

UNIVERZITET U BEOGRADU  
ELEKTROTEHNIČKI FAKULTET

MASTER RAD

Implementacija vremenske sinhronizacije u  
Namenskim računarskim sistemima

mentor:  
Prof. dr Lazar Saranovac

student:  
Lazar Caković  
br. indeksa: 3083/2016

Beograd, septembar 2018.

# Sadržaj

<b>1</b>	<b>Apstrakt</b>	<b>3</b>
<b>2</b>	<b>Uvod</b>	<b>4</b>
<b>3</b>	<b>Protokol</b>	<b>5</b>
3.1	OSI model . . . . .	5
3.1.1	Sloj 1: Fizicki sloj (Physical Layer) . . . . .	6
3.1.2	Sloj 2: Sloj veze (Data Link Layer) . . . . .	6
3.1.3	Sloj 3: Mrezni sloj (Network Layer) . . . . .	7
3.1.4	Sloj 4: Transportni sloj (Transport Layer) . . . . .	7
3.1.5	Sloj 5: Sloj sesije (Session Layer) . . . . .	8
3.1.6	Sloj 6: Sloj prezentacije (Presentation Layer) . . . . .	8
3.1.7	Sloj 7: Aplikativni sloj (Application Layer) . . . . .	9
3.2	Internet protocol suite <i>TCP/IPmodel</i> . . . . .	9
<b>4</b>	<b>Operativni sistem</b>	<b>12</b>
4.1	FreeRTOS . . . . .	12
4.2	lwIP . . . . .	15
<b>5</b>	<b>Hardverska implementacija</b>	<b>18</b>
5.1	Time stamping unit - TSU . . . . .	20
<b>6</b>	<b>Softverska implementacija</b>	<b>22</b>
<b>7</b>	<b>Zaključak</b>	<b>23</b>
<b>8</b>	<b>CHECK</b>	<b>24</b>
8.1	Precision Time Protocol . . . . .	24
8.2	Algoritam najboljeg sata <i>Bestmasterclockalgorithm</i> . . . . .	26
8.3	Sinhronizacija (Synchronization) . . . . .	27

# Slike

5.1	Razvojna ploča SAMA5D27-SOM1-EK1 (pogled odozgo) . . .	18
5.2	Moduli - Razvojna ploča SAMA5D27-SOM1-EK1 . . . . .	19
5.3	Moduli - SAMA5D27-SOM1 . . . . .	20

# Glava 1

## Apstrakt

## Glava 2

### Uvod

# Glava 3

## Protokol

### 3.1 OSI model

Open Systems Interconnection model (OSI model) je konceptualni model koji karakterise i standardizuje komunikacione funkcije u telekomunikacionim ili kompjuterskim sistemima, i to bez obzira na unutrašnju strukturu uređaja ili njihovu tehnologiju. Cilj ovog modela je da se postigne kompatibilnost različitih komunikacionih sistema sa standardnim protokolima komunikacije. OSI model razdvaja komunikacione sisteme u apstraktne slojeve. Originalna verzija modela ima sedam slojeva.

Sloj unutar modela služi sloj iznad njega, i koristi sloj ispod njega u hijerarhiji. Na primer, sloj koji postize komunikaciju preko mreže bez gresaka, služi aplikacijama iznad koje ga koriste, i to dok poziva jednostavne funkcije za prijem i predaju paketa na mreži. Dve instance istog sloja su vizualizovane tako što su povezane horizontalno u istom sloju.

Ovaj model je proizvod Open Systems Interconnection projekta u International Organization for Standardization (ISO), i ima oznaku ISO/IEC 7498-1.

## [lazar] slika svih slojeva u OSI modelu

Na svakom nivou N, dva entiteta na komunikacionim uređajima razmenjuju jedinice protokola (PDU - protocol data units) pomoću sloja N protokola. Svaki PDU sadrži podatke od interesa (payload) (SDU - service data unit), zajedno sa zaglavlјima koji odgovaraju protokolu.

Obrada podataka između dva uređaja koji su OSI-kompatibilni se odvija u sledećim koracima:

- Podaci koji se prenose se formiraju na najvišem sloju u uređaju koji predaje podatke na mreži (sloj N) u jedinicu protokola (PDU).
- PDU se prosledjuje sloju N-1, gde je poznat kao SDU.

- Na sloju N-1 se na SDU dodaju zaglavlja, na osnovu cega se formira PDU za sloj N-1. Nakon cega se prosledjuje na sloj N-2.
- Ovaj postupak se ponavlja sve dok se ne dostigne najnizi sloj u modelu, nakon cega se podaci prenose ka uređaju koji prima podatke.
- Na strani prijemnog uređaja se podaci prenose od najnižeg sloja u modelu, do najvišeg, gde se serije SDU struktura uspešno obrađuju, pri čemu se skidaju zaglavlja sa svakog sloja, dok se ne dostigne najviši sloj u modelu, nakon čega su dostupni sirovi podaci.

## [lazarc] dodati jos nesto o ovome ukoliko se nadje.

### 3.1.1 Sloj 1: Fizicki sloj (Physical Layer)

Fizicki sloj je odgovoran za prenos i prijem nestrukturiranih sirovih podataka izmedju uređaja i fizickog medijuma za prenos. On pretvara digitalne bitove u elektricne, radio ili opticke signale. Specifikacije sloja definisu karakteristike poput nivoa napona, fizicke brzine prenosa podataka, maksimalne udaljenosti prenosa i fizickih konektora. Ovo ukljucuje raspored pinova, napona, linijske impedanse, specifikacije kablova, vremenskih signala i frekvencije za bezicne uređaje. Kontrola brzine bitova se vrši na fizickom nivou i moze definisati nacin komunikacije kao simpleks, polu dupleks ili dupleks komunikaciju. Komponente fizickog sloja mogu se opisati u smislu topologije mreze. Bluetooth, Ethernet i USB, sve imaju specifikacije za fizicki sloj.

### 3.1.2 Sloj 2: Sloj veze (Data Link Layer)

Sloj veze podataka obezbedjuje prenos podataka izmedju dva cvora u komunikaciji - vezu izmedju dva direktno povezana uređaja na mrezi. Ovaj sloj otkriva i eventualno ispravlja greske koje se mogu javiti u fizickom sloju. On definise protokol za uspostavljanje i prekid veze izmedju dva fizicki povezana uređaja. Takodje, definise protokol za kontrolu protoka izmedju njih.

IEEE 802 standard deli sloj veze na dva podsloja:

- **Kontrola pristupa medijumu (MAC - Medium access control):** ## [lazarc] [prevod] odgovorna je za kontrolu nacin na koji uređaji na mrezi dobijaju pristup medijumu i dozvolu za prenos podataka.
- **Kontrola logicke veze (LLC - Logical link control):** ## [lazarc] [prevod] odgovorna je za identifikaciju i enkapsulaciju slojeva mreznog protokola, i kontrolu gresaka i sinhronizaciju paketa koji se salju.

MAC i LLC slojevi IEEE 802 mreznog standarda kao sto su 802.3 Ethernet, ili 802.11 Wi-Fi su slojevi veze (data link layer).

Point-to-Point Protocol (PPP) - je protokol sloja veze koji radi na nekoliko razlicitih fizickih slojeva, kao sto su sinhrona ili asinhrona serijske linije.

### 3.1.3 Sloj 3: Mrežni sloj (Network Layer)

Mrežni sloj obezbedjuje funkcionalno i proceduralno sredstvo prenosa sekvenci podataka promenljive duzine, koji se nazivaju jos i paketi, od jednog cvora do drugog, povezanih u "razlicite mreze". ## [lazarc] [prevod] ## Mreza je medijum na koji moze biti povezano vise cvorova, u kom svaki cvor ima adresu i koji dozvoljava cvorovima povezanim sa njim da prenose poruke ka ostalim cvorovima, i to samo dajuci sadrzaj poruke i adresu cvora na koji poruka treba da bude dostavljena, omogucavajući mrezi da nadje nacin da isporuci poruku odredisnom cvoru, eventualno ga usmeravajući kroz sredisnje cvorove (uredjaje koji su izmedju dva uredjaja koji pokusavaju da komuniciraju). Ako je poruka prevelika da bi se prenela sa jednog uredjaja na drugi samo koriscenjem sloja veze (Data Link Layer), mreza moze preneti podatke tako sto ce ih podeliti u nekoliko delova na jednom uredjaju, poslati delove nezavisno, i onda spojiti delove na drugom uredjaju. Pri cemu moze, iako nije uvek potrebno, prijaviti greske u isporuci.

Isporuka poruka na mreznom sloju nije garantovano pouzdana. Mrežni sloj moze pruziti pouzdanu isporuku poruka, ali nije obavezno da to mora biti ispunjeno.

Neki broj protokola koji upravljaju slojevima, imaju funkciju koja je definisana u aneksu upravljanja, ISO 7498/4, i pripadaju mreznom sloju. Oni ukljucuju protokole rutiranja, upravljanja grupomsa vise uredjaja, informacije o mreznom sloju, o greskama, kao i dodeljivanje adresa mreznog sloja medju uredjajima. Ustvari, to je funkcija podataka koji se prenose uz pomoc protokola, sto ih cini da pripadaju mreznom sloju, a ne protokolu. ## [lazarc] poslednja recenica [prevod] ##

### 3.1.4 Sloj 4: Transportni sloj (Transport Layer)

Transportni sloj obezbedjuje funkcionalno i proceduralno sredstvo prenosa sekvenci podataka promenljive duzine od predajnog do prijemnog uredjaja, uz odrzavanje kvaliteta.

Transportni sloj kontrolise pouzdanost date konekcije kroz kontrolu protoka, segmentaciju/desegmentaciju i kontrolu gresaka. Neki protokoli su orjentisani na stanje mreze, a neki na konekciju u mrezi. ## [lazarc] [prevod] ## Ovo znaci da Transportni sloj moze da prati segmente i ponovo preneti one



koji nisu isporučeni prijemnom uređaju. Transportni sloj takođe omogućava i potvrdu uspešnog prenosa podataka i šalje naredne podatke ako nije došlo do greške prilikom prenosa. Transportni sloj formira i segmente koji su primljeni iz visih slojeva, npr Aplikativnog sloja (Application Layer). Segmentacija je proces podele dugih poruka u kraće poruke kako bi se lakše prenele preko nizih slojeva u modelu.

