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, april 2019.

Sadržaj

Slike

Apstrakt

U ovom radu biće prikazana implementacija vremenske sinhronizacije u namenskim računarskim sistemima. U implementaciji je korišćen IEEE 1588 standard vremenske sinhronizacije. U implementaciji je korišćena razvojna ploča kompanije Atmel, ATSAMA5D27-SOM1-EK, na kojoj se nalazi ARM A5 mikrokontroler. Implementacija je odradjena u C programskom jeziku, uz korišćenje FreeRTOS operativnog sistema, i lwIP IP (Internet Protocol) steka. Test sistem za ovu implementaciju se sastoji od jednog računara opšte namene, na kome se pokreće Linux operativni sistem, i razvojne ploče na kojoj je implementirana vremenska sinhronizacija.

Ključne reči: Linux, C, Vreme, Sinhronizacija, FreeRTOS, lwIP, IEEE 1588, Atmel, ARM

Uvod

Vremenska sinhrnozacija igra fundamentalnu ulogu u svakoj mreži, ali je ipak najčešće dodata kao naknadna funkcionalnost. Ipak, može značiti razliku izmedju egzaktnog odredjivanja grešaka unutar sistema, u tačnim vremenskim trenucima, i nepostojanje ideje zašto se server ponaša onako kako nije predvidjeno. Za finansijske i naučne institucije, vremenska sinhronizacija mora biti tačna na milijarditi, ili nekad na trilioniti deo sekunde, medjutim sve više komercijalnih i industrijskih organizacija se zalažu za ideju da preciznost vremenske sinhronizacije bude u opsegu ispod milisekunde. Nažalost, kašnjenje postoji svugde, i prosto je nemoguće postići savršenu sinhrnoizaciju. 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. Dodati svičeve (eng. switches), rutere (eng. routers), i ostalu mrežnu infrastrukturu, i ta desetina sekunde se uveća nekoliko puta. Bez specijalizovane opreme, sama mreža lako može dodati kašnjenje čak i veće od sekunde. Još veći izazov predstavlja sinhronizacija različitih uredjaja unutar iste mreže.

Sinhronizacija postaje neophodna kada uredjaji koji rade zajedno na odredjenoj udaljenosti moraju raditi u vezi jedni sa drugima. U ovakvim sistemima, lokalni ili Master Sat, sinhronizuje sve uredjaje unutar istog sistema. Zbog potrebe za sinhronizacijom, IEEE 1588 standard je objavljen kao standardni protokol 2002. godine. Iako su dva sata unutar uredjaja podešeni da rade na istoj frekvenciji, ne postoji garancija da će ostati sinhronizovani. Upravo zbog ovoga proces sinhronizacije je neprekidan. Nekoliko faktora može uticati na to da dva identična sata izgube sinhronizaciju. Razlozi mogu biti različiti, kao na primer razlika u temperaturi, starosti uredjaja, kao i frekvenciji na kojoj uredjaji rade, koja može uticati na kvalitet sinhronizacije. Upravo iz ovih razloga je i nastala potreba za sinhronizacijom uredjaja.

S tim u vezi, u ovom radu će biti predstavljen primer implementacije vremenske sinhronizacije izmedju dva uredjaja, kao ogled primene Precision Time Protocol-a, i prikazivanje njegovih mogućnosti. Predložena implementacija je napravljena celokupno na osnovu *open source* softverskih modula.

U skladu sa tim, sve što je iskorišćeno može biti ponovo iskorišćeno u neke druge svrhe.

Implementirani sistem je vrlo jednostavan, sastoji se od jednog kompjutera opšte namene, PC-a, i razvojne ploče kompanije Atmel, bazirane na ARM A5 mikrokontroleru. Sam sistem je zamišljen tako da bude modularan, i da na osnovu neke naknadne potrebe, moze biti nadogradjen, ili iskorišćen u neke druge svrhe.

U prvom poglavlju će biti dat pregled modela komunikacije, sa posebnim osvrtom na TCP/IP i OSI modele komunikacije, kao i potpun prikaz samog PTP protokola. U drugom poglavlju biće prikazan operativni sistem koji je korišćen, kao i stek (eng. stack) za komunikaciju koji je iskorišćen za implementaciju. U trećem poglavlju je data hardverska implementacija sa opisom razvojne ploče koja je izabrana i razloga zbog kojih je izabrana ova ploča. U četvrtom poglavlju je dat opis softverske implementacije, rezultati same implementacije unutar oglednog sistema, kao i predlozi za poboljšanja.

Glava 1

Protokol

U ovom poglavlju će biti dat pregled konceptualnih modela Ethernet komunikacije, kao i objašnjenje celog PTP protokola. Na početku će biti predstavljen OSI konceptualni model, nakon njega TCP/IP model, i na kraju sam protokol koji je korišcen. Ovaj pregled je dat u svrhu dobijanja slike o tome koliko je kompleksan sam stek (eng. stack) koji se koristi, kao i da se dobije slika o nekim delovima koji će kasnije biti spominjani.

1.1 OSI model

Open Systems Interconnection model (OSI model) je konceptualni model koji karakteriše i standardizuje komunikacione funkcije u telekomunikacionim ili kompjuterskim sistemima, i to bez obzira na unutrašnju strukturu uredjaja 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 postiže komunikaciju preko mreže bez grešaka, 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.

Na svakom nivou N, dva entiteta na komunikcionim uredjajima razmenjuju jedinice protokola (PDU - protocol data units) pomoću sloja N protokola. Svaki PDU sadrzi podatke od interesa (eng. payload) (SDU - service

data unit), zajedno sa zaglavljima koji odgovaraju protokolu.

Obrada podataka izmedju dva uredjaja koji su OSI-kompatibilni se odvija u sledećim koracima:

- Podaci koji se prenose se formiraju na najvišem sloju u uredjaju koji predaje podatke na mreži (sloj N) u jedinicu protokola (PDU).
- PDU se prosledjuje sloju N-1, na kome je poznat kao SDU.
- Na sloju N-1 se na SDU dodaju zaglavlja, na osnovu čega se formira PDU za sloj N-1. Nakon čega se prosledjuje na sloj N-2.
- Ovaj postupak se ponavlja sve dok se ne dostigne najniži sloj u modelu, nakon čega se podaci prenose ka uredjaju koji prima podatke.
- Na strani prijemnog uredjaja se podaci prenose od najnižeg sloja u modelu, do najvišeg, gde se serije SDU struktura uspešno obradjuju, pri čemu se skidaju zaglavlja sa svakog sloja, dok se ne dostigne najviši sloj u modelu, nakon čega su dostupni sirovi podaci.

OSI Model			
Layer		Protocol data unit (PDU)	Function[3]
Host layers	7. Application	Data	High-level APIs, including resource sharing, remote file access
	6. Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5. Session		Managing communication sessions, i.e. continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4. Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3. Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2. Data link	Frame	Reliable transmission of data frames between two nodes connected by a physical layer
	1. Physical	Symbol	Transmission and reception of raw bit streams over a physical medium

Slika 1.1: Prikaz svih slojeva unutar OSI modela komunikacije

1.1.1 Sloj 1: Fizički sloj (eng. Physical Layer)

Fizički sloj je odgovoran za prenos i prijem nestruktuiranih sirovih podataka izmedju uredjaja i fizičkog medijuma za prenos. On pretvara digitalne bitove u električne, radio ili optičke signale. Specifikacije sloja definišu karakteristike poput nivoa napona, fizičke brzine prenosa podataka, maksimalne udaljenosti prenosa i fizičkih konektora. Ovo uključuje raspored pinova, napona, linijske impedanse, specifikacije kablova, vremenskih signala i frekvencije za bežične uredjaje. Kontrola brzine bitova se vrši na fizičkom nivou i može definisati način komunikacije kao simpleks, polu dupleks ili dupleks komunikaciju. Komponente fizičkog sloja mogu se opisati u smislu topologije mreže.

1.1.2 Sloj 2: Sloj veze (eng. Data Link Layer)

Sloj veze podataka obezbedjuje prenos podataka izmedju dva uredjaja u komunikaciji - vezu izmedju dva direktno povezana uredjaja na mreži. Ovaj sloj otkriva i eventualno ispravlja greške koje se mogu javiti u fizičkom sloju. On definiše protokol za uspostavljanje i prekid veze izmedju dva fizički povezana uredjaja. Takodje, definiše protokol za kontrolu protoka izmedju njih.

IEEE 802 standard deli sloj veze na dva podsloja:

- **Kontrola pristupa medijumu (eng. MAC - Medium access control):** odgovorna je za kontrolu načina na koji uredjaji na mreži dobijaju pristup medijumu i dozvolu za prenos podataka.
- **Kontrola logicke veze (eng. LLC - Logical link control):** odgovorna je za identifikaciju i enkapsulaciju slojeva mrežnog protokola, i kontrolu grešaka i sinhronizaciju paketa koji se šalju.

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

Point-to-Point Protocol (PPP) - je protokol sloja veze koji radi na nekoliko različitih fizičkih slojeva, kao što su sinhrone ili asinhrone serijske linije.

