Rješenje protokolskog stoga za među-procesorsku Ethernet komunikaciju u automobilskoj industriji

Tomljenović, Davor

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

RJEŠENJE PROTOKOLSKOG STOGA ZA MEĐU-PROCESORSKU ETHERNET KOMUNIKACIJU U AUTOMOBILSKOJ INDUSTRIJI

Diplomski rad

Davor Tomljenović

Osijek, 2017.

SADRŽAJ

1. UVOD

Razvojem automobilske tehnologije i povećanjem funkcionalnosti automobila, eksponencijalno raste i količina podataka koja se obrađuje tijekom vožnje. Računala koja obrađuju prikupljene podatke nazivaju se elektroničkim upravljačkim jedinicama ili kraće ECU (engl. *Electronic Control Unit* - ECU). U početku su to bila jednostavna računala s jednim procesorom koji je radio s taktom od nekoliko stotina hertza, a danas su to više-procesorska računala čiji procesori imaju i više od jedne jezgre. Kako bi procesori mogli zajedno raditi, mora postojati komunikacija među njima. Komunikacija se može odvijati putem dijeljene memorije ili putem nekog transportnog protokola, npr. *Ethernet*. Dijeljenjem memorije procesori direktno pristupaju istoj memoriji i tako razmjenjuju podatke, dok kod *Ethernet* komunikacije, komunikacija može biti istorazinska (engl. *Peer to peer*) ili klijent-server.

ECU na kojem je potrebno obaviti mjerenja ima dva procesora povezana *Ethernet* vezom brzine do 100 Mbit/s (100 Base-TX). Prilikom razmjene poruka između procesora, dolazi do pitanja efikasnosti prijenosa. Slanjem velikog broja paketa koji su manje veličine ne iskorištava se u potpunosti dostupnih 100 Mbit/s jer je količina korisnog podatka relativno mala s obzirom na količinu podataka koji se nalaze u zaglavlju prijenosnog paketa. Povećanjem količine korisnih podataka koji se prenose unutar jednog prijenosnog paketa povećava se efikasnost jer prenosi veća količina informacija, ali se povećava vrijeme potrebno za obradu prenesenih podataka, što uzrokuje povećanje vremenskog kašnjenja. Zbog opisane dinamike prijenosa, javlja se pitanje optimalnog protokolskog stoga, odnosno optimalne veličine paketa kojom će se moći prenijeti dovoljna količina korisnog podatka uz što manje vremensko kašnjenje. Ovaj rad se bavi analizom te dinamike. Budući da su elektroničke upravljačke jedinice sustavi stvarnoga vremena, operacijski sustavi na ploči se razlikuju od operacijskih sustava na osobnim računalima. Implementacija Ethernet komunikacije ovisi o operacijskom sustavu pa će se tako sa jednog procesora koji radi pomoću RT-Linux operacijskog sustava, poruke slati putem Linux socketa, a sa drugog procesora koji radi bez operacijskog sustava, poruke će se izmjenjivat putem „*lightweight IP"* biblioteke.

Diplomki rad je strukturiran na slijedeći način. U drugom poglavlju je objašnjena teorijska pozadina protokola pomoću kojih su obavljana mjerenja, kao i teorijska pozadina UDP protokola, čije su performanse mjerene u samom radu. Treće poglavlje opisuje način provedbe programske podrške

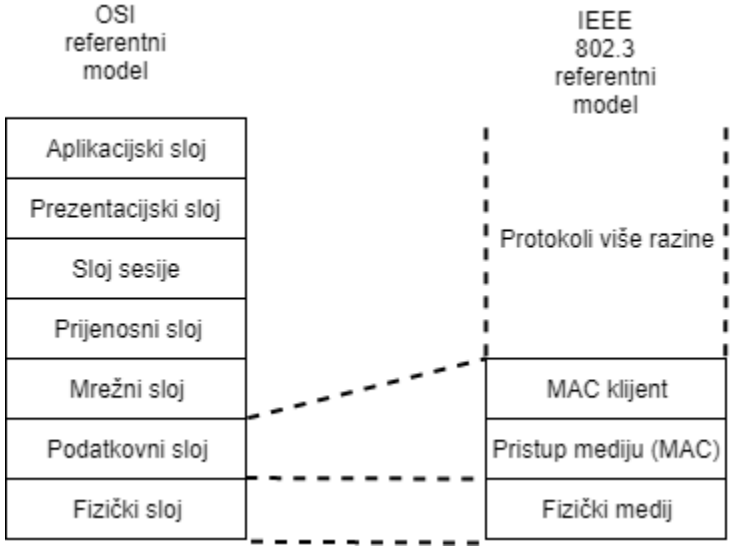
pomoću koje se vršila analiza performansi, dok se u četvrtom poglavlju nalazi prikaz i tumačenje dobivenih rezultata.

2. ETHERNET KOMUNIKACIJA

U slijedećem poglavlju bit će opisana teorijska pozadina Ethernet komunikacije. Kratko je opisana povijesti i građa Ethernet standarda i zatim su detaljnije opisani najznačajniji protokoli pojedinog sloja Ethernet modela s naglaskom na građu paketa pojedinog protokola.

2.1. Ethernet standard

Ethernet standard, poznat kao i IEEE 802.3, mrežni je standard koja definira pravila prijenosa podataka unutar računalnih mreža. *Ethernet* je razvijen 1970-ih godina u Palo Alto istraživačkom centru u svrhu povezivanja računala i pisača i omogućavao je brzine do 3 Mbit/s. Originalni *Ethernet* standard, koji je uvelike sličan današnjem, službeno je objavljen 30. rujna 1980. godine pod nazivom "*The Ethernet, A Local Area Network. Data Link Layer and Physical Layer Specifications*". Kao što naslov standarda kaže, *Ethernet* je sadržavao specifikacije za lokalne mreže računala i to samo na razini podatkovnog i fizičkog sloja (Sl. 2.1.). Za razliku od konceptualnog standarda iz 1970-ih godina, ovaj standard je podržavao brzine i do 10 Mbit/s. Druga verzija *Etherneta* objavljena je u studenom 1982. godine na temelju koje je 1983. godine objavljen današnji IEEE 802.3 standard [1].



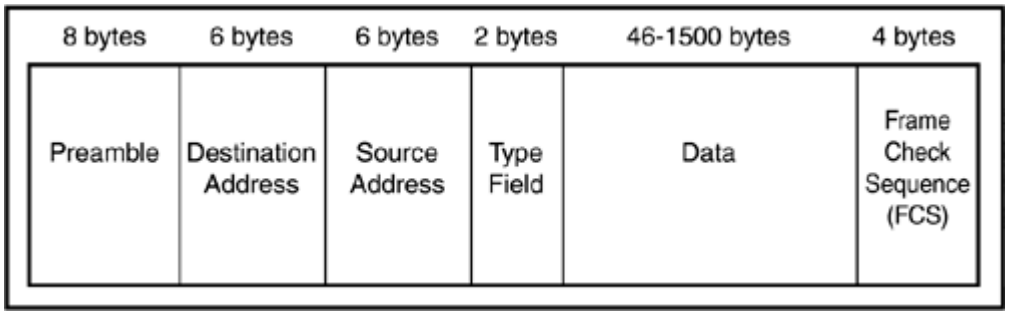
Sl. 2.1. Slojeviti prikaz Ethernet modela

Današnji *Ethernet* standard je puno složeniji i napredniji od originalnog. Podržani su veliki rasponi brzina prijenosa podataka uključujući originalnu brzinu od 1 Mbit/s pa čak i preko 1 Gbit/s. Podržano je i više smjerova komunikacije. Komunikacija više nije samo jednosmjerna u jednom trenutku, odnosno poludupleks (engl. *Half-Duplex*), nego je omogućena i komunikacija dva računala istovremeno, odnosno puni dupleks (engl. *Full-Duplex*). Za razliku od originalnog standarda koji je podržavao mreže između dva uređaja, današnji *Ethernet* podržava i više uređaja spojenih u različite topologije.

Za razumijevanje *Etherneta* potrebno je znati da *Ethernet* čine četiri osnovna elementa: podatkovni okvir, protokol za pristup mediju, komponente za signalizaciju i fizički medij. Ovi elementi čine sklopovsku i programsku podršku za prijenos podataka [1].

2.1.1. Ethernet podatkovni paket

Ethernet podatkovni okvir (engl. *Ethernet frame*) je osnovni element Ethernet standarda. Svaka poruka koja se prenosi putem računalne mreže dio je *Ethernet* podatkovnog okvira jer on sadrži sve potrebne informacije da se poruka prenese od izvora do odredišta, a da pritom njen sadržaj ostane nepromijenjen. Veličina podatkovnog okvira je promjenjiva i ovisi o količini podataka koji se prenose unutar njega. Maksimalna veličina podatkovnog okvira je 1526 bajtova ili okteta (engl. *Byte*), a minimalna veličina je 70 okteta. Informacije sadržane unutar okvira dijele se na 6 dijelova (Sl. 2.2.) [1].



Sl. 2.2. Sadržaj Ethernet podatkovnog paketa [6].

Prvi dio je preambula (engl. *Preamble*) i zauzima prvih osam okteta. Preambula se koristi u 10 Mbit/s *Ethernet* sustavima kao signalizacija prijenosa podatkovnog okvira. Prvih sedam okteta su niz naizmjeničnih bitova (npr. 101010101) koji služe za sinkronizaciju signala takta prijemnika i pošiljatelja. Primanjem preambule, primatelju je signaliziran početak prijenosa podataka i sukladno svakom bitu preambule, signal takta primatelja poprima svoju vrijednost (nisku naponsku razinu za logičku nulu i visoku naponsku razinu za logičku jedinicu). Razlog za uporabu naizmjeničnih bitova je problem prepoznavanja promjene bita. Prijemnik više uzastopnih bitova može tumačiti kao jedan te isti bit, stoga svaki bit mora biti drugačiji od njegovog prethodnika. Zadnji oktet preambule se koristi kao zastavica koja označava početak samog podatkovnog okvira. Preambula se ne koristi u sustavima bržim od 10 Mbit/s jer se sinkronizacija mora vršiti konstantno, a ne samo u slučaju prijenosa [2].

Naredna dva dijela podatkovnog okvira su odredišna adresa (engl. *Destination Address*) i izvorišna adresa (engl. *Source Address*), svaka duljine šest okteta. Adrese zapisane unutar ovih dijelova podatkovnog okvira su jedinstvene za svaki uređaj na mreži i često se nazivaju fizičke ili MAC adrese. Kako bi adresa svakog uređaja doista bila jedinstvena uveden je 24-bitni organizacijski jedinstveni identifikator (engl. *Organizationally Unique Identifier*). Ovaj identifikator dodjeljuje IEEE-SA svakoj organizaciji koja proizvodi mrežna sučelja. Organizacija na temelju dobivenog jedinstvenog identifikatora kreira jedinstvenu fizičku adresu tako što na kraj univerzalnog identifikatora dodaje još 24 bita imajući na umu da ta 24 bita ne definiraju adresu nekog od postojećih mrežnih sučelja koja ta organizacija proizvodi [1].

Sljedeći dio podatkovnog okvira (*Type field*) definira tip protokola više razine koji služi za prijenos podataka ili duljinu podatkovnog dijela. Duljina ovog dijela je dva okteta. Vrijednost zapisana unutar ovih dva okteta definirat će tip protokola više razine koji se koristi za prijenos ili duljinu podatkovnog dijela. Ako je vrijednost upisana u ovom dijelu podatkovnog paketa manja ili jednaka 1500 okteta, definirana je duljina podatkovnog dijela. Ako je upisana vrijednost veća od 1500 okteta, radi se o informaciji o protokolu više razine. Moguće vrijednosti ovog dijela podatkovnog okvira su [3]:

- 0 - 1500 Duljina podatka (IEEE 802.3 i/ili 802.2),
- 0x0800 Internet protokol verzije 4 (IPv4),
- 0x0806 *Address Resolution Protocol* (ARP),
- 0x8137 *Internet Packet eXchange* (IPX), i
- 0x86DD Internet protokol verzije 6 (IPv6).

Narednih 46 do 1500 okteta su podatak koji se prenosi unutar podatkovnog okvira. Minimalna duljina od 46 okteta je potrebna kako bi signali koji nose informacije sadržane u paketu bili prisutni na mreži dovoljno dugo kako bi svaki uređaj spojen na mrežu pročitao poruku unutar zadanih vremenskih granica. Ako je veličina korisnog podatka manja od 46 okteta, onda se dodaju se bitovi čija je svrha isključivo popunjavanje duljine poruke kako bi bila 46 okteta [1].

Zadnjih četiri okteta su zaštitni dio podatkovnog okvira (*Frame Check Sequence*). Cijeli podatkovni okvir je zaštićen provjerom cikličke redundancije (CRC). CRC je jedinstveni broj koji se generira dijeljenjem sadržaja koji se štiti sa proizvoljnim polinomom. Operacija se vrši na razini bita. Prijemna strana također računa CRC broj i uspoređuje dobiveni CRC sa CRC-om zapisanim unutar podatkovnog okvira. Ako su vrijednosti jednake, poruka je primljena bez pogreške u sadržaju [1].

2.1.2. Protokol za pristup mediju

Protokol za pristup mediju ili kraće MAC protokol (engl. *Media Access Control Protocol*) sadrži skup pravila za pristup zajedničkom komunikacijskom kanalu unutar računalne mreže. MAC protokol omogućava komunikaciju unutar mreže bez posredstva središnjeg kontrolera, odnosno komunikaciju u kojoj je svaki uređaj neovisan o ostalima. Slanje podatkovnih okvira putem Etherneta se obavlja globalnim odašiljanjem (engl. *Broadcast*). Na ovaj način je pojednostavljena izvedba prijenosnog medija jer se podatkovni okvir dostavlja svim uređajima na mreži koji samostalno provjeravaju je li poruka namijenjena njima ili ne. Kada podatkovni paket stigne na uređaj unutar mreže, provjerava se odredišna adresa zapisana unutar podatkovnog okvira. Ako

zapisana adresa odgovara adresi uređaja koji čita podatkovni okvir, uređaj zna da je poruka namijenjena njemu [1].

Kada se radi o poludupleks komunikaciji unutar mreže, potrebno je uvesti dodatan protokol koji će definirati kada koji uređaj smije slati poruku. Protokol kojim je definirana poludupleks komunikacija naziva se CSMA/CD (engl. *Carrier Sense, Multiple Access, Collision Detection)* protokol. Protokol definira faze kroz koje uređaji na mreži prolazi kako bi se omogućila uspješna komunikacija. Kako bi uređaj mogao poslati poruku, prvo mora čekati da prijenosni medij bude slobodan (engl. *Carrier Sense*). Kada je prijenosni medij slobodan, odnosno nema signala na kanalu, svi uređaji imaju jednako pravo pristupa mediju i mogućnost slanja poruke (engl. *Multiple Access*). Ako se dogodi da više uređaja istovremeno počne slati poruke u isto vrijeme, detektira se sukob signala (engl. *Collision Detection*) i svi uređaji prestaju sa slanjem. Vrijeme ponovnog pokušaja slanja se odabire nasumično, tako će uređaj sa najmanjim dobivenim vremenom ponovnog slanja prvi poslati poruku. Kod puni dupleks komunikacije, uređaji ne čekaju da se kanal oslobodi nego šalju poruke čim imaju što za poslati. Također se i ignorira sukob signala jer je normalno da se nalazi više signala na kanalu kada dva ili više uređaja komuniciraju istovremeno [1].

