# VREMENSKA SINHRONIZACIJA SENZORSKIH ČVOROVA U BEŽIČNIM SENZORSKIM MREŽAMA
# TIME SYNCHRONIZATION OF SENSOR NODES IN WIRELESS SENSOR NETWORKS

Mirko Kosanović, Visoka tehnička škola strukovnih studija u Nišu
Mile Stojčev, Elektronski fakultet u Nišu

**Sadržaj –** *Vremenska sinhronizacija predstavlja najkritičniji deo infrastrukture u bilo kom distribuiranom sistemu, a naročito se to odnosi na Bežične Senzorske Mreže (BSM), u kojima se sinhronizacija vremena u velikoj meri koristi. Mnoge aplikacije u BSM zahtevaju da lokalni časovnici individualnih senzorskih čvorova (SČ) budu sinhronizovani. Skoro bilo koji vid grupisanja podataka ili koordiniranog sakupljanja informacije, zahteva sinhronizovanje sa fizičkim vremenom, kako bi se pravilno rezonovalo o zbivanjima u fizičkom svetu. Međutim, iako su potrebe za tačnošću sata i njegova preciznost, znatno strožiji u BSM nego u tradicionalnim distribuiranim sistemima, ograničeni resursi kao što su raspoloživa energija i ograničenja kanala, otežavaju izvršenje ovog cilja. Ovaj rad predstavlja novi pogled na ovaj problem i predlože jednostavno rešenje koje nudi kompenzaciju razlike vremena između dva SN.*

***Abstract*** *– Time synchronization represent the most critical piece of infrastructure in any distributed system, especially for Wireless Sensor Networks (WSNs) which make particularly extensive use of synchronized time. Many applications of WSN need local clocks of individual sensor nodes (SN) to be synchronized. Almost any form of sensor data fusion or coordinated actuation requires synchronized physical time for reasoning about events in the physical world. However, while the clock accuracy and precision requirements are often stricter in WSN than in traditional distributed systems, energy and channel constraints limit the resources available to meet these goals. This paper presents a new view on the synchronization problem and suggests simply solution with delay compensation between two SN.*

## 1. UVOD

Veliki broj međusobno povezanih senzorskih čvorova (SČ), koji mogu potpuno samostalno da izvrše prikupljanje neke informacije, da tu informaciju obrade i pošalju nekom drugom SČ, predstavljaju Bežičnu Senzorsku Mrežu (BSM). Zbog svoje relativno male cene, gabaritno malih SČ, mogućnošću primene u nepristupačnim terenima, kao i potpuno autonomnog rada, one sve više nalaze primenu za rešavanje mnogih problema koji muče čovečanstvo. Gotovo da ne postoji ni jedna oblast ljudskog stvaralštva gde one ne mogu da nađu svoju primenu: od praćenja prirodnih pojava (vulkana, zemljotresa, poplava), kontrole gustine saobraćaja, nadgledanje važnih objekata, vojnog obaveštavanja, pa do praćenja pacijenata u zdravstvu. Gotovo sve ove aplikacije rade na jednostavnom principu da prate neke fizičke promene koje se događaju u nadgledanom regionu. Svi ti prikupljeni podaci šalju se dalje kroz mrežu do nekog centralnog SČ, u literaturi poznatom kao *sink*. Uloga ovog SČ je da sve te podatke prikupi, obradi, analizira i donese određene zaključke. U većini aplikacija od presudnog je značaja da se tačno zna vreme kada se neki događaj dogodio, kako bi se doneli ispravni zaključci o tom događaju. Neke aplikacije imaju jako stroge zahteve u pogledu vremenske tačnosti tj. razlike u vremenu kada je potrebno da donesu neki zaključak. Tako na primer, kod određivanja tačne pozicije nekog objekta koji se kreće, zahteva se da SČ koji šalju audio informaciju o tom kretanju, imaju vremensku usklađenost od max. 100μs [1]. Sa druge strane, znajući da je ušteda potrošnje električne energije uvek primaran cilj u gotovo svim aplikacijama u BSM, vremenska sinhronizacija svih SČ u BSM može da odigra jako veliku ulogu. Poznato je da BSM predstavljaju mreže koje se aktiviraju samo kada se desi neka promena. Tada se intenzitet detekcije i slanja podataka jako poveća. Međutim, to je u životnom veku BSM samo jedan mali period. Ostatak vremena svi SČ su dosta neaktivni pa nije potrebno da oni stalno budu u aktivnom stanju jer se tako nepotrebno rasipa dragocena električna energija. Ako se zna da je RF primopredajnik najveći potrošač energije u SČ, samo prostim njegovim periodičnim uključivanjem i isključivanjem, moguće je životni vek jedne BSM znatno produžiti. Odgovarajuća vremenska sinhronizacija svih SČ u mreži, omogućava nam, da pored periodičnog uključivanja i isključivanja RF modula, isto primenimo i na ostale delove SČ. Tako mikroprocesor može da radi jedan period u režimu smanjene potrošnje i da se kasnije vrati u normalni režim rada a da se to ne oseti na kvalitetu aplikacije koju on izvršava. Međutim, zbog jako siromašnih resursa SČ, kao i otežanih uslova u kojima BSM rade, odgovarajuča vremenska sihronizacija svih SČ nije tako jednostavan zadatak kao što je to kod standardnih žičanih mreža. Većina razvijenih sinhronizacionih protokola koji sa uspehom rade kod ovih mreža, u uslovima BSM su potpuno neupotrebljivi [2]. Vremenska sinhronizacija u BSM zahteva razvoj potpuno novih metoda, koje će biti primenljivije uslovima koji vladaju u njima. Neke od osnovnih osobina po kojima se BSM razlikuju od standardnih žičanih mreža, i koje znatno utiču na nepodudarnost metoda vremenske sinhronizacije su [3]:

