

Napomenimo da svaki čvor virtuelnim kolima nezavisno dodeljuje brojeve, pa se isto virtuelno kolo u različitim čvorovima možda identifikuje različitim brojevima. Zbog toga, svaki čvor informiše svog prethodnika u virtuelnom kolu koji broj koristi za dolazeće pakete. Tako prethodni čvor zna koji se broj virtuelnog kola dodeljuje u sledećem čvoru i smešla ga u paket. Dakle, dolazeđ paketi sadrže broj virtuelnog kola koje se stiče u čvor. Logika za rutiranje u čvoru pristupa zapisu u tabeli rutiranja koji odgovara tom broju i šalje ga do sledećeg čvora. Ako se u sledećem delu virtuelnog kola koristi drugi broj, čvor ga smešta u paket. Na primer, na slici 13.10 prikazan je put paketa kroz sva virtuelna kola sa slike 13.9. Virtuelnom kolu izmedu A i C dodeljeni su brojevi 1 (od X), 5 (od Y) i 3 (od C). Kolu izmedu B i D dodeljeni su brojevi 2 (od X), 3 (od Y) i 1 (od D).

U tabeli 13.2 prikazani su relevantni zapisi u tabelama rutiranja čvorova X i Y. I'aket koji dolazi u X iz A kao broj virtuelnog kola sadrži 1. Tabela rutiranja čvora X ukazuje da je sledeći čvor Y i da on kao broj virtuelnog kola koristi 5. Zbog toga, X smešta 5 u paket i šalje ga do Y. Paket koji dolazi u Y iz X može da sadrži broj virtuelnog kola 5, ili 3. Ako sadrži broj 5, tabela rutiranja čvora Y nalaže da sledeči čvor bude C, a broj odlazećeg virtuelnog kola treba da bude 3. Ako sadrži broj 3, onda je sledeći čvor D, a broj odlazećeg virtuelnog kola je 1.



SLIKA 13.10 *Slanje paketa duž virtuelnog kola*

Tabela 13.2: Tabele rutiranja za čvorove X i Y sa slike 13.9

	Tabela rutiranja za čvor X		Tabela rutiranja za čvor Y		
Broj dolazećeg VC-a	**Broj odlazećeg VC-a**	**Sledeći čvor**	**Broj dolazećeg VC-a**	**Broj odlazećeg VC-a**	**Sledeći čvor**
1	**5**	**Y**	**5**	3	**C**
2	**3**	**Y**	**3**	1	**D**

/C-virtuelno koio

Servis datagrama Prednost virtuelnih kola je što se odluke o rutiranju donose samo jednom za svako kolo, čime se eliminiše potreba za donošenje takvih odluka za svaki paket. Možda ćete pomisliti da je ovo izuzetna prednost kada se šalje više paketa. ipak, možda važi upravo suprotno, jer veći broj paketa obično znači da će proteći više vremena dok se kolo ne uspostavi. Zbog toga, uslovi koji su neku putanju definisali kao dobru možda više ne važe. Odnosno, uslovi mogu da se promene tako da tekuća putanja traje duže. Rezultat je u tom slučaju redukovanje efikasnosti.

Sledeća opcija je servis datagrama kod koga svaki paket sadrži izvornu i odredišnu adresu. Kada paket ude na mrežu, čvorovi primenjuju logiku za rutiranje na svaki paket zasebno, koristeći najnovije informacije za rutiranje. Ovaj pristup smo već obradili tako da ga nećemo ponavljati. U tabeli 13.3 navedene su neke prednosti i nedostaci virtuelnih kola i datagrama.

Tabela 13.3: Poređenje virtuelnih kola i datagrama

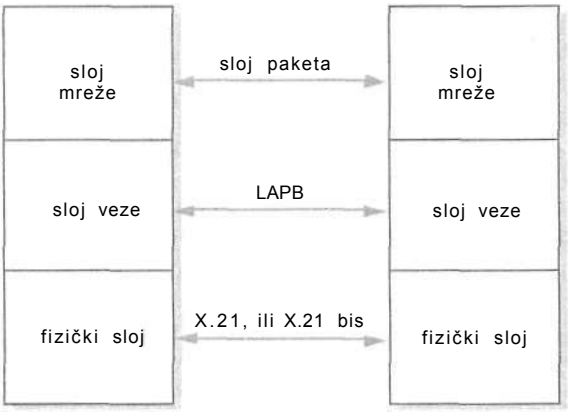
Pomaže sprečavanje zagušenja. Pošto čvor zna da je deo virtueinog kola, može da rezerviše prostor za predvideni dolazak paketa.	Neočekivani paketi mogu da otežaju kontrolu zagušenja.
Ako je virtuelno kolo isuviše dugo otvoreno, tekuća putanja možda nije najbolja sa stanovištč tekućih uslova na mreži.	Čvorovi rutiraju svaki paket, koristed najnovije informacije o stanju na mreži.
Odluka o rutiranju se donosi samo jednom za svaki skup paketa i šalje se zajedno sa paketom.	Donose se zasebne odluke o rutiranju za svako virtuelno kolo.
Paketi stižu istim redosledom kojim su poslati.	Paketi mogu da stignu van redosleda, tako da je neophodno preuređivanje u odredištu.
"Otkaz" čvora prekida konekciju virtuelnog kola, izazivajući gubitak paketa.	Ako čvor "otkaže", paketi mogu da se rutiraju tako da ga zaobiđu.

SLIKA 13.11 *Interfejs za javnu mrežu za razmenu podataka sa X.25 protokolom*

X.25 standard za interfejse

Značajan deo rada sa mrežama jeste njihov interfejs. Jedan takav interfejs je ITU X.25 standard. Mnogi ljudi koriste termin *X.25 mreža,* tako da neki veruju da X.25 defmiše mrežne protokole, što nije tačno. X.25 definiše protokol između DTE-a i DCE-a koji su povezani na mrežu (slika 13.11). Napominjemo da su ranije verzije bile fokusirane uglavnom na asimetričnu DTE-DCE relaciju. U kasnijim verzijama je prepoznata potreba za peer-to-peer komunikacijama između dva DTE-a. Zato X,25 može da se koristi striktno kao interfejs korisnika-ka-mrežl, ili kao konekcija između dva korisnika preko mreže.

X.25 definiše sinhroni prenos analogan onome sa tri najniža sloja OSI modela (slika 13.12). Sloj mreže prihvata podatke i postavlja ih u X.25 paket. X.25 paket je prenet do sloja veze, gde se ugraduje u LAPB okvir (LAPB je defmisan u odeljku 9.2). Nakon toga, fizički sloj prenosi LAPB okvir pomoču X.21 protokola, koji je prikazan u odeljku 4.4.



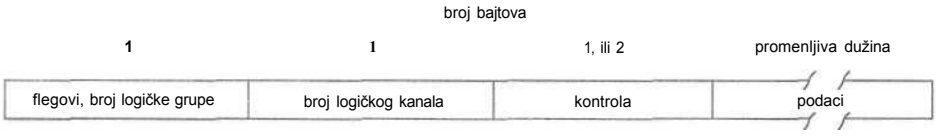
SLIKA 13.12 *Slojevi X.25 protokola*

Alternativno, X.25 može da koristi X.21bis standard, koji je dizajniran kao privremeni standard za povezivanje V serija modema sa mrežama sa komutacijom paketa. X.21 standard je trebalo da ga zameni, ali se to nije desilo. Opširnije razmatranja o X.21bis standardu možete da pronadete u referenci [Bl950. U nekim slučajevima X.25 može čak da koristi i EIA-232 protokol.

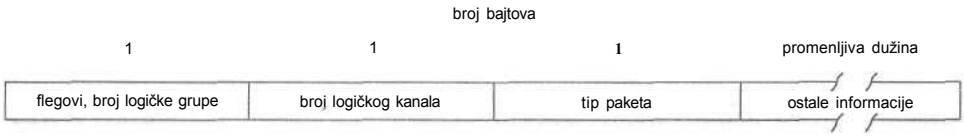
Pošto smo već predstavili dva niža sloja, ovde ćemo se fokusirati samo na protokol razmene paketa na sloju mreže.

Format paketa Prvi korak je definisanje formata paketa. Kao i kod prethodnih protokola, oni se razlikuju u zavisnosti od tipa paketa. Na slici 13.13 prikazana su dva primarna. Sledi pregled relevantnih polja u paketu:

- **Flegovi** Prva četiri bita definišu *General Format lndicator* (GFI) i, do određene mere, format paketa. Na primer, dva bita definišu da li se 3-bitn,i ili 7-bitni brojevi koriste za sekvenciranje i potvrde. Sledeći bit, nazvan *D bitom,* definiše kako se interpretiraju potvrde. Ako je D bit O, potvrda dolazi od DCE-a. Ako je D bit I, potiče od udaljenog DTE-a. Tako D bit utvrduje da li se kontrola toka izvodi za DTE-DCE konekciju, ili za logičku konekciju sa udaljenim DTE-om.
- **Broj logičke grupe (Logical Group Number)** i **Broj logičkog kanala (Logical Channel Number)** Ova dva polja zajedno definišu 12-bitni broj za virtuelno kolo koje je DTE uspostavio. Na ovaj način, DTE može da uspostavi do 4.096 virtuelnih kola.
- **Kontrola (Control)** (paket sa podacima) Sadrži 3-bitne, ili 7-bitne brojeve sekvence i potvrda koji se koriste za kontrolu toka. X.25 kontrola toka koristi prozore i ne razlikuje se značajno od kontrole toka kod ranije prikazanih protokola.



(a) Paket sa podacima



(b) Kontrolni paket

SLIKA 13.13 *Formati X.25 paketa*

Tabela 13.4: Tipovi X.25 paketa

Tip	Funkcna
Call Request	Kada DTE treba da uspostavi konekciju (poziva drugi DTE), on šalje Call Request paket.
Call Accepted	Ako pozvani DTE prihvata poziv, to potvrđuje vraćanjem Call Accepted (ili Call Confirmation) paketa.
Data	Koristi se za transfer podataka protokola visokog nivoa izmedu DTE-a. Obično postoji do 128 bajtova podataka, ali protokoli za usaglašavanje mogu da se dogovore o transferu do 4.096 bajtova u paketu.
Clear Request	Šalje ga čvor koji želi da okonča virtuelno kolo. Osim toga, može da se koristi ako DTE ne želi da prihvati poziv. Prijemni DTE ovaj paket vidi kao Clear Indication paket.
Clear Confirmation	Šalje se kao odgovor na Clear Request paket.

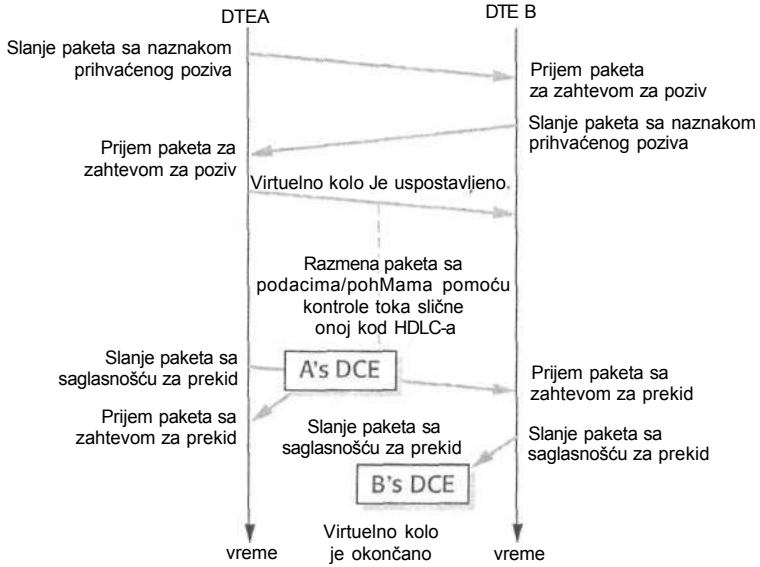
Ovo polje sadrži i bit koji se postavlja u poslednjoj seriji paketa kako bi se ukazalo na kraj niza paketa.

- **Podaci (Data)** (paket sa podacima) Ovo polje je jasno samo po sebi.
- **Tip paketa (Packet Type)** (kontrolni paket) Postoji nekoliko tipova paketa; neki od njih su definisani u tabeli 13.4.

Virtuelni pozivi X.25 obezbeđuje dva tipa vittuelnih kola izmedu DTE-a. **Permanentno virtuelno kolo** je slično iznajmljenoj telefonskoj liniji: bilo koji DTE može da šalje podatke bez dodatnih troškova za kreiranje i uspostavljanje poziva. Ovo je izuzelno korisno kada je potrebno preneti veliku količinu podataka. **Virtuelni** poziv, drugi tip, zahteva izvršavanje protokola za nadmetanje pre transfera bilo kakvih podataka.

Na slici 13.14 prikazani su povezivanje poziva i okončavanje procesa izmedu dva DTE-a (da bismo očuvali sažetost u ovoj knjizi, nismo prikazali DCE-e, ili mrežu, ali ne zaboravite da su i oni prisutni). DTE koji treba da inicira poziv konstruiše *CaII Rec/uest paket* koji sadrži broj virtuelnog poziva (ili logičkog kanala) i šalje ga preko svog DCE-a i mreže. Kada prijemni DCE dobije paket, on zahtevu dodeljuje broj virtuelnog poziva i isporučuje paket do prijemnog DTE-a. Napomenimo da se ne zahteva da brojevi kanala moraju da budu isti na oba kraja. Kao što je ranije opisano, oni se definišu dinamički. Ako je taj DTE spreman i sposoban da prihvati poziv, on šalje *CaII Accepted paket.* Kada prvi DTE primi Call Accepted paket, virtuelni poziv je uspostavljen.

Zatim, DTE-i razmenjuju pakete sa podacima i potvrdama u full-duplex modu, koristeći kontrolu toka sličnu onoj koja se koristi kod HDLC-a (videti odeljak 9.2). Kada bilo koji DTE odluči da okonča konekciju (DTE A sa slike 13.14), on kreira i šalje *Clear Request paket.* Lokalni DCE reaguje tako što šalje Clear Request paket do udaljenog DTE-a i odgovara svom lokalnom DTE-u slanjem *Clear Confirmation* paketa. Što se tiča lokalnog DTE-a, virtuelni poziv je okončan i broj logičkog kanala je raspoloživ za buduće pozive.



SLIKA 13.14 *Virtuelni poziv*

Na kraju, udaljeni DTE prima Clear Request paket. On ga vidi kao Clear Indication paket i odgovara slanjem Clear Confirmation paketa do svog DCE-a. Taj DCE briše i broj kanala, oslobadajući ga za druge pozive.

Iako je ranije široko korišćen, X.25 je bio izložen brojnim kritikama. Na primer, jedna od najozbiljnijih zamerki je bila što su protokoli zasnovani na X.25 standardu nudili samo konekciji orijendsani servis. Druga zamerka se ticala nekompletnosti sloja mreže (sloj 3), koji je, u stvari, sadržavao karakteristike koje se nalaze u protokolima višeg sloja. Na primer, sloj 3 OSI modela obezbeduje karakteristike za rutiranje, dok sloj 3 kod X.25 nema takve mogućnosti. Osim toga, X.2S obezbeduje neke konekciji orijentisane karakteristike sa udaljenim DTE-om. Pošto su konekcije između krajnjih korisnika karakteristične za protokole sloja 4, neki su ovo videli kao spajanje dva sloja u jedan, čime se gubi jasna razlika izmedu slojeva koji su definisani OSi modelom.

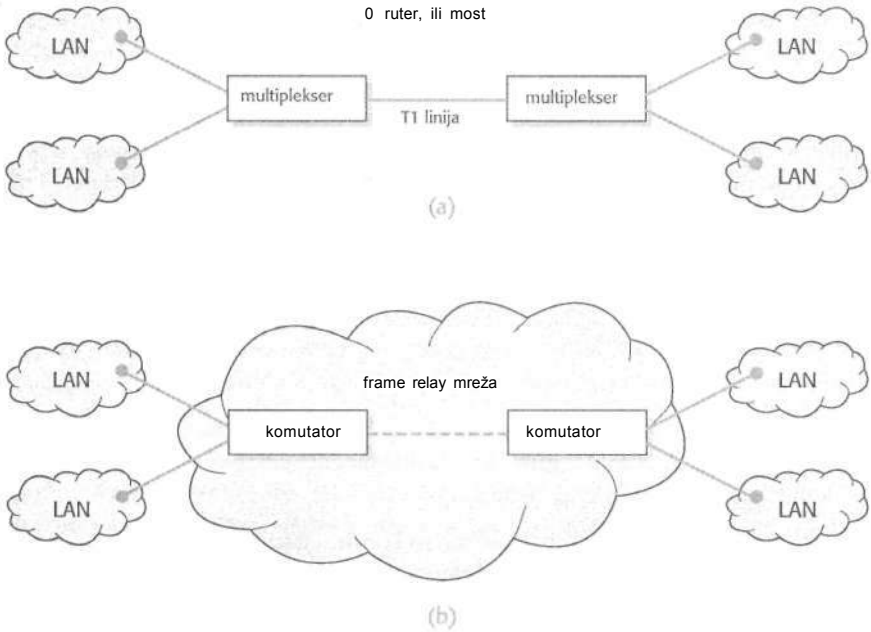
Frame Relay

Frame Relay ima neke zajedničke karakteristike sa X.25 standardom, ali ima i neke jedinstvene karakteristike;

- Dizajniran je tako da bude eftkasan i da maksimizira propusnost okvira (broj okvira koji se prosleduje u jedinici vremena).
- Defmiše WAN infrastrukturu. Masovno ga koriste telefonske kompanije, a tipičnu primenu nalazi kod povezivanja LAN-ova.

- Kao i X.25, podržava i *permanentno virtuelno kolo* (PVC) i **komutirano virtuelno kolo (SVC - svvitched virtual circuit).** Ako se koristi X.25 terminologija, druga opcija je označena kao virtuelni poziv.
- Za razliku od X.25, ne obezbeđuje kontrolu grešaka. Glavni razlog je što se prenos okvira izvodi preko digitalne opreme, kao što su optički fiberi. Kod takve opreme greške su veoma retke.
- Ne obezbeđuje kontrolu toka. Pretpostavlja se da se sva kontrola toka reguliše **na** višim slojevima kod krajnjih korisnika. Pošto su personalni kompjuteri i serveri zamenili neme terminale kod mrežnih konekcija, ova pretpostavka je opravdana.
- Originalno je bio dizajniran za brzine koje postiže Tl, mada može da dostigne i brzine zaT3.
- Dizajniran je za sporadični saobraćaj. Korisnik može da prenese veću količinu podataka u kratkom periodu, a zatim se ništa ne dešava u sledećem periodu. Ovo je tipično za situacije kada klijent zadaje upite za bazu podataka, ili preuzima slike sa ekrana.
- Možda je najznačajnija razlika u činjenici da Frame Relay funkcioniše samo na slojevima 1 i 2, što je značajan faktor za redukovanje troškova protokola. U stvari, pošto ne implementira kontrolu toka, ili kontrolu grešaka, ne implementira ni mnoge funkcije sloja 2 koje smo ranije opisali.

Na slici 13.15 prikazana je uobičajena primena Frame Relay protokola u mrežnom okruženju. Da bi se povezalo više LAN-ova iz različitih organizacija, Tl linija može da se koristi kao na slici 13.15a.



SLIKA 13.15 *Korišćenje Frame Relay mreže za povezivanje LAN-ova*

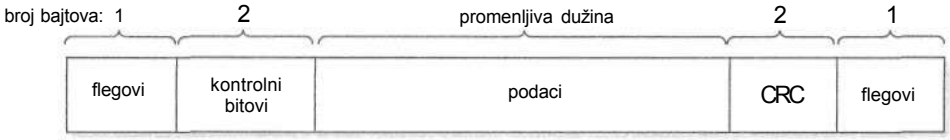
Medutim, postoji potencijalni problem. Sećate da smo u Poglavlju naglasili da se kanal (deo Tl propusnog opsega) rezerviše za svaku konekciju. Ovo je korisno ako se podaci neprestano prenose pomoću konekcije. Međutim, ako je prenos sporadičan, postojaće periodi kada se taj deo propusnog opsega ne koristi i konekcija nije dovoljno iskorišćena. Na slici 13.15b prikazana je frame relay mreža koja povezuje LAN-ove. Ruter iz svakog LAN-a je povezan na frame relay komutator i on šalje podatke. Frame relay protokoli u svakom komutatoru na mreži prenose podatke do njihovih odredišta. Dodatna karakteristika je što korisnik može da priušti sebi određenu brzinu prenosa podataka preko frame relay mreže. Ovo znači da su mrežni protokoli obavezni da obezbede bitsku brzinu koju je korisnik platio. Korisnik koji plati više može da koristi istu mrežu za slanje podataka na većim brzinama od korisnika koji plati manji iznos. Ubrzo ćemo opisati kako ovo funkcioniše.

Format okvira Počnimo opisom formata frame relay okvira (slika 13.16). On podseća na HDLC okvir, ali to nije ništa ćudno, jer se deo sloja 2 kod frame relay protokola zasniva na LAPD-u, varijanti HDLC-a.

Oba polja za flegove sadrže delimitere za početak i kraj okvira, kao i kod HDLC-a. U slučaju da se ovi flegovi jave u drugim delovima okvira, koristi se dopunjavanje bitova da bi taj uzorak bio transparentan za frame relay protokol. CRC polje koristi 16-bitni ITU-T V.41 generator polinoma $x^{16} + x^{12} + x^5 + 1$ za proveru grešaka. Ovo možda deluje kontradiktorno, jer smo ranije istakli da nema kontrole grešaka. Frame relay ne proverava greške koristeći CRC metod iz Poglavlja 6; medutim, ako detektuje grešku, on jednostavno odbacuje okvir. Ovo nije toliko ozbiljno kao što zvuči. Filozofija se zasniva na činjenici da su današnje mreže izuzetno pouzdane, tako da bi implementiranje kontrole grešaka uvelo dodatne troškove sa malim dobitkom. Ako dođe do tih retkih grešaka, okvir se jednostavno odbacuje. Protokol na višem sloju će sigurno detektovati tu grešku i rešiti nastali problem. Kompromis je napravljen na račun jednostavnijeg protokola sloja 2 i bržeg prosledivanja okvira. Polje podataka (Data) je, kao i obično, jasno samo po sebi. Maksimalna veličina ovog polja zavisi od proizvođača i može da bude i 4.096 bajtova.

Ovo nas dovodi do polja kontrole (Control). U nekim referencama ono se označava kao adresno polje (Address), jer sadrži informacije za rutiranje. Ipak, ne sadrži adresu kao takvu i uključuje neke kontrolne bitove koji se koriste za specijalno upravljanje, ili za ukazivanje na pojavu zagušenja. Kontrolno polje sadrži sledeće:

- **Data Link Connection Identifier (DLCI)** U stvari, postoji 6-bitno DLCI polje u prvom i 4-bitno DLCI polje u drugom bajtu. 4-bitno DLCI polje je ekstenzija za 6-bitno DLCI polje;y zajedno defmišu 10-bitni broj virtuelnog kola preko koga okvir putuje.



SLIKA 13.16 *Format Frame Relay okvira*

Ako je kolo PVC, provajder dodeljuje broj. Ako se koristi SVC, broj se dodeljuje za vreme postavljanja poziva. Logika u svakom frame relay komutatoru ispituje 10-bitni broj virtuelnog kola i rutira okvir u skladu sa njim. Ovaj pristup je veoma sličan onome koji se korisli kod X.25 i zato ga ovde nećemo ponavljati. Zašto je DLCI polje podeljeno? Objašnjenje sledećih polja će dati odgovor na to pitanje.

- Extended Address (EA) Postoji EA bit na kraju svakog bajla. Ovo omogućava kontrolnom polju da proširi i postigne DLCI vrednosti veće od 10 bitova (omogućen je veći broj virtuelnih kola). U našoj tekućoj konfiguraciji (slika 13.16), prvi EAbit je 0, a drugi je 1. Ako je potrebna proširena DLCI vrednost, drugi EA bit bi bio 0, čime se ukazuje na treći bajt u kontrolnom polju (koji bi imao još DLCI bitova i sledeći EA bit). Ako je potrebno koristiti četiri bajta, onda bi EA bit u trečem bajtu bio 0, a četvrti bajt bi imao još 6 DLCI bitova.

- Fonvard Explicit Congestion Notification (FECN) i Backvvard Explicit Congestion Notification (BECN) Jedan, ili oba bita su postavljena kada frame relay komutator detektuje zagušenje. Uskoro ćemo prikazati kontrolu zagušenja.

- Discard Eligibility (DE) Vrednost 0 ukazuje na visoki prioritet: Frame relay mora da učini sve da se okvir isporuči. Vrednost 1 ukazuje na nizak prioritet. Komutator može, pod odgovarajućim uslovima, da ispusti okvir niskog prioriteta. Uskoro ćete videti kako protokol ovo koristi za obezbeđivanje naznačene bitske brzine za korisnika i kako se izvodi kontrola zagušenja.

- Command/Response Indicator (C/R) Ovo je 1-bitno polje koje zavisi od višeg sloja. Može, na primer, da koristi bit da bi se ukazalo da li je okvir komanda, ili odgovor na prethodnu komandu.

Angažovana brzina prenosa informacija (CIR) Značajan deo frame relay protokola je angažovana brzina prenosa informacija (CIR - committed information rate). Meri se brojem bitova u sekundi i predstavlja propusni opseg koji je frame relay rezervisao za obezbedivanje prenosa preko virtuelnog kola. Ako korisnik iznajmi PVC, cena može da zavisi od CIR-a. Korisnik plaća više za veću bitsku brzinu. Ako korisnik generiše podatke na CIR, ili ispod njega, onda je propusni opseg dovoljan da iznese tu količinu podataka. Ukoliko korisnik generiše podatke većom brzinom, nema nikakvih garancija. Podaci mogu, ali i ne moraju da se isporuče, u zavisnosti od uslova na mreži.

Postoji i nekoliko drugih parametara koje moramo da defmišemo da bismo objasnili kako ovo funkcioniše:

- Vremenski interval T_1 koji se koristi za merenje bitskih brzina i naglih povećanja saobraćaja
- Angažovana veličina naglog povećanja saobraćaja B_c, meri se bitovima. Ako korisnik generiše B_c bitova u toku T, postoji dovoljan propusni opseg za njihovu isporuku. B_c, T i ClR su povezani formulom CIR = B_c/T.
- Višak naglog povećanja saobraćaja B_e - Ovo je broj bitova koji predstavljaju višak u odnosu na B_l. koje korisnik može da generiše u toku intervala T. Frame relay može da ih isporući ako postoji dovoljna količina propusnog opsega.

