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

Slike

Glava 1

Apstrakt

Glava 2

Uvod

Sinhronizacija postaje neophodna kada su uredjaji koji rade zajedno na odredjenoj udaljenosti moraju raditi u vezi jedni sa drugima. U ovim scenarijima, 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.

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 unutrasnju strukturu uredjaja ili njihovu tehnologiju. Cilj ovog modela je da se postigne kompatibilnost razlicitih komunikacionih sistema sa standardnim protokolima komunikacije. OSI model razdvaja komunikacione sisteme u apstraktne slojeve. Originalna verzija modela ima sedam slojeva.

Sloj unutar modela sluzi sloj iznad njega, i koristi sloj ispod njega u hijerarhiji. Na primer, sloj koji postize komunikaciju preko mreze bez gresaka, sluzi aplikacijama iznad koje ga koriste, i to dok poziva jednostavne funkcije za prijem i predaju paketa na mrezi. Dve instance istog sloja su vizualizovane tako sto 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.

[lazarc] slika svih slojeva u osi modelu

Na svakom nivou N, dva entiteta na komunikcionim uredjajima razmenjuju jedinice protokola (PDU - protocol data units) pomocu sloja N protokola. Svaki PDU sadrzi podatke od interesa (payload) (SDU - service data unit), zajedno sa zaglavljima koji odgovaraju protokolu.

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

- Podaci koji se prenose se formiraju na najvisem sloju u uredjaju koji predaje podatke na mrezi (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 uredjaju koji prima podatke.
- Na strani prijemnog uredjaja se podaci prenose od najnizeg sloja u modelu, do najviseg, gde se serije SDU struktura uspesno obradjuju, pri cemu se skidaju zaglavlja sa svakog sloja, dok se ne dostigne najvisi sloj u modelu, nakon cega 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 uredjaja 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 uredjaje. Kontrola brzine bitova se vrsi 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 uredjaja 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 uredjaja. 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 nacina na koji uredjaji 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 sinhrone ili asinhrone serijske linije.

3.1.3 Sloj 3: Mrezni sloj (Network Layer)

Mrezni sloj obezbedjuje funkcionalno i proceduralno sredstvo prenosa sekvenci podataka promenljive duzine, koji se nazivaju jos i paketi, od jednog cvora do drugog, poveznih 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, omogucavajuci mrezi da nadje nacin da isporuci poruku odredisnom cvoru, eventualno ga usmeravajuci 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. Mrezni 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 protokli 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 isporuceni prijemnom uredjaju. Transportni sloj takodje omogucava i potvrdu uspesnog prenosa podataka i salje naredne podatke ako nije doslo do greske 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 krace poruke kako bi se lakse prenele preko nizih slojeva u modelu.

OSI model definise 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 mreze, slicne Internetu). Klasa 0 (TP0) nema mogucnost oporavka od greske i bila je dizajnirana za koriscenje na mreznim slojevima koji pruzaju konekciju bez gresaka. Klasa 4 (TP4) je najbliza TCP, iako TCP sadrzi neke funkcije koje se u OSI modelu dodeljuju visim slojevima. Takodje, sve klase u OSI modelu omogucavaju brzu upotrebu podataka i ocuvanje granica podataka ## [lazarc] [prevod] ##. Detalje karakteristika svih klasa prikazane su u sledecoj tabeli:

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

3.1.5 Sloj 5: Sloj sesije (Session Layer)

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

3.1.6 Sloj 6: Sloj prezentacije (Presentation Layer)

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

Ovaj sloj obezbedjuje nezavisnost od predstavljanja podataka u razlicitim aplikacijama i mreznim formatima. Sloj prezentacije pretvara podatke u oblik koji prihvata zadata aplikacija. Ovaj sloj formatira podatke koji se salju preko mreze. Ponekad se naziva i sintaksni sloj. Takodje, moze ukljucivati 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 uredjaja 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

Internet protocol suite (TCP/IP) 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 uredjaja 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 uredjajima 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 odrzavanje stanja i sveobuhvatne inteligencije na ivice, i pretpostavio da internet koji je povezivao krajeve nije zadrzao nikakvo stanje i koncentrisao se na brzinu i jednostavnost. Potrebe za zastitnim zidovima (##[lazarc] firewalls ##), prevodiocima mreznih adresa, kesiranju veb sadrzaja i slicno, su izazvale promene u ovom principu.

Princip robusnosti kaze: "Uopsteno, implementacija mora biti konzervativna u ponasanju prilikom slanja i liberalna u svom ponasanju prilikom prijema. Sto znaci, da mora biti oprezna pri slanju dobro formiranih paketa (##[lazarc] datagrams ##), ali mora prihvatiti bilo koji paket koji moze protumaciti. (na primer, ne primecivati tehnicke greske gde nije poznato sta ih uzrokuje.)". Đrugi deo principa je gotovo jednako vazan: softver na drugim uredjajima moze sadrzati razlike koji cine nerazumnim da se iskoriste legalne, ali nejasne karakteristike protokola".

Enkapsulacija se koristi za obezbedjivanje apstrakcije protokola i usluga. Enkapsulacija je obicno uskladjena sa podelom unutar protokola na slojeve funkcionalnosti. Uopsteno, aplikacija (najvisi 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 cetvoroslojevnom protokolu. Sto je u upotrebi i danas. I oni su unutar protokola korisceni u istom redosledu u kom ce i ovde biti navedeni.