1. *Dinamička promena topologije* – SČ moraju da imaju sposobnost da potpuno samostalno uspostave međusobnu mrežnu komunikaciju između sebe. Kako su oni jako podložni kvarovima (gubitak napajanja, nemogućnost uspostavljana veze, otežano komuniciranje) topologija

ovih mreža se i kod stacionarnih SČ dinamički često menja.

2. *Neravnomerna zastupljenost* - SČ se najčešće proizvoljno raspoređuju u regionu koji se posmatra. Njihov broj nije svuda ravnomeran i kreće se od nekoliko SČ po do nekoliko hiljada na istom prostoru, tako da može da se desi da veliki broj SČ daje istu informaciju.
3. *Adresibilnost SČ* – većina aplikacija u BSM ne zahteva da svi SČ imaju jedinstvenu adresu već se slanje podataka odvija na principu *broadcast*-a.
4. *Ograničeni resursi* – osobina koja verovatno najviše otežava primenu standardnih protokola za sinhronizaciju vremena. Ograničena količina energije, mala memorija, smanjeni računarski kapaciteti kao i mali gabariti, predstavljaju glavne odlike SČ. Ove osobine doprinele su da cena ovih jedinica bude jako mala, ali su sa druge strane jako ograničile njihove mogućnosti.

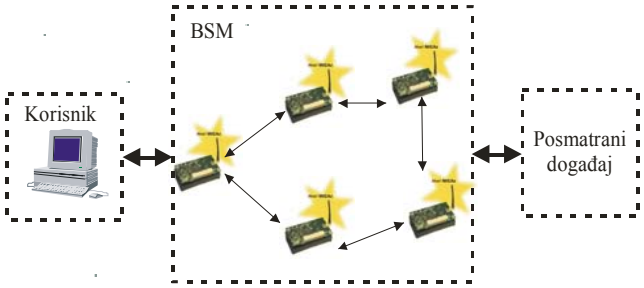
Sagledavajući sve ove razlike, razvoj jednog pouzdanog protokola za vremensku sinhronizaciju SČ, predstavlja jako veliki izazov. Upravo jedan takav protokol biće u fokusu interesovanja u ovom radu. Nakon uvoda u problem, u poglavlju 2 detaljnije ćemo objasniti potrebu za vremenskom sinhronizacijom u BSM i prikazati zašto nije moguće primeniti tradicionalne razvijene protokole. U poglavlju 3 daćemo generalni matematički model vremenskog sata, sa svim elementima koji utiču na razlike koje se javljaju između vremena u različitim SČ, kao i na kasniju njihovu vremensku sihronizaciju. Poglavlje 4 uvešće nas u strukturu našeg novog metoda koji predložemo u ovom radu. Na kraju, poglavlje 5 zaključuje ovaj rad.

## 2. VREMENSKA SINHRONIZACIJA U BSM

Mnogi razlozi postoje za razvoj novog pouzdanog protokola za vremensku sinhronizaciju u BSM. Dva tradicionalno najrasprostranjenija metoda za vremensku sinhronizaciju, koja se danas primenjuju su NTP (*Network Time Protocol*) i GPS (*Global Positioning System*) [4]. NTP je dobro poznati protokol za vremensku sinhronizaciju koji se primenjuje na Internetu i zahteva povećane resurse u pogledu električne energije i računarske snage. Sa druge strane GPS predstavlja poseban, dodatni uređaj, koji bi povećao cenu i gabarite svakog SČ u BSM, a on ne bi ni garantovao vremensku sihronizaciju svih čvorova u BSM. Vidimo da oba ova metoda zahtevaju značajne resurse u pogledu računarske snage, memorije, električne energije, cene kao i povećane gabarite (GPS). Imajući u vidu da su upravo svi ovi nabrojani resursi jako kritični kod BSM, jasno je da je primena ovih metoda potpuno nepogodna i neopravdana.