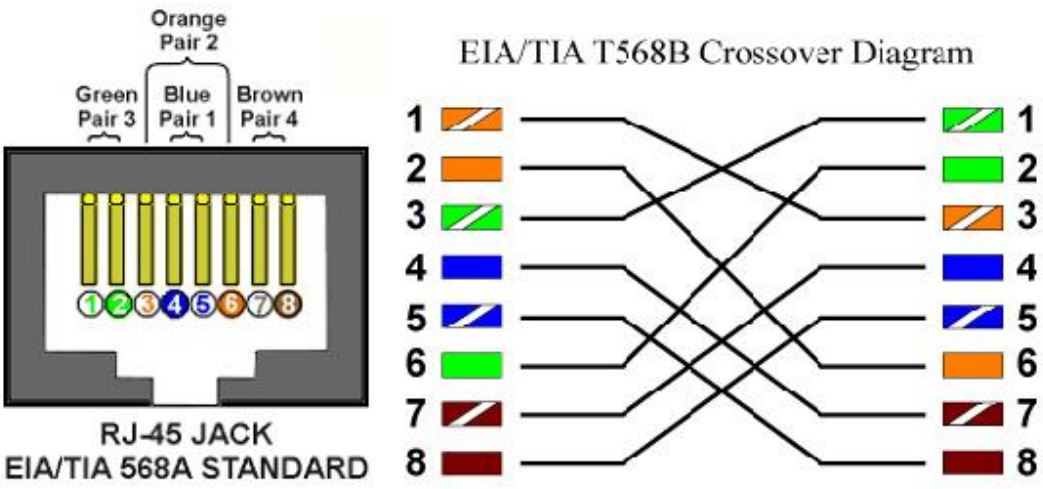
2.1.3. Komponente za signalizaciju

Komponente za signalizaciju unutar sustava koji kao prijenosni medij koristi upletenu paricu su: *Ethernet* sučelje koje se nalazi unutar računala, primopredajnik (engl. *Transceiver)* i kabel. Dva uređaja mogu biti spojena izravno jednom upletenom paricom, a ako se na mrežu spoji više uređaja, potrebno je spojiti i *Ethernet* ponavljač (engl. *Repeater*). Ponavljač ponavlja signal koji primi i šalje na višestruke segmente mreže. Za uspješnu komunikaciju uređaja na računalnoj mreži, svaki uređaj mora imati *Ethernet* sučelje. Sučelje je elektronička kartica koja omogućava generiranje i slanje podatkovnih okvira, ali i primanje i obradu primljenih podatkovnih okvira kako bi se iz njih dobila poruka. Postoje dvije vrste *Ethernet* sučelja. Sučelje može biti dio matične ploče računala ili može biti izvedeno kao vanjsko sklopovlje koje se spaja na računalo. Obje vrste sučelja se spajaju na prijenosni medij preko primopredajnika. Primopredajnik je uređaj koji prima ili šalje električne signale koji čine podatkovni okvir. Kao i sučelje, primopredajnik može biti izveden kao dio sučelja ili kao vanjski uređaj [1].

2.1.4. Fizički medij

Fizički mediji su sve komponente računalne mreže koje služe isključivo za prijenos električnih signala od jednog uređaja do drugog. Vodič koji se koristi za prijenos ovisi o vrsti komponenti za signalizaciju. Ako sustav za signalizaciju odašilje električne signale, fizički medij mora biti upletena parica, a ako sustav za signalizaciju odašilje svjetlosne signale, koristi se optičko vlakno kao medij za prijenos. U posebnim slučajevima kada su uređaji na istoj ploči, fizički medij može biti vod na tiskanoj pločici. Ako se za spajanje uređaja koristi upletena parica, mora se odabrati odgovarajuća izvedba kabela. Za spajanje uređaja na mrežni preklopnik (engl. *Switch*) koristi se konvencionalni LAN kabel ili tzv. *Patch* kabel (Sl. 2.3). Za spajanje dva uređaja izravno, koristi se tzv. *Crossover* kabel (Sl. 2.4). Razlika između ove dvije izvedbe je u rasporedu parica koje su spojene na RJ-45 priključak [1].

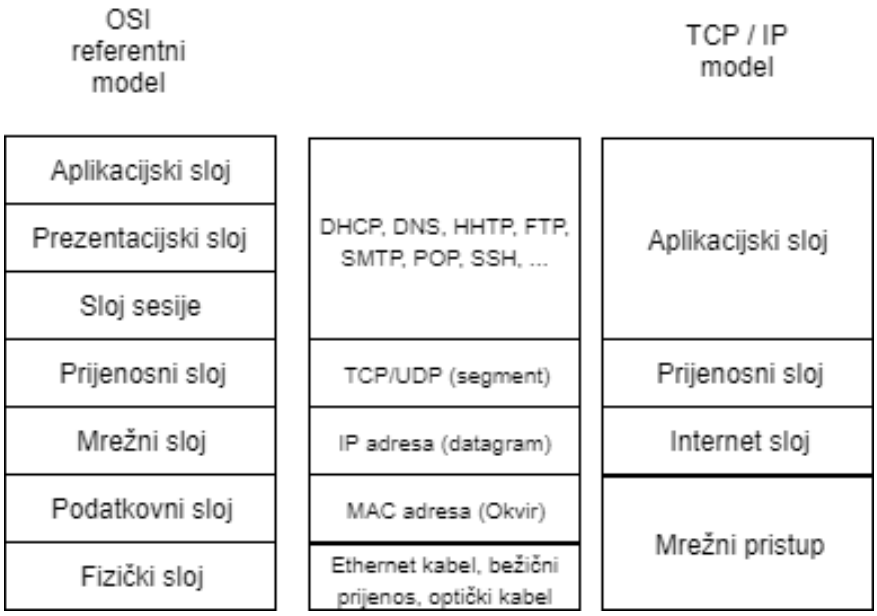
Sl. 2.3. Način spajanja parica *Patch* kabela [7].



Sl. 2.4. Način spajanja *Crossover* kabela [7].

2.2. TCP/IP protokolski stog

Internet protocol suite protokolski stog, poznatiji kao TCP/IP (engl. *Transmission Control Protocol / Internet Protocol)* je skup protokola koji omogućavaju komunikaciju između različitih računalnih mreža. Naziv je dobio prema dva protokola koji su prvi definirani unutar ovog protokolskog stoga, tj. TCP protokol i IP protokol. TCP/IP model je uređen vrlo slično OSI modelu (Sl. 2.5.) [4].



Sl. 2.5. Prikaz slojeva OSI modela i TCP/IP modela

2.2.1. Aplikacijski sloj

Aplikacijski sloj (engl. *Application layer*) predstavlja korisničku razinu. To je najčešće korisnička aplikacija koja treba komunicirati s nekom udaljenom aplikacijom. Aplikacija, odnosno pokrenuti proces je identificiran od strane TCP/IP protokola pomoću jednog ili više priključaka (engl. *Port*). Priključak je 16 bitni broj u rasponu od 0 do 65535 i jedinstven je za svaku aplikaciju. Postoje dvije vrste portova: predefinirani i privremeni [5].

Predefinirani priključci su u rasponu od 1 do 1023 i definirani su unutar standarda. Koriste se za usluge široke primjene kao na primjer: FTP (engl. *File Transfer Protocol*) na priključku 21, *Telnet* na priključku 23, HTTP (engl. *Hypertext Transfer Protocol*) na priključku 80, itd.. Priključci u rasponu od 256 do 1023 su rezervirani isključivo za UNIX usluge [5].

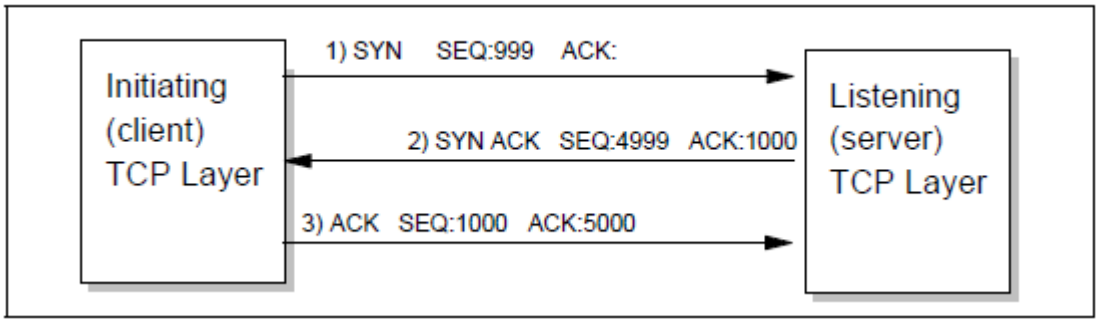
Privremeni priključci su korisnički definirani priključci koji nisu dio standarda i u rasponu su od 1024 do 65535. Aplikacija privremeno zauzima priključak i ima vlasništvo nad njim dokle god joj je potreban. Kada aplikaciji priključak više nije potreban, priključak se oslobađa. Informacije o priključku su sadržane unutar TCP ili UDP podatkovnih paketa (engl. *Datagram*) [5].

Uz vanjske priključke (*portove*), za komunikaciju se koriste i unutarnji aplikacijski priključci (engl. *Socket*). Za razliku od vanjskih priključaka koji su globalni i koji se koriste za komunikaciju između računala iz različitih mreža, unutarnji priključak je lokalan i definiran je isključivo unutar jedne aplikacije. Informacije sadržane u unutarnjem priključku su: korišteni protokol za prijenos podataka (TCP ili UDP), lokalna adresa i broj vanjskog priključka [5].

2.2.2. Prijenosni sloj

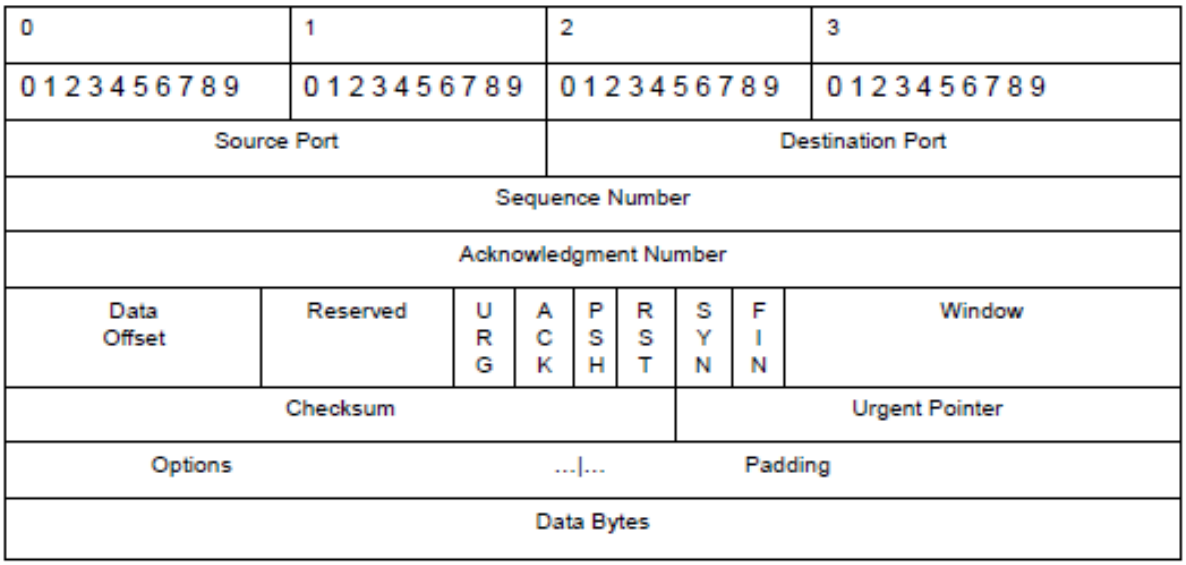
Prijenosni ili transportni sloj je po apstrakcijskom TCP/IP modelu odmah ispod aplikacijskog sloja i iznad internet sloja. Prijenosni sloj definira način prijenosa i zaštitu podataka koji se šalju i primaju između aplikacija. Prijenos se obavlja pomoću TCP protokola ili UDP protokola. Protokol koji se koristi za prijenos je definiran u unutarnjem priključku aplikacije [5].

TCP ili *Transmission Control Protocol* konekcijski je orijentiran protokol. Za prijenos podataka između dva računala, odnosno dvije aplikacije na mreži, mora postojati veza između njih. Prema modelu klijent-poslužitelj, poslužitelj ulazi u pasivno otvoreno stanje, signalizirajući TCP protokolu da je spreman prihvatiti zahtjev za vezom od nekog od klijenata. Poslužitelj ostaje u pasivnom otvorenom stanju i ne obavlja nikakvu radnju sve dok klijent ne pređe u aktivno otvoreno stanje i pošalje TCP zahtjev za vezom pomoću sinkronizacijske (SYN) poruke. SYN poruka stiže do poslužitelja koji nema informacije o klijentu koji mu je poslao zahtjev. Ako poslužitelj želi ostvariti konekciju, odgovara klijentu slanjem poruke s potvrdom (SYN ACK). Klijent prima SYN ACK poruku i šalje svoju poruku s potvrdom (ACK). Kada poslužitelj primi ACK poruku, veza je uspostavljena. Cijeli proces uspostave veze naziva se trostruko rukovanje (engl. *Three-way handshake*) i vidljiv je na slici 2.6. [5].



Sl 2.6. Princip metode trostrukog rukovanja [5].

Nakon uspostave veze omogućen je prijenos podataka. Prijenos podataka s pogleda aplikacije je kontinuiran, što znači da aplikacija ne mora brinuti o dijeljenju poruke na manje dijelove kako bi stali unutar podatkovnog paketa tj. *datagrama*. TCP grupira oktete u segmente (Sl. 2.7.) koji se spremaju u međuspremnik i zatim prosljeđuju donjem apstrakcijskom sloju koji omogućava slanje preko interneta. Kako bi se aplikaciji osiguralo da će svi podaci biti poslani, veza se ne prekida sve dok međuspremnik sa segmentima nije prazan [5].



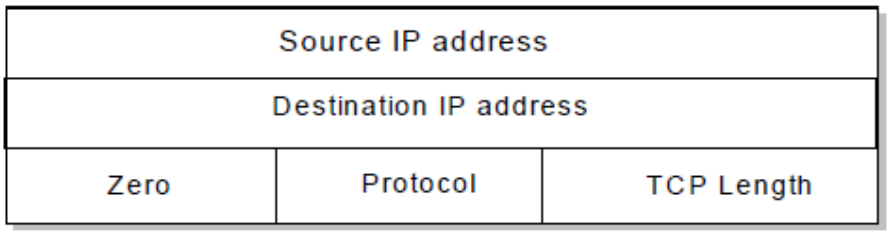
Sl. 2.7. Prikaz sadržaja TCP segmenta [5].

TCP segment se sastoji od zaglavlja i podatka. Zaglavlje sadrži minimalno deset cjelina koje sadržavaju informacije potrebne za prijenos i jednu dodatnu cjelinu. U nastavku su detaljno opisani pojedini dijelovi TCP segmenta [5].