- Aplikativni sloj (Application layer) Aplikativni sloj je opseg unutar kog aplikacije kreiraju korisnicke podatke i prenose ove podatke drugim aplikacijama na istom ili drugom uredjaju (hostu). Aplikacije ili procesi, koriste usluge koje pruzaju donji slojevi, posebno transportni sloj koji obezbedjuje pouzdane ili nepouzdane veze ka drugim procesima. Komunikacione partnere karakterise arhitektura aplikacije, kao sto su model klijent-server i umrezavanje ravnopravnih korisnika. Ovo je sloj u kome su svi protokoli viseg nivoa, kao sto su SMTP, FTP, SSH, HTTP, i td. Procesi se adresiraju preko portova koji u sustini predstavljaju usluge.

- Transportni sloj (Transport layer) Transportni sloj obavlja komunikacije izmedju domacina, ili domacina na istim ili razlicitim uredjajima (hostovima) i na lokalnoj mrezi ili udaljenim mrezama razdvojenim od rutera. Ovaj sloj obezbedjuje kanal za komunikacione potrebe aplikacija. UDP je osnovni protokol transportnog sloja koji pruza nepouzdanu uslugu sa paketima. Protokol za kontrolu prenosa (TCP) omogucava kontrolu protoka, uspostavljanje veze i pouzdan prenos podataka.

- Internet sloj (Internet layer) Internet sloj razmenjuje pakete preko

mreze. Ovaj sloj obezbedjuje uniforman mrezni interfejs koji skriva stvarnu topologiju (raspored) osnovnih mreznih veza. Zbog toga se naziva i slojem koji uspostavlja rad na mrezi. Zaista, ovaj sloj definise i uspostavlja internet. Ovaj sloj definise strukture adresiranja i usmeravanja koje se koriste za pakete TCP/IP protokola. Primarni protokol u ovom opsegu je Internet protokol, koji definise IP adrese. Njegova sledeca funkcija u usmeravanju je da prenosi pakete pakete na sledeci IP ruter koji ima vezu sa mrezom blize kranjem odredistu podataka.

- Sloj veze (Link layer) Sloj veze definise metode umrezavanja u okviru lokalne mrezne veze na kojoj uredjaji (hostovi) komuniciraju bez rutera u medjukomunikaciji. Ovaj sloj ukljucuje protokole koji se koriste za opisivanje topologije lokalne mreze i potrebnih interfejsa da bi se zavrsio prenos paketa sa Internet sloja na ostale uredjaje.

Slojevi protokola blizu vrha su logicno blizi korisnickoj aplikaciji, dok su oni blize dnu logicki blizi fizickom prenosu podataka. Pregled slojeva u smislu pruzanja i koriscenja usluge je metoda aplikacije koja izoluje gornje slojeve protokola od detalja kao sto je prenos bitova, detekcije loseg prenosa, na primer, dok su nizi slojevi izolovani od detalja aplikacije i principa rada aplikacije.

3.3 Precission Time Protocol

Protokol preciznog vremena (Precision Time Protocol (PTP)) je protokol korišćen za sinhronizaciju satova preko kompjuterske mreže. U lokalnoj kompjuterskoj mreži (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" i 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.

3.3.1 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. #Obični uredjaj# 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. #Sporedni uredjaj# 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. #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 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.

mora slika ovde kako izgleda arhitekura tacno

Simplifikovano, PTP sistem se sastoji od Ordinary clocks #Obicnog uredjaja# poveznih na jednostavnu mrežu, i bez Boundary clocks #Sporednih uredjaja#. Grandmaster se bira, i svi ostali uredjaji se direktno sinhronišu na njega.

IEEE 1588-2008 predstavljaju Clock koji je povezan sa mrežnom opremom koja prenosi PTP poruke. Transparent clock #Transparentni uredjaj# 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 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 izračuna od primljenog PTP vremena.

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

3.3.2 Detalji protokola

Sinhronizacija i obrada u PTP sistemu se postiže 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 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 da mere 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 i Best master clock algorithm 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.
- 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 opste poruke. Event poruke su vremenski kritične i to u preciznosti predaje i prijema preciznosti vremenskih pečata (TIMESTAMPs) i direktno utiču na distribuciju preciznosti vremena. (JOS JEDNOM POGLEDAJ OVAJ PREVOD). Sync, Delay_Req, Pdelay_Req i Pdelay_resp su poruke dogadjaja. Opšte poruke su ubičajene 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.

3.3.3 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) (#pogldedaj opet ovaj prevod#), dok je u IEEE 1588-2008 to uvedeno kao opcija.

3.3.4 Algoritam najboljeg sata

BMC *Bestmasterclockalgorithm* 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 (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 specifičan statički dizajniran prioritet svakom satu pre svega odredjujući prioritet medju njima. Manje vrednosti prioriteta označavaju veći prioritet.
- Klasa: Svaki sat je član odredjene klase, svaka klasa dobija svoj prioritet.
- Preciznost: Preciznost izmedju sata i 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 (## [lazarc] [prevod]##) 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 pormenu Master sata. Kada trenutni Master prepozna bolji sat, tada Master sat zaustavlja emitovanje poruka za sinhronizaciju (Sync Messages), ili poruke ograšavanja (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.

3.3.5 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 (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:

$$o(t) = s(t) - m(t) \tag{3.1}$$

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 (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.