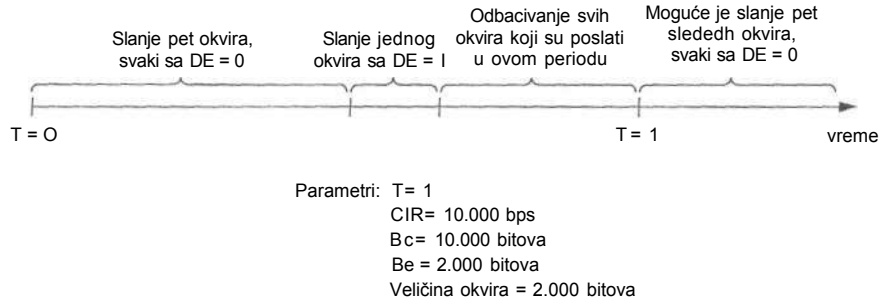
Da biste videli kako ovo funkcioniše, pretpostavimo da korisnik šalje okvire do frame relay komutatora. Virtuelno kolo je već postavljeno i dodeljen mu je CIR. Sve dok ukupni broj bitova

u okvirima ne prelazi B_c u toku intervala T_1 oni se tretiraju kao okviri visokog priorketa. Odnosno, komutator postavlja DE bit u svakom okviru na 0. Tako se drugim komutatorima ukazuje da su ovi okviri visokog prioriteta i da treba obavezno da ih isporuče. Ako u toku perioda T broj bitova premašuje B_0 (ali je i dalje manji od $B_c + BJ$, komutator postavlja DE bit na 1. Okvir i dalje "putuje" preko virtuelnog kola; medutim, komutator može da odbaci okvir čiji je DE bit postavljen na 1 ako utvrdi da dolazi do zagušenja. Ako korisnik pošalje više od $B_0 + B_e$, bitova u toku intervala T_1 višak okvira se odmah odbacuje.

Na slici 13.17 prikazan je primer. Pretpostavimo slučaj u kome je T = 1 sekunda, CIR = 10.000 bps, B_c = 10.000 bitova, B_e = 2.000 bitova i veličina okvira je 2.000 bitova. Počevši od trenutka t = 0, korisnik šalje pet okvira. Ukupan broj bitova je jednak B_c, i komutator postavlja DE bit u svakom okviru na 0. Ako korisnik ne pošalje još okvira, onda se generisanje podataka koji se nalaze u okviru CIR dešava u periodu dužine T. Medutim, ako korisnik pošalje još okvira pre trenutka t = 1, on zahteva bitsku brzinu koja premašuje CIR. Frame relay je spreman da dopusti povećanje do te tačke, ali ne daje nikakve garancije. Kao rezultat, komutator postavlja DE bit u šestom okviru na 1, Ipak, vrednost B_e, od 2.000 bitova ukazuje da je ovo poslednji okvir u periodu T koji će frame relay pokušati da isporuči. Komutator će odbaciti sve okvire koje korisnik isporuči pre trenutka t = 1. Kada dođe trenutak t = 1, korisniku će ponovo biti dopušteno da pošalje sledećih pet okvira visokog prioriteta (DE bit = 0) i jedan okvir niskog prioriteta (DE bit = 1) do trenutka t = 2. Osnovna ideja je da je frame relay rezervisao 10.000 bitova u sekundi propusnog opsega i da će pokušati da "izade na kraj" (ako to uslovi dopuštaju) sa najviše 12.000 bitova u sekundi. Svi bitovi preko tog broja u odredenom periodu T se odbacuju.

Kontrola zagušenja

Pošto frame relay ne implementira kontrolu toka između komutatora, zagušenje može da bude problem. Srećom, frame relay ima nekoliko načina da to reši. Prvo logično pitanje može da bude kako komutator utvrduje da postoji zagušenje. Svaki izlazni port ima bafer, ili red čekanja u koji smešta okvire. Okviri čekaju u redu sve dok ne dode vreme da budu prosleđeni.



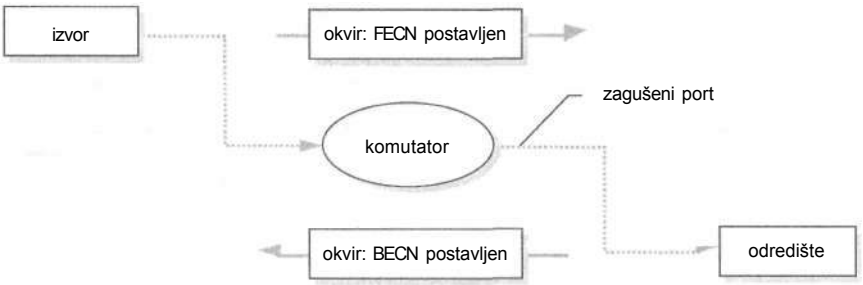
SLIKA 13.17 *Utvrđivanje okvira koji će biti poslati i njihovog prioriteta*

Ako broj okvira u redu premaši ustanovljenu granicu, postoji opasnost od prekoračenja. Drugim rečima, postoji suviše veliki broj okvira i odlazeći port je zagušen.

Jedna od mogućih reakcija na zagušenje je odbacivanje svih okvira kod kojih je DE bit postavljen na 1. Medutim, problem može da nastane zbog toga što protokoli krajnjih korisnika uvode suviše veliku količinu informacija na mrežu. Ako odgovarajući predajni i prijemni protokoli redukuju količinu okvira koje razmenjuju (vrše kontrolu toka na višem sloju), ovo bi redukovalo zagušenje. Zato frame relay sadrži način da se protokol krajnjeg korisnika obavesti o problemima zagušenja pomoću FECN i BECN bitova u okviru.

Na slici 13.18 prikazano je kako ovo funkcioniše. Pretpostavimo da virtuelno kolo od izvora do odredišta ide preko komutatora sa slike. Pretpostavimo i da dolazi do zagušenja na odlazećem portu u smeru odredišta. Ako se okvir prosleđuje preko tog porta, komutator postavlja njegov FECN bit na 1. Kada taj okvir stigne na odredište, prijemni protokoli mogu da utvrde da postoji zagušenje virtuelnog kola u tom smeru. Imajte na umu da okviri preko virtuelnog kola putuju u oba smera. Ako primi okvir sa zagušenog porta, komutator postavlja BECN bit na I. Kada taj okvir stigne u izvor, predajni protokoli mogu da zaključe da je došlo do zagušenja virtuelnog kola u suprotnom smeru. Značajno je što oba kraja virtuelnog kola mogu da detektuju da je došlo do zagušenja i u kojem smeru. Napomenimo da zagušenje može da se desi samo u jednom smeru.

Kako ovo može da pomogne? Zavisi od viših slojeva. Ako kontrola toka kod predajnog protoko-Ia (na višem sloju) detektuje zagušenje, može da redukuje svoj prozor i da redukuje broj okvira koje šalje. Može čak i da prestane da šelje okvire u odredenom periodu. Ako kontrola toka prijemnog protokola detektuje zagušenje, može da odloži slanje potvrde. Ako odgovarajuči pošiljalac čeka duže na potvrdu, on će poslati manji broj okvira u jedinici vremena. Ako istekne vreme za okvire pošiljaoca, pošiljalac može da poveća vrednost brojača, ili da smanji veličinu prozora, ili oboje. Sledeća opcija (na primer, sa TCP-jem) je da prijemni protokol redukuje svoj kredit. Kada predajna strana redukuje kredit, reaguje smanjivanjem svog prozora.



SLIKA 13.18 *Postavljanje FECN/BECN bitova zbog zagušenja*

Naravno, ovo je samo osnovno upoznavanje tehnologije u razvoju; postoje brojni detalji koje nismo obuhvatili, kao što su prenos glasa preko frame relay protokola, opširnije poredenje sa X.25 i ATM, problemi konfigurisanja i detaljnije objašnjenje kontrole zagušenja. Zainteresovani čitaoci mogu da pronadu više informacija u referencama [BlOO], [Hu03], [St99] i [McOl].

13.4 Asinhroni prenos

Uticaj koji je Intemet imao na ceo svet skoro je nemerljiv. Bez dvoumljenja može da se kaže da je uneo "revoluciju" u način međusobne komunikacije među ljudima, traženje i organizovanje informacija, korišćenje slobodnog vremena i, uopšte, način na koji obavljamo svoje poslove. U prethodnoj deceniji on je zabeležio ogroman napredak (postoje neke procene da se udvostručavao na svakih 18 meseci). Internet je veoma dobar u onome za šta je dizajniran: u obezbeđivanju sredstava za deljenje ogromne količine informacija izmedu različitih sistema. Ipak, neki su postavili pitanja u vezi njegove mogućnosti da ispuni zahteve korisnika u budućnosti. Već smo razmotrili aspekte zastarevanja lnternet protokola (IP) i razvoj IPv6 kako bi se zahtevi ispunili. Kreatori IPv6 su veoma naporno radili da bi se prevazišlo zaslarevanje IP-ja i obavile pripreme za globalnu komunikacionu mrežu 21. veka. Ključni problemi u dizajnu IPvG bilo je obezbedivanje povećanja brzine kojom se IP paketi mogu rutirati preko Interneta.

Međutim, u poslednjoj deceniji je došlo do neverovatnog napretka video i glasovnih komunikacija. Neki tvrde da za takve aplikacije IPv6 neće biti adekvatan, čak ni sa protokolima kao što su RTP i RSVP (videti Poglavlje 11). Čitaoci treba da budu svesni da ovde ne mislimo na situacije u kojima se korisniku omogućava da pristupi video fajlu (setite se MPEG-a) i da ga pokreće lokalno. Video aplikacie o kojima ovde govorimo podrazumevaju mogućnost prikazivanja videa u real-time modu.

ledan primer je prikazivanje videa na zahtev, kod koga korisnik može da zahteva prikazivanje filma u vreme koje njemu odgovara. Provajder čuva digitalizovanu kopiju filma i prenosi je do korisnika, koji je gleda dok se prenosi. Sledeći primer su video konferencije. Zamislhe da sedite za personalnim kompjuterom čiji je ekran podeljen na nekoliko prozora, u kojima se nalaze slike drugih osoba. Svaka od tih osoba ima iste mogućnosti. Možete da govorite u mikrofon i svaka od njih može da Vas vidi i čuje dok govorite. Slično tome, i Vi možete da ih vidite i čujete. Osim što se koriste male slike na ekranu, isto je kao da se svi nalazite u istoj prostoriji i učestvujete u konverzaciji (sledeća prednost može da bude pozivanje lokalnog programa za crtanje brkova i spojenih obrva onima koji nastoje da dominiraju u konverzaciji).

Ovakve aplikacije zahtevaju mnogo više od brze isporuke digitalizovanog glasa i slika. Zahtevaju isporuku sa ograničenjima koje nameće prikazivanje u realnom vremenu, tj. konzistentan i predvidljiv tok informacija. Ukoliko postoje, kašnjenja moraju da budu veoma mala, jedva primetna, jer izazivaju pauze u rečenicama, ili video efekat sličan onome iz filmova u kojima su video slikeizgledale, u stvari, kao sekvence nepokretnih slika. Neki su smatrali da IP neće moći da zadovolji ova ograničenja i da je tehnologija Asynchronous Transfer Mode (ATM) budućnost za audio i video aplikacije.

ATM je dizajniran tako da podseća na tehnologiju sa komutacijom kola kod telefonskog sistema po tome što se sve dešava u realnom vremenu. Međutim, ATM predstavlja potpuni zaokret u odnosu na tehnologiju sa komutacijom kola. Na primer, održava mogućnost rutiranja individualnih paketa podataka. Sledeće stavke opisuju primarne atribute ATM mreže i, na prvi pogled, pomoći će Vam da uočite neke glavne karakteristike:

- Orijentisanost vezi
- Komutacija paketa
- Paketi fiksne veličine nazvani ćelije
- Prenos ćelija velikim brzinama, sa malim kašnjenjima
- Celije ne sližu van redosleda.
- Brzine od 155,5 Mbps (ovo je neophodna brzina prenosa podataka za full-motion video), ili 622 Mbps (četiri 155,5 Mbps kanala) preko SONET-a; kako se tehnologija razvija, nema sumnje da će se u budućnosti koristiti gigabitske brzine
- Mreža je dizajnirana većim delom za rad sa real-time video aplikacijama i aplikacijama za prenos glasa.
- Značajno je promovišu telefonske kompanije.
- Ovu tehnologiju koristi B-[SDN.

U opštem slučaju, ATM inicijalno postavlja konekciju izmedu sajtova tako da se između njih uspostavlja virtuelno kolo. U ATM terminologiji ovo se naziva *signaliziranje.* Zasniva se na ITU-T protokolu Q.2931, koji predstavlja podskup od Q.931. Virtuelno kolo odgovara specifičnoj putanji koja se utvrđuje za vreme signaliziranja i sve ćelije se šalju preko virtuelnog kola, prateći istu putanju. Kada se prenosi završe, ATM protokoli uključuju fazu raskidanja konekcije.

Veći deo ovoga možda zvuči slično prethodno obrađenim temama, posebno X.25. Naravno, razlike se ogledaju u detaljima i operacijama. U ovom odeljku dajemo opšti pregled ATM-a i razmatramo kako su ostvareme neke prednosti i opisujemo neke tehnike za komutaciju (rutiranje) ćelija, definiciju ćelije, virtuelna kola, upravljanje konekcijom i slojeviti referentni model. Naš glavni cilj je da čitaocima obezbedimo osnove ATM-a i da prikažemo kako se on razlikuje od drugih protokola. Naravno, zainteresovani čitaoci mogu da pronadu dodatne detalje u referencama [BI99] i [Ha98J.

Prednosti malih ćelija fiksne veličine

ledan od ključnih aspekata ATM-a je što prenosi sve informacije u ćelijama veličine 53-bajta (48 bajtova podataka i 5 bajtova zaglavlja, čiji ćemo format opisati nešto kasnije). Ovo nameće nekoliko logičnih pitanja. Šta je toliko specijalno u prenosu infbrmacija u ćelijama fiksne veličine i zašto se koristi baš 48 bajtova podataka? Odgovor na drugo pitanje glasi: zato što su oni koji su razvijali protokol tražili nešto drugo. Iako zvuči besmisleno, ustanovljena veličina je rezultat kompromisa, rešenje kojim niko nije dobio ono što je stvarno zahtevao.

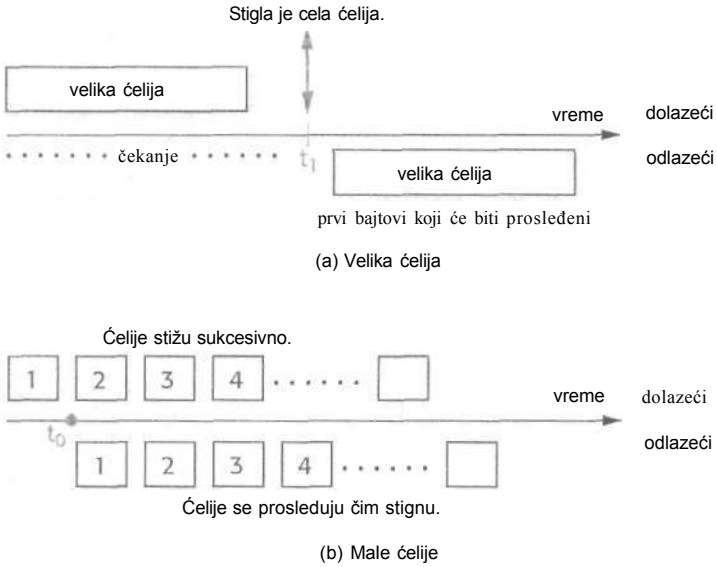
Dok se ATM nalazio u ranim stadijumima svog razvoja, njegov komilet je tražio veličinu ćelija koja bi zadovoljila nekoliko ograničenja. Na primer, ATM je morao da radi dobro sa postoječom opremom. Ćelije su morale da budu dovoljno male da mogu brzo da produ posredničke komutatore i da interni redovi čekanja ostanu mali. Veličina je morala da bude dovoljno mala da se efikasno implementiraju tehnike za korekciju grešaka. U komitetu je postojala evropska frakcija koja je promovisala korišćenje 32 bajta sa korisnim informacijama. Evropske zemlje su relativno male - 32 bajta korisnih informacija su mogla da se implementiraju bez potrebe da se u telefonskim kompanijama instaliraju kola za poništavanje eha (kola koja uklanjaju signale koji se vracaju kao eho iz svojih odredišta). Američke telefonske kompanije su ionako instalirale ova kola i njima je više odgovaralo da se koriste 64 bajta korisnih informacija kako bi se smanjio koeficijent zaglavlje-veličina ćelije. I Japanu je odgovaralo korišćenje 64 bajta korisnih informacija. Rešenje je bio numerički prosečna veličina između 32 i 64 bajta; rezultat je 48 bajtova korisnih informacija.

Prenos informacija u malim paketima fiksne veličine ima nekoliko prednosti. Prva je jednostavnost. I'rogrameri lako pamte ovu činjenicu. Pisanje programa koji manipulišu strukturama sa zapisima fiksne veličine je mnogo lakše od pisanja programa koji moraju da rade sa strukturama promenljive veličine. Potrebno je izvesti i manji broj provera. Ovo naravno uprošćava i hardverske i softverske zahteve za obavljanje posla, a, ujedno, snižava cene.

Sledeća prednost korišćenja malih ćelija je što jedna ćelija neće dugo zauzimati odlazeći link link. Na primer, pretpostavimo da je komutator upravo počeo da prosleđuje ćeliju kada stiže druga ćelija sa visokim prioritetom. Visoki prioritet obično znači da može da se probije na početak reda čekanja, ali neće prekidati prenos ćelije koji je u toku. Zato mora da čeka da se taj prenos kompletira. Da je ta ćelija velika, postojala bi verovatnoća da ćelija visokog prioriteta duže čeka. Manja ćelija znači da se ćelije visokog prioriteta brže prosleduju.

Sledeći razlog je što se ulazne i izlazne operacije mogu preklapati, jer komutator može brže da uzima manje čelije. Da biste videli kako ovo funkcioniše, pretpostavimo da ćelije stižu u celosti i da se baferuju pre prosledivanja. Ako ćelija sadrži 1.000 bajtova korisnih informacija, onda se svih 1.000 bajtova mora primiti i baferovati pre nego što se prvi bajt pošalje preko odlazećeg linka. To je ilustrovano na slici 13.19. Na slici 13.19a prikazana je vremenska linija u okviru koje se primaju dolazeći bajtova od velike ćelije. Prvi bajtovi se ne prosleđuju pre trenutka $t_{1(}$ nakon što stignu svi bajtovi. Na slici 13.19b prikazano je šta se dešava ako se velika ćelija podeli na manje 48-bajtne čelije. U ovom slučaju prva ćelija može da se prosledi čim stigne u trenutku t_0 (mnogo pre trenutka ^), jer ne mora da čeka da budu primljene sve ostale ćelije. U okviru vremenske linije sa slike 13.19b informacije se istovremeno primaju i prosleđuju, čime se povećava brzina kojom se ćelije prosleđuju ka njihovom konačnom odredištu.

Postoje i druge prednosti. Na primer, pošto se informacije brže prosleđuju, manja količina informacija mora da se ćuva u komutatoru, što doprinosi smanjenju odlazecoh redova čekanja. Krajnji rezultat je da korisne informacije manje vremena provode u baferima. Sledeća prednost je što bajtovi do odredišta stižu konzistentnom učestalošću, umesto da se na odredištu javljaju samo sporadično, sa naglim povećanjem količine saobraćaja u određenim trenucima i dugačkim periodima u kojima nema nikakvog saobraćaja (to je kao da u svakom 10. minutu imate nagli skok u količini saobraćaja, a u periodima od devet minuta između tih trenutaka se ne dešava ništa). Ovo je posebno važno kod video aplikacija i aplikacija za prenos glasa, kod kojih podaci moraju da pristižu brzo i konzistentno.



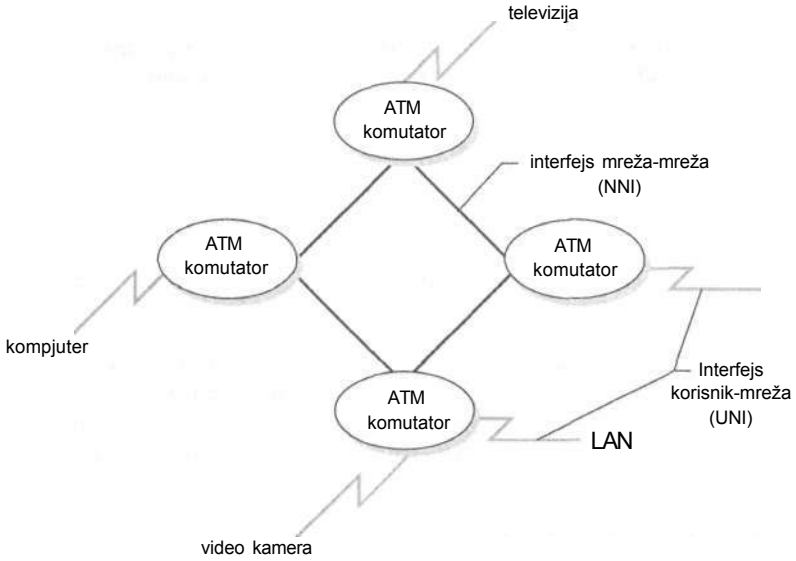
SLIKA 13.19 *Preklapanje ulaza i izlaza ćelija*

Poslednja prednost je što male ćelije fiksne veličine olakšavaju dizajniranje komutatora koji mogu konkurentno da prosleduju više paketa. Ovo je posebno korisno, jer se redukuje vreme koje jedna ćelija mora da čeka dok se druga ne prosledi. Teško je razumeti kako ovo funkcioniše dok ne opišemo neke tehnologije komutacije.

Opšti pregled ATM mreže

Počinjemo opisom izgleda ATM mreže i prikazom nekih njenih komponenti. Na slici 13.20 pokazano je da se ATM mreža sastoji od ATM komutatora. U opštem slučaju, *ATM komutator (switch)** je analogan IP ruteru po tome što je odgovoran za prijem dolazećih poziva i njihovo prosleđivanje dalje preko odgovarajućeg linka.

* Razlika izmedu komutatora i rutera u ovom kontekstu nije definisana. Neki koriste termin *ruter* kada govore o Internetu i njegovoj mogućnosti za povezivanje različitih tehnologija. Drugi koriste termin *komulalor* kada govore o kolekciji homogenih linkova koji povezuju slične tehnologije. Razlika se ogleda u tome da li uredaj može da povezuje različite tehnologije. Medutim, prodavci reklamiraju komutatore kao raznovrsnije uredaje koji mogu da povezuju različite tehnologije, tako da razlika nije najjasnija. Osim toga, prethodno ste videli da se kao *komutator* označava višepormi most koji fimkcioniše na sloju 2. Krajnji zaključak je da na ovom nivou razmatranja te razlike nisu bitne.



SLIKA 13.20 *ATM mreža*

ATM komutatori imaju linkove između sebe, tako da formiraju mrežu sa razlidtim putanjama. Svaki komutator može da poveže različite korisničke uređaje, kao što su kompjuter, televizor, video kamera, ili lokalna mreža. Slika to ne pokazuje, ali svakom korisničkom uredaju je neophodna interfejs kartica koja razume ATM.

Postoje dva tipa linkova u ATM mreži: *interfejs mreža-mreža* (NNI - *netivork-netivork interface)* povezuje dva ATM komutatora, a *interfejs korisnik-mreža* (UNI - *user-nettvork interface)* komutator sa korisničkim uredajem. Postoje neke razlike između linkova; na primer, u načinu kako interpretiraju pozive. Uskoro ćemo i to obraditi. Za sada razlike samo pominjemo da znate da postoje.

Komutacija

Kao što je ranije istaknuto, ATM komunikacija je orijentisana konekciji. Kada neki sajt ima informacije za slanje do drugog sajta, zahteva konekciju slanjem poruke. Poruka prolazi kroz različite komutatore, uspostavljajući virtuelnu putanju na svom putu do odredišta. Za razliku od IP-ja, naredne ćelije sa podacima sadrže *ID virtuelne putanje,* koji komutator koristi za mtiranje ćelije preko odlazećih linkova. Komutator održava tabelu u kojoj svaki zapis ima dva para: ulazni port/ID virtuelne putanje i izlazni port/ ID virtuelne putanje. Kada ćelija stigne preko odredenog ulaznog porta, komutator koristi identifikator porta i ID virtuelne putanje iz ćelije da bi pronašao odgovarajući zapis u tabeli. Nakon toga, menja ID virtuelne putanje ćelije da bude uparen sa pridruženim odlazećim portom i šalje ćeliju na taj port.

Na primer, pretpostavimo da se u zapisu u tabeli nalazi sledeće:

Ulazni po rt	ID virtuelne putanje	Izlazni port	**ID virtuelne putanje**
⋮	⋮	⋮	⋮
C	5	F	8
⋮	⋮	⋮	⋮

Tako ćelija sadrži ID virtuelne putanje 5 kada stiže na port C i prosleđuje se preko porta F uz promenu lD-a virtuelne putanje na 8.

Prcoes je veoma sličan onom koji je opisan na početku prethodnog odeljka i nema potrebe da ponavljamo celu "priču". Jedino vredi istaći razlike u terminologiji. X.2S koristi ID virtuelnog kola u svojim paketima, dok ATM ćelije sadrže ID virtuelne putanje. Na ovom nivou njihova uloga je identična. Međutim, kasnije ćete videti da ATM definiše i virtuelno kolo pored virtuelne putanje.

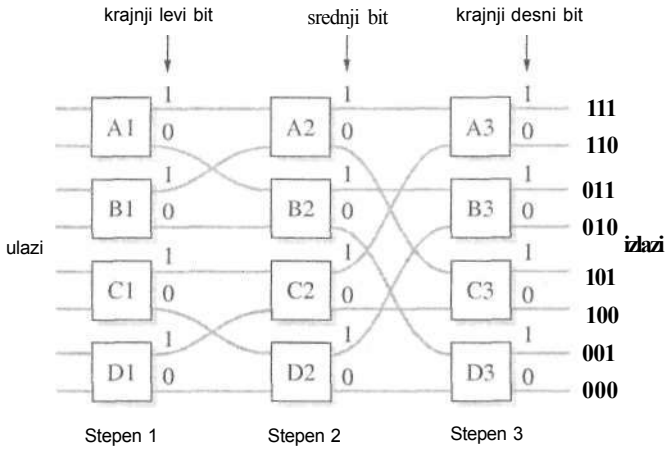
Banyan komutatori Korišćenjem tabela rutiranja komutatori mogu veoma brzo da proslede ćelije, posebno ako su zapisi u tabeli hašovani identifikatorom dolazećeg porta i ID-om virtuelne putanje. Putanje su uspostavljene i oslobođene jednostavno umetanjem i brisanjem novih zapisa iz tabele.

Postoje i druge tehnike za obezbedivanje funkcija komutacije na high-speed mrežama, koje mogu da komutiraju nekoliko ćelija konkurentno. Jedan primer je Banyan komutator. Banyan komutator ima jednak broj ulaznih i izlaznih linija (obično se radi o stepenu broja 2). Unutar komutatora postoji nekoliko stepena. Broj stepena zavisi od broja izlaznih linija. Na primer, ako postoji 2^k izlaznih linija, u komutatoru postoji k stepeni.

