4 – Komponente klijent server tehnologije - klijent

S A D R Ž A J

4.1 Karakteristike klijentskog računara

- Bilo koju radnu stanicu (*desktop workstation*) koju koristi jedan korisnik smatramo **klijentskom radnom stanicom** ili samo klijentom.
- Ukoliko jednoj istoj stanici pristupaju istovremeno više korisnika simultano, tu istu stanicu nazivamo **serverom**.
- U klijent server modelu ne postoje neke tehnološke razlike, sa stanovništva računara, između klijenta i servera.
- Zadnjih 35 god. performanse radnih stanica su dramatično napredovale
  - ✓ performanse CPU povećale su se **850** puta (4,7MHz → 4GHz),
  - ✓ veličina RAM-a memorije **16 000** puta (256kB → 4GB)
  - ✓ kapacitet hard diskova **100 000** puta (10MB → 1TB).
- Napredak tehnologije učinio je da aplikacije koje su se izvršavale na desktop računarima postanu jako sofisticirane (pametne) tj. napredne
- Sa druge strane ovakav veliki razvoj tehnologije desktop računara prouzrokovao je i veliki razvoj drugih računarskih tehnologija a pre svega komunikacionih (razvoj velikog broja protokola) a one su dovele do razvoja mrežnih tehnologija (LAN, WAN i Internet mreže).

Danas je nezamislivo da imamo desktop računar koji nije umrežen

4.1 Karakteristike klijentskog računara

- Klijent je bilo koji proces koji zahteva usluge od serverskog procesa.
- Poželjne osobine hardverske i softverske komponente klijent računara:
  - ✓ Ne toliko snažan hardver
  - ✓ Operativni sistem koji je sposoban da podrži multitasking
  - ✓ Grafički korisnički interfejs (**GUI** – *Graphic User Interface*)
  - ✓ Komunikacione sposobnosti (mrežnu karticu)
- Hardver klijenta je obično PC računar a u poslednje vreme se koristi i X-terminal koji jedan deo potrebnog klijent/server softvera drži u ROM-u, a ostatak se puni u RAM memoriju sa servera preko mreže.
- Postoji široki opseg različitih OS koji mogu da zadovolje minimalne zahteve klijentskih OS (DOS/Windows,WinXP,Win7,Win10,Linux,...)
- Hardver i OS mora obezbediti adekvatno povezivanje sa mrežnim OS.
- Klijent aplikacije su uglavnom zasnovane na **grafičkom korisničkom interfejsu**, čija je uloga da sakrije kompleksnost od krajnjeg korisnika
- Klijent aplikacija interaguje sa OS radi korišćenja multitaskinga, GUI interfejsa i sa mrežnom komponentom komunikacionog posrednika
- Dok se zahtev izvršava na serveru, klijent je potpuno slobodan

4.2 Uloga klijenta

- Klijent je osnovni korisnik servisa koji se izvršavaju na serverima
- Postoji jasna podela funkcija između klijenta i servera
- Jedan od osnovnih servisa koji se izvršavaju na klijent radnoj stanici je ***prezentacioni servis*** - kontroliše prihvatanje i prikazivanje podataka.
- Današnje tehnologije omogućile su pored GUI interfejsa i multimedijalni pristup podacima putem audio-vizuelnog pristupa.
- Tehnika **višestrukih prozora** (***windowing enviroment***) omogućuje klijentu da istovremeno bude uključen u nekoliko simultanih sesija.
- Unošenje teksta, praćenje E-maila, gledanje nekog filma ili slušanje muzike, mogu potpuno simultano da se odvijaju na jednom klijentu
- Klijentski softver podržava napredne mogućnosti kao što su:
  - ✓ **DDE** (*Dynamic Data Echange*) dinamička razmena podataka,
  - ✓ **OLE** (*Object Level Embedding*)
  - ✓ **CORBA** (*Communicating Object Request Broker Architecture*)
- One omogućuju prostu operaciju ***cut and paste*** (premeštanje podataka) između različitih aplikacija: tekst procesora, baza podataka, tabelarnih prikaza, grafičkih prikaza i to u različitim višestrukim prozorima.