[lazarc] [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 dobaviti 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 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 sadrzi 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:

$$T_1' - T_1 = \tilde{o} + d \tag{3.2}$$

i

$$T_2' - T_2 = -\tilde{o} + d \tag{3.3}$$

Odakle je:

$$\tilde{o} = \frac{1}{2}(T_1' - T_1 - T_2' + T_2) \tag{3.4}$$

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. 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 šalju ili primaju poruke. Stepen primene ovih pretpostavki utiče na to koliko će se dobro sinhronizovati dva uredjaja.

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

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 odrzavao projekat, pisan je uglavnom u C programskom jeziku, sa izuzetkom da su neke funkcionalnosti napisane u asembleru, gde je to bilo potrebno, i to uglavnom rutine u Scheduler-u (Rasporedjivacu??? ##[lazarc] [prevod] ##) koje su specificne za samu arhitekturu.

FreeRTOS omogucava 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 dinamicki 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 sussednih memorijskih blokova;
- alternativa za jos kompleksiniju semu koja ukljucuje spajanje susednih memorijskih blokova koja omogucava da hip (HEAP) bude podeljen na vise memorijskih delova;

- i na kraju C biblioteka za alociranje i oslobadjanje sa zastitom medjusobnog iskljucivanja.

Unutar FreeRTOS-a ne postoji ni jedan od slozenijih svojstava operativnih sistema koji se uobicajeno 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 izvrsavanja. 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. Ovaj 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 izvrsavanju, 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 vrmena (ili rasporedjivac(##[lazarc] [prevod]##) realnog vremena) na koji se nadogradjuje 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 izvrsavati u jednom trenutku. Kernel odlucuje koja nit se izvrsava 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 izvrsavanjem. Ovim bi se osiguralo da niti koje imaju strozije zahteve, imaju prioritete izvrsavanja i pristupa resursima nad ostalim nitima, ali odluke za dodelu izvrsavanja 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 izvrsavanja 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 evoluira 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 takskovi dobro definisani kao nezavisni moduli sa cistim interfejsima, mogu biti testirani nezavnisno.

- **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 popunosti moze prebaciti na opkretanje dogadjajima (event driven programming), i time bi se ustedelo 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 izvrsavanje sa jednog taska na drugi. Kako god, i

aplikacije koje ne koriste RTOS normalno ukljucuju neku formu prekida.

- **Idle time (##[lazarc] [prevod]##):** Idle task je task koji se automatski kreira prilikom startovanja Rasporedjivaca (##[lazarc] [prevod]##). I izvrsava se kad nema taskova unutar aplikacije koji bi se izvrsavali. Ovaj task se moze koristiti za merenje procesorske moci koja se trosi, za izvrsavanje provera u pozadini, ili da jednostavno pokrene rezim smanjene potrosnje u sistemu.

- **Upravljanje snagom (Power management):** Efikasnost koja se dobija koriscenjem RTOS-a dozovoljava procesoru da provede vise vremena u rezimu smanjenje potrosnje. Potrosnja se moze znacajno smanjiti time sto procesor odlazi u rezim smanjenje potrosnje kad god je pokrenut Idle task. FreeRTOS takodje ima i specijalni tick-less mod, u kome procesor odlazi u rezim smanjene potrosnje na duze.

- **Fleksibilno upravljanje prekidima (Flexible interrupt handling):** Upravljanje prekidima se moze drzati veoma kratko tako sto 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 postici mesanjem periodicnog, kontinualnog i procesiranja pokretanog dogadjajima. Pored toga, ispunjavanje strogih i manje strogih zahteva za realnim vremenom u aplikacijama moze se postici izborom odgovoarajucih 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 inzenjera 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 moze 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 funcionalnost 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.

- *Mrezni 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 mrezni 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 (štreaming")
- *Protokoli visokog nivoa: (High-level protocols)*
  - **DHCP**: dobijanje IP adrese sa podrskom servera
  - **AUTOIP**: dobijanje IP adrese bez podrske 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 pruza nacin za obicno, sekvencijalno programiranje koje koristi lwIP stek (##[lazarc] stack ##). Model izvrsavanja je baziran na blokirajucoj otvori-procitaj-upisi-zatvori paradigmi. Kako je TCP/IP stek baziran na dogadjajima, TCP/IP kod i aplikativni program, moraju da se pozivaju sa razlicitim kontekstima izvrsavanja, u razlicitim nitima.

Prilikom mesanja sekvencijalnog i širovog"API-a u programima, treba biti pazljiv. Funkcije koje pripadaju nesekvencijalnom API-u u stvari mogu biti pozvane iz glavne tcpip_thread niti. Takodje, 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 izvrsavanja unutar tcpip_callback rutine).

