XI Aplikativni sloj

1. **Karakteristike Aplikacionog sloja**
2. **Protokoli aplikacionog sloja**
3. **Kodiranje podataka - kompresija**
4. **Obrada upita**
5. **Karakteristike operativnih sistema u BSM**
    - **5.1 Tiny OS**
    - **5.2 Contiki**
    - **5.3 Mantis**
    - **5.4 SOS**

11.1 Karakteristike aplikacionog sloja

- Ovaj nivo je jedan od najmanje istraženih delova u BSM
- Velika primena BSM u različitim aplikacijama daje mogućnost da se razvijaju mnogi upravljački protokoli koji će se prilagođavati datoj aplikaciji i tako optimizovati parametre mreže
- Osnovna uloga ovog sloja je da približi softver i hardver sa nižih nivoa mrežnoj aplikaciji
- Vrši apstrakciju fizičke topologije BSM za aplikacije koje rade u njoj.
- Sloj aplikacije svojim interfejsima omogućuje korisniku da uspostavi direktnu interakciju sa fizičkim svetom putem BSM.
- BSM se primenjuju u okruženjima gde SČ mogu biti izloženi okolnostima koje znatno utiću na uspešnu realizaciju ovog sloja
- Takve okolnosti uključuju česte promene pritiska, temperature, zračenja i elektromagnetnog šuma.
- Sva merenja koje obave SČ mogu biti neprecizna pa čak i pogrešna.
- Da bi omogućio da podaci koji se prikupljaju sa SČ budu verodostojni i da pravovremeno stignu do ***sink***-a, ovaj sloj mora da da odgovore na mnoga pitanja koja se odnose na definisanje protokola

11.1 Karakteristike aplikacionog sloja

- **Data Aggregation/Fusion**: sažimanje velike količine sličnih informacija u jednom SČ i prosleđivanje drugim SČ
- **Sinhronizacija vremena:** mnoge aplikacije zahtevaju da SČ-ovi imaju jedinstveno vreme u celoj BSM
- **Određivanje lokacije/pozicije SČ-ova:** tačna lokacija SČ utiće da donošenje mnogih odluka u BSM
- **Kompresija podataka**: tehnika od posebnog značaja za prenos multimedije jer smanjuje ukupnu količinu podataka za prenos
- **Kontrola potrošnje**: zasebni moduli se bave potrošnjom u svakom SČ kao i informacijom koliko je energije preostalo u njemu. Ako se zaključi da nije ostalo dovoljno energije, aplikacija prelazi u mod izvođenja samo vitalnih zahteva radi produženja životnog veka tog SČ a time i cele BSM.
- ***Load Partitioning***: mogućnost da se energetski zahtevne operacije prebace SČ-ima koji nemaju ograničene energetske potencijale.

11.1 Fuzija podataka

- Fuzija podataka se koristi za prevazilaženje grešaka SČ, tehnoloških ograničenja i problema prostornog i vremenskog praćenja događaja.
- Fuzija podataka je uopšteno definisana kao upotreba različitih tehnika koje kombinuju podatke iz više izvora (SČ) i prikupljaju informacije
- To dovodi do efikasnijeg i potencijalno tačnijeg podatka o događaju nego ako bi se taj podatak dobio samo od jednog SČ.
- Termin **efikasno**, u ovom slučaju, može značiti pouzdaniji prenos odgovarajućeg podatka koji je kompletniji i sa **većom tačnošću** odgovara podacima o nadgledanom događaju.
- Fuzija se implementira u **centralizovanim i distribuiranim** sistemima
- U centralizovanom sistemu svi prikupljeni podaci o događaju se šalju prema jednom SČ, gde se izvršava fuzija podataka.
- Kod distribuiranih sistema, različiti moduli fuzije bi se implementirali na distribuiranim komponentama tj. pojedinačnim SČ.
- Tehnike obrade podataka podrazumevaju tehnike od jednostavnog izveštavanja do komplikovanijih složenih komunikacija: agregacija podataka, ***broadcast***, ***multicast*** i plavljenje (***gossiping***).

11.1 Sinhronizacija vremena

- Očitani podaci imaju ograničenu upotrebu, ako nisu praćeni koordinatama položaja SČ i vremenskim oznakama (***timestamp***).
- Ovo predstavlja primarni motiv za sinhronizaciju vremena u BSM.
- Prosto je nemoguće uraditi ispravnu fuziju podataka bez tačne vremenske sinhronizacije SČ-ova.

Primer: *SČ-ovi koji su opremljeni akustičnim senzorima, mogu tokom čitavog puta otkriti kretanje nekog objekta u različitim vrem.trenucima*

- Fuzioni čvor koji prima sirove informacije iz SČ može na osnovu toga da izvrši procenu brzine i smera kretanja praćenog objekta.
- Za ovu aplikaciju, između ostalog, neophodni su precizno vremenski sinhronizovani SČ-ovi kako bi dobijeni podaci bili validni.
- Takođe, od SČ se očekuje da budu gabaritno mali i jevtini kako bi se u što većem broju rasporedili u nadgledanom regionu.
- Kada se SČ rasporede u regionu njihov raspored se retko menja
- Neophodno je primeniti tehnike da bi se produžio njihov životni vek.
- Većina ovih tehnika svoj rad bazira na sinhronizovanom vremenu svih SČ-ova u mreži – ***duty cycle***, ***sleeping*** protokoli.