4.3 Klijentski servisi

- Idealna klijent server platforma operiše u otvorenom sistemskom okruženju i koristi serverske servise na bazi postavljenih zahteva
- Zahvaljujući ovim osobinama omogućeno je da veliki broj potpuno različitih hardverskih/softverskih rešenja međusobno dobro sarađuju
- Serveri mogu da se dograđuju i da menjaju svoj OS ili hardversku platformu a da se sa gledišta klijenta to uopšte i ne oseti
- Promene se odvijaju po unapred definisanim i usvojenim standardima
- Klijenti mogu da menjaju takođe svoj OS, aplikaciju ili hardversku platformu a da to server uopšte i ne primeti.
- Mogućnost prikazivanja multimedijalnih i tekstualnih podataka
- **Višestruke prednosti** (primer sistema zaštite prostora): automatski se uključe kamere i prikazuje se slika - jednostavna i skraćena obuka
- Primena GUI može da poveća produktivnost i efikasnost i do 200 %
- Funkcionalnost klijentskog procesa može se povećati **dodavanjem logike** koja nije implementirana u serversku aplikaciju: lokalno sređivanje teksta, davanje HELP informacija, automatski ulaz i kontrola podataka, padajući meniji, grafički prikazi, automatsko izvršavanje.

4.3 Klijentski servisi

- Klijent radna stanica zahteva neki servis od servera a **NOS** (***N**etware **O**perating **S**ystem*) softver prenosi ili dodaje specificirani zahtev od izvora do ciljnog servera na kome se izvršava neka aplikacija.
- Uvek kada je taj server isti kao i radna stanica ili pak predstavlja neki mrežni, fizički odvojeni server, aplikacioni format zahteva je uvek isti.
- **IPC** (***I**nter **P**rocess **C**ommunication*) predstavlja generički izraz koji se koristi da opiše komunikaciju između dva ili više procesa.
- U klijent server modelu ti procesi mogu biti na istom računaru, na računarima koju su povezani u LAN, WAN ili Internet mreži.
- Najćešći servis koji **NOS** pokreće je **servis redirekcije**.
- Ovaj servis prihvata pozive od klijentskog OS i prenosi ih prema NOS
- Dugi niz godina programeri su se trudili da razvijaju modularni kod koristeći strukturnu tehniku i logiku poziva podprograma.
- Danas se zahteva da ti podprogrami (*subroutines*) budu i zapamćeni negde kao objekti koji će sada biti dostupni svima ko želi da ih koristi

4.3 Klijentski servisi

- Tehnika poznata kao RPC (***Remote Procedure Call***) upravo omogućava ovakav vid programiranja.

RPC zbog toga predstavlja srce klijent server modela i u suštini predstavlja program koji može da pozove bilo koju proceduru sa umrežnih računara sa ciljem da zadovolji zahteve klijenta.

- Pri tome ne mora da se zna ni fizičko mesto gde se nalazi tražena procedura ili servis ili gde se izvršava.
- Klijent upućuje svoj zahtev i dobija odgovor preko samo jedne ***send*** primitive, što znači da RPC troši samo **jednu operaciju jezgra**.
- Pojednostavljeno je baferisanje zahteva i odgovora, jer isti bafer može da se koristi za pamćenje upućenog zahteva i primljenog odgovora.
- RPC omogućava da se poziv i izvršenje zahteva može odvijati na potpuno različitim OS i potpuno različitim hardverskim platformama.
- Mnogi RPC omogućavaju i servis translacije.
- On podrazumeva da se vrši prevođenje (translacija) podataka između procesora sa različitim fizičkim formatima podataka.
- Ti standardi se stalno usavršavaju i prilagođavaju zahtevima tržišta.