OSI model definiše pet klasa transportnih protokola za povezivnje od klase 0 (koja je takođe poznata i kao TP0 i ima najslabije karakteristike) do klase 4 (TP4, koja je dizajnirana za manje pouzdane mreže, slične Internetu). Klasa 0 (TP0) nema mogućnost oporavka od greške i bila je dizajnirana za korišćenje na mrežnim slojevima koji pružaju konekciju bez gresaka. Klasa 4 (TP4) je najbliža TCP, iako TCP sadrži neke funkcije koje se u OSI modelu dodeljuju visim slojevima. Takođe, sve klase u OSI modelu omogućavaju brzu upotrebu podataka i očuvanje granica podataka ## [lazarc] [prevod] ##. Detalje karakteristika svih klasa prikazane su u sledećoj tabeli:

## [lazarc] tabela sa interneta ## [lazarc] prebaci tabelu na srpski

### 3.1.5 Sloj 5: Sloj sesije (Session Layer)

Sloj sesije kontroliše dijaloge (veze) između uređaja. Uspostavlja, upravlja i uklanja veze između lokalnih i udaljenih aplikacija. Obezbeđuje funkcije Full-Duplex, Half-Duplex ili Simplex i uspostavlja procedure Checkpoint-a, prekida ili ponovnog pokretanja procedura. OSI model je učinio ovaj sloj odgovornim za dobro završavanje sesija, što je osobina TCP (Transmission Control Protocol), i takođe proveru sesija i oporavak, što se obično ne koristi u Internet Protocol Suite. Sloj sesije se obično eksplicitno primenjuje u sredinama aplikacija koje koriste proceduralne pozive na udaljenim uređajima.

### 3.1.6 Sloj 6: Sloj prezentacije (Presentation Layer)

Sloj prezentacije uspostavlja kontekst između dva entiteta aplikativnog sloja, u kom entiteti aplikativnog sloja mogu koristiti različitu sintaksu i semantiku ukoliko sloj prezentacije pruža mapiranje između njih. Ukoliko je dostupno mapiranje, jedinice protokola su enkapsulirane u jedinice sesije i prosledjene na nize slojeve.

Ovaj sloj obezbeđuje nezavisnost od predstavljanja podataka u različitim aplikacijama i mrežnim formatima. Sloj prezentacije pretvara podatke u oblik koji prihvata zadata aplikacija. Ovaj sloj formatira podatke koji se šalju preko mreže. Ponekad se naziva i sintaksni sloj. Takođe, može uključivati i funkcije kompresije.

### 3.1.7 Sloj 7: Aplikativni sloj (Application Layer)

Aplikativni sloj je OSI sloj najblizi krajnjem korisniku, sto znaci da i OSI aplikativni sloj i korisnik interaguju direktno sa aplikacijom. Ovaj sloj komunicira sa softverskom aplikacijom koja sadrzi komponentu za komunikaciju. Takve aplikacije ne spadaju u okvir OSI modela. Funkcije u aplikativnom sloju obicno ukljucuju identifikovanje partnera u komunikaciji, odredjivanje dostupnosti resursa i sinhronizaciju komunikacije. Prilikom identifikacije uređaja za komunikaciju, aplikativni sloj se razlikuje od samih aplikacija. Na primer, internet aplikacija (web strana) moze imati dva entiteta - dve aplikacije: jedna koja koristi HTTP za komunikaciju sa korisnicima, i drugu za udaljenu bazu podataka koja cuva podatke. Ni jedan od ovih protokola nemaju nista sa podacima koji se cuvaju, to se nalazi samo u aplikaciji. Aplikacijski sloj nema nacina za odredjivanje resursa u mrezi.

## 3.2 Internet protocol suite *TCP/IPmodel*

Internet protocol suite je konceptualni model i set komunikacionih protokola koji se koriste za Internet i slicne kompjuterske mreze. Opste je poznat kao TCP/IP zbog toga sto su osnovni protokoli u ovom modelu TCP (Transmission Control Protocol) i IP (Internet Protocol). Ponekad se naziva i DoD (Department of Defense) model zbog toga sto je razvijanje ovog modela potpomoglo Ministarstvo odbrane SAD-a kroz DARPA.

Internet protokol suite (## [lazarc] [prevod]##) omogucava razmenu podataka izmedju dva uređaja na mrezi, i to specificirajuci kako ce se podaci deliti u pakete, adresirati, prenositi, rutirati, i primati. Ove funkcionalnosti su organizovane u cetirir apstraktna sloja, koja klasifikuju sve protokole s obzirom na to u kom delu povezivanja se nalaze (##[lazarc] [prevod] scope of networking involved ##). Od najnizeg do najviseg, slojevi se dele na Link Layer (Sloj povezivanja), koji sadrzi komunikacione metode za podatke koji ostaju unutar jednog segmenta mreze; Internet Layer (Sloj interneta), koji omogucava povezivanje izmedju nezavisnih mreza; Transport Layer (Sloj prenosa), koji omogucava komunikacione servise za aplikacije na uređajima u mrezi; i Application Layer (Sloj aplikacije), koji omogucava servise korisnicima i sistemskim aplikacijama.

Tehnicki sandardi koji specificiraju Internet protocol suite (##[lazarc] [prevod]##) i mnogi od protokola koji cine IPS odrzava Internet Engineering Task Force (IETF). IPS je model koji prethodi OSI modelu, koji je dosta detaljniji i opisuje vise u mreznom sistemu.

Key architectural principles: Princip od kraja do kraja je evoluirao tokom

vremena. Njegov prvobitni izraz je stavio održavanje stanja i sveobuhvatne inteligencije na ivice, i pretpostavio da internet koji je povezivao krajeve nije zadržao nikakvo stanje i koncentrisao se na brzinu i jednostavnost. Potrebe za zaštitnim zidovima (`##[lazar] firewalls ##`), prevodiocima mrežnih adresa, kesiranju veb sadržaja i slično, su izazvale promene u ovom principu.

Princip robusnosti kaže: "Uopšteno, implementacija mora biti konzervativna u ponasanju prilikom slanja i liberalna u svom ponasanju prilikom prijema. Sto znači, da mora biti oprezna pri slanju dobro formiranih paketa (`##[lazar] datagrams ##`), ali mora prihvatiti bilo koji paket koji može protumacirati. (na primer, ne primećivati tehničke greske gde nije poznato šta ih uzrokuje.)". Drugi deo principa je gotovo jednako važan: softver na drugim uređajima može sadržati razlike koji čine nerazumnim da se iskoriste legalne, ali nejasne karakteristike protokola".

Enkapsulacija se koristi za obezbeđivanje apstrakcije protokola i usluga. Enkapsulacija je obično uskladjena sa podelom unutar protokola na slojeve funkcionalnosti. Uopšteno, aplikacija (najviši nivo modela) koristi skup protokola za slanje svojih podataka kroz slojeve. Podaci se dalje enkapsuliraju na svakom sloju.

Rani dokumenti o ovom protokolu, govore o četvoroslojevnom protokolu. Sto je u upotrebi i danas. I oni su unutar protokola korišćeni u istom redosledu u kom će i ovde biti navedeni.

- Aplikativni sloj (Application layer) Aplikativni sloj je opseg unutar kog aplikacije kreiraju korisničke podatke i prenose ove podatke drugim aplikacijama na istom ili drugom uređaju (hostu). Aplikacije ili procesi, koriste usluge koje pružaju donji slojevi, posebno transportni sloj koji obezbeđuje pouzdane ili nepouz dane veze ka drugim procesima. Komunikacione partnere karakteriše arhitektura aplikacije, kao što su model klijent-server i umrežavanje ravnopravnih korisnika. Ovo je sloj u kome su svi protokoli višeg nivoa, kao što su SMTP, FTP, SSH, HTTP, i td. Proces se adresiraju preko portova koji u sustini predstavljaju usluge.
- Transportni sloj (Transport layer) Transportni sloj obavlja komunikacije između domaćina, ili domaćina na istim ili različitim uređajima (hostovima) i na lokalnoj mreži ili udaljenim mrežama razdvojenim od rutera. Ovaj sloj obezbeđuje kanal za komunikacione potrebe aplikacija. UDP je osnovni protokol transportnog sloja koji pruža nepouzdanu uslugu sa paketima. Protokol za kontrolu prenosa (TCP) omogućava kontrolu protoka, uspostavljanje veze i pouzdan prenos podataka.
- Internet sloj (Internet layer) Internet sloj razmenjuje pakete preko

mreze. Ovaj sloj obezbeđuje uniforman mrežni interfejs koji skriva stvarnu topologiju (raspored) osnovnih mrežnih veza. Zbog toga se naziva i slojem koji uspostavlja rad na mreži. Zaista, ovaj sloj definiše i uspostavlja internet. Ovaj sloj definiše strukture adresiranja i usmeravanja koje se koriste za pakete TCP/IP protokola. Primarni protokol u ovom opsegu je Internet protokol, koji definiše IP adrese. Njegova sledeća funkcija u usmeravanju je da prenosi pakete na sledeći IP ruter koji ima vezu sa mrežom blizu kranjem odredistu podataka.

- Sloj veze (Link layer) Sloj veze definiše metode umrežavanja u okviru lokalne mrežne veze na kojoj uređaji (hostovi) komuniciraju bez rutera u međjukomunikaciji. Ovaj sloj uključuje protokole koji se koriste za opisivanje topologije lokalne mreže i potrebnih interfejsa da bi se završio prenos paketa sa Internet sloja na ostale uređaje.

Slojevi protokola blizu vrha su logično blizu korisničkoj aplikaciji, dok su oni blizu dnu logički blizu fizičkom prenosu podataka. Pregled slojeva u smislu pružanja i korišćenja usluge je metoda aplikacije koja izoluje gornje slojeve protokola od detalja kao što je prenos bitova, detekcije lošeg prenosa, na primer, dok su niži slojevi izolovani od detalja aplikacije i principa rada aplikacije.

# Glava 4

## Operativni sistem

### 4.1 FreeRTOS

FreeRTOS je kernel operativnog sistema koji radi u realnom vremenu, i to za namenske sisteme, i moze se koristiti na preko 35 mikrokontrolera.

(Wiki) Implementacija: FreeRTOS je dizajniran tako da bude mali i jednostavan. Kernel (srce operativnog sistema) se sastoji od samo 3 fajla, i pisan je u C programskom jeziku. Kako bi se kod napravi da bude citljiv, lako portabilan, i kako bi se lako održavao projekat, pisan je uglavnom u C programskom jeziku, sa izuzetkom da su neke funkcionalnosti napisane u assembleru, gde je to bilo potrebno, i to uglavnom rutine u Scheduler-u (Rasporedjivacu???) `##[lazar] [prevod] ##`) koje su specifične za samu arhitekturu.