1.1.3 Sloj 3: Mrežni sloj (eng. Network Layer)

Mrežni sloj obezbedjuje funkcionalno i proceduralno sredstvo prenosa sekvenci podataka promenljive dužine, koji se nazivaju još i paketi, od jednog čvora do drugog, poveznih u "različite mreže". Mreža je medijum na koji može biti povezano više čvorova, u kom svaki čvor ima adresu i koji dozvoljava čvorovima povezanim sa njim da prenose poruke ka ostalim čvorovima. Prenos poruka se obavlja tako što se daje samo sadržaj poruke i adresu čvora na koji poruka treba da bude dostavljena, omogućavajući mreži da nadje način da isporuči poruku odredišnom čvoru, eventualno ga usmeravajući kroz središnje čvorove (uredjaje koji su izmedju dva uredjaja koji pokušavaju da komuniciraju). Ako je poruka prevelika da bi se prenela sa jednog uredjaja na drugi samo korišćenjem sloja veze (eng. data link layer), mreža može preneti podatke tako što će ih podeliti u nekoliko delova na jednom uredjaju, poslati delove nezavisno, i onda spojiti delove na drugom uredjaju. Pri čemu može, iako nije uvek potrebno, prijaviti greške u isporuci.

Isporuka poruka na mrežnom sloju nije garantovano pouzdana. Mrežni sloj može pružiti pouzdanu isporuku poruka, ali nije obavezno da to mora biti ispunjeno.

Odredjeni protokoli koji upravljaju slojevima, imaju funkciju koja je definisana u aneksu upravljanja, ISO 7498/4, i pripadaju mrežnom sloju. Oni uključuju protokole rutiranja, upravljanja grupom sa više uredjaja, informacije o mrežnom sloju, o greškama, kao i dodeljivanje adresa mrežnog sloja medju uredjajima. Ustvari, to je funkcija podataka koji se prenose uz pomoć protokola, što ih čini da pripadaju mrežnom sloju, a ne protokolu.

1.1.4 Sloj 4: Transportni sloj (eng. Transport Layer)

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

Transportni sloj kontroliše pouzdanost uspostavljene konekcije kroz kontrolu protoka, segmentaciju/desegmentaciju i kontrolu grešaka. Neki protokli su orjentisani na stanje mreže, a neki na konekciju u mreži. Ovo znači da Transportni sloj može da prati segmente i ponovo prenese one koji nisu isporučeni prijemnom uredjaju. Transportni sloj takodje 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 viših slojeva, npr. Aplikativnog sloja (eng. application layer). Segmentacija je proces podele dugih poruka u kraće poruke kako bi se lakše prenele preko nižih slojeva u modelu.

OSI model definiše pet klasa transportnih protokola za povezivnje od klase 0 (koja je takodje 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 grešaka. Klasa 4 (TP4) je najbliža TCP, iako TCP sadrži neke funkcije koje se u OSI modelu dodeljuju višim slojevima. Takodje, sve klase u OSI modelu omogućavaju brzu upotrebu podataka i očuvanje granica podataka. Detalje karakteristika svih klasa prikazane su u sledećoj tabeli:

Feature name	TP0	TP1	TP2	TP3	TP4
Connection-oriented network	Yes	Yes	Yes	Yes	Yes
Connectionless network	No	No	No	No	Yes
Concatenation and separation	No	Yes	Yes	Yes	Yes
Segmentation and reassembly	Yes	Yes	Yes	Yes	Yes
Error recovery	No	Yes	Yes	Yes	Yes
Reinitiate connection[a]	No	Yes	No	Yes	No
Multiplexing / demultiplexing over single virtual circuit	No	No	Yes	Yes	Yes
Explicit flow control	No	No	Yes	Yes	Yes
Retransmission on timeout	No	No	No	No	Yes
Reliable transport service	No	Yes	No	Yes	Yes

[a] If an excessive number of PDUs are unacknowledged.

1.1.5 Sloj 5: Sloj sesije (eng. Session Layer)

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

1.1.6 Sloj 6: Sloj prezentacije (eng. Presentation Layer)

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

Ovaj sloj obezbedjuje 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. Takodje, može uključivati i funkcije kompresije.

1.1.7 Sloj 7: Aplikativni sloj (eng. Application Layer)

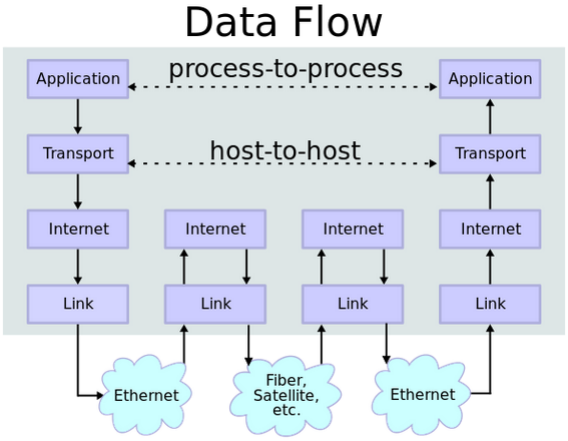
Aplikativni sloj je OSI sloj najbliži krajnjem korisniku, što znači da i OSI aplikativni sloj i korisnik interaguju direktno sa aplikacijom. Ovaj sloj komunicira sa softverskom aplikacijom koja sadrži komponentu za komunikaciju. Takve aplikacije ne spadaju u okvir OSI modela. Funkcije u aplikativnom sloju obično uključuju identifikovanje partnera u komunikaciji, odredjivanje dostupnosti resursa i sinhronizaciju komunikacije. Prilikom identifikacije uredjaja za komunikaciju, aplikativni sloj se razlikuje od samih aplikacija. Na primer, internet aplikacija (web strana) može imati dva entiteta - dve aplikacije: jedna koja koristi HTTP za komunikaciju sa korisnicima, i drugu za udaljenu bazu podataka koja čuva podatke. Ni jedan od ovih protokola nemaju ništa sa podacima koji se čuvaju, to se nalazi samo u aplikaciji. Aplikativni sloj nema načina za odredjivanje resursa u mreži.

1.2 Internet protocol suite

Internet protocol suite (TCP/IP) je konceptualni model i set komunikacionih protokola koji se koriste za Internet i slične kompjuterske mreže. Opšte je poznat kao TCP/IP zbog toga što 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 što je razvijanje ovog modela potpomoglo Ministarstvo odbrane SAD-a kroz DARPA.

Internet protokol suite omogućava razmenu podataka izmedju dva uredjaja na mreži, i to specificirajući kako će se podaci deliti u pakete, adresirati, prenositi, rutirati, i primati. Ove funkcionalnosti su organizovane u četiri apstraktna sloja, koja klasifikuju sve protokole s obzirom na to u kom delu povezivanja se nalaze. Od najnižeg do najvišeg, slojevi se dele na Link Layer (Sloj povezivanja), koji sadrži komunikacione metode za podatke koji ostaju unutar jednog segmenta mreže; Internet Layer (Sloj interneta), koji omogućava povezivanje izmedju nezavisnih mreža; Transport Layer (Sloj prenosa), koji omogućava komunikacione servise za aplikacije na uredjajima u mreži; i Application Layer (Sloj aplikacije), koji omogućava servise korisnicima i sistemskim aplikacijama.

Tehnički sandardi koji se specificiraju u Internet protocol suite i mnogi od protokola koji čine IPS održava Internet Engineering Task Force (IETF). IPS je model koji prethodi OSI modelu, koji je dosta detaljniji.



Slika 1.2: Prikaz slojeva unutar TCP/IP modela komunikacije

1.2.1 Glavni principi arhitekture modela

Princip od kraja do kraja (eng. end-to-end) 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 (eng. firewalls), prevodiocima mrežnih adresa, keširanju sadržaja sa interneta i slično, su izazvale promene u ovom principu.

Princip robusnosti kaže: "Uopšteno, implementacija mora biti konzervativna u ponašanju prilikom slanja i liberalna u svom ponašanju prilikom prijema. Što znači, da mora biti oprezna pri slanju dobro formiranih paketa, ali mora prihvatiti bilo koji paket koji može protumačiti. (na primer, ne primećivati tehničke greške gde nije poznato šta ih uzrokuje.)". "Drugi deo principa je gotovo jednako važan: softver na drugim uredjajima može sadržati razlike koje se čine nerazumne kako bi se iskoriste legalne, ali nejasne karakteristike protokola".

Enkapsulacija se koristi za obezbedjivanje 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, što je u upotrebi i danas. I oni su unutar protokola korišćeni u istom redosledu u kom će i ovde biti navedeni.