4.3 Uloga klijentskih servisa

1. Odgovoran za upravljanje korisničkim interfejsom.
2. Obezbeđuje prezentaciju usluga.
3. Prihvata i proverava sintaksu korisničkih ulaza i prikazuje bilo koje korisničke ulazne ili izlazne podatke na klijentskoj radnoj stanici.
4. Ponaša se kao potrošač usluga koje dobija sa jednog ili više servera.
5. Prilagođava se i podržava logiku aplikacije.
6. Generiše zahtev bazi podataka i prenosi ga prema serveru.
7. Prenosi odgovor nazad do servera.
8. Uloga klijentskog procesa može se proširiti dodavanjem neke logike koje se ne izvršava na serverskom procesu kao što su:
   a) lokalno uređivanje podataka,
   b) automatski unos i kontrola ulaznih podataka,
   c) dodavanje help-a,
   d) bilo koji drugi logički procesi koji se nisu izvršili na serveru.

4.3 Klijentski servisi

- ***Fax/Print servis*** - ogućuje klijentu da generiše zahtev za štampanjem bez obzira što je tog trenutka štampač zauzet. Zahtev se prenosi putem NOS redirektora i smešta u print server red kojim upravlja print server manager. Fax servis se izvršava na potpuno isti način kao i print zahtev.
- ***E-mail servis*** - funkcioniše na isti način kao i gore opisani servis.
- ***Window servis*** – omogućuje da se na klijentskom računaru mogu istovremeno otvoriti više prozora u okviru kojih se izvršavaju različite aplikacije. Mogućnost da se aktivira, vidi, pomera, smanji ili poveća, sakrije određeni prozor omogućena je baš zahvaljujići ovom servisu.
- Ovaj servis je jedan od osnovnih servisa u klijent server modelu jer je on u neposredom kontaktu sa servisom poruka (*pop-up* prozori) koji je zadužen da obavesti korisnika o događajima koji se dešavaju na serveru
- ***Remote Boot Service*** - Neke aplikacije mogu da se izvršavaju na klijentskim radnim stanicama bez bilo kakvog lokalnog diska sa koga bi se učitao OS ili aplikacija (primer X-terminala). Da bi to moglo da funkcioniše klijent mora da obezbedi odgovarajući softver koji se nalazi u EPROM-u koji treba da startuje **IPL** (***I**nitial **P**rogram **L**oad*)

4.3 Klijentski servisi

- ***Backup Service*** - servis koji se može pozvati sa klijentske strane a koji se izvršava na serveru kako bi se uradila zaštita fajlova tkz. ***backup***.
- ***Utility Service*** - omogućava lokalne funkcije kao što su: kopiranje, premeštanje, editovanje, upoređivanje i funkciju helpa.
- ***Message Service*** - omogućava baferovanje (skladištenje), pozadinsko prikazivanje (*scheduling*) i servis izbora poruka (*arbitration service*) kako bi poruke mogle da se primaju i šalju potpuno sinhronizovano
- ***Network Service*** - klijent komunicira sa mrežom putem više servisa i API-a koji kreiraju, šalju, primaju i formatiraju mrežne poruke. Da bi to sve mogao da uradi neophodna je podrška za komunikacione protokole kao što su: NetBIOS, IPX, TCP/IP, APPC, Ethernet, Token Ring, X.25.
- ***Application Service*** - omogućuje da mnoge aplikacije imaju sopstveni RPC kojim se koriste da bi mogle da pozovu neke servise sa servera.
- ***Database Service*** – omogućuje da se zahtevi za podacima iz baze podataka zasnivaju na SQL sintaksi.
- ***Netware Mangment Service-Alerts*** – prikazivanje upozoravajućih poruka (*alert*) koje većina mrežnih interfejs kartica može da generiše.