FreeRTOS omogućava koriscenje metoda za stvaranje vise programskih niti, ili Taskova, stvaranje mehanizama za Sinhronizaciju niti, Mutexa, Semafora i softverskih tajmera. Takodje, postoje mogucnosti koriscenja FreeRTOS-a i za aplikacije niske potrosnje. Aplikacije koje se koriste FreeRTOS mogu biti kompletno staticki alocirane. Alternativno RTOS objekti mogu dinamički alocirane sa 5 sema alokacije memorije i one cine:

- samo alocirati;
- alocirati i osloboditi sa jednostavnim, brzim algoritmom;
- kompleksnija ali brza alokacija i oslobadjanje uz algoritam spajanja susednih memorijskih blokova;
- alternativa za jos kompleksiniju semu koja ukljucuje spajanje susednih memorijskih blokova koja omogućava da hip (HEAP) bude podeljen na vise memorijskih delova;

- i na kraju C biblioteka za alociranje i oslobađanje sa zaštitom međusobnog isključivanja.

Unutar FreeRTOS-a ne postoji ni jedan od složenijih svojstava operativnih sistema koji se uobicaeno mogu naci u operativnim sistemima poput Linux-a ili Microsoft Windows-a, kao sto su drajveru uredjaja, napredno upravljanje memorijom, korisnicki nalozi, i umrezavanje. Akcenat ovog operativnog sistema je na kompaktnosti i brzini izvršavanja. O FreeRTOS-u se moze misliti kao o "biblioteci niti" vise nego kao o "operativnom sistemu". (## [lazarc] , although command line interface and POSIX-like I/O abstraction add-ons are available ##)

FreeRTOS implementira vise niti tako sto postoji jedan program koji poziva metode niti u jednakim kratkim vremenskim intervalima. Metoda promene niti zavisi od prioriteta niti i ukljucuje round-robin semu promene niti. Uobicajen interval promene je do 1/1000 sekunde do 1/100 sekunde, i to kroz prekid hardverskog tajmera, ali interval promene se cesto menja tako da zadovolji potrebe specificne aplikacije.

FreeRTOS Documentation: FreeRTOS je idealno sklopljen za duboke namenske aplikacije u realnom vremenu koje koriste mikrokontrolere ili male mikroprocesore. Ovak nacin projektovanja aplikacija ukljucuje kombinaciju kako strogih zahteva za realnim vremenom u aplikaciji, tako i manje strogih.

Strogi zahtevi za aplikacijama realnog vremena su oni u kojima postoji vremenski rok u izvršavanju, i ako se taj rok probije, doci ce do apsolutnog pada funkcionalnosti sistema. Na primer, airbag u kolima ima potencijal da napravi vise stete nego dobrog ukoliko je odziv sistema samo malo sporiji nego sto treba.

FreeRTOS je kernel realnog vremena (ili rasporedjivac(##[lazarc] [prevod]##) realnog vremena) na koji se nadograđuje aplikacija tako da ispuni stroge zahteve za realnim vremenom aplikacije. To dozvoljava da aplikacija bude organizovana kao kolekcija nezavisnih programskih niti. Na procesoru koji ima samo jedno jezgro, samo jedna programska nit se moze izvršavati u jednom trenutku. Kernel odlucuje koja nit se izvršava tako sto odredjuje prioritet koji se dodeljuje svakoj niti. U najjednostavnijem slucaju, dizajner aplikacije moze odrediti vise prioritete nitima koje implementiraju stroge zahteve za realnim vremenom, a nize prioritete onim nitima koje nemaju tako stroge zahteve za izvršavanjem. Ovim bi se osiguralo da niti koje imaju strozije zahteve, imaju prioritete izvršavanja i pristupa resursima nad ostalim nitima, ali odluke za dodelu izvršavanja nisu uvek tako jednostavne.

Napomena: Unutar FreeRTOS-a se programska nit naziva "task". Tako da ce se u daljem tekstu i koristiti naziv Task za programsku nit.

U projektovanju aplikacija za namenske sistema postoji ustaljena praksa

projektovanja aplikacija koja ne zahteva koriscenje kernela za realno vreme, i ove tehnike mogu dati bolje resenje problema. Mada, u kompleksnijim slucajevima, verovatnije je koriscenje kernela za aplikacije u realnom vremenu, i takodje moze biti kombinacija koriscenja kernela, i drugih tehnika projektovanja aplikacije.

Kao sto je vec opisani, prioriteti taskova mogu pomoci da se osigura da aplikacija ispuni sve zahteve, ali kernel moze doneti i neke manje ocigledne beneficije. Neke od njih su navedene ispod:

- **Skracivanje informacija o vremenskom rasporedu (Abstracting away timing information):** Kernel je odgovoran za vreme izvršavanja i dodeljuje API kojim se unutar aplikacije moze upravljati vremenom. Ovim se omogucava jednostavnija strukturiranost koda, i ukupna velicina koda je manja.
- **Odrzavanje/Prosirivanje (Maintainability/Extensibility):** Uskracivanjem informacija o vremenskom rasporedu rezultuje u manjim zavisnostima izmedju modula, i dozvoljava aplikaciji da evoluiru u kontrolisanom i predvidjenom nacinu. Takodje, kernel je odgovoran za rasporedjivanje vremena, tako da performanse aplikacije manje mogu biti promenjene u hardveru na kome se pokrecu.
- **Modularnost (Modularity):** Taskovi su nezavisni moduli, pri cemu svaki od njih mora imati dobro definisanu svrhu.
- **Timski razvoj (Team development):** Taskovi bi trebalo da imaju dobro definisane interfejse, kako bi se lakse razvijali u timovima.
- **Lakse testiranje (Easier testing):** Ako su taskovi dobro definisani kao nezavisni moduli sa cistim interfejsima, mogu biti testirani nezavisno.
- **Ponovno koriscenje koda (Code reuse):** Veca modularnost sa vecom nezavisnoscu koda koji se moze ponovo koristiti sa manje ulozenog truda.
- **Poboljsana efikasnost (Improved efficiency):** Koriscenjem kernela softver se u potpunosti moze prebaciti na opkretanje dogadjajima (event driven programming), i time bi se uštedelo procesorsko vreme koje se trosi na poliranje dogadjaja koji se ne dogadjaju. Kod se pokrece samo ukoliko postoji nesto sto je potrebno uraditi.

Protiv poboljsane efikasnosti stoji to da je potrebno pocesuirati RTOS prekid, i promeniti izvršavanje sa jednog taska na drugi. Kako god, i

aplikacije koje ne koriste RTOS normalno uključuju neku formu prekida.

- **Idle time (##[lazarc] [prevod]##):** Idle task je task koji se automatski kreira prilikom startovanja Rasporedjivaca (##[lazarc] [prevod]##). I izvršava se kad nema taskova unutar aplikacije koji bi se izvršavali. Ovaj task se može koristiti za merenje procesorske moci koja se troši, za izvršavanje provera u pozadini, ili da jednostavno pokrene režim smanjene potrošnje u sistemu.
- **Upravljanje snagom (Power management):** Efikasnost koja se dobija koriscenjem RTOS-a dozvoljava procesoru da provede više vremena u režimu smanjenje potrošnje. Potrošnja se može značajno smanjiti time što procesor odlazi u režim smanjenje potrošnje kad god je pokrenut Idle task. FreeRTOS takodje ima i specijalni tick-less mod, u kome procesor odlazi u režim smanjene potrošnje na duže.
- **Fleksibilno upravljanje prekidima (Flexible interrupt handling):** Upravljanje prekidima se može držati veoma kratko tako što se odlaze obrada bilo kog taska koji je kreirao sam dizajner, ili taska unutar FreeRTOS-a.
- **Razliciti zahtevi za obradom (Mixed processing requirements):** Jednostavni oblici dizajniranja programa mogu se postići mesanjem periodičnog, kontinualnog i procesiranja pokretanog događajima. Pored toga, ispunjavanje strogih i manje strogih zahteva za realnim vremenom u aplikacijama može se postići izborom odgovarajućih taskova i prioriteta prekida.

## 4.2 lwIP

lwIP (light-weight IP) je implementacija TCP/IP komplet-a (## [lazarc] [prevod] suite ##) je originalno napisao Adam Dunkels u Computer and Networks Architectures (CNA) laboratoriji na SHvetskom institutu za kompjuterske nauke (Swedish Institute of Computer Science) ali ga sad aktivno razvija tim inženjera sirom sveta kojim rukovodi Kieran Mansley.

lwIP je open-source projekat koji je besplatan za preuzimanje i koriscenje (pod BSD licencom), pisan u C programskom jeziku i može se preuzeti sa internet stranice tima koji ga razvija.

Fokus lwIP implementacije TCP/IP je da se smanji koriscenje RAM memorije i da se i dalje dobija potpuna funkcionalnost TCP. Ovim lwIP postaje



interesantan za koriscenje u namenskim sistemima koji raspolazu sa RAM memorijom od nekoliko desetina kilobajta kB i prostorom od oko 40 kilobajta u ROM memoriji.

Od kada je prvi put objavljen, lwIP izaziva dosta interesovanja, i danas se koristi u dosta komercijalnih projekata. lwIP je do sad iskoriscen na mnogim platformama i operativnim sistemima, i moze se koristiti bez i sa operativnim sistemom. U ovoj implementaciji, lwIP se koristi u okviru FreeRTOS operativnog sistema, kao jedan njegov deo.

LwIP je veoma modularan i ima podrsku za dosta protokola, od kojih vecina moze da se ukloni za manju velicinu koda.

- *Mrežni protokoli i protokoli veze: (Link and network protocols)*
  - **ARP**: protokol veze koji se koristi za prevod prirodne hardver adrese ("MAC adresa") u IP adresu
  - **IPv4**: dominantni mrežni protkol koji se koristi danas, posebno za Internet
  - **IPv6**: naslednik IPv4, koji, narocito, prosiruje velicinu IP adrese na 128 bita
  - **ICMP**: kontrolni protokol za IP
  - **IGMP**: protokol za urpravljanje grupa unutar IP-a
- *Transportni protokoli: (Transport protocols)*
  - **UDP**: protokol bez prikljucka, i bez mehanizma pouzdanosti
  - **TCP**: protokol orjentisan ka konekciji, za kontinualni tok podataka (streaming")
- *Protokoli visokog nivoa: (High-level protocols)*
  - **DHCP**: dobijanje IP adrese sa podrskom servera
  - **AUTOIP**: dobijanje IP adrese bez podrške servera
  - **SNMP**: koriscen za nadgledanje stanja mreze
  - **PPP**: koriscen za stvaranje direktne konekcije izmedju dva cvora na mrezi

IPv4: (##[lazarc] opisati u delu za softversku implementaciju ##)

lwIP pruza tri API-a (Application Program's Interface) za programe koji komuniciraju sa TCP/IP kodom:

- low-level čore"/čallback"ili "raw" API

- dva API-a viseg nivoa (sekvencijalni API-i):
  - netconn API
  - socket API

Sekvencijalni API pruža način za obično, sekvencijalno programiranje koje koristi lwIP stek (`##[lazarc] stack ##`). Model izvršavanja je baziran na blokirajućoj otvori-pročitaj-upisi-zatvori paradigmi. Kako je TCP/IP stek baziran na događajima, TCP/IP kod i aplikativni program, moraju da se pozivaju sa različitim kontekstima izvršavanja, u različitim nitima.