Svakoj ćeliji je dodeljen *niz bilova* (još se naziva i *komutacioni string - suniching slring)* dužine *k,* gde svaki bit u stringu vodi ćeliju sa jednog stepena na sledeći. Ovaj string može da se nalazi u ćeliji, ili može da se smesti u tabelu zajedno sa ID-om virtuelne putanje. Njegova lokacija nije bitna za naše razmatranje.

Na slici 13.21 prikazan je Banyan komutator sa osam ulaza i izlaza i tri stepena izmedu njih. Celija može da uđe na bilo koji od osam ulaza i prosleđuje je jedan od četiri komutaciona elementa (prvi stepen). Na osnovu prvog bita u komutacionom stringu, ćelija sledi jedan od dva moguća izlaza u jedan od četiri elementa u narednom stepenu. Tamo logika za komutaciju ispituje drugi bit u komutacionom stringu i prosleduje ćeliju do jednog od dva moguća izlaza ka trećem stepenu. Proces se postavlja sve dok se ne iskoristi poslednji bit, kada se ćelija konačno prosleđuje do jednog od izlaza komutatora.

Svaki izlaz na slici 13.21 označen je jedinstvenim 3-bitnim stringom koji definiše izlaznu liniju preko koje se ćelija prosleduje. Postoji nekoliko interesantnih detalja u vezi ovog komutatora. Prvo, izlaz zavisi samo od komutacionog stringa, a ne od ulazne linije preko koje je ćelija stigla. Na primer, na slici 13.21 istaknute su tri moguće putanje koje odgovaraju ćeliji sa komutacionim stringom 100. Celije koje ulaze preko Al i Bl prenose se na A2 u drugom stepenu. Odatle obe idu do C3.



SLIKA 13.21 *Banyan komutator*

Ćelija koja ulazi na Dl prenosi se do C2, pa do C3. Kao što možete da vidite, svaka ćelija startuje od daigog elementa u prvom stepenu, ali na kraju završava na izlazu koji odgovara 100.

Sledeća interesantna karakteristika je što komutaciona logika može da rukuje sa nekoliko ćelija istovremeno. Na primer, jedna ćelija sa komutacionim stringom 100 prelazi preko Al, A2 i C3. Pretpostavimo da druga ćelija sa komutacionim stringom 110 stiže na D1. Ona će proći preko Dl, C2 i A3. Pošto ove dve ćelije uključuju različite elemente, rutiranje može da se izvede konkurentno. Drugim rečima, jedna ćelija ne mora da čeka drugu. Ovo je izuzetno važna karakteristika, jer omogućava prosleđivanje većeg broja ćelija i time se redukuje vreme čekanja, što je poželjno kod prenosa real-time videa, ili glasa.

Sa daige strane, pretpostavimo da ćelija sa komutacionim stringom 100 stiže na A1, a druga sa komutacionim stringom 101 stiže na B1. Ćelije imaju različite ulaze i izlaze, ali obe moraju da produ preko A2. Ako stignu u isto vreme, doći će do kolizije. Ovde je značajno napomenuti da komutator može da bude sposoban za konkurentni rad sa dve ćelije, ali da nema garancija da će biti uspešan. U slučaju da dođe do kolizije, svaki komutacioni element mora da ima mogućnost donošenja odluke koja se ćelija prva prosleduje, a koja se postavlja u red čekanja na sledeće prenošenje.

Sledeću komutacionu tehnologiju predstavlja unakrsni komutator (crossbar svvitch) sa *n* ulaza i izlaza. Uiazi se obično prikazuju kao nizovi horizontalnih linija, a izlazi kao nizovi vertikalnih linija. Svaka ulazna linija je ukrštena sa svih *n* izlaznih linija, dok je svaka izlazna linija ukrštena sa svih *n* ulaznih linija. Svako ukrštanje odgovara komutacionom elementu koji može da komutira čeliju sa od ulazne do izlazne linije. Nedostatak je što postoji veliki broj komutacionih elemenata.

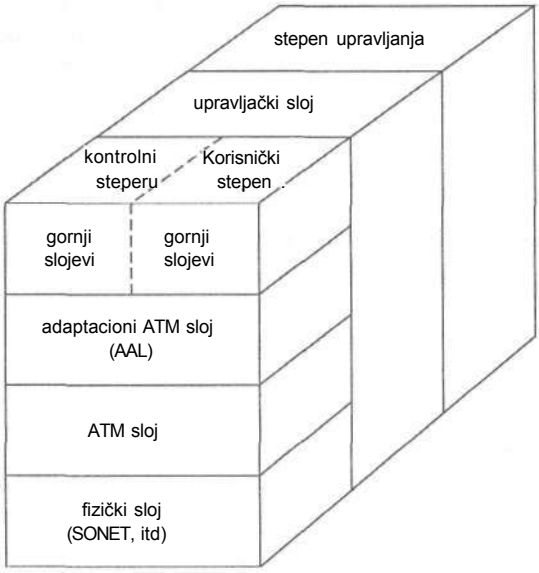
Postoji i *eliminacioni komutalor (knockout switch),* koji predstavlja oblik unakrsnog komutatora. Kod unakrsnog komutatora je problem što istovremeni dolazak dve ćelije namenjene istom izlazu izaziva kolizije.

Osim toga, ćelije mogu da stignu na različite ulazne linije i prelaze preko različitih komutacionih elemenata do željene izlazne linije, tako da je rešavanje kolizije još teže. Kod eliminacionog komutatora svaki izlaz ima *arbitra* koji je povezan na svaku ulaznu liniju. Ako su dve ćelije sa različitim ulazima namenjene istom izlazu, unakrsni komutator ih rutira do arbitra, koji odlučuje koja ćelija ide u red čekanja, a koja se šalje na izlaz.

Referentni model

ATM je, u stvari, deo B-ISDN specifikacije koju je defmisao ITU-T. Osim toga, međunarodna neprofitna organizacija ATM Forum (www.atmforum.com) ima tehnički komitet koji sarađuje sa agencijama za donošenje standarda radi izbora odgovarajućih standarda i preporuku novih. Komitet takode nastoji da se obezbedi interoperabilnost između proizvodača koji plasiraju ATM proizvode.

Na slici 13.22 prikazan je slojeviti referentni model. Naša namera je da damo kratki opšti pregled modela, a zatim da predemo na detaljnije razmatranje relevantnih tema. Prvo što ćete možda primetiti kod ovog modela u poređenju sa ostalima je dnjenica da ima trodimenzionalni izgled. To je zbog toga što predstavlja kombinovani ATM/B-ISDN model koji definiše i korisničke i upravljačke funkcije. Na primer, *kontrolni stepen (control plane)* definiše način uspostavljanja i oslobađanja konekcija. *Korisnički stepen (user plane)* definiše transport podataka i odgovarajuće detalje, kao što su kontrola toka, detekcija i korekcija grešaka. Iza scene, *upravljački sloj (layer management)* obezbeđuje upravljačke funkcije.



SLIKA 13.22 *ATM/B-ISDN referentni model*

On je odgovoran, izmedu ostalog, za obezbeđivanje operacija, administraciju i održavanje (OAM servisi) preko informacionih paketa koje komutatori razmenjuju da bi sistem efikasno funkcionisao. Upravljački servisi mogu da se obezbede i preko SNMP-ja, ili CMIP-ja. Konačno, *upravljački stepen (plane management)* osigurava da različiti stepeni ispravno koordinišu svoje aktivnosti.

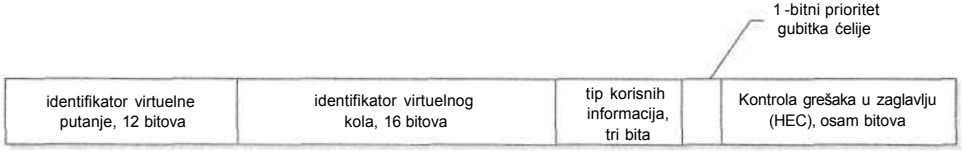
Na dnu modela nalazi se fizički sloj koji defimše fizičke karakteristike prenosa. Iako ATM ne defmiše pravila za fizički prenos, originalno je dizajniran za pokretanje preko SONET-a (Synchronous Optical Network - sinhrona optička mreža), Originalno predložen od Bellcorea (Bell Communications Research), SONET je optička mreža koja koristi multipleksiranje sa podelom vremena da bi bili omogućeni simultani kanali. Kontrola je ostvarena pomoću takta i bitovi se prenose brzinama od oko 155,5 Mbps do ekstremna 2 Gbps. Defmiše način na koji telefonske kompanije prenose svoje podatke preko optičkih mreža. Osim SONET-a, ATM može da se pokreće i preko FDDI, Tl i T3 sistema, i sa zaštićenim i nezaštićenim upredenim paricama, pa, čak, i sa bežičnim medijumima (ref. [Va97]).

Iznad fizičkog sloja nalazi se *ATM sloj.* On izvršava aktivnosti slične onima koje se nalaze na slojevima 2 i 3 OSI modela. Na primer, definiše format ćelije i način na koji se tretiraju informacije iz zaglavlja. Osim toga, odgovoran je za postavljanje i oslobađanje konekcija i uspostavljanje virtuelnih kola i virtuelnih putanja (uskoro ćemo objasniti razliku). Konačno, izvršava i kontrolu zagušenja.

Adaptacioni ATM sloj (AAL - ATM adaptation layer) obezbeđuje interfejs između aplikacija i ATM sloja, a podeljen je na dva podsloja: podsloj za segmentaciju i ponovno sastavljanje (niži podsloj) i podsloj konvergencije. *Podsloj konvergencije* obezbeđuje interfejs za različite aplikacije koje koriste ATM. Njegov zadatak zavisi od konkretne aplikacije i tipa saobraćaja koji generiše. U stvari, kasnije ćete videti da postoji nekoliko verzija AAL-a, u zavisnosti od toga da li je saobraćaj nekompresovani video, kompresovani video, ili neki drugi tipovi. *Podsloj segmentacije i ponovnog sastavljanja* smešta informacije sa viših slojeva u ATM ćelije i može da doda sopstveno zaglavlje medu korisne informacije. Na prijemnoj strani izvlači korisne informacije iz svake ćelije i ponovo ih sastavlja u niz informacija koje aplikacija dalje obraduje.

Definicija ćelije

Ovo je dobar trenutak da navedemo neke detalje o načinu funkcionisanja ATM-a. Logično je startovati defmisanjem ATM ćelije i opisom načina na koji ATM upravlja njenim sadržajem. Kao što je ranije pomenuto, ATM ćelija ima 53 bajta, od kojih su 48 korisne informacije. To znači da zaglavlje ćelije ima samo pet bajtova. Na slici 13.23 opisani su dok prolaze kroz NNI.



SLIKA 13.23 *ZaglavljeATM ćelije (NNl)*

Za ćelije koje prelaze kroz UNI postoji neznatna razlika u interpretaciji sadržaja zaglavlja. U drugom slučaju prva četiri bila defmišu polje Generic Flow Control (GFC), tako da preostaje samo osam bitova za identifikator virtuelne putanje. Medutim, kada se ćelija nađe na mreži, komutatori mogu da upisuju informadje preko GFC polja, koristeći 12-bitni ID virtuelne putanje.

Polje GFC je dizajnirano za kontrolu toka saobraćaja od uređaja do ATM mreže (ne i u suprotnom smeru). U suštini, postoje dve klase konekcija, kontrolisane i nekontrolisane, koje su ili deo konfiguracije, ili pregovora u toku uspostavljanja konekcije. Kada je konekcija *kontrolisana,* mreža obezbeduje informacije za korisniđd uređaj o tome koliko ćelija može da pošalje. Ovo pomalo liči na mehanizam kredita koji se koristi za kontrolu toka kod TCP protokola. Kod *nekontrolisanih* konekcija mreža jednostavno dopušta, ili zabranjuje slanje ćelija. Kada je slanje dopušteno, korisnički uređaj može da šalje ćelije sve dok to mreža ne zabrani. To liči pomalo na X-ON/X-OFF kontrolu toka.

Mali broj bajtova u zaglavlju utiče na brzinu ATM-a: potrebno je proveriti manje "stvari" u svakoj ćeliji koja stiže u komutator. Pogledaćemo sada svako polje ponaosob.

Polje Header Error **Control** Polje Header Error Control (HEC) je donekle jasno samo po sebi: obezbeduje proveru grešaka. Međutim, postoji nekoliko elemenata koje vredi istaći. Za početak, obezbeduje kontrolu grešaka samo za druga četiri bajta zaglavlja, tako da korisne informacije ostaju nezaštićene. To, naravno, proveru grešaka i, samim tim, isporuku ćelije čini mnogo bržim (zapamtite, za ATM mreže brzina prenosa ima suštinski značaj). Nezaštićenost korisnih informacija možda na prvi pogled deluje kao problem, ali, zbog nekoliko razloga, to nije tako. Jedan razlog je što je ATM dizajniran za pokretanje preko optičkih fiber sistema, koji su izuzetno pouzdani. Zato je verovatnoća za pojavu grešaka veoma mala. Čak i ako dođe do greške, ona se može detektovati na višem sloju, ako je to neophodno. Sledeći razlog je što mnoge aplikacije uključuju video, ili audio zapise. Zbog toga se greške u okviru manifestuju u deliću sekunde, što nije ni primetno, ili, u najgorem slučaju, može da izazove jedva primetno treperenje. Da li u ovakvim situacijama ima smisla čekati na retransmisiju takvog okvira? Odgovor je negativan. Medutim, zaštita zaglavlja je od suštinskog značaja, jer se u njemu nalaze identifikatori virtuelne putanje i kola, koji određuju kuda ćelije idu.

Takođe je bitno što se za proveru grešaka koristi adaptivna tehnika zasnovana na CRC metodu. Može da detektuje više od 90 odsto grešaka na više bitova, a može, čak, i da ispravi greške u jednom bitu. Ova mogućnost ispravke je posebno značajna, jer je studija koju su AT&T i Bellcore objavili 1989. godine (ref. [AT89]) pokazala da više od 99,5 odsto grešaka u optičkim fiber sistemima čine greške u jednom bitu.

Polje HEC izvršava još jednu "nevezanu" funkciju. Sećate se da smo u odeljku 9.2, prilikom predstavljanja protokola veze, istakli da okviri sadrže specijalni bitski uzorak koji označava početak okvira. ATM ne definiše takav uzorak, tako da se nameće logično pitanje kako protokoli niskog nivoa mogu da detektuju početak ćelije. Ovo zavisi delom od fizičkog sloja. Na primer, SONET enkapsulira ATM ćelije u envelopu (slično ranijim protokolima kod kojih su paketi enkapsulirani u okvir, osim što se u omotu može naći nekoliko okvira). Informacije u envelopi lociraju start prve ATM ćelije.

Ipak, svi fizički slojevi ne obezbeduju ovakvu envelopu. U stvari, sa sinhronim medijumom, ćelije moraju da se prenose u regularnom uzorku, koji je defmisan taktom. Ne postoje envelopa, niti specijalni bitski uzorci koji bi locirali stan ćelije. Celije jednostavno stižu u regularnim intervalima i prijemni uređaj mora da bude sposoban da utvrdi njihove granice. Prepoznavanje granica ćelije se naziva *definisanje okvim (framing),* kao i u slučaju kada se koriste sinhroni bilski uzorci. Na primer, razmotrimo sekvencu ATM ćelija koje sadrže zapise iz video kamere. Prijemni uredaj mora da se sinhronizuje lociranjem granica ćelije da bi bilo obezbedeno ispravno prikazivanje.

lako ne posloji poseban bitski uzorak za identifikovanje start ATM ćelije, jedna tehnika koristi činjenicu da sva neoštećena zaglavlja imaju nešto zajedničko: primenom metoda za proveru grešaka koji koristi 40-bitni string za defmisanje zaglavlja generiše se vrednost koja je konzistentna sa poslednjih osam bitova u tom stringu. Moguća tehnika za lociranje granica ćelije može da se opiše na sledeći način:

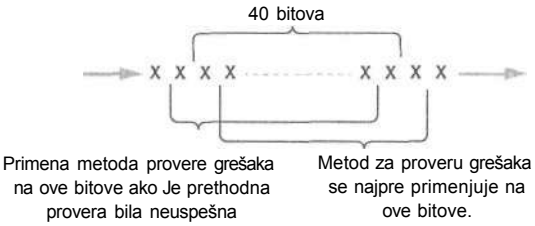
1. Primenjuje se metod za proveru grešaka sa 40 uzastopna bita. Ako to ne generiše rezultat koji je konzistentan sa poslednjih osam bitova, pomera se jedan bit i pokušava se ponovo (slika 13.24).

2. Korak 1 se ponavlja sve dok se ne pronade konzistentan rezultat. Ovo ukazuje da se tih 40 bitova mogu koristiti kao legitimno zaglavlje. Ipak, i slučajnost može da dovede do pronalaženja konzistentnog rezultata među 40 drugih bitova (pogrešno zaglavlje).

3. Kada se potencijalno zaglavlje pronađe, "preskače" se narednih 48 bajtova (korisne informacije) i ista tehnika se primenjuje na narednih 40 bitova. Pretpostavljamo da su legitimna zaglavlja razdvojena sa po 48 bajtova. Ako tehnika ne obezbedi konzistentan rezultat, onda je prediodni rezultat bio slučajan i mora se krenuti iz početka.

4. Pretpostavimo da smo pronašli nekoliko 40-bitnih stringova, međusobno razdvojenih sa po 48 bajtova, i da su svi generisali konzistentnu HEC vrednost. U tom slučaju postoji visoka verovatnoća da su sva ta zaglavlja legitimna - defmisanje okvira bilo je uspešno i prijemni uređaj je sinhronizovan sa dolazećim ćelijama.

Upravo opisana tehnika zvuči složeno, ali sa cikličnim pomeračkim registrima, sličnim onima koje smo opisali u odeljku 4.3, implementacija može da se izvede efikasno. Osim toga, postavlja se pitanje koliko uzastopnih potencijalnih zaglavlja treba tražiti pre nego što se zaključi da je uređaj sinhronizovan. Pošto polje HEC ima samo osam bitova, verovatnoća lociranja pogrešnog zaglavlja je $1/2^8 = 1/256$ za konkretni 40-bitni string.



SLIKA 13.24 *Traženje granica ćelija*

Verovatnoća lociranja dva uzastopna pogrešna zaglavlja iznosi $1/(256)^2$. U opštem slučaju, verovatnoća lociranja n pogrešnih zaglavlja iznosi 1/(256)". Za veće vrednosti *n* treba više vremena, ali je manja verovatnoća da će doći do pogrešne sinhronizacije. U stvari, ako je n samo 4, verovatnoća pogrešne sinhronizacije je 1 prema 4,3 milijarde.

Bit CeIl Loss Priority Bit CeIl Loss Priority (CLP) ukazuje na prioritet ćelije. Uvek kada dođe do zagušenja, ATM ima opciju za brisanje ćelija da bi bilo ublaženo zagušenje. Najpre se biraju ćelije čija je CLP vrednost 1. Aplikacija treba da utvrdi koje ćelije nisu kritične. Na primer, MPEG-kompresovani video koristi razlike između okvira i stvarnih kompresovanih okvira (setite se razlike između 1 i P okvira iz odeljka 5.7) u prenetim slikama. Ako se izbrišu okviri koji daju veoma male promene, krajnji efekat će prilikom prikazivanja biti neprimetan. Zato aplikacija u ćelijama sa tim okvirima mora da postavi CLP bit na I.

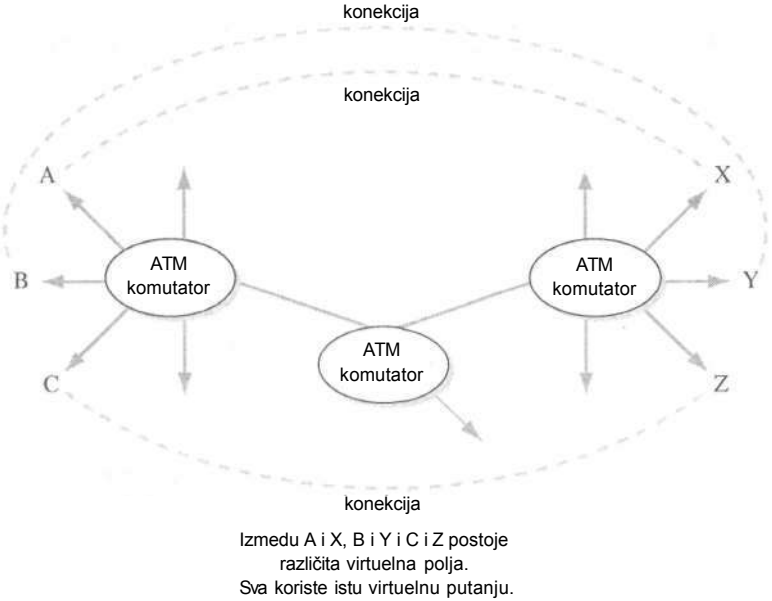
Polje Payload Type Polje Payload Type obezbeduje neke specifične informacije o ćeliji. Krajnji levi bit definiše da li se u korisnim informacijama nalaze korisnički podaci, ili OAM informacije. Komutatori mogu da razmenjuju OAM poruke da bi bile izvršene provere statusa i omogućeno "glatko" funkcionisanje sistema. Drugi bit ukazuje da li se ćelija prenosi preko zagušenih komutatora. Tako se primalac obaveštava o eventualnim problemima zbog zagušenja na određenoj putanju. U nekim slučajevima treći bit može da se koristi da bi bila označena poslednja ćelija u sekvenci ATM ćelija.

Virtuelna kola i putanje

Sve do sada nismo pravili nikakve razlike između virtuelnog kola i virtuelne putanje. Ovo je promenjeno sa ATM-om, kao što je prikazano na slici 13.25. Konceptualno, *virtuelno kolo* predstavlja logičku konekciju između dve krajnje tačke. Na primer, na slici 13.25 postoji logička konekcija izmedu A iX, BiYiCiZ. Medutim, svaka konekcija odgovara istoj virtuelnoj putanji.

Svaka ćelija sadrži 12-bitni ID virtuelnog kola (osam bitova preko UNl-a) i 16-bitni ID virtuelne putanje. Dakle, iz perspektive korisnika, svaka konekcija može da se defmiše 24-bitnim identifikatorom konekcije, koji se satoji iz dva dela: ID kola (osam bitova) i ID putanje (16 bitova). Ovo je analogno Intemet adresiranju, gde smo naznačili da se 32-bitna adresa sastoji od ID-a mreže i ID-a hosta. Prednost je u tom slučaju bila u činjenici da su interni mehanizmi za rutiranje mogli u potpunosti da rutiraju podatke isključivo na osnovu ID-a mreže. ID hosta nije korišćen sve dok paket ne stigne do odredišne mreže, odakle može da se šalje direktno do hosta.

strana 716 Slično tome, ATM komutatori prosleđuju ćelije isključivo na osnovu 16-bitnog ID-a virtuelne putanje. Donošenje odluka o rutiranju ATM ćelija na osnovu 16-bitnih identifikatora, umesto na osnovu kompletnog 28-bitnog identifikatora, ubrzava proces komutacije (kada ćelija uđe na mrežu, ID kola se proširuje na 12 bitova), tako da je omogućeno brže komutiranje. Ovo je glavni cilj ATM-a. Korišćenje 16-bitnih, umesto 28-bitnih brojeva, omogućava čuvanje manjih internih tabela za komutaciju (maksimalno 2^{16} zapisa, umesto 2^{28} zapisa). ID kola je neophodan samo kada ćelija stigne do poslednjeg ATM komutatora, koji mora da prosledi ćeliju direktno do korisnika.

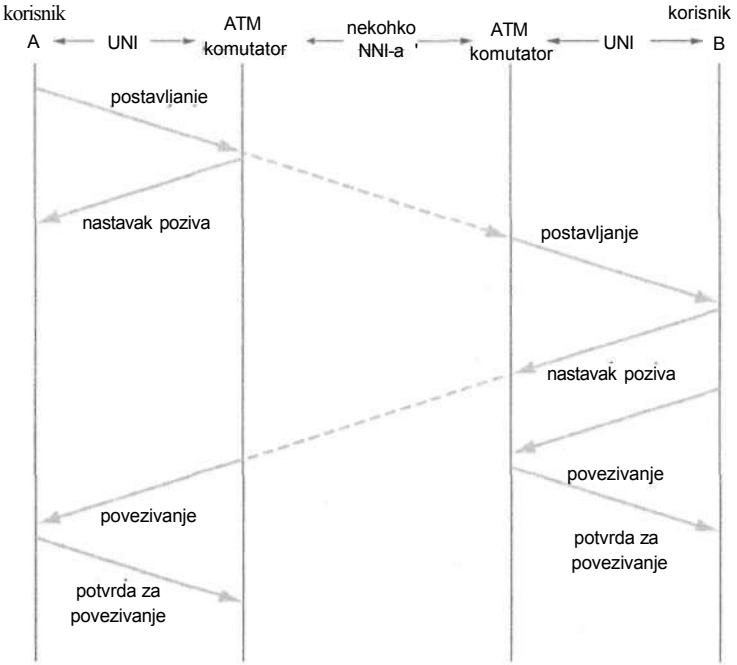


SLIKA 13.25 *Virtuelna kola i virtuelne putanje*

Sledeća prednost korišćenja ID-a kola i ID-a putanje prepoznaje se kada dode do problema na linku. Na primer, pretpostavimo da je uspostavljeno stotinu konekcija i da sve koriste istu putanju. Ako na nekoj putanji dođe do problema, mora se pronaći nova. Ako se komutacija zasniva na 28-bitnom broju, onda će interne tabele za svaki novi komutator na altemativnoj ruti zahtevati nove zapise za svaku konekciju. U ovom slučaju moralo bi da se izvede stotinu promena u svakoj tabeli. Ipak, pošto se komutacija zasniva samo na ID-u putanje, sve konekcije koriste istu putanju i svaka tabela zahteva samo jednu promenu da bi bile zadovoljene sve konekcije. Jasno, ovo je mnogo brže.

Upravljanje konekcijom

Postoje dva tipa konekcija: *permanentno virtuelno kolo* i *komutirano virtuelno kolo.* Permanentno kolo je analogno iznajmljenoj telefonskoj liniji, a komutirano mora da se uspostavi pomoću protokola konekcije. Protokol konekcije je zasnovan na algoritmima u ITU-T Q.2931 (predstavlja podskup od Q.931). Na površini, uspostavljanje ATM konekcije sledi procedure koje podsećaju na one koje se koriste za uspostavljanje konekcija u drugim protokolima koje smo do sada razmatrali. Jedna strana inicira zahtev za poziv, naznačavajući neke poželjne atribute konekcije, i čeka na potvrdu. Kontrolni stepen je odgovoran za postavljanje konekcije.