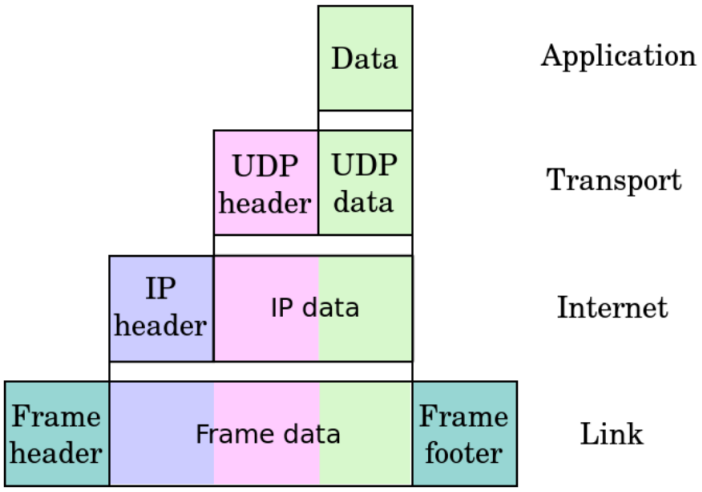
- **Aplikativni sloj (eng. Application layer)**: Aplikativni sloj je opseg unutar kog aplikacije kreiraju korisničke podatke i prenose ove podatke drugim aplikacijama na istom ili drugom uredjaju (eng. host). Aplikacije ili procesi, koriste usluge koje pružaju donji slojevi, posebno transportni sloj koji obezbedjuje pouzdane ili nepouzdane 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. Procesi se adresiraju preko portova koji u suštini predstavljaju usluge.

- **Transportni sloj (eng. Transport layer)**: Transportni sloj obavlja *host-to-host* komunikaciju, na istim ili različitim uredjajima, i u lokalnoj mreži ili udaljenim mrežama razdvojenim od rutera. Ovaj sloj obezbedjuje 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 (eng. Internet layer)**: Internet sloj razmenjuje pakete preko mreže. Ovaj sloj obezbedjuje 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 bliže kranjem odredištu podataka.

- **Sloj veze (eng. Link layer)**: Sloj veze definiše metode umrežavanja u okviru lokalne mrežne veze na kojoj uredjaji komuniciraju bez rutera u medjukomunikaciji. 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 uredjaje.



Slika 1.3: Prikaz poruke koja se prenosi prema TCP/IP modelu komunikacije

Slojevi protokola blizu vrha su logično bliži korisnickoj aplikaciji, dok su oni bliže dnu logički bliži 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.

1.3 Precision time protokol

Protokol preciznog vremena (Precision Time Protocol (PTP)) je protokol korišćen za sinhronizaciju satova preko kompjuterske mreže. U lokalnoj kompjuterskoj mreži (eng. local area connection), postiže se preciznost sata i u rangu ispod mikrosekunde, što ga čini 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", objavljen 2002 godine. U 2008 godini, IEEE 1588-2002 je objavljen kao preradjen standard, poznat i kao PTP Version 2, sa poboljšanom tačnošću, preciznošću i robusnošću, medjutim 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. Takodje je dizajniran za aplikacije koje se ne mogu nositi sa cenom GPS prijemnika na svakom 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.

Postoji nekoliko faktora koji mogu uticati na egzaktnost sinhronizacije izmedju dva uredjaja unutar IEEE 1588 mreže. Promene frekvencije na uredjaju koji daje tačno 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 mogućnost za gubljenje sinhronizacije, mogu se koristiti uredjaji sa veoma velikom stabilnošću, kao i da se skrati 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 (TCXOs) i Oven Controlled Crystal Oscillators (OCXOs). Rezulucija sata može uticati na preciznost vremenskih žigova unutar sinhronizacionih paketa. Džiter (eng. jitter) iz susednih uredjaja u mreži, kao sto su habovi i svičevi (eng. hubs and switches), takodje mogu uticati na preciznost sinhronizacije. Kvalitet IEEE 1588 mrežnog sistema i kako je podešen može takodje uticati na kvalitet sinhronizacije. Kako bi se podesio sistem sa što boljom sinhronizacijom, mora se napraviti kompromis izmedju egzaktnosti sinhronizacije, cene, kao i raz-

daljine izmedju uredjaja u sistemu. Za sporije dogadjaje 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 sinronizaciju na nivou ispod mikroseunde.

Precizna sinhronizacija se može iskoristiti u sledećim aplikacijama:

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

1.3.1 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 (eng. strata). Statum 0 uredjaji 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 1 satom, 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 postigao viši nivo sinhronizacije i poboljšala preciznoste. NTP podržava maksimum od 15 strata uredjaja, ali svaki strata uredjaj unosi malu grešku u sinhronizaciji sa Stratum 0 uredjajima.

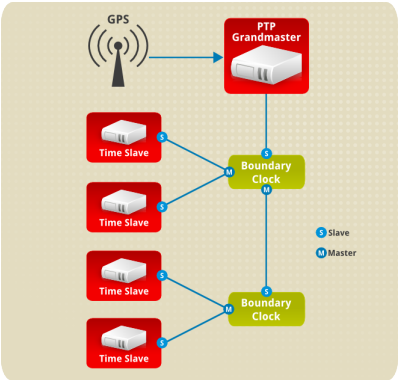
64-bitni vremenski žig je trenutno implementiran kako bi se podelio u dva 32-bitna dela.

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

Predložena je promena na 128-o bitne vremenske žigove 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.

1.3.2 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 više komunikacionih medijuma (segmenata koji su povezani na mrežu), i jednog ili više izvora tačnog vremena. Izvor "običnog" vremena ("ordinary clock") je uredjaj sa jednim pristupom mreži i ima jednu od dve uloge, ili je izvor (master) tačnog vremena, ili čeka na tačno vreme (slave) u komunikaciji na mreži. Granični sat (Boundary clock) ima više pristupa, na različite mreže, i može precizno sinhronizovati jedan segment mreže na drugi. Master sinhronizacije se bira za svaki segment mreže u sistemu. 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 mrežu sa njim. Ukoliko se u nekom delu mreže nalazi Boundary clock, on prosledjuje tačno vreme ka ostalim uredjajima koji su direktno na njega povezani.



Slika 1.4: Prikaz arhitekture jednog sistema koji podržava PTP

Simplifikovano, PTP sistem se sastoji od izvora "običnog" vremena poveznih na jednostavnu mrežu, i bez graničnih satova. Grandmaster se bira, i svi ostali uredjaji se direktno sinhronišu na njega.

IEEE 1588-2008 predstavlja Clock koji je povezan sa mrežnom opremom koja prenosi PTP poruke. Transparentni uredjaj (eng. Transparent clock) modifikuje PTP poruke koje prolaze kroz uredjaj. Vremenski pečati (TIMESTAMPs) u porukama su modifikovani tako da se uzme u obzir i vreme za koje poruka prolazi kroz dodatne uredjaje u komunikaciji. Ova šema komunikacije povecava distribuciju preciznosti tako što se kompenzuje promenljivost dostave podataka preko mreže.

PTP tipično koristi EPOCH vreme, standardno vreme za UNIX sisteme (1. Januar 1970. kao početak računanja vremena). Dok je UNIX vreme ba-

zirano na Univerzalnom vremenu UTC, i mora da postoji sekunda preskoka, 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 izračuna u odnosu na primljeno PTP vreme.

1.3.3 Detalji protokola

Sinhronizacija i obrada u PTP sistemu se postiže razmenom poruka preko komunikacionog medijuma. PTP standard propisuje samo ove tipove poruka:

- Sync, Follow_Up, Delay_Req i Delay_Resp poruke se koriste u Ordinary i Boundary uredjajima i služe samo za razmenu informacija o vremenu koje se koriste za sinhronizaciju uredjaja na mreži.
- Pdelay_Req, Pdelay_Resp i Pdelay_Res_Follow_Up se koriste u Transparent Clock uredjajima kako bi se izmerilo kašnjenje kroz uredjaj tako da se može iskoristiti u kompenzaciji vremena u sistemu. Transparent Clock i definicija ovih poruka nisu dostupne u IEEE 1588-2002 standardu.
- Announce poruke se koriste u Best master clock algoritmu specificiranom u IEEE 1588-2002 standardu, za algoritam odredjivanja najtačnijeg sata na mreži, i to kako bi se izgradila hijerarhija uredjaja i kako bi se odredio Grandmaster Clock.
- Management poruke se koriste u upravljanju mrežom, za posmatranje performansi na mreži, konfiguraciju mreže i održavanje PTP sistema.
- Signalne poruke se koriste u komunikaciji izmedju uredjaja koje nisu vremenski kritične. Signalne poruke su uvedene u IEEE 1588-2002 standard.

Poruke se karakterizuju kao Event i General, odnosno poruke dogadjaja i opšte poruke. Event poruke su vremenski kritične, čija se kritičnost ogleda u preciznosti predaje i prijema vremenskih pečata (TIMESTAMPs), čime direktno utiču na distribuciju preciznosti vremena u sistemu. Sync, Delay_Req, Pdelay_Req i Pdelay_resp su poruke dogadjaja. Opšte poruke su uobičajene jedinice protokola, zato što su podaci u ovim porukama 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.

1.3.4 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 modu (modu objavljivanja na mreži), dok je u IEEE 1588-2008 to uvedeno kao opcija.