Prilikom mesanja sekvencijalnog i širokog API-a u programima, treba biti pažljiv. Funkcije koje pripadaju nesekvencijalnom API-u u stvari mogu biti pozvane iz glavne `tcpip_thread` niti. Takođe, registrovanje programski rutina (ili inicijalizovanje delova u lwIP) mora biti odradjeno unutar tog konteksta (na primer, u vreme startovanja aplikacije u `tcpip_init_callback` rutini ili u vreme izvršavanja unutar `tcpip_callback` rutine).

Jos neke činjenice o API-ima koje utiču na korišćenje lwIP steka:

- **netconn- i raw-API su samo unutar lwIP-a :** kod koji koristi ovaj API se ne može koristiti u drugim stekovima koji imaju iste mogućnosti kao lwIP (na primer uIP i td.)
- **socket API** je u suprotnosti sa gore navedenom stavkom, napravljen je tako da je kompatibilan i može se koristiti u drugim stekovima.
- **socket- i netconn-API** su sekvencijalni API-i koji zahtevaju programske niti (jedna nit je za aplikaciju koja koristi API, jedna nit upravlja tajmerima unutar steka, paketima koji dolaze, i td.)
- **raw API** koristi mehanizam povratnih rutina (na primer. aplikacija poziva rutinu kada dodje novi podatak). Ukoliko se koristi u programu koji radi na sekvencijalni način, može biti teže korišćenje.
- **raw API** daje bolje performanse kako ne zahteva promenu izvršavanja programskih niti.
- **raw API i netconn API** podržavaju zero-copy <sup>1</sup> kako za TX tako i za RX. (kako za predaju, tako i za prijem)

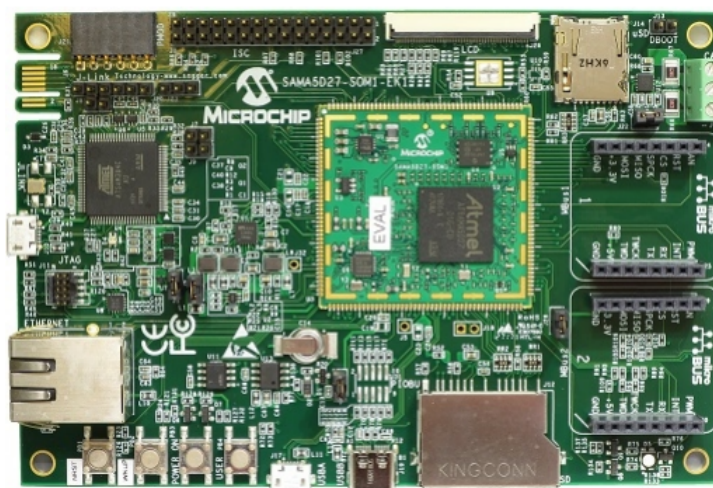
---

<sup>1</sup>predstavlja operaciju pri kojoj procesor ne vrši kopiranje podataka iz jedne memorijske oblasti u drugu. Ovo se često koristi kako bi se sačuvali ciklusi procesora i propusnog opsega memorije prilikom prenosa kontinualnih podataka (datoteka, fajlova) preko mreže.

## Glava 5

# Hardverska implementacija

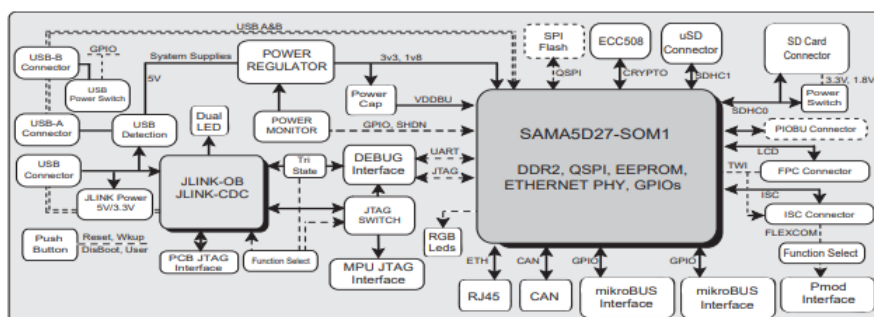
Razvojno okruženje koje je korišćeno za hardversku implementaciju je razvojna ploča SAMA5D27-SOM1-EK1 proizvođača Microchip. Na ploči se nalazi SAMA5D27 SOM (System on Module) modul koji je ključan za implementaciju. Na modulu se nalazi SAMA5D27-D1G-CU SIP (System in Package) koji sadrži 1 Gbit DDR2 SDRAM memorije. Modul nudi puzdanu i niskobudžetnu platformu za razvoj namenskih računarskih sistema koji će na kraju i završiti u finalnoj proizvodnji, kao i malu formu, dopunjenu sa velikim brojem interfejsa koji se mogu koristiti u delu projektovanja krajnjeg sistema.



Slika 5.1: Razvojna ploča SAMA5D27-SOM1-EK1 (pogled odozgo)

SOM je potpuno opremljen industrijski sertifikovan kompjuter dizajniran za integraciju korisničke aplikacije. SOM modul je namenski napravljen kao

mala hardverska platforma opremljena širokim spektrom modula za brzo povezivanje kako bi podržali projektovanje podrške za razne IoT (Internet of Things) aplikacije, prenosnih uređaja, ali i aplikacija u industrijske svrhe. SOM integriše 1Gbit DDR2 SDRAM i QSPI memoriju kao 10/100 Mbit Ethernet interfejs. Takodje, SOM poseduje i 128 GPIO pinova koji obezbeđuju pristup SOM-a za različite upotrebe. Svi GPIO pinovi su nezavisni, i mogu se konfigurisati kao ulazi ili izlazi, sa ili bez PULL-UP otpornika. Razvojna ploča poseduje i širok spektar periferija, kao i korisnički interfejs i način za proširenje funkcionalnosti, uključujući i dva microBUS Click interfejsa firme Mikroelektronika kojim se dobija mogućnost za proširenje funkcionalnosti svim modulima koje ova firma nudi u svom asortimanu.

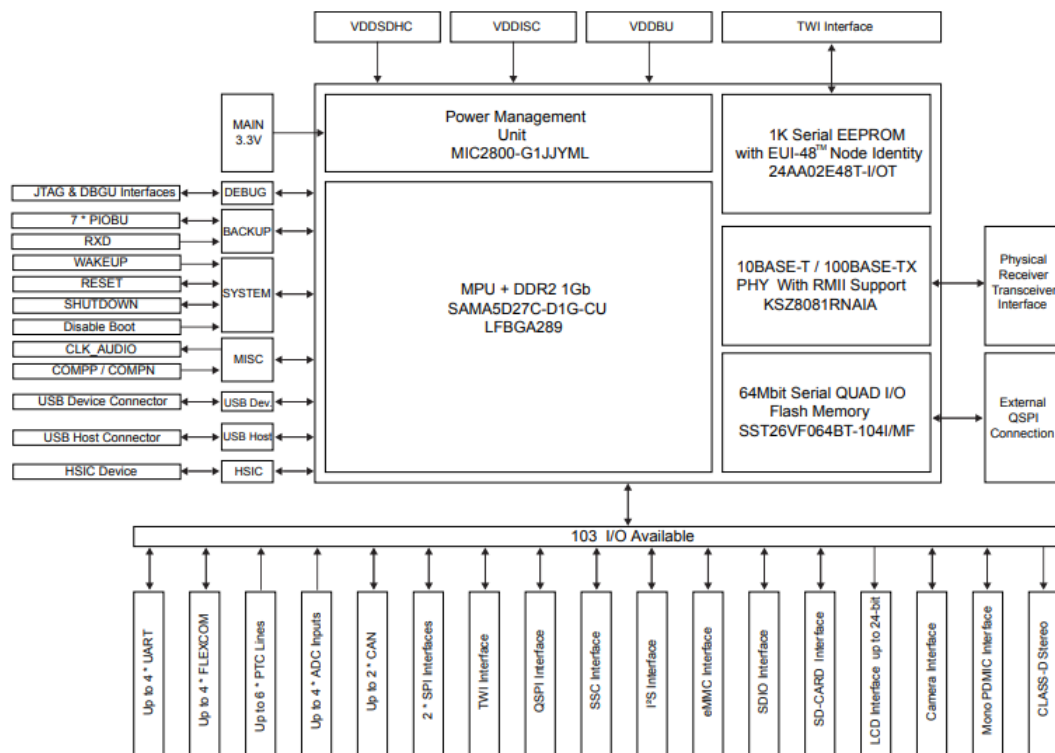


Slika 5.2: Moduli - Razvojna ploča SAMA5D27-SOM1-EK1

Na samom SAMA5D27-SOM1 postoje i:

- Ultra mali SIP (SAMA5D27-D1G-CU) koji sadrži štedljivi SAMA5D27 Arm Cortex A5 procesor i 1Gbit DDR2 SDRAM memoriju
- SST26VF064 64 Mb QSPI Flash memoriju
- 24AA02E48 2 Kb serijski E2PROM (Electrically Erased Programmed Read Only Memory) sa programiranom EUi identifikacijom pristupa
- MIC2800 cip za kontrolu napajanja
- KSZ8081RNA Ethernet Phy 10/100 MHz RMII

Razvojno okruženje SAMA5D27-SOM1-EK1 je veoma moćno i može se koristiti za širok spektar aplikacija. Najčešće je izbor za razvoj aplikativnog softvera u namenskim računarskim sistemima, uz korišćenje Embedded Linux operativnog sistema, i s tim ciljem se i bira. Za ovu implementaciju nije korišćen u tom smislu, već je korišćen sa drugim operativnim sistemom



Slika 5.3: Moduli - SAMA5D27-SOM1

kojim je moguće istaći neke druge njegove karakteristike. Ali ono što je najviše interesantno za ovu primenu je postojanje TSU (Timestamping unit) za harversku podršku PTP-a. I to u sklopu periferije za povezivanje preko Ethernet-a, čime je omogućeno prepoznavanje PTP poruka koje dolaze na interfejs.