- Prva cjelina je izvorišni vanjski priključak (engl. *Source Port*) duljine dva okteta. Informacija zapisana unutar ove cjeline sadrži broj vanjskog priključka aplikacije koja šalje poruku.
- Nakon izvorišnog vanjskog priključka, zapisan je odredišni vanjski priključak (engl. *Destination Port*), također duljine dva okteta. Odredišni vanjski priključak je broj vanjskog priključka aplikacije koja prima poruku.
- Nakon informacija o vanjskim priključcima, u naredna 4 okteta zapisan je broj sekvence (engl. *Sequence Number*). Broj sekvence služi za praćenje količine podataka koja je dosad poslana. Ako je SYN zastavica postavljena, onda se radi o inicijalnoj vrijednosti broja sekvence (n), a prvi oktet podatka je (n+1). Ako SYN zastavica nije postavljena, vrijednost broja sekvence je redni broj paketa.
- Nakon broja sekvence slijedi broj potvrde (engl. *Acknowledgement Number*), također duljine četiri okteta. Ako je ACK zastavica postavljena, vrijednost zapisana u ova četiri okteta će biti slijedeći redni broj podatka koji primatelj očekuje.

- Sljedeći broj zapisan u 32 bitnoj riječi pokazuje na početak podatka koji se prenosi (engl. *Data Offset*).
- Narednih šest bita je rezervirano za buduću uporabu.
- Nakon rezerviranog dijela, slijede zastavice:
    - URG – signalizira značajnost pokazivača na hitne podatke, odnosno je li polje pokazivača hitnosti značajano ili ne.
    - ACK – signalizira značajnost broja potvrde unutar segmenta koji se prenosi.
    - PSH – signalizira posebnu vrstu slanja segmenta. Ako se PSH zastavica ne koristi, svi podaci za prijenos se spremaju u međuspremnik sve dok se ne skupi dovoljno podataka da se može poslati segment maksimalne moguće veličine. Ako je PSH zastavica postavljena, podaci se ne skupljaju u međuspremnik, već se odmah šalju udaljenoj aplikaciji. Aplikacija pri primitku čita PSH zastavicu i ako je postavljena, ne sprema podatak u međuspremnik, već ga odmah prosljeđuje aplikaciji.
    - RST – zastavica za ponovno prekidanje i ponovno pokretanje veze.
    - SYN – služi za sinkronizaciju brojeva sekvenci.
    - FIN – označava kraj slanja podataka.

- Nakon zastavica slijedi cjelina koja definira maksimalnu veličinu okvira (najčešće u oktetima) koju primatelj definira pošiljatelju. Na ovaj način pošiljatelj zna koliko maksimalno podataka može poslati. [5]
- Narednih dva okteta su zaštitni bitovi. U svrhu zaštite kreira se pseudozaglavlje koje sadrži najbitnije informacije sadržane unutar TCP zaglavlja i IP podatkovnog paketa (Sl. 2.8.). Pseudozaglavlje se nakon kreiranja sprema u međuspremnik zajedno sa segmentom koji se treba poslati. Zatim se računa kontrolna suma nad pseudozaglavljem i TCP segmentom. Nakon izračuna sume, vrijednost se zapisuje unutar TCP segmenta, a pseudozaglavlje se odbacuje [5].

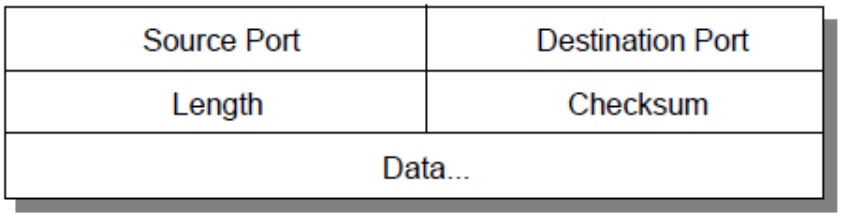


Sl. 2.8. Prikaz sadržaja pseudo zaglavlja [5].

- Nakon kontrolne sume, unutar TCP segmenta se nalazi još pokazivač na hitne podatke i dodatni segment za posebne funkcije.

Drugi prijenosni protokol TCP/IP protokolskog stoga je UDP (engl. *User Datagram Protocol*). Za razliku od TCP protokola, UDP nije konekcijski protokol, odnosno za prijenos podataka ne mora postojati veza između aplikacija. UDP je također puno jednostavniji od TCP protokola jer ne omogućava kontrolu toka, detekciju i ispravak pogreški. Glavna mu je uloga multipleksiranje i demultipleksiranje podataka za slanje i primanje. Zbog svoje jednostavne izvedbe, UDP je vrlo efikasan prijenosni protokol i omogućava veće brzine prijenosa [5].

Podaci se prenose unutar UDP podatkovnog paketa (engl. *Datagram*) koji se cijeli inkapsulira unutar IP prijenosnog paketa. UDP podatkovni paket se sastoji od zaglavlja duljine 8 okteta i podataka koji se prenose (Sl. 2.9.). Usporede li se slika 2.7. i slika 2.9., očigledno je koliko je UDP jednostavniji. Zaglavlje UDP podatkovnog paketa se sastoji od izvorišnog vanjskog priključka, odredišnog vanjskog priključka, duljine cijelog podatkovnog paketa, kontrolne sume i podatka. Polje za kontrolnu sumu je opcionalno i računa se na isti način kao i kontrolna suma za TCP. Formira se pseudo IP zaglavlje nad kojim se računa kontrolna suma. Provjera pogreške se na radi u sklopu UDP protokola, već se provjera točnosti podataka mora obaviti unutar same aplikacije [5].



Sl. 2.9. Prikaz sadržaja UDP paketa [5].

U usporedbi TCP protokola sa UDP, TCP je konekcijski orijentiran protokol i zahtjeva razmjenu tri paketa samo za uspostavu komunikacije. Pomoću sustava potvrde, detekcije pogreške i ponovnog slanja omogućava visoku pouzdanost. Podaci se prenose kao tok okteta i primaju se slijedno. Ako se slijed poruka naruši, slanje se odgađa dok se tok ne posloži. TCP se najčešće koristi u aplikacijama koje zahtijevaju pouzdan prijenos podataka, uz manje brzine prijenosa. UDP nije konekcijski orijentiran pa se podaci šalju detaljno definiranim adresama. Bez konekcijski način rada omogućuje odašiljanje poruka, što nije moguće putem TCP protokola. Prijenos nije pouzdan, protokol ne sadrži mehanizam provjere uspješnosti prijenosa, pa aplikacija ne zna ako paket ne stigne na odredište. Zaštita ispravnosti paketa je moguća, ali na aplikacijskoj razini. Podaci se prenose kao paketi i njihov prijenos nije slijedan pa prijemna strana ne može očekivati specifičan paket. UDP se koristi u aplikacijama u kojima je potrebna velika brzina prijenosa uz manju sigurnost prijenosa [5].

2.2.3. Internet sloj

Internet sloj je apstrakcijski sloj TCP/IP modela koji se nalazi između prijenosnog sloja i sloja mrežnog pristupa (engl. *Network Access*). Ovaj sloj je posrednik između prijenosnog sloja i sloja za mrežni pristup i služi za usmjeravanje podatkovnih paketa kroz više različitih mreža koje su povezane u jednu veliku mrežu koja se naziva internetwork ili kraće internet. Dakle, Internet sloj definira način prolaska podatkovnih paketa kroz internet kako bi se podatak prenio od jedne aplikacije do druge. Najbitniji protokoli ovog sloja su: IP, ICMP, ARP i DHCP [5].

IP protokol (engl. *Internet Protocol*) nekonekcijski protokol koji definira model „virtualne mreže“ koja služi kao sučelje sloju za mrežni pristup. Glavna svrha IP protokola je dostavljanje paketa primatelju. Način dostave je nepouzdan, protokol ne zna redoslijed kojim će paketi pristići primatelju, niti je li koji paket izgubljen i je li sadržaj paketa točan. Za pouzdanost prijenosa se brinu protokoli prijenosnog sloja. Glavna karakteristika IP protokola je IP adresiranje. Kako bi se poslužitelji na internetu razlikovali, potrebno ih je adresirati. Adresiranje je obavljeno pomoću 4 oktetnog pozitivnog binarnog broja koji se naziva IP adresa. IP adresa se sastoji od dva para brojeva: adrese mreže i adrese poslužitelja. Adresa mreže je dio IP adrese o kojem odlučuju regionalni internet registri (engl. *Regional Internet Registries*). Za područje Amerike zadužen je „American

Registry for Internet Numbers“, za Europu je zadužen „Reseaux IP Europeans“ i za Aziju „Asia Pacific Network Information Centre“. Adresa mreže i adresa poslužitelja se definiraju prema početnim bitovima adrese pomoću kojih se adresa dijeli u klase na sljedeći način [5].

- Ako je prvi bit 0, radi se o klasi A. Narednih 7 bita definira adresu mreže, a preostalih 24 bita definira adresu poslužitelja.
- Ako su prva dva bita 10, radi se o klasi B. Narednih 14 bita definira adresu mreže, a preostalih 16 definira adresu poslužitelja.
- Ako su prva tri bita 110, radi se o klasi C. Narednih 21 bit definira adresu mreže, a preostalih 8 bita definira adresu poslužitelja.
- Ako su prva četiri bita 1110, radi se o klasi D. Ovo je posebna adresa koja se koristi za odašiljanje poruka unutar podmreže (engl. *Multicasting*).
- Ako su prvih pet bita 11110, radi se o klasi E. Ova klasa je rezervirana za testiranje i buduću uporabu.

Uz definirane IP adrese, postoje i rezervirane IP adrese koje imaju posebnu namjenu. Ako su svi bitovi unutar adrese mreže ili adrese poslužitelja jednaki nuli, radi se o trenutnom poslužitelju (ako su svi bitovi adrese poslužitelja jednaki nuli) ili trenutnoj mreži (ako su svi bitovi unutar adrese mreže jednaki nuli). Ako aplikacija treba poslati poruku drugoj aplikaciji koja se nalazi u istoj mreži, ali ne zna koja je njena adresa, mrežna adresa se postavlja u nula i na taj način će se poruka poslati aplikaciji koja je u istoj mreži kao i aplikacija koja šalje poruku. Druga rezervirana adresa je definirana svim bitovima unutar adrese mreže ili adrese poslužitelja jednakim jedan. Ovo je adresa za selektivno odašiljanje, ako su svi bitovi unutar adrese mreže jednaki jedan, poruka će biti poslana poslužiteljima sa jednakim adresama, ali u svim mrežama, a ako su svi bitovi jednaki jedan unutar adrese poslužitelja, onda će poruka biti poslana svim poslužiteljima unutar definirane mreže. Adresa 127.0.0.0 je takozvana „*loopback*“ adresa. Koristi se za procesiranje podataka lokano, odnosno poruka nikad ne stigne na fizičku mrežu. Popis svih rezerviranih IP adresa vidljiv je u tablici 2.1. [5].

IP adresa	Namjena
0.0.0.0 / 8	„Trenutna“ mreža
14.0.0.0 /8	Javna podatkovna mreža
24.0.0.0 /8	Mreža kabelske televizije
39.0.0.0 /8	Rezervirano za buduću uporabu
128.0.0.0 /16	Rezervirano za buduću uporabu
169.254.0.0 /16	„Trenutni“ mrežni segment (Link)
191.255.0.0 /16	Rezervirano za buduću uporabu
192.0.0.0 /24	Rezervirano za buduću uporabu
192.0.2.0 /24	TEST-NET
192.18.0.0 /15	Testiranje komunikacije podmreža
223.255.255.0 /24	Rezervirano za buduću uporabu
224.0.0.0 /4	Odašiljanje unutar pod mreže (Multicast)
240.0.0.0 /4	Rezervirano za buduću uporabu

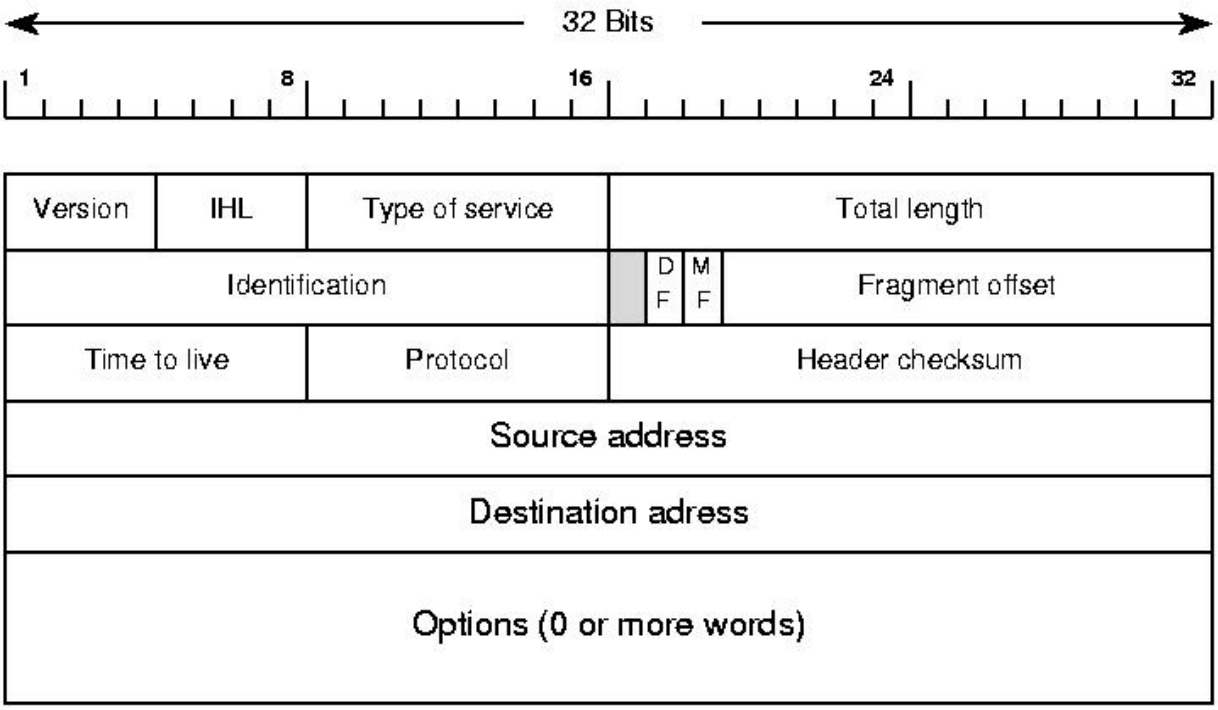
Tab. 2.1. Popis rezerviranih IP adresa

Zbog brzog širenja interneta, odnosno brzog povećanja broja uređaja povezanih na mrežu, uveden je koncept podmreža (engl. *Subnet*). Podmreža se definira lokalno tako da cijela mreža ostaje globalno vidljiva. Adresa poslužitelja se dodatno dijeli na dva dijela. Jedan dio ostaje adresa poslužitelja, a drugi dio je adresa podmreže. IP adresa se u tom slučaju sastoji od tri dijela: adresa mreže, adresa podmreže i adresa poslužitelja. Koncept podmreža je transparentan udaljenim mrežama, što znači da su lokalni uređaji svjesni o postojanju podmreže, dok udaljeni uređaji vide samo jednu pristupnu mrežu, tj. ne vide hijerarhijsko uređenje te mreže. Kako bi se unutar IP adrese odredio podmrežni dio, koriste se podmrežne maske (engl. *Subnet Mask*). Bitovi vrijednosti nula unutar podmrežne maske definiraju pozicije adrese poslužitelja, a bitovi vrijednosti 1 definiraju pozicije adrese podmreže [5].