1.3.5 Algoritam najboljeg sata

BMC (*Best master clock algorithm*) algoritam obavlja deljenu selekciju najboljeg kandidata za tačno vreme prema sledećim karakteristikama:

- **Identifikator**: Univerzalni jednistveni identifikator za sat. Tipično je baziran na MAC adresi uredjaja.
- **Kvalitet**: Obe verzije IEEE 1588 standarda pokušavaju da kvantifikuju kvalitet sata na osnovu očekivanih devijacija u vremenu, tehnologije koja je korišćena za implemntaciju vremena ili lokacije u hijerarhiji satova, u šemi kvaliteta satova (eng. clock stratum scheme).
- **Prioritet**: Administrativno dodeljen prioritetni znak koji BMC koristi kako bi što bolje odredio Grandmaster u PTP domenu. Dok je IEEE 1588-2002 standard imao samo jednu logičku promenljivu kako bi odredio prioritet, IEEE 1588-2008 ima dva 8-bitna polja prioriteta.
- **Varijansa**: Procena stabilnosti sata zasnovana na zapažanju njegovog učinka prema PTP refernci.

IEEE 1588 koristi hijerarhijski algoritam selekcije zasnovan na sledećim osobinama, u naznačenom redosledu:

- **Prioritet 1**: korisnik može dodeliti specifično dizajniran statički prioritet svakom satu, time odredjujući prioritet medju njima. Prioritet 0 se smatra najvećim prioritetom.
- **Klasa**: Svaki sat je član odredjene klase, svaka klasa dobija svoj prioritet.
- **Tačnost**: Tačnost uredjaja, u odnosu na UTC, u nanosekundama.
- **Varijansa**: Varijabilnost sata.

- **Prioritet 2**: Definisan prioritet, definišući redosled rezervne kopije u slučaju da drugi kriterijumi nisu dovoljni. Manje vrednosti prioriteta označavaju veći prioritet.

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

Svojstva sata se daju, u IEEE 1588-2002 standardu, porukama za sinhronizaciju (Sync messages) i, u IEEE 1588-2008 standardu, u porukama za oglašavanje (Announce messages). Trenutni Master clock prenosi sve informacije u rednovnim intervalima. Sat koji sebe smatra boljim od trenutnog Master sata prenosiće ove informacije kako bi se pozvali svi uredjaji za promenu Master sata. Kada trenutni Master prepozna bolji sat, tada Master sat zaustavlja emitovanje poruka za sinhronizaciju (Sync Messages), ili poruke oglašavanja (Announce messages), u zavisnosti od verzije protokola, i bolji sat preuzima ulogu Master sata. BMC algoritam uzima u obzir samo osobine koje su već poznate, i koje su deklarisali sami satovi, i ne uzima u obzir kvalitet veze na mreži.

1.3.6 Sinhronizacija

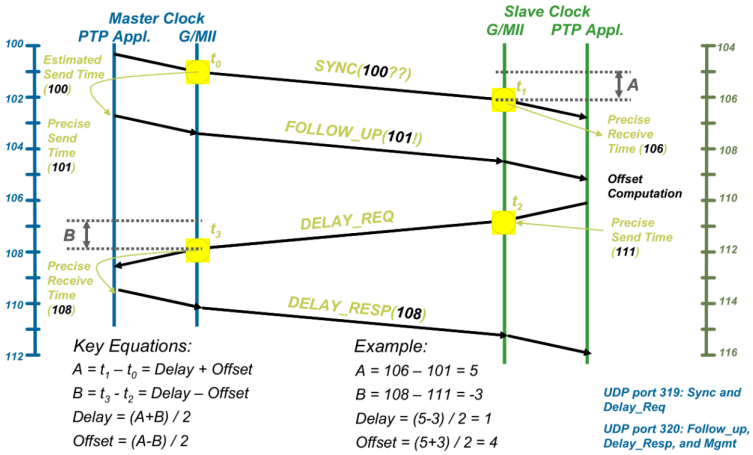
Koristeći BMC algoritam, PTP bira Master sat za IEEE 1588 domen i za svaki segment mreže unutar tog domena. Satovi odredjuju razliku izmedju njih (eng. offset) i Master-a na mreži. Neka promenjiva t predstavlja fizički vreme. Za dati Slave uredjaj, razlika $o(t)$ u vremenu t se definiše kao:

$$o(t) = s(t) - m(t) \tag{1.1}$$

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 (eng. broadcasts) trenutno vreme kao poruku ostalim uredjajima na mreži. 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.

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 dobaviti tačne vremenske trenutke stizanja Sync



Slika 1.5: Prikaz sinhronizacione sekvence PTP protokola

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 počinje tako što Slave uredjaj šalje Delay_Req poruku u vremenskom trenutku T_2 ka svom Master uredjaju. Master uredjaj 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 uredjaj 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 prodje kroz medijum za povezivanje, a $\tilde{o}$ konstantna razlika satova izmedju Master i Slave uredjaja, onda je:

$$T_1' - T_1 = \tilde{o} + d \tag{1.2}$$

i

$$T_2' - T_2 = -\tilde{o} + d \tag{1.3}$$

Odakle je:

$$\tilde{o} = \frac{1}{2}(T_1' - T_1 - T_2' + T_2) \tag{1.4}$$

Sada dva uredjaja znaju koliki je ofset $\tilde{o}$ 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. Još 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 šalju ili primaju poruke. Stepen primene ovih pretpostavki utiče na to koliko će se dobro sinhronizovati dva uredjaja.

Glava 2

Operativni sistem

2.1 FreeRTOS

FreeRTOS (Real-time Operating System) je kernel operativnog sistema koji radi u realnom vremenu, i može se koristiti prilikom projektovanja namenskih sistema na velikom broju mikrokontrolera. FreeRTOS je idealno sklopljen za duboke namenske aplikacije u realnom vremenu koje koriste mikrokontrolere ili male mikroprocesore. Ovaj način projektovanja aplikacija uključuje 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, doći ce do apsolutnog pada funkcionalnosti sistema. Na primer, airbag u kolima ima potencijal da napravi više štete nego dobrog ukoliko je odziv sistema samo malo sporiji nego što treba.

FreeRTOS je kernel (ili rasporedjivač realnog vremena) na koji se nadogradjuje aplikacija tako da ispuni stroge zahteve za realnim vremenom. To dozvoljava da aplikacija bude organizovana kao kolekcija nezavisnih programskih niti. Na procesoru koji ima samo jedno jezgro, samo jedna programska nit se može izvršavati u jednom trenutku. Kernel odlučuje koja nit se izvršava tako što odredjuje prioritet koji se dodeljuje svakoj niti. U najjednostavnijem slučaju, dizajner aplikacije može odrediti više prioritete nitima koje implementiraju stroge zahteve za realnim vremenom, a niže prioritete onim nitima koje nemaju tako stroge zahteve za izvršavanjem. Ovim bi se osiguralo da niti koje imaju strožije 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 će se u daljem tekstu i koristiti naziv Task za programsku nit.

U projektovanju aplikacija za namenske sisteme postoji ustaljena praksa koja ne zahteva korišćenje kernela za realno vreme, i ove tehnike mogu dati bolje rešenje problema. Mada, u kompleksnijim slučajevima, verovatnije je korišćenje kernela za aplikacije u realnom vremenu, i takodje može biti kombinacija korišćenja kernela, i drugih tehnika projektovanja aplikacije.

Kao što je već opisano, prioriteti taskova mogu pomoći da se osigura da aplikacija ispuni sve zahteve, ali kernel može doneti i neke manje očigledne beneficije. Neke od njih su navedene ispod:

- **Skraćivanje informacija o vremenskom rasporedu (eng. Abstracting away timing information)**: Kernel je odgovoran za vreme izvršavanja i dodeljuje API kojim se unutar aplikacije može upravljati vremenom. Ovim se omogućava jednostavnija struktuiranost koda, i ukupna veličina koda je manja.

- **Održavanje/Proširivanje (eng. Maintainability/Extensibility)**: Uskraćivanjem informacija o vremenskom rasporedu rezultuje u manjim zavisnostima izmedju modula, i dozvoljava aplikaciji da evoluira na kontrolisan i predvidjen način. Takodje, kernel je odgovoran samo za rasporedjivanje vremena, tako da performanse aplikacije mogu biti promenjene u zavisnosti od hardvera na kome se pokreću.

- **Modularnost (eng. Modularity)**: Taskovi su nezavisni moduli, pri čemu svaki od njih mora imati dobro definisanu svrhu.

- **Timski razvoj (eng. Team development)**: Taskovi bi trebalo da imaju dobro definisane interfejse, kako bi se lakše razvijali u timovima.

- **Lakše testiranje (eng. Easier testing)**: Ako su taskovi dobro definisani kao nezavisni moduli sa čistim interfejsima, mogu biti testirani nezavnisno.

- **Ponovno korišćenje koda (eng. Code reuse)**: Veća modularnost sa većom nezavisnošću koda koji se može ponovo koristiti sa manje uloženog truda.