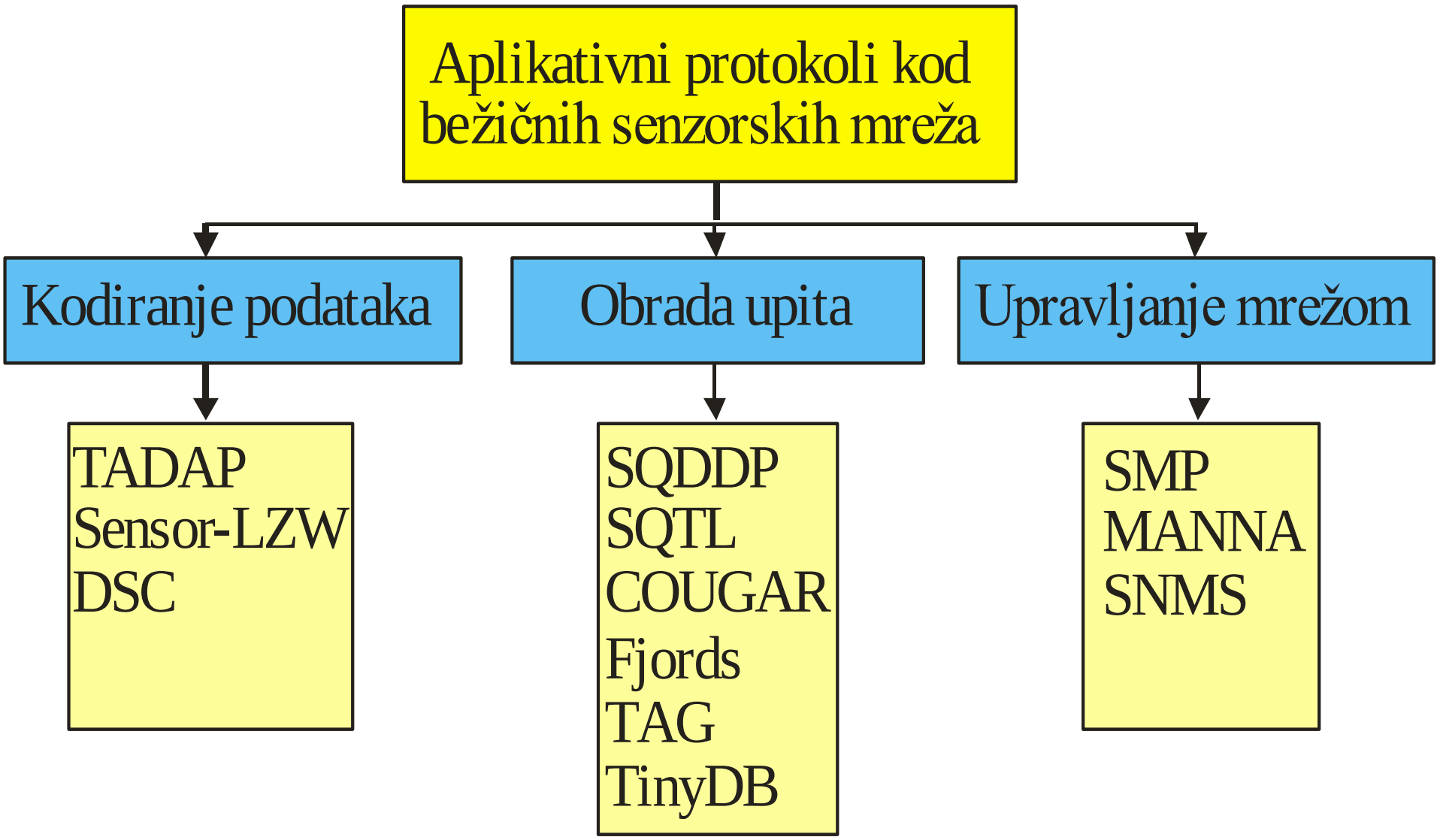
11.1 Lokalizacija senzorskih čvorova

- Pozicioniranje, tj. poznavanje položaja SČ-ova u BSM je suštinski deo mnogih BSM operacija i aplikacija.
- Mnoge BSM aplikacije zahtevaju da se zna tačna lokacija u prostoru sa koga SČ izveštavaju o pojavama tj. prikupljaju podatke.
- Neophodno je obezbediti da SČ moraju biti svesni svog položaja kako bi mogli da daju informaciju o mestu gde se nalaze.
- Mnogi mrežni protokoli kao što je rutiranje zahtevaju informacije o lokaciji kako bi se obezbedila adekvatna usluga protokola.
- Podaci o lokacijama SČ-ova su jako bitne informacije koje mogu da utiču na donošenje mnogih odluka u BSM.
- SČ su često raspoređeni u neprijateljskim okruženjima, gde zlonamerni protivnici pokušavaju da pošalju netačne podatke o njihovoj lokaciji.

Primer: *napadač može izmeniti procenu udaljenosti SČ na nekoliko referentnih tačaka ili poslati beacon iz jednog dela mreže na neki udaljeni deo mreže i na taj način pružiti lažne informacije*

- Zato, postoji potreba da se obezbedi da se procena lokacije vrši na jedan robustan način koji će svaki napad ovakve vrste onemogućiti.

11.2 Protokoli na aplikacionom nivou



11.2 Protokoli na aplikacionom nivou

1. ***Sensor Management Protocol*** (SMP) – predstavlja **upravljački protokol** koji omogućuje softverske operacije potrebne za izvršavanje mnogih administratorskih zadataka u BSM koji upravljaju SČ-ima.

✓ Za razliku od drugih mreža, BSM se sastoje od SČ koji nemaju globalne ID-ove i obično su bez neke infrastrukture.

✓ SMP treba da pristupi SČ-ima pre svega koristeći imenovanje zasnovano na **atributima** i adresiranje zasnovano na **lokaciji**

✓ SMP je upravljački protokol koji obezbeđuje:

- pravila koja se odnose na prikupljanje podataka
- vrši vremensku sinhronizaciju čvorova
- zadužen za razmenu podataka vezanih za pronalaženje lokacija.
- prati kretanje senzorskih čvorova
- kontroluše uključivanje i isključivanje SČ
- čuva mrežnu konfiguraciju i status SČ
- proverava mrežnu konfiguraciju SČ i njihov status i rekonfiguriše je
- provera ispravnosti komunikacije dva čvora
- vrši autentifikaciju,distribuciju ključa i zadužen je za sigurnost mreže

11.2 Protokoli na aplikacionom nivou

2. ***Task Assignment and Data Advertisement Protocol*** (TADAP)

- ✓ Jedna od važnijih operacija u BSM je širenje zahteva za određenim podatkom (***interest dessimination***).
- ✓ Korisnik ima potrebe da pošalje zahtev za nekim podatkom određenom senzorskom čvoru , određenoj grupi ili celoj mreži.
- ✓ Sa druge strane u nekim protokolima sam SČ ima potrebe da objavi sa kojim podacima on raspolaže (***advertisement of available data***).
- ✓ Upravo ovaj protokol omogućuje mrežnoj aplikaciji da može da koristi ove operacije i prosleđuje ih nižim nivoima radi efikasnog upravljanja

3. ***Sensor Query and Data Dissemination Protocol*** (SQDDP)

- ✓ ovaj protokol dozvoljava korisničkim aplikacijama da mogu da postavljaju upite SČ ili grupi i da tako smanjuju frekvenciju saobraćaja
- ✓ Obično su okrenuti protokolima zasnovanim na vrsti informacije (***attribute base***) ili na mestu lokacije (***location-base***).

Primer: *upiti koji se postavljaju* :

1. *Informaciju neka pošalju svi SČ koji izmere temperaturu* $\geq 30^0 C$
2. *Informaciju o temperaturi neka pošalju čvorovi iz regiona A*

11.3 Kodiranje podataka - kompresija

- U suštini sva glavna rešenja aplikacionog sloja mogu se svrstati u:
  1. kompresija podataka ili izvorno kodiranje,
  2. obrada upita
  3. upravljanje mrežom.
- **Kodiranje** podataka predstavlja prvi korak koji treba preduzeti kako bi se informacija prenela putem bežične komunikacije.
- Uvek kada SČ ima neku informaciju za prenos koju je dobio od SČ, potrebno je tu informaciju predstaviti u odgovarajućem obliku
- Izvorno kodiranje koristi tehniku da sa što manje bitova omoguči jedinstveno predstavljanje očitanog podatka koji se šalje.
- Često se izvorno kodiranje poredi sa kompresijom podataka.
- Redundantne informacije su komprimirane da bi se smanjila veličina podataka ali pod uslovom da se sačuvaju neki ili svi informacioni sadržaji kako se osnova informacije ne bi promenila ili izgubila.
- Osnovna rešenja za kompresiju mogu se klasifikovati u dva rešenja:
  1. **kompresija bez gubitaka** (*lossless compression*)
  2. **kompresija sa gubicima** (*lossy compression*).