Jos neke cinjenice o API-ima koje uticu na koriscenje lwIP steka:

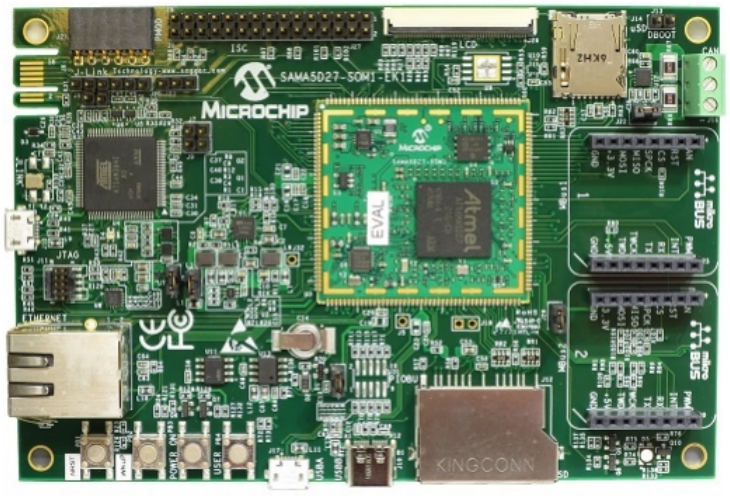
- **netconn- i raw-API su samo unutar lwIP-a :** kod koji koristi ovaj API se ne moze koristiti u drugim stekovima koji imaju iste mogucnosti kao lwIP (na primer uIP i td.)
- **socket API** je u suprotnosti sa gore navedenom stavkom, napravljen je tako da je kompatibilan i moze se koristiti u drugim 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 nacin, moze biti teze koriscenje.
- **raw API** daje bolje performanse kako ne zahteva promenu izvrsavanja programskih niti.
- **raw API i netconn API** podrzavaju zero-copy [1] kako za TX tako i za RX. (kako za predaju, tako i za prijem)

[1] predstavlja operaciju pri kojoj procesor ne vrsi kopiranje podataka iz jedne memorijske oblasti u drugu. Ovo se cesto koristi kako bi se sacuvali ciklusi procesora i propusnog opsega memorije prilikom prenosa kontinualnih podataka (datoteka, fajlova) preko mreze.

Glava 5

Hardverska implementacija

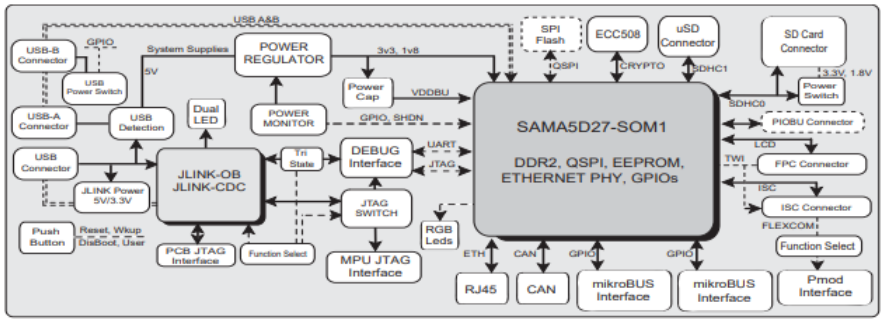
Razvojno okruženje koje je korišćeno za hardversku implementaciju je razvojna ploča SAMA5D27-SOM1-EK1 proizvodjača Microchip. Na ploči se nalazi SAMA5D27 SOM (System on Module) modul koji je ključan za implementaciju. Na modulu se nalazi SAMA5D27-D1G-CU SIP (System in Package) koji sadrži 1 Gbit DDR2 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 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 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-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.

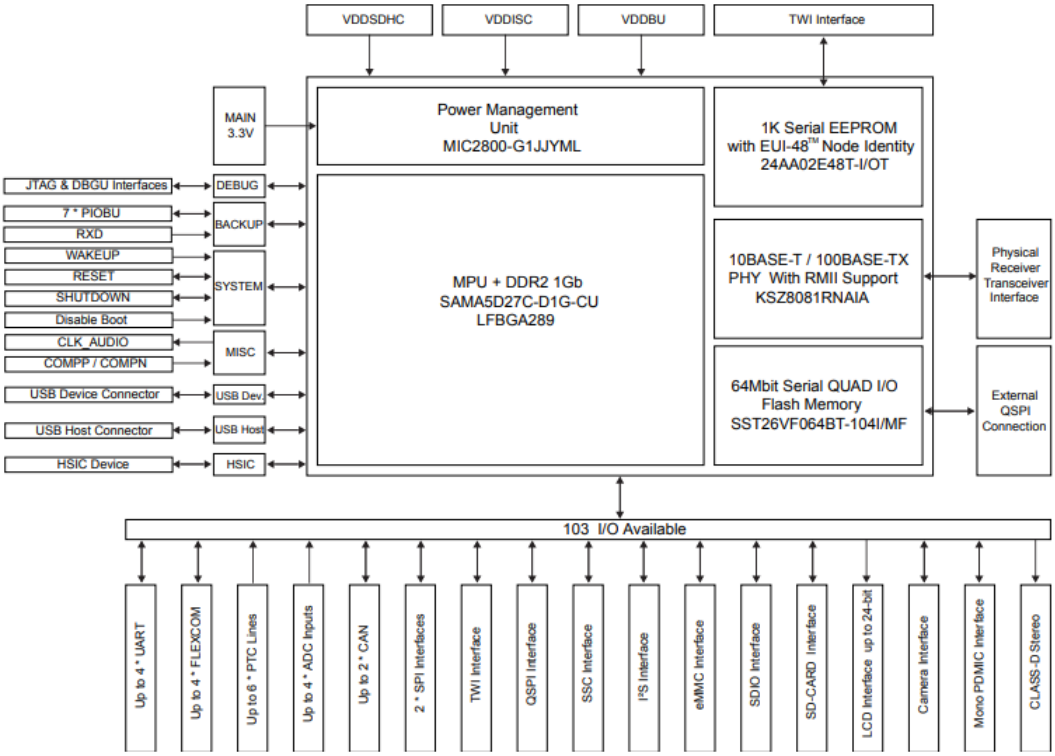


Slika 5.2: Moduli - Razvojna ploča SAMA5D27-SOM1-EK1

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 š irok 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 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, cime 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 izmedju dva uredjaja 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 uredjajima u lokalnoj mrezi (Multicast). GMAC periferija je dizajnirana tako da prepoznaje poruke koje su ključne za PTP protokol.