Neka se za primjer uzme IP adresa klase B i maska podmreže 255.255.255.0. Prvih 14 bita IP adrese definira adresu mreže. Idućih osam bita definira adresu podmreže. Pomoću tih osam bita može se adresirati 254 ($2^8 - 2$) podmreža. Zadnjih osam bita definira adrese poslužitelja kojih također može biti 254.

Za prijenos podataka unutar mreža i podmreža koriste se IP podatkovni paketi (*datagrami*). Prijenos velikih količina podataka osmišljen je konceptom fragmentiranja i defragmentiranja podatkovnih paketa. Ako duljina podatkovnog dijela jednog paketa prelazi maksimalnu duljinu podatka koji se

može prenijeti unutar jednog paketa, podatak se dijeli u više odvojenih podatkovnih paketa. Taj postupak se naziva fragmentacija i odvija se prije slanja podatka. Kako podatak ne dolazi na prijemnu stranu unutar jednog paketa, nego unutar više paketa koji ne moraju pristizati slijedno, pri prijemu se odvija proces defragmentacije, odnosno ponovnog formiranja podatka u jednu cjelinu. Maksimalna duljina IP podatkovnog paketa je 65535 okteta, a svaki poslužitelj mora podržavati najmanje 576 okteta bez fragmentiranja. Zaglavlje IP podatkovnog paketa duljine je 20 okteta i njegova građa je vidljiva na slici 2.10. Opis svakog polja IP podatkovnog paketa dan je u nastavku [5].



Sl. 2.10. Prikaz sadržaja zaglavlja IP podatkovnog paketa [8].

Version – polje duljine četiri bita koje sadrži verziju IP protokola. Verzija može biti IPv4 ili IPv6.

IHL – sadrži duljinu zaglavlja IP podatkovnog paketa. U duljinu nije uključen podatak koji se prenosi.

Type of service – polje duljine osam bita koje sadrži oznaku kvalitete usluge (engl. Quality of service).

Total Length – sadrži ukupnu duljinu podatkovnog paketa, uključujući zaglavlje i podatak.

Identification – polje duljine 16 bita koje sadrži jedinstveni broj kojim se omogućava defragmentiranje paketa.

Zastavice – Zastavice su DF (engl. *Do Not Fragment*) i MF (engl. *More Fragments*). Ako je DF postavljena u nulu, fragmentacija je omogućena, a ako je 1, fragmentacija nije dozvoljena. Ako je MF postavljena u nulu, znači da je trenutni podatkovni paket zadnji fragment, a ako je jedinica, znači da postoji još fragmenata paketa.

Fragment Offset – Sadrži redni broj fragmenta i na taj način omogućava defragmentaciju paketa. Prvi fragment ima indeks nula.

Time to live – Polje duljine osam bita koje sadrži vrijeme u sekundama. Vrijeme zapisano u ovo polje definira koliko dugo paket smije putovati kroz mrežu.

Protocol – Sadrži brojčani indeks protokola više razine koji će se koristiti za prijenos podataka.

Header checksum – Kontrolna suma zaglavlja IP podatkovnog paketa.

Source address – IP adresa pošiljatelja podatkovnog paketa.

Destination address – IP adresa primatelja podatkovnog paketa.

Options – Dodatno polje duljine 32 bita koje se ne mora implementirati sa strane pošiljatelja, ali ga primatelj mora biti u stanju obraditi.

Uz IP protokol potrebno je objasniti i ARP protokol. ARP (engl. *Address Resolution Protocol)* povezuje IP adrese definirane IP protokolom s fizičkim adresama definiranim MAC protokolom. Unutar jedne fizičke mreže, svaki uređaj je identificiran svojom fizičkom, odnosno MAC adresom. Budući da svi protokoli više razine koriste IP adrese za identifikaciju, ARP protokol pomoću pregledne tablice (engl. *Lookup table*) traži za određenu IP adresu pripadajuću fizičku adresu. Ako se IP adresa ne nalazi u tablici, ARP protokol odašilje zahtjev za identifikacijom. Ako uređaj na mreži prepozna svoju IP adresu, on u odgovoru šalje svoju fizičku adresu i putanju do izvora. Pristigla fizička adresa i putanja se spremaju u preglednu tablicu [5].

3. IMPLEMENTACIJA TESTIRANJA

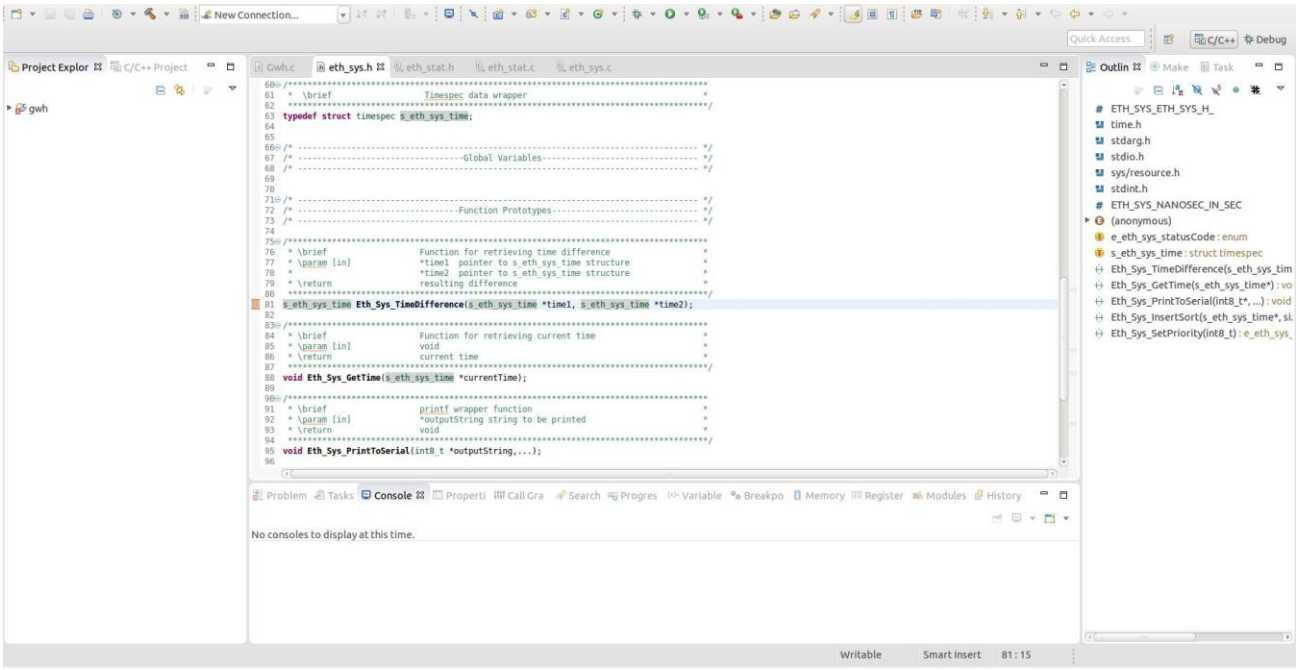
Zadatak diplomskog rada bio je odrediti optimalnu veličinu paketa za slanje putem UDP protokola sa jednog procesora na drugi. U ovom poglavlju opisano je razvojno okruženje i programsko riješenje kojim su izvršena mjerenja performansi TCP/IP protokolskog stoga za prijenos UDP paketa između dva procesora na specifičnoj platformi. Performanse prijenosa UDP paketa iskazane su mjerenjem vremenskog kašnjenja, propusnosti i propusnosti korisnih podataka. Pretpostavka je da se povećanjem veličine korisnih podataka u paketu povećava propusnost sustava, ali u isto vrijeme raste vremensko kašnjenje. Kod sustava stvarnog vremena potrebno je imati minimalno moguće vremensko kašnjenje, stoga je potrebno naći optimalnu veličinu paketa kojom se može prenijeti sav traženi promet uz što manje vremensko kašnjenje.

3.1. Razvojna okruženja

Mjerenja su se obavljala na TTTechovoj „TTA Drive“ ploči. TTA Drive je „*System On A Chip*“ sustav sa dva procesora (Sl. 3.1.). Jedan procesor je upravljan stvarnovremenskom (engl. *RealTime)* distribucijom Linux operacijskog sustava, dok drugi procesor nema operacijski sustav. TTA Drive ima dva *Ethernet* sučelja, vanjsko i unutarnje. Vanjsko sučelje je brzine 1 GB/s i koristi se za spajanje cijelog sustava na mrežu, dok unutarnje sučelje služi za komunikaciju između procesora i brzine je 100 Mbit/s.

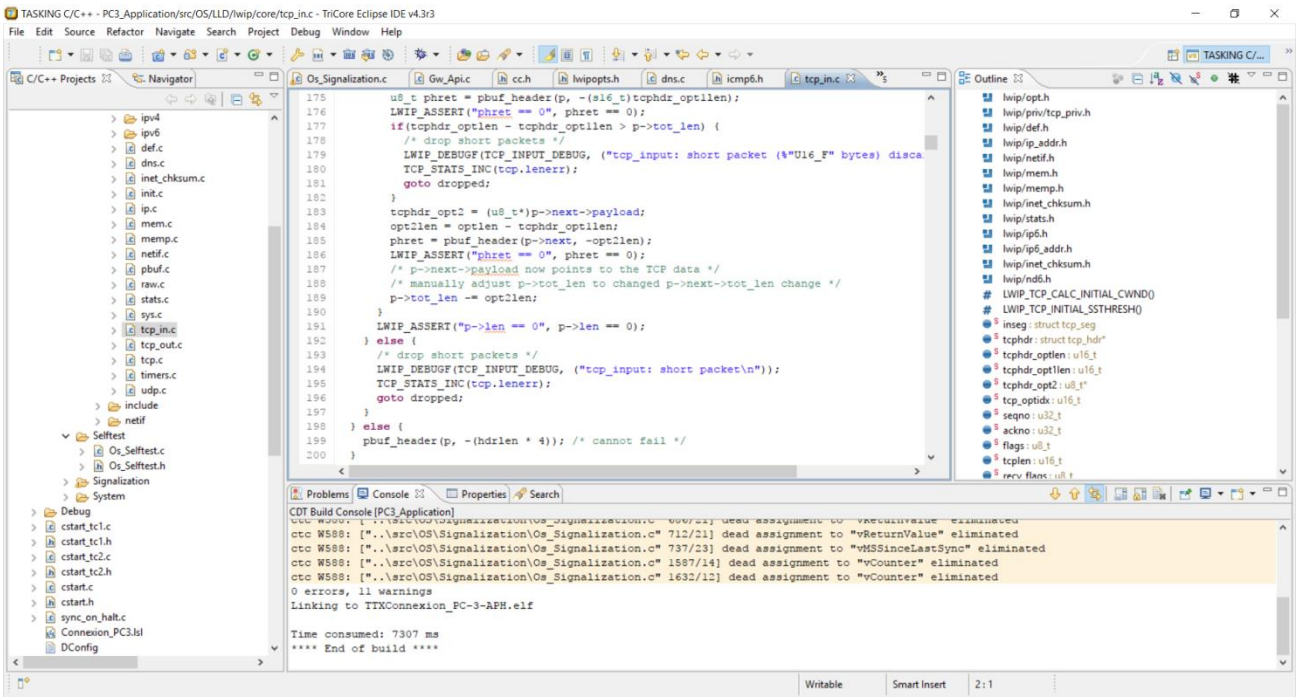
Sl. 3.1. TTA Drive

Za razvoj programske podrške korištena su dva razvojna okruženja: „Eclipse IDE“ i „Tasking“. Eclipse je integrirano razvojno okruženje (engl. *Integrated development environment* - IDE) koje omogućava razvoj programske podrške u raznim programskim jezicima poput Jave, C, C++, C#, Ade, COBOL-a i sl. (Sl. 3.2.). Glavni program za mjerenje performansi slanja poruka preko UDP protokola pisan je u Eclipse okruženju i u C programskom jeziku. TTA Drive je putem vanjskog *Etherneta* spojen na računalo, te je unutar Eclipsa postavljena IP adresa procesora pokretanog Linuxom. Nakon prevođenja (engl. *Compile*) programskog koda i kreiranja izvršne datoteke, ista se putem *Etherneta* prenosi na Linux operacijski sustav. Eclipse se također koristi i kao udaljeni sustav za analizu grešaka (engl. *Debugger*), te se putem Ethernet veze može pratiti rad programa dok se izvršava na samom TTA Driveu.

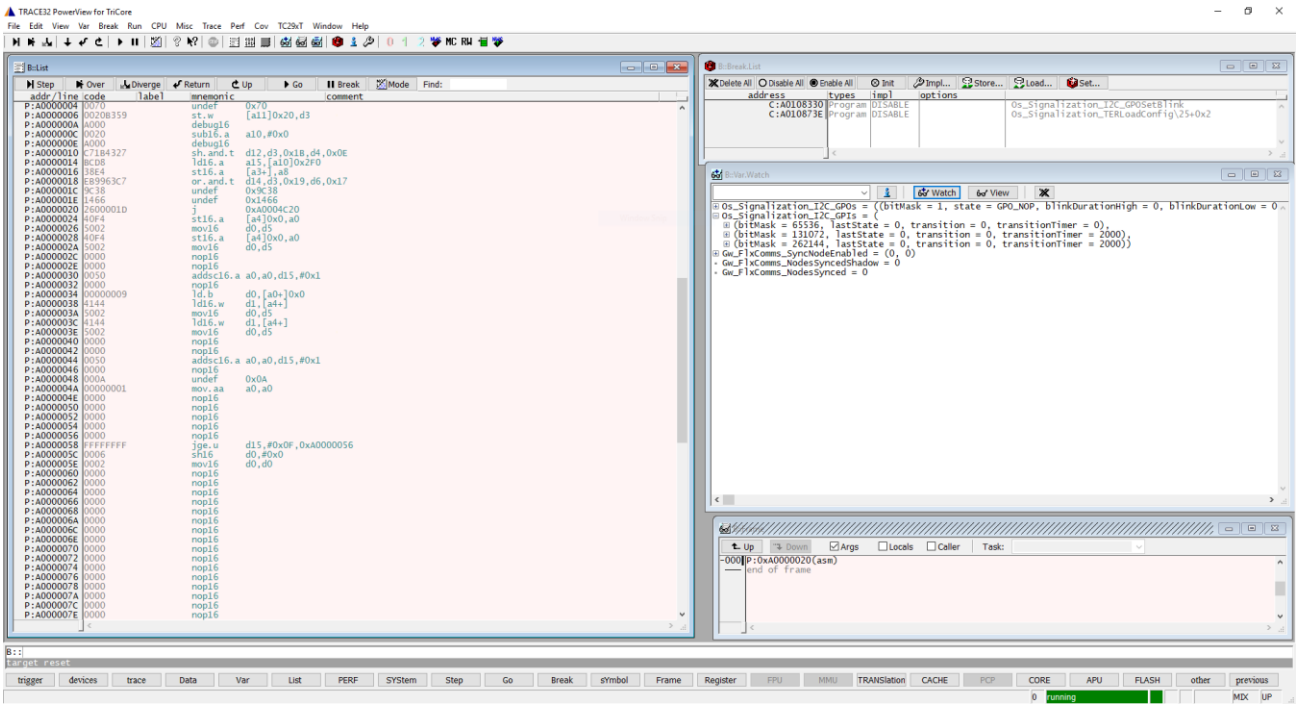


Sl. 3.2. Eclipse IDE razvojno okruženje

Tasking (Sl. 3.3.) je razvojno okruženje dizajnirano od strane Infineona koje se temelji na Eclipse razvojnom okruženju. Tasking je specijalizirano okruženje koje se koristi za razvoj programske podrške za ugradbene sustave, kao i za sustave sa sigurnosni zahtjevima (u skladu sa ISO 26262). Tasking je podešen tako da se pri prevođenju programskog koda kreira izvršna .elf datoteka i skripta koja služi za spremanje izvršne datoteke u memoriju kojoj pristupa procesor bez operacijskog sustava. Za prijenos datoteke sa računala na TTA Drive, tj. za spremanje izvršne datoteke u memoriju procesora, koristi se „Lauterbach“ sustav za analizu pogrešaka (engl. *Debugger*). Lauterbach se spaja na TTA Drive putem JTAG konektora. Nakon spajanja pokreće se Trace32 (slika 3.4.) okruženje koje se koristi za praćenje izvršavanja programskog koda i prebacivanje izvršne datoteke na ploču koja je spojena putem Lauterbacha. Lauterbach *debugger* je vidljiv na slici 3.5..