11.3 Kodiranje podataka - kompresija

- **Kompresija bez gubitaka** smanjuje sadržaj podataka, tj. veličinu paketa, bez narušavanja integriteta informacija.
- Bolja efikasnost može se postići **kompresijom sa gubicima**, gde se postiže veće smanjenje podataka ali na račun gubitka nekih podataka.
- Zbog veće složenosti algoritama koji se koriste kod kompresije sa gubicima, koji zahtevaju znatno jače resurse, tehnike kompresije **bez gubitaka** generalno su usvojene za primenu u BSM.
- Razvijeno je nekoliko algoritama kompresije bez gubitaka koji su prilagodljivi kako za digitalnu komunikaciju tako i za izračunanje.
- Međutim, primena ovih algoritama u BSM je jako ograničena.
- Pošto je prog.memorija mnogih SČ ograničena, složeniji algoritmi kompresije kao što su **LZW** ili **gzip** ne mogu se direktno primeniti
- Neki algoritmi zahtevaju da postoji tablica konverzije za kompresiju /dekompresiju, što zahteva veće mem.resurse i veće angažovanje CPU.
- Pošto operacije čitanja/pisanja rezultiraju većoj potrošnji energije kod SČ, ova rešenja nisu energetski efikasna i opravdana za BSM.
- Redundantnost podataka takođe ne omogućava efikasnu kompresiju

11.3 Kodiranje podataka - kompresija

- Kako su SČ-ovi u nadgledanom regionu raspoređeni sa različitim gustinama moguće je da više SČ detektuje isti podatak
- Ako se ti podaci komprimuju na svakom SČ to predstavlja nepotrebno opterećenje za BSM, kako u vidu potrošnje električne energije tako i u pojačanom saobračaju (šalje se isti podatak).
- Umesto kompresije na individualnoj osnovi, redundantne informacije koje prikupljaju više SČ mogu se komprimovati samo kod jednog SČ
- To rezultira znatno efikasnijem rešenjem koje dovodi do znatnog smanjenja podataka koji se isporučuju *sink*-u a samim tim i manjoj potrošnji energije i manjem saobraćaju.
- Kao rezultat ovoga biće opisane dve distribuirane tehnike koje se koriste za kompresiju podataka u okviru BSM a koje koriste korelaciju različitih izvora za kompresiju podataka:

1. tehnike kodiranja kod centralnih, izvornih SČ - **Sensor LZW** tehnika
2. distribuirano izvorno kodiranje, koje raspoređuje kompresiju na više SČ-ova u mreži.

11.3 Senzor Lempel-Ziv-Welch S-LZW

- Postoji nekoliko algoritama kompresije opšte namene koji se koriste sa osnovnim ciljem da se poveća stepen kompresije podataka.
- Kod tih kompresija ne vodi se računa o energetskoj i memorijskoj efikasnosti tih algoritama
- Kodni prostor postojećih algoritama je suviše veliki da bi se implementirao u SČ sa ograničenom memorijom
- Postojeći algoritmi nisu pogodni za BSM, jer se zahtevaju energetski i memorijski efikasni algoritmi kompresije.
- U tipičnoj platformi SČ potrošnja energije za izračunavanje je znatno **manja** od one koja je potrebna za pristup memoriji ili komunikaciju.
- Neophodna su računarski intenzivna rešenja sa **malim** zahtevima za korišćenje memorije.
- **Senzor LZW** (S-LZW) je varijanta algoritma kompresije **Lempel–Ziv–Welch** (LZW) koji je prilagođen ograničenim resursima SČ.
- LZW je metod zasnovan na rečniku, koji kodira niz simbola kao posebnu reč (*token*) koja se pamti u rečniku.
- Rečnik je delimično inicijalizovan pre kodiranja i popunjen

11.3 Senzor Lempel-Ziv-Welch S-LZW

- Glavni izazov za eksploataciju LZW algoritma u BSM je negativan efekat grešaka u bežičnom kanalu.
- LZW dekodiranje se ne može izvršiti ako izlazni tok iz kodera-predaja nije u potpunosti (**bez greške**) primljen kod dekodera-prijemni SČ
- Podaci koji se komprimuju treba da budu ograničeni na **male blokove** tako da se retransmisije mogu obavljati bez velikog gubitka podataka i potrošnje energije.
- Zbog velikih ograničenja u memoriji, veličinu rečnika treba ograničiti
- S-LZW je dizajniran da odgovori na ove probleme i koristi veličinu rečnika od **512** token-a koji vrše kompresiju podataka u blokovima od **528** bajtova (dve stranice fleša).
- Kompresija posmatranih podataka može se dodatno poboljšati korišćenjem njihove redudantne prirode.
- Pošto svaki SČ posmatra fizički fenomen, podaci koje treba komprimirati su u velikoj meri zavisni i određeni vremenom.
- To znači da će biti nekoliko ponavljanja (istih podataka) u podacima koji se kompresuju.

11.3 Senzor Lempel-Ziv-Welch S-LZW

- Ovo se može rešiti dodavanjem strukture mini-keša koja će čuvati nekoliko zadnjih prikupljenih podataka.
- **S-LZW** sa mini-keš-om (**S-LZW-MC**) daje znatno bolji odnos kompresije u slučajevima kada podaci sadrže mnogo ponavljanja.
- Pored oslanjanja na interne međuzavisnosti podataka, efekat kompresije može se dodatno poboljšati pretvaranjem podataka u formu sa nekoliko šablona pre kompresije.
- Ovo se obično naziva **korak preduslova**, a S-LZW algoritam je poboljšan korišćenjem dva različita preduslovna metoda:
  1. Prva metoda odnosi se na ***Burrovs-Wheeler*** transformaciju (**BWT**), koja se generalno koristi za kompresiju slika, zvuka i teksta.
  2. Druga preduslovna metoda koja se koristi kod S-LZW bazira se na internoj strukturi podataka koji se prate.
- Na sledećem slajdu na primeru uzorka "**sviss_miss**" prikazan je princip rada ***Burrows–Wheeler*** transformacije.

11.3 Senzor Lempel-Ziv-Welch S-LZW

swiss_miss	_missswiss	s
wiss_misss	iss_misssw	w
iss_misssw	issswiss_m	m
ss_missswi	missswiss_	_
s_missswis	s_missswis	s
_missswiss	ss_missswi	i
missswiss_	ssswiss_mi	i
issswiss_m	sswiss_mis	s
ssswiss_mi	swiss_miss	s
sswiss_mis	wiss_misss	s

swiss_miss ==> swm_siisss

11.3 Senzor Lempel-Ziv-Welch S-LZW

- Druga preduslovna metoda koja se koristi kod S-LZW bazira se na internoj strukturi podataka koji se prate.
- U aplikacijama koje vrše nadgledanje neke pojave, dužina podataka i sadržaj su uglavnom unapred dobro poznati i fiksirani.
- Za niz vrednosti posmatranja, čitava opservacija (svi posmatrani podaci) imaju jako malo razlike u svojim vrednostima.
- Tako im se u kodiranoj reći bitovi največe težine im se poklapaju.
- Shodno tome, strukturni prenos (*structured transpose* - ST) se može izvršiti kako bi se dodatno poboljšao odnos kompresije.
- Preciznije, podaci za svako posmatranje se koriste da bi se popunili redovi u matrici sve dok se ne dostigne maksimalna dužina matrice.
- Nakon toga matrica se transponuje i podaci se komprimuju pomoću transponirane matrice.
- Pošto transponovana matrica ima nekoliko ponavljanja, odnos kompresije se može dodatno smanjiti.
- Ova verzija protokola označena je kao **S-LZW-MC-ST**.