Kao što je već navedeno, sinhronizacija izmedju dva uredjaja se izvodi u dva stadijuma, odredjivanje razlike (offset) izmedju dva sata (na strani Master i Slave uredjaja), 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 odredjuju tačno vreme kada je poruka stigla, i kada je poruka poslata. Što je ključno za odredjivanje 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 uredjaja. Podrška ovom protokolu u hardveru se ogleda u postojanju TSU(Timestamping unit) periferije koja se sastoji od tajmera i registara u koje se smeštaju tacna 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 implemntiran 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. Takodje, prijavljuje se hardverski prekid nakon sto se izbroji jedna sekunda. Vrednosti tajmera, kao i parametri za njegovo (##[lazarc] CHECK! ##) koji je glavni takt unutar procesora, i njime se takodje moze manipulisati. Promene podešavanja tajmera unutar TSU su od velike važnosti za tačnu sinhronizuju dva uredjaja. 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 6

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.

Prikaz sistema, nparaviti sliku gde se vide Linux PC i ploca

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 se pravljena za operativne sisteme bazirane na Linux-u.

Kako je u nekim trenucima potrebno da vise uredjaja na mrezi dobija informacije potrebne za sinhronizaciju, sam PTP deamon je konfigurisan da radi u Master only modu, i takodje u Multicast modu. Sto znaci da je Linux PC u ovoj konfiguraciji oglednog sistema uvek Master i da svi uredjaji moraju da se sinhronisu na njegovo tacno vreme. Takodje, kako je PC jedini uredjaj na mrezi 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 sinhronisu se u potpunosti. S tim u vezi, dodata su jos neka podesavanja samog deamon-a, cime je odredjen period za odgovor sa slave strane, odnosno da ukoliko ne dodje do odgovora u nekom roku, PC nastavi da salje genericke poruke, kako bi pokrenuo proces sinhronizacije. Takodje, u svrhe provere da li je sve podeseno na najbolji nacin, vrsi se pracenje svih poruka koje se dobijaju i skladiste se na PC-u kako bi kasnije mogle da se provere, ukoliko dolazi do gresaka na mrezi.

Kako se koristi PC, odnosno kompjuter opste namene, koji je preko jednog od interfejsa, povezan na internet, sam PC dobija tacno vreme preko NTP-a, nakon cega to vreme postaje najbolje za lokalnu oglednu mrezu. Ukoliko bi se koristio specijalizovani kompjuter, koji ima mogucnost da tacno vreme dobije preko GPS-a, i takodje specijalizovane mrezne kartice, koje podrzavaju PTP, mogla bi se dobiti mnogo veca tacnost, i time bi se povecala tacnost na slave uredjaju koji pokusava da se sinhronise.

Implementacija PTP deamon-a na PC-u je preuzeta i koriscena u skladu sa uslovima i preporukama Open Source zajednice, i sem konfiguracije nista nije promenjeno, kako bi se prilagodilo ovoj implementaciji.

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

Softverska implementacija je zamisljena tako da se moze obezbediti lako dodavanje novih funkcionalnosti, kao i promena vec postojecih 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 sto 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 stack je izabran lwIP stack, koji je pogodan za implementacije na Embedded uredjajima, i takodje je Open Source i ima siroko dostupnu bazu znanja.

Implementacija ostavlja mogucnosti za dodavanje novih funkcionalnosti, i to kao dodvanje novih thread-ova unutar vec postojeceg sistema. Medjutim, prilikom dodavanja novih funkcionalnosti mora se voditi racuna o vec postojecim mehanizmima i kako nova funkcionalnost utice na njih. S tim u vezi, implementacija PTP mehanizma je jedna od niti unutar sistema, koja vodi racuna o tacnom vremenu sistema, i ima najveci prioritet. Kako sama sinhrnoizacija nije vremenski zahtevna, i ne javlja se tako cesto, blokirajucim prekidima same niti, postignuto je da ostale funckionalnosti unutar sistema, ukoliko ih bude, mogu nesmetano da se izvrsavaju, sve dok se ne javi potreba sistema za sinhrnoizacijom.

Kao sto je vec navedeno u poglavlju Hardverske implementacije, procesor koji se koristi, ima harversku podrsku za PTP, i to u vidu TSU (Time stamping unit). Konfiguracijom modula GMAC unutar samog procesora moze se omoguciti da procesor prepozna pakete koji stizu preko interfejsa i odredjene podatke iz paketa preuzme i prosledi na dalje koriscenje. 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 ploce. Takodje, TSU je potrebno konfigurisati tako da se obezbedi puna funkcionalnost samog modula. TSU kao modul unutar GMAC-a dobija klok (## [lazarc] clock) kako bi se dobilo nezavisno vreme na ploci, i odnosu na ostatak sistema. Kao sto je navedeno u odeljku hardverske implementacije, TSU je implementiran kao 94-bitni brojac, koji ima rezoluciju od oko 15 femto sekundi. Na inicijalizaciji celog sistema TSU dobija odredjeni klok (## [lazarc] clock), koji je u ovom slucaju 82 MHz. TSU se ne startuje sve dok se u polju konfiguracije ne dodeli inkrement koji je razlicit od nule. Takodje, unutar konfiguracije TSU postoje polja za specificiranje inkrementa ispod nanosekunde, koje je potrebno setovati tako da se dobije sto priblizniji reprezentacija realnog vremena.

