Razvoj i primjena aplikacije otvorenog koda za analizu toka paketa i mrežnog prometa

Hilc, Borna

SVEUČILIŠTE U ZAGREBU

FAKULTET PROMETNIH ZNANOSTI

DIPLOMSKI RAD

RAZVOJ I PRIMJENA APLIKACIJE OTVORENOG KODA ZA ANALIZU TOKA PAKETA I MREŽNOG PROMETA

DEVELOPMENT AND UTILIZATION OF OPEN SOURCE APPLICATION FOR PACKET FLOW AND NETWORK TRAFFIC ANALYSIS

Mentor: doc. dr. sc. Marko Matulin

Student: Borna Hilc

JMBAG: 0135236885

Zagreb, rujan 2021.

SVEUČILIŠTE U ZAGREBU
FAKULTET PROMETNIH ZNANOSTI
POVJERENSTVO ZA DIPLOMSKI ISPIT

Zagreb, 30. kolovoza 2021.

Zavod: **Zavod za informacijsko komunikacijski promet**
Predmet: **Tehnologija telekomunikacijskog prometa II**

DIPLOMSKI ZADATAK br. 6147

Pristupnik: **Borna Hilc (0135236885)**
Studij: Promet
Smjer: Informacijsko-komunikacijski promet

Zadatak: **Razvoj i primjena aplikacije otvorenog koda za analizu toka paketa i mrežnog prometa**

Opis zadatka:

Opisati karakteristike podatkovnih mreža koje se temelje na slojevitim arhitekturama. Osvrnuti se na različite protokole koji se koriste u prijenosu pojedinih vrsta prometnih tokova. Istražiti i opisati funkcionalnosti dostupnih aplikacija koje se koriste za analizu prometa u podatkovnim mrežama. Temeljem tih analiza, osmisliti i izraditi vlastitu aplikaciju za analizu prometa. Upotrijebiti aplikaciju u analizi mrežnog prometa na razini paketa i prometnog toka u različitim mrežnim scenarijima. Diskutirati rezultate.

Mentor:

doc. dr. sc. Marko Matulin

Predsjednik povjerenstva za diplomski ispit:

prof. dr. sc. Štefica Mrvelj

SAŽETAK

U ovome radu prikazana je arhitektura OSI (*Open Systems Interconnection*) modela i TCP/IP (*Transmission Control Protocol / Internet Protocol*) složaja. Objašnjeni su slojevi i njihove funkcije, te opisani najznačajniji protokoli koji se nalaze na svakom od slojeva te njihove zadaće i način rada.

Mjerenje mrežnog prometa te praćenje toka paketa pruža velik broj mogućnosti upravljanja i kontrole mreže, npr. korištenjem ovih funkcija moguće je primjerice uočiti razne anomalije unutar mreže, riješiti probleme koji usporavaju mrežu, kreirati stvarno vremenske ili povijesne zapise o događanjima unutar mreže. Stoga, izrađena je aplikacija za tu namjenu te su njen postupak izrade i način rada u cijelosti objašnjeni kao i funkcija svakog pojedinog elementa koji se nalazi unutar grafičkog sučelja.

Prikazan je način rada same aplikacije te njene mogućnosti pri prikupljanju i filtriranju prometa kao i prikaz informacija o prikupljenim paketima.

KLJUČNE RIJEČI: paket, TCP, IP, analiza mrežnog prometa, C#, aplikacija

SUMMARY

This paper presents the architecture of OSI (*Open Systems Interconnection*) model and TCP/IP (*Transmission Control Protocol / Internet Protocol)* suite and explains the functions of the protocol suite layers. The most important protocols on each layer are described, as well as their tasks and the way they work.

Measuring network traffic and tracking packet flow provides a large number of network management and control options, e.g., using these functions it is possible to detect various anomalies within the network, solve problems that slow down the network, create real-time or historical records of events within networks. Hence, a traffic monitoring application is created whose process of development is explained, including the use of the application itself, as well as the function of each individual element located within the graphical interface.