4.4 Programski zahtevi za klijente

- Zbog velikog razvoja novih tehnologija i OS, pred programerima su se postavljali sve veći zahtevi kod pisanja aplikacija
- To su bili krupni zahtevi koji su znatno usporavali razvoj aplikacija
- Jednom aplikacijom nije bilo moguće da se objedine sve te tehnologije kao i da se vodi računa o upravljanju svim deljivim resursima servera (memorija, CPU vreme), upravljanje kontekstom ili nitima.
- To bi zahtevalo da se svaka transakcija koja se dešava u sistemu posebno programira a razvoj takvih aplikacija bio bi jako težak i skup.
- Rešenje je nađeno u razvoju ***komponentnih modela***
- Komponente u kojima je implementiran mehanizam izvršavanja transakcija u mnogome oslobađaju programere od ovog teškog zadatka i predstavljaju jedan od najefikasnijih načina da se brzo i kvalitetno naprave aplikacije koje će u potpunosti zadovoljiti zahteve tržišta.
- Komponentni model razvijanja aplikacije omogučava lakše izmene a samim tim i jednostavno odžavanje aplikacije.
- Vreme potrebno za razvijanje, pisanje i implementaciju aplikacije se drastično smanjuje jer postoji veliki broj već razvijenih komponenti

4.4 Programski zahtevi za kljinete

- Ovakva impementacija transakcija se naziva deklerativnim pristupom.
- Najpopularniji komponentni modeli kod razvijanja aplikacije danas su:

Microsof: **COM** (*Component Object Model*),
DCOM, COM+, ActveX

Sun Microsystem: **JB** (*Java Beans*)
EJB (*Enterprise Java Beans*)

Object Managment Group (OMG):
CORBA (*Common Object Request Broker Architecture*)

- Za razvoj komponentnih modela veliku ulogu igraju i nekoliko pojmova ili tehnika koje su doprinele razvoju ovakvih komponentnih modula tj. bili su neophodni preduslovi koji su doprineli da dođe do ovakvih rešenja na nivou globalne distributivne mreže računara:
  - ✓ ***Inter process communication (IPC)***
  - ✓ ***Dynamic Data Excange (DDE)***
  - ✓ ***Object Linking and Embedding (OLE)***
  - ✓ ***Distributed Computing Environment*** (***OSF DCE***)

4.4 Inter process communication (IPC)

- Komunikacija između dva procesa obično se odvija preko bafera
- Jednostavan mehanizam koji sinhronizuje aktivnost procesa bez potrebe da se deli isti zajednički adresni prostor naziva se IPC.
- Ovakav način komunikacije predstavlja osnovu za komuniciranje procesa u jednom složenom distribuiranom sistemu.

1. **Razmena poruka** (*message passing*) - ava procesima da komuniciraju bez obnavljanja zajedničkih, deljenih podataka. Postoje najmanje dva procesa koja su uključena u ovakav vid komunikacije i to: **process slanja** za slanje poruke i **process prijema** koji prima poruku. Poruka koja se šalje može biti fiksne ili promenljive veličine.
2. **Direktna komunikacija** – procesi koji komuniciraju moraju navesti ime procesa koji šalju i ime procesa koji primaju poruke:
   a) veza između pošiljaoca i primaoca je uspostavljena uz potuno poznavanje podataka o njihovim imenima i adresama
   b) između procesa pošiljaoca i primaoca potrebno je uspostaviti bar jednu vezu.
   c) postoji simetrija u komunikaciji između procesa.

4.4 Inter process communication (IPC)

3. Indirektna komunikacija – poruke se smeštaju u poštansko sanduče (*mail box*) i iz njega se one kasnije čitaju.

Indirektna komunikacija može takođe da komunicira sa drugim procesima preko jednog ili više poštanskih sandučeta:

1. veza je uspostavljena između para procesa, ukoliko dele *mail box*
2. veza je uspostavljena između više od jednog procesa.
3. između dva procesa može se ostvariti različiti broj linkova

Komunikacija između procesa odvija se izvršavanjem naredbi za slanje i prijem poruka koje mogu biti blokirajuće i neblokirajuće i to:

Blocking send,*Non-blocking send*,*Blocking receive*, *Non-blocking receive*

4. Poziv udaljene procesure (*Remote Procedure Call*) - RPC je na tehnika za izgradnju distribuirane, klijent-server aplikacije. Suština ove tehnologije je da i programima na različitim računarima da mogu da komuniciraju korišćenjem jednostavne procedure poziva ili prijema. Ona se zasniva na proširenom standardnom ili lokalnom postupku pozivanja procedure, tako da procedura koja poziva ne mora da bude u istom adresnom prostoru kao procedura koja se poziva.