11.3 Senzor Lempel-Ziv-Welch S-LZW

- S-LZW utiče na potrošnju energije u SČ u zavisnosti od arhitekture
- Kako se kompresija vrši na izvornim SČ-ima , glavna ušteda se može postići baš kod tih čvorova.
- U slučajevima kada potrošnja energije za primopredaju podataka dominira nad energijom koja se potroši za pristup memoriji, moguće je smanjiti potrošnju energije i **2,6** puta.
- Za platforme gde je potrošnja energije za komunika. i pristup memoriji slična(MicaZ),uštede energije zavise od vrste podataka koji se kompri.
- Glavna prednost korišćenja tehnika kompresije u BSM je potrošnja energije na globalnom nivou, tj. ušteda od kraja do kraja (***end-to-end***).
- Putem kompresije smanjujemo količinu podataka koje treba preneti
- Kao rezultat toga, manja količina podataka se prenosi širom mreže.
- Zbog toga, čak i ako kompresija dovodi do blago povećane potrošnje energije u izvornim SČ, ukupna ušteda energije na globalnom nivou su značajna za prenošenje informacija od izvornog SČ do sink-a
- Istraživanja su pokazala da to dovodi do smanjenja potrošnje energije do **2,9 puta**.

11.3 Senzor Lempel-Ziv-Welch S-LZW

- Preduslovne metode takođe poboljšavaju odnos kompresije, kao i energetsku efikasnost uprkos sankciji u izračunavanju transformacija i potrebnog dodatnog skladištenja podataka.
- Za **S-LZW-MC-BWT**, smanjuje se do **1,96 puta** za lokalne SČ i **2,1 puta** za *end-to-end* na nivou globalne mreže.
- Kod aplikacija za nadgledanje, **S-LZW-MC-ST** štedi energiju za **2,8 puta** i to i u lokalu i na globalnom nivou.
- Kodiranje izvornih podataka (kompresija) takođe pomaže u uštedi energije i zbog prirode bežičnog kanala.
- Zbog velikog procenta grešaka koje se javljaju prilikom prenosa paketa potrebno je izvršiti veći broj retransmisija tih oštećenih paketa kako bi se pouzdanost podigla na veći nivo.
- Pošto kompresija smanjuje količinu informacija koje treba poslati, troškovi reemitovanja su takođe smanjeni.
- Ovo dovodi do značajne uštede energije za višenamenske BSM-ove.

11.4 Obrada upita

- BSM se sastoje od velikog broja SČ koji prate fizički fenomen u skladu sa zahtevima aplikacije.
- *Sink* osigurava isporuku zahtevanih podataka putem upita upućenih u SČ-ove koji sadrže informacije o traženim informacijama.
- Odgovori na upit se mogu izvršiti jednostavno slanjem traženih sirovih podataka odmah u *sink*.
- Kako SČ poseduju mogućnost računarske obrade tih podataka, pružaju se i alternativni načini obrade ovih upita
- To može da omogući značajnu uštedu u potrošnji električne energije.
- Ovaj fenomen se naziva **procesiranje upita**.
- BSM se može posmatrati kao distribuirana baza podataka gde SČ neprestano šalju razne podatke prema *sink*-u.
- Iako postoji veliki broj sistema za upravljanje bazama podataka (DBMS) koji su razvijeni za tradicionalne distribuirane baze podataka, jedinstvene karakteristike BSM eliminišu te sisteme za BSM primenu
- Navešćemo samo neke od tih karakteristika:

11.4 Obrada upita

1. ***Streaming* podaci**: SČ stvaraju podatke kontinuirano, obično u dobro definisanim vrem.intervalima, bez eksplicitnih zahteva za tim podacima
2. **Obrada u realnom vremenu**: podaci sa SČ obično predstavljaju neki događaj koji se dogodio u realnom vremenu. Često je neizvodljivo sve te podatke čuvati u memoriji *sink*-a. Zbog toga, sve te podatke koji dolaze potrebno je obraditi u realnom vremenu (*real time*).
3. **Greške u komunikaciji**: kako u BSM postoji više *multi-hop* bežičnih linkova, velika je verovatnoća da se dogodi neka greška a prisutno je i određeno kašnjenje kod distribuiranih informacija koje dolaze do sink-a
4. **Neizvesnost**: podaci koje sakupljaju SČ podložni su jako velikom broju smetnji koje se nalaze u prirodi koje mogu izazvati pojedinačne procese koji mogu da ugroze dostavljanje informacije do *sink*-a.
5. **Ograničeni memorijski prostor**: SČ imaju strogo ograničenu memoriju koja je relativno jako mala. Zbog toga, informacije poslate od strane SČ nije moguće pamtiti (smestiti u bafer) u većoj količini.
6. **Obrada podataka u odnosu na komunikaciju**: u procesu upita treba prioroitet dati obradi podataka kako bi se uštedela elek.energija u SČ.

11.4 Obrada upita

- Procesorska snaga kojom raspolaže svaki SČ pruža potencijalna rešenja za probleme sa kojima se susrećemo kod obrade upita u BSM.
- Jasno je da se na upite koje je *sink* poslao lako može odgovoriti slanjem traženog podatka.
- Ovaj pristup se naziva ***skladištenjem*** (*warehousing*), gde je obrada senzorskih upita i pristup mreži SČ-ova odvojen.
- Kao rezultat, može se razviti centralizovani DBMS kako bi se omogućio pristup prikupljenim podacima putem klasičnih tehnika
- Pristup skladištenja, međutim, dovodi do prevelike upotrebe komunikacionih resursa u BSM-u, akumulacije visoko-redundantnih podataka u *sink*-u i povećanih zahteva za memorijom.