Use of the application is shown, including its ability to capture and filter network traffic while displaying information about the captured packets.

KEY WORDS: packet, TCP, IP, network traffic analysis, C #, application

Sadržaj

1. Uvod

U današnje vrijeme kada je skoro svaki korišteni uređaj dio neke mreže, bilo da se radi o klasičnim uređajima koji se spajaju na mrežu kao što su računala, pametni telefoni itd. ili o sve rastućem segmentu IoT (*Internet of Things*)uređaja, a u novije vrijeme čak i automobilima. Zbog toga, ali i sve kompleksnijeg i obujmom većeg prometa koji se prenosi mrežama bitno je pratiti i analizirati prometne tokove. Informacije o tome tko generira i kakav promet, koji se protokoli koriste te otkud promet potječe i koje mu je odredište predstavljaju iznimno važne informacije za rješavanje mogućih problema unutar mreže. Takvo rješenje pružaju alati za analizu mrežnog prometa, popularni zvani *packet snifferi*.

Temeljna svrha izrade ovog rada bila je bolje upoznavanje se procesima koji se događaju unutar mreže, načinom na koji se obavlja komunikacija između uređaja te prvenstveno interpretacijom i korištenjem dobivenih podataka od strane uređaja. Sam rad podijeljen je u sedam cjelina:

1. Uvod
2. Podatkovne mreže i slojevite arhitekture
3. Funkcionalnosti aplikacija za analizu mrežnog prometa
4. Izrada aplikacije za analizu toka paketa
5. Analiza toka paketa
6. Analiza mrežnog prometa
7. Zaključak.

Unutar drugog poglavlja opisana je arhitektura OSI (*Open Systems Interconnection*) referentnog modela i TCP/IP (*Transmission Control Protocol/Internet Protocol*) protokolarnog složaja. Objašnjene su funkcije svakog sloja te su opisani načini rada najpoznatijih protokola na svakom od slojeva TCP/IP složaja. Uz to objašnjen je i postupak enkapsulacije paketa po slojevima.

U trećem poglavlju analizirana su postojeća softverska rješenja za analizu mrežnog prometa i objašnjene njihove funkcionalnosti te način rada.

Četvrto poglavlje prikazuje postupak izrade same aplikacije, analizira grafičko sučelje i objašnjava funkcije pojedinih elementa uz objašnjenje njihova načina rada.

U petom poglavlju prikazan je način korištenja aplikacije za analizu toka paketa. Objašnjeno je korištenje filtera za filtriranje prometa prema određenim adresama i sadržaju zaglavlja paketa.

Šesto poglavlje prikazuje mogućnost korištenja aplikacije za analizu mrežnog prometa, koristeći statističke pokazatelje prikupljenih paketa.

Zadnje poglavlje donosi pregled cijelog rada te zaključke izvedene iz istog.

2. Podatkovne mreže i slojevite arhitekture

Mreža je grupa povezanih uređaja koji međusobno komuniciraju. Dok internet predstavlja skupinu dvije ili više mreža koje međusobno komuniciraju, a najpoznatija takva mreža je Internet. Internet predstavlja dobro strukturiran i organiziran sustav, a kako bi takav sustav mogao funkcionirati bitno je imati pravila i standarde. Prema oxfordskom rječniku protokol predstavlja skup pravila koji kontroliraju način na koji se podatci prenose između računala. Skup međusobno povezanih protokola naziva se složaj protokola (*protocol suite*), dok njegov dizajn određuje kako određeni protokoli složaja međusobno korespondiraju te dijele zadatke koje je potrebno izvršiti naziva arhitektura ili referentni model.