SLIKA 13.26 *Uspostavljanje konekcije*

Na slici 13.26 opisana je logika uspostavljanja konekcije kada A šalje zahtev za konekciju do B. Prilikom postavljanja konekcije koriste se četiri različita tipa pomka; u sledećim koracima opisan je način na koji se koriste.

1. U ime A, kontrolni stepen šalje *Setup pomku* do B preko UNI-a. Ova poruka sadrži informacije koje su relevantne za zahtevanu konekciju. Na primer, sadrži adresu korisnika B i neke druge stavke koje čemo uskoro opisati.

2. Komutator koji primi Setup poniku šalje *CaII Proceeding poruku* nazad do A da bi ga informisao da je zahtev primljen i da se obrađuje. Ova poruka sadrži i lD-e virtuelnog kola i putanje, koji će se koristiti kada se konekcija uspostavi. Osim toga, komutator prosleđuje Setup poaiku do B.

3. Setup poruka "putuje" kroz mrežu preko različitih NNI-a ka B, koristeći algoritam za rutiranje koji je implementiran (ATM ne definiše koji se algoritam koristi). Ruta koja se izabere defmiše virtuelnu putanju i kolo. Setup poruka nastavlja da "putuje" kroz komutatore; svaki od njih reaguje vraćanjem CaII Proceeding poruke do komutatora koji mu je poslao Setup poruku. Zahvaljujući tome, svi komutatori koji su eventualno deo virtuelne putanje znaju da je konekcija u toku.

4. Na kraju, B dobija Setup poruku. B može odmah da odgovori slanjem *Connect poruke,* ili, ako očekuje kašnjenje, može da vrati Cali Proceeding poruku. Ako sve prode kako treba, B šalje Connect poruku preko UNI-a do komutatora. Komutator vraća *Connect Acknoivledgment poruku.* B je sada slobodan da počne slanje informacija ka A. Connect poruka "putuje" nazad do A preko istih komutatora (suprotnim redosledom) kojim je stigla Setup ponika. Svaki komutator reaguje prosledivanjem poruke i vraćanjem Connect Acknowledgment poaike. Kada svi komutatori dobiju Connect poruku, znaju da su deo virtuelne putanje i kreiraju odgovarajući zapis sa ID-om putanje u svojim tabelama. Komutator je sada pripremljen za rutiranje narednih ćelija koje će biti prosledene preko odgovarajuće putanje.

5. Na kraju Connect poruka stiže do A, koji reaguje sa Connect Acknowledgment porukom. A sada može da šalje informacije do B.

Na sličan način se izvodi i otpuštanje konekcije. Bilo koja strana može da inicira otpuštanje slanjem *Release poruke.* Poruka "putuje" preko virtuelne putanje, a svaki komutator vraća *Release Complete poruku.* Komutatori ovo koriste za uklanjanje informacija iz njihovih tabela i prosleduju Release ponrku do sledećeg komutatora na putanji.

Parametri konekcije Ranije smo istakli da Setup poruka sadrži informacije koje su relevantne za zahtevanu konekciju. U opštem slučaju, defmiše neophodni]<valitet servisa i tip očekivanog saobraćaja. Na primer, prilikom prenosa videa na zahtev postoje viši kriterijumi za kvalitet konekcije u poređenju sa kriterijumima za konekciju koja treba da obezbedi transfer običnih fajlova. Ovo je značajna informacija u okviru Setup poruke, jer je jedan od ciljeva ATM-a održavanje traženog kvaliteta servisa.

Kada se Setup poruka nade na mreži, svaki komutator mora da proceni da li može da bude deo zahtevane putanje bez ugrožavanja kvaliteta servisa na postojećim putanjama. Preko istog komutatora može da ide više putanja za video koji se prikazuje na zahtev. Komutator nastavlja na ranije opisani način, ili odbacuje Setup zahtev. Ovde je složeniji deo procesa. Moguće je da je neophodno ispttati više mogućih putanja pre nego što se pronađe ona koja obezbeđuje traženi kvalitet servisa. Naravno, moguće je da takva putanja neće ni biti pronadena. Clij je da se izbegne uspostavljanje putanja koje će imati negativan uticaj na postojeće korisnike, uključujući onog koji pokušava da uspostavi konekciju. Efektivno, ovaj metod garantuje korisnicima određeni kvalitet servisa i u neku mku može da se smatra oblikom kontrole zagušenja.

Frazu *kvalhet servisa* do sada smo koristili uglavnom u nekom opštem smislu. Sledi lista specifičnih stavki koje Setup poatka može da sadrži.

- Klasa B-ISDN servisa B-ISDN defmiše ćetiri klase. *Saobraćaj Klase A* zahteva konstantnu bitsku brzinu i stogu sinhronizaciju izmedu pošiljaoca i primaoca. Ovo je neophodno za transfer nekompresovanog videa i audia, gde se slike moraju snimiti, preneti, a, zatim, prikazivati (ili čuti) dok se budu primale. *Klasa B* dopušta promenljivu bitsku brzinu, ali i dalje zahteva sinhronizaciju pošiljaoca i primaoca. Ovo je tipično za kompresovani video, kod koga se bitske brzine razlikuju u zavisnosti od toga u kojoj je meri kompresija izvršena. Ipak, sinhronizacija je i dalje neophodna, jer se zapisi prikazuju u realnom vremenu. *Klasa C* je namenjena konekciji orijentisanom saobraćaju bez vremenskih ograničenja. *Klasa D* predstavlja servis obezbeden bez uspostavljanja konekcije, bez vremenskih ograničenja. Poslednje dve klase više su

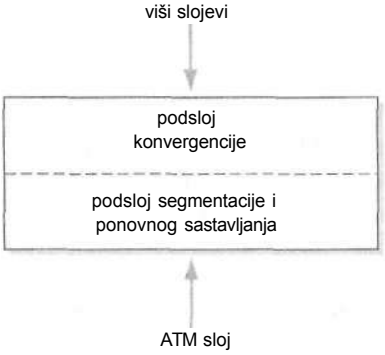
namenjene aplikacijama koje obezbeđuju prenos podataka i fajlova, nego aplikacijama kod kojih postoje ograničenja, zbog izvršavanja u realnom vremenu.

- **Maksimalna brzina ćelija (Peak cell rate)** To je maksimalna brzina kojom će se ćelije slati.
- **Podržana brzina ćelija (Sustainable cell rate)** To je gornja granica prosečne brzine kojom se ćelije šalju u određenom periodu. Maksimalna brzina prenosa ćelija može da bude viša **u** kraćim periodima, ali "na duže staze" ne premašuje brzinu koja je ovde definisana.
- **Vreme (Time)** To je parametar koji se koristi za utvrđivanje maksimalne podržane brzine ćelija.
- **Minimalna brzina ćelija (Minimum cell rate)** Definiše najnižu brzinu kojom se ćelije moraju primati. Sve što se nalazi ispod ove brzine smatra se neprihvadjivim.
- **AAL verzija (AAL version)** Ovo je sledeća tema koju ćemo obraditi.

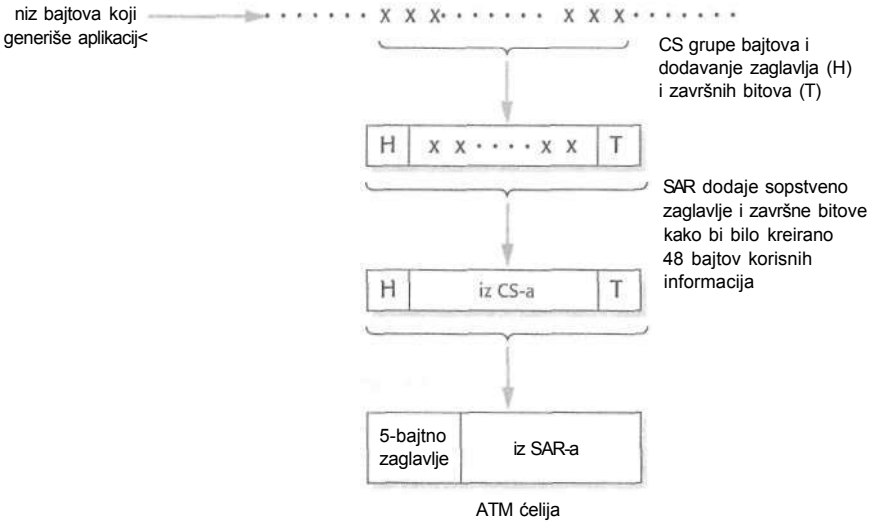
Adaptacioni slojevi

Odeljak o ATM-u zaključujemo opisivanjem **ATM adaptacionog sloja (AAL—ATM adaptation Iayer**). Na slici 13.27 prikazano je da postoji interfejs između viših slojeva i ATM sloja i da se sastoji iz dva podsloja: *podloja segmentacije i ponovnog sastavljanja* (SAR) i *podsloja konvergencije* (CS). Odgovornost podsloja konvergencije zavisi od tipa informacija koje generišu aplikacije na višim nivoima.

strana 720 Na slici 13.28 prikazane su opšte akcije koje se izvode na podslojevima konvergencije **i** SAR-a. Nameravamo da ovo bude opšti pregled; postoje različiti tipovi AAL-a koji mogu, **ali** ne **moraju da** izvrše neke od ovih zadataka. Llskoro ćemo uzeti u obzir te razlike. U opštem slučaju, neke aplikacije **na** višem nivou generišu podatke koje posmatramo kao niz bajtova. Podsloj konvergencije izvlači neke od tih bajtova i dodaje zaglavlje i završne bitove (trailer), tako da kreira sopstveni CS paket. SAR podsloj dobija CS paket i dodaje sopstveno zaglavlje i završne bitove, tako da se kreira 48 bajtova korisnih informacija koje se smeštaju u ATM ćeliju. Na osnovu slike može da se kaže da oba sloja rade isto, po tome što dodaju zaglavlja i dopune na podatke primljene od višeg sloja. Dobro, to jeste tako, ali postoje razlike u sadržaju polja koje dodaju.



SLIKA 13.27 *AAL podsloievi*



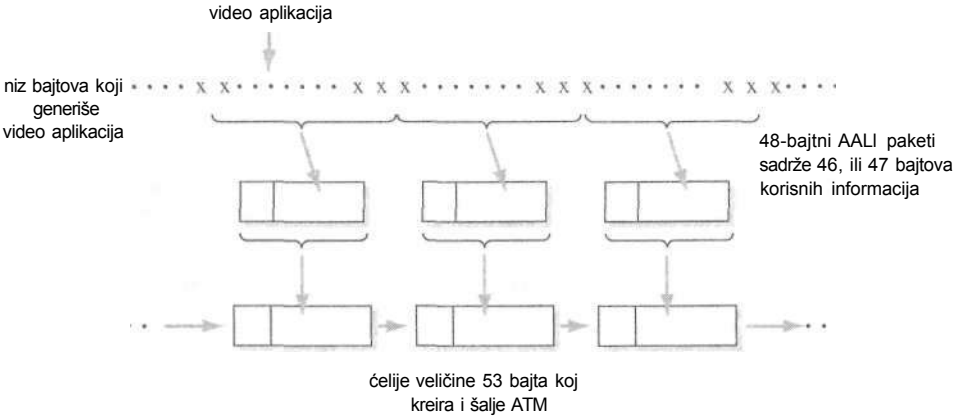
SLIKA 13.28 *Opšte AAL aktivnosti*

Međutim, da bismo istakli razlike, predstavićemo različite AAL tipove.

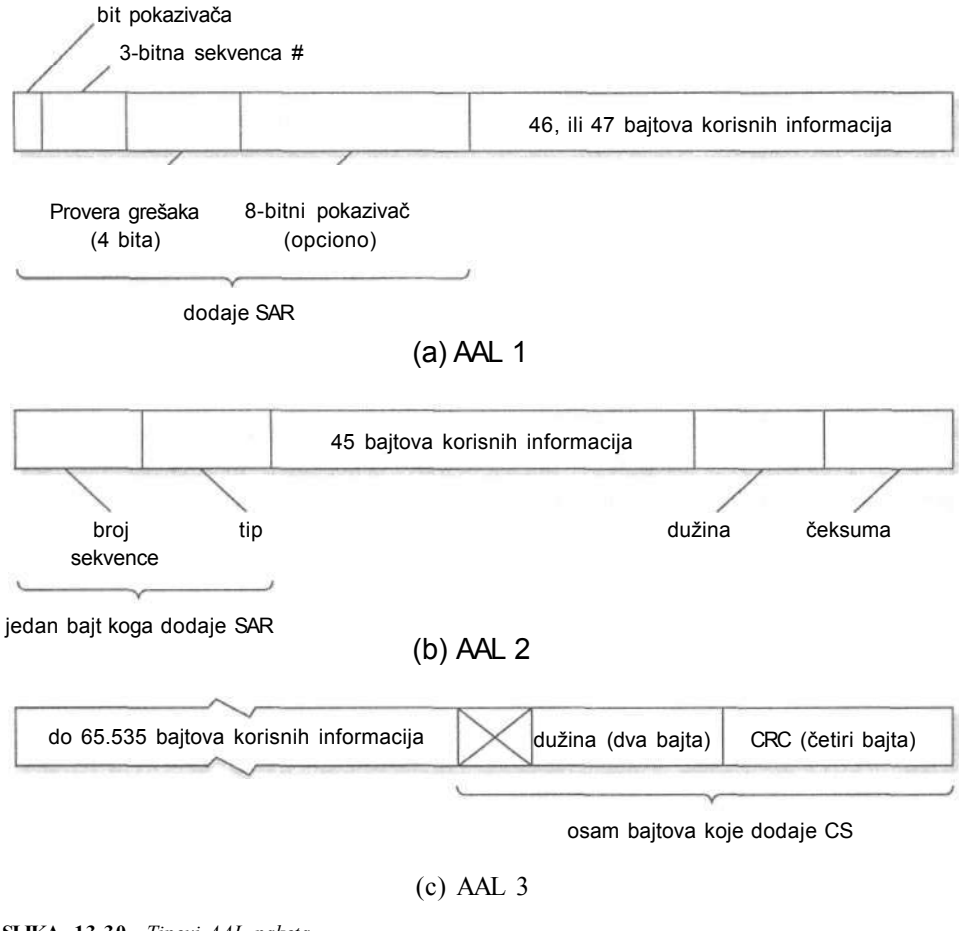
AAL tipovi se razlikuju prvenstveno po klasama saobraćaja koje kontrolišu. AAL 1 se bavi saobraćajem Klase A, a AAL 2 saobraćajem Klase B. Na osnovu ovoga, može da se zaključi da se AAL 3 i AAL 4 bave saobraćajem Klase C i D. U određenoj meri to je tačno, jer je ITU-T razvio AAL 3 i AAL 4. Ipak, kako je razvoj napredovao, postalo je očigledno da nema značajnih razlika između tih slojeva, tako da je ITU-T odlučio da ih kombinuje. Neki čitaoci mogu da pomisle da se u tu svrhu koristi oznaka AAL 3.5, ali, u stvari, naziva se AAL 3/4. Nakon što je AAL 3/4 razvijen, neki su se brinuli zbog onoga što su smatrali neefikasnim. Zato je AAL 5 razvijen kao naslednik AAL 3/4. Mi se nećemo baviti sa AAL 3, AAL 4, ili AAL 3/4. U referenci [Bl95] razmotrene su sve AAL verzije. Ipak, ukratko ćemo predstaviti AAL 1, AAL 2, i AAL 5.

AAL 1 Na slici 13.29 pokazano je kako se upravlja saobraćajem Klase A. Video aplikacija generiše nekompresovani niz bajtova u realnom vremenu. AAL 1 uzima 46, ili 47 bajtova u jednom trenutku i postavlja ih u AAL 1 paket. Razlika u odnosu na sliku 13.28 je to što samo SAR podsloj kreira zaglavlje, dugačko jedan, ili ili bajta. Zbog toga, svaki paket predstavlja korisne informarije ATM ćelije koja se, nakon toga, prenosi ka svom odredištu.

Na slici 13.30a prikazan je format AAL 1 paketa. Prvi bit ukazuje da li je prisutan drugi bajt zaglavlja.



SLIKA 13.29 *Slanje saobraćaja Klase A*



SLIKA 13.30 *Tipovi AAL paketa*

Ovaj bajt zaglavlja, ako je prisutan, koristi se kada polje sa korisnim informacijama (Payload) nije puno i locira podatke u tom polju. Ovaj metod može da se koristi kada popunjavanje polja Payload može da traje duže nego što aplikacija može da toleriše. Parcijalnim popunjavanjem polja Payload informacije mogu da se prenesu u ATM ćeliju i eventualno brže pošalju. To može da bude neophodno kako bi se održala konstantna bitska brzina koja se zahteva za saobraćaj Klase A. Posledica je da podsloj konvergencije mora da bude sinhronizovan sa taktom da bi odgovarajuća količina podataka pravovremeno bila isporučena SAR podsloju.

Od preostalih sedam bitova prvog bajta zaglavlja, tri se koriste za broj sekvence. Podsloj konvergencije obezbeđuje broj sekvence, a SAR podsloj kreira zaglavlje u kome ga pamti. Tako prijemni CS podsloj detektuje gubitak informacija, ili pogrešno umetnutu ćeliju, što može da se desi ako se javi nedetektovana greška na ID-u putanje ćelije na putu do odredišta. Ćelija može da bude usmerena na pogrešnu lokaciju, tako da bi se greška mogla manifestovati u vidu pogrešno umetnute ćelije. Tačno odredište će primetiti prazninu u pristiglim brojevima sekvence koja ukazuje na gubitak informacija. Sve pogrešno umetnute ćelije se ignorišu (zamislite da vidite pogrešne kadrove u poslednjim odlučujuđm sekundama plejofa). Ako primalac detektuje gubitak, obaveštava pošiljaoca, ali ne zahteva retransmisiju. Pretpostavljamo da radite sa real-time video, ili audio aplikacijom - zato retransmisija slika, ili zvukova ne bi bila korisna. Gledaoci će morati da lolerišu treperenje slike, ili šuštanje zvtika (uz pretpostavku da je gubitak jedva primetan).

Preostala četiri bita se koriste za ptoveru grešaka 3-bitnog polja Sequence. Obezbeduje ih SAR podsloj i tri od četiri bita odgovaraju CRC-u za tri bita broja sekvence, a četvrti je bit parnosti za kombinaciju broja sekvence i CRC bitova. Bit parnosti obezbeduje dodatnu meru zaštite. U stvari, može da ispravi greške u jednom bitu i da detektuje dvostruke greške. Korisne informacije u paketu nisu zaštićene. Međutim, kao što smo ranije istakli, ATM se u opštem slučaju pokreće na pouzdanom medijumu. Čak i ukoliko dođe do gubitka podataka, verovatno je neprimetan. Opsims toga, provera samo polja Sequence zahteva manje vremena i pomaže pravovremenu isporuku paketa u realnom vremenu.

AAL 2 Na slici 13.3Ob prikazan je format AAL 2 paketa. Slično AAL 1, jedino SAR podsloj dodaje nešto u korisne informacije. U ovom slučaju dodaje i zaglavlje (jedan bajt) i završne bitove (dva bajta). Broj sekvence u zaglavlju ima istu namenu kao i broj sekvence u AAL 1 paketu. Polje *Type* odražava promenljivu bitsku brzinu karakterističnu za saobraćaj Klase B. Prilikom saobračaja Klase A podaci su striktni nizovi bitova kod kojih nema potrebe za definisanjem granica poruke. Slike se prikazuju čim se podaci koji ih predstavljaju prime. Pošti saobraćaj Klase B uključuje nekompresovane slike, granice poruka su neophodne da bi se olakšalo identifikovanje novih okvira. Polje Type olakšava identifikovanje poruke, tako što naznačava da li je ćelija prva, poslednja, ili je u sredini poruke.

Polje *lenglh* definiše broj bajtova u korisnim informacijama. Konačno, polje *Checksum* obezbeduje proveru grešaka za ceo paket. AAL 2 se i dalje razvija, tako da postoje aspekti koji još uvek nisu u potpunosti definisani.

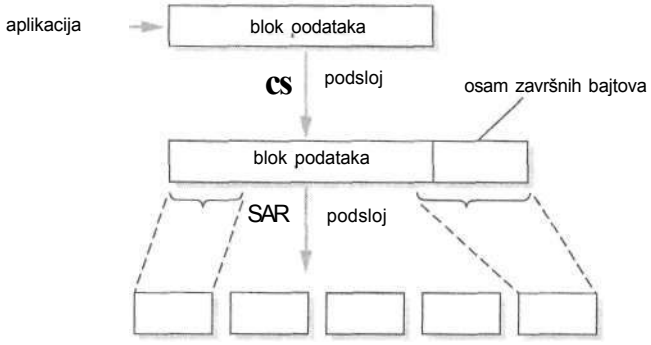
Na primer, veličina dodatnih polja (Overhead) nije defmisana. Postojali su predlozi da se u polje Type uključe i informacije o tajmingu relevantne za audio, ili video podatke.

AAL 5 Razmatranje ATM-a završavamo opisom AAL 5, čiji je format paketa opisan na slici 13.30c. AAL 5 paketi su znatno veći od ostalih AAL paketa. Razlog je delimično to što AAL 5 nije dizajniran za real-time video, ili glas; dakle, potreba za malim paketima nije mnogo kritična. Sledeća razlika se ogleda u tome što dodatne podatke u korisne informacije dodaje podsloj konvergencije, umesto SAR podsloja. Na slici 13.31 prikazani su osnovni koraci prilikom slanja potencijalno velikog bloka podataka. Aplikacija generiše blok podataka i predaje ga CS-u, koji dodaje osam bajtova završnih podataka i prenosi paket do SAR podsloja. SAR podsloj deli paket na 48 bajtova korisnih informacija, koji se postavljaju u ATM ćeliju. Koraci su obrnuti na prijemnoj strani.

Primećujete da slika 13.31 sugeriše da je veličina celog paketa (blok podataka i završni bitovi) umnožak 48 bajtova. Pošto količina podataka može da bude promenljiva, blok podataka može, u stvari, da se dopuni jednim od 47 dodatnih bajtova. Broj se bira tako da veličina paketa bude umnožak 48.

U završnim bitovima postoje dve stavke koje treba pomenuti. Prva je 32-bitno CRC polje, koje obezbeđuje proveru grešaka za ceo paket, što je još jedna naznaka da ovaj format nije dizajniran za real-time audio, ili video: korišćenje 32-bitnog CRC-a na potencijalno hiljadama bitova može da traje izuzetno dugo. Drugo polje definiše broj bajtova u polju korisnih informacija (Payload).

Pošto se aplikacija pokreće preko ATM-a, podaci se i dalje deie na manje ćelije. Ipak, značajan aspekat ovog deljenja je da se izvodi na drugom sloju. Kod AAL 1 i AAL 2 podaci se ranije dele na male pakete tako da mogu da se obrade i pošalju što pre.



SLIKA 13.31 *Deljenje bloka podataka AAL 5 paketa na ATM ćelije*

Ovde podsloj konvergencije radi sa potencijalno velikim blokom podataka i kreira završne bitove za njega. Zbog velidne bloka, ovo može da potraje, što može da bude neprihvatljivo za saobračaj Klase A, ili Klase B. Ipak, kod saobraćaja Klase C, ili Klase D ovakva kašnjenja nisu problem.

Service-specific connection-oriented protokol (SSCOP)

AAL protokoli koje smo opisali bave se prvenstveno transferom podataka. Još uvek nismo predstavili protokol čiji je primarni zadatak obezbedivanje pouzdane konekcije. Rani kreatori ATM-a su prepoznali da će detekcija grešaka i oporavak biti potrebni za signaliziranje iza odredene aplikacije za rad sa podacima. Zato je AAL podeljen na dva dela: zajednički deo i deo specifičan za konkretni servis. AAL 5 je primer zajedničkog dela. Dva dodatna protokola *Service-Specific Coordination Function* (SSCF) i *Service-Specific Connection-Oriented Protocol* (SSCOP) čine deo specifičan za servis. SSCOP i SSCF se nalaze između AAL 5 i protokola za signalizaciju na višem sloju, kao što je Q.2931, gde SSCF obezbeđuje interfejs između SSCOP i Q.2931. Uzeti zajedno, SSCF i SSCOP nazivaju se i *signalni ATM adaptacioni sloj* (SAAL - *signaling ATM adaptation layer).*

SSCOP je sinhroni bitovima orijentisani protokol čije su glavne funkcije kontrola toka i akcije koje se preduzimaju kada se detektuju greške. Koristi se oblik protokola klizajućih prozora sa selektivnom retransmisijom, kod koga se veličina prozora može podešavati dinamički i može eksplicitno da se zahteva prenos tih okvira.

Signalni sloj (Q.2931) komunicira sa SSCOP preko SSCF-a. SSCOP kreira specifične pakete, koje šalje do ravnopravnog SSCOP entiteta na drugoj strani. Prijemni SSCOP izvlači informacije iz tih paketa i prosleđuje ih preko svog SSCF-a do signalnog sloja na toj strani. SSCOP paketi mogu da budu promenljive dužine, sa maksimalnom veličinom 64 Kbytes.

U sledećoj listi su navedeni tipovi SSCOP paketa:

- Begin i Begin Acknovvledgment paketi se koriste za uspostavljanje i potvrdu uspostavljanja SSCOP konekcije.
- End i End Acknowledgment paketi se koriste za oslobadanje i potvrdu oslobađanja SSCOP konekcije.
- Reject paketi se koriste za odbacivanje zahteva ravnopravnog SSCOP entiteta za konekciju.
- Resynchronize i Resynchronize Acknowledgment paketi se koriste za ponovno uspostavljanje konekcija i parametara konekcije u slučaju da dode do "otkaza" konekcije.
- Sekvencirani paketi sa podacima sadrže korisničke informacije. Oni koriste 2'4-bitne brojeve sekvence koji dopuštaju potencijalno velike prozore. Ovo je značajno zbog toga što mali prozori mogu da ograniče propusnost protokola.
- PoIl paketi se koriste za zahtev za informacije o statusu primaoca. SSCOP se ne oslanja na tajmer za ponovo slanje paketa koji nikada nije stigao. Tako se redukuje zavisnost od implementacije tajmera. Umesto toga, koristi se PoIl paket, koji sadrži naredni broj sekvence nakon poslednjeg prenetog paketa. Ovaj paket ujedno zahteva od primaoca da pošalje svoj status. Po prijemu PoIl paketa, primalac može da kaže da li neki od prethodno poslatih paketa nisu stigli.

- Status paket šalje primalac kao odgovor na Poll paket. On sadrži granični broj sekvence za prijemni prozor, broj sekvence sledećeg očekivanog paketa i brojeve sekvence svih paketa koji nedostaju. Po prijemu Status paketa, pošiljalac zna koliko paketa može da pošalje bez prekoračenja prijemnog prozora i koji paketi nisu stigli. Nakon toga, reaguje na odgovarajući način.
- Unsolicited Status paket je, u suštini, isti kao Status paket, osim što se ne šalje kao odgovor na Poll paket.