Sinhronizacija vremena u BSM je još komplikovanija jer ona zahteva sinhronizaciju tri elementa: korisnika, SČ i posmatranu pojavu (slka br.1) [5]. Ona može da se posmatra sa lokalne i globalne strane. Lokalna sihronizacija podrazumeva sinhronizaciju samo nekoliko susednih SČ, dok globalna podrazumeva sinhronizaciju svih SČ u mreži sa nekim referentnim vremenom.

*Sinhronizacija sa korisnikom* – U većini aplikacija za BSM, SČ prikupljaju i prosleđuju podatke do krajnjih korisnika, koji mogu da budu ili ljudi ili neki autonomni računarski sistemi. U većini slučajeva realna slika posmatranog fizičkog događaja, zavisi od informacije kada se stvarno dogodila promena na posmatranom objektu. Uzmimo za primer da posmatrtamo neki objekat koji se kreće. Ukoliko želimo da izračunamo njegovu brzinu kretanja ili njegovo ubrzanje, veoma je važno da dobijemo tačnu poziciju gde se on nalazio u određenim vremenskim trenutcima.




Slika br. 1. Vremenska sinhronizacija u BSM

*Sinhronizacija između SČ* – Jedan od glavnih ciljeva svake BSM je da maksimalno produži svoj radni vek. Glavna stavka u tome je ušteda u potrošnji električne energije. Kako su komunikacije najveći potrošaći energije jasno je da se tu može postići najveća ušteda. Od mnogih rešenja koja se ovde primenjuju najefiksanija ušteda se postiže primenom metode vremenskog multipleksiranja rada SČ, TDMA (*Time Division Multiplex Access*) [6]. Ova tehnika se svodi na dodeljivanju malih vremenskih intervala - slotova svakom SČ, u okviru kojih samo oni mogu da primaju ili šalju informaciju. Na taj način izbegnute su mnoge kolizije koje prouzrokuju ponovna slanja podataka koja su pak najveći potrošaći energije. Sa druge strane, omogućeno je da SČ mogu duži period da budu neaktivni, jer se oni uključuju samo kada dodje njihov dodeljeni vremenski interval. Međutim, da bi mogli da ostvarimo ovu podelu potrebna nam je jako precizna vremenska sinhronizacija svih SČ u mreži.

*Sinhronizacija sa posmatranom pojavom* – U BSM većina SČ može da detektujue istu pojavu. Na taj način javlja se veliki broj redudantnih podataka koji se nepotrebno šalju kroz mrežu a samim tim i nepotrebno troši energija. Osnovni cilj svake BSM je da redukuje te redudantne podatke i smanji nepotreban saobraćaj u mreži. Jedan od načina za to je da SČ izvrše redukovanje tih podataka, bilo putem eliminisanja ili agregacije istih. Da bi to mogli da urade neophodna im je informacija o vremenu kada je neki podatak očitan. Ukoliko postoji precizna vremenska sinhronizacija moguće je da sve podatke koji nose ‘’zastarelu’’ informaciju odbacimo.

Vremenska sinhronizacija između SČ može se postići i softverskim i hardverskim metodama. Kako hardverski metodi zahtevaju dodatne uređaje koji bi gabaritno pvećali veličinu SČ oni nisu preporučljivi za BSM. Softverski metodi su znatno prikladniji jer su jevtiniji i ne zahtevaju dodatnu opremu. Generalno gledano, svaka vremenska sinhronizacija može se posmatrati kroz više faktora koje definiše aplikacija u kojoj se ona primenjuje. Tako na primer stepen tačnosti, vremenski interval koji je potreban za sinhronizaciju, potrošena energija, broj poruka koje se razmenjuju, dužina trajanja kao i mogućnost kada ona može da se uradi, u mnogome određuju primenjeni vremenski sinhronizacioni protokol [7]. Svi ovi parametri služe nam kao referentne tačke kada všimo upoređivanje do sada razvijenih različitih sinhronizacionih metoda. Jasno je da je veoma teško razviti jednu metodu koja bi imala optimalne vrednosti po svim nabrojanim faktorima jer su oni dosta oprečni. Međutim, za

razvoj jednog pouzdanog vremenskog sinhronizacionog protokola za BSM, potrebno je uvek da imamo u vidu sledeće parametre [8]:

*Energetska efikasnost* – kako SČ imaju ograničenu količinu energije (kapacitet baterije), svaki sinhronizacioni protokol mora da bude efikasan u pogledu štednje energije, tj. potrebno je da uz minimalan utrošak energije postigne postavljeni cilj.