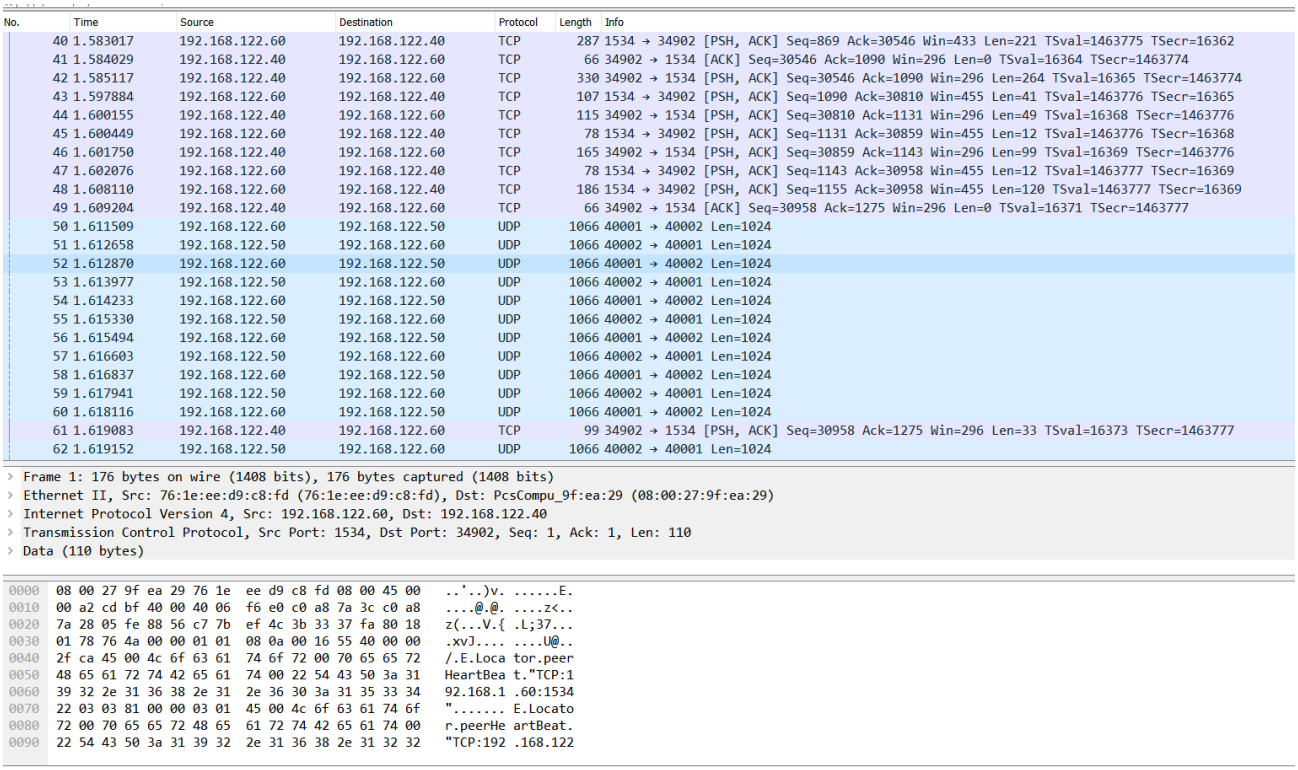
Sl. 3.3. Tasking razvojno okruženje



Sl. 3.4. Trace32 okruženje za analizu pogrešaka

Sl. 3.5. Lauterbach uređaj za analizu pogrešaka

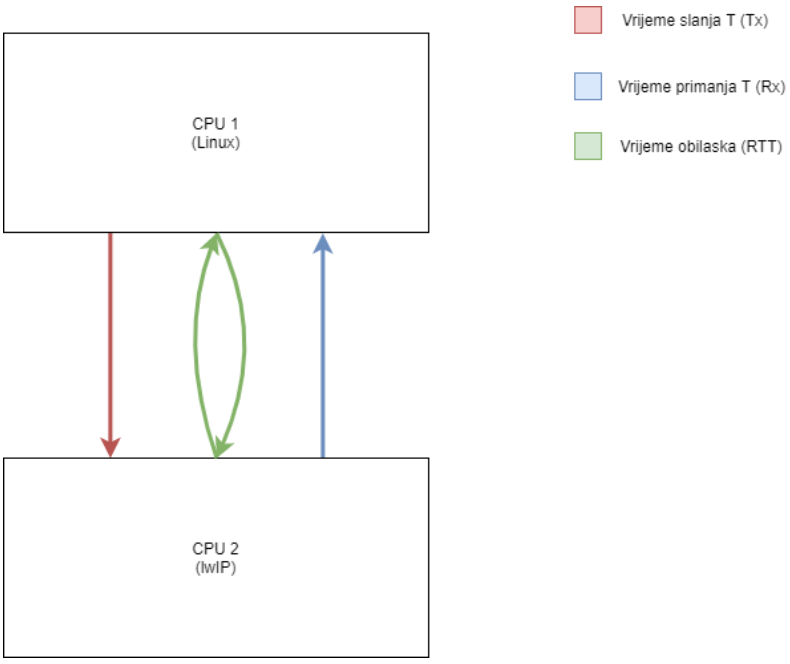
Za praćenje rada aplikacije s pogleda razmjene podataka između procesora, koristio se „Wireshark“ (Sl. 3.6.). Wireshark je analizator mrežnih protokola koja omogućava praćenje prometa na odabranom sučelju u stvarnom vremenu. Putem Wiresharka moguće je vidjeti sastav podatkovnih paketa koji se razmjenjuju raznim mrežnim protokolima, te nakon praćenja, moguće je spremiti praćeni promet u datoteku za kasniju analizu.



Sl. 3.6. Wireshark, program za analizu mrežnog prometa

3.2. Testna aplikacija

Mjerenja su definirana u tri faze: mjerenje performansi slanja, mjerenje performansi primanja i mjerenje performansi obilaska (engl. *Round Trip*). Slika 3.7. prikazuje faze mjerenja i smjer podataka koji će se razmjenjivati.



Sl. 3.7. Grafički prikaz provedbe analize protokolskog stoga

Procesor s Linux operacijskim sustavom je vođa mjerenja. Pri pokretanju programa navodi se naziv izvršne datoteke i zatim ulazni parametri programa. Prvi ulazni parametar je IP adresa procesora na koji se šalju podaci. Drugi parametar je vanjski priključak (engl. *Port*) preko kojeg će aplikacije komunicirati. Nakon dva obavezna parametra dolazi izborni parametri definirani znakom „-“ i slovom. Cijela naredba sa navedenim parametrima je:

./main <IP_Adresa> <Port> -p <Vrijednost> -n <Vrijednost> -m <Vrijednost> -s <Vrijednost>

-p (Port glavne aplikacije),

-n (Broj paketa koji se razmjenjuju),

-m (Prioritet procesa, manja vrijednost znači veći prioritet),

-s (Minimalna veličina paketa).

Ako izborni ulazni parametri nisu definirani, koriste se predefinirane vrijednosti. Nakon obrade ulaznih parametara poziva se funkcija za inicijalizaciju Ethernet komunikacije. Unutar ove funkcije se kreira unutarnji priključak (engl. *Socket*) za UDP komunikaciju pozivom na „bind“ funkciju. Kreira se struktura sa IP adresom i vanjskim priključkom udaljene aplikacije, ta struktura se koristi

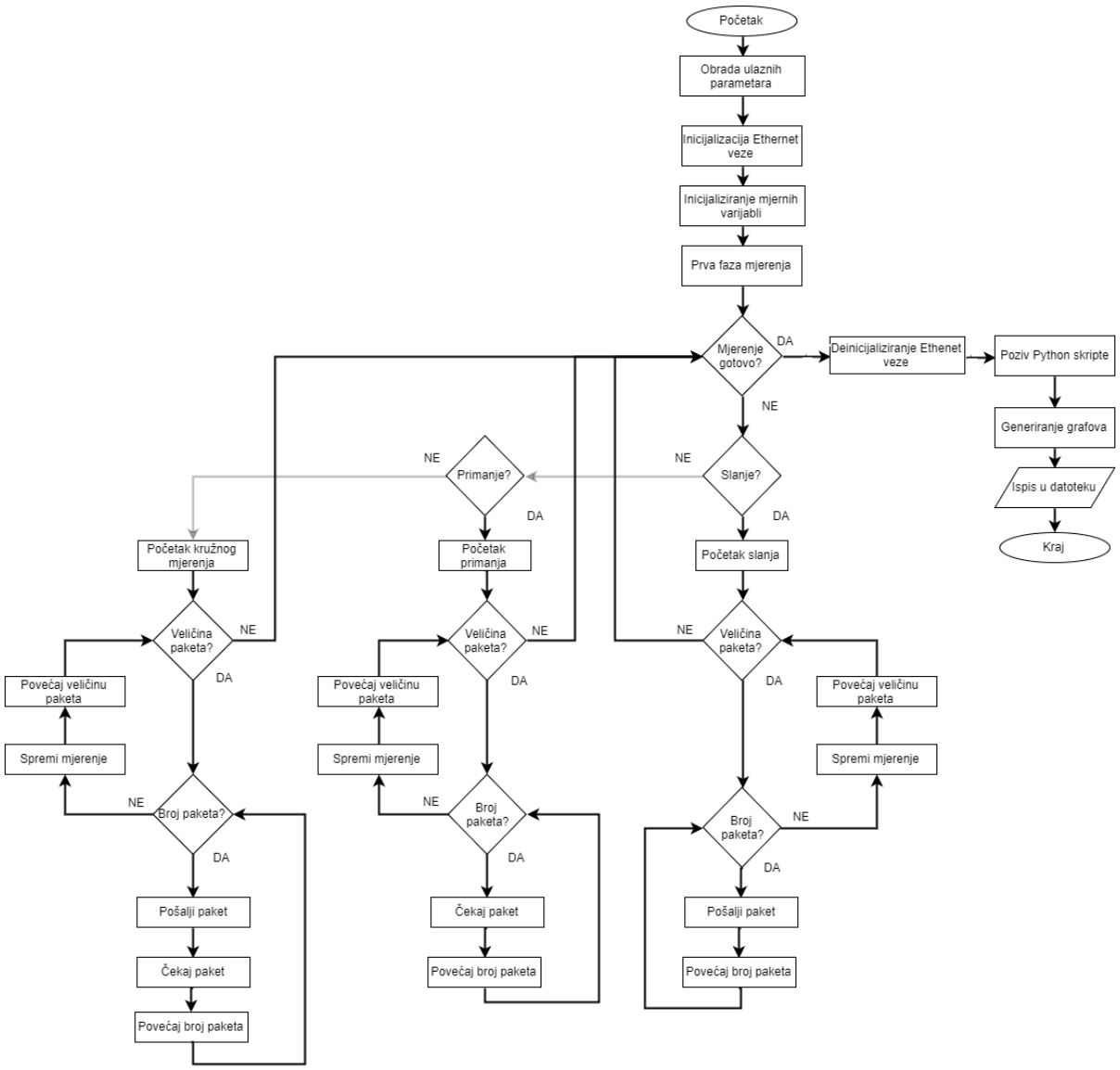
za definiranje odredišta na koje će se slati poruke. Nakon inicijaliziranja veze, inicijaliziraju se strukture i varijable koje sadrže parametre mjerenja i kreiraju se CSV (engl. *Comma Separated Values)* datoteke u koje će biti zapisane sve vrijednosti mjerenja, te im se kreiraju zaglavlja. Nakon inicijalizacije, postavlja se zastavica koja signalizira početak mjerenja.

Kada je inicijalizacija završena, aplikacija se postavlja u početno stanje mjerenja, a to je mjerenje performansi slanja. Otvara se CSV datoteka u koju će biti spremljeni podaci o slanju. Nakon toga se kreiraju dvije niti: jedna nit služi za kontrolu programa dok se izvršava mjerenje, a druga služi za izvršavanje mjerenja. Nit za kontrolu pri inicijalizaciji ne prima parametre i služi isključivo za prekid mjerenja, dok nit za izvršavanje mjerenja prima strukturu parametara. Struktura sadrži rukovatelja datotekom, trenutno stanje aplikacije, minimalnu, odnosno početnu veličinu paketa koja se šalje i broj paketa koji se šalje za svaku veličinu. Nit, nakon što je kreirana šalje podatke jedne veličine, počevši od najmanje. Kada je broj poslanih paketa jednak broju paketa definiranom unutar strukture koja je primljena kao parametar, veličina podatka se udvostručuje. Na primjer, ako je aplikacija krenula sa slanjem poruke duljine osam okteta, nakon što je zadovoljen broj poslanih paketa, slijedeća veličina paketa će biti 16 okteta. Proces slanja se ponavlja sve dok nije poslana maksimalna veličina paketa, koja je predefinirana na 1024 okteta. Vrijeme se bilježi prije svakog slanja i u niz se sprema razlika vremena zabilježenih između dva slanja. Kada je mjerenje završeno, izračunata vremena u nizu se uzlazno sortiraju i sprema se minimalno vrijeme, maksimalno vrijeme i medijan vrijeme. Medijan vrijeme se koristi za računanje broja paketa po sekundi, propusnost (engl. *Throughput*) i propusnost korisnog sadržaja (engl. *Goodput*). Formule prema kojima su izračunati navedeni parametri mjerenja su vidljive u poglavlju četiri. Kada su svi parametri mjerenja izračunati, poziva se funkcija za zapisivanje u datoteku. U datoteku se zapisuju sva izmjerena vremena i izračunati parametri. Nakon zapisa, nit završava s izvođenjem, zatvara se datoteka i započinje nova faza mjerenja.