Dodatne informacije o SSCOP-u možete da nađete u referencama [Ha98] i [He95] radi.

Gigabit Ethernet naspram ATM-a mreže

Razvoj Gigabit Etherneta (sećate se Poglavlja 9) inicirao je poredenja sa ATM-om, jer se ATM ponekad smatra glavnom tehnologijom za implementaciju okosnica (backbones) koje povezuju više LAN-ova. Ranije smo istakli da je ATM protokol koji je bio razvijen većim delom kao reakcija na sve masovnije korišćenje audio i video aplikacija. Takve aplikacije zahtevaju kvalitet servisa koji se razlikuje od kvaliteta koji se zahteva za konvencionalni transfer fajlova, email, ili Web aplikacije. QoS za real-time aplikacije sa video i audio zapisima zahteva brzu isporuku sa veoma malim, ili nikakvim kašnjenjima u isporuci podataka. Starije verzije Etherneta nisu mogle da garantuju takav QoS zbog malih brzina i mogućih kolizija. Zato nije bilo neuobičajeno da se ATM koristi za implementaciju širokopojasnih linkova izmedu LAN-ova.

Sa novim brzinama i full-duplex modovima, Ethemet sada može da obezbedi zahtevani QoS za real-time aplikacije. Zbog toga, postaje prihvatljiva opcija i, u stvari, već je počeo da zamenjuje ATM kod okosnica za više LAN-ova na različitim lokacijama. Ovo ne znači da se ATM napušta, pogotovo zato što se i dalje koristi za povezivanje mnogih mreža na velikim gradskim područjima, pa, čak, i za mreže koje pokrivaju veće geografske oblasti. Ove dve tehnologije se i dalje značajno razlikuju. Na primer, ATM je orijentisan konekciji, a Ethernet nije. ATM prenosi okvire fiksne veličine, dok veličina okvira kod Etherneta varira.

ATM QoS je implementiran pomoću protokola za signalizaciju koji se izvršava pre uspostavljanja konekcije. Na taj način je omogućeno uspostavljanje konekcije samo ako može da se postigne traženi QoS. Ethernet je dizajniran za ravnopravno tretiranje svih vrsta saobracaja, Dakle, ne pravi nikakvu razliku između okvira koji prenose podatke i podataka video zapisa. Zbog toga, pristalice ATM-a tvrde da Ethernet, u stvari, ne može da obeća isti QoS kao ATM. Pristalice Etherneta smatraju da povećanje opsega signala i dodatni protokoli, kao što su Resource Reservation Protocol (RSVP) i Real-Time Transport Protocol (RTP), čine Ethernet prihvatljivom alternativom. Sećate se iz Poglavlja 11 da se RSVP, koji je IETF razvio, koristi na hostovima i ruterima za postavljanje QoS zahteva i rezervisanje resursa potrebnih za ispunjavanje tih zahteva. RTP, takođe projekat IETF-a, predstavlja protokol između krajnjih tačaka koji je dizajniran za podršku real-time aplikacija.

13.5 Zaključak

Ovim poglavljem je obuhvaćeno nekoliko specifičnih protokola koje mnogi smatraju značajnim WAN tehnologijama: ISDN, X.25 protokol sa komutacijom paketa, frame relay i ATM. Sledi njihov rezime.

- ISDN je dugogodišnji protokol koji definiše potpuno digitalizovani komunikacioni sistem. Originalno je dizajniran kao eventualna zamena za telefonski sistem, ali nikada nije prihvaćen kao opšti sistem, delom zbog toga što su mnoge digitalne komponente već bile uvedene u postojeći telefonski sistem, a delom zato što prednosti zamene stotina miliona analognih telefona i njihovih kola jednostavno nisu opravdavale cenu.
- X.25, koji je originalno razvijen za WAN konekcije preko evropskih granica, definiše standard za postavljanje i komuniciranje pomoću virtuelnih kola. Reč je o troslojnom protokolu koji definiše format paketa, uspostavljanje poziva, okončavanje poziva, kontrolu toka i kontrolu grešaka.
- Frame relay su mnogi videli kao naslednika X.25 protokola. Izvršava slične funkcije, ali nema kontrolu toka, niti kontrolu grešaka. Korisnicima omogućava da plate željenu bitsku brzinu i obezbeduje resurse koji omogućavaju ispunjavanje korisnikovih zahteva. Frame relay sa komutatora uklanja dosta funkcionalnosti, kao što su kontrola toka i grešaka, ali pretpostavlja se da se time bave korisnički uredaji.
- ATM je sledeća tehnologija sa komutacijom kola koja nudi više opcija za ispunjavanje zahteva kvaliteta servisa u poređenju sa frame relay protokolom. Zato je prikladniji ne samo za standardni saobraćaj, već i za audio, ili video aplikacije koje zahtevaju konstante bitske brzine. Za razliku od velikih okvira promenljive veličine, ATM koristi 53-bajtne ćelije za sve vrste saobraćaja. Manje ćelije omogućavaju bržu obradu informacija u zaglavlju i brže prosledivanje.

Pitanja i zadaci za proveru

1. ŠtajelSDN?
2. Navedite razliku izmedu osnovnog i primarnog ISDN servisa.
3. Navedite i rezimirajte četiri funkcionalne grupe kod lSDN-a.
4. Navedite i rezimirajte četiri referentne tačke kod ISDN-a.
5. U čemu je razlika izmedu signaliziranja u opsegu (in-band) i van opsega (out-of-band)?
6. Šta je Signaling System 7?
7. Šta je pseudoternarno kodiranje?
8. Šta je širokopojasni ISDN?
9. Da li su sledeće konstatacije tačne, ili netačne (zašto)?
   a. Potpuno digitalizovana svojstva ISDN-a odmah obezbeduju prednosti za sve one koji ga koriste.
   b. X.25 protokol definiše operacije mreže sa komutacijom paketa.

c. Različita virtuelna kola mogu da se preklapaju.

d. Servis datagrama ne isporučuje pakete u ispravnom redosledu.

e. Frame relay je bio dizajniran za rad sa high-speed digitalnom opremom koja obezbeduje kvalitet servisa neophodan za real-time aplikacije, kao što video aplikacije i aplikacije za prenos glasa.

f. Angažovana brzina prenosa informacija kod frame relay protokola garantuje korisniku da će njegovi podaci stići na svoje odredište naznačenom brzinom.

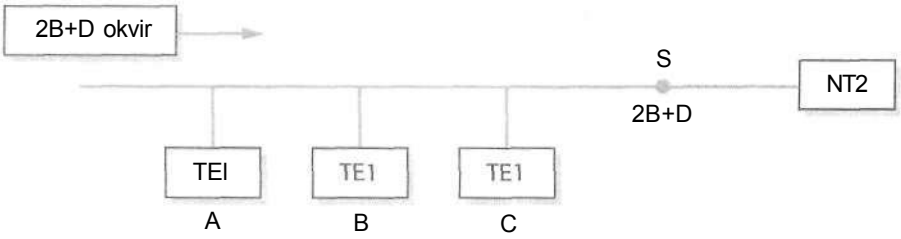
g. Nedostatak malih ćelija fiksne veličine kod ATM-a je što se manje informacija može postaviti u zaglavlja i zato komutatorima treba više vremena za rosleđivanje podataka.

10. Šta je virtuelno kolo?
11. Šta je mreža sa komutacijom paketa?
12. Zašto svi čvorovi u virtuelnom kolu ne mogu da koriste isti broj za identifikovanje kola?
13. Šta je je servis datagrama?
14. Šta definiše X.25 standard?
15. Navedite tipove X.25 paketa koji se tiču uspostavljanja poziva.
16. Po čemu se permanentno virtuelno kolo razlikuje od običnog virtuelnog kola?
17. Koje su sličnosti X.25 i frame relay protokola? Koje su razlike?
18. Sta je angažovana brzina prenosa informacija kod frame relay protokola?
19. Kako vrednost DE bita u okvirima frame relay protokola utiče na njihovu isporuku?
20. Frame relay će postaviti ili BECN, ili FECN bitove, ili oba kada dođe do zagušenja. U čemu je razlika između njih?
21. Kako frame relay protokol rešava zagušenja?
22. Zašto neki smatraju da je ATM protokol budućnosti?
23. Zašto postoji nekoliko AAL slojeva?
24. Navedite razlike između saobraćaja klasa A, B, C i D na ATM mreži.
25. Zašto su ATM ćelije dugačke 53 bajta?
26. Navedite neke prednosti korišćenja malih ćelija fiksne dužine kod ATM-a.
27. U ćemu je razlika između virtuelne putanje i virtuelnog kola kod ATM-a i zašto ta ona postoji?
28. U čemu je prednost korišćenja Banyan komutatora u odnosu na uobičajenu tabelu rutiranja?
29. Zašto polje Header Error Control u zaglavlju ATM ćelije proverava samo zaglavlje i ostavlja korisne informacije nezaštićene?
30. Opišite kako polje Header Error Control može da se koristi za lociranje granica ATM ćelije.
31. Navedite neke elemente koje može da sadrži ATM Setup poruka.
32. Zašto je prihvatljivo da AAL 5 paketi mogu da sadrže potencijalno velika polja sa korisnim informacijama promenljive dužine kada je to neprihvatljivo kod AAL 1 i AAL 2?

Vežbe

1. Razmotrite sliku 13.10. Pretpostavimo da A uspostavlja virtuelno kolo ka D, a da B uspostavlja virtuelno kolo ka C. Ako oba virtuelna kola prolaze kroz X i Y, kako će izgledati tabele rutiranja u X i Y?

2. Pretpostavite da svaki TEl sa sledeće slike pokušava istovremeno da pošalje nešto do NT2.



Pretpostavite i da svaki od njih pokušava da pošalje sledeće informacije preko D kanala:

- A šalje 10011101.
- B šalje 10001100.
- C šalje 10011001.

Koji TEl "pobeduje" u "nadmetanju"?

3. Pri dizajniranju kapaciteta B kanala kod ISDN-a uzet je u obzir prenos glasa. U odeljku 3.6 smo pokazali da su za zahvatanje većih glasovnih karakteristika dovoljni 8-bitni semplovi uzeti sa frekvencijom od 8.000 semplova u sekundi, i da zahtevaju bitsku brzinu od 64.000 bps. Međutim, neke tehnike kodiranja omogučavaju, u stvari, prenos govornih podataka na brzini od 32.000 bps, dok druge verovatno dopuštaju i 16.000 bps kao dovoljnu brzinu. Kakav efekat ovo može da ima na ISDN standard?

4. Nacrtajte dijagram sličan onom sa slike 13.7, koji prikazuje proceduru raskidanja ISDN konekcije.

5. Za kontrolu toka kod frame relay protokola postoje dve opcije za rešavanje zagušenja, koje uključuju postavljanje BECN, ili FECN bita. Koji od ta dva bita ima direktniji i brži efekat na redukovanje zagušenja?

6. U poređenju sa X.25, frame relay obavlja mnogo manje funkcija za upravljanje paketima (okvirima) u posredničkim komutatorima. U čemu se sastoji prednost takvog pristupa? U čemu su nedostaci?

7. Pretpostavimo da je kod frame relay protokola CIR = 20.000 bps, veličina okvira 1.000 bitova, B_e = 3.000 bitova i da se brzine izračunavaju u periodu od 0,5 sekundi. Koliko okvira u prvom intervalu od 0,5 sekundi može da pošalje prvi komutator ako je DE bit postavljen na 0? Koliko ih šalje ako je DE bit postavljen na 1? Koliko se okvira odbacuje?

8. Opišite scenario u kome se četiri ćelije mogu istovremeno rutirati bez kolizija preko Banyan komutatora sa slike 13.21.

Reference

[Ap86] Appenzeller, H. R. "Signaling System No. 7, ISDN User Part". *IEEE lournal on Selected Areas in Communication,* vol. SAC-4, no. 3 (May 1986), 366-37 1.

[AT89] AT&T and Bellcore. "Observations of Error Characteristics of Fiber Optic Transmission Systems". CCJTTSGXVi//, San Diego, CA, January, 1989.

[B195] Black, U. *The X Series Recommendations: Standards for Data Communications,* 2nd ed. New York: McGraw-Hill, 1995.

[B199] Black, U. *ATM Volume T. Foundation for Broadband Netivorks,* 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[Bl97a] Black, U. *Emerging Communications Technologies,* 2nd ed. Upper Saddle River, NJ: Prentice-Hall, 1997.

[Bi97b] Black, U. *ISDN and SS7. Architectures for Digital Signaling Netuiorks.* Upper Saddle River, NJ: Prentice-Hall, 1997.

[BlOO] Black, U. *QOS in Wide Area Netuiorks.* Upper Saddle River, NJ: Prentice-Hall. 2000.

[Gr98] Griffiths, J. M. *ISDN Explained: Worldwide Nelivork and Applications Technology,* 3rd ed. New York: Wiley, 1998.

[Ha98] Handel, R., M. Huber, and S. Schroder. *ATM Netiuorks: Concepts, Protocols, Applications,* 3rd ed. Reading, MA: Addison-Wesley, 1998.

[He95] Henderson, T. "Design Principles and Performance Analysis of SSCOP: A New ATM Adaptation Layer Protocol". *Computer Communications Revieiv,* vol. 25, no. 2 (April 1995), 47-59.

[Hu03] Hunt, R. "Frame Relay". *Encyclopedia of Information Systems,* vol. II. New York: Academic Press, 2003, 371-390.

[McOl] McQuerry, S., and K. McGrew. *Cisco Voice over Frame Relay, ATM$_1$ and IP.* Upper Saddle River. NJ: Prentice-Hall, 2001.

[Me03] Mesher, G. "Integrated Services Digital Network (Broadband and Narrowband ISDN)". *Encyclopedia of Information Systems,* vol. II. New York: Academic Press, 2003, 627-638.

[Sc86] Schlanger, G. G. "An Overvievv of Signaling System No. 7". *IEEE Journal on Selected Areas in Communication,* vol. SAC-4, no. 3 (May 1986), 360-365.

[St99] Stallings, W. /SDN *and Broadband ISDN with Frame Relay and ATM,* 4th ed. Upper Saddle River, NJ: Prentice-Hall, 1999.

[Va97] Varshney, U. "Supporting Mobility with Wireless ATM." *Computer,* vol. 30, no. I (January 1997), 131-133.

Rečnik

10Base2 10 Mbps Ethernet standard za tanki koaksijalni kabl

lOBaseS lOMbps Ethemet standard za debeli koaksijalni kabl

10BaseFx 10 Mbps Ethernet standard za optički fiber

lOBaseT 10 Mbps Ethernet standard za UTp kablove kategorije 3, 4, ili 5

100BaseFX 10 Mbps Ethernet standard za optički fiber

100BaseT4 100 Mbps Ethernet standard za UTP kabl kategorije 3

100BaseTX 100 Mbps Ethernet standard za UTP kabl kategorije 5

lOOOBaseT Gigabit Ethernet standard za UTP kabl kategorije 5

1000BaseX Gigabit Ethernet standard za optički fiber

802.3 standard Zajednički termin za više razliatih Ethernet standarda za različite medijume

802.5 standard IEEE standard za token ring mrežu

802.11 standard IEEE standard za bežični LAN

4B/5B kodiranje Šema kodiranja kod koje se 4-bitne vrednosti menjaju sa 5-bitnih vrednosti

8B/6T kodiranje Šema kodiranja kod koje se osmo-bitne vrednosti menjaju sa šest tritskih vrednosti

8B/10B kodiranje Šema kodiranja kod koje se 8-bitne vrednosti menjaju sa 10-bitnim vrednostima

abort - prekid Iznenadni završetak aktivnosti, obično zbog neke greške

Abstract Syntax Notation 1 (ASN.l) Formalni jezik dizajniran iskljuavo radi definisanja formata za jedinice podataka protokola i za predstavljanje operadja nad distribuiranim informacijama

access point - pristupna tačka Uredaj povezan na kabliranu mrežu koji može da komunicira sa bežičnim uredajima

Acknovvledgment - potvrda Ukazivanje da je

□□□ tajmer Tajmer koji se koristi kod protokola za kontrolu toka za utvrđivanje kada treba poslati zasebnu potvrdu, u slučaju da nema odlazećih okvira

adaptive routing - adaptivno rutiranje Strategija ratiranja koja reaguje na promene na mreži

add/drop multiplekser Uredaj koji se koristi za izvlačenje saobraćaja sa SONET prstena i za mešanje sa postojeđm saobrac'ajem na prstenu

address learning - učenje adresa Mogućnost LAN mosta, ili komutatora da dinamički ažuriraju svoje tabele rutiranja

Address Resolution Protocol Videu: *dinamičko povezivanje*

address spoofing - lažiranje adrese Slanje Intemet paketa sa lažnom izvomom adresom

Advanced Encryption Standard (AES) NIST standard zasnovan na Rijndaelovom algoritmu

Advanced Research Projects Agency Agencija američkog ministarstva odbrane (U.S. Department of Defense)

Projects Agency Network (ARPANET) Mreža koju je razvila Advanced Research Projects Agency, a koja je prerasla u protokole koji se koriste na Intemetu

Aloha protokol Protokol slanja paketa radio difuzijom kod koga uredaj šalje paket i, ako dode do kolizije sa drugim paketom, ponovo šalje taj paket nakon nasumice izabranog perioda vremena

alternate mark inversion - inverzno naizmenično označavanje Digitalno kodiranje kod koga se 1 predstavlja nultim naponom, a 0 naizmenično sa pozitivnim i negativnim signalom

American National Standards Institute (ANSI) Privatoa, nevladina agencija za uspostavljanje standarda aji su članovi proizvodači, korisnici i drage zainteresovane kompanije

American Standard Code for Information Interchange (ASCII) Sedmobitni kod koji dodeljuje jedinstvenu kombinaciju svakom karakteru sa tastature i nekim specijalnim funkcijama

amplitude - amplituda Najveća vrednost analognog signala

amplitude modulation (amplitude shift keving) - amplitudska modulacija Metod predstavljanja bitova pomocu analognih signala razliđtih amplituda

analog signal - analogni signal Kontinuelno promenljivi signal

analog-to-digital conversion - analogno-digitalna konverzija Proces konvertovanja analognih signala u digitalne

anonvmous FTP - anonimni FTP Aplikacija koja udaljenim korisnicima omogućava pristup setu fajlova na konkretnom sajtu

application layer - sloj aplikacije Sedmi i najviši sloj OSI protokola, koji radi direktno sa korisnikom i programskim aplikacijama

application-level gateway Firewal koji funkcioniše na sloju aplikacije

arbitration - arbitraža Proces donošenja odluka koji uredaj dobija pristup resursu

arithemtic compression - aritmetička kornpresija Metod kompresije koji generiše frekventno-zavisni kod zasnovan na interpretiranju stringa kao realnog **broja**

asvmmetric DSL (ADSL) - asimetrični DSL DSL tehnologija kod koje se bitske brzine prilikom preuzimanja i slanja podataka razlikuju

asvnclironous balanced mode - asinhroni balansirani mod Mod za HDLC kod koga svaka stanica može da šalje podatke, kontrolne informacije, ili komande

asvnchronous response mode - mod asinhronog odziva Mod za HDLC kod koga primarna stanica može da šalje podatke, kontrolne informacije, ili komande, a sekundarna stanica samo kontrolne informacije, ili podatke

Asvnchronous Transfer Mode (ATM) Veoma brzi protokol za komutaciju paketa koji koristi male pakete fiksne velidne, optimizovan za multimedijalne aplikadje

asvnchronous transmission - asinhroni prenos Mod za prenos kod koga se bitovi dele na manje grupe (bajtove ili oktele) i šalju nezavisno

ATM adaptation layer (AAL) - ATM adaptacioni sloj Komponenta ATM protokola koja obezbeduje interfejs između aplikacija **i** ATM sloja

attenuation - slabljenje Degradadja signala dok putuje duž medijuma

auditory masking - čujno maskiranje Fenomen koji sprečava registrovanje određenih zvukova u prisustvu jačeg zvuka slične frekvencije

authentification - autentifikacija Verifikovanje pošiljaoca poruke, ili autentidiosti dokumenta

automatic repeat request - automatski zahtev za retransmisiju Kontrola grešaka kod koje uredaj zahteva od predajnog uredaja da ponovo pošalje poruku ako dode do greške

autonomous svstem - autonomni sistem Domen koji funkcioniše nezavisno od ostalih domena

B channel - B kanal ISDN kanal koji može da prenosi podatke brzinom od 64 Kbps

backbone - okosnica Deo mreže koji sadrži primame prenosne putanje

backvvard search algorithm - algoritam pretraživanja unazad Algoritam za aitiranje kod koga čvor uči od svojih suseda koja je najjeftinija ruta do sledečeg čvora (poznat je i kao algoritam učenja unazad)

balanced circuit - balansirano kolo Kolo koje koristi dve linije koje prenose iste signale suprotnih polariteta

bandpass filter - propusni filter Uredaj koji se koristi za izvlačenje individualnih modulisanih signala

bandwiddi - opseg signala Razlika izmedu najviše i najniže frel<vencije koju medijum može da prenese

□□□□□□ switch - □□□□□□ komutator Komutator koji povezuje svoje ulazne i izlazne portove preko niza intemih stepena

baseband mode - mod osnovnog opsega Mod u kome je opseg signala koji se prenose preko kabla rezervisan za jedan niz podataka

basic service set (BSS) - osnovni servisni skup Skup bežičnih uredaja koji komuniciraju sa jednom pristupnom tačkom

Baudot code - Bodov kod Petobitni kod originalno dizajniran za francuski telegraf

baud rate - brzina bauda Brzina kojom komponente signala mogu da se menjaju

Beacon frame - Beacon okvir Kontrolni okvir kod token ring mreža koji se koristi za informisanje uredaja o pojavi problema zbog koga je zaustavljeno prosledivanje tokena

beam shaping - uobličavanje mlaza Proces u okviru koga se omogučava koncentrisanje satelitskog signala na manju oblast

bel Jedinica mere za jačinu signala u odnosu na jačinu šuma

Bellman-Ford algorithm - Bellman Fordov algoritam Tip algoritma sa učenjem unazad

binary-coded decimal (BCD) - binarno kodirani decimalni kod Kod koji je korišćen za rane IBM mainframe kompjutere

binary exponential backoff algorithm - binarni eksponencijalni backoff algoritam Algoritam koji se koristi za utvrdivanje kada uredaj treba ponovo da pošalje okvir nakon kolizije

binary svnchronous communications (BSC) protokol Bajtovima orijentisani protokol veze koga je kreirao IBM

birthday attack - napad rođendana Tehnika koje se koristi za pokušaj pronalaženja dva dokumenta koji generišu istu vrednost digitalnog sažetka

bisvnc Videti: *bimry synchranous communications (BSC) protokol*

bit Osnovna jedinica inforraacija; obično se predstavlja sa 0, ili 1

bit-level ciphering - šifrovanje na nivou bita Šifrovanje koje se izvodi manipulacijom bitova

bit-oriented protocol - bitovima orijentisani protokol Protokol koji tretira okvire kao nizove bitova

bit pipe - bitski vod Stvarni prenos bita kod ISDN protokola

bit stufflng - popunjavanje bitova tlmetanje dodatnog bita kako bi se izbegli dugački nizovi istog bita

block chaining mode - mod ulančavanja blokova Mod šifrovanja kod koga šifrovanje jednog bloka zavisi od šifrovane verzije prethodnog bloka

Block Check Character Karakter koji se koristi za provem grešaka kod BSC protokola

block cipher - blokovsko šifrovanje Metod za šifrovanje koji deli obični tekst na blokove i šifruje svaki blok

Bluetooth Tehnologija koja koristi mikročip u kome se nalazi radio predajnik/prijemnik za bežićnu komunikaciju

Border Gateway Protokol Protokol koji ruterima omogućava implementiranje specifičnih polisa, ili ograničenja koje ruta mora da ispuni

brdige - most Konekcija sloja 2 izmedu dve mreže

bridge port - port mosta Most za LAN konekcije

bridge protocol data unit (BPDU) Jedinične informacije koje mostovi razmenjuju

broadband ISDN (B-ISDN) - širokopojasni ISDN Protokoli koji implementiraju ISDN servise preko high-speed širokopojasne mreže

broadband mode - širokopojasni mod Mod kod koga se propusni opseg deli na podopsege koji prenose zasebno kodirane informacije

broadcast address - emisiona adresa Adresa koja ukazuje da okvir treba poslati do svih uredaja na mreži

broadcast domain - emisioni domen Skup dogadaja do kojih okvir stiže preko emisione adrese

browser - pretraživač Program koji korisniku omogućava pristup i prikazivanje fajlova smeštenih na udaljenim Web serverima

buffering - baferovanje Privremeno smeštanje informacija

burst error - navalna greška Greška koja utiče na veliki broj bitova

bus topology - topologija magistrale Način povezivanja uređaja tako da svi komuniciraju preko zajedničkog kabla

byte multiplexer - bajtski multiplekser Llredaj koji kombinuje bajtove iz različitih izvora u jedan zajednički niz podataka

byte-oriented protocol - bajtovima orijentisani protokol Protokol koji okvire trelira kao nizove bajtova

byte stuffing - popunjavanje bajtova Umetanje dodatnog bajta kako bi se sprečilo pogrešno interpretiranje bajta podataka kao kontrolnog bajta

cable modem - kablovski modem Uredaj dizajniran za povezivanje kompjutera pomoću signala koje obezbeduje provajder kablovskih usluga

Caesar cipher - Cezarova šifra Jednostavna tehnika za šifrovanje kod koje se svaki karakter menja drugim karakterom koji zavisi samo od vrednosti originalnog karaktera

Carriage Return ASCII kontrolni karakter koji pomera mehanizam za štampanje na krajnju levu poziciju za štampanje

Carrier Sense Multiple Access (CSMA) Protokol koji se koristi kako bi se utvrdilo da li je medijum zauzet pre pokušaja slanja podataka

Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) Strategija nadmetanja koju koristi bežični LAN standard definisan sa 802.11

Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Sličan protokolu Carrier Sense Multiple Access, samo što ima mogućnost detektovanja da li je došlo do kolizije

carrier signal - noseći signal Signal koji se moduliše ulaznim signalom kako bi se kreirao drugi signal

CAT 5 cable - CAT 5 kabl Vrsta UTP kabla koji se danas koristi u mnogim mrežama i koji može da podrži brzine od 100 megabita u sekundi na rastojanjima do 100 metara

cell - ćelija (1) Geografska oblastu kojoj prijemna i predajna stanica komuniciraju sa mobilnim telefonima; (2) Jedinica prenosa defmisana ATM protokolom

cellular telephone - mobilni telefon Prenosivi telefon koji se povezuje na telefonski sistem preko radio komunikacija

centralized routing - centralizovano rutiranje Rutiranje koje koristi informacije generisane i čuvane na jednoj centralnoj lokaciji

certificate authority - nadležno telo za izdavanje sertifikata Agencija koja kreira □.509 sertifikate

channel - kanal Širina opsega signala

Cheapemet IEEE 802.3 mreža koja koristi 10Base2 kabl

checksum - čeksuma Vrednost koja se koristi za detektovanje grešaka, a formira se sabiranjem bitskih stringova koji se interpretiraju kao celi brojevi

chipping sequence - chipping sekvenca Bitski string koji se koristi za proširivanje bitova kod DSSS tehnologije