4.4 Dynamic Data Exchange (DDE)

- ***DDE*** predstavlja funkciju koja omogućava klijentu da razmenjuje podatke između različitih aplikacija zahvaljujiči jedinstvenom API-u.
- Primer:grafički prikaz nekog podatka može biti povezan sa podacima u bazi podataka. Kako se oni menjaju tako se menja i grafički prikaz.
- DDE se obično opisuje kao razgovor između dve aplikacije
- DDE je karakteristika OS koja omogućuje korisnicima da direktno menjaju podatke između različitih aplikacija na različitim računarima
- DDE sprega se uvek odnosi na određeni dokument i određenu stavku
- Kod ine programa, najjednostavniji način da se uspostavi DDE link je da se kopira blok podataka iz server aplikacije u Clipboard (*Copy*).
- Nakon toga se startuje klijentska aplikacija i direktno se iz Clipboard-a iskopiraju podaci (*Paste*) na željeno mesto u klijentskom izveštaju.
- Neki programi koji deluju kao DDE klijenti imaju komande koje avaju da se DDE veza podesi bez prethodnog *Copy and Paste*
- Postoje i DDE aplikacije koje imaju makro jezik koji možete da koristite za uspostavljanje veze za DDE kao što su MS Exel ili Word.
- DDE veza može biti automatska ili manuelna.

4.4 Object Linking and Embedding (OLE)

- ***OLE*** predstavlja jedno proširenje DDE koje je omogučilo da sa objekti kreiraju sa objektnim softverskim komonentama.
- Postoji referenca između objekata i odgovarajučeg softvera kojim su te komponente formirane tako da oni rade kao jedan servis.
- OLE je softverski paket koji pristupa podacima koje je kreirala druga aplikacija korišćenjem pregledača (***Viewer***) ili izvršavača (***Launcher***).
- Za svaku pojedinu aplikaciju mora da se uspostavi *Viewer* i *Launcher*
- Klijentu je omogućena je puna funkcionalnost pozvane aplikacije
- Za uspostavu OLE veze potrebno je da oba programa podržavaju OLE standard i to **OLE klijent** i **OLE server** standard
- Postupak uspostavljanja veze sličan je kao i kod DDE
- OLE je 1992 god. razvio Microsoft koji je kasnije 1995 god. prerastao u **COM** (*Component Object Model*). Od 1996 godine COM počinje sa podrškom distribuiranog procesiranja i dobija ime **DCOM**
- Microsoft je objavio i svoj OLE 2.0 Softver Development Kit (SDK)

OLE je poznat kao proširenje tehnike DDE i ava objektima da automatski pozivaju i softver za ažuriranje tih podataka.