Kako je frekvencija kojom broji TSU 82MHz, izracunavanjem se dolazi do toga da je za jedan period kloka (## [lazarc] clock) proslo 12.19512 ns. Pri cemu se cifre iza decimalne tacke se periodicno ponavljaju, i to ponavljanje je 19512. Nakon cega se odredjuje da inkrement nanosekunda unutar brojaca TSU iznosi 12, dok je inkrement vremena ispod nanosekunde celobrojni umnozak broja 1951, i u ovoj implementaciji je 9755, odnosno 5 puta ostatka. Odabirom ove frekvencije za TSU, i dobijanjem ovih vremena, potrebno je izvrsiti jos jednu modifikaciju TSU. TSU ima mogucnost i alternativnog inkrementa. Tj. zadavanjem vrednosti za alternativno inkrementiranje brojaca, TSU moze nakon odredjenog zadatog vremena, promeniti vrednosti kojima inkrementira brojac. U ovom slucaju, ispravljanje greske se vrsi na svakih 83 ciklusa unutar TSU, i u tom ciklusu se dodaje 16ns na vec postojecu vrednost unutar TSU brojaca. Ovim podesavanjima modula, dobija se sto vernija predstava realnog vremena. Tj. dobija se da ono sto dobijamo od oscilatora, odgovara stvarnim vrednostima vremena. Naravno, zbog vrednosti koje dolaze sa oscilatora i podesavanja PLL-ova unutar samog procesora, mora postojati odredjena greska, koja se moze tolerisati, i koja se moze smanjiti na odredjene nacine.

Konfiguracija GMAC modula podrazumeva i registrovanje prekida za Sync i Delay_Req pakete koji stizu na interfejs, cime se dobijaju vremena koja su potrebna. Nakon ispravne konfiguracije, dobijaju se prekidi na koje stizu paketi, i iz kojih je moguce iscitati vremena. Na Sync i Delay_Req pakete, i prekide koji javljaju, moguce je iscitati trenutnu vrednost vremena koje se nalazi u TSU, sto je bitno za proces sinhronizacije. U skladu sa arhitekturom softverske implementacije, vremena koja se dobijaju, se prosledjuju u red sa porukama, koji su potrebni za celokupnu sinhronizaciju.

[lazarc] prikaz razmene poruka

Protokol tacnog vremena odredjuje i to da slave uredjaj mora prepoznati pakete koji stizu, sto se postize hardverskom podrskom za neke od paketa. Ali i odgovoriti na odredjene pakete. Ovo se postize implementiranjem masine

stanja koja vodi racuna o trenutnom stanju uredjaja u smislu sinhronizacije. S tim u vezi, masina stanja ostaje u istom, pocetnom stanju, sve dok ne stigne prvi paket koji je bitan za proces sinhronizacije, Sync frame. Nakon cega se iz prekida dobija poruka sa trenutnom vrednosti vremena u TSU, i prelazi se u sledece stanje. Sledece stanje u masini stanja predstavlja stanje cekanja na sledeci paket, tj. Follow_up frame. 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 masini stanja, nakon cega se prelazi u sledece stanje, i slanje uzetog vremena u red sa porukama. Sledece 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 tacno vreme kada je paket napustio uredjaj. To vreme se takodje salje u red sa porukama. Nakon cega se prelazi u sledece stanje, odnosno cekanje prijema Delay_Resp paketa. Nakon sto Delay_Resp paket pristigne, i raspakuje se na odgovarajuci nacin, dobija se cetvrto i poslednje vreme koje je potrebno za sinhronizaciju. I ono se salje u red sa porukama.

[lazarc] prikaz masine stanja

Nakon ovoga, sva vremena koja su potrebna za sinhronizaciju su tu, i moze se izvrsiti promena TSU, tako da racunanje vremena odgovara vremenu na master uredjaju. Racunanjem vrednosti kojim se treba promeniti TSU, koriste se vec 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 nacin implementacije, potrebne je samo jedna nit, i to ona koja ce da vodi racuna o masini stanja. Unutar ove niti, odnosno masine stanja, implementirani su blokirajuci pozivi za dobijanje vremena iz reda sa porukama, cime se smanjuje vreme i resursi koje procesor trosi na opsluzivanje ove niti. Takodje, ovakav nacin implementacije dozvoljava implementaciju ostalih niti koje mogu izvrsavati ostale funkcionalnosti sistema, s obzirom da ova nit nije zahtevna i javlja se relativno retko potreba za sinhronizacijom, 1s.

6.0.1 Rezultati

U ovom delu ce biti dati rezultati trenutnog izvrsavanja i sinhronizacije u oglednom sistemu.