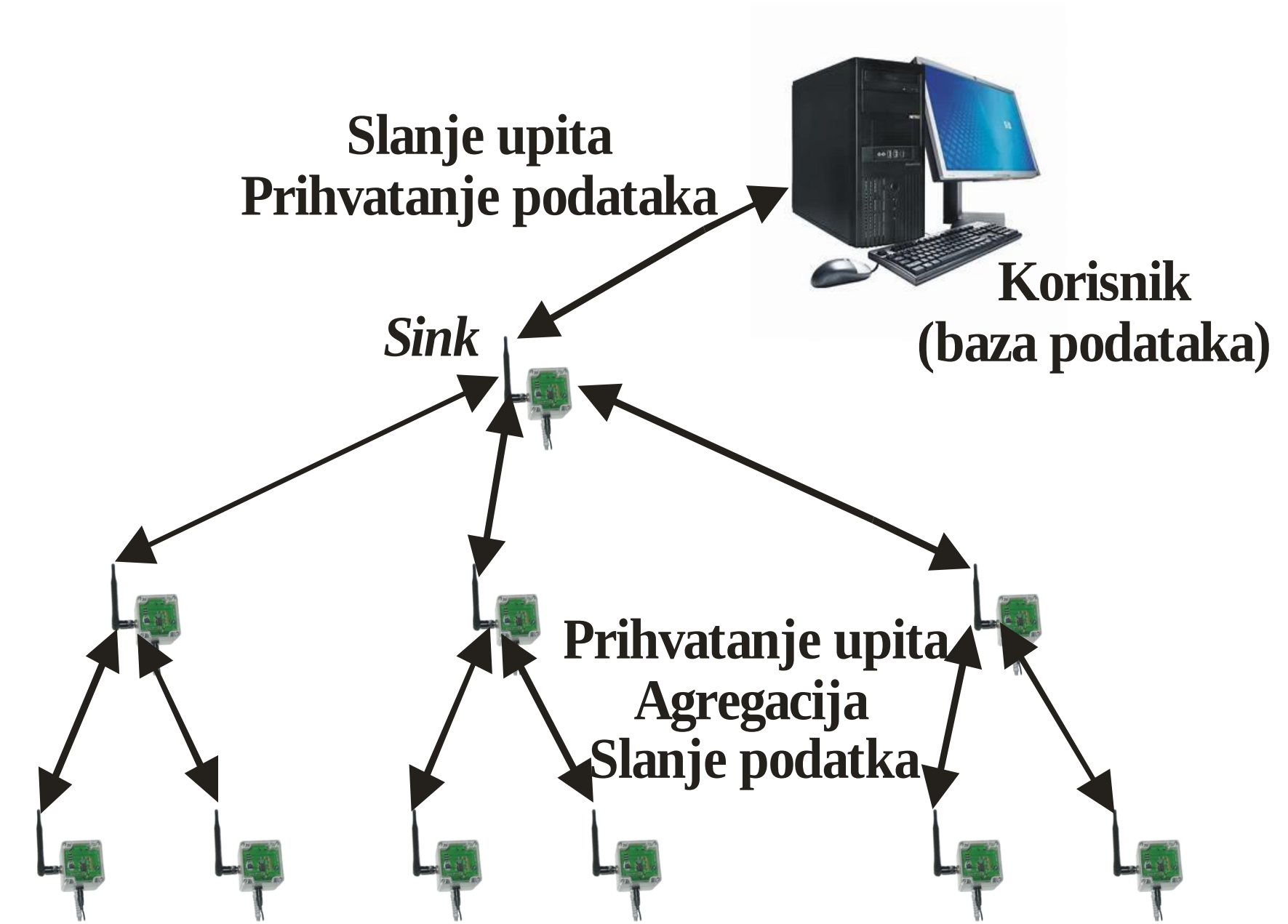
Primer: *ako je aplikacija zainteresovana samo za prosečnu vrednost specifičnih informacija na određenoj lokaciji, bilo bi efikasnije za SČ-ove na ovoj lokaciji da se izračuna prosek na lokalnom nivou i da se ova informacija kao jedinstveni jedan paket pošalje sink-u*

- To je razlog što korišćenje postojećih tehnika za obradu upita u BSM dovodi do neefikasnih rešenja.

11.4 Obrada upita

- Umesto toga, neophodni su novi pristupi prilagođeni uslovima BSM
- Rešenja za obradu upita pružaju neophodne alate za realizaciju ove interakcije između korisnika i distribuiranog skupa SČ.
- Usluge koje pružaju ova rešenja mogu se podeliti na dva dela: serverske usluge i BSM usluge.
- U svakom generisanju upita korisnički upit je predstavljen kroz dobro definisanu sintaksu koja se lako pretvara i u upite za ostatak mreže
- Ovaj upit predstavlja serversku stranu upita koji se nalazi u okviru rešenja za obradu upita.
- Usluge BSM mogu se široko podeliti u dva koraka:
  1. diseminacija upita - vrši se prenošenjem upita koji se generišu u skladu sa interesima korisnika, svim ili podskupovima SČ u BSM
  2. prikupljanje podataka - SČ odgovaraju na ove upite ako se njihovi lokalno posmatrani podaci sa senzora **poklapaju** sa zahtevima primljenog upita.
- Energetsko efikasno zastupanje i diseminacija upita, kao i prikupljanje podataka imaju veliki značaj za dizajniranje algoritama za obradu upita

11.4 Obrada upita



11.5 Operativni sistemi

- U primeni BSM postavljaju se različiti zahtevi koje je vrlo teško realizovati zbog mnogih ograničenja koje imaju SČ-ovi.
- Uzimajući u obzir sve to jasno je da standardni operativni sistemi kao što su Windows i Linux nisu primenljivi.
- Mnogi OS koji su razvijeni za *embeded* sisteme i *ad hoc* mreže (***QNX***, ***WinCE***, ***Ariel***, ***MagnetOS***) takođe nisu pogodni za primenu u BSM jer zahtevaju veće memorijske resurse i jače CPU.
- Neophodno je da se razviju potpuno novi, odgovarajući OS koji će sa jedne strane uspešno upravljati reduciranim hardver.mogućnostima SČ a sa druge strarne efikasno omogućiti **modularnost** i **robusnost** BSM.
- Glavna karakteristika jednog takvog OS je da omogući što jednostavnije razvijanje aplikacija za BSM-u, bez obzira na različite senzorske arhitekture koje se primenjuju i bez poznavanja hardverskih komponenti u samom SČ-u.
- Razvijeni su mnogi novi OS za rad SČ-ova u BSM-a:***TinyOS***, ***Contiki***, ***MANTIS***, ***LiteOS***, ***Nano-RK***, ***eCos***, ***SOS***, ***SenOS***, ***EYES***, ***kOS***,***DCOS***, ***Bertha***, ***Jallad***, ***Mate***, ***Accent***,***Chimes II***,***RetOS***,***MagnetOS***,***CORMOS***

11.5 Operativni sistemi

- Većina od ovih OS su strogo aplikativno ili hardverski orijentisani, pa nisu prilagodljivi na svim hardverskim platformama, niti mogu da predstavljaju rešenje koje će biti pogodno za sve BSM aplikacije.
- Kod izbora OS preporučuje se da se izabere neki ***open-source*** operativni sistem specijalno razvijen za BSM-e.
- Takav OS omogućava prilagođavanje različitim SČ arhitekturama uz minimizovanje potrebnog koda za njegovo izvršavanje.
- Jedan od najpopularnijih takvih OS, koji je i najviše zastupljen u senzorskim mrežama, je ***Tiny OS***.
- On predstavlja otvoreni, (*open source*) višekomponentni OS koji uključuje upravljanje memorijom, upravljanje jedinicama SČ, upravljanje zadacima i upravljanje protokolima.
- Kao namenski OS on reaguje na hardverske događaje (***event***), rukuje njima i podržava koncept taskova (***task***) koji su ekvivalentni funkcijama u drugim programskim okruženjima.
- TinyOS-a je napravljen da omogući realizaciju aplikacija koje se pokreću događajima koje jedino mogu da prekinu izvršavanje zadataka

11.5 Tiny OS

- **TinyOS** je OS koji je zbog svoje rasprostranjenosti u primenama u BSM aplikacijama postao sinonim za BSM.
- Zbog svojih karakteristika, kao i primenljivosti na gotovo svim poznatim hardverskim platformama za SČ, predstavlja OS koji se i najviše koristi na polju istraživanja u BSM.
- Kolika je popularnost ovog OS govori podatak da kada bi se sakupili svi ostali razvijeni OS za BSM, ne bi dostigli toliku primenljivost.
- TinyOS počiva na komponentnoj arhitekturi kod koje su i jezgro (***kernel***) OS i aplikacija smeštene u jednu celinu.
- Ne postoji klasična podeljenost na jezgro OS koje je nezavisno i izvršava različite aplikacije, već je to jedna jedinstvena celina
- Pri kompajliranju sve neophodne komponente, i TinyOS komponente i komponente potrebne za aplikaciju, smeštaju se u jednu izvršnu celinu
- Sada se samo ta komponenta instalira u SČ i ona izvršava sve potrebne zadatke koji se od aplikacije zahtevaju.
- Svaku komponentu karakteriše njeno **stanje** (***state***) kao i **zadaci** (***tasks***) koji se izvršavaju.