Druga faza mjerenja je mjerenje performansi primanja. Kao i u prethodnoj fazi, kreiraju se dvije niti sa identičnim parametrima. U ovoj fazi se mjeri vrijeme prije svake primljene poruke koju *lightweight IP* strana šalje. Razlika zabilježenih vremena se sprema u niz koji se uzlazno sortira i iz kojeg se dobivaju minimalna, medijan i maksimalna vrijednost vremena primanja. Na temelju medijan vremena se računaju isti parametri mjerenja kao i u prethodnoj fazi. Izvođenje treće faze je

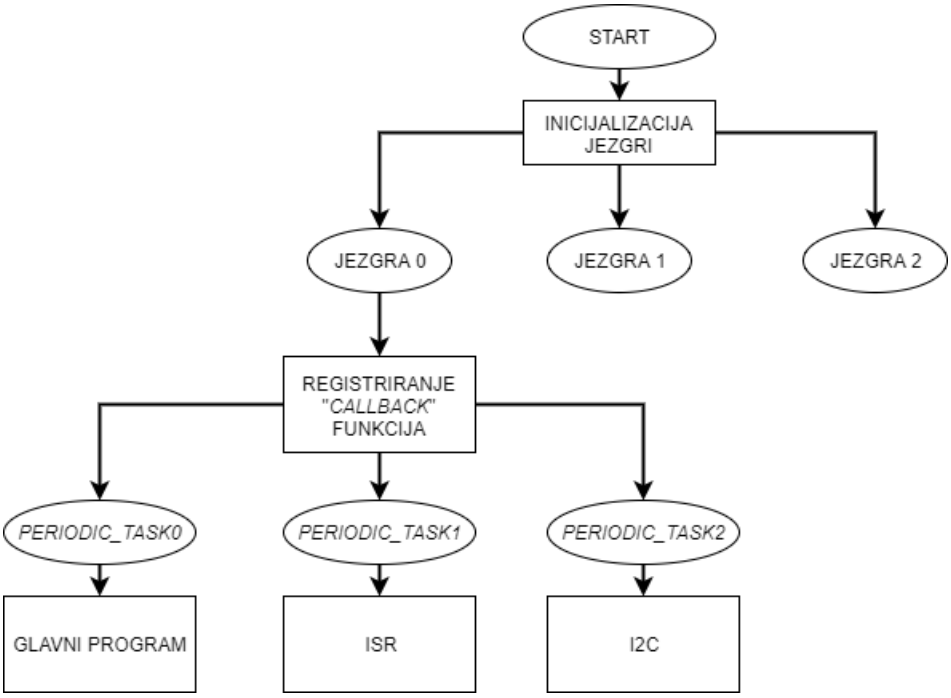
analogno izvođenju prethodne dvije, ali u ovom slučaju se bilježi vrijeme prije nego što je poruka poslana na *lightweight IP* stranu i primljena nazad od strane *lightweight IP*.

Nakon što su mjerenja izvršena i nakon što su sve izračunate vrijednosti spremljene u pripadajuće datoteke, poziva se funkcija za obradu podataka. Funkcija poziva Python interpreter i predaje mu Python skriptu koja je određena ulaznim parametrom funkcije. Python skripta otvara kreirane datoteke s izvršenim mjerenjima i spaja mjerenja u jednu zajedničku .xlsx datoteku, u kojoj je svako mjerenje unutar svog radnog lista. Nakon spajanja podataka, čitaju se zapisane vrijednosti i iscrtavaju se grafovi. Nakon što je .xlsx datoteka kreirana i nakon što je Python skripta završila s izvođenjem, CSV datoteke se uklanjaju i program završava s radom. Dijagram toka programa vidljiv je na slici 3.8..



Sl. 3.8. Blok dijagram toka programa za analizu protokolskog stoga

Drugi procesor pokreće program koji se sastoji samo od „lightweight IP“ biblioteke. Program se pokreće odmah pri uključivanju napajanja za ploču, ako sustav za analizu pogrešaka nije spojen ili pokretanjem programa iz Trace32, ako je sustav za analizu pogrešaka je spojen. Lightweight IP je biblioteka koja definira TCP/IP protokolski stog za računala slabijih performansi ili računala s vrlo malo memorije. Najčešće se koristi u ugradbenim računalima jer implementacija TCP/IP protokolskog stoga pomoću ove biblioteke zauzima samo nekoliko desetaka kilo okteta. Izvršavanje programa sa Lightweight IP bibliotekom se izvršava putem funkcija povratnog poziva (engl. *Callback function*) koje poziva jezgra biblioteke (Sl. 3.9.).



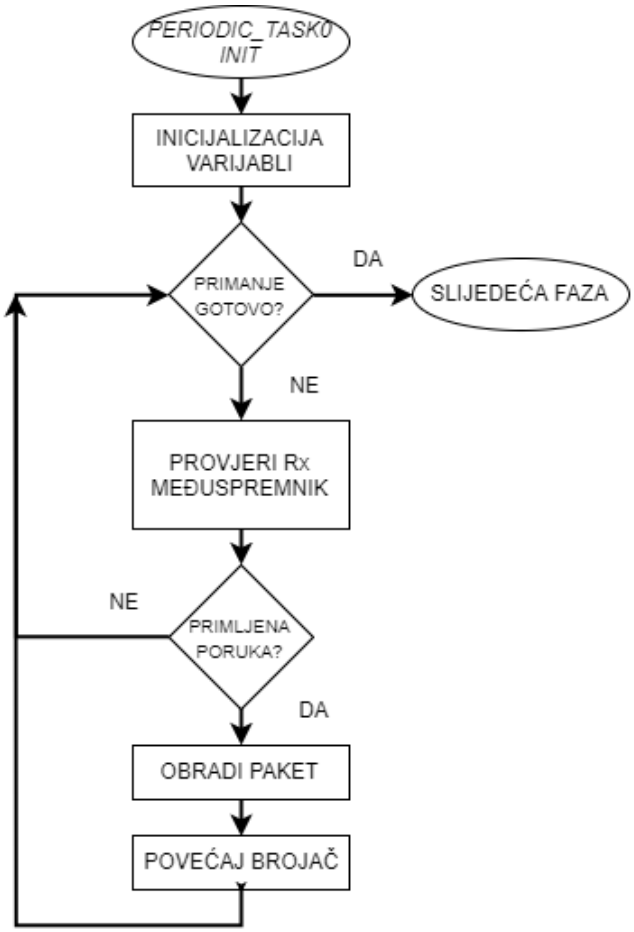
Sl. 3.9. Inicijalizacija *Lightweight IP* biblioteke

Glavna funkcija u kojoj se obavlja slanje i/ili primanje podataka je funkcija „PeriodicTask“. Ovisno o kojoj se fazi mjerenja radi, poziva se odgovarajuća *Send* ili *Receive* funkcija. Za razliku od Linux strane gdje je za slanje i primanje podataka dovoljno pozvati funkciju *SendTo* ili *ReceiveFrom*, unutar *Lightweight IP* biblioteke generiranje zaglavlja paketa nije automatsko. Pri slanju podataka, svaki puta se mora popuniti struktura koja sadrži potrebne parametre za slanje (adresa primatelja, adresa pošiljatelja, kontrolna suma, itd…). Za primanje podataka se također prvo provjerava zaglavlje i tek onda se sprema podatak.

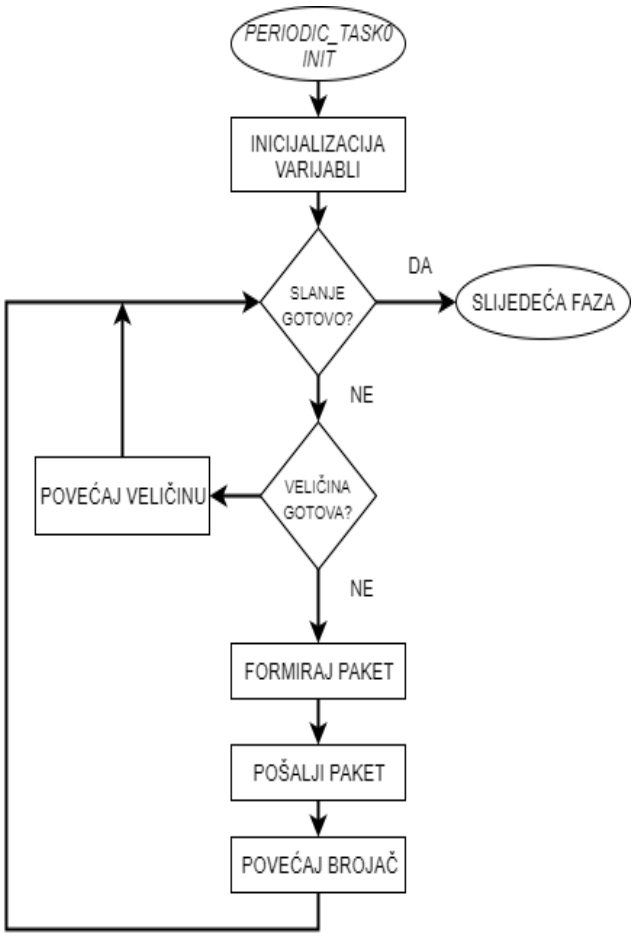
Početak mjerenja na *Lightweight IP* strani se kao i na Linux strani obavlja inicijalizacijom mjernih varijabli. Za razliku od Linux strane gdje se vrijednosti varijabli zadaju putem ulaznih parametara, na *Lightweight IP* strani se „ulazni parametri“ definiraju ručno, unutar koda, nakon čega se izvršna datoteka mora prebaciti na ploču. Raspored faza je drugačije definiran nego na Linux strani.

Prva faza je primanje poruka. *PeriodicTask* funkcija se unatoč nazivu ne poziva periodički, nego što je prije moguće. Pri svakom pozivu funkcije provjerava se ulazni međuspremnik kako bi se provjerilo je li stigla poruka. Ako međuspremnik nije prazan, obrađuje se sadržaj primljenog paketa i povećava se brojač pristiglih poruka. Kada je pristigao definirani broj paketa, prelazi se u slijedeću

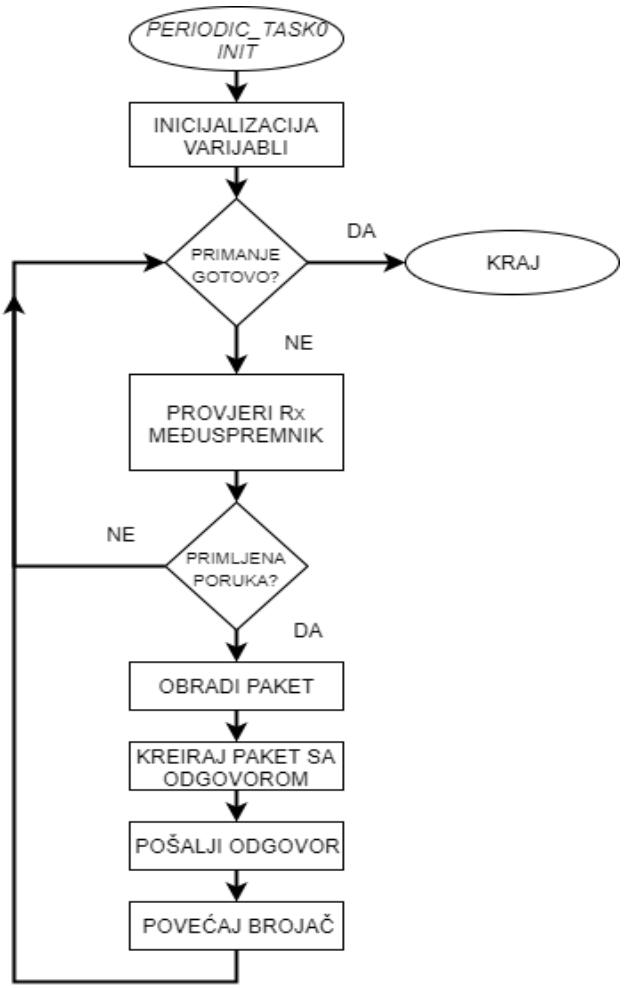
fazu. Bitno je napomenuti da se prijelaz između faza temelji na pretpostavki da nijedan paket neće biti izgubljen u prijenosu. Ako se samo jedan paket ne dostavi na Lightweight IP stranu ili obrnuto (ovisno o fazi mjerenja), gubi se cijela dinamika mjerenja. Preostale faze mjerenja su dosta slične fazi primanja poruka, te je dovoljno prikazati ih dijagramima toka (Sl. 3.10. – 3.12.).



Sl. 3.10. Dijagram toka faze primanja poruka



Sl. 3.11. Dijagram toka faze slanja poruka

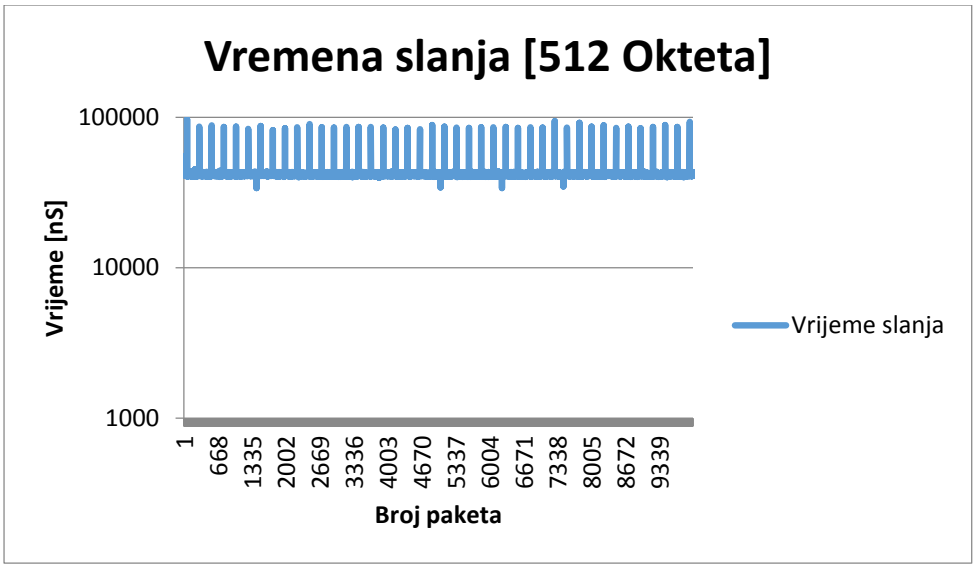


Sl. 3.12. Dijagram toka faze odgovora

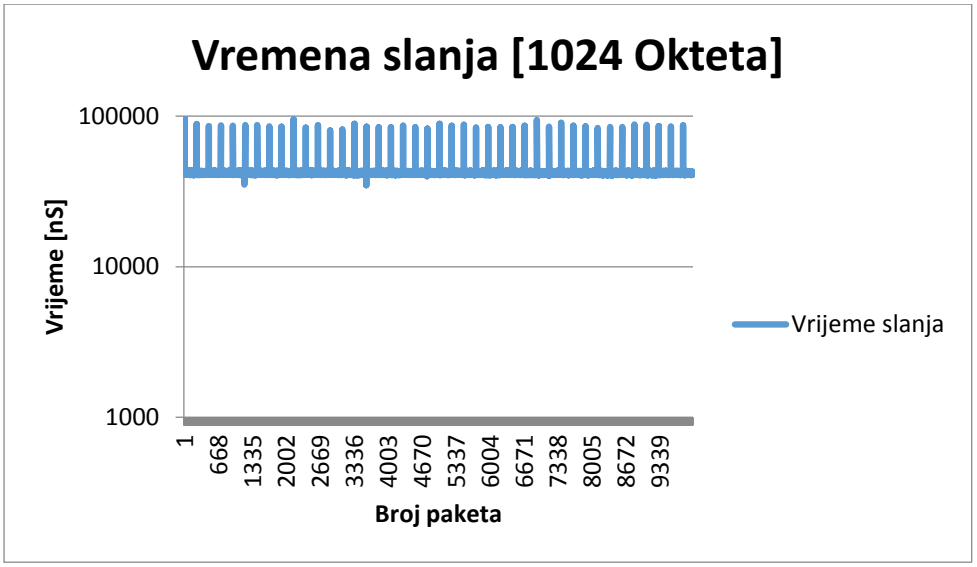
4. MJERENJA I REZULTATI

Pomoću programske podrške objašnjene u prošlom poglavlju, napravljena su mjerenja performansi prijenosa podataka putem UDP protokola za TTA Drive platformu. U ovom poglavlju će biti prikazani rezultati mjerenja i tumačenje istih.