4.4 Distributed Computing Enviroment (OSF DCE)

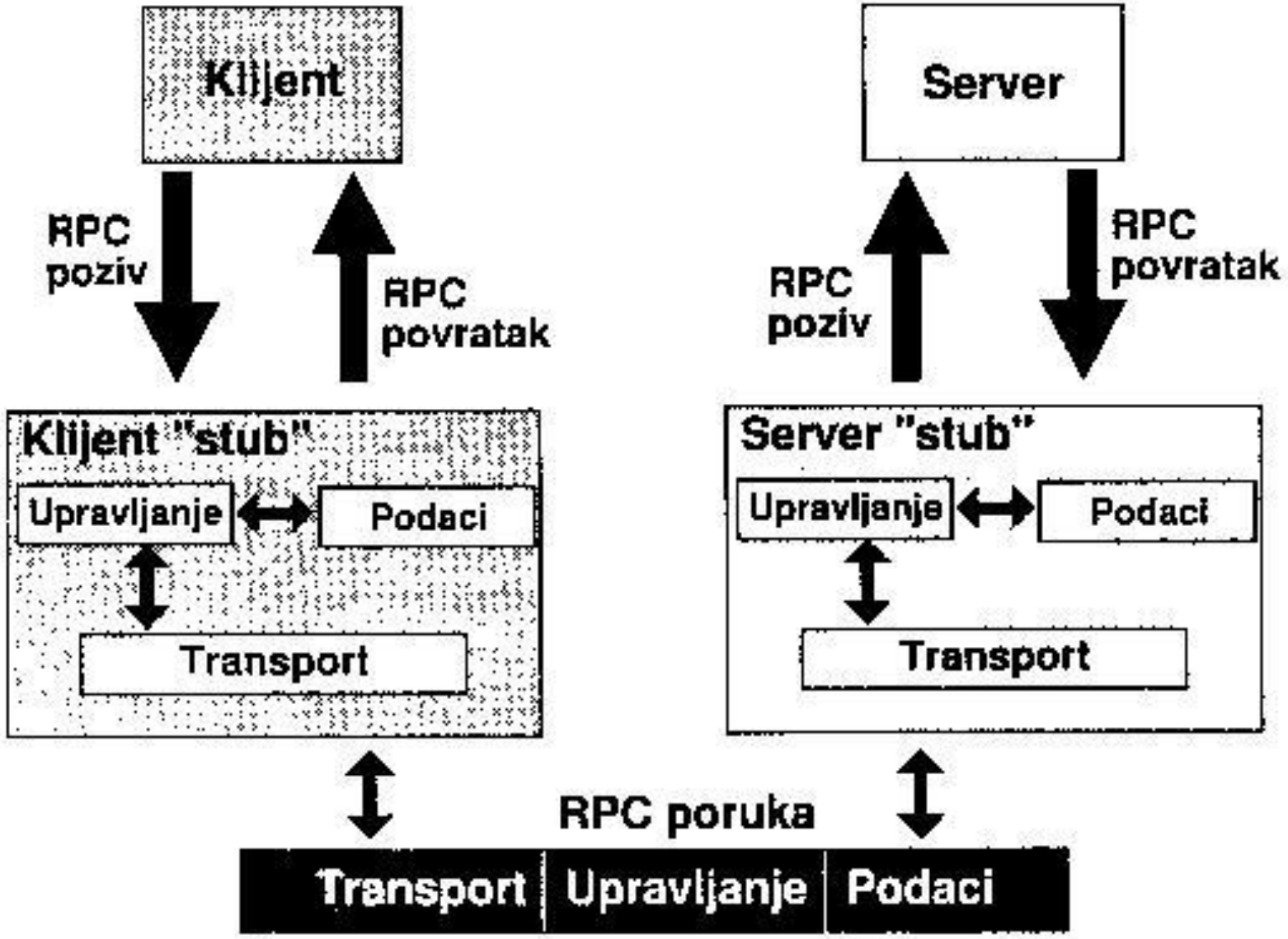
- Predstavlja komunikacioni mehanizam koji nam dozvoljava kako rad tako i razvoj distribuiran.aplikacija baziranih na klijent server modelu
- Filozofija klijent server sistema podrazumeva veoma visok nivo komunikativnosti između računara bez obzira na njihovu veličinu, hardversku platformu, vrste OS ili softversk.aplikacije koja se izvršava
- Bez obzira na različite ciljeve i okruženja u kojima se izvršavaju te aplikacije sve one imaju zajedničke osnove koje moraju da zadovolje:

RPC (*Remote Procedure Call*) je osnovna premisa distribuiranih heterogenih komunikacija.

- RPC se može posmatrati kao protokol na nivou prezentacije u ISO OSI
- Tako posmatrani RPC može se opisati preko pet osnovnih komponenti:

1. ***Stub*-ovi**
2. **Protokol za uspostavljanje veze (*binding protocol*)**
3. **Reprezentacija podataka**
4. **Transportni protokol**
5. **Upravljački protokol**