*Dostupnost(Lifetime)* – svaki sinhronizacioni metod treba uvek da bude dostupan tj. da može da izvrši sinhronizaciju vremena bilo kada i u bilo kojim uslovima.

*Stepen tačnosti* – vremenska sinhronizacija može da varira u svoj tačnosti od nekoliko μs do nekoliko sekundi pa i sati. Uglavnom ona zavisi od aplikacije u kojoj je primenjena. Na primer, bilo koja multimedijalna aplikacija zahteva tačnost sinhronizacije u μs, dok se kod sakupljanja informacije sa nekih on/off senzora (nadgledanje nekog prostora ili detekcija dana/noći) ona meri u satima pa i danima.

*Otpornost* – BSM obično rade u jako otežanim uslovima, kao što su nepristupačni tereni i sredine podložne raznim greškama. Usled toga, često dolazi do promena u tokovima kojima se šalju podaci. Sinhronizacioni metod mora ostati validan i funkcionalan, bez obzira na te promene, bilo da su one izazvane prestankom rada nekih SČ ili prekidom u vezi.

*Dinamička promena topologije mreže* – BSM predstavljaju mreže koje dinamički menjaju svoju topologiju. Više razloga utiče na ovu karakteristiku kao: otežana komunikacija, otkazi pojedinih SČ, usmeravanje podataka na optimalne rute, višeskokovita (multihop) topologija, vremenski uslovi, mobilnost SČ i td. Upravo zbog toga ovde se zahteva da se pre početka prenosa podataka primeni inicijalna faza uspostavljanja veze.

*Asimetrična komunikacija* – tradicionalne žičane mreže predstavljaju mreže kod kojih se razlike u trajanju prenosa poruka kao i nepredviđenih kašnjenja relativno veoma male. One i ako se jave, mogu se unapred predvideti i izračunati. Takođe, ove mreže spadaju u mreže sa simetričnim tokovima podataka tj. prenos poruka u oba pravca vremenski isto traje. U BSM, zbog njihove višeskokovite topologije koja zahteva višestruko preusmeravanje podataka koji se šalju, gore pomenute karakteristike ne važe. Kašnjenja su nepredviljiva i znatno veća, a uz sve to većina ruta je asimetrična tj. pravci kretanja podataka nisu isti u oba smera.

*Skalabilnost* – Različite gustine rasprostranjenosti SČ u nadgledanom prostoru, zahtevaju od sinhronizacionog metoda da potpuno isto razreši problem, bez obzira da li se radi o samo nekoliko SČ ili kada je taj broj znatno veći i meri se sa hiljadama SČ.

*Opseg sinhronizacije* – globalna sinhronizacija svih SČ u mreži može da bude jako skupo rešenje. Zato se od metoda sinhronizacije zahteva, da uz minimalne troškove, ostvari bar vremensko sinhronizovanje dva susedna SČ.

*Cena i veličina* – zbog ograničene veličine SČ, svaki metod koji zahteva dodatan uređaj koji bi se dodao SČ, nije logično rešenje za sinhronizaciju vremena. Takođe, sa obzirom na malu cenu SČ, potpuno je neprihvatljivo rešenje koje zahteva dodatna novčana ulaganja za dodatni uređaj koji bi po ceni znatno premašio cenu samog SČ.

*Ograničeni resursi* – sinhronizacioni metod ne sme da zahteva velike resurse u pogledu CPU snage i veličine memorije.