- **Poboljšana efikasnost (eng. Improved efficiency)**: Korišćenjem kernela softver se u popunosti može prebaciti na pokretanje dogadjajima (eng. event driven programming), i time bi se uštedelo procesorsko vreme koje se troši na poliranje dogadjaja koji se ne dogadjaju. Kod se pokreće samo ukoliko postoji nešto sto je potrebno uraditi. Protiv poboljšane efikasnosti stoji to da je potrebno pocesuirati RTOS prekid,

i promeniti izvršavanje sa jednog taska na drugi. Medjutim, i aplikacije koje ne koriste RTOS normalno uključuju neku formu prekida.

- **Idle time**: Idle task je task koji se automatski kreira prilikom startovanja Scheduler-a. I izvršava se kad nema taskova unutar aplikacije koji bi se izvršavali. Ovaj task se može koristiti za merenje procesorske moći koja se troši, za izvršavanje provera u pozadini, ili da jednostavno pokrene režim smanjene potrošnje u sistemu.

- **Upravljanje snagom (eng. Power management)**: Efikasnost koja se dobija korišćenjem RTOS-a dozovoljava procesoru da provede više vremena u režimu smanjene potrošnje. Potrošnja se može značajno smanjiti time što procesor odlazi u režim smanjene 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 vreme.

- **Fleksibilno upravljanje prekidima (eng. Flexible interrupt handling)**: Upravljanje prekidima se može držati veoma kratko tako što se odlaže obrada bilo kog taska koji je kreirao sam dizajner, ili taska unutar FreeRTOS-a.

- **Različiti zahtevi za obradom (eng. Mixed processing requirements)**: Jednostavni oblici dizajniranja programa mogu se postići mešanjem periodičnog, kontinualnog i procesiranja pokretanog dogadjajima. Pored toga, ispunjavanje strogih i manje strogih zahteva za realnim vremenom u aplikacijama može se postići izborom odgovoarajućih taskova i prioriteta prekida.

2.1.1 Implementacija

FreeRTOS je dizajniran tako da bude mali i jednostavan. Kernel (srce operativnog sistema) se sastoji od samo 3 fajla. Kako bi se kod napravio da bude čitljiv, lako portabilan, i kako bi se lako održavao projekat, pisan je uglavnom u C programskom jeziku, sa izuzetkom nekih funkcionalnosti koje su napisane u asembleru, gde je to bilo potrebno, i to uglavnom rutine u Scheduler-u (Rasporedjivaču niti) koje su specifične za samu arhitekturu.

FreeRTOS omogućava korišćenje metoda za stvaranje više programskih niti, ili Taskova, stvaranje mehanizama za Sinhronizaciju niti, Mutexa, Semafora i softverskih tajmera. Takodje, postoje mogućnosti korišćenja FreeRTOS-a i za aplikacije niske potrošnje. Aplikacije koje koriste FreeRTOS mogu biti kompletno statički alocirane. Alternativno RTOS objekti mogu biti dina-

mički alocirani sa 5 šema alokacije memorije i one su rasporedjene tako da pružaju sledeće mogućnosti:

- samo alociranje;
- alociranje i oslobodjanje sa jednostavnim, brzim algoritmom;
- kompleksnije ali brže alociranje i oslobadjanje uz algoritam spajanja susednih memorijskih blokova;
- alternativa za još kompleksniju šemu koja uključuje spajanje susednih memorijskih blokova koja omogućava da hip (eng. heap) bude podeljen na više memorijskih delova;
- i na kraju C biblioteka za alociranje i oslobadjanje sa zaštitom medjusobnog isključivanja.

Unutar FreeRTOS-a ne postoji ni jedan od složenijih svojstava operativnih sistema koji se uobičajeno mogu naći u operativnim sistemima poput Linux-a ili Microsoft Windows-a, kao što su drajveri uredjaja, napredno upravljanje memorijom, korisnički nalozi, i umrežavanje. Akcenat ovog operativnog sistema je na kompaktnosti i brzini izvršavanja. O FreeRTOS-u se može misliti kao o "biblioteci niti" vise nego kao o "operativnom sistemu".

FreeRTOS implementira više niti tako što postoji jedan program koji poziva metode niti u jednakim kratkim vremenskim intervalima. Metoda promene niti zavisi od prioriteta niti i uključuje round-robin šemu promene niti. Uobičajen interval promene je do 1/1000 sekunde do 1/100 sekunde, i to kroz prekid hardverskog tajmera, ali interval promene se često menja tako da zadovolji potrebe specifične aplikacije.

2.2 lwIP

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

lwIP je open-source projekat koji je besplatan za preuzimanje i korišćenje (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 korišćenje RAM memorije, pri čemu se i dalje dobija potpuna funcionalnost TCP. Ovim lwIP

postaje interesantan za korišćenje u namenskim sistemima koji raspolažu sa RAM memorijom od nekoliko desetina kilobajta (kB) i prostorom od oko 40kB 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 iskorišćen na mnogim platformama i operativnim sistemima, i može 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 podršku za dosta protokola, od kojih većina može da se ukloni za manju veličinu koda.

- **Mrežni protokoli i protokoli veze: (eng. 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, naročito, proširuje veličinu IP adrese na 128 bita;
  - **ICMP**: kontrolni protokol za IP;
  - **IGMP**: protokol za upravljanje grupama unutar IP-a;
- **Transportni protokoli: (eng. Transport protocols)**
  - **UDP**: protokol bez priključka, i bez mehanizma pouzdanosti;
  - **TCP**: protokol orjentisan ka konekciji, za kontinualni tok podataka (eng. streaming);
- **Protokoli visokog nivoa: (eng. High-level protocols)**
  - **DHCP**: dobijanje IP adrese sa podrškom servera;
  - **AUTOIP**: dobijanje IP adrese bez podrške servera;
  - **SNMP**: nadgledanje stanja mreže;
  - **PPP**: stvaranje direktne konekcije izmedju dva čvora na mreži;

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

- low-level "core"/"callback" ili "raw" API

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

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

Prilikom mešanja sekvencijalnog i "sirovog" API-a u programima, treba biti pažljiv. Funkcije koje pripadaju nesekvencijalnom API-u u stvari mogu biti pozvane iz glavne *tcpip_thread* niti. Takodje, registrovanje programskih 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).

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

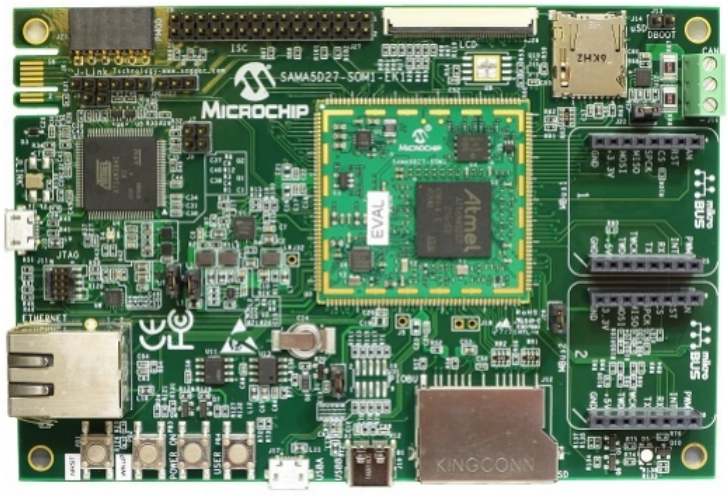
- **netconn- i raw-API su samo unutar lwIP-a :** kod koji koristi ovaj API se ne može koristiti u drugim stekovima koji imaju iste mogućnosti kao lwIP (na primer uIP i td.)
- **socket API** je u suprotnosti sa gore navedenom stavkom, napravljen je tako da je kompatibilan i može se koristiti u drugim 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[1] operaciju kako za TX tako i za RX, tj. kako za predaju, tako i za prijem podataka.

[1]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 3

Hardverska implementacija

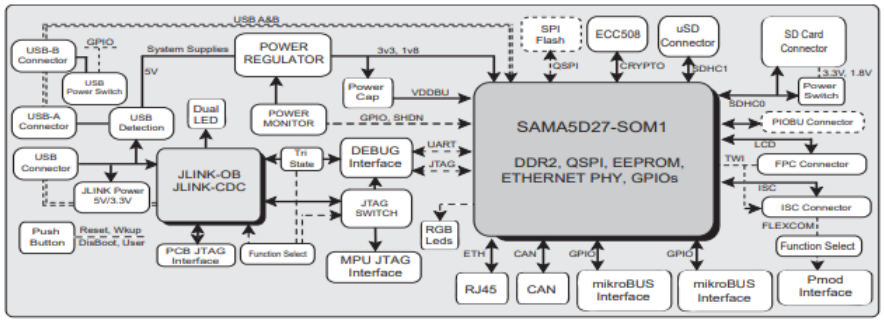
Razvojno okruženje koje je korišćeno za hardversku implementaciju je razvojna ploča SAMA5D27-SOM1-EK1 proizvodjača Microchip. Na ploči se nalazi SAMA5D27 SOM (System on Module) modul koji je ključan za implementaciju. Na modulu se nalazi SAMA5D27-D1G-CU SIP (System in Package) koji sadrži 1 Gbit DDR2 SDRAM memorije. Modul nudi puzdanu i niskobudžetnu platformu za razvoj namenskih računarskih sistema koji ce 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 3.1: 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 uredjaja, 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 obezbedjuju pristup SOM-u za različite upotrebe. Svi GPIO pinovi su nezavisni, i mogu se konfigurisati kao ulazni ili izlazni, 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.