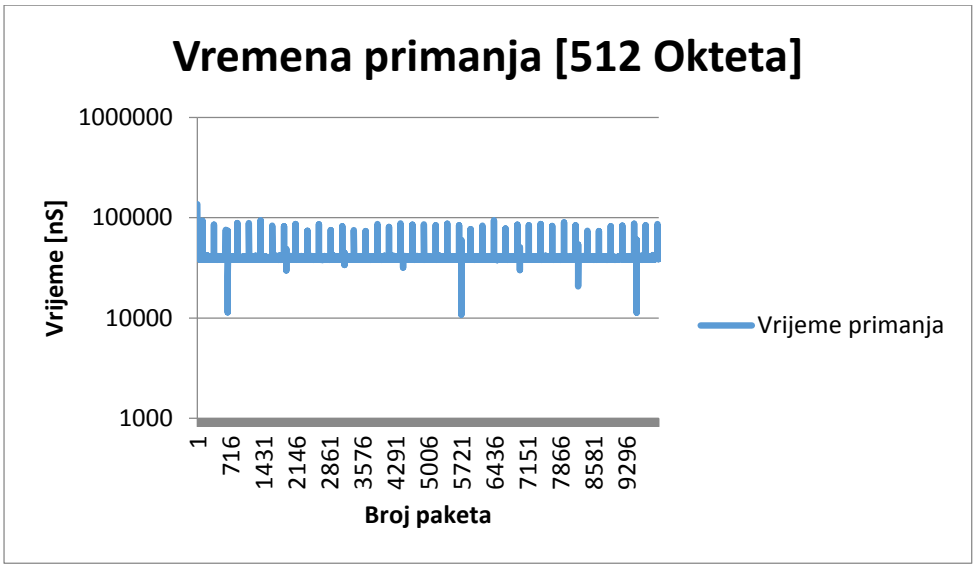
Jedan od zahtjeva mjerenja bio je izmjeriti utjecaj postojeće dinamike rada sustava na performanse slanja i primanja podataka. Arhitektura programa je ostala ista, za potrebe mjerenja promijenjena je samo Python skripta koja generira grafove. Umjesto grafova propusnosti svih podataka i propusnosti korisnih podataka, generirani su grafovi koji prikazuju vrijeme prijenosa svakog paketa (Sl. 4.1 – 4.6.). Odabrani grafički prikaz mjerenja je linijski graf jer se pomoću njega najbolje vidi periodičnost skokova u vremenima prijenosa. Grafovi propusnosti i vremenskog kašnjenja nisu generirani jer nisu relevatni za trenutno mjerenje.



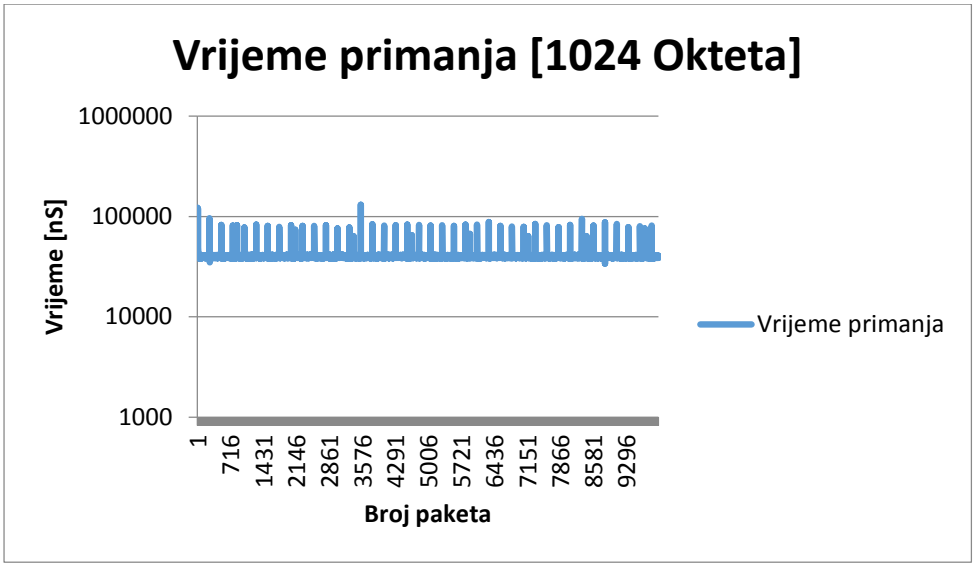
Sl. 4.1. Grafički prikaz vremena slanja paketa duljine 512 okteta



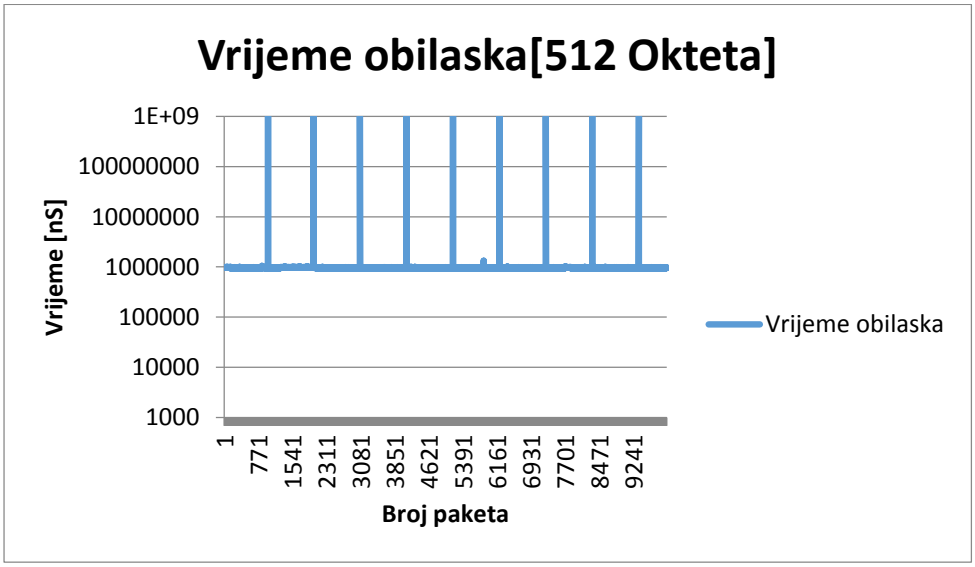
Sl. 4.2. Grafički prikaz vremena slanja paketa duljine 1024 okteta



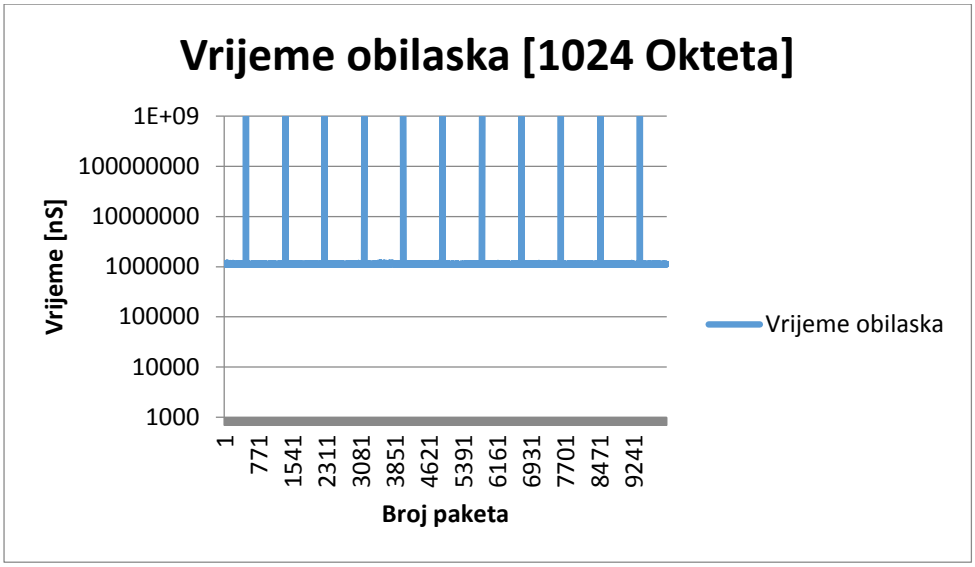
Sl. 4.3. Grafički prikaz vremena primanja paketa duljine 512 okteta



Sl. 4.4. Grafički prikaz vremena primanja paketa duljine 1024 okteta



Sl. 4.5. Grafički prikaz vremena obilaska paketa duljine 512 okteta



Sl. 4.6. Grafički prikaz vremena obilaska paketa duljine 1024 okteta

Na prikazanim grafovima može se vidjeti utjecaj Linux raspodjelitelja zadataka (engl. *Scheduler*). Periodično se pojavljuju skokovi u sve tri faze mjerenja (mjerenje performansi slanja, mjerenje performansi primanja i mjerenje performansi obilaska). Što je veća duljina podataka koji se prenose, to je razmak, odnosno period između vremenskih skokova manji. Za vrijeme obilaska vremenski skokovi su još veći. Uzrok većim skokovima u mjerenju performansi obilaska je čekanje na red za izvršavanje, ali i čekanje da se paket vrati sa procesora kojemu je poslan. Iz grafova se može zaključiti da je za slanje do 250 uzastopnih paketa bolje slati pakete veličine 512 okteta jer je mala vjerojatnost povećanja vremena kašnjenja. Za broj uzastopnih paketa iznad 250 bolje je slati pakete veličine 1024 jer je vrijeme između skokova manje, a s manjom duljinom podataka, potreban je veći broj paketa koji se šalje pa se može dogoditi da prijenos bude sporiji. Odabir duljine paketa za ostala mjerenja je analogan mjerenju za slanje, osim što je granični broj paketa za primanje oko 420, ako se izuzmi skokovi na samom početku i 1020 paketa za cijeli obilazak.

Pomoću dobivenog broja uzastopnih paketa koji se mogu poslati prije nego što se dogodi skok, može se izračunati period pojavljivanja skoka. Na primjer, testna aplikacija je u fazi mjerenja vremena slanja za duljinu podatka od 512 okteta, izračunala medijan vrijeme u iznosu od 41220 nano sekundi. Iz grafa je očitano da se skokovi pojavljuju svakih 250 uzastopno poslanih paketa. Ako se uzme da je vrijeme slanja svakog paketa medijan vrijeme, množenjem medijan vremena i

broja uzastopnih paketa dobije se vrijeme od 10.305 mili sekundi. Izračunato vrijeme je period pojavljivanja skoka. Na isti način su izračunati periodi skokova za preostala mjerenja.

Nakon dobivenih rezultata dinamike rada programa, izmjerene su veličine koje definiraju brzinu prijenosa podataka. Ponovno su mjerena vremena između svakog slanja i/ili primanja poruke. Vremena su spremana u niz koji se pri završetku faze mjerenja sortira. Odabire se element iz sredine niza koji predstavlja medijan vrijednost vremena. Medijan je odabran jer je bolja mjera prikaza ponašanja sustava od aritmetičke sredine zato što razdioba nije Gaussova. Nakon izračuna medijan vremena, računa se broj paketa u sekundi. Sva izmjerena vremena su u nano sekundama pa da bi se dobio broj paketa u jednoj sekundi potrebno je podijeliti broj nano sekundi u jednoj sekundi sa medijan vremenom jer je to vrijeme koje predstavlja vrijeme prijenosa jednog paketa. Na primjer, ako je vrijeme prijenosa jednog paketa 5000 nano sekundi, onda se u jednoj sekundi može poslati 200 000 paketa (Formula 4.1.). Pomoću izmjerenog broja paketa koji se prenese u jednoj sekundi moguće je izračunati ukupnu propusnost i propusnost korisnog sadržaja. Propusnost je količina podataka koji se prenese u jednoj sekundi, dok je propusnost korisnog sadržaja, količina korisnih podataka koji se prenesu u jednoj sekundi. Pri računanju propusnosti uzima se u obzir i zaglavlje svakog podatkovnog paketa, dok se za propusnost korisnog sadržaja uzima samo duljina podatka koji se prenosi. Formule prema kojima su računate ove vrijednosti vidljive su u formulama 4.2. i 4.3.. Duljina zaglavlja koja je dodana pri računanju propusnosti iznosi 52 okteta. Duljina *Ethernet* zaglavlja je 24 okteta. Unutar *Ethernet* paketa inkapsuliran je IPv4 paket sa zaglavljem duljine 20 okteta i unutar njega je inkapsuliran UDP paket sa zaglavljem 8 okteta. Kada se zbroje duljine svih zaglavlja dobije se 52 okteta. Izračunate vrijednosti se nalaze u tablicama 4.1. – 4.3., a grafički prikaz promjene propusnosti i propusnosti korisnog sadržaja i njihove ovisnosti s obzirom na veličinu paketa vidljiv je na slikama 4.7. – 4.9..