## 5.1 Time stamping unit - TSU

IEEE-1588 je standard za preciznu sinhronizaciju vremena u lokalnim mrežama. Radi se o razmeni preciznog vremena između dva uređaja u lokalnoj mreži. PTP poruke se mogu prenositi preko IEEE802.3/Ethernet, IPv4 ili IPv6 protokola kako je to već opisano u odeljcima pre. Periferija unutar razvojnog okruženja GMAC označava tačku vremenske oznake poruke (na početku slanja i na kraju slanja) poruke. IEEE 802.1AS je podskup IEEE 1588 standarda. Jedina razlika je u adresi koja služi za slanje PTP poruka svim uređajima u lokalnoj mreži (Multicast). GMAC periferija je dizajnirana tako da prepoznaje poruke koje su ključne za PTP protokol.

Kao što je već navedeno, sinhronizacija između dva uređaja se izvodi u dva stadijuma, određivanje razlike (offset) između dva sata (na strani Master i Slave uređaja), nakon čega se šalje tačno kašnjenje na linijama za prenos podataka, čime se tačno sinhronizuju dva sata. Hardverski moduli koji pomažu u ovoj razmeni poruka određuju tačno vreme kada je poruka stigla, i kada je poruka poslata. Što je ključno za određivanje vremena kojim se izračunavaju razlika i kašnjenje. Pojava ovih poruka uzrokuje prijavljivanje hardverskih prekida, tako da je moguće operisati sa vremenima koja se dobijaju tako da se dobija fina sinhronizacija vremena dva uređaja. Podrška ovom protokolu u hardveru se ogleda u postojanju TSU (Timestamping unit) periferije koja se sastoji od tajmera i registara u koje se smeštaju tačna vremena u trenucima kada stignu ili odu poruke koje su ključne za dobijanje tačnog vremena. Prekid se prijavljuje kada se ovi registri osveže vremenom slanja ili primanja poruke bitne za PTP protokol. Tajmer je implementiran kao 94-bitni brojač, u kome viših 48 bita broje sekunde, narednih 30 bita nanosekunde, i preostalih 16 bita broje vreme ispod nanosekunda. Nižih 46 bita se prevrte (roll-over) kad se izbroji tačno jedna sekunda. Takođe, prijavljuje se hardverski prekid nakon sto se izbroji jedna sekunda. Vrednosti tajmera, kao i parametri za njegovo (##[lazar] CHECK! ##) koji je glavni takt unutar procesora, i njime se takođe može manipulirati. Promene podešavanja tajmera unutar TSU su od velike važnosti za tačnu sinhronizuju dva uređaja. Ova periferija najviše utiče na tačno vreme koje je potrebno sinhronizovati, i takođe daje informacije o tačnom vremenu na uređaju koji se koristi. Više detalja o samoj implementaciji i korišćenju ove periferije biće dato u delu softverske implementacije.

## Glava 6

# Softverska implementacija

Glava 7

Zaključak



# Glava 8

## CHECK

### 8.1 Precision Time Protocol

Precision Time Protocol (PTP) (prevod: protokol preciznog vremena) je protokol koriscen za sinhronizaciju satova preko kompjuterske mreze. U lokalnoj kompjuterskoj mrezi (local area connection), postize se precisnost sata i u rangu ispod mikrosekunde, sto ga cini pogodnim za merenja i kontrolne sisteme.

PTP je originalno definisan u IEEE 1588-2002 standardu, i zvanično nazvan "Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" i objavljen 2002 godine. U 2008 godini, IEEE 1588-2002 je objavljen kao preradjeni standard, poznat i kao PTP Version 2, sa poboljšanom tačnošću, preciznošću i robusnošću, međutim nije kompatibilan sa prethodnom verzijom koja je objavljena 2002 godine.

"IEEE 1588 je dizajniran da popuni prazninu koja nije dobro obradjena ni jednim od dva dominantna protokola, NTP i GPS. IEEE 1588 je dizajniran za lokalne sisteme u kojima je potrebna preciznost izvan one koja je dostupna NTP protokolom. Takođe je dizajniran za aplikacije koje se ne mogu nositi sa cenom GPS prijemnika na svakoj uredjaju, ili sa onima u kojima nije moguće dobijanje GPS signala.»> "IEEE 1588 is designed to fill a niche not well served by either of the two dominant protocols, NTP and GPS. IEEE 1588 is designed for local systems requiring accuracies beyond those attainable using NTP. It is also designed for applications that cannot bear the cost of a GPS receiver at each node, or for which GPS signals are inaccessible.-Eidson, John C. (April 2006). Measurement, Control and Communication Using IEEE 1588. Springer. ISBN 1-84628-250-0.

Arhitektura: IEEE 1588 standard opisuje hijerarhijsku master-slave arhitekturu za distribuciju vremena. Pod ovom arhitekturom podrazumeva se

distribucija vremena u sistemu koji se sastoji od jednog ili vise komunikacionih medijuma (segmenata koji su povezani na mrezu), i jednog ili vise izvora tacnog vremena. #Obicni uredjaj# Izvor "obicnog" vremena ("ordinary clock") je uredjaj sa jednim pristupom mrezi i ima jednu od dve uloge, ili je izvor (master) tacnog vremena, ili ceka na tacno vreme (slave) u komunikaciji na mrezi. #Sporadni uredjaj# Granicni sat (Boundary clock) ima vise pristupa, na razlicite mreze, i moze precizno sinhronizovati jedan segment mreze na drugi. Master sinhronizacije se bira za svaki segment mreze u sistemu. #Glavni uredjaj# Referentno vreme koje se uzima za izvor sinhronizacionog sata se zove Grandmaster clock. Grandmaster dostavlja sinhronizacione informacije do svih uredjaja koji su povezani na istu mrezu sa njim. Ukoliko se u nekom delu mreze nalazi Boundary clock on prosledjuje tacno vreme ka ostalim uredjajima koji su direktno na njega povezani.

# mora slika ovde kako izgleda arhitektura tacno

Simplifikovano, PTP sistem se sastoji od Ordinary clocks #Obicnog uredjaja# povezanih na jednostavnu mrezu, i bez Boundary clocks #Sporadnih uredjaja#. Grandmaster se bira, i svi ostali uredjaji se direktno sinhronisu na njega.

IEEE 1588-2008 predstavlja Clock koji je povezan sa mreznom opremom koja prenosi PTP poruke. Transparent clock #Transparentni uredjaj# modifikuje PTP poruke koje prolaze kroz uredjaj. Vremenski pecati (TIMESTAMPS) u porukama su modifikovani tako da se uzme u obzir i vreme za koje poruka prolazi kroz dodatne uredjaje u komunikaciji. Ova sema komunikacije povecava distribuciju preciznosti tako sto se kompenzuje promenljivost dostave podataka preko mreze.

PTP tipicno koristi EPOCH vreme, standardno vreme za UNIX sisteme (1 Januar 1970 kao pocetak racunanja vremena). Dok je UNIX vreme bazirano na Univerzalnom vremenu UTC, i mora da postoji sekunda preskoka (#Ovo dodati, mozda u uvod#), PTP je baziran na Medjunarodnom Atomskom Vremenu (TAI - International Atomic Time). PTP Grandmaster daje trenutnu razliku izmedju UTC i TAI, kako bi UTC vreme moglo da se izracuna od primljenog PTP vremena.

# mora slika da se vidi razlika izmedju UTC i PTP, TAI

Detalji protokola: Sinhronizacija i obrada u PTP sistemu se postize razmenom poruka preko komunikacionog medijuma. Do sad, PTP standard propisuje samo ove tipove poruka.

- Sync, Follow\_Up, Delay\_Req i Delay\_Resp poruke se koriste u Ordinary i Boundary uredjajima i sluze samo za komunikaciju informacija o vremenu koje se koriste za sinhronizaciju uredjaja na mrezi. - Pdelay\_Req, Pdelay\_Resp i Pdelay\_Res\_Follow\_Up se koriste u Transparent Clock uredjajima da mere kasnjenje kroz uredjaj tako da se moze iskoristiti u kom-

penzaciji vremena u sistemu. Transparent Clock i definicija ovih poruka nisu dostupne u IEEE 1588-2002 standardu. - Announce poruke se koriste i Best master clock algorithm u IEEE 1588-2002 standardu za algoritam odredjivanja najtacnijeg sata na mrezi, i to kako bi se izgradila hijerarhija uredjaja i kako bi se odredio Grandmaster. - Management poruke se koriste u upravljanju mrežom za posmatranje performansi na mrezi, konfiguraciju mreže i održavanje PTP sistema. - Signalne poruke se koriste u komunikaciji između uredjaja koje nisu vremenski kritične. Signalne poruke su uvedene u IEEE 1588-2002 standard.

Poruke se karakterizuju kao Event i General, odnosno poruke događaja i opšte poruke. Event poruke su vremenski kritične i to u preciznosti predaje i prijema preciznosti vremenskih pečata (TIMESTAMPS) i direktno uticu na distribuciju preciznosti vremena. (JOS JEDNOM POGLEDAJ OVAJ PREVOD). Sync, Delay\_Req, Pdelay\_Req i Pdelay\_resp su poruke događaja. Opšte poruke su ubacjene jedinice protokola, zato što podaci u ovim porukama su od značaja za PTP, ali njihovi vremenski pečati za predaju i prijem nisu. Announce, Follow\_Up, Delay\_Resp, Pdelay\_Resp\_Follow\_Up, Management i Signalne poruke su opšte poruke.

Prenos poruka: PTP poruke mogu da koriste UDP (User datagram portocol) preko Internet protokola (UDP/IP) za prenos poruka. IEEE 1588-2002, koristi samo IPv4 prenos, ali je ovo prošireno da uključuje i IPv6 u IEEE 1588-2008 standardu. U IEEE 1588-2002, sve PTP poruke se šalju u Multicast (modulu objavljivanja na mreži) (#pogledaj opet ovaj prevod#), dok se u IEEE 1588-2008 je uveo kao opciju.

## 8.2 Algoritam najboljeg sata *Bestmasterclockalgorithm*

BMC algoritam obavlja deljenu selekciju najboljeg kandidata za tačno vreme prema sledecim karakteristikama:

- Identifikator: Univerzalni jedinstveni identifikator za sat. Tipicno je baziran na MAC adresi uredjaja.
- Kvalitet: Obe verzije IEEE 1588 standarda pokusavaju da kvantifikuju kvalitet sata na osnovu očekivanih devijacija u vremenu, tehnologije koja je koriscena za implemntaciju vremena ili lokacije u hijerarhiji satova, u semi kvaliteta satova (clock stratum scheme).