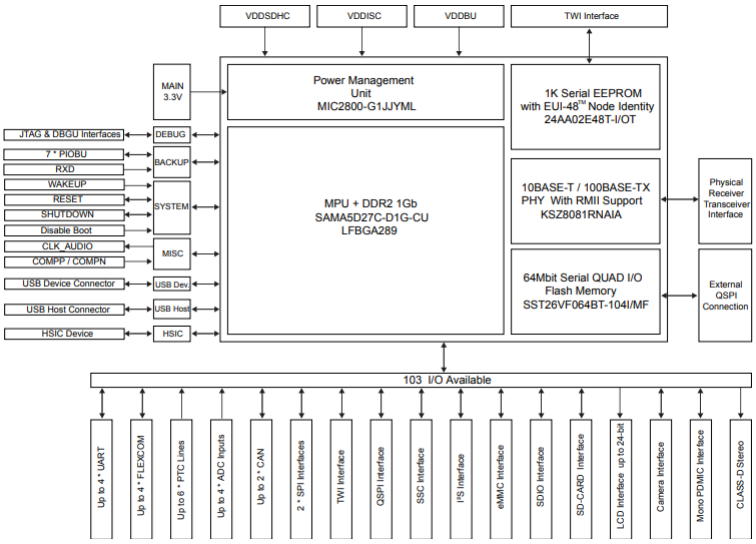


Slika 3.2: Moduli - Razvojna ploča SAMA5D27-SOM1-EK1

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 čip 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 3.3: Moduli - SAMA5D27-SOM1

kojim je moguće istaći neke druge njegove karakteristike. Ali ono sto 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.

3.1 Timestamping unit - TSU

Timestamping unit (TSU), je periferija unutar procesora koja pruža podršku prilikom vremenskog označavanja paketa koji učestvuju u razmeni poruka u PTP protokolu. TSU se sastoji od tajmera, koji je implementiran kao brojač, i registara u koji se smeštaju vremena u kojima PTP paketi prelaze granicu vremenskog označavanja. Hardverski prekid se registruje prilikom osvežavanja registara, pri čemu se osvežavanje registara dogadja svaki put kada tačno specificirani paketi predju preko interfejsa.

Tajmer unutar periferije je implementiran kao 94 bitni brojač, kod koga najviših 48 bita predstavlja broj sekundi, sledećih 30 bita predstavlja broj nanosekundi i najnižih 16 bita predstavlja vreme nivoa ispod nanosekunde. Donjih 46 bita se prebroje nakon što je izbrojena jedna sekunda, i takodje prijavljuje se prekid kada se izbroji jedna sekunda. Vrednost tajmera je moguće pročitati, kao i modifikovati preko APB interfejsa. Frekvencija tajmera se dobija od MCK procesora. U ovom slučaju MCK je 82MHz.

Veličina inkrementa tajmera je podesiva i može se kontrolisati preko re-

gistra GMAC_TI, gde je donjih 8 bita vrednost za koju će se uvećati vrednost tajmera u nanosekundama, i dodatnih 16 bita unutar registra GMAC_TISUBN, unutar koga se specificira vrednost inkrementa na nivou vremena ispod nanosekunde. Ukoliko je ostatak registra GMAC_TI nula, na svaku uzlaznu ivicu kloka koji dolazi na periferiju, vrednost će se uvećavati za vrednost donjih 8 bita GMAC_TI registra, plus vrednost GMAC_TISUBN registra. Registar GMAC_TISUBN dozvoljava rezoluciju od otprilike 15 femtosekundi.

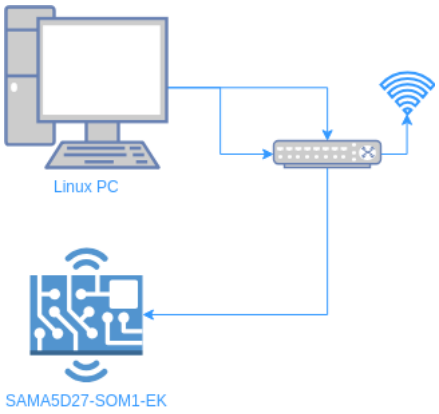
Takodje, registar GMAC_TI ima mogućnost podešavanja alternativnog broja nanosekundi, i to na bitima od 15 do 8, i na bitima od 23 do 16 broj ciklusa nakon koga će se iskoristiti vrednost alternativnog broja nanosekundi prilikom uvećanja vrednosti brojača. Ovaj deo periferije je u velikoj meri značajan za podešavanje tačnog vremena, i to u slučajevima kada se koriste drugi oscilatori za pogon procesora, koji daju drugačije vrednosti frekvencije kojom se uvećava vrednost brojača.

Ova periferija najviše utiče na tačno vreme koje je potrebno sinhronizovati, i takodje daje informacije o tačnom vremenu na uredjaju koji se koristi. Više detalja o samoj implemntaciji i korišćenju ove periferije biće dato u delu softverske implementacije.

Glava 4

Softverska implementacija

U ovom delu će biti opisana softverska implementacija projekta, sa datim rezultatima na kraju odeljka. Rezultati se odnose na razliku u satovima dva uredjaja u sistemu koji je napravljen kako bi se prikazala funkcionalnost protkola, i izvršila sinhronizacija dva uredjaja unutar oglednog sistema.



Slika 4.1: Arhitektura oglednog sistema

Ogledni sistem za implementaciju protokola preciznog vremena se sastoji od PC-a, na kome se nalazi Ubuntu 16.04 LTS operativni sistem, i razvojne ploče SAMA5D27-SOM1-EK1. Povezivanje izmedju razvojne ploče i PC-a je ostvareno preko standardnog UTP kabla (UTP - Unshielded-Twisted-Pair). Na PC-u se pokreće *open source* implementacija PTPd, odnosno deamon-a koji obavlja funkciju PTP protokola na standardnim operativnim sistemima. U ovom slučaju je izabran PTPd kao najjednostavnija i najdostupnija verzija ovakvog programa koja je napravljena za operativne sisteme bazirane na Linux kernelu.

Kako je u nekim trenucima potrebno da više uredjaja na mreži dobija

informacije potrebne za sinhronizaciju, sam PTP deamon je konfigurisan da radi u Master only modu, i takodje u Multicast modu. Što znači da je Linux PC u ovoj konfiguraciji oglednog sistema uvek Master i da svi uredjaji moraju da se sinhronišu na njegovo tačno vreme. Takodje, kako je PC jedini uredjaj na mreži koji daje takve informacije, postavljen je u Mulitcast mod, tako da svi uredjaji dobijaju informacije, dok samo neki koji mogu da prepoznaju odredjene poruke, mogu da odgovore i sinhronišu se u potpunosti. S tim u vezi, dodata su jos neka podešavanja samog deamon-a, čime je odredjen period za odgovor sa slave strane, odnosno da ukoliko ne dodje do odgovora u nekom roku, PC nastavi da šalje generičke poruke, kako bi pokrenuo proces sinhronizacije. Takodje, u svrhu provere da li je sve podešeno na najbolji način, vrši se praćenje svih poruka koje se dobijaju i skladište na PC-u kako bi kasnije mogle da se provere, ukoliko dolazi do grešaka na mreži. Kako se koristi PC, odnosno kompjuter opšte namene, koji je preko jednog od interfejsa, povezan na internet, sam PC dobija tačno vreme preko NTP-a, nakon čega to vreme postaje najbolje za lokalnu oglednu mrežu. Ukoliko bi se koristio specijalizovani kompjuter, koji ima mogućnost da tačno vreme dobije preko GPS-a, i takodje specijalizovane mrežne kartice, koje podržavaju PTP, mogla bi se dobiti mnogo veća tačnost, i time bi se povećala tačnost na slave uredjaju koji pokušava da se sinhronizuje. Implementacija PTP deamon-a na PC-u je preuzeta i korišćena u skladu sa uslovima i preporukama *open source* zajednice, i sem konfiguracije ništa nije promenjeno, kako bi se prilagodilo ovoj implementaciji.

Sa druge strane, implementacija na slave strani, u ovom slučaju na evaluacionoj ploči, je morala biti prilagodjena samom PTP protokolu, i morala je obezbediti mehanizme za korektno korišćenje PTP protokola.