4.5 Način funkcionisanja RPC



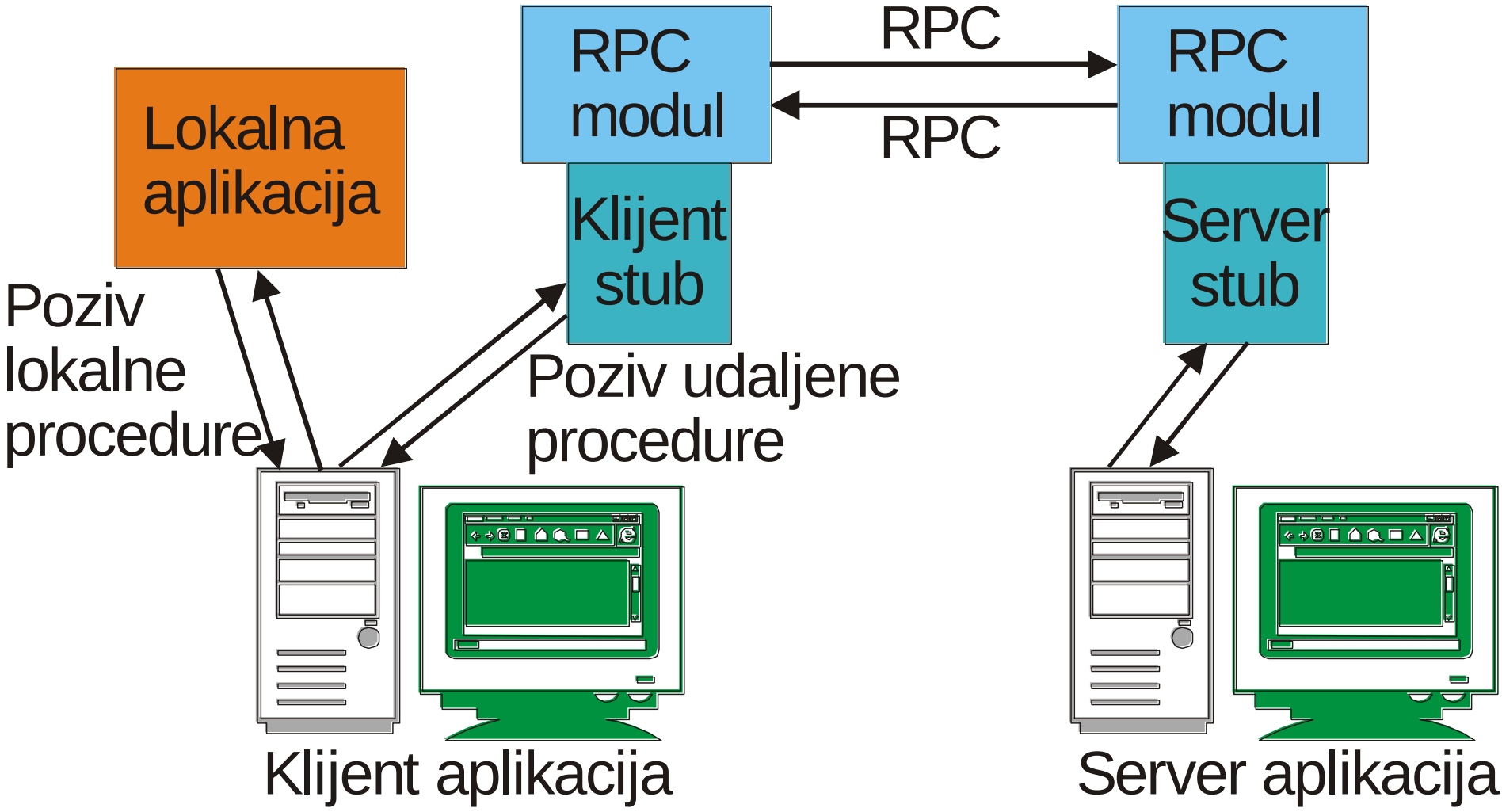
4.5 Način funkcionisanja RPC

- Razvoj i upotreba distribuirane aplikacije prolazi kroz tri faze:
  I. vreme potrebno za kompilaciju,
  II. vreme uspostavljanja veze,
  III. vreme poziva.
- Podrazumeva se da se programiranje klijent server modula vrši na način kao da će oba modula biti linkovana tj. povezana zajedno.
- Server implementira poseban interfejs napisan u nekom opisnom jeziku (**IDL - *I**nterface* *D**escription* *L**anguage***) čiji se opis onda procesira proizvodeći dva ***stub***-a: klijentov i serverov ***stub***.
- Klijentov *stub* se linkuje sa klijentom koji ga sada vidi kao server i obrnuto server posmatra svoj *stub* kao klijent.
- *Stub* odvaja klijenta/srevera od detalja uspostavljanja veze i transporta
- RPC mehanizam takođe ovde upravlja sinhronizacijom, kodiranjem podataka, *timeout*-ima kod uspostave veze i zaštitnim informacijama
- Administracija imena resursa je od izuzetne važnosti jer distribuirani sistemi upotrebljavaju imena da opišu različite resurse kao što su računari, fajlovi, adresari, programi, korisnici i td.

4.5 Način funkcionisanja RPC

- Sa obzirom na potencijalno veliki broj resursa, jasno je da kreiranje i upravljenje resursima mora biti maksimalno pojednostavljeno i da:
  - ✓ rešenje mora biti nezavisno od broja računara u mreži,
  - ✓ rešenje mora da obezbedi zaštitu integriteta podataka.
- Jasno je da u jednom distribuiranom računarskom sistemu zaštita podataka mnogo kopleksija nego kod samostalnih računarskih sistema.
- Posebni serveri zaštite, kriptovanje podataka, sistem autentičnosti, lozinke, specijalni protokoli i mnogi drugi mehanizmi su razvijeni.
- Zahvaljujući OSF DCE komunikacionom mehanizmu obezbeđen je jedan potpuno novi ambijent za razvoj aplikacija u distribuiranim mrežnim okruženjima.
- Složeni programi mogu se razvijati na više lokacija širom sveta