- Prioritet: Administrativno dodeljen prioritetni znak koji BMC koristi kako bi sto bolje odredio Grandmaster u PTP domenu. Dok je IEEE 1588-2002 standard imao samo jednu logicku promenljivu kako bi odredio prioritet, IEEE 1588-2008 ima dva 8-bitna polja prioriteta.

- Varijansa: Procena stabilnosti sata zasnovana na zapazanju njegovog ucinka prema PTP refernci.

IEEE 1588 koristi hijerarhijski algoritam selekcije zasnovan na sledecim osobinama, u naznacenom redosledu:

- Prioritet 1: korisnik moze dodeliti specifican staticki dizajniran prioritet svakom satu pre svega odredjuci prioritet medju njima. Manje vrednosti prioriteta oznacavaju veci prioritet.
- Klasa: Svaki sat je clan odredjene klase, svaka klasa dobija svoj prioritet.
- Preciznost: Preciznost izmedju sata i UTC, u nanosekundama.
- Varijansa: Varijabilnost sata.
- Prioritet 2: Definisan prioritet, definisuci redosled rezervne kopije u slucaju da drugi kriterijumi nisu dovoljni. Manje vrednosti prioriteta oznacavaju veci prioritet.
- Jedinstveni identifikator: selekcija zasnovana na MAC adresi se koristi kao metod odlucivanja kada su sve ostale osobine iste.

Svojstva sata se daju u IEEE 1588-2002 porukama za sinhronizaciju (Sync messages) i u IEEE 1588-2008 u porukama za oglasavanje (Announce messages). Trenutni Master clock (## [lazarc] [prevod]##) prenosi sve informacije u rednovnim intervalima. Sat koji sebe smatra boljim od trenutnog Master sata prenosi ove informacije kako bi se pozvali svi uredjaji za pormenu Master sata. Kada trenutni Master prepozna bolji sat, tada Master sat zaustavlja emitovanje poruka za sinhronizaciju (Sync Messages), ili poruke oglasavanja (Announce messages), u zavisnosti od verzije protokola, i bolji sat preuzima ulogu Master sata. BMC algoritam uzima u obzir samo osobine koje su vec poznate, i koje su deklarirali sami satovi, i ne uzima u obzir kvalitet veze na mrezi.

## 8.3 Sinhronizacija (Synchronization)

Koristeci BMC algoritam, PTP bira Master sat za IEEE 1588 domen i za svaki segment mreze unutar tog domena. Satovi odredjuju razliku izmedju njih (offset) i Master-a na mrezi. Neka promenljiva  $t$  predstavlja fizicki vreme. Za dati Slave uredjaj, razlika  $o(t)$  u vremenu  $t$  se definise kao:  $o(t) = s(t) - m(t)$  gde  $s(t)$  predstavlja vreme mereno satom na Slave uredjaju u vremenu  $t$ , dok  $m(t)$  predstavlja vreme mereno satom na Master uredjaju u vremenu  $t$ .

Master uredjaj periodicno objavljuje (Broadcasts) trenutno vreme kao poruku ostalim uredjajima na mrezi. IEEE 1588-2002 protokolom je definisana objava vremena na svaku sekundu. Dok je IEEE 1588-2008 protokolom dozvoljeno i do 10 objava vremena u jednoj sekundi.

## [lazarc] [slika] sync

Svaka objava vremena krece u vremenskom trenutku  $T_1$ , i to Sync porukom koju salje Master uredjaj svim uredjajima u domenu. Uredjaj koji prima ovu poruku pamti vreme  $T'_1$  u kom je primio Sync poruku. Master

može naknadno poslati Follow\_up poruku u kojoj će se nalaziti tačno vreme  $T_1$  u kom je poslata prethodna poruka. Nemaju svi Master uređaji sposobnost da pošalju tačne vremenske oznake unutar Sync poruke. Tek nakon što je prenos završen, oni mogu dobiti tačne vremenske trenutke stizanja Sync poruke iz hardvera za povezivanje na mrežu. Master uređaji sa ovim ograničenjima šalju Follow\_up poruke kako bi preneli vreme  $T_1$ . Master uređaji koji poseduju PTP mogućnosti unutar hardvera za povezivanje mogu ubaciti tačne vremenske oznake unutar Sync poruka, i ne moraju koristiti Follow\_up poruke.

Kako bi se tačno sinhronizovali na Master uređaj, satovi moraju individualno odrediti vreme prenosa poruka kroz medijum za povezivanje. Vreme progresije poruke kroz medijum za povezivanje se radi merenjem vremena koje je potrebno da poruka ode od svakog uređaja do njihovog Mastera u domenu, i da se vrati nazad. Ovu razmenu iniciraju Slave uređaji i pri tome mere vreme progresije poruke  $d$ . Razmena poruka počinje tako što Slave uređaj šalje Delay\_Req poruku u vremenskom trenutku  $T_2$  ka svom Master uređaju. Master uređaj primi ovu poruku, i kao odgovor pošalje tačnu vremensku oznaku kada je primio Delay\_Req poruku. Poruka odgovora Delay\_Resp sadrži tačno vreme  $T'_2$  u kome je primljena poruka Delay\_Req.

Nakon razmene ovih poruka Slave uređaj ima spoznaju o četiri vremenska trenutka  $T_1$ ,  $T'_1$ ,  $T_2$  i  $T'_2$ .

Ukoliko je  $d$  vreme koje je potrebno Sync poruci da prođe kroz medijum za povezivanje, a  $\tilde{o}$  konstantna razlika satova između Master i Slave uređaja, onda je:

$$T'_1 - T_1 = \tilde{o} + d \quad (8.1)$$

i

$$T'_2 - T_2 = -\tilde{o} + d \quad (8.2)$$

Odakle je:

$$\tilde{o} = \frac{1}{2}(T'_1 - T_1 - T'_2 + T_2) \quad (8.3)$$

Sada dva uređaja znaju koliki je ofset  $\tilde{o}$  prilikom prenosa i mogu se ispraviti tako da budu u skladu sa Master uređajem.

Jedna pretpostavka je da se prenos poruka odvija u periodu vremena koji je tako mali, da se razlika može smatrati konstantnom u tom periodu. Još jedna pretpostavka je da je vreme koje je potrebno da poruka stigne od Master do Slave uređaja ista kao i u obrnutom smeru. I na kraju, pretpostavka

je da i Master i Slave uredjaji mogu da precizno mere vremenske trenutke u kojima salju ili primaju poruke. Stepem primene ovih pretpostavki utice na to koliko ce se dobro sinhronizovati dva uredjaja.

## [lazarc] Pogledaj jos da se ubaci sa Teams za Valeo

### **IZ PTP\_CONTINUE\_TO\_MERGE.TXT**

Algoritam odredjivanja najboljeg sata (Best master clock algorithm): BMC algoritam obavlja deljenu selekciju najboljeg kandidata za tacno vreme prema sledecim karakteristikama:

- Identifikator: Univerzalni jedinstveni identifikator za sat. Tipicno je baziran na MAC adresi uredjaja.
- Kvalitet: Obe verzije IEEE 1588 standarda pokusavaju da kvantifikuju kvalitet sata na osnovu ocekivanih devijacija u vremenu, tehnologije koja je koriscena za implemntaciju vremena ili lokacije u hijerarhiji satova, u semi kvaliteta satova (clock stratum scheme).
- Prioritet: Administrativno dodeljen prioritetni znak koji BMC koristi kako bi sto bolje odredio Grandmaster u PTP domenu. Dok je IEEE 1588-2002 standard imao samo jednu logicku promenljivu kako bi odredio prioritet, IEEE 1588-2008 ima dva 8-bitna polja prioriteta.
- Varijansa: Procena stabilnosti sata zasnovana na zapazanju njegovog ucinka prema PTP refernci.

IEEE 1588 koristi hijerarhijski algoritam selekcije zasnovan na sledecim osobinama, u naznacenom redosledu:

- Prioritet 1: korisnik moze dodeliti specifcan staticki dizajniran prioritet svakom satu, pre svega odredjujuci prioritet medju njima. Manje vrednosti prioriteta oznacavaju veci prioritet.
- Klasa: Svaki sat je clan odredjene klase, svaka klasa dobija svoj prioritet.
- Preciznost: Preciznost izmedju sata i UTC, u nanosekundama.
- Varijansa: Varijabilnost sata.
- Prioritet 2: Definisan prioritet, definisuci redosled rezervne kopije u slucaju da drugi kriterijumi nisu dovoljni. Manje vrednosti prioriteta oznacavaju veci prioritet.

- Jedinstveni identifikator: selekcija zasnovana na MAC adresi se koristi kao metod odlucivanja kada su sve ostale osobine iste.

Svojstva sata se daju u IEEE 1588-2002 porukama za sinhronizaciju (Sync messages) i u IEEE 1588-2008 u porukama za oglasavanje (Announce messages). Trenutni Master clock (## [lazar] [prevod]##) prenosi sve informacije u rednovnim intervalima. Sat koji sebe smatra boljim od trenutnog Master sata prenosice ove informacije kako bi se pozvali svi uredjaji za pormenu Master sata. Kada trenutni Master prepozna bolji sat, tada Master sat zaustavlja emitovanje poruka za sinhronizaciju (Sync Messages), ili poruke oglasavanja (Announce messages), u zavisnosti od verzije protokola, i bolji sat preuzima ulogu Master sata. BMC algoritam uzima u obzir samo osobine koje su vec poznate, i koje su deklarirali sami satovi, i ne uzima u obzir kvalitet veze na mrezi.

Sinhronizacija (Synchronization):

Koristeci BMC algoritam, PTP bira Master sat za IEEE 1588 domen i za svaki segment mreze unutar tog domena. Satovi odredjuju razliku izmedju njih (offset) i Master-a na mrezi. Neka promenljiva  $t$  predstavlja fizicki vreme. Za dati Slave uredjaj, razlika  $o(t)$  u vremenu  $t$  se definise kao:

$$o(t) = s(t) - m(t)$$

gde  $s(t)$  predstavlja vreme mereno satom na Slave uredjaju u vremenu  $t$ , dok  $m(t)$  predstavlja vreme mereno satom na Master uredjaju u vremenu  $t$ .

Master uredjaj periodično objavljuje *Broadcasts* trenutno vreme kao poruku ostalim uredjajima na mrezi. IEEE 1588-2002 protokolom je definisana objava vremena na svaku sekundu. Dok je IEEE 1588-2008 protokolom dozvoljeno i do 10 objava vremena u jednoj sekundi.