[lazarc] dodati da je samo jedna nit za sync, i jedna dummy nit

U neopterecenoj mrezi, tj. u mrezi 1 na 1, PC i razvojna ploca, dobijaju se rezultati sinhrnoizacije od otprilike 0.06ms, sto je vise nego zadovoljava-

juce za ovakav tip sinhronizacije. Takodje, postoji akumulacija greske koja je uzeta u obzir, i ona se ogleda u tome da u nekim trenucima, uredjaj koji se sinhronizuje izgubi sinhronizaciju, nakon cega se vraca u roku od jednog ciklusa sinhronizacije. Ova greska se pojavljuje usled prekoracenja (## [lazarc] overflow) dela brojaca za vremena ispod nanosekunde.

[lazarc] ubaciti slike sa putty-a

6.0.2 Predlozi za poboljsanja

Glava 7

Zaključak

Glava 8

CHECK_PTP

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 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.

SYNCHRONIZATION IEEE 1588 obezbedjuje sinhronizaciju dva sata u istoj mreži koja je otporna na greške. Takodje, koristi se veoma mali opseg frekvencija, procesna moć, ali i podešavanje samog protokola. (##[lazarc] There is very little bandwidth consumption, processing power, and setup.##) IEEE 1588 standard postiže sve ovo koristeći protokol preciznog vremena (## [lazarc] Precision time protocol.##), ili PTP. Vremenski protokol koji se koristi sinhronizuje dva sata na mreži podešavajući satove ka "najkvalitetnijemšatu. IEEE 1588 definiše ospege vrednosti za standardan set karakteristika satova. Algoritam najboljeg sata, BMC algorithm, odlučuje koji sat na mreži je "najkvalitetniji", odnosno koji sat na mreži je najtačniji, i najbolji kako bi se ostali uredjaji sinhronizovali na njega. BMC algoritam onda sinhronizuje sve ostale satove (slave clocks) sa ovim satom na mreži. Ukoliko se BMC (Best Master Clock) ukloni iz mreže, ili BMC algoritam odluči da taj sat nije više najtačniji, algoritam redefiniše novi "najkvalitet-

nijišat, i prilagodi vremena ostalih satova u skladu sa tim. Nije potrebno da se administrator mreže uključuje u bilo kom trenutku kako bi se promenio najbolji sat u mreži, zbog toga što je ovaj algoritam otporan na greške.

U ovu svrhu koristi se Bidirekciona multikast komunikacija (##[lazarc] Bidirectional Multicast Communication ##) od strane slave uredjaja kako bi se sinhronizovali na najbolji sat, IEEE 1588 grandmaster clock. Šync", Sinhronizacioni paket sadrzi vremenski žig od najboljeg sata, koji predstavlja tačno vreme kada je paket poslat sa grandmaster clock-a. "Follow up", Prateći paket takodje može biti poslat sa grandmaster sata, koji sadrži vremenski žig Šync"(sinhronizacionog) paketa. Ovakav princip razmene podataka omogućava tačan vremenski žig sinhronizacionog paketa koji je prosledjen sa grandmaster sata. Takodje, postoje slučajevi u kojima se ne otkriva tačno vreme slanja paketa, zbog odredjenih smetnji na mreži. Ovo je omogućeno kroz Collision detection i Random Back-off mehanizam u EthernetIP komunikaciji. Samo onda kad je paket kompletno poslat, nemoguće je promeniti sadržaj paketa.

Grandmaster i Slave satovi na mreži razmenjuju Šync", Sihnronizacione pakete sa jedne na drugu strane, i postavljaju vremenske žigove pri prijemu paketa. Kombinijuci razliku u satovima, kao i kašnjenje na mreži, razlika izmedju slanja i prijema sinhronizacionih paketa može biti izračunata. Koristeći razliku koja je izračunata u ovom slučaju, sat može biti podešen sa novim vrednostima, i time se može smanjiti razlika izmedju Master i Slave satova u ovoj mreži. Razlika izmedju master i slave sinhrozniacionih paketa, i obrnuto, implicira da IEEE 1588 standard radi pod pretpostavkom da je propagacija paketa po mreži simetrična. To je zbog prepostavke da slave uredjaj može da odredi i podesi kašnjenje prilikom propagacije paketa po mreži. Kako bi se kašnjenje na mreži odredilo, slave kreira đelay request", zahtev za odredjivanjem kašnjenja, i postavlja vremenski žig prilikom slanja paketa. Master sat onda postavlja vremenski žig prilikom pristizanja tog zahteva, i vraća ga ka Slave uredjaju, i to u obliku đelay response"paketa, odgovora za kašnjenjem. Nakon toga, odredjuje se kašnjenje po linijama na mreži, i izračunava se iz ovih paketa koji se razmenjuju.

Slanje i primanje sinhronizacionih paketa dozvoljava da Slave uredjaji tačno izmere razliku izmedju lokalnogSlave sata, i Master sata. Standardne metode podešavanja sata na uredjajima nije opisano prema IEEE 1588 standardu; samo omogućava standardni protokol za razmenu poruka izmedju uredjaja. Poenta ovoga je da se uredjaji i satovi različitih proizvodjača mogu sinhronizovati izmedju sebe.

QUALITY OF SYNCHRONIZATION 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 Šync", mogu uzrokovati da se izgubi sinhronizacija sa ostalim uredjajima u istom sistemu. Kako bi se predupredila svaka mogućnost za gubljenje sinhronizacije, (##[lazarc] High stability ##) 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 (TXCOs) i Oven Controlled Crystal Oscillators (OCXOs). Rezulucija sata može uticati na preciznost vremenskih žigova unutar sinhronizacionih paketa. Džiter (##[lazarc] Jitter ##) iz susednih uredjaja u mreži, kao sto su habovi i svičevi (##[lazarc] 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 podesi sistem sa što boljom sinhronizacijom, mora se napraviti kompromis izmedju egzaktnosti sinhronizacije, cene, kao i razdaljine 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.