## 3. MODEL RAČUNARSKOG TAKTA

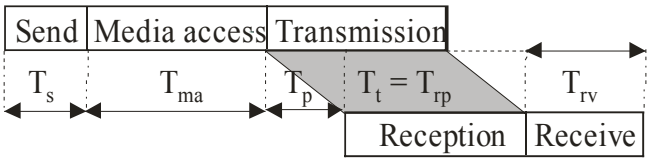
Lokalno vreme kod svakog SČ zasniva se taktu koji generiše prirodna frekvencija kvarc oscilatora. Kod svakog SČ razlikujemo dva takta i to: sistemski i spoljni. Tako na primer kod popularnog SČ *Mica*, postoji 4MHz sistemski takt i jedan spoljni takt koji generiše kristalni oscilator od 32,768 Hz [9]. Ovaj spoljnji oscilator može se programski konfigurisati da nam daje određeni vremenski interval. Još jedna dobra osobina ovog spoljnjeg takta je da on ne menja svoju frekvenciju kada SČ, tj. CPU promeni svoj režim rada, za razliku od sistemskog takta. Pošto se frekvencija ne menja, vremenski interval koji smo unapred programirali je konstantan, pa se može iskoristiti za beleženje lokalnog vremena u SČ. Potrebno je samo da nam taj interval periodično generiše prekidni signal koji će aktivirati prekidni program u kome će se jednostavno vršiti brojanje tih intervala. Međutim, većina ovih oscilatora nisu mnogo precizni, jer frekvencija kvarcih oscilatora nije kod svih ista. Mnogi parametri utiču da se ona razlikuje kao: tehnologija proizvodnje, temperatura, pritisak, napon napajanja SČ, starost kvarcnog oscilatora i td. Preciznost nekog oscilatora prikazuje se u jedinicama od 1ppm (*parts per million*) do 50 ppm [10]. To znači da ukoliko je nominalna frekvencija od samo 1MHz, vremenska razlika između dva SČ sa istom frekvencijom može da varira od 1-50 μs u okviru samo jedne sekunde. Drugim rečima, ako se ta razlika preračuna, za jedan dan dobija se da može da bude od 0,864-4,32 sekundi. Kako većina aplikacija u BSM jako zavise od tačnosti lokalnih satova u SČ, a kako ovaj period nije uopšte zanemarljiv, potrebno je eliminisati ove razlike koje se pojavljuju. Upravo zato neophodno je primeniti odgovarajuće sinhronizacione protokole koji treba da reše ovaj problem, i obezbede pravilno funkcionisanje aplikacije.

Da bi smo objasnili funkcionisanje tih protokola uvešćemo matematički model jednog kvarcnog oscilatora [11]. Tako, za neki SČ *i* u BSM, njegov lokalni takt možemo prikazati kao:

$$\mathbf{C}_i\,(t) = \mathbf{a}_i t + \mathbf{b}_i \qquad (1)$$

Ovde je $\mathbf{a}_i$ odstupanje takta (*clock drift*) a $\mathbf{b}_i$ razlika (*offset*) između taktova dva SČ. *Offset* predstavlja razliku u odnosu na vrednost nekog realnog, referentnog vremena, dok *drift* označava odstupanje u frekvencijama. Preciznije, *drift* predstavlja vremensku razliku dolaznog vremena dva signala koji bi trebalo da stignu u istu tačku istovremeno. On je direktna posledica izvitoperenosti (*skew*) takt frekvencije u dva SČ, bez obzira što oni imaju potpuno iste kvarc oscilatore. Pored već navedenih parametara koji mogu da utiču na ove razlike javljaju se i drugi kao što su: tipovi SČ, vrste primenjenog softvera, tipovi integrisanih kola, štampana ploča i drugi. Svi ovi parametri utiču da se takt frekvencije dva istovetna kvarc oscilatora tokom vremena znatno razlikuju. Jedan od načina da se te razlike usaglase je primena vremenskih sinhronizacionih protokola. Gotovo svi oni rade na principu razmenjivanja poruka koje sadrže lokalna vremena SČ. Na osnovu tog primljenog vremena, koje sada predstavlja referentno vreme, i svog lokalnog vremena, prijemni SČ može da izvrši korekciju i tako usaglasi svoje vreme. Do sada smo izložili parametre koji su uticali na

frekvenciju lokalnog takta kod SČ. Ovi parametri se često nazivaju i hardverski parametri koji utiču na različita vremena u SČ. Međutim u procesu sinhronizacije javljaju se i softverski parametri koji direktno utiču na kvalitet sinhronizacije vremena kod SČ. To su pre svega razna kašnjenja slanja i prijema paketa kod komunikacije između dva SČ (slika br. 2). Ovaj tip greške je mnogo važniji jer unosi znatno veće vremensko kašnjenje od hardverskih komponenti pa zato protokoli za sinhronizaciju vremena moraju naročitu pažnju da obrate na ove parametre.