Choke packet - Choke paket Kontrolni paket koji, kada se pošalje do uređaja, izaziva redukovanje broja paketa koje taj uredaj šalje

ciphertext - šifrovani tekst Šifrovana poruka

circuit switching - komutacija kola Konekcija koja je rezervisana za komuniciranje dve stanice

cladding - obloga Deo optičkog fibera koji ima manju optičku gustinu od jezgra, tako da se svetlost reflektuje nazad ka jezgra

Class A, B, C address - adrese Klase A, B, C Internet adrese kod kojih odredeni bitovi predstavljaju ID mreže, a ostali se dodeljuju lokalno u okviru mreže

Class D address - adresa Klase D Internet adresa koja se koristi za slanje paketa na više odredišta (multicasting)

classless address - besklasna adresa Internet adresa koja nije pridružena tradidonalnim adresama klaseA, B, C, ili D

classless interdomain routing (CIDR) Inlernet rutiranje koje se zasniva na strukturi besklasnih adresa

client - klijent Uredaj, ili program kod klijent/server modela koji šalje zahteve do servera

dient/server model - klijent/server model Model komunikacija izmedu dva uređaja, od kojih jedan ima mogućnost obrade podataka

Clipper Chip Konlroverzni uredaj za šifrovanje koji može da se postavi na telefone radi skremblovanja konverzacija

coaxiaI cable - koaksijalni kabl Provodna žica obavijena izolacionim slojem, žičanom oplatom i spoljšnjim zaštitnim omotačem

code - kod Asocijacija bitskih uzoraka specifičnim informacijama, kao što su karakteri, ili akcije

codec Uredaj koji se koristi za prevođenje analognog signala u digitalni ekvivalent

collision - kolizija Rezultat simultanog slanja podataka iz dve, ili više stanica preko zajedničkog medijuma koji je dizajniran za prenos jednog signaia u jednom trenutku

collision avoidance - izbegavanje kolizije Protokol koga koristi bežični LAN standard koji značajno redukuje broj okvira koji se sudaraju

collision detection - detektovanje kolizije Moguc'nost uredaja da registruje nastalu koliziju

collision domain - domen kolizije Skup uredaja u LAN okruženju dji podaci mogu da se sudare

combined station - kombinovana stanica Stanica koja se ponaša i kao primama i kao sekundardna kod HDLC protokola

Comite Consultatiflnternational de Telegraphique et Telephonique (□□□□) Raniji naziv organizacije za uspostavljanje standarda, čiji su članovi razne naučne i industrijske organizacije, telekomunikacione agencije, nadležna tela za telefoniju i LSO; sada nosi naziv □□-□

committed information rate - angažovana brzina prenosa informacija Bitska brzina koju nosilac rezerviše radi obezbedivanja prenosa preko vinuelnog kola

common bus topologv - topologija zajedničke magistrale Nadn povezivanja uredaja kod koga svi uredaji komuniciraju preko zajedničkog kabla

Common Gateway Interface (CGI) program Serverski program koji može da se aktivira iz HTML dokumenta

Common Management Information Protocol ISO upravljački protokol

communications subnet - komunikaciona podmreža Kolekcija prenosnih medijuma i komutacionih elemenata koji su neophodni za rutiranje i prenos podataka

compact disc - kompakt disk Medijum za skladištenje kod koga se informacije beleže optički kreiranjem veoma malih udubljenja na reflektujućem materijalu

compression - kompresija Redukovanje broja bitova u fajlu, pri čemu se zadržava kompletno značenje, ili najveci deo

congestion - zagušenje Prekomema količina saobrac'aja na mreži koja izaziva degradaciju servisa i vremena odziva

congestion control - kontrola zagušenja Protokoli koji se koriste za ublažavanje zagušenja na mreži

connection - konekcija Mehanizam pomoću koga dva uredaja razmenjuju informacije

connection management - upravljanje konekcijom Protokol koji se koristi za uspostavljanje, održavanje i otpuštanje konekcija

connection request - zahtev za konekciju Zahtev za uspostavljanje konekcije

connection strategy - strategija povezivanja Strategija koja se koristi za implementiranje konekcije

content addressable □□□□□□ - memorija sa adresabilnim sadržajem Brzi metod pretraživanja memorije zasnovan na njenom sadržaju

contention - "nadmetanje" Termin koji se koristi kada dva, ili više uredaja treba istovremeno da prenose podatke preko zajedničkog medijuma

contention protocol - protokol za "nadmetanje" Tehnika koja se koristi za kontrolu pristupa zajedničkom medijumu iz više ulaznih tačaka

control bits - kontrolni bitovi Deo okvira koji se koristi za kontrolne funkcije, kao što su rutiranje i rukovanje

control characters - kontrolni karakteri Binarni kodovi koji identifikuju akciju, ili status

conlrol-Q Kontrolni karakter koji se koristi kada uredaju treba saopštiti da može da nastavi slanje podataka

control-S Kontrolni karakter koji se koristi da bi se uredaju saopštilo da prekine slanje podataka

credit - kredit Koristi se kod konuole toka, a definiše broj bajtova koje TCP entitet može da primi od drugog entiteta

crossbar svvitch - tmakrsni komutator Komutacioni uređaj koji sadrži logiku za komutaciju svakog para ulaznih i izlaznih linija

crosstalk - **preslušavanje** Interferenca na liniji izazvana paralelnim prenosom signala

cyclic redundancy check (CRC) - ciklična provera redundantnosti Metod za detekciju grešaka zasnovan na interpretiranju niza bitova kao polinoma sa binarnim koeficijentima

daisy chain Strategija povezivanja koja uredaje povezuje u sekvencu

data circuit-terminating equipment (DCE) - **oprema za razmenu podataka** Uredaj koji se koristi kao interfejs između DTE-a **i** mreže

data compression - kompresija podataka Videti: *hompresija*

Data Encrvption Standard (DES) Tehnika za šifrovanje, koju je razvio IBM i koju je kao standard usvojila američka vlada (U.S. government); više se ne smatra bezbednom

data frame - okvir podataka Videti: *okvir,* definicija **(1)**

datagram Nezavisna jedinica prenosa podataka u mreži sa komutacijom paketa

Data Link Escape character - Data Link Escape karakter Kontrolni karakter koji se koristi kao preklopnik koji izaziva različito interpretiranje primljenih karaktera

data link layer - sloj veze Protokol sloja 2

data link protocols - protokoli veze Protokoli koji funkcionišu na sloju 2

data rate - brzina prenosa podataka Mera koja definiše broj bitova podataka koje protokol može da pošalje u jedinici vremena

data strobe encoding - kodiranje sa strob signalom Komunikacioni metod koga koristi FireVVire čiji je zadatak očuvanje sinhronizacije izmedu dva uredaja koji koriste strob signal

data terminal equipment (DTE) - oprema terminala Uredaj koji se povezuje na komunikacioni medijum preko DCE

D channel - D kanal ISDN kanal koji se koristi za prenos komrolnih informacija i za sporije aplikacije

dcadlock - "samrtnl zagrljaj" Situacija u kojoj uredaji ćekaju na dogadaje koji nikada neće da se dese zbog njihovih tekućih stanja

deadly embrace Videti: *deadlodt*

decibel Mera koja se koristi za koeficijent signal-šum

decryption - dešifrovanje Proces obnavljanja šifrovane poruke u njenu originalnu formu

Deep Space Network NASA-ino postrojenje za praćenje svemirskih istraživanja bez posade

de facto standards - de facto standardi Standardi koji postoje zbog opšte upotrebe nečega

demodulation - demodulacija Obnavljanje signala koji se modulisan u njegov originalni oblik

denial of service attack - napad odbijanja servisa Napad na sajt koji izaziva slanje ogromne količine saobrac'aja do tog sajta, tako da se serveri tog sajta gase

DES Cracker Specijalno dizajnirani kompjuter kreiran radi "razbijanja" DES šifre

designated bridge - označeni most Za vreme izvršenja spanning tree algoritma za medusobno povezivanje LAN-ova, izabrani most koji prosleđuje okvire sa LAN-a

destination address - odredišna adresa Adresa na koju se okvir, ili paket moraju isporučiti

differential cryptanalysis - diferencijalna kriptoanaliza Metod analiziranja tehnika za šifrovanje traženjem parova blokova običnog teksta koji se razlikuju na odredene načine

differenlial encoding - diferencijalno kodiranje Tehnika kompresije kod koje je svaki okvir predstavljen razlikom izmedu njega i prethodnog okvira

differential Manchester encoding - diferencijalno Mančester kodiranje Mančester kodiranje kod koga se 0 i 1 razlikuju po tome da li je došlo do promene signala na početku bitskog intervala

differential phase shift keving - diferencijalna modulacija Razlikovanje analognog sigala merenjem faznog pomaka u odnosu na prethodni signal

Differentiated Services Arhitektura za diferenciranje različitih tipova saobračaja na Internetu

Diffie-Hellman □□□ exchange - Diffie-Hellman razmena ključa Metod distribuiranja ključa kod koga pošiljalac i primalac razmenjuju izračunate vrednosti na osnovu kojih je moguc'e izračunati ključ za šifrovanje

digital signal - digitalni signal Četvorougaoni signal koji može da ima samo nisku, ili samo visoku vrednost

digital signature - digitalni potpis Metod autentifikacije koji uključuje šifiovanje poruke na način za koji zna samo pošiljalac

Digital Subscriber Line (DSL) - digitalna pretplatnička linija Tehnologija koja korisnicima omogućava komunikacije preko postojećih telefonskih linija, koristeći frekvencije izvan opsega koji se koristi kod obične telefonije

Dijsktra's algorithm - Dijkstrin algoritam Algoritam koji se korisli za pronalaženje najjeftinije (najkraće) putanje izmedu dva čvora na grafu

direct-sequence spread spectrum Metod širokog spektra koji proširuje jedan bit na više bitova i koristi širi frekventni opseg za prenos rezultujućeg signala

disconnect - raskidanje konekcije Okončavanje konekcije

discrete cosine transforms - diskretna kosinusna transformacija Izračunavanje koje se koristi u prve tri faze IPEG šeme za kompresiju koja generiše skup prostomih frekvencija

discrete multitone - diskretni tonovi Komunikacioni metod kod DSL-a koji deli signal na razlidte opsege (tonove) i svaki moduliše zasebno

distance-vector algorithm - algoritam vektorskog rastojanja Videti: *Bellman-Fordov algoritam*

Distance Vector Multicast Routing Protocol (DVMRP) Protokol koji se koristi za rutiranje ka više odredišta za vreme kreiranja multicast stabla

distortion - izobličenje Promena signala zbog električnog šuma, ili interference

distributed routing - distribuirano rutiranje Strategija rutiranja kod koje svaki čvor utvrđuje i održava sopstvene informacije o rutiranju

distributed system - distribuirani sistem Infrastruktura koja se koristi za povezivanje više uredaja i mreža

domain - domen (1) Grupa čvorova na mreži; (2) Logička kolekcija sajtova odredenog tipa na Internetu, grupisanih radi lakše administracije

Domain Name System Distribuirani protokol koji prevodi simboličke Internet adrese u 32-bitne numeričke adrese

downlink - silazni link Konekcija od satelita ka zemaljskoj stanici

downstream neighbor - naredni sused Susedni čvor u okvim prstena u smeru u kome se krec'e token

DSL access multiplexer (DSLAM) - DSL pristupni multiplekser Llredaj koji prima DSL signale i rutira ih na Internet

DTE-DCE interface - DTE-DCE interfejs Protokol koji se koristi za definisanje komunikacije izmedu DTE i DCE

dvnamic binding - dinamičko povezivanje Metod koji se koristi za utvrdivanje fizičke adrese uredaja na osnovu njegove IP adrese

Dvnamic Host Configuration Protocol (DHCP) Protokol koji alocira IP adresu za uredaj

Echo Reply ICMP paket koji se šalje kao odgovor na Echo Request paket

Echo Request ICMP kontrolni paket koji se koristi da bi se utvrdilo da li je odredište dostupno

edio supressor - supresor eha Uredaj koji uklanja električni eho iz signala

e-commerce - e-komerc Termin koji se odnosi na primenu Intemet i Web aplikacija u obavljanju poslova

effective data rate - efektivna brzina prenosa podataka Mera koja defmiše stvarni broj bitova podataka koji može da se prenese u jedinici vremena

EIA-232 standard Pravila koja definišu komunikaciju između DCE i DTE

electrical ground - električno uzemljenje Naponski nivo u odnosu na koji se mere naponi signala

electromagnetic waves - elektromagnetni talasi Kontinuelno promenljivo elektromagnetno polje

electronic codebook mod Mod za šifrovanje kod koga se svaki blok šifruje nezavisno od ostalih

Electronic Industries Association (EIA) Agendja za uspostavljanje standarda, dji su članovi elekttonske firme i proizvodad telekomunikacione opreme; đan je ANSI

electronic mail (email) - elektronska pošta Servis koji omogućava elektronsko slanje fajlova, ili poruka do neke osobe

Electronic Numerical Integrator and Calculator (ENIAC) Prvi elekuonski kompjuter

electronic telephone directories - elektronski telefonski imenici Online baza podataka koja obezbeduje slične servise kao i konvencionalne telefonske službe za informisanje

encrvption - šifrovanje Renderovanje informacija u drugadji, nerazumljivi oblik

encryption □□□ - ključ za šifrovanje Podaci koji se koriste za šifrovanje poruke

error - greška Neplanirani dogadaj koji utiče na tačnost podataka, ili protokola

error control - kontrola grešaka Specifikacija načina na koji uredaj proverava da li u okviru postoje greške i šta se preduzima ako se one otkriju

error correction - korekcija grešaka Proces ispravljanja bitova u okviru koji su promenjeni u toku prenosa

error detection - detekcija grešaka Proces utvrdivanja broja bitova u okviru koji su promenjeni za vreme prenosa

error recovery - oporavak od grešaka Sposobnost protokola da se oporavi u slučaju "otkaza" mreže, □ "otkaza" protokola nižeg nivoa

escape character - escape karakter Kontrolni karakter koji izaziva pridruživanje jednog, ili više karaktera nekoj akciji

Eskimo pies Poslastica za letnje doba

ether - etar Zamišljena supstanca koja zauzima sav spoljašnji prostor

Ethernet Bilo koji od nekoliko protokola za lokalne mreže, zasnovan na IEEE 802.3 standardu

even parity - parna parnost Metod za detektovanje grešaka kod koga se dodaje jedan ekstra bit čija je vrednost 0, ili 1, tako da u jedinici prenosa ukupni broj bitova sa vrednošću 1 bude paran

Extended Binarv Coded Decimal Interchange Code (EBCDIC) Osmobitni kod za karaktere i ontrolne funkcije koji je korišćen prvenstveno na IBM mainframe kompjuterima

Exterior Gateway Protocol Protokol koji razmenjuje informacije o rutiranju izmedu dve autonomne mreže u većoj mreži sačinjenoj od većeg broja povezanih mreža

exterior routing protocol - protokol za spoljašnje rutiranje Protokol za rutiranje preko granica autonomnog sistema

extremely Iow frequency (ELFj • izuzetno niska frekvencija Komunikacioni signali čija je frekvencija ispod 300 Hz

facsimile compression - faksimil kompresija Metod kombinovanja run-length i Hafmanovog kodiranja

facsimile (fax) machine - faksimil (faks) mašina Uredaj koji skenira i digitalizuje slike za prenos preko telefonskih linija

Fast Ethemet Ethernel standard koji dostiže brzine do 100 Mbps

Federal Communications Commission (FCC) Federalna agencija koja reguliše i izdaje licence za komunikacije

Federal Information Processing Standards (FIPS) Set publikacija o zaštiti kompjutera koje je izdao NIST

flle server - fajl server Uređaj koji je odgovoran za održavanje i zaštitu mrežnih fajlova

file transfer protocol - protokol za transfer fajlova Pravila za pristup i razmenu fajlova izmedu dva sajta

filter Uredaj koji omogućava prolazak signala odredene frekvencije

filter bank - banka filtera Kolekcija fihera koji kreiraju niz koji predstavlja komponente signala u okviru naznačenog frekventnog opsega

fingerd UNIX program koji korisnicima omoguc'ava pribavljanje infonnacija o drugima

finite state machine - konačni automat Formalni model koji opisuje skup moguc'ih stanja i prelaza izmedu stanja sistema

firewall Uredaj koji razdvaja sajt od Interneta i kontroliše tok informacija ka sajtu i od sajta

FireWire Peer-to-peer tehnologija koja omogućava povezivanje više uredaja, često sa personalnim kompjuterom

fixed routing - fiksno rutiranje Tehnike rutiranja zasnovane na informacijama koje se ne menjaju, osim u slučaju reprogramiranja

flooding algorithm - algoritam plavljenja Algoritam za rutiranje koji prenosi poruku do svih mogućih lokacija

flow control - kontrola toka Protokol koji reguliše razmenu informacija izmedu dva uredaja

foil-electret condenser microphone - mikrofon sa provodnom metalnom folijom Metod koji se u telefonima koristi za konvertovanje zvuka u elektricitet pomoću vibrirajuće dijafragme koja izaziva promenu električnog polja izmedu nje 1 elektrode u slušalici

fonvarding database - baza prosleđivanja Informacije koje uredaj koristi za rutiranje okvira sa jedne mreže na drugu

fonvard search algoritltm - algoritam pretraživanja unapred Videti: *Đijkslrin algoritam*

Fourier series - Furijeovi redovi Matematička formula koja se koristi za opis proizvoljnog periodičnog signala

fragmentation - fragmentacija Proces deljenja paketa na manje delove kako bi se očuvala kompatibilnost sa mrežnim protokolom

frame - okvir (1) ledinica informacija koje razmenjuju protokoli nižeg nivoa; (2) Kod USB-a delićvremena u trajanju od jedne milisekunde

frame bursting - sporadično slanje olcvira Metod kombinovanja okvira kod Gigabit Ethernet standarda

frame check sequence - sekvenca za proveru okvira Polje u okviru koje se koristi za proveru grešaka

frame relay Protokol koji implementira virtuelna kola

frame timer - tajmer okvira Tajmer koji se koristi kod protokola za kontrolu toka za utvrdivanje trenutka kada se ponovo šalje okvir za koji nije stigla potvrda

free space optics (FSO) Metod komuniciranja pomoću lasera bez ograničenja koja nameće optički fiber

frequency - frekvencija Učestalost ponavljanja signala

frequency-dependent code - frelcventno-zavisni kod Kod za kompresiju koji koristi prednost činjenice da se određeni karakteri javljaju češće od ostalih

frequency-dfvision multiplexing - multipleksiranje sa podelom frekvencije Proces prihvatanja analognih signala u okviru različitih propusnih opsega i kombinovanje tih signala u složeniji signal sa većim propusnim opsegom

frequency- hopping spread spectrum Metod širokog spektra koji vrši prenos, koristeć različile frekvencije u različitim trenucima

frequency modulation (frequency shift keying) - frekventna modulacija Melod predstavljanja bitova pomoću analognih signala sa različitim frekvencijama

full duplex Mod prenosa kod koga uredaj može istovremeno i da šalje i da prima podatke

fully connected topology - potpuno povezana topologija Strategija povezivanja kod koje se svaki uredaj na mreži povezuje direktno sa svakim drugim uređajem

gateway Konekcija OSI sloja 7 izmedu dve mreže

generator polynomial - generator polinoma Polinom koji se korisli kao delitelj kod CRC metoda za detekciju grešaka

geosynchronous orbit - geostacionarna orbita Orbita na kojoj se sateliti kreću istom brzinom kojom Zemlja rotira (izgledaju statično za posmatrača sa Zemlje)

Gigabit Ethernet Ethernet standard koji dostiže brzine od 1 Gbps

go-back-n protocol - go-back-n protokol Protokol za kontrolu toka sa klizajućim prozorom, kod koga primalac mora da prima okvire tačnim redosledom

graded-index multimode fiber Optički fiber čija obloga ima promenljivi indeks refrakctje

graph - graf Matematički model koji se sastoji od čvorova i grana koje povezuju čvorove

Graphics Interchange Format (GIF) Format koji se koristi za slike sa relativno manjim brojem boja i strogo definisanim granicama

gremlin Cuveno stvorenje koje je odgovorno za sve izgubljene okvire na mreži

group address - grupna adresa Adresa koja deftniše nekoliko stanica u unapred definisanoj grupi

guard band - zaštitni opseg Neiskorišćene frekvencije izmedu dva susedna kanala

guest station - gostujuća stanica Sekundardna stanica kod HDLC protokola

hacker - haker Osoba koja voli da programira radi zabave

half duplex Mod prenosa kod koga uredaj mora naizmenično da šalje i prima podatke

Hamming code - Hamingov kod Metod za korekciju grešaka koji izvodi nekoliko provera parnosti na ugovorenim pozicijama

handshake - usaglašavanje Proces defmisanja i postavljanja konekcije

hash function - hash funkcija Funkcija koja se koristi za izračunavanje vrednosti digitalnog sažetka pomke

hash value - hash vrednost Videti: *iređnost digitalnog sažetka poruke*

herlz (Hz) - herc Mera broja ciklusa signala u jednoj sekundi

hierarchical routing - hijerarhijsko rutiranje Tehnika za rutiranje kod koje su čvorovi podeljeni na grupe nazvane domeni

high-defmition television - televizija visoke defmicije Televizijska tehnologija koja obezbeduje mnogo oštriju sliku od konvencionalne televizije

High-level Data Link Control (HDLC) Protokol veze koga je defmisao ISO

hop count - broj skokova Mera kod tehnika rutiranja koja broji uređaje koje paket prede duž rute

horn antenna - antena u obliku roga Uredaj koji se koristi kod mikrotalasnih prenosa

host Kompjuter priključen na mrežu koji može da pokreče mrežne aplikacije

hot pluggable Karakteristika koja omogućava priključivanje (i isključivanje) novih uredaja bez isključivanja kompletnog sistema, ili učitavanja novog softvera koji bi omogućio funkcionisanje uređaja

HTML document - HTML dokument Fajl koji definiše šta korisnik vidi u prozoru pretaživača

hub Uredaj sloja 1 koji prima signal preko jednog porta, regeneriše ga i šalje na sve ostale portove

Huffman code - Hafmanov kod Frekventno-zavisna tehnika kompresije

Hypertext Markup Language (HTML) Jezik koji se koristi za kreiranje Web dokumenata

Hypertext Transfer Protocol (HTTP) Protokol sloja aplikacije koji se koristi na World Wide Webu za pristup i transfer Web dokumenata

IEEE 802 standards - IEEE 802 standardi Set mrežnih standarda za lokalne mreže i mreže koje obuhvataju veća gradska podaičja

in-band signaling - signaliziranje u opsegu Tehnika za signaliziranje kod koje se signali šalju istim kanalom, ili u istom nizu bitova kao i podaci

index of refraction - indeks refrakcije (prelamanja) Mera koja određuje količinu svetlosti koja se savija dok prelazi sa jednog medijuma na drugi

infrared waves - infracrveni talasi Deo spektra elektromagnetnih talasa čije se frekvencije nalaze odmah ispod frekvencije vidljive svetlosti

initialization vector - vektor inicijalizacije Podaci koji utiču na šifrovanje prvog bloka podataka u modu ulančavanja blokova

Institute of Electrical and Electronic Engineers (IEEE) Profesionalna organizacija koja objavljuje časopise, organizuje konferencije i razvija standarde

Integrated Services Digital Netvvork (ISDN) Standard za globalni digitalni komunikacioni sistem

interior routing protocol - protokol za unutrašnje rutiranje Protokol za rutiranje u okviru autonomnog sistema

intermedia synchronization - sinhronizacija medijuma Karakteristika real-time apHkacije kod koje su video i audio podati sinhronizovani

International Organization for Standardization (ISO) Svetska organizacija koju čine lela za standardizaciju iz više zemalja

Intemational Telecommunications Union (ITU) Organizacija sa uspostavljanje standarda, čiji su članovi razne naučne i industrijske organizaci-je, telekomunikacione agencicije, nadležna tela za telefoniju i ISO; ranije je bila poznata kao □□□□

Internet Kolekcija mreža koja pokreče TCP/IP protokola

Internet address - Internet adresa 32-bitni, ili 128-bitni broj koji identifikuje uredaj na Internetu

Internel Assigned Numbers Authoritv (IANA) Federalna organizacija koja je odgovorna za upravljanje i alociranje IP **adresa**

Intemet Control Message **Protocol (ICMP)** Internet protokol za podnošenje izveštaja o greškama i obezbedivanje ažuriranja rutera o najnovijim uslovima na Internetu

Inlernet Corpoiation for Assigned Names and Numbers (ICANN) Neprofitna organizacija odgovorna za upravljanje i alociranje IP adresa

Internet Engineerinq Task Force (IETF) Medunarodna zajednica čiji su članovi dizajneri mreža, prodavci i istraživači, u đjem je interesu stabilno funkcionisanje Interneta i njegov razvoj

Intemet Group Management Protocol (IGMP) Protokol koji funkcioniše izmedu hosta i lokalnog rutera koji omoguc'ava hostu da se pridružuje i napušta različite multicast grupe

Internet Protoco] (IP) Protokol sloja mreže koji je originalno razvila Advanced Research Projects agencija

Internet Protocol version □ (IPv6) Ažurirana verzija tekučeg Internet Protocola (IPv4)

Internet Radio Aplikacija koja korisnicima personalnih kompjutera omogućava pronalaženje i slušanje radio stanica preko kompjutera i Interneta

Internet Securitv Association and □□□ Management Protocol Protokol koji definiše formate paketa i pravila za razmenu paketa sa informacijama o ključu šifrovanja

Intemet service provider (ISP) Organizacija koja svojim korisnitima obezbeduje pristup Internetu

Internet worm - Internet "crv" Čuveni "uljez" na Internetu koji zagušuje sisteme i mnoge od njih primorava na gašenje

IPSec Protokol dizajniran za obezbedivanje bezbednog prenosa na nivou paketa

I series -1 serije Set dokumenata koji opisuje ISDN arhitekturu, konfiguracije, principe rutiranja i interfejse

isochronous transmission - izohroni prenos Mod prenosa u realnom vremenu kod koga podaci pristižu naznačenom brzinom

JavaScript Jezik koji se koristi za pisanje skripiova za Web aplikacije na sirani klijenta

JPEG compression - JPEG kompresija Standard za kompresiju crno-belih slika i slika sa fotografskim kvalitetom, koji je razvila Joint Photographic Experts Group (JPEG)