11.5 Tiny OS

- Komponente međusobno komuniciraju putem funkcijskih poziva koji mogu da budu **komande** (***commands***) i **događaji** (***events***).
- Komponenta koristi **komande** da inicira neku akciju i one se inicijalno šalju prema drugim komponentama od kojih se zahtevaju specif.akcije.
- Nakon što završe zadatak, komponenta obaveštava nadređenu komponentu putem **događaja**.
- TinyOS koristi dva nivoa u hijerahiji dodeljivanja CPU procesima.
- **Događaji** imaju veći prioritet u odnosu na **zadatke** i oni mogu da prekinu njihovo izvršavanje.
- Pored toga **događaji** mogu da daju signal za novi **događaj** ili za pozivanje **nove komande**.
- I **komande** i **događaji** ne smeju da traju dugo tj. da svojim izvršavanjem blokiraju rad procesora.
- Sve vremenski zahtevane akcije izvršavaju se putem zadataka koji imaju niži prioritet.
- U suštini **zadaci** predstavljaju formu odloženih procedura (***deferred procedure***) koje mogu da sačekaju svoje izvršavanje.

11.5 Tiny OS

- Na taj način omogućeno je da **događaji** unapred označe zadatke koji će se kasnije izvršiti.
- Nakon završetka, **zadaci** daju signal za jedan **događaj** koji obaveštava nadležnu komponentu da je zadatak završen.
- Zadaci i događaji mora da se potpuno završe nakon njihovog pozivanja
- U SČ događaji su predstavljeni putem hardverskih interapta, a zadaci se implementiraju putem linearnog FIFO raspoređivača (***dispatcher***)
- Raspoređivač formira red za zadatke koji treba da se izvršavaju gde je svaki zadatak predstavljen jednim pointerom na funkciju.
- Ukoliko je taj red prazan tada nema ni jednog zadatka koji čeka na izvršenje, pa sistem može da ide u stanje mirovanja (*sleep state*).
- TinyOS ne podržava višenitno izvršavanje jer se ovde radi o sistemima koji se upravljaju putem događaja/prekida (***event-driven system***)
- Procesi se u takvim sistemima implementiraju kao prekidni programi
- To nam omogućuje da svi ti prekidni programi koriste isti memorijski magacin što nam i smanjuje potrebu za operativnom memorijom u SČ
- Jedini problem se javlja kod procesa koji su vremenski jako zahtevni

11.5 Tiny OS

- Osnovna komunikacija u TinyOS bazira se na ***Active Messages*** (AM).
- Svaka AM poruka sastoji se od fiksne veličine od **36 B**, i svakoj poruci pridružen je jedinstveni broj jedno-bajtni ***ID handler***.
- **AM** omogućuju jedan nepouzdan protokol ali njegova prednost je da je on potpuno isti i za RF primopredajnik i za serijski port na SČ.
- Pod određenim uslovima TinyOS omogućava promenu aplikacije koja se nalazi u SČ bežično, i to preko posebno razvijenog **XPN** protokola
- Ti uslovi su: jedno-skokovita (*single-hop*) organizacija mreže, poseban punilac (*boot loader*) mora da bude u rezervisanom delu memorije SČ kao i da je komponenta **XPN** protokola povezana sa aplikacijom.
- Novi program se prihvata u spoljnu memoriju SČ odakle se nakon restovanja SČ putem specijalnog punioca prenosi u radnu memoriju.
- Posebnu pažnju TinyOS poklanja upravljanju potrošnjom SČ:

1. svaki servis koji se izvršava može se zaustaviti sa **StdControl.stop**
2. komponenta HPLPowerManagement vodi računa o modu rada CPU;
3. vremenski servis omogučuje da CPU radi u modu najmanje potrošnje;
4. raspoređivač zadataka automatski prebacuje CPU u režim smanjene potrošnje kad god je red koji čuva zadatke prazan

11.5 Tiny OS

- Kompletan TinyOS pisan je u specijalnom programskom jeziku **NesC**, koji predstavlja jednu modifikaciju standardnog C jezika.
- Cilj ovog jezika je da omoguči striktnu proveru aplikacionog koda kod kompajliranja kao i da omoguči lak razvoj TinyOS komponenata.
- Tokom kompajliranja progr.koda vrši se njegova provera na moguće logičke greške kao što su: mogućnost pojave trke (*race condition*) i provera korišćenja jedne iste promenljive u više asihronih događaja
- Pored osnovnih prednosti koje nam TinyOS pruža u vidu malog programskog koda (**400 B**), malim zahtevima za operativnom memorijom i komponentnim modelom koji nam omogućava lake izmene jedna od prednosti predstavlja i ugrađeni simulator TOSSIM.
- On nam omogučava da kompletnu aplikaciju razvijamo, debagiramo i testiramo na PC-ju i da takav nepromenjeni kod direktno prebacimo na senzorske čvorove.

11.5 Contiki OS

- **Contiki** je mešoviti model OS baziran na veoma malom jezgru, čija je uloga da na osnovu događaja (***event-driven kernel***), vrši aktiviranje procesa i to po metodi raspoređivanja poslova sa istiskivanjem (***preemptive multithreading***).
- Ovo jezgro je odgovorno za dinamičko punjenje ili pražnjenje OM modulima i programima koji zahtevaju izvršenje.
- Sam OS se sastoji od jezgra, biblioteka, punioca i različitih procesa
- Proces može da bude bilo koji aplikacioni program ili servis koji obezbeđuje sam OS.
- Servis predstavlja jednu funkciju OS koju mogu istovremeno da koriste više aplikacija.
- Svi procesi, i aplikacije i servisi, mogu se dinamički zamenjivati u toku samog rada.
- Međusobna komunikacija između procesa uvek se odvija preko jezgra.
- Jedna Contiki aplikacija prilikom kompajliranja uvek se deli na:
    1. jezgro aplikacije (core)
    2. programske module koji se učitavaju.