Slika 2. Vremenska kašnjenja kod slanja i prijema paketa

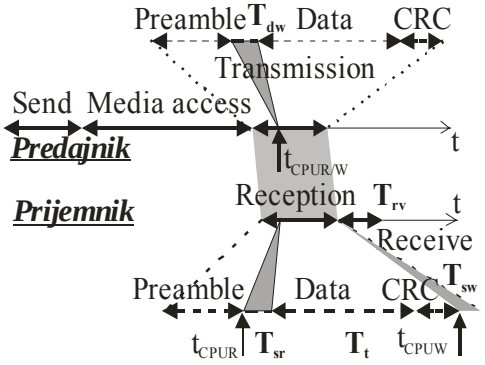
Kao što se sa slike 2. vidi process slanja jednog paketa može se podeliti na sledećih šest vremenskih faktora [12]:

1. *Send delay* ($T_s$) – ovo je vreme koje je potrebno da SČ koji šalje paket isti pripremi, formira strukturu paketa i isti preda u bafer primopredajnika. Njegova dužina zavisi od programa koji to radi, kao i od operativnog sistema koji treba da dodeli CPU za ovaj posao. To je nedeterminističko vreme i njegovo trajanje može da bude oko 100 ms.
2. *Media access delay* ($T_{ma}$) – predstavlja vreme koje potrebno da primopredajnik proveri medijum i sačeka da se on oslobodi kako bi mogao da pošalje paket i izbegne koliziju. Predstavlja jako specifično vreme koje u mnogome zavisi od jačine saobraćaja na medijumu kao i od primenjenih tehnika za izbegavanje kolizije (TDMA, CSMA or RTS/CTS). Njegovo trajanje je nedetrministično i iznosi od nekoliko milisekundi pa sve do nekoliko sekundi.
3. *Transmission delay* ($T_t$) – vreme koje protekne od trenutka kada je medijum slobodan pa sve do trenutka kada se ceo paket prenese do prijemnog SČ. Kako paket ima fiksnu dužinu, ukoliko znamo brzinu prenosa paketa lako možemo da izračunamo ovo vreme. Kako su razdaljine kod BSM jako male ovo vreme iznosi nekoliko milisekundi.
4. *Radio propagation delay* ($T_p$) – vreme koje je potrebno da bi signal prešao put od izvornog do odredišnog SČ. Kod tradicionalnih mreža ovo je dominantno vremensko kašnjenje zbog velikih rastojanja i dodatnih kašnjenja koje unose razni uređaji kroz koje prolazi paket. Međutim kod BSM, zbog jako kratkih rastojanja, od nekoliko desetina metara, pri brzini prostiranja signala od 300 m/μs, ovo vreme je manje od 1μs pa ga možemo potpuno zanemariti.
5. *Reception delay* ($T_{rp}$) – trajanje ovog perioda je identično kao i *transmission delay*, i predstavlja vremenski period koji je potreban da bi se ceo paket prihvatio u odredišnom SČ.
6. *Receive delay* ($T_{rv}$) – predstavlja vreme koje protekne od trenutka kada je paket primljen u baferu primopredajnika, pa do trenutka kada se primljena vremenska informacija postavi u lokalni programski brojač. Ovo vreme najviše zavisi od karakteristika prijemnog SČ i vrlo je slično po trajanju sa *send delay*.

Posmatrajući ove vremenske faktore možemo da primetimo da $T_t$, $T_p$ i $T_{rp}$ ne predstavljaju neku prepreku u vremenskoj sinhronizaciji jer su ta kašnjenja deterministička pa se lako mogu izračunati. Za razliku od ovih kašnjenja druga tri vremenska perioda $T_s$, $T_{ma}$ i $T_{rv}$, predstavljaju glavne probleme koje treba rešavati kod sinhronizacije vremena kod dva SČ. To se naročito odnosi na $T_{ma}$ koje je potpuno nedeterminističko vreme a i vremenski može najduže da traje.