## [lazar] [slika] sync

Svaka objava vremena kreće u vremenskom trenutku  $T_1$ , i to Sync porukom koju šalje Master uredjaj svim uredjajima u domenu. Uredjaj koji prima ovu poruku pamti vreme  $T'_1$  u kom je primio Sync poruku. Master može naknadno poslati Follow\_up poruku u kojoj će se nalaziti tačno vreme  $T_1$  u kom je poslata prethodna poruka. Nemaju svi Master uredjaji sposobnost da pošalju tačne vremenske oznake unutar Sync poruke. Tek nakon što je prenos završen, oni mogu dobiti tačne vremenske trenutke stizanja Sync poruke iz hardvera za povezivanje na mrežu. Master uredjaji sa ovim ograničenjima šalju Follow\_up poruke kako bi preneli vreme  $T_1$ . Master uredjaji koji poseduju PTP mogućnosti unutar hardvera za povezivanje mogu ubaciti tačne vremenske oznake unutar Sync poruka, i ne moraju koristiti Follow\_up poruke.

Kako bi se tačno sinhronizovali na Master uredjaj, satovi moraju individualno odrediti vreme prenosa poruka kroz medijum za povezivanje. Vreme

progresije poruke kroz medijum za povezivanje se radi merenjem vremena koje je potrebno da poruka ode od svakog uredjaja do njihovog Mastera u domenu, i da se vrati nazad. Ovu razmenu iniciraju Slave uredjaji i pri tome mere vreme progresije poruke  $d$ . Razmena poruka pocinje tako sto Slave uredjaj salje `Delay_Req` poruku u vremenskom trenutku  $T_2$  ka svom Master uredjaju. Master uredjaj primi ovu poruku, i kao odgovor posalje tacnu vremensku oznaku kada je primio `Delay_Req` poruku. Poruka odgovora `Delay_Resp` sadrzi tacno vreme  $T'_2$  u kome je primljena poruka `Delay_Req`.

Nakon razmene ovih poruka Slave uredjaj ima spoznaju o cetiri vremenska trenutka  $T_1$ ,  $T'_1$ ,  $T_2$  i  $T'_2$ .

Ukoliko je  $d$  vreme koje je potrebno Sync poruci da prodje kroz medijum za povezivanje, a  $o$  konstantna razlika satova izmedju Master i Slave uredjaja, onda je:  $T'_1 - T_1 = o + d$  i  $T'_2 - T_2 = -o + d$ .

Odakle je:

$$o = \frac{1}{2}(T'_1 - T_1 - T'_2 + T_2).$$

Sada dva uredjaja znaju koliki je ofset  $o$  prilikom prenosa i mogu se ispraviti tako da budu u skladu sa Master uredjajem.

Jedna pretpostavka je da se prenos poruka odvija u periodu vremena koji je tako mali, da se razlika moze smatrati konstantnom u tom periodu. Jos jedna pretpostavka je da je vreme koje je potrebno da poruka stigne od Master do Slave uredjaja ista kao i u obrnutom smeru. I na kraju, pretpostavka je da i Master i Slave uredjaji mogu da precizno mere vremenske trenutke u kojima salju ili primaju poruke. Stepem primene ovih pretpostavki utice na to koliko ce se dobro sinhronizovati dva uredjaja.

## [lazarc] Pogledaj jos da se ubaci sa Teams za Valeo

## **IZ PTP\_NEXT\_1.TXT**

IEEE 1588 Unplugged – An introduction to IEEE 1588

SYNOPSIS Sinhronizacija postaje neophodna kada su uredjaji koji rade zajedno na odredjenoj udaljenosti moraju raditi u vezi jedni sa drugima. U ovim scenirijima, lokalni sat, ili Master Sat, sinhronizuje sve uredjaje u istom sistemu sa tim satom. Zbog ove potrebe za sinhronizacijom, IEEE 1588 standard je objavljen kao standardni protokol 2002. godine. (##[lazarc] However, prevod ##) Iako su dva sata unutar uredjaja podeseni da rade na istoj frekvenciji, ne postoji garancija da ce ostati sinhronizovani. Upravo zbog ovoga proces sinhronizacije je neprekidan. Nekoliko faktora mogu uticati na to da dva identicna sata izgube sinhronizaciju. Razlozi mogu biti razliciti, kao na primer razlika u temperaturi, starosti uredjaja, kao i frekvenciji na kojoj uredjaji rade, koja moze uticati na kvalitet sinhronizacije. Upravo iz ovih razloga je i nastala potreba za sinhronizacijom uredjaja.

SYNCHRONIZATION IEEE 1588 obezbedjuje sinhronizaciju dva sata u istoj mrezi koja je otporna na greske. Takodje, koristi se veoma mali



opseg frekvencija, procesna moc, ali i podesavanje samog protokola. (##[lazar] There is very little bandwidth consumption, processing power, and setup.##) IEEE 1588 standard postize sve ovo koristeći protokol preciznog vremena (## [lazar] Precision time protocol.##), ili PTP. Vremenski protokol koji se koristi sinhronizuje dva sata na mrezi podesavajući satove ka "najkvalitetnijem" satu. IEEE 1588 definiše ospege vrednosti za standardan set karakteristika satova. Algoritam najboljeg sata, BMC algorithm, odlucuje koji sat na mrezi je "najkvalitetniji", odnosno koji sat na mrezi je najtacniji, i najbolji kako bi se ostali uredjaji sinhronizovali na njega. BMC algoritam onda sinhronizuje sve ostale satove (slave clocks) sa ovim satom na mrezi. Ukoliko se BMC (Best Master Clock) ukloni iz mreze, ili BMC algoritam odluci da taj sat nije vise najtacniji, algoritam redefinise novi "najkvalitetniji" sat, i prilagodi vremena ostalih satova u skladu sa tim. Nije potrebno da se administrator mreze ukljucuje u bilo kom trenutku kako bi se promenio najbolji sat u mrezi, zbog toga sto je ovaj algoritam otporan na greske.

U ovu svrhu koristi se Bidirekciona multikast komunikacija (##[lazar] Bidirectional Multicast Communication ##) od strane slave uredjaja kako bi se sinhronizovali na najbolji sat, IEEE 1588 grandmaster clock. Šync", Sinhronizacioni paket sadrzi vremenski zig od najboljeg sata, koji predstavlja tacno vreme kada je paket poslat sa grandmaster clock-a. "Follow up", Prateci paket takodje moze biti poslat sa grandmaster sata, koji sadrzi vremenski zig Šync"(sinhronizacionog) paketa. Ovakav princip razmene podataka omogucava tacan vremenski zig sinhronizacionog paketa koji je prosledjen sa grandmaster sata. Takodje, postoje slucajevi u kojima se ne otkriva tacno vreme slanja paketa, zbog odredjenih smetnji na mrezi. Ovo je omoguceno kroz Collision detection i Random Back-off mehanizama na EthernetIP komunikaciji. Samo onda kad je paket kompletno poslat, nemoguće je promeniti sadrzaj paketa.

Grandmaster i Slave satovi na mrezi razmenjuju Šync", Sinhronizacione pakete sa jedne na drugu stranu, i postavljaju vremenske zigo ve pri prijemu paketa. Kombinujući razliku u satova, kao i kasnjenje na mrezi, razlika između slanja i prijema sinhronizacionih paketa moze biti izracunata. Koristeći razliku koja je izracunata u ovom slucaju, sat moze biti podesen sa novim vrednostima, i time se moze smanjiti razlika između Master i Slave satova u ovoj mrezi. Razlika između master i slave sinhronizacionih paketa, i obrnuto, implicira da IEEE 1588 standard radi pod pretpostavkom da je propagacija paketa po mrezi simetricna. To je zbog pretpostavke da slave uredjaj moze da odredi i podesi kasnjenje prilikom propagacije paketa po mrezi. Kako bi se kasnjenje na mrezi odredilo, slave kreira delay request", zahtev za odredjivanjem kasnjenja, i postavlja vremenski zig prilikom slanja paketa. Master sat onda postavlja vremenski zig prilikom pristizanja tog zahteva, i vraca ga ka

Slave uredjaju, i to u obliku "delay response" paketa, odgovora za kasnjenjem. Nakon toga, određuje se kasnjenje po linijama na mrezi, i izracunava se iz ovih paketa koji se razmenjuju.

Slanje i primanje sinhronizacionih paketa dozvoljava da Slave uredjaji tacno izmere razliku izmedju lokalnog Slave sata, i Master sata. Standardne metode podesavanja sata na uredjajima nije opisano prema IEEE 1588 standardu; samo omogucava standardni protokol za razmenu poruka izmedju uredjaja. Poenta ovoga je da se uredjaji i satovi razlicitih proizvođača mogu sinhronizovati izmedju sebe.

QUALITY OF SYNCHRONIZATION Postoje nekoliko faktora koji mogu uticati na egzaktnost sinhronizacije izmedju dva uredjaja unutar IEEE 1588 mreze. Promene frekvencije na uredjaju koji daje tacno vreme, koje se može desiti izmedju dva sinhronizaciona paketa "Sync", mogu uzrokovati da se izgubi sinhronizacija sa ostalim uredjajima u istom sistemu. Kako bi se predupredila svaka mogucnost za gubljenje sinhronizacije, ([lazar] High stability) mogu se koristiti uredjaji sa veoma velikom stabilnoscu, kao i da se skрати vreme izmedju razmene sinhronizacionih paketa. Kako bi se još više unapredila sinhronizacija, mogu se koristiti druge vrste oscilatora u uredjajima, narocito Temperature Controlled Crystal Oscillators (TXCOs) i Oven Controlled Crystal Oscillators (OCXOs). Rezulucija sata može uticati na preciznost vremenskih zigova unutar sinhronizacionih paketa. Džiter ([lazar] Jitter) iz susednih uredjaja u mrezi, kao što su habovi i svicevi ([lazar] hubs and switches), takodje mogu uticati na preciznost sinhronizacije. Kvalitet IEEE 1588 mreznog sistema i kako je podesen može takodje uticati na kvalitet sinhronizacije. Kako bi se podesi sistem sa što boljom sinhronizacijom, mora se napraviti kompromis izmedju egzaktnosti sinhronizacije, cene, kao i razdaljine izmedju uredjaja u sistemu. Za sporije događaje unutar sistema koji ne zavise od vremena, standardna NTP sinhronizacija preko interneta, koja daje sinhronizaciju na nivou milisekunde, zadovoljava sve potrebe. IEEE 1588 je i dalje izvanredna alternativa za sisteme koji zahteva sinhronizaciju na nivou ispod mikrosekunde.