Softverska implementacija je zamišljena tako da se može obezbediti lako dodavanje novih funkcionalnosti, kao i promena već postojećih funckionalnosti u vrlo kratkom vremenskom roku. S tim u vezi izabran je FreeRTOS kao operativni sistem koji ce biti osnova, i odabrana je Multi-Threaded arhitektura softvera koji implementira ovu funkcionalnost. Odabir ovog operativnog sistema i ove arhitekture je usko vezan sa svim što je potrebno iskoristiti kako bi se obezbedila korektna implementacija ovog protokola. Kao i dostupnost baze znanja za ovaj operativni sistem, i njegova *open source* priroda. Takodje, za IP stek je izabran lwIP stack, koji je pogodan za implementacije na Embedded uredjajima, i takodje je *open source* i ima široko dostupnu bazu znanja.

Implementacija ostavlja mogućnosti za dodavanje novih funkcionalnosti, i to kao dodavanje novih niti unutar već postojećeg sistema. Medjutim, prilikom dodavanja novih funkcionalnosti mora se voditi računa o već postojećim mehanizmima i kako nova funkcionalnost utiče na njih. S tim u vezi, imple-

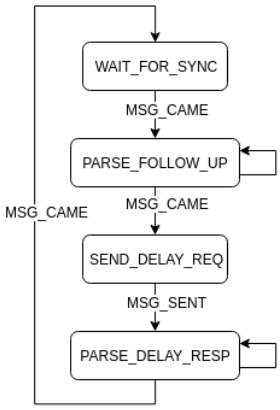
mentacija PTP mehanizma je jedna od niti unutar sistema, koja vodi računa o tačnom vremenu sistema, i ima najveći prioritet. Kako sama sinhrnoizacija nije vremenski zahtevna, i ne javlja se tako često, blokirajućim prekidima same niti, postignuto je da ostale funckionalnosti unutar sistema, ukoliko ih bude, mogu nesmetano da se izvršavaju, sve dok se ne javi potreba sistema za sinhrnoizacijom.

Kao što je već navedeno u poglavlju Hardverske implementacije, procesor koji se koristi, ima harversku podršku za PTP, i to u vidu TSU. Konfiguracijom modula GMAC unutar samog procesora može se omogućiti da procesor prepozna pakete koji stižu preko interfejsa i odredjene podatke iz paketa preuzme i prosledi na dalje korišćenje. S tim u vezi, sam modul mora se konfigurisati tako da se dozvole hardverski prekidi prilikom prepoznavanja odredjenih paketa. Ova funkcionalnost se dozvoljava prilikom inicijalizacije same ploče. Takodje, TSU je potrebno konfigurisati tako da se obezbedi puna funkcionalnost samog modula. TSU kao modul unutar GMAC-a dobija radni takt (eng. clock) kako bi se dobilo nezavisno vreme na ploči, i odnosu na ostatak sistema. Kao što je navedeno u odeljku hardverske implementacije, TSU je implementiran kao 94-bitni brojač, koji ima rezoluciju od oko 15 femtosekundi. Na inicijalizaciji celog sistema TSU dobija odredjeni radni takt, koji je u ovom slučaju 82 MHz. TSU se ne startuje sve dok se u polju konfiguracije ne dodeli inkrement koji je različit od nule. Takodje, unutar konfiguracije TSU postoje polja za specificiranje inkrementa ispod nanosekunde, koje je potrebno postaviti tako da se dobije što približnija reprezentacija realnog vremena.

Kako je frekvencija kojom broji TSU 82MHz, izračunavanjem se dolazi do toga da je za jedan period radnog takta prošlo 12.19512 ns. Pri čemu se cifre iza decimalne tačke periodično ponavljaju, i to ponavljanje je 19512. Nakon čega se odredjuje da inkrement nanosekundi unutar brojača TSU iznosi 12, dok je inkrement vremena ispod nanosekunde celobrojni umnožak broja 1951, i u ovoj implementaciji je uzet 9755. Odabirom ove frekvencije za TSU, i dobijanjem ovih vremena, potrebno je izvršiti još jednu modifikaciju TSU. TSU ima mogućnost i alternativnog inkrementa. Tj. zadavanjem vrednosti za alternativno inkrementiranje brojača, TSU može nakon odredjenog zadatog vremena, promeniti vrednosti kojima inkrementira brojač. U ovom slučaju, ispravljanje greške se vrši na svakih 83 ciklusa unutar TSU, i u tom ciklusu se dodaje 16ns na već postojeću vrednost unutar TSU brojača. Ovim podešavanjima modula, dobija se što vernija predstava realnog vremena. Tj. dobija se da ono što dobijamo od oscilatora, odgovara stvarnim vrednostima vremena. Naravno, zbog vrednosti koje dolaze sa oscilatora i podešavanja PLL-ova unutar samog procesora, mora postojati odredjena greška, koja se može tolerisati, i koja se može smanjiti na odredjene načine. Konfiguracija

GMAC modula podrazumeva i registrovanje prekida za Sync i Delay_Req pakete koji stižu na interfejs, čime se dobijaju vremena koja su potrebna. Nakon ispravne konfiguracije, dobijaju se prekidi na koje stižu paketi, i iz kojih je moguće iščitati vremena. Na Sync i Delay_Req pakete, i prekide koji javljaju, moguće je iščitati trenutnu vrednost vremena koje se nalazi u TSU, što je bitno za proces sinhronizacije. U skladu sa arhitekturom softverske implementacije, vremena koja se dobijaju, prosledjuju se u red sa porukama.

Protokol tačnog vremena odredjuje i to da slave uredjaj mora prepoznati pakete koji stižu, što se postiže hardverskom podrškom za neke od paketa. Kao i odgovoriti na odredjene pakete, što se postiže implementiranjem mašine stanja koja vodi računa o trenutnom stanju uredjaja u smislu sinhronizacije. S tim u vezi, mašina stanja ostaje u istom, početnom stanju, sve dok ne stigne prvi paket koji je bitan za proces sinhronizacije, Sync paket. Nakon čega se iz prekida dobija poruka sa trenutnom vrednosti vremena u TSU, i prelazi se u sledeće stanje. Sledeće stanje u mašini stanja predstavlja stanje čekanja na sledeći paket, tj. Follow_up paket. Unutar Follow_up paketa se nalazi vreme kada je poslat Sync frame, i potrebno je raspakovati taj paket, i uzeti to vreme. To se obavlja unutar ovog stanja u mašini stanja, nakon čega se prelazi u sledeće stanje, i slanje uzetog vremena u red sa porukama. Sledeće stanje Delay_Req_Send, implementira slanje Delay_Req paketa sa slave uredjaja na master, u skladu sa specifikacijom protokola. Slanje Delay_Req paketa, inicira prekid unutar GMAC, u kome se uzima tačno vreme kada je paket napustio uredjaj. To vreme se takodje šalje u red sa porukama. Nakon čega se prelazi u sledeće stanje, odnosno čekanje prijema Delay_Resp paketa. Nakon što Delay_Resp paket pristigne, i raspakuje se na odgovarajući način, dobija se četvrto i poslednje vreme koje je potrebno za sinhronizaciju. I ono se šalje u red sa porukama.



Slika 4.2: Mašina stanja

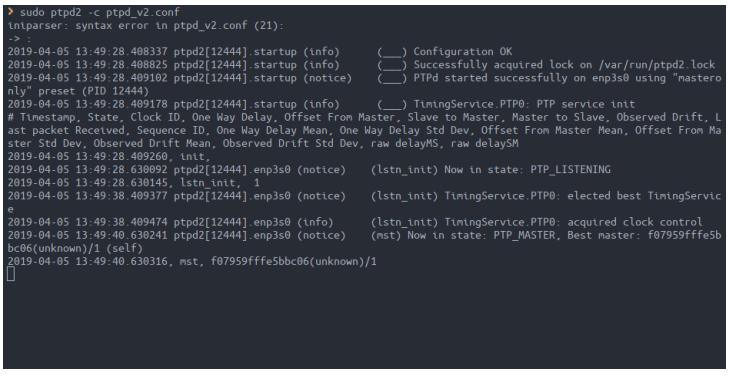
Nakon ovoga, sva vremena koja su potrebna za sinhronizaciju su tu, i može se izvršiti promena TSU, tako da računanje vremena odgovara vremenu na master uredjaju. Izračunavanjem vrednosti kojim se treba promeniti TSU, koriste se već ugradjeni makroi i mehanizmi kako bi se iskoristile funkcionalnosti koje nudi TSU. S tim u vezi modifikuje se trenutna vrednost koju broji TSU, i takodje se modifikuju vrednosti za delove vremena ispod nanosekunde.

Za ovakav način implementacije, potrebna je samo jedna nit, i to ona koja će da vodi računa o mašini stanja. Unutar ove niti, odnosno mašine stanja, implementirani su blokirajući pozivi za dobijanje vremena iz reda sa porukama, čime se smanjuje vreme i resursi koje procesor troši na opsluživanje ove niti. Takodje, ovakav način implementacije dozvoljava implementaciju ostalih niti koje mogu izvršavati ostale funkcionalnosti sistema, s obzirom da ova nit nije zahtevna i javlja se relativno retko potreba za sinhronizacijom.