11.5 Contiki OS

- Jezgro jedne aplikacije se sastoji četiri interna modula:
  1. jezgro (kernel),
  2. programski punilac,
  3. sistemske biblioteke (*run-time libraries*)
  4. modula za servisiranje komunikacije.
- Kompletni binarni kod (*program image*) jezgra se čuva u SČ i ne može se menjati u toku rada, osim ako se ne koristi specijalni punilac.
- Veličina tog programskog jezgra koje je obavezno je nešto veća od TinyOS i iznosi oko **3874 B** (tačan broj zavisi od CPU).
- Kompajlirani programi ne mogu se direktno smeštati u operativnu memoriju već se pozivaju sa flash/EEPROM-a gde su ranije smešteni.
- Pozivanje programskih modula je u nadležnosti jezgra.
- Jezgro na osnovu događaja koje dobija od procesa koji se izvršavaju, koristeći metodu prozivke (*pooling*), poziva označene procese.
- Takav proces koji je počeo svoje izvršavanje ne može biti prekinut od strane jezgra, već se izvršava do kraja, ali u okviru procesa mogu biti implementirani interni mehanizmi koji mogu da dovedu do prekida

11.5 Contiki OS

- To znači da jezgro Contiki sistema jedino ima ulogu u dodeljivanju CPU raznim procesima i u razmenjivanju poruka.
- Ukoliko se pojavi aplikacija koja zahteva više-nitni model izvršavanja primenjuje se model raspoređivanja sa predpražnjenjem (*preemptive multitreading*) koji se naknadno učitava iz opcionalne biblioteke.
- U OS Contiki komunikacija je implementirana kao jedan servis u nameri da se omogući njegova promena u toku samog izvršavanja.
- Omogučeno je da istovremeno više komunikac. kanala bude aktivno
- Istovremeno može da ispitujemo/upoređujemo više različitih protokola
- Velika prednost ovog sistema sastoji se u tome što ima implementiranu podršku za povezivanje sa standardnim TCP/IP mrežama.
- **lwIP**(*light-weight IP*) i **µIP**(*micro IP*) predstavljaju 2 mala protokola, preko kojih se SČ sa Contiki OS, jednostavno povezuju sa Internetom.
- Jezgro Contiki sistema ne sadrži nikakvu podršku za kontrolu potrošnje električne energije, pa je to ostavljeno aplikacijama i mrežnim protokolima da vode računa o tome.

11.5 MANTIS OS

- **MANTIS** (*Multimodal Networks of In-situ Sensors*) OS (MOS) predstavlja OS koji koristi prednosti višenitnog (*multithread*) izvršavanja procesa: smanjenje vremena odziva, ekonomičnost izvršavanja i bolje iskoričćenje višeprocesorske arhitekture.
- Predstavlja višeslojeviti OS, sličan UNIX, koji se sam izvršava kao jedna nit i koji zajedno obezbeđuje efikasno funkcionisanje SČ.
- Poznat je kao višenamenski OS koji je primenljiv u različitim BSM
- Lako se implementira na gotovo svim poznatijim SČ platformama
- Kompletan programski kod napisan je u standardnom C progr.jeziku.
- Posebna pogodnost je da ima ugrađenu podršku za limitirane hardverske resurse SČ kao što su:
  1. upravljanje potrošnjom,
  2. dinamičko programiranje,
  3. brza promena konteksa procesa,
  4. kružno opsluživanje procesa (***raund robin***),
  5. veoma mali zahtevi za OM (min. **500 B** uključujući i prostor za jezgro OS, stek i raspoređivač/planer procesa (***scheduler***)).

11.5 MANTIS OS

- Višenitni sistem omogućuje da se paralelno izvršavaju procesi bez opasnosti da neki proces blokira drugi.
- Svakoj niti se dodeljuje jedan kvant vremena za izvršavanje zadatka.
- MOS sprečava da dugi tj. vremensko zahtevni procesi, blokiraju ili uspore izvršavanje vremensko osetljivih procesa.
- Dodeljivanje memorije u MOS-u vrši se putem statičkog kontinualnog razmeštanja, jer se ovde unapred poznaju memorijski zahtevi procesa.
- MOS vodi računa da ne dođe do preklapanja memor. prostora kako višenitnih procesa, tako i njihovih odgovarajučih magacina (*stack*).
- U inicijalnom postupku, kod uključenja SČ, nit MOS jezgra postavlja svoj magacin na zadnju adresu raspoloživog memorijskog prostora.
- Svaka naredna nit koja se izvršava svoj magacin nadovezuje na ovaj.
- Po prestanku rada niti taj se prostor oslobađa za neku drugu nit.
- Na taj način omogučeno je da aplikacija može dinamički da se izvršava na ograničenom, malom RAM prostoru.

11.5 MANTIS OS

- Komunikacija sa spoljašnjim svetom svodi se na dve komponente:
  1. **DEV** komponenta predstavlja sinhronu komunikaciju sa okolinom bez potrebe baferisanja podataka. Očitavanje senzora, pamćenje podataka i generator slučajnih brojeva predstavljaju tipične primere sinhronih komponenti koje daju podatak nakon što se operacija završi
  2. **COMM** komponenta radi asihrono.Za njeno ispravno funkcionisanje su neophodni baferi. Tipični predstavnici ovakve komunikacije su RF primo-predajnik i serijski port kod SČ.
- Da bi uštedeo potrošnju el.energije, MOS jezgro koristi jednostavan algoritam za upravljanje CPU jedinicom.
- Samo kada postoji jedna nit koja se izvršava,CPU je u aktivnom stanju
- Ako njih nema,tada CPU prelazi u mod smanjene potrošnje(*idle mode*)
- Izvršavanje MOS jezgra predstavlja takođe samo jednu nit, ali se i ona izvršava u *idle* modu ukoliko ne postoji neka druga nit za izvršavanje.
- Za efikasno upravljanje potrošnjom el.energije, sve komponente u MOS, u početnom, inicijalnom stanju, su isključene.

11.5 SOS OS

- **SOS** je OS koji se sastoji od jezgra koje ima implementiranu podršku za dinamičko ubacivanje ili izbacivanje programskih modula.
- Dinamička rekonfigurabilnost je primarna motivacija i cilj ovog OS.
- Moduli šalju poruke SOS jezgru i komuniciraju sa njim putem jedne posebne tabele preko koje mogu i da pozivaju druge module.
- SOS moduli ne predstavljaju zaokružene procese, već su to izvršni kodovi koji izvršavaju neki zadatak ili funkciju.
- Oni predstavljaju delove nekog procesa koji se dinamički pozivaju iz biblioteka kojima raspolaže SOS.
- Svi moduli pisani su u standadnom C jeziku gde svaki modul ima implementiran specifični jedinstveni kod (***basic message handler***) koji je povezan sa porukom koja zahteva taj modul.
- Kontrola izvršavanja modula definiše se preko dva mehanizma: putem poruka od planera preko specijalne funkcije, i poziva funkcija koje su registrovane od strane modula i potrebne su za simultano izvršavanje.
- Jos jedna prednost SOS-a je da on podržava TinyOS module direktno u svom aplikacionom kodu bez bilo kakvih prethodnih izmena u kodu.