□□□ distribution (□□□ exchange) - distribuiranje ključa (razmena ključa) Probiem slanja ključeva šifrovanja do onih kojima su namenjene šifrovane poruke

□□□ escrovv agency - agencija za čuvanje ključeva Agencija koja je odgovorna za čuvanje privatnih ključeva za šifrovanje

laser Veoma čist uski mlaz svetlosti

last mile - poslednja milja Termin koji se primenjuje na bakarne provodnike koji povezuju telefone na telefonsku mrežu

Lempel-Ziv compression - Lempel-Ziv kompresija Tehnika kompresije koja menja stringove koji se ponavljaju sa odredenim kodovima

liglit-emitting diode (LED) Uređaj koji proizvodi manje koncentrisanu svellosi u poređenju sa laserom i često se koristi kao alternativa za njega u fiber optičkim komunikacijama

Link Access Protocol (LAP) Protokol sloja veze koji upravlja logičkim linkovima izmedu uredaja

link state routing - rutiranje zasnovano na stanju linka Slrategija rutiranja kod koje ruteri šalju i prosleduju pakete sa statusom linka izmedu dva naznačena mtera; svaki ruter koristi akumulirane informacije za kreiranje tabela mtiranja

local area network (LAN) - lokalna mreža Mreža koja se prostire na relativno maloj geografskoj oblasti i povezuje različite vrste uredaja

local exchange - lokalna centrala Lokalni ured u kome se nalazi logika za usmeravanje telefonskih poziva

local loop - lokalna petlja Linije koje povezuju telefone na lokalnu centralu

lock-up Videti: *deadlock*

Iogical link control (LLC) - kontrola logičke veze IEEE standard protokola veze koji se koristi u lokalnim mrežama za upravljanje logičkim linkovima izmedu uredaja

lossless compression - kompresija bez gubitka informacija Metod kompresije kod koga nema gubitaka informacija u toku kompresije

lossy compression - kompresija sa gubicima Metod kompresije kod koga postoje manji gubici u toku kompresije

low eartlr orbit (LEO) satellites - sateliti koji se kreću u niskoj orbiti (LEO sateliti) Sateliti koji

kraže na niskim orbitama i koriste se za definisanje svetske komunikacione mreže

machine state - stanje automata Kolekcija vrednosti pridruženih sistema u odredenom trenutku

Management Information Base (MIB) Baza podataka koju koristi SNMP, a u kojoj se naJaze opisi rutera i hostova

Manchester code - Mančester kod Šema digitalnog kodiranja kod koje signal uvek menja stanje na sredini intervala

man-in the-middle attack - man-in-the-middle napad Napad kod koga se kriminalac postavlja izmedu dva entiteta koji komuniciraju, predstavljajuči se obojici kao legitimni entitet

maximum transmission unit (MTU) - maksimalna jedinica prenosa Maksimalna veličina okvira koja može da se prenese preko mreže

MBone Mreža u okviru Interneta koja podržava rutiranje adresa ldase D

medium access control (MAC) - kontrola pristupa medijumu Niži podsloj protokola veze koji kontroliše pristup prenosnom medijumu

medium independent interface (MII) - medijum nezavisan od interfejsa Ethemet termin koji se odnosi na konekciju izmedu mrežne kartice i primopredajnika

□□□□□□ resident viruses - virusi koji su rezidentni u memoriji Kompjuterski virusi koji čekaju da se u memoriju postavi izvršni fajl

Menehune Cemralno postrojenje Aloha sistema (paketni radio) na Havajskim ostrvima

message age timer - tajmer starosti poruke Tajmer koji se održava na mostu i defmiše maksimalni period u kome se očekuje da se oglasi root most koji je izabran za spanning tree algoritam

message digest value - vrednost digitalnog sažetka poruke Broj koji se izračunava na osnovu sadržaja dokumenta

message switching - komutacija poruka Alternativa komutaciji paketa, ili komutaciji kola kod koje se preneta poruka smešta u svakom čvoru, ali razlidte poruke mogu da "putuju" preko različitih mla

microvvave transmissions - mikrotalasni prenosi Metod prenosa koji koristi elektromagnetne talase sa frekvencijama ispod frekvencije infracrvene svetlosti

modal dispersion - modalna disperzija Fenomen koji nastaje kada se svetlost reflektuje pod različitim uglovima u optičkom fiberu, tako da joj treba nešto više vremena da stigne do drugog kraja fibera

modem Uredaj koji konvertuje analogne signale u digitalne i obratno

modulation - modulacija Proces korišćenja jednog signala za promenu drugog

monitor station - monitor Token ring stanica koja je odgovorna za održavanje i kontrolu

monoalphabetic chiper - monoalfabetska šifra Primitivna tehnika za šifrovanje, kod koje se tekstualni karakter menja drugim izabranim samo na osnovu karaktera koji se menja

Morse code - Morzeov kod Kod za prenos razvijen za telegrafske sisteme kod koga se svaki karakter predstavlja nizom tačaka i crtica

Moving Pictures Expert Group (MPEG) Često se koristi za metod kompresije video fajlova; tačnije, reč je o grupi koja defmiše standarde za video kompresiju

□□□ Metod kompresije koji se obično primenjuje na audio fajlove

multicast Slanje informacija ka odredenoj grupi kompjutera

multicast address - multicast adresa Adresa koja identifikuje grupu host kompjutera

multicast tree - multicast stablo Kolekdja mtera koji učestvuju u slanju podataka ka više odredišta

Multilevel Line Transmission-Three Levels (MLT.3) Šema signalizadje kod Fast Etherneta koja koristi signal sa tri stanja

multiplexer - multiplekser Uređaj koji kombinuje signale sa nekoliko ulaza i šalje ih dalje preko jednog kanala

multiport repeater - višeportni repetitor Sinonim za *hnb*

mux Videti: *multiplekser*

National Institute of Standards and Tedmologv (NIST) Agencija američkog ministarstva trgovine (United States Department of Commerce) za uspostavljanje standarda

near-end crosstalk (NEXT) Interferenca na liniji koja izaziva paralelni prenos signala

negative acknowledgment (NAK) negativna potvrda Indikacija kontrole toka da okvir nije ispravno primljen, ili uopšte nije primljen

network - mreža Kolekcija uredaja koji pokreću softver koji im omoguc'ava medusobnu komunikaciju preko nekog prenosnog medijuma

network interface card - mrežna interfejs kartica (NIC) Logičko kolo koje se postavlja u personalni kompjuter radi povezivanja na mrežu

network layer - sloj mreže Protokol sloja 3 odgovoran za rutiranje nekih paketa preko mreže

network termination 1 (NTI) ISDN oznaka za "neinteligentne" uredaje koji su zaduženi za fizičke i elektriare karakteristike prenosa signala

network termination 2 (NT2) ISDN oznaka za "inteligentne" uredaje koji mogu da izvršavaju funkcije OSI slojeva 2 i 3

network topology - mrežna topologija Način fizičkog povezivanja mrežnih uredaja

noise - šum Neželjeni signali koji ometaju preneti signal

noiseless channel - bešumni kanal Kanal bez šumova

noisy lines - linije podložne šumovima Linije na kojima postoji veća količina šuma

nonbroadcast frame - neemisioni okvir Okvir namenjen specifičnom odredištu

non-data-J i non-data-K Digitalni signal koji nije u skladu sa bilo kojim Mančester kodom za definisanje bitova

nonpersistent CSMA - neperzistentni CSMA Protokol kod koga, nakon što dode do kolizije, uredaj ne nadgleda prenosni medijum, umesto da čeka jedan vremenski slot pre provere aktivnosti

nonreturn to zero (NRZ) Šema digitalnog kodiranja kod koje se 0 i 1 predstavljaju specifičnim naponskim nivoima

no-prefix property - no-prefix svojstvo Svojstvo koje nalaže da se odredeni karakter u kodu nikada ne pojavi kao prefiks drugog koda

normal response mode - mod normalnog odziva HDLC mod kod koga primarna stanica kontroliše komunikaciju

null modem Uredaj koji se koristi za direktno povezivanje dva DTE-a

Nyquist theorem - Nikvistova teorema Teorijski rezultat koji povezuje brzinu prenosa podataka sa brzinom bauda signala i brojem komponenala signala

octet - oktet Grupa od osam bitova

odd parity - neparna parnost Metod za detekciju grešaka kod koga se ekstra bit dodaje sa vrednošču 0, ili 1, tako da ukupni broj jedinica u jedinici prenosa bude neparan

one-time pad Tehnika šifrovanja koju je nemoguće "razbiti", a kod koje se koristi ključ iste dužine kao i obični tekst (koristi se samo jednom)

one-way hash function - jednosmerna hash funkcija Hash funkcija koja uvek daje jedinstvenu vrednost digitalnog sažetka za svaki dokument

Open Shortest Path First (OSPF) Varijacija mti-ranja zasnovanog na stanju Iinka koja obezbeduje dodatne karakteristike, kao što su autentifikacija i balansiranje opterećenja

open system - otvoreni sistem Set protokola koji omogućava komunikaciju dva različita sistema, bez obzira na njihovu osnovnu arhi tekturu

Open Systems Interconnect (OSI) Standardni protokol koji je razvila International Standards Organization za implementiranje otvorenog sistema

operation, administration, and maintenance (OAM) - operacija, administriranje i održavanje Protokoli koji se koriste za održavanje i praćenje komunikacija i za olklanjanje problema u njima

optical fiber - optički fiber Komunikadoni medijum koji se sastoji od tankog staklenog kabla pomoću koga se prostire svellost

orphan frame - ispušteni okvir Token ring okvir koji beskonačno cirkuliše prstenom, jer ni jedan uredaj neće da ga ukloni sa prstena

out-of-band signaling - signaliziranje van opsega Tehnika signaliziranja kod koje se signali šalju zasebnim kanalom, ili izvan niza bitova podataka

outstanding frames - nerešeni okviri Okviri koji su poslati, ali ih još uvek nije potvrdio protokol za kontrolu toka

packet - paket (1) Prenosna jedinica naznačenog protokoia; (2) Kod USB-a pakovana grupa bitova

packet elimination - eliminacija paketa Šema kontrole zagušenja koja eliminiše neke pakete, ako ih postoji isuviše u nekom čvoru

packet filtering - filtriranje paketa Metod koga koristi ftrewall za ograničavanje paketa na osnovu njihovog sadržaja

packet header - zaglavlje paketa Kontrolne informacije u paketu

packet jitter - pomeranje paketa Sporedni efekat kod real-time aplikacija kod kojih kašnjenja na mreži izazivaju dolazak paketa u nepravilnim intervalima

packet sniffer Program koji pregleda sadržaj paketa podataka dok "putuju" preko mreže

packet-switched network - mreža sa komutacijom paketa Mreža kod koje se poruke dele na manje delove nazvane paketi koji se prenose nezavisno

parabolic dish reflector - parabolićna satelilska antena Mikrotalasna antena čiji je tanjir paraboličnog oblika

parallel transmission - paralelni prenos Mod prenosa kod koga se nekoliko bitova prenosi istovremeno

paritv bit - bit parnosti Ekstra bit koji se koristi za detekciju grešaka, a čija je vrednost 0, ili 1, tako da je ukupni broj jedinica ili paran, ili neparan

path - putanja Sekvenca čvorova na mreži preko koje podaci moraju da pređu na putu od pošiljaoca do primaoca

peer-to-peer networking - peer-to-peer umrežavanje Mogućnost grupisanja kompjutera tako da svi komuniciraju bez posredstva centralizovanog servera

peer-to-peer protocol - peer-to-peer protokol Protokol koji omogućava komunikaciju izmedu više uredaja, gde bilo ko može da inicira konekciju bez posredovanja centralnog uredaja za koordinaciju

period - period Vreme potrebno periodičnom signalu da se jednom ponovi

periodic signal - periodični signal Signal koji se menja u vremenu, ali kontinuelno ponavlja odredeni rrzorak

Perl Skript jezik koji se često koristi u Unix i Linux okruženjima

permanent virtual circuit (PVC) - permanentno virtuelno kolo Tip virtuelnog kola kod koga nije neophodno izvršavanje protokola za uspostavljanje konekrije pre transfera podataka

personal area netvvork - personalna mreža Kolekcija manjih potrošačkih uredaja opremljenih radio primopredajnicima za medusobnu komunikaciju

Petri net Način modelovanja protokola pomoću grafa za predstavljanje stanja i prelaza

phase modulation - fazna modulacija Metod promene signala promenom faznog pomaka

phase shift - fazni pomak Horizontalni pomak periodičnog signala

phase shift keving Videti: *fazna modulacija*

physical layer - fizički sloj Protokol sloja 1 odgovoran za definisanje električnih i fizičkih svojstava prenosnog medijuma

physical medium dependent sublayer - podsloj zavisan od fizičkog medijuma Elhernet termin za mesto na kome se signali generišu

picture element - element slike Videti: *piksel*

piggyback acknow!edgment - piggvback potvrda Tehnika slanja potvrde u okviru zajedno sa podacima

pixel - piksel Najmanja vidljiva komponenta slike na televizijskom, ili video ekranu.

plaintext - obični tekst Nešifrovana poruka

point-to-point link HDLC link preko koga primarna stanica komunicira sa sekundarnom stanicom

Poll bit HDLC fleg koji primarnoj staniri omogućava slanje zahteva za odziv sekundarne stanice

polyalphabetic chiper - polialfabetsko šifrovanje Tehnika šifrovanja kod koje se svaka pojava karaktera menja drugačijim karakterom, u zavisnosti od originalnog karaktera i njegove pozicije u poruri

polymorphic virus - polimorfni virus Virus koji može da mutira, ili da se menja da bi izbegao detekciju kada "inficira" fajl

polinomial - polinom Matematićki izraz formiran sabiranjem članova koji predstavljaju umnoške konstante i nepoznatog člana stepenovanog poizitivnim celim brojem

port number - broj porta TCP vrednost koja identifikuje serversku aplikaciju

p-persistent CSMA - p-perzistentni CSMA Protokol kod koga, nakon kolizije, stanica nadgleda prenosni medijum i, kada registruje neaktivnost, prenosi podatke sa verovatnoćom p $(0 < p < 1)$

preamble Specijalni bitski uzorak koji se javlja na početku nekih formata za okvire

presentation layer - sloj predstavljanja OSI protokol sloja 6

Pretty Good Privacy Prograraski paket koji može da se koristi sa emailom za šifrovanje i đigitalno potpisivanje poruka

primary rate - primarna brzina □□ standard za ISDN koji se sastoji od 23 B kanala i 1 D kanala

□□□□□ station - primarna stanica Vrsta slanice koju HDLC bira za upravljanje tokom podataka, izdavanje komandi drugim stanicama i delovanje u skladu sa njihovim odzivima

private branch exchange (□□□) - privatna telefonska centrala Privatni telefonski sistem

protocol - protokol Skup pravila po kojima dva, ili više uredaja komuniciraju

protocol converters - konvertori protokola Logika za konvertovanje jednog protokola u drugi

pseudoternary coding - pseudoternarno kodiranje Videli: *inverzno naizmenično označavanje*

psvchoacoustic model - psihoakustični model Model zasnovan na čulu sluha, koje odreduje koji se zvuci maskiraju drugim zvucima i koji se mogu ukloniti uz manje, ili bez primetnih gubitaka

public data netvvorks - javne mreže za prenos podataka Mreže sa komutacijom paketa kojima upravlja vlada, ili javna dobra

public □□□ cryptosystem - kriptosistem javnog ključa Tehnika šifrovanja kod koje nema pokušaja da se zaštiti identitet ključa šifrovanja

pulse amplitude modulation - impulsna amplitudska modulacija Tehnika semplovanja analognog signala u pravilnim inlervalima i generisanja impulsa sa amplitudom jednakora amplitudi semplovanog signala

pulse code modulation - impulsna kodna modulacija Slična impulsnoj amplitudskoj modulaciji, osim što amplituda impulsa mora da bude iz unapred defmisanog skupa vrednosti

pure Aloha - čisti Aloha Protokol razvijen na University of Hawaii za paketni radio sistem

Purge frame - Purge okvir Kontrolni okvir kod token ring mreže koji uklanja irelevantne signale sa prstena

Q series - Q serije Serije □□-□ dokumenata koji opisuju slojeviti protokol poznat pod nazivom Signaling System 7, standard za obezbedivanje funkcionalnosti integrisanih digitalnih mreža

quadrature amplitude modulation - kvadratuma amplitudska modulacija Tehnika modulacije kod koje se bitovi dodeljuju analognom signalu u zavisnosti od kombinacije amplitude i faznog pomeraja

quality od service (QoS) - kvalitet servisa Specifikacija tipa servisa koji zahteva aplikacija

Real-Time Transport Protocol (RIP) Protokol transportnog sloja dizajniran za podršku real-time aplikacija

reassembh/ deadlock - samrtni zagrljaj koji nastaje zbog sastavljanja paketa "Samrtni zagrljaj" koji nastaje zbog nedostatka prostora u baferu kada se paketi prihvataju iz više izvora

reassembh/ timer - tajmer ponovnog sastavljanja Tajmer koji Internet Protocol koristi za definisanje intervala u kome se očekuje prijem svih fragmenata paketa

receiving window - prijemni prozor Parametar kontrole toka koji definiše koji okviri mogu biti primljeni

Record Route option - Record Route opcija Parametar Internet protokola koji definiše rutu kojom paket treba proslediti

reference points (R, S, T U) - referentne tačke (R, S, T, U) □□-T-definisane referentne tačke koje se koriste za deljenje ISDN funkcionalnih grupa

refraction - refrakcija Fenomen promene pravca svetlosti dok prelazi sa jednog medijuma u drugi

Relative encoding - Relativno kodiranje Videti: *diferencijalno hodiranje*

remote login - daljinsko logovanje Proces logovanja na kompjuter smešten na udaljenoj lokaciji

Remote Monitoring (RMON) Protokol za upravljanje mrežom

repeater - repetitor Konekcija sloja 1 koja prima signale i regeneriše ih pre daljeg slanja

replay attack - napad odgovora Neautorizovano kopiranje legitimnih paketa radi kasnijeg slanja kako bi se imitirala legitimna aklivnost

reply - odgovor Poruka koja se šalje kao odgovor na neku drugu

Request for Comments (RFC) Serija dokumenata sa beleškama o istraživanjima

reservation system - sistem rezervacija Token ring protokol koji omogućava uredaju da pokuša da rezerviše token u sledećem prolasku

Resource Reservation Protool (RSVP) Protokol za transport □□□□□ koje sadrže informacije o odredenom toku podataka i zahtev za rezervisanje resursa koji su neophodni za ispunjavanje naznačenog kvaliteta servisa

reverse-path broadcasting - emitovanje obrnutom putanjom Protokol koji se koristi za multicasting kod koga ruter utvrduje izvor primljenog paketa i emituje ga samo ako je paket stigao iz smera tog izvora

Rijndael algorithm - Rijndaelov algoritam Složeni standard za šifrovanje, zasnovan na različitim operacijama na nivou bitova, u kombinaciji sa složenim matematičkim operacijama

ring topology - topologija prstena Kružno postavljanje uredaja koji mogu direktno da komuniciraju samo sa svojim susedom

roaming Karakteristika koja mreži omogućava praćenje lokacije pokretnog uredaja

root bridge - root most Spedjalno dizajnirani most koji spanning tree algoritam odreduje kao koren spanning stabla

root port Port mosta koji odgovara najjeftinijoj putanji do root mosta u spanning stablu

route - ruta Sekvenca uredaja preko kojih paket, ili okvir moraju da predu na putu od izvora do odredišta

routed Program za rutiranje razvijen na Universitv of California - Berkelev za aitiranje u okviru lokalne mreže

route designators - oznake ruta Sekvenca ID-ova LAN-a i mostova koja definiše putanju

route discovery - otkrivanje rute Proces koji se koristi na mostovima sa izvornim rutiranjem za utvrđivanje putanje do odredenog uredaja

route learning - učenje rute Proces kod koga svaki most, ili komutator "uče" šta treba da postave u svoju tabelu aitiranja

router - ruter Konekcija sloja 3 izmedu dve mreže

routing algorithm - algoritam za rutiranje Metod koji se koristi za utvrdivanje rute

routing directorv - direktorijum rutiranja Videti: *tabala rutiranja*

Routing Information Protocol (RIP) Strategija za rutiranje koja koristi brojač "skokova" za utvrdivanje putanje na mreži

routing matrix - matrica rutiranja Matrica koja definiše sledeći čvor na ruti između para čvorova

routing table - tabela ratiranja Baza podataka koju most, ili ruter koriste za utvrdivanje pravca u kome treba poslati primljene okvire

RS-232 standard Videti: *EIA -232 standarđ*

run-lengdi encoding - run-length kodiranje Tehnika kompresije koja menja nizove bitova (ili bajtova) brojem bitova (ili bajtova) u nizu

sampling frequency - frekvencija semplovanja Učestalost semplovanja analognih signala

satellite - satelit Objekat koji kruži oko Zemlje

satellite radio - satelitski radio Servis koji emituje radio signale preko cele zemaljske kugle tako da pretplatnici mogu da ih slušaju iz bilo kog dela sveta

satellite transmission - satelitski prenos Mikrotalasni prenos ka satelitu u orbiti, ili iz njega

S-box Struktura koja defmiše kako se jedan bajt menja drugim u loku šifrovanja

script - skrip Fajl koji sadrži kolekciju komandi koje će biti izvršene

scripting language - skript jezik Jezik čiji se izvorni kod interpretira u vreme izvršenja

secondary station - sekundardna stanica Tip stanice koju HDLC označava kao stanicu koja odgovara primamoj stanici

Secure □□□□ (scp) Protokol za bezbedni transfer fajlova

Secure Hash Algorithm (SHA) Jednosmema hash funktija koja se korisli za autentifikaciju dokumenata

Secure Shell (ssh) Protokol za bezbedno daljinsko logovanje

Secure Sockets Layer (SSL) Protokol koji autentifikuje sajtove i pregovara o strategijama šifrovanja izmedu njih

security - bezbednost Tiče se zaštite, ili skrivanja informacija od neautorizovanih ljudi

segment Kolekcija formatiranih informacija za protokol (obično TCP)

selective repeat protocol - protokol selektivne retransmisije Protokol za kontrolu toka sa klizajućim prozorima kod koga pošiljalac i primalac definišu prozor koji odreduje okvire koje mogu da pošalju, ili prime

self-svnchronizing code - samosinhronizujući kod Šema digitalnog kodiranja kod koje signal uvek menja stanje na sredini bitskog intervala

sending window - predajni prozor Parametar kontrole toka koji odreduje koji okviri mogu da se pošalju

sequence number - broj sekvence Broj koji se koristi za defmisanje redosleda okvira, ili paketa

serial transmission - serijski prenos Mod prenosa kod koga se svi bitovi šalju u nizu

server Mrežni uredaj koji reaguje na zahteve korisnika mreže

session - sesija Logička konekcija izmedu dva krajnja korisnika

session layer - sloj sesije OSI protokol sloja 5

Shamir's memod - Shamirov metod Metod distribuiranja ključa kod koga za utvrdivanje ključa mora da bude prisutan odredeni broj ljudi

shielded twisted pair - zašticene upredene parice Dve upredene provodne žice sa metalnim zaštitnim omotačem koji ih štiti od spoljašnje električne interference

shift register - pomerački registar Deo kola koji se koristi za implementiranje ciklične provere redundantnosti

shortest path algorithm - algoritam najkraćeg puta Logika koja se koristi za utvrđivanje najkraćeg puta između dve tačke

signal constellation - konstelacija signala Dijagram koji koristi iscrtane tačke za definisanje svih legitimnih promena signala koje modem prepoznaje

signal ground - uzemljenje Naponski nivo u odnosu na koji se mere svi ostali signali

signaling data link - signalizacija veze Najniži sloj protokola Signaling Svstem 7, koji obezbeduje fizičke i električne specifikacije

signaling link laver - signalizacija sloja veze Drugi sloj protokola Signaling System 7, koji obezbeduje pouzdane komunikacije između dve susedne taćke na mreži

signaling network layer - signalizacija sloja mreže Treči sloj protokola Signaling System 7, koji obezbeduje pouzdan transfer poruka izmedu dve signalne tačke

signal speed - brzina signala Brzina kojom se signali prostim kroz medijum

Signaling System 7 Četvoroslojni protokol koji defmiše standard za funkcionalnost integrisane digitalne mreže.

signal-to-noise ratio - koefkijent signal-šum Mera koja se koristi za utvrđivanje količine šuma u signalu

Simple Mail Transfer Protocol (SMTP) Standardni protokol za email u TCP/IP slatpu protokola