NETWORK HIERARCHIES IEEE 1588 sporedni satovi (##[lazarc] boundary clocks##), koji se takodje i nazivaju transparentni svicevi (##[lazarc] transparent switches ##), pružaju efektivan način za smanjenje džitera (##[lazarc] jitter ##) unutar mrežnog sistema baziranog na IEEE 1588 standardu. Svič (##[lazarc] switch ##), koji se koristi kao sporedni sat (##[lazarc] boundary clock ##), pokreće PTP protokol i sinhronizuje se na master sat (##[lazarc] master clock ##). Sporedni sat (##[lazarc] boundary clock ##), u navratima (## [lazarc] in turns ##) preuzima ulogu master sata za sve slejvove unutar iste mreže. Koristeći ovo podešavanje mreže, sva interna kašnjenja i džiter mogu biti kompenzovani i ne utiču na egzaktnost sinhronizacije.

Delay_Resp, Delay_Req, Follow_up i Sync poruke se ne prenose kroz sporedne satove. Sporedni sat se ponasa kao običan sat u smislu sinhronizacije i koristi algoritam najboljeg sata unutar podmreže. Unutar podmreže koja se posmatra, ovaj uredjaj je slejv. Ovo će uticati na to da se svi ostali uredjaji koji se povezuju na sporedni sat sinhronišu 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 korišćenje transparentnih svičeva (##[lazarc] switches ##). Transparentni svič se ne ponaša kao PTP čvor unutar

IEEE 1588 sistema. Umesto toga, transparetni svič podešava vremenski deo PTP paketa kako bi se kompenzovalo kašnjenje koje unosi svič. Transparetni svič nakon toga preračunava koliko je vremena sinhronizacioni 'Sync' paket proveo unutar sviča, i modifikuje vremenski žig unutar sledećeg 'Follow_up' paketa kako bi se nadoknadilo kašnjenje. PTP čvorovi mogu raditi kao da su deo većeg podsistema lokalne mreže i to kao da su povezani habovima (##[lazarc] Hubs ##) koristeći transparentne svičeve.

USES FOR IEEE 1588 Precizna sinhronizacija se može iskoristiti u sledećim aplikacijama:

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

IZ PTP_NEXT_2.TXT

NETWORK TIMING TECHNOLOGY: NTP vs. PTP 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. (##[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 naučne institucije, vremenska sinhronizacija mora biti tačna na milijarditi, ili nekad na trilioniti deo sekunde, medjutim sve više komercijalnih i industrijskih organizacija se sve više zalažu za ideju da preciznost vremenske sinhronizacije bude u sub-milisecond opsegu. Zašto nije dovoljno da samo sinhronizujemo svoje uredjaje preko javno dostupnog NTP protokola. Nažalost, kašnjenje postoji svugde, i prosto je nemoguće postići savršenu sinhrnoizaciju. Brzina svetlosti je brza, u vakumu, foton moze 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 (##[lazarc] switches ##), rutere, i ostalu mrežnu infrastrukturu, i ta desetina sekunde se uveća nekoliko puta. Bez specijalizovane opreme, naša mreža lako može dodati kašnjenje čak i veće od sekunde. Još veća briga je

sinhronizacija različitih uredjaja 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 odluči da proda tih 100 akcija ne samo jednom investitoru, već nekoliko njih u razmaku od sekunde. Ali pošto su serveri ove institucije nesihronizovani jedan na drugi, ne postoji uopšte ideja kako odrediti koja je ponuda stigla prva. U ovom realnom scenariju sinhronizacija uredjaja 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 (##[lazarc] strata eng. ##). 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 veći level sinhronizacije i poboljšala preciznost, i tako dalje. 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.

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 žigovima, umesto softverskih, i kao svaki fini naučni instrument, PTP oprema je specijalizovana za samo jednu specijalizovanu svrhu: očuvanje

sinhronizacije izmedju uredjaja. Samo iz ovih razloga, PTP mreže imaju mnogo tačniju vremensku rezoluciju, i ne kao NTP uredjaji, PTP uredjaji ce ustvari uključiti vreme koje svaka od sinhronizacionih poruka provede u svakom od uredjaja, što uključuje i kašnjenje u uredjaju.

Svaka od PTP razmena uključuje seriju od 4 poruke koje se razmenjuju izmedju 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 odredjivanje kašnjenja od slejva ka masteru Delay_Req message
- Finalni odgovor sa kašnjenjem od mastera ka slejvu Delay_Resp message

Ova razmena pruža četiri različita vremena:

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

Master šalje sva 4 vremenska žiga ka slejvu tokom faze odgovora na zahtev za kašnjenjem, i slejv može sa tim vremenima da izračuna kašnjenje po mreži izmedju mastera i slejva u oba smera. Imajući specijalizovani hardver koji može da uhvati vremena lokalnog sata, slejv uredjaji mogu izbeći dodatno kašnjenje koje je uslovljeno lokalnim operativnim sistemom.

NTP mreže imaju dodatno kašnjenje 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 vrmena kako bi se rešili svi konflikti u dogledno vrmene, 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.