NETWORK HIERARCHIES IEEE 1588 sporedni satovi ([lazar] boundary clocks), koji se takodje i nazivaju transparentni svicevi ([lazar] transparent switches), pružaju efektivan način za smanjenje džitera ([lazar] jitter) unutar mreznog sistema baziranog na IEEE 1588 standardu. Svic ([lazar] switch), koji se koristi kao sporedni sat ([lazar] boundary clock), pokrece PTP protokol i sinhronizuje se na master sat ([lazar] master clock). Sporedni sat ([lazar] boundary clock), u navratima ([lazar] in turns) preuzima ulogu master sata za sve slejvove unutar iste mreze. Koristeci ovo podesavanje mreze, sva interna kasnjenja i džiter mogu biti kompenzovani i ne uticu na

egzaktnost sinhronizacije.

Delay\_Resp, Delay\_Req, Follow\_up i Sync poruke se ne prenose kroz sporedne satove. Sporedni sat se ponasa kao obican sat u smislu sinhronizacije i koristi algoritam najboljeg sata unutar podmreze. Unutar podmreze koja se posmatra, ovaj uredjaj je slejv. Ovo ce uticati na to da se svi ostali uredjaji koji se povezuju na sporedni sat sinhronisu svoje vreme prema njemu. Hijerarhija Roditelj-Dete (##[lazarc] Parent-Child hierarchy ##) na master-slejv sate je odredjena prema sporednim satovima. (##[lazarc] If cyclic path occur in the network hierarchy, the best master clock algorithm lowers the logical hierarchy to an acyclic graph.##) Naravno, postoji alternativa za sporedne sate, i to je koriscenje transparentnih sviceva (##[lazarc] switches ##). Transparentni svic se ne ponasa kao PTP cvor unutar IEEE 1588 sistema. Umesto toga, transparentni svic podesava vremenski deo PTP paketa kako bi se kompenzovalo kasnjenje koje unosi svic. Transparentni svic nakon toga preračunava koliko je vremena sinhronizacioni 'Sync' paket proveo unutar svica, i modifikuje vremenski zig unutar sledeceg 'Follow\_up' paketa kako bi se nadoknadilo kasnjenje. PTP cvorovi mogu raditi kao da su deo veceg podsistema lokalne mreze i to kao da su povezani habovima (##[lazarc] Hubs ##) koristeći transparentne sviceve.

USES FOR IEEE 1588 Precizna sinhronizacija se moze iskoristiti u sledecim aplikacijama:

- Telekomunikacije
- Energetska postrojenja
- Industrijska automacija
- Testiranje i merenja
- Robotska kontrola

## IZ PTP\_NEXT\_2.TXT

NETWORK TIMING TECHNOLOGY: NTP vs. PTP Vremenska sinhronizacija igra fundamentalnu ulogu u svakoj mrezi, ali je ipak najcesce dodata kao naknadna funkcionalnost. Ipak, moze znaciti razliku izmedju egzaktnog odredjivanja gresaka unutar sistema, u tacnim vremenskim trenucima, i nepostojanje ideje zasto se server ponasa onako kako nije predvidjeno. (##[lazarc] Time synchronization serves a fundamental role in any network, but it's too often added as an afterthought. However, it can mean the difference between correctly troubleshooting a conflict in minutes and having no idea why the server is figuratively on fire. ##) Za finansijske i naucne institucije, vremenska sinhronizacija mora biti tacna na milijarditi, ili nekad

na trilioniti deo sekunde, međutim sve više komercijalnih i industrijskih organizacija se sve više zalazu za ideju da preciznost vremenske sinhronizacije bude u sub-milisecond opsegu. Zasto nije dovoljno da samo sinhronizujemo svoje uređaje preko javno dostupnog NTP protokola. Nazalost, kasnjenje postoji svugde, i prosto je nemoguće postići savršenu sinhronizaciju. Brzina svetlosti je brza, u vakumu, foton može napraviti krug oko zemlje više od 7 puta u sekundi, iako putuje aproksimativno 31% sporije kroz običnu optičku mrežu, lako se može preneti jedan bit informacije preko pola sveta za manje od desetine sekunde. Ali, svi znamo da idealan svet ne postoji. Dodati svičeve (`##[lazar] switches ##`), rutere, i ostalu mrežnu infrastrukturu, i ta desetina sekunde se uveća nekoliko puta. Bez specijalizovane opreme, naša mreža lako može dodati kasnjenje čak i veće od sekunde. Još veća briga je sinhronizacija različitih uređaja unutar iste mreže. Scenario u kom finansijska institucija koja ima tačno 100 deonica kompanije X na berzi. U nekom trenutku se pojavi informacija koja se tiče kompanije X, i finansijska institucija odluci da proda tih 100 akcija ne samo jednom investitoru, već nekoliko njih u razmaku od sekunde. Ali postoje su serveri ove institucije nesinhronizovani jedan na drugi, ne postoji uopšte ideja kako odrediti koja je ponuda stigla prva. U ovom realnom scenariju sinhronizacija uređaja unutar mreže je od vitalnog značaja.

NETWORK TIME PROTOCOL NTP, ili Network Time Protocol, je široko prihvaćen kao sredstvo za čuvanje vremena na mreži, i trenutno je u upotrebi četvrta verzija samog protokola. Hijerarhijski sistem ima različite slojeve koji se nazivaju STRATA (`##[lazar] strata eng. ##`). Stratum 0 uređaji su u samom vrhu i uključuju atomske satove, kao one koji se nalaze u GNSS satelitima. Stratum 1, ili primarni vremenski server, svaki od njih ima jedan na jedan direktnu konekciju sa Stratum 0 satovima, i postižu sinhronizaciju red mikrosekunde sa Stratum 0 satovima, i povezuju se na ostale Stratum 1 severe za brzu proveru satova i čuvanje podataka. Stratum 2 serveri se mogu povezivati na više primarnih vremenskih servera kako bi se postigla veći level sinhronizacije i poboljšala preciznost, i tako dalje. NTP podržava maksimum od 15 strata uređaja, ali svaki strata uređaj unosi malu grešku u sinhronizaciji sa Stratum 0 uređajima.

64-bitni vremenski zig je trenutno implementiran kako bi se podelio u dva 32-bitna dela.

- Prva polovina broji broj sekundi do nešto preko 136 godina.
- Druga polovina predstavlja deo sekunde do razmere pikosekunde

Predložena je promena na 128-bitne vremenske zigoze u NTPv4 protokolu, i trebalo bi da poveća vremensku razmeru na nešto manje od 600

milijardi godina, pri čemu bi vremenska rezolucija bila manja od femtosekunde.

PRECISION TIME PROTOCOL PTP, ili Precision time protokol, je još jedan standard za sinhronizaciju vremena preko mreže, ali umesto sinhronizacije reda veličine milisekunde, PTP mreže mogu da postignu sinhronizaciju reda veličine nanosekunde, ili čak pikosekunde. Za većinu komercijalnih i industrijskih aplikacija, NTP je više nego precizan, ali ukoliko je potrebna tačnija sinhronizacija i tačnije obeležavanje vremena, potrebno je migrirati sistem na PTP.

Zbog čega je PTP toliko tačan? Koristi obeležavanje paketa hardverskim vremenskim zigoima, umesto softverskih, i kao svaki fini naučni instrument, PTP oprema je specijalizovana za samo jednu specijalizovanu svrhu: očuvanje sinhronizacije između uređaja. Samo iz ovih razloga, PTP mreže imaju mnogo tačniju vremensku rezoluciju, i ne kao NTP uređaji, PTP uređaji će ustvari uključiti vreme koje svaka od sinhronizacionih poruka provede u svakom od uređaja, što uključuje i kasnjenje u uređaju.

Svaka od PTP razmena uključuje seriju od 4 poruke koje se razmenjuju između master-a i slave-a:

- Inicijalna sinhronizaciona poruka od mastera ka slejvu Sync message
- Poruka koja prati sinhronizacionu poruku od mastera ka slejvu Follow\_up message
- Poruka sa zahtevom za određivanje kasnjenja od slejva ka masteru Delay\_Req message
- Finalni odgovor sa kasnjenjem od mastera ka slejvu Delay\_Resp message

Ova razmena pruža četiri različita vremena:

- T1 -> vreme kad master pošalje inicijalnu sinhronizacionu poruku
- T2 -> vreme kad slejv dobije inicijalnu sinhronizacionu poruku
- T3 -> vreme kad slejv pošalje poruku sa zahtevom za kasnjenjem
- T4 -> vreme kad master dobije zahtev za kasnjenjem

Master šalje sva 4 vremenska ziga ka slejvu tokom faze odgovora na zahtev za kasnjenjem, i slejv može sa tim vremenima da izračuna kasnjenje po mreži između mastera i slejva u oba smera. Imajući specijalizovani hardver koji

može da uhvati vremena lokalnog sata, slejv uređaji mogu izbeći dodatno kasnjenje koje je uslovljeno lokalnim operativnim sistemom.

NTP mreže imaju dodatno kasnjenje i manju preciznost jednostavno zbog toga što su softverski bazirane, i svi zahtevi za vremenima moraju da čekaju na lokalni operativni sistem. Za većinu kompanija, NTP pruža dovoljno tačnu rezoluciju vremena kako bi se rešili svi konflikti u dogledno vreme, dok neke organizacije zahtevaju dosta veći nivo sinhronizacije.

**WHY BOTHER WITH A TIME SERVER AT ALL?** Timestamping and client synchronization is vital for your network, but some network engineers still feel like they can get away with simply syncing their servers to a public internet clock. While perfectly fine for consumer devices like smartphones, internet clocks are poorly suited for business networks for one simple reason: security.

To connect your server to an internet clock requires you to first open up port 123 on your firewall. Will something horrible happen as a result? We don't know, but we don't know in the same way that we don't know if a burglar will break in because you left the front door unlocked on your home. Why take the chance? A dedicated NTP server keeps your network secure while providing more accurate timestamping.

**WHAT HAPPENS IF MY TIME SERVER IS DISCONNECTED?** No network is perfect, and all you can hope to do is minimize downtime instead of eliminating it. If your NTP or PTP time server is unable to connect to a GPS satellite or other input for whatever reason, you can rest assured that it will continue to synchronize your devices and maintain accurate timestamping.

For example, our NTP100-GPS NTP server has a holdover stability of 3 seconds per year, meaning that your server will still be synchronized to within 3 seconds of UTC after an entire year in the dark. The high-stability model with an oven-controlled crystal oscillator boasts even greater holdover stability of 250 milliseconds per year — that's less than 1 millisecond per day. Our HSO-3 oscillator option, which is only available on our GMR5000 NTP Server and PTP Grandmaster, further reduces drift to a maximum of 1 millisecond per year.