## 4. METOD KOMPENZACIJE KAŠNJENJA

Glavna ideja predloženog metoda za sinhronizaciju vremena je da se izbegnu nedeterministička kašnjenja kod slanja paketa i da se svedu na kašnjenja koja se mogu na osnovu unapred poznatih parametara izračunati. Predložena metoda odnosi se na sinhronizaciju dva SČ, ali je ona potpuno primenljiva i na ostale SČ u celoj mreži. Razmotrimo sada sva prethodna nabrojana vremenska kašnjenja i objasnimo kako su ona rešena u našoj predloženoj metodi. Prva dva kašnjenja $T_s$ i $T_{ma}$, predstavljaju najveći problem za određivanje njihove dužine trajanja [13]. U predloženom rešenju ta vremenska kašnjenja su izbegnuta tako što se očitavanje lokalnog sata vrši u prekidnoj rutini u trenutku kada prenos paketa započne. Kako se svaki paket koji se šalje sastoji iz tri dela: preambule, podataka i CRC koda, moguće je da se vrednost lokalnog vremena ubaci u sekciju podataka nakon što primopredajnik počne da emituje preambula bajtove. Ovde je potrebno da se vreme potrebno za ovu operaciju izračuna ($T_{CPUR/W}$) i doda očitanom lokalnom vremenu koje se šalje (slika br.3). Sledeće kašnjenje unosi *Transmission delay* ($T_t$), koje je determinističko i može se izračunati ako se zna rastojanje između SČ, kao i brzina kojom se paket šalje. Kako je ovo vreme identično *Reception delay* ($T_{rp}$), mi ga izračunavamo kod prijemnog SČ. *Propagation delay* ($T_p$) je zanemarljivo malo pa ga mi i ne tretiramo u našem predlogu. *Reception delay* predstavlja vremenski period koji je potreban da se prihvati celokupni paket u prijemnom SČ. Da bi izračunali to vreme uveli smo još dva vremenska intervala, $T_{CPUR}$ i $T_{CPUW}$. Naime, nakon prijema preambula bajtova, a pre očitavanja *payload* podataka, beleži se trenutno stanje lokalnog sata prijemnog SČ ($T_{CPUR}$). Nakon prijema svih podataka po drugi put se beleži to isto vreme ($T_{CPUW}$). Jednostavnim oduzimanjem ovih beleženih vremena mi dobijamo koliko je trajao proces slanja paketa. I na kraju imamo zadnje izračunavanje *Reception delay* ($T_{rp}$), koje možemo lako izračunati pošto nam je poznat broj taktova za programski kod kao i brzina rada CPU prijemnog SČ.




Slika 3. Vremenski dijagram slanja i prijema paketa

Kao što vidimo za izračunavanje ukupnog kašnjenja potrebno nam je da izračunamo pet vremenskih kašnjenja i to: $T_{dw}$, $T_t$, $T_{sr}$, $T_{sw}$ i $T_{rv}$. Od toga jedno kašnjenje izračunava

predajni SČ a ostala kašnjenja se vrše kod prijemnog SČ. Bitno je istaći da izračunavanja vremenskih kašnjenja, a samim tim i celokupnog kašnjenja, može da se izvrši samo na prijemnom SČ i da je za njihovo izračunavanje potrebno samo da znamo ukupan broj CPU ciklusa za programski kod koji to treba da izvrši, kao i brzinu na kojoj radi CPU SČ. Ovde treba napomenuti da je predložen metod posebno pogodan za CPU kod SČ, jer se kod njih programski kod izvršava sekvencijalno i nema nekih dodatnih kašnjenja koja mogu da se pojave kod *pipelining* ili *caching* metoda izvršavanja programa. Sa druge strane imamo veliki broj primopredajnih čipova iz serije *Chipcon Smart RF*(CC1000, CC2420) koji nam dozvoljavaju da ubacimo poruku u paket koji je već počeo da se šalje (preambula bajtovi) [14].

Ukupno vreme kašnjenja može se predstaviti sa sledećim jednačinama prikazanim (1) do (6) :

$$T_{dw} = n_1 t_{SCPU} \quad (1)$$

$$T_{sr} = n_2 t_{DCPU} \quad (2)$$

$$T_{sw} = n_3 t_{DCPU} \quad (3)$$

$$T_{rv} = n_4 t_{DCPU} \quad (4)$$

$$T_t = t_{CPUW} - t_{CPUR} \quad (5)$$

$$T_{sinh} = T_{dw} + T_{sr} + T_{sw} + T_t + T_{rv} \quad (6)$$

Gde su:

$T_{dw}$ – vreme koje protekne da bi CPU predajnog SČ pročitao lokalno programsko vreme, izračunao vreme koje je potrebno za izvršavanje ovog koda, da ga doda pročitanom lokalnom vremenu i da se ta vrednost ubaci u *payload* podatke paketa.

$n_1$ – broj CPU ciklusa programskog koda kod predajnog SČ.

$t_{SCPU}$ – vremensko trajanje jednog CPU ciklusa kod predajnog SČ.

$T_{sr}$ – vremenski period potreban za očitavanje lokalnog programskog sata kod prijemnog SČ.

$n_2$ - broj CPU ciklusa programskog koda za čitanje lokalnog sata kod prijemnog SČ.

$t_{DCPU}$ – vremensko trajanje jednog CPU ciklusa kod prijemnog SČ.

$T_{sw}$ - vremenski period potreban za očitavanje lokalnog programskog sata kod prijemnog SČ

$n_3$ - broj CPU ciklusa programskog koda potrebnog za očitavanje lokalnog programskog sata kod prijemnog SČ

$T_t$ – vremenski period potreban za prenos celog paketa.

$T_{rv}$ - vremenski period potreban za izračunavanje celokupnog kašnjenja i ažuriranja lokalnog programskog sata kod prijemnog SČ.