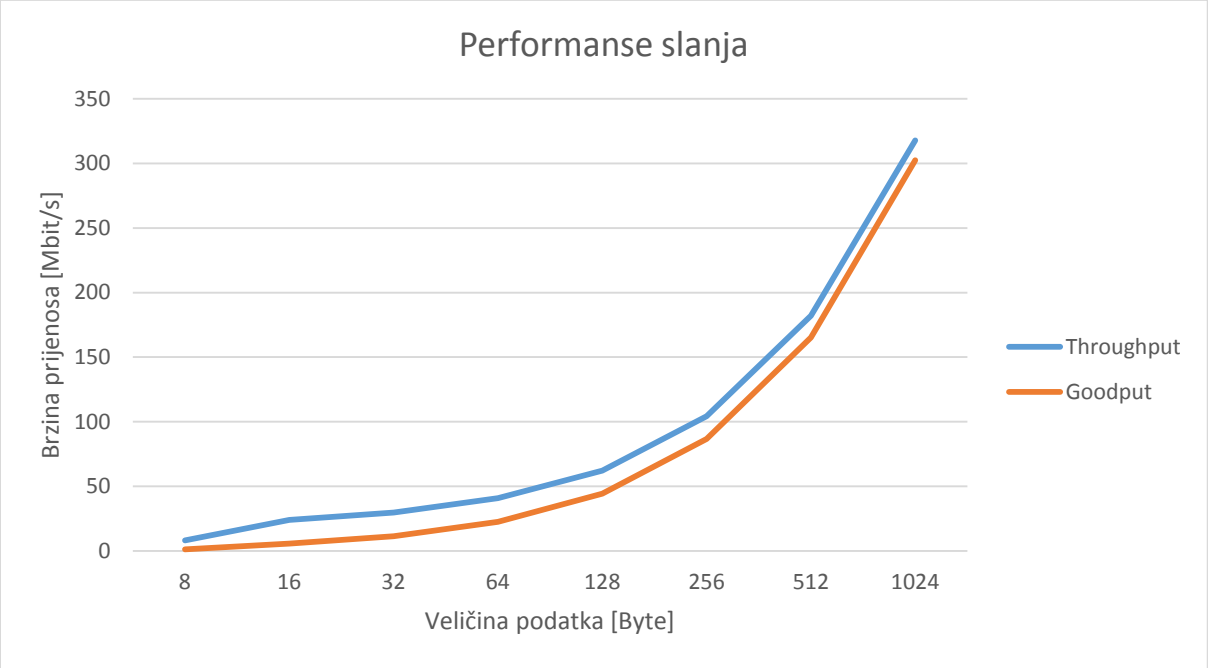
$$n_{paket\ po\ sec} = \frac{10^9}{t_{med}} \tag{4-1}$$

$$throughput = n_{paketa} * (duljina_{paketa} + duljina_{zaglavlja}) \tag{4-2}$$

$$goodput = n_{paketa} * duljina_{paketa} \quad (4\text{-}3)$$

Veličina paketa	Broj paketa po sekundi	Propusnost u Mbit/s	Propusnost korisnog sadržaja u Mbit/s
8	17815	8,15506	1,087341309
16	46190	23,96332	5,638427734
32	45910	29,5806	11,26879883
64	46042	40,74762	22,48144531
128	45226	62,10846	44,16601563
256	44337	104,1855	86,59570313
512	42308	182,0504	165,265625
1024	38731	317,9516	302,5859375

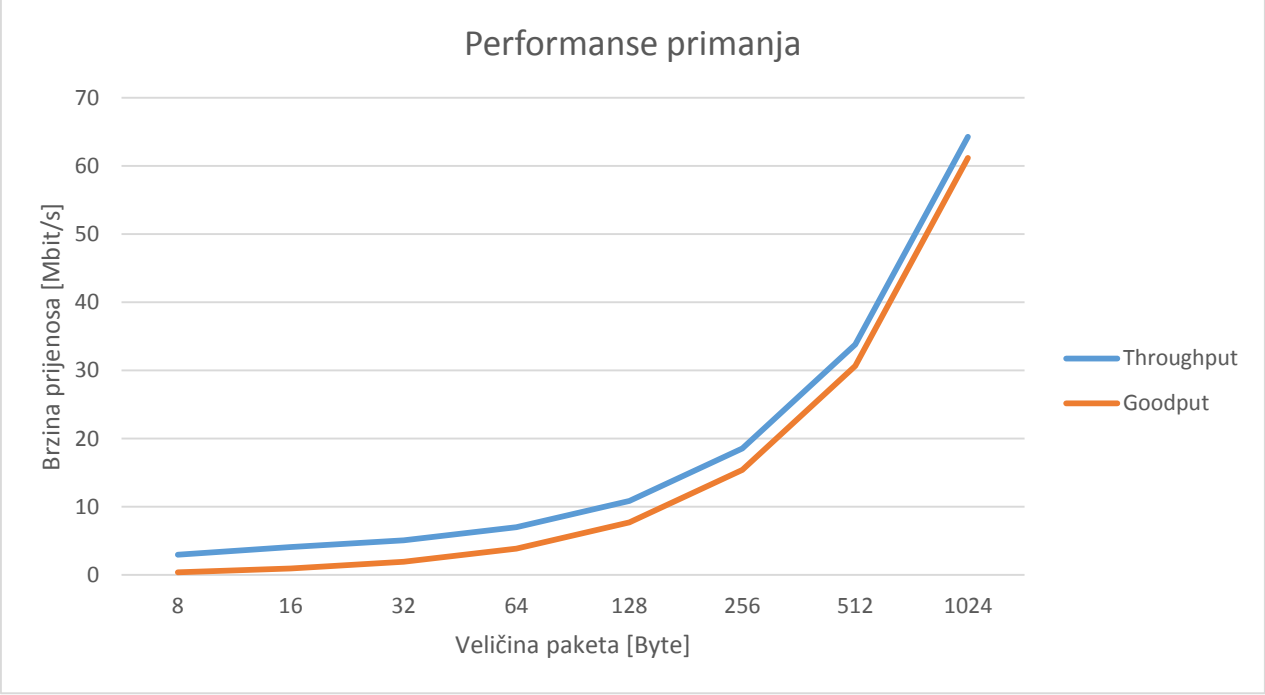
Tab. 4.1. Veličine izmjerene tokom analize performansi slanja



Sl. 4.7. Graf propusnosti za analizu performansi slanja

Veličina paketa	Broj paketa po sekundi	Propusnost u Mbit/s	Propusnost korisnog sadržaja u Mbit/s
8	6426	2,941589	0,392212
16	7915	4,106293	0,966187
32	7910	5,069275	1,931152
64	7905	6,996002	3,859863
128	7905	10,85587	7,719727
256	7886	18,53094	15,40234
512	7854	33,79559	30,67969
1024	7830	64,27826	61,17188

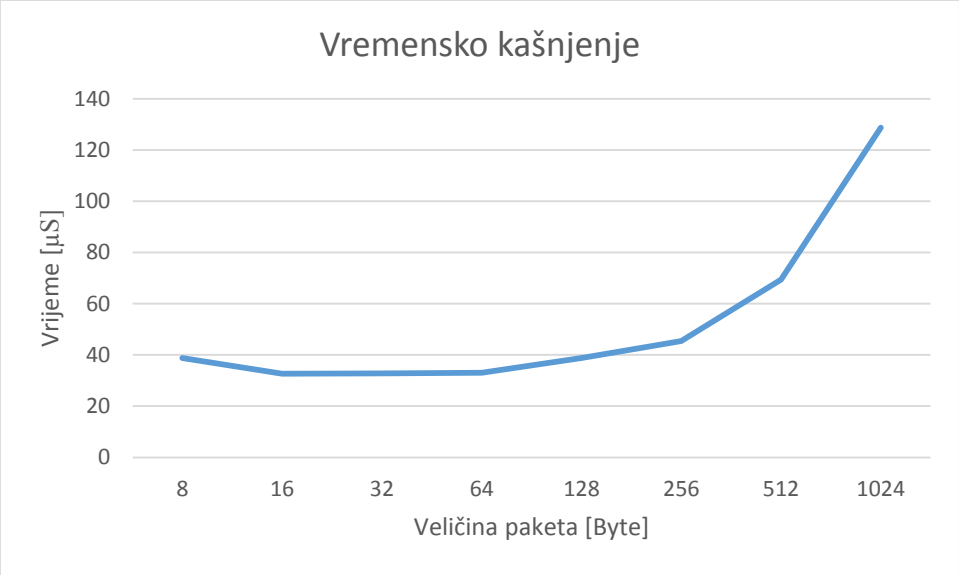
Tab. 4.2. Veličine izmjerene tokom analize performansi primanja



Sl. 4.8. Graf propusnosti za analizu performansi primanja

Veličina paketa	Broj paketa po sekundi	Vrijeme kašnjenja u µS	Propusnost u Mbit/s	Propusnost korisnog sadržaja u Mbit/s
8	23981	38,82891978	10,97763	1,4636841
16	30623	32,65519381	15,88718	3,7381592
32	30477	32,81162844	19,53177	7,4406738
64	30209	33,10271773	26,73526	14,750488
128	21434	38,75067814	29,43512	20,931641
256	22006	45,44215214	51,71088	42,980469
512	14415	69,37218176	62,02744	56,308594
1024	7767	128,7498391	63,76108	60,679688

Tab. 4.3. Veličine izmjerene tokom analize performansi obilaska



Sl. 4.9. Graf vremenskog kašnjenja za analizu obilaska paketa kroz mrežu

Iako je veza između procesora brzine 100 Mbit/s, mjerenja performansi slanja pokazuju brzine preko 100 Mbit/s. Uzrok tome je nemogućnost uzimanja vremenskog uzorka u trenutku kada podatak bude pušten na sabirnicu od Linux strane. Brzine izračunate u ovom mjerenju su zapravo brzine formiranja *Ethernet* paketa i spremanja istog u odlazni međuspremnik. Mjerenje brzine primanja podataka je od veće važnosti. Budući da u mjerenju performansi primanja poruka, mjerna strana mora čekati svaki paket da pristigne u međuspremnik i provjeriti zaglavlje, ovo mjerenje je korisnije. Iz grafa se može vidjeti da brzine prijenosa naglo rastu nakon duljine paketa u iznosu od

128 okteta. Ponašanje prikazano grafom je normalno ponašanje mreže jer je iskoristivost brzine prijenosa unutar mreže manja za manje veličine paketa. Mjerenja su odrađena do duljine paketa u iznosu od 1024 okteta. Prateći trend rasta krivulje na grafu performansi primanja, za maksimalnu duljinu paketa (MTU) u iznosu od 1500 okteta, očekivana iskoristivost propusnosti je oko 90%. Pri većim duljinama paketa iskorištava se više dostupne propusnosti, ali se javlja problem povećanja vremenskog kašnjenja, odnosno vremena potrebnog da se podatak obradi, pošalje i uspješno primi na drugoj strani. Mjerenjem performansi obilaska mjeri se vrijeme kašnjenja koja se iskazuje u vremenskim jedinicama. Prvi vremenski uzorak se uzima pri formiranju UDP paketa na Linux strani, a drugi vremenski uzorak se uzima kada je paket uspješno vraćen Linux strani i kada je njegov sadržaj obrađen. Vrijeme kašnjenja je vrijeme koje se dobije oduzimanjem drugog vremenskog uzorka s prvim. Na grafu 4.7. se vidi krivulja vremenskog kašnjenja mjerenog u mikrosekundama. Na temelju dobivenog grafa može se odrediti vrijeme potrebno da se podatak dostavi prijemnoj strani za odabranu duljinu podatka. Na temelju te informacije se može znati kolika mora biti maksimalna duljina podatka, ako je sustav vremenski kritičan.

5. ZAKLJUČAK

U radu je opisana *Ethernet* komunikacija po slojevima i problemi koji se javljaju pri razmjeni podataka putem *Etherneta*. Razmjena poruka putem TCP/IP protokolskog stoga se obavlja preko četiri sloja. Svaki sloj sadrži protokole neophodne za jednoznačno definiranje pravila za prijenos na toj razini. U radu su analizirani aplikacijski i transportni sloj. Analiza je obavljena pomoću aplikacije koja je slala određeni broj paketa veličina: 8, 16, 32, 64, 128, 256, 512 i 1024 okteta. Performanse aplikacijskog sloja su mjerene analiziranjem rada raspodjelitelja zadataka Linux operacijskog sustava, a performanse transportnog sloja, točnije UDP protokola izmjerene su mjerenjem propusnosti (engl. *Throughput*), propusnosti korisnih podataka (engl. *Goodput*) i vremenskog kašnjenja (engl. *Latency*). Analizom aplikacijskog sloja uočeni su periodični skokovi u vremenu slanja podataka. Periodični skokovi su definirani brojem uzastopnih poslanih paketa, a ne vremenom slanja. Analizom UDP protokola izračunata je propusnost Ethernet veze između procesora u iznosu od 60 Mbit/s za veličine paketa u iznosu od 1024 okteta. Za maksimalnu prijenosnu jedinicu očekivana propusnost je oko 90%. Pomoću analize obilaska paketa kroz mrežu analizirano je vrijeme kašnjenja za svaku definiranu veličinu paketa. Analizom je izračunat nagli skok u vremenskom kašnjenju za duljine podataka veće od 128 okteta. Na temelju provedenih analiza može se predvidjeti ponašanje sustava u određenim situacijama što uvelike pomaže pri dizajniranju programskih rješenja za navedenu platformu.

LITERATURA

[1] C.E. Spurgeon, Ethernet: The Definitive Guide, O'Reilly & Associates, Inc, Sebastopol, 2000.
[2] S.B. Cooper, What Is an Ethernet Preamble?, https://www.techwalla.com/articles/what-is-an-ethernet-preamble, September, 2017.
[3] https://wiki.wireshark.org/Ethernet, September, 2017.
[4] B. Mitchell, https://www.lifewire.com/transmission-control-protocol-and-internet-protocol-816255, September, 2017.
[5] L. Parziale, D.T. Britt, C. Davis, J. Forrester, W. Liu, C. Matthews, N. Rosselot, TCP/IP Tutorial and Technical Overview, International Technical Support Organization, 2006.
[6] http://flylib.com/books/en/4.153.1.99/1/, September, 2017.
[7] https://www.warehousecables.com/landing/how-to-make-a-cat6-patch-cable.php, September, 2017.
[8] http://mars.netanya.ac.il/~unesco/cdrom/booklet/HTML/NETWORKING/node020.html, September, 2017.

SAŽETAK

Ovaj diplomski rad opisuje teorijsku podlogu *Ethernet* komunikacije kao i provedbu analize TCP/IP protokolskog stoga za 100 Base-TX Ethernet na TTTechovoj TTA-Drive ploči. Analiza se provodila nad komunikacijom dva procesora na TTA-Drive ploči u tri faze: analiza maksimalne brzine slanja podataka, maksimalne brzine primanja podataka i analiza obilaska mreže. Analiziran je utjecaj Linux raspodjelitelja zadataka (engl. Scheduler) na performanse slanja, problem omjera korisnog sadržaja i zaglavlja podatkovnog paketa (engl. *Goodput*), ukupne propusnosti (engl. *Throughput*) i povećanje vremenskog kašnjenja (engl. *Latency*) za različite duljine podataka. Na temelju provedene analize moguće je predvidjeti ponašanje sustava u određenim uvjetima i na taj način poboljšati arhitekturu programske podrške za navedenu ploču.

Ključne riječi: Ethernet, UDP, vrijeme kašnjenja, Linux, analiza, TTA-Drive, propusnost

ABSTRACT

This thesis describes the theoretical background of Ethernet communication as well as the implementation of TCP/IP protocol stack analysis for 100 Base-TX Ethernet on the TTTech TTA-Drive board. The analysis was conducted over the communication of two processors on the TTA-Drive board and in three phases: the analysis of maximum send rate, maximum receive rate and round-trip analysis. The impact of the Linux scheduler on the performance of sending data, the problem of the ratio of useful content and package header (Goodput), the overall Throughput, and the increase of the latency for different lengths of data were analyzed. Based on the performed analysis, it is possible to predict the behavior of the system under certain conditions and thus improve the architecture of the program support for the specified board.

Keywords: Ethernet, UDP, latency, Linux, analysis, TTA-Drive, throughput

ŽIVOTOPIS

Davor Tomljenović, rođen 6.10.1993. godine u Osijeku, student je 2. godine Diplomskog studija Računarstva, bloka Informacijske i podatkovne znanosti na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. Zanima ga automotiv, strojno učenje, te obrada slike i računalni vid. Trenutno je zaposlen u Institutu RT-RK kao dio Power Connexion 3 tima u sklopu TTTech grupe.