- Koristeći RPC, klijent može pozvati proceduru na udaljenom sistemu na način kao da je to lokalni poziv procedure.
- RPC sa sobom nosi potrebne argumente i vraća rezultat.
- Problem povezivanja klijenta i servera se odvija na posebnom serveru na kome se izvršava proces povezivanja koji traje svega nekoliko ms.

4.5 Način funkcionisanja RPC



4.5 Ograničenja RPC tehnike

1. RPC zahteva uspostavljanje sinhrone veze.
   - ✓ Ako aplikacija koristi RPC i želi da se poveže sa serverom koji je tog trenutka zauzet, tada će aplikacija morati da sačeka a ne da pređe na izvršavanje drugog zadatka.

2. Nije moguće uvek uspostaviti poziv lokalne procedure (RPC)
   - ✓ U uslovima kada su loše komunikacije veze potrebno je više puta da se kontaktira server a nekada to i nije moguće

3. Potrebno je prvo da se klijent i server povežu
   - ✓ Komunikacija između klijenta i servera se vrši uz pomoc standardne procedure poziva
   - ✓ Zato je neophodno prethodno uspostaviti vezu klijent - server tj. da se procesi povezivanja zamene sa adresama klijenta i servera
   - ✓ Opšti problem koji postoji je da ne postoji zajednička deljena memorija između njih, pa je problem kako da server i klijent saznaju adrese da bi se povezali.

4.5 Ograničenja RPC tehnike

4. Informacija vezivanja mora biti unapred određena

- ✓ Potrebno je prethodno znati port adresu aplikacije koja se poziva
- ✓ To se radi za vreme kompajliranja RPC poziva koji ima fiksni broj porta.
- ✓ Javlja se problem jer ne može da se promeni taj broj porta.

5. Dinamičko povezivanje klijenta i servera.

- ✓ Da bi se omogućilo dinamičko povezivanje klijenta i srvera koristi se mehanizam – ***rendezvous*** **(sastanak)**
- ✓ Obično, OS ava *rendezvous*, putem *daemon* programa, koji traži adresu porta za RPC.
- ✓ Adresa porta se tada vraća i RPC poziv se može poslati na dobijenu adresu sve dok se proces ne završi.

Hvala na pažnji !!!

Pitanja

? ? ?