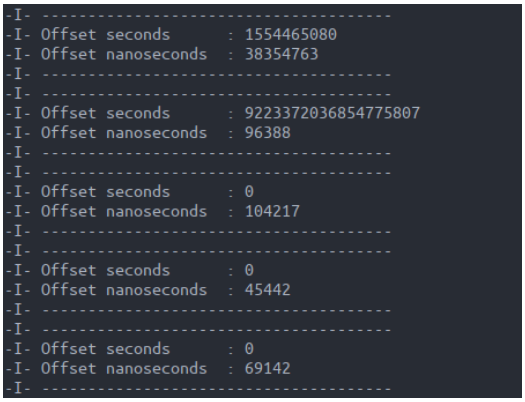
4.1 Rezultati

U ovom delu su dati rezultati trenutnog izvršavanja i sinhronizacije u oglednom sistemu.

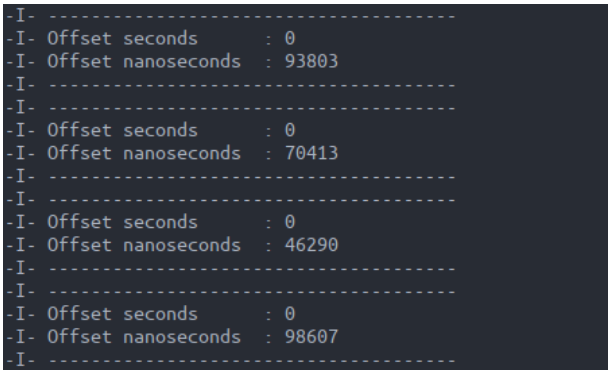
U neopterećenoj mreži, tj. u mreži 1 na 1, PC i razvojna ploča, dobijaju se rezultati sinhrnoizacije od otprilike 0.06ms, što je više nego zadovoljavajuće za ovakav tip sinhronizacije. Takodje, postoji akumulacija greške koja je uzeta u obzir, i ona se ogleda u tome da u nekim trenucima, uredjaj koji se sinhronizuje izgubi sinhronizaciju, nakon čega se vraća u roku od jednog ciklusa sinhronizacije. Ova greška se pojavljuje usled prekoračenja (eng. overflow) dela brojača za vremena ispod nanosekunde.



Slika 4.3: Prikaz pokretanja alata PTPd, i postavljanje PC-a za master u oglednom sistemu



Slika 4.4: Početak sinhronizacije



Slika 4.5: Sinhronizacija nakon nekog vremena

Na slikama je dat ispis sa konzole razvojne ploče. I predstavlja razliku izmedju sata na razvojnoj ploči, i sata na PC-u. Iz priloženih slika se vidi smanjenje razlike izmedju dva sata, ali i u kom trenutku se dešava prekoračenje, i kako se sam uredjaj vraća iz tog stanja.

4.2 Predlozi za poboljšanja

Unutar ovog dela biće dati predlozi za poboljšanja već postojećeg mehanizma, a i predlozi za poboljšanja algoritma koji je implementiran.

Naime, kako je u ovoj implementaciji korišćena verzija 1 originalnog IEEE 1588 standarda, na lwIP steku, moguće je pružiti podršku za verziju 2 standarda. Korišćenjem druge verzije standarda, poboljšala bi se sinhronizacija i pružila podrška za različite softverske i hardverske implementacije ostatka sistema, odnosno integracije razvojne ploče u druge sisteme koji ne podržavaju

ovu verziju standarda. Takodje, na ovaj način bi se pružila podrška za ostali hardver koji podržava standard, čime bi se povećala i tačnost sinhronizacije.

Takodje, kao još jedan predlog za poboljšanje može se navesti i korišćenje drugačije softverske implementacije standarda na PC-u, ili korišćenje drugog operativnog sistema na samoj razvojnoj ploči. Pri čemu je moguće vršiti uporednu analizu koliko zaista sam operativni sistem utiče na sinhronizaciju i propagaciju paketa kroz stek i operativni sistem.

I kao poslednji predlog za poboljšanje, moguće je implementirati IEEE 802.1 AS koji specificira korišćenje IEEE 1588 standarda, ali bez korišćenja specijalizovanog IP steka. Implementacijom ovog standarda, prepoznavanje i formiranje paketa se spušta u OSI modelu na Data Link sloj, čime se ukida propagacija kroz stek, i time povećava tačnost sinhronizacije. Medjutim, u ovoj implementaciji standarda, vrlo je bitno da postoji podrška unutar drajvera same periferije, i to može dosta zavisiti od softverskog paketa, kao i operativnog sistema koji se koristi.

Zaključak

Cilj ovog rada je prikaz kompletnog postupka implementacije vremenske sinhronizacije u Namenskim računarskim sistemima, koja je sve više i više prisutna i zahtevana. Zamisao je da predložena implementacija bude modularna, i lako iskorišćena bez ikakvih promena.

Pre svega, dat je uvod u sam model komunikacije, i prikaz samog protokola koji je implementiran. Utvrdjeno je da je Precision Time Protocol veoma moćan, i trenutno zbog sve prisutnije potrebe za automatizacijom, sve više u upotrebi. Ono što je vrlo značajno je da je protokol u upotrebi trenutno, i da se i dalje razvija, što je vrlo bitno. Takodje, implementacija samog protokola dosta zavisi od hardvera, ali i softvera koji se koristi, što dosta utiče na sam sistem u kom je potrebno ostvariti sinhrnoizaciju. S tim u vezi potrebno je dobro poznavati protokol, ali i sam sistem u kom je potrebno sve implementirati, tako da se protokol iskoristi na najbolji način.

Korišćenje FreeRTOS operativnog sistema je omogućilo da se pokaže kako se jedna ovakva funcionalnost može biti implementirana i na dosta jednostavnijim sistemima, koji imaju dosta svojih prednosti, naročito u vremenski kritičnim aplikacijama, kao što je u ovom slučaju. Takodje, sam FreeRTOS ima dosta mogućnosti koje u ovoj implementaciji nisu iskorišćene. Naravno, u skladu sa implementacijom dat je pregled samog operativnog sistema, koji naravno može biti dosta detaljniji, kao i opis lwIP-a. Medjutim, ovde je dat osnovni pregled, koji predstavlja uvid u mogućnosti koje poseduju i lwIP i FreeRTOS.

Sama razvojna ploča je izabrana zbog mikrokontrolera koji se nalazi na njoj, i zbog hardverske podrške koju ima izabrani mikrokontroler. Kao i široko dostupne baze znanja, vezane za sam mikrokontroler, kao i softverske podrške koju pruža sama kompanija Atmel, kao i *open source* zajednica, koja podržava ovaj hardver.

Ovaj rad se može posmatrati kao primer inicijalne vremenske sinhronizacije u jednom namenskom računarskom sistemu. Takodje, može se posmatrati kao i deo većeg sistema, kojim bi zadati sistem dobio potrebnu vremensku sinhronizaciju. Takodje, ovaj rad predstavlja kombinaciju modu-

larnog projektovanja jednog namenskog sistema u koji je moguće dodavanje novih funkcionalnosti, i to na vrlo lak način. Sama priroda implementacije koja se bazira na *open source* projektima, omogućava da korišćenje već postojeće implementacije, kao i proširivanje i unapredjenje može biti lako i brzo uradjeno, kao i da nema potrebe za odredjenim komercijalnim softverskim modulima ili komercijalnim alatima.

Literatura

[1] IEEE 1588-2008 - IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems;

dostupno online:

https://standards.ieee.org/findstds/standard/1588-2008.html

[2] IEEE 1588 Unplugged – An introduction to IEEE 1588;

dostupno online:

https://www.rtaautomation.com/technologies/ieee-1588/

[3] P802.1AS-Rev – Timing and Synchronization for Time-Sensitive Applications;

dostupno online:

https://1.ieee802.org/tsn/802-1as-rev/

[4] FreeRTOS documentation;

dostupno online:

https://www.freertos.org/Documentation/161204_Mastering_the_FreeRTOS_Real_Time_Kernel-A_Hands-On_Tutorial_Guide.pdf

https://www.wikiwand.com/en/FreeRTOS

[5] lwIP documentation;

dostupno online:

https://lwip.fandom.com/wiki/LwIP_Wiki

https://savannah.nongnu.org/projects/lwip/

[6] Microchip documentation;

dostupno online:

http://ww1.microchip.com/downloads/en/DeviceDoc/SAMA5D27-SOM1-Kit1-User -Guide-DS50002667B.PDF

https://www.digikey.com/eewiki/display/linuxonarm/SAMA5D27-SOM1-EK1

https://www.microchip.com/developmenttools/ProductDetails/atsama5d27-som1-ek1

http://ww1.microchip.com/downloads/en/DeviceDoc/SAMA5D2-Plus-DDR2- LPDDR2-System-in-Package-(SIP)-60001484b.pdf

[7] Precision Time Protocol (PTP/IEEE-1588);

https://endruntechnologies.com/pdf/PTP-1588.pdf

[8] PTPd documentation;

dostupno online:

https://github.com/ptpd/ptpd

http://manpages.ubuntu.com/manpages/bionic/man8/ptpd.8.html