Simple Network Management Protocol (SNMP) Upravljački protokol dizajniran za obezbedivanje ispravnog funkcionisanja mrežnih protokola i uredaja

simplex communication - simplex komunikacija Mod kod koga se komunikacija obavlja samo u jednom smeru

single-bit error - greška u jednom bitu Greška koja utiče samo na jedan bit

single-mode fiber - fiber sa jednim modom Optički fiber veoma malog prečnika dizajniran za redukovanje broja uglova pod kojima se svetlost reflektuje od omotača samo na jedan mogući ugao

sliding window protocol - protokol klizajućih prozora Protokol za kontrolu toka kod koga predajni i prijemni uredaji ograničavaju broj okvira koje mogu da pošalju, □ prime

slotted Aloha protocol - slotovani Aloha protokol Aloha protokol koji od uredaja zahteva da prenosi podatke samo na početku slota

slow start - lagani start Deo procesa startovanja kod TCP lagoritma kontrole toka, kod koga predajni entitet pokušava da pošalje maksimalan broj okvira pre nego šlo zagušenje izazove kašnjenja

smurf attack - smurf napad Vrsta napada odbijanja servisa

socket - soket UNIX konstmkcija i mehanizam koji se koristi za čitanje i upis na mreži

source address - izvorna adresa Adresa uredaja koji šalje paket, ili okvir

Source Quench ICMP kontrolna poruka koja zahteva redukovanje brzine kojom se paketi šalju

source-routing bridge - most sa izvornim rutiranjem Most koji rutira okvir na osnovu sadržaja oznake mte u njemu

spanning tree algorithm - spanning tree algoritam Distribuirani algoritam koji se izvršava na mostovima, ili komutatorima radi utvrdivanja konekcije izmedu svih LAN-ova učesnika bez redundantnih putanja

spatial frequency - prostorne učestalosti Vrednosti kod JPEG šeme kompresije koje direktno odreduju koliko se vrednosti piksela mogu menjati u funkdji njihove pozicije u bloku

split horizon - podeljeni horizont Modifikacija algoritma za rutiranje kod koga čvor ne šalje informacije koje je primio od susednog čvora nazad do tog čvora

splitter - delitelj Uredaj koji deli signal na razlidte komponente

spoofmg Videti: *address spoofing*

spot beam - ciljni mlaz Tip antene koji omogućava emitovanje satelitskih signala samo u veoma maloj oblasti

spread spectrum - široki spektar Mod kod bežičnih prenosa koji rasipa spektralnu energiju signala preko šireg frekvenmog opsega

spyware Program koji se umec'e na sistem bez znanja korisnika i beleži sve aktivnosti koje korisnik izvede na kompjuteru

stacking station - stanica za odlaganje Token ring stanica koja je podigla prioritet tokena

standard llgovoreni način da se nešto **uradi**

start bit Signal koji se koristi kod asinhronog prenosa kako bi se primalac obavestio da stižu podaci

start of frame delimiter Specijalni bitski uzorak koji označava početak okvira

star topology - topologija zvezde Postavka uređaja kod koje se komunikacija odvija preko centralnog uredaja

stateful inspection Metod koji firewall koristi za ograničavanje paketa na osnovu njihovog sadržaja i onoga što se ranije desilo

state transition - prelaz stanja Prelazak sistema iz jednog stanja u dmgo zbog nekog dogadaja

state transition diagram - dijagram prelaza stanja Model sistema koji opisuje sva moguća stanja i dogadaje koji izazivaju promene stanja

static routing - statičko rutiranje Strategija za rutiranje koja koristi informacije koje se ne menjaju u vremenu

statistical multiplexer - statistički multiplekser Multiplekser sa podelom vremena koji kreira okvire promenljive velidne

step-index multimode fiber Optiđci fiber velikog prečnika koji omoguc'ava prelamanje svetlosti od omotača pod nekoliko razliđtih uglova

stop-and-wait protocol - stop-and-wait protokol Protokol za kontrolu toka kod koga pošiljalac šalje okvir i čeka na potvrdu pre nego što pošalje sledeći

stop bit Signal koji se koristi kod asinhronog prenosa kako bi se označio kraj prenosa

store and fonvard - smesti i prosledi Mrežni protokol kod koga se poruka u potpunosti smešta u prelaznom čvoru na putu do odredišta

store-and-forward deadlock "Samrtni zagrljaj" koji nastaje kada ni jedan čvor ne može da šalje okvire do sledećeg čvora, jer su im baferi puni

strobe signal - strob signal Signal koji ostaje konstantan kada se signal podataka menja, a menja se kada je signal podataka konstantan

studv - studiranje Proces u okviru koga neko čita materijal, beleži ono što ne razume, ponovo čita materijal kako bi ga bolje razumeo, kreira sledeću listu onoga što i dalje ne razume, ponavljajući taj postupak sve dok lista ne bude prazna

subnet - podmreža Nezavisna fizička mreža koja je deo veće mreže

subnet mask - maska podmreže Bitski string jedinica koji određuje koji bitovi u adresi predstavljaju ID podmreže

supernetting - definisanje super mreže Grupisanje nekoliko manjih mreža i vizuelno predstavljanje u vidu jedne velike

supervisorv frame - supervizorski okvir Okvir koji HDLC koristi da bi ukazao na status stanice, ili za slanje negativnih potvrda

switch - komutator Uredaj sloja 2 koji može da prosleduje okvire sa jedne mreže na drugu; termin se obično primenjuje na slučajeve kada se povezuju mreže sa sličnim tehnologijama

switched Ethernet - komutirani Ethernet Ethernet strategija povezivanja kod koje se komutatori koriste za povezivanje svih uredaja

switched virtual circuit (SVC) - komutirano virtuelno kolo Vrsta virtuelnog kola koja zahteva protokol za uspostavljanje poziva

SYN character - SYN karakter Karakter koji se koristi kod protokola orijentisanih bajtovima za označavanje početka okvira

Synchronous Data Link Control (SDLC) Bitovima orijentisani protokol veze koji j< razvio IBM; sličan je HDLC protokolu

Svnchronous Digital Hierarchy (SDH) Optički noseći sistem sličan SONET-u

Synchronous Optical Network (SONET) Optička mreža koja koristi multipleksiranje sa podelom vremena kako bi se omoguđo simultani prenos preko kanala; kontroliše se taktom i prenosi podatke na brzinama od 155,5 Mbps, pa do 2 Gbps u ekstremnim situadjama

synchronous payload envelope (SPE) Podad smešteni u SONET okviru

svnchronous transmission - sinhroni prenos Mod prenosa kod koga se bitovi postavljaju u okvir i šalju u sekvenci

Tl Digitalni nosioc sa 24 kanala koji dostiže bitske brzine od 1,544 Mbps

teleconferencing - telekonferencije Komunikacioni sistem koji omogućava ljudima na razlifitim lokacijama da vide i čuju šta ostali učesnid konverzacije govore

telegraph - telegraf Primitivni komunikacioni uredaj koji se sastoji od izvora napajanja, preklopnika i senzora

telemetry - telemetrija "Osluškivanje" statusa, ili očitavanje podataka na udaljenim lokacijama

telepresence - prisustvo na daljinu Koncept projektovanja na udaljenu lokaciju

Telnet Protokol virtuelnog terminala na sloju aplikacije koji omoguc'ava daljinsko logovanje

temporal redundancy - privremena redundansa Redundansa definisana prisustvom slićnih podataka u sukcesivnim okvirima

terminal adapter - terminalni adapter Uredaj koji je dizajniran za korišćenje sa ISDN TE2 opremom za konvertovanje signala u ISDN-kompatibilni format

terminal equipment 1 (TEl) Oznaka primarne funkdonalne grupe za ISDN ureclaje

terminal equipment 2 (TE2) Oznaka primarne funkdonalne grupe za ne-ISDN uređaje

terminatori Elektronski uredaji koji se postavljaju na kraj medijuma kako bi se sprečio eho signala

ThickNet Debeli, manje fleksibilni koaksijalni kabl koji je korizšćen u starijim lokalnim mrežama

ThinNet Tanji, fleksibilniji koaksijalni kabl koji se koristi u lokalnim mrežama; obično se koristi za povezivanje mrežnih uredaja

three-way handshake - trosmerno usaglašavanje Mehanizam za uspostavljanje konekcije, koji se sastoji od zahteva za konekcijom, potvrde zahteva i potvrde za potvrdu

time-division multiplexing - multipleksiranje sa podelom vremena Proces koji uključuje prihvalanje digitalnih signala iz nekoliko izvora, smeštanje u jedan okvir i slanje okvira preko jednog kanala

Time Exceeded ICMP kontrolna poruka koja se šalje do izvornog uredaja kada se paket, ili nesastavljeni fragmenti ispuste sa mreže zbog isteka tajmera

Timestamp Reply ICMP kontrolna poruka koja se šalje kao odgovor na Timestamp Request poruku

Timestamp Request ICMP kontrolna □□□□□ koja se šalje do udaljenog hosta kada lokalni host treba da proceni vrerae punog kruga od njega do udaljenog hosta i nazad

Time to Live Polje u paketu Internet Protokola koje definiše maksimalni period u kome paket može da se nalazi na mreži

token Specijalni okvir koji kruži izmedu svih uredaja na mreži i koristi se za utvrdivanje kada uredaj može da šalje informacije preko mreže

token ring network - token ring mreža Mreža sa topologijom prstena koja kontroliše pristup prstenu pomoću protokola za prosledivanje tokena

traceroute Komanda za praćenje rute kojom "putuje" Internet paket

transceiver - primopredajnik Logičko kolo na mrežnoj kartici koje implementira neke protokole neophodne za pristup mreži

Transmission Control Protocol (TCP) Transportni protokol koji se koristi na Internetu

transparent bridge - transparentni most Most koji kreira i ažurira sopstvene tabele rutiranja

transparent data - transparentni podaci Mod prenosa kod koga prijemni uredaj ne reaguje na sadržaj dolazećih bajtova

transponder Satelitski uredaj koji prihvata signal na jednoj frekvenciji, a šalje ga dalje pomoc'u druge frekvencije

transport layer - transportni sloj Protokol sloja 4 odgovoran za komunikaciju krajnjih tačaka

Transport Layer Security (TLS) Protokol koji autentifikuje sajtove i pregovara o strategijama šifrovanja izmedu njih

transposition diiper - šifrovanje premeštanjem Tehnika šifrovanja koja preureduje karaktere u običnom tekstu

triple DES - trostruki DES Metod šifrovanja koji se definiše primenom DES algoritma tri puta zaredom

trit Jedinica informacija sa tri moguće vrednosti

tunneling - tunelovanje Ugradivanje IPv6 paketa unutar IPv4 paketa radi prenosa IPv6 paketa preko mtera koji nisu kompatibilni sa IPv6

twisted pair - upredene parice Komunikaciono kolo koje se sastoji od dve upredene izolovane žice

two-way handshake - dvosmemo usaglašavanje Mehanizam za uspostavljanje konekcije, koji se sastoji od zahteva za konekcijom i potvrde zahteva

unbalanced circuit - nebalansirano kolo Kolo koje koristi jednu liniju za prenos signala i zajedničko uzemljenje

unicast address - unicast adresa Adresa koja identifikuje jedan host

Unicode Šesnaestobitni kod koji omogućava kodiranje različitih karaktera i simbola iz raznih jezika

Uniform Resource Locator (URL) Tekstualni siring koji defmiše Imemet lokaciju i protokol koji se koristi za pristup dokumentu na toj lokaciji

universal serial bus (USB) - univerzalna serijska magistrala Master/slave protokol koji omoguc'ava povezivanje više uredaja na personalni kompjuter

unrestricted flow control - neograničena kontrola toka U suštini, predstavlja nedostatak kontrole toka; predajni uredaj šalje okvire i ne pokušava da ograniči broj okvira koje šalje

unshielded ttvisted pair - nezaštićene upredene parice Dve upredene provodne žice koje nemaju metalni zaštitni omotač za zaštitu od spoljašnje električne interference

uplink - uzlazni link Konekcija od zemaljske stanice do satelita

upstream neighbor - prethodni sused Termin kod prosledivanja tokena koji se odnosi na uredaj od kojeg je token primljen

urgent data - urgentni podaci Podaci na TCP segmentu koji moraju da se isporuže višim slojevima što **pre**

Urgent Pointer Polje u TCP segmentu koje ukazuje na urgentne podatke

User Datagram Protocol (UDP) Transportni protokol bez uspostavljanja konekcije

very high frequency (VHF) - veoma visoka **frekvencija** Televizijski prenos koji koristi elektromagnetne talase izmedu 30 i 300 MHz

very **small aperture terminal (VSAT) sistem** Satelitski komunikacioni sistem koji koristi male staletiske antene

videoconferencing - videokonferencije Sistem koji ljudima sa različitih lokacija omogučava da se vide **i** čuju u realnom vremenu

virtual call - virtuelni poziv Vrsta virtuelnog kola koje zahteva protokol za uspostavljanje poziva pre transfera podataka

virtual circuit (route) - virtuelno kolo (ruta) Logička konekcija koja se postavlja pre transfera podataka sa komutacijom paketa i za pakete koji "putuju" preko istih mrežnih čvorova.

virtual file structure - struktura virtuelnog fajla Struktura fajla koji mreža podržava radi transfera fajla

virtual LAN (VLAN) - virtuelni LAN Grupa uredaja logički grupisanih nezavisno od njihove fizičke lokacije

virtual palh - virtuelna putanja Kod ATM-a sekvenca ATM komutatora duž virtuelnog kola

virtual terminal protocol - protokol virtuelnog terminala Protokol koji definiše komunikaciju izmedu aplikacije i terminala, nezavisno od karakteristika terminala

virus Neaulorizovani set instrukcija koji se širi sa jednog kompjntera na drugi, ili preko mreže, ili perifemih transfera

voice over IP Isporuka govornih podataka preko IP paketa

waveguide - talasni vodič Cilindrična cev koja predstavlja deo antene u obliku roga

wave-division multiplexing - multipleksiranje sa podelom frekvencije Kombinovanje signala predstavljenih različitim frekvencijama vidljive svetlosti

well-known ports - opštepoznati portovi TCP brojevi portova koji su dodeljeni uobičajenim Internet aplikacijama

wide area network (WAN) - mreža šireg geografskog područja Mreža koja se prostire na većim geografskim rastojanjima i često povezuje više manjih nireža koje koriste razlićite protokole

window - prozor Apstraktni koncept za defmisanje podskupa okvira radi kontrole toka izmedu dva uredaja

Wired Equivalency Privacy Bezbednosni protokol koji se koristi za 802.11 bežični LAN standard

wireless communications - bežične komunikacije Komunikacije koje ne zavise od fizičkih konekcija

wireless fldelity (Wi-Fi) Varijacija 802.11 standarda, koja koristi radio talase iz opsega od 2.4 GHz

wireless LAN - bežlčni LAN LAN protokol koji omogućava komunikaciju uredaja nezavisno od fizičkih konekcija, kao što su vodovi, koaksijalni kabl, ili fiber

World Wide Web Termin koji se koristi za kolekciju dokumenata, linkova ka drugim dokumentima i protokola za pristup tim dokumentima preko Interneta

worm - "crv" Program koji upada u sistem i potencijalno namšava njegovu bezbednost

□.21 interface standard - □.21 standard za interfejsc Standard koji omoguc'ava komunikaciju DTE i DCE

□.25 standard Standard za povezivanje uredaja na mrežama sa komutacijom paketa

□.509 certifkate - □.509 serlifikat Sertifikat za dokazivanje identiteta hosta koji izdaje nadležno telo

X-OFF Karakter za kontrolu toka koji zaustavlja dolazak podataka

X-ON Karakter za kontrolu toka koji omogućava nastavak prislizanja podataka

Skraćenice

2B1Q (Two binary, one quarternary) dvobinarni, jednokvaternarni
3DES (Triple DES) tiostruki DES
AAL (ATM adaptation layer) ATM adaptacioni sloj
ABM (Asvnchronous balanced mode) asinhroni balansirani mod
ACK (Acknowledgment) potvrda
ADSL (Asvmmetric DSL) asimetrićni DSL
AES Advanced Encryption Standard
AM (Amplitude modulation) amplitudska modulacija
ANSI (American National Standards Institute)
AP (Access point) pristupna tačka
ARM (Asynchronous response mode) mod asinhronog odziva
ARP (Address Resolution Protocol)
ARPA Advanced Research Projects Agency
ARQ (Automatic repeat request) automatski zahtev za retransmisiju
AS (Autonomous system) autonomni sistem
ASCII American Standard Code For Information Interchange
ASK Amplitude shift keying amplitudska modulacija
ASN.l Abstract Syntax Notation 1
AT&T American Telephone and Telegraph
ATM (Asynchronous Transfer Mode) asinhroni mod prenosa
□□ Attachment unit interface
BCC Block Check Character
BCD (Binary-coded decimal) binarno kodirani decimalni kod
BGP Border Gateway Protocol
B-ISDN (Broadband ISDN) širokopojasni **ISDN**
BPDU Bridge protocol data unit
BSC (Binary synchronous communication) binarne sinhrone komunikacije
BSS (Basic service set) osnovni skup servisa
CA (Certificate authority) nadležno telo za izdavanje sertifikata
CATV (Cable TV) kablovska televizija
□□□ (Computer branch exchange) kompjuterska centrala
CCITT Comite Consultatif International de Telegraphique et Telephonique
CGI Common Gateway Interface
CIDR (Classless interdomain routing) besklasno ratiranje između domena
CIA (Committed information rate) angažovana brzina prenosa informacija
CMIP Common Management Information Protocol
CMOT CMIP over TCP
CPU (Central processing unit) centralna procesorska jedinica
CRC (Cyclic redundancy check) ciklična provera redundanse
CS (Convergence sublayer) podsloj konvergencije
CSMA Carrier Sense Multiple Access
CSMA/CA Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD Carrier Sense Multiple Access with Collision Detection
DCE (Data circuit-terminating equipment) oprema za razmenu podataka
DCF Distributed Coordination Function

DCI (Discrete cosine transform) diskretna kosinusna transformacija

DES Data Encryption Standard

DHCP Dynamic Host Configuration Protocol

DIFS DCF interframe space

DLE Data Link Escape

DNS Domain Name System

DoD (Department of Defense) Ministarstvo odbrane

DoS (Denial of service) odbijanje servisa

DPSK Differential phase shift keying diferencijalna fazna modulacija

DS (Distributed system) distribuirani sistem

DSL (Digital Subscriber Line) digitalna pretplatnička linija

DSLAM (DSL access multiplexer) DSL pristupni multiplekser

DSN Deep Space Network

DSS Digital Signature Standard

DSSS Direct-sequence spread spectrum

DIE (Data terminal equipment) oprema terminala

DVMRP Distance Vector Multicast Routing Protocol

EBCDIC Extended Binary-Coded Decimal Interchange Code

EGP Exterior Gateway Protocol

EIA Electronic Industries Association

ELF (Extremely low frequency) izuzetno niska frekvencija

ENIAC Electronic Numerical Integrator and Calculator

ESP Encapsulating Security Payload

FCC Federal Communications Commission

FDM (Frequency-division multiplexing) multipleksiranje sa podelom frekvencije

FHSS Frequency-hopping spread spectrum

FM (Frequency modulation) frekventna modulacija

FSK (Frequency shift keying) frekventna modulacija

FSO Free space optics

FTP (File Transfer Protocol) protokol za transfer fajlova

GIF Graphics Interchange Format

GMII Gigabit medium independent interface interfejs nezavisan od Gigabit medijuma

HDLC High-level Data Link Control

HDSL High-bit-rate DSL

HDTV High-definition television televizija visoke definicije

HTML Hypertext Markup Language

□□□□ Hypertext Transfer Protocol

IANA Internet Assigned Numbers Authority

IBM International Business Machines

ICANN Intemet Corporation for Assigned Names and Numbers

ICMP Internet Control Message Protocol

IEC International Electrotechnical Commission

IEEE Institute of Electrical and Electronic Engineers

EIF Internet Engineering Task Force

IGMP Internet Group Management Protocol

IKE Internet □□□ Exchange

IP Internet Protocol

IPv6 Internet Protocol version 6

ISAKMP Internet Security Association and □□□ Management Protocol

ISDN Integrated Services Digital Network digitalna mreža sa integrisanim servisima

ISO International Organization for Standardization

ITU Intemational Telecommunications Union

JPEG Joint Photographic Experts Group

LAN Local area netvvork lokalna mreža

LAP Link Access Protocol

LED Light-emitting diode

LEO Low earth orbit

LLC Logical link control kontrola logičkog linka

MAC Medium access control kontrola pristupa medijumu

MaGIC Media-accelerated Global Information Carrier

MII (Medium independent interface) interfejs nezavisan od medijuma

MLT-3 Multilevel Line Transmission-Three Levels

□□□ MPEG audio layer 3

MPEG Moving Pictures Expert Group

MTSO Mobile telephone switching office centrala mobilne telefonske mreže

MTU (Maximum transfer unit) maksimalna jedinica prenosa

NAK (Negative acknowledgment) negativna potvrda

NASA National Aeronautics and Space Administration

NBS National Bureau of Standards

NEXT Near-end crosstalk

NIC (Network interface card) mrežna interfejs kartica

NIST National Institute of Standards and Technology

NNI (Network-network interface) interfejs mreža-mreža

NRM (Normal response mode) mod normalnog odziva
NRZ Nonreturn to zero
NRZI Nonretum to zero inverted
NSA (National Security Agency ili No Such Agency Nacionalna) bezbednosna agencija, ili Ne postoji takva agencija
NTSC Nalional Television Standards Committee
OAM (Operations, adrainistration, and maintenance) operacije, adminislriranje i održavanje
OC-n Optical carrier-n
ONU (Optical network) unit jedinica optičke mreže
OSI Open Systems Interconnect
OSPF Open Shortest Path First
PABX (Private automalic branch exchange) privatna automatska centrala
PAM (Pulse amplitude modulation) impulsna amplitudska modulacija
PAM5 (Pulse amplitude modulalion with five levels) impulsna amplitudska modulacija sa pet nivoa
PAN (Personal area netvvork) personalna mreža
PBX (Private branch exchange) privatna centrala
PC (Personal computer) personalni kompjuter
PCM (Pulse code modulation) impulsna kodna moduladja
PCS (Physical coding sublayer) podsloj za fizičko kodiranje
PDU (Protocol data unit) jedinica podataka protokola
Perl Practical Extension and Report Language
PGP Pretty Good Privacy
PIN (Personal identification number) personalni identifikadoni broj
PM (Phase modulation) fazna modulacija
PMD Physical medium dependent
POTS (Plain old telephone service) tradicionalni telefonski servis
PSK (Phase shift keying) fazna modulacija
PVC (Permanent virtual circuit) permanentno virtuelno koio
QAM (Quadrature amplitude modulation) kvadraturna amplitudska modulacija
QoS (Quality of service) kvalitet servisa
QPSK Quaternary phase shift keying
RADSL (Rate-adaptive asymmetric DSL) asimetrični DSL sa adaptacijom brzine
RFC (Request for Comments) zahtev za komentare
RIP Routing Information Protocol
RMON Remote Monitoring
RPB Reverse-path broadcasting
RSA Rivest, Shamir, Adelman
RSVP Resource Reservation Protocol
RTCP RIP Control Protocol
RIP Real-Time Transport Protocol
SAR Segmentation and reassembly sublayer SDH Synchronous Digital Hierarchy
SDLC Synchronous Data Link Control
SDSL Symmelric DSL ili single-wire DSL
SHA-I Secure Hash Algorithm 1
SHDSL Single-pair high-speed DSL
SIFS Short interframe space
SIPP Simple Inlernet Protocol Plus
SMTP Simple Mail Transfer Protocol
SNMP Simple NetWork Management Protocol
SONET (Synchronous Optical Network) sinhrona optička mreža
SPE Synchronous payload envelope
SSCF Service-Specific Coordination Function
SSCOP Service-Specific Connection-Oriented Protocol
ssh Secure Shell
SSL Secure Sockets Layer
SS7 Signaling Syslem 7
STD State transition diagram dijagram prelaza stanja
STP Shielded twisted pair zaštićene upredene parice
STS-n Synchronous transport signal n
SVC Switched virtual circuit komutirano virtuelno kolo
TCP Transmission Control Protocol
TCP/IP Transmission Control Protocol/Internet Protocol
TIA Telecommunications Industry Association
TDM (Time division multiplexing) multipleksiranje sa podelom vremena
TLS Transport Layer Security
TPDU (Transport protocol data unit) jedinice podataka transportnog protokola
UDP User Datagram Protocol
UHF (Llltra-high frequency) ultravisoka frekvencija
UNI (User-network inlerface) interfejs korisnik mreža
URL Uniform Resource Locator
USB (Universal serial bus) univerzalna serijska magistrala
UTP (Unshielded twisted pair) nezaštićene upredene parice
VDSL Very high data rate DSL

VHF (Very high frequency) veoma visoka frekvencija

VLAN (Virtual UN) virtuelni UN

VLSI Very large-scale integration

VSAT Very small aperture lerminal

WAN (Wide area netvvork) mreža šireg geografskog područja

WEP Wired Equivalent Privacy

WDDM (Wave-division deniultiplexer) demulti-plekser sa podelom frekvencije

WDM (Wave-division multiplexing) multiple-ksiranje sa podelom frekvencije

Wi-Fi Wireless fidelity

WIS WAN interface sublayer

WLAN (Wireless UN) bežični UN

WWW World Wide Web

XAUI (XGMJI attachment unil interface) XGMll pridružena jedinica interfejsa

XGMIII (10-Gigabit medium independent interface) interfejs nezavisan od 10 Gigabit medijuma

Indeks

B

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

CIP - □□□□□□□□□□□□j□ □ □□□□□□□□□j□
□□□□□□□ □□□□□□□□□□ □□□□j□,□□□□□□□

004.7

□ □**j,** □□□□**j**□□ □**.**

Savremene komunikacione tehnologije i mreže : stvarni svet / VVilliam A. Shay ; [prevod Dijana Ivanišević]. - 1. izd. - Beograd : Kompjuter biblioteka, 2004 (Čačak : Svetlost). - XXVI, 766 str. : ilustr. ; 24 cm. - (Kompjuter biblioteka ; knj. 292)

Prevod dela: Understanding Data Communications and Networks. - Rečnik: str. 731-747. - Registar.

ISBN86-7310-310-X

a) □□□□□□□□□□ □□□□□
COBISS.SR-ID 117847564

Savremene Komunikacione tehnologije

William A. Shay

Knjiga je napisana je i za studente na nižim godinama studija na odseku za kompjutersku tehniku koji su odslušali najmanje dva semestra o izradi softvera i imaju dobru osnovu iz matematike, uključujući i diskretnu matematiku. Obuhvaćene su standardne teme na uvodnom kursu o komunikacijama i kompjuterskim mrežama, kao što su mediji za prenos, analogni i digitalni signali, prenos podataka, metodi za kompresovanje i šifrovanje, mrežne topologije, zaštita mreža, LAN protokoli, Internet protokoli i aplikacije, tehnologije sa komutacijom kola i Web aplikacije.

Cilj nam je da pomognemo čitaocu da razume:
Razlike, prednosti i nedostatke različitih medija za prenos analogne i digitalne signale, tehnike modulacije i demodulacije i način funkcionisanja uređaja za modulaciju, kao što su modemi, kablovski modemi i DSL modemi
Efekat šuma u toku prenosa i kako protokoli detektuju da su informacije promenjene
Kako protokoli reaguju u situacijama kada šum izaziva oštećenja, ili gubitak informacija
Standarde kao što su AES, ATM, DES, EIA–232, HDLC, IEEE 802.3, IEEE 802.5, IEEE 802.11, IPv6, JPEG, MP3, MPEG, OSI, SONET, TCP/IP i X.25, organizacije za uspostavljanje standarda i razloge zašto su standardi neophodni
Tehnike za kompresovanje podataka, tipove podataka koji se mogu kompresovati i poređenje različitih metoda koji se danas koriste
"Crve", viruse i ostale pretnje umreženim kompjuterima
Potrebu za zaštitom i efektivnim metodima šifrovanja
Razlike između sistema šifrovanja javnim i privatnim ključem
Kako se uspostavljaju bezbedne konekcije između udaljenih sajtova
Potrebu za kontrolom toka i različite načine za njenu implementaciju
Protokole koji se koriste na lokalnim mrežama i strategije nadmetanja na deljenim medijima za prenos podataka
Bežične standarde
Metode povezivanja lokalnih mreža
Strategije rutiranja
Potrebu za protokolima koji podržavaju real–time video aplikacije i koji zadovoljavaju zahteve za kvalitetnim servisima
Kako se dizajniraju i postavljaju različite funkcionalne klijent/server aplikacije
Kako sve veće korišćenje Weba i multimedijalnih aplikacija utiče na postojeće protokole i šta je učinjeno da bi se prevazišle eventualne poteškoće.

Korisnički nivo Početni Srednji Napredni

EDICIJA STVARNI SVET

PREPORUČUJEMO

ISBN: 86-7310-131-x

CISCO tehnologije

ISBN: 86-7310-159-x

Administracija TCP/IP mreže

ISBN: 86-7310-253-7

Hakerski vodič za zaštitu

ISBN: 86-7310-310-x