11.5 SOS OS

- SOS koristi metod najboljeg pogodka (*best fit fixed block*) kod dinamičkog dodeljivanja memorije.
- Postoje 3 veličine blokova memorije koji se dodeljuju modulima
- Većina memor.zahteva, kao što su poruke, staju u najmanje blokove
- Veći blokovi su namenjeni kod aplikacija koje zahtevaju da se celi moduli pozovu i smeste u kontinualni deo memorije.
- Povezane liste slobodnih blokova za sve tri veličine, omogučuju jednostavno upravljanje slobodnim blokovima.
- Komunikacija sa periferijama se takođe odvija putem posebnih modula
- SOS jezgro uključuje senzorski API, koji nam pomaže kod očitavanja senzora a postoje posebni moduli za kontrolu UART-a i tajmera.
- Dinamička rekonfigurabilnost modula u SOS dozvoljava nam da modifikujemo softver na svakom individualnom SČ tokom rada BSM.
- To nam omogučava da menjamo namenu BSM, dodajemo nove softverske module ili skidamo neke koje SČ više ne koristi kao i da koristimo nove programske metodologije.

11.5 SOS OS

- SOS ne dozvoljava potpunu izmenu jezgra ali unapređuje **XNP** (mehanizam za promenu programskog koda kod TinyOS) jer omogućuje modularnu izmenu a ne potpunu promenu putem binarne slike (*image*) koja je znatno veća i vremenski duže traje.
- Pored toga omogučeno nam je da se izmene direktno instaliraju u programsku memoriju SČ, bez pristupa spoljnoj memoriji i bez dodatnog resetovanja SČ.
- Na ovaj način vrši se ušteda električne energije.
- Kreiranje jednog pouzdanog sistema koji omogučuje BSM programerima jedno jednostavno okruženje sa svim potrebnim servisima za efikasan rad SČ je drugi osnovni cilj SOS.
- Zahvaljujuči rekonfigurabilnoj, modularnoj strukturi SOS, kao i sistemskim servisima koji su na raspolaganju programeru (dinamička dodela memorije, planer procesa sa poštovanjem prioriteta) pisanje aplikacija je dosta pojednostavljeno.
- Imajući u vidu da se aplikacije, kao i SOS, pišu u jeziku C, na raspolaganju su nam mnoge prednosti koje nam taj jezik daje

11.5 Operativni sistemi

Mogučnosti	TinyOS	Contiki	MANTIS	SOS	LiteOS
Hardverske platf.	Gotovo sve	MSP430 AVR	Mica2, TelosB mica2dot	Mica2,Micaz XYZ	Micaz,IRIS, AVR CPU
ROM	432 B	3874 B	12 kB	20 kB	2080 B
RAM	46 B	> 250 B	500 B	>2 kB	104 B
Program. jezik	***nesC***	C	C	C	LiteC++
Izvršni model	Događaj Komponente	Događaj Nit	Nit	Moduli	Moduli, *Multi Threading*
Učenje	Složeno	Srednje	Lako	Lako	Lako
Komunikacija	*Active messages*	lwIP µIP	COMM DEV	Sensor API	*File-Assisted*
Zaštita memorije	Ne	Ne	Ne	Ne	Da
Rekonfigurabilni	Da	Da		Da	Da
Real-time podrš.	Ne	Ne	Ne	Ne	Ne
Kontrola potroš.	Da	Da		Da	
Kontrola greške		Da	Da	Da	
Multitasking		Da	Da	Da	Da
Prioritet raspore.		Da	Da	Da	Da
Promen.konteksa	Da				Da
Interni simulator	TOSSIM				AVRORA
Godina nastajanja	2000	2004	2005	2005	2008

11.5 Operativni sistemi

- Kao što se iz Tabele može videti operativni sistem ***TinyOS***, generalno gledano, ima najveći broj prednosti u odnosu na ostale OS.
- Upravo zbog toga on se veoma lako instalira na gotovo svim hardverskim platformama koje se primenjuju za SČ.
- Svi OS osim ***TinyOS***-a, podržavaju simultano izvršavanje više procesa
- Međutim, to i nije neka velika prednost u odnosu na ***TinyOS***, jer zbog prirode namene SČ, veoma mali broj BSM aplikacija ima tu potrebu
- U pogledu memorijskih resursa (ROM i RAM), ***TinyOS*** ubedljivo ima najmanje zahteve.
- To proizilazi iz činjenice da se aplikacioni kod formira u toku kompajliranja same aplikacije, pa tako kompajler može da ugradi samo komponente OS koje su neophodne za izvršavanje samo te aplikacije.
- Kod ostalih OS to nije slučaj, jer je jezgro OS odvojeno od aplikacije.
- Zato su zahtevi kako za ROM tako i za RAM memorijom znatno veći.
- Ono što nije dobro kod upravljanja memorijom je da nijedan OS nema ugrađenu podršku za zaštitu od preklapanja memorijskih prostora.
- Komunikacija sa spoljašnjim svetom u svim OS je podržana

11.5 Softverski moduli

1. **Drajveri SČ** – predstavljaju softverske module koji obavljaju osnovne funkcije oko funkcionisanja SČ-a. Mogu biti modularnog tipa (*plug-in*) i zavise od vrste, konfiguracije i stepena inteligencije SČ-a. Njihova uloga je da omoguće aplikaciji da komunicira sa hardverom SČ
2. **Komunikacioni programi** – upravljaju međusobnom komunikacijom između SČ-ova i omogućuju izvršavanje različitih protokola. Zaduženi su za: energetski efikasno usmeravanje paketa, pamćenje i prosleđivanje paketa, staraju se o pravovremenom pristupu medijumu za prenos, upravljaju topologijom BSM-e, vrše šifrovanje podataka i vrše kontrolu njegove ispravnosti.
3. **Komunikacioni drajveri** –zaduženi su da na fizičkom nivo upravljaju linijskim signalom koji se šalje. Tako su oni zaduženi da upravljaju prenosnim bežičnim kanalom, uključujući noseću frekvenciju signala i njihovu sihronizaciju, kodiranjem linijskog signala, detekcijom i korekcijom greške, određivanje nivoa snage predajnog signala kao i njegovom modulacijom.

11.5 Softverski moduli

4. **Programi za obradu podataka** – predstavljaju numeričke programe koji služe za obradu podataka, memorisanje niva snage predajnog signala i ostale bazične aplikacije: **agregacija** i **selekcija podataka**.
5. **Razvoj aplikacija** – prikupljanje podataka ne bi dalo efekta ukoliko ne bi postojala konkretna realizacija tih podataka tj. predstavljanje tih podataka korisniku. Zato treba približiti relativno male BSM prema mreži svih mreža, a to je Internet – ***Internet of things***.
6. **Razvoj simulatora senzorskih mreža** – da bi uspešno mogli da projektujemo BSM, potrebni su nam neki preduslovi, a jedan od najvažnijih preduslova je da imamo mogućnost da proverimo ispravnost rada jedne takve mreže pre nego što je postavimo i pustimo u rad. Dostupni simulatori se slabo bave simulacijom potrošnje i aspektom očitavanja podataka sa izvorišta. Postoji veliki broj simulatora, jer se oni vezuju za određeni operativni sistem na kome čvor radi, a neki od najpoznatijih su: Nido, Ns-2, OMNet++, GloMoSim, SENSE, BOIDS, Shawn, TOSSIM i drugi.

Hvala na pažnji !!!

Pitanja

? ? ?