$n_4$ - broj CPU ciklusa programskog koda potrebnog za izračunavanje $t_{rv}$ .

$T_{sinh}$ – ukupno trajanje svih kašnjenja kod prenosa paketa.

Navešćemo sada neke od prednosti našeg predloženog metoda: ne zahteva neke jake računarske resurse u vidu memorije i jačine CPU, jednostavna implementacija, zahteva se samo prenos jedne poruke a ne međusobnu razmenu poruka(*round-trip*), mala potrošnja energije, pogodan za mobilne SČ jer ne zavisi od međusobnog rastojanja SČ.

## 5. ZAKLJUČAK

Prednosti koje nam donosi sinhronizacija vremena svih SČ u BSM doprinela je da je ona postala neophodna i sastavni deo gotovo svih aplikacija u BSM. Međutim zbog mnogih ograničavajućih faktora koji vladaju u BSM, nije jednostavno pronaći efikasan vremenski protokol koji bi ispunio sve zahteve. Ovim radom pokušali smo da predložimo jedno jednostavno rešenje koje neće zavisiti od mnogih nedeterminističkih faktora koji utiču na raznolikost vremena koja se beleže u SČ. Osnovna prednost predloženog rešenja je da je ovaj metod vrlo jednostavan za implementaciju i ne zahteva velike resurse za njegovu realizaciju. Svi parametri za izračunavanje ukupnog kašnjenja svode se na poznavanju isključivo determinističkog podataka kao što su brzina rada CPU i dužina programskog koda.

## LITERATURA:


[1] I.Rhee, J.Lee, J.Kim, E.Serpedia, Y.Chung Wu, ``Clock Synchronization in WSN: An Overview'', Sensors 2009, www.cs.uic.edu/~ajayk/ext/ClockSyncWSNsurvey.pdf, acc. 20.12.2010

[2] S.Williams, K.Frampton, I.Amundson, P.Schmidt, ``Decentralized acoustic source localization in a distributed sensor network``, Applied Acoustics Vol. 67, Issue 10, Pages 996-1008, 2006

[3] F.Sivrikaya, B.Yener, ``Time Synchronization in Sensor Networks: A Survey'', http://www.cs.rpi.edu/~yener/PAPERS/WINET/timesync04.pdf, acc. 21.01.2011

[4] J.Elson, D.Estrin, ``Time Synchronization in WSN'', http://research.cens.ucla.edu/people/estrin/resources/conferences/2001apr-Elson-Estrin-Time.pdf, acc. 5.12.2010

[5] Peng Cheng, ``Synchronization and Timing Issues in WSN'', http://www2.hh.se/staff/tola/cooperating_embedded_systems/papers/cheng_peng_final.pdf , acc.5.12.2010

[6] B.Sundararaman, U.Buy, A.D.Kshemkalyani, ``Clock Synchronization for WSN: A Survey'', http://www.cs.uic.edu/~ajayk/ext/ClockSyncWSNsurvey.pdf, a.5.12.2010

[7] Jeremy Eric Elson, ``Time Synchronization in WSN``, Doctor dissertat., University of California, Los Angeles, 2003

[8] B.Kulaklib, ``Time synchronization in WSN``, Master thesis, Izmir Institute of Technology, October 2008

[9] K.Romer, P.Blum, L.Meier, `` Time Synchronization and Calibration in WSN``, http://www.vs.inf.ethz.ch/publ/papers/wsn-timebook.pdf, acc. 20.12.2010

[10] J.Greunen,J.Rabaey,``Lightweight Time Synchronization for Sensor Networks``, http://bwrc.eecs.berkeley.edu/Publications/2003/presentations/LightweightTime/2144597576.pdf, acc. 20.12.2010

[11] X.Chao-Nong,Z.Lei,X.Yong-Jun,L.Xiao-Wei, ``Simsync: A Time Synchronization Simulator for Sensor Networks``, Acta Automatica Sinica, Vol.32, No.6, 2006

[12] S.Rahamatkar, A.Agarwal, N.Kumar, `` Analysis and Comparative Study of Clock Synchronization Schemes in WSN``, http://www.enggjournals.com/ijcse/doc/ IJCSE10-02-03-28.pdf, acc. 20.12.2010

[13] S.Ping,`` Delay Measurement Time Synchronization For WSN``,http://www.intel-research.net/Publications/Berkeley/ 081120031327_137.pdf, acc. 20.12.2010

[14] Texas Instruments, Application Report, `` MSP430 Interface to CC1100/2500 Code library``, http://focus.ti.com/lit/an/slaa325a/slaa325a.pdf, acc. 20.12.2010