2.1 OSI referentni model

OSI (*Open Systems Interconnection*) je teoretski model koji pokazuje kako dva različita sustava mogu komunicirati jedan s drugim. Rad na OSI modelu pokrenuo je ISO (*International Organization for Standardization*) 1977.godine, te je prihvaćen 1984.godine kao ISO norma dokumentom ISO 7948, te kao preporuka CCITT (kasnije preimenovan u *International Telecommunication Union Telecommunication Standardization Sector* - ITU-T) X.200 na Plenarnoj sjednici CCITT-a [1]. OSI model predstavlja slojevitu arhitekturu za dizajn mrežnih sustava koji omogućavaju komunikaciju između svih tipova računala. Sastoji se od sedam slojeva koji su međusobno povezani te svaki sloj definira jedan dio unutar procesa prijenosa informacija mrežom [2]. Slojevi su prikazani na slici 1.

Slika 1. OSI referentni model [1]

2.1.1 Fizički sloj

Glavna razlika fizičkog sloja u odnosu na ostale slojeve OSI modela je ta da samo kod fizičkog sloja postoji fizički prijenos podataka. Ostali slojevi izvršavaju svoje funkcije pri kreiranju poruke ili čitanju poruke dok sam fizički sloj budući da se nalazi na kraju modela obavlja slanje toka bitova fizičkim medijem [3]. Uzimajući u vidu izvore [2] i [3], funkcije fizičkog sloja obuhvaćaju kako je navedeno u nastavku.

- Definiranje hardverskih specifikacija sučelja između kojih se obavlja prijenos te karakteristika medija putem kojih se prenose podatci (kablovi, konektori, mrežne kartice, bežični prijamnici itd.) .
- Prikaz bitova – podatci na fizičkom sloju sastavljeni su od niza bitova bez interpretacije. Kako bi takve bitove bilo moguće prenijeti mrežom potrebno ih je prebaciti (kodirati) u signale koji će putovati medijem (električne ili optičke ovisno o tipu medija), tu funkciju obavlja fizički sloj.
- Brzina prijenosa – definira broj poslanih bitova u sekundi odnosno trajanje određenog bita.
- Sinkronizacija bitova – satovi koji se koriste pri sinkronizaciji bitova pošiljatelja i primatelja moraju biti sinkronizirani.
- Topologija mreže – definira način na koji su uređaji unutar mreže povezani (isprepletena, zvjezdasta, prsten, sabirnička).
- Način prijenosa – definira smjer prijenosa podataka između dva uređaja (*simplex, half-duplex, full-duplex*).

Izvrstan primjer rada fizičkog sloja predstavljaju „glupi“ mrežni uređaji poput repetitora (engl. *repeater*) i koncentratora (engl. *hub*). Takvi uređaji rade isključivo na fizičkom sloju te nemaju nikakvo znanje o porukama koje primaju već samo prebacuju bitove s ulaza na izlaz uređaja [3].

2.1.2 Podatkovni sloj

Podatkovni sloj drugi je po redu sloj prema OSI modelu, te se na njemu nalaze razne mrežne tehnologije kao što su Ethernet, Token Ring, FDD, LAN itd. Može ga se podijeliti na dva podsloja LLC (*Logical Link Control*) koji služi za uspostavu i kontrolu logičkih linkova između dva uređaja na mreži te omogućava korištenje raznih tehnologija prikrivajući dio podatkovnoga sloja kako bi omogućio nesmetan rad viših slojeva bez obzira na korištenu tehnologiju. Drugi podsloj je MAC (*Media Access Control*), on se odnosi na procedure koje koriste uređaji kako bi kontrolirali pristup prijenosnom mediju. Budući da mnogo mreža dijeli prijenosni medij potrebno je imati pravila za upravljanje medijem kako ne bi došlo do sukoba [3]. Prema [2] i [3] ostale bitne funkcije fizičkog sloja su:

- Sastavljanje okvira – skup bitova dobivenih od mrežnog sloja pohranjuje se u podatkovne jedinice koje se nazivaju okviri (eng. *frames*), to je ujedno i zadnji korak enkapsulacije paketa te se okviri šalju mrežom putem fizičkog sloja.
- Fizičko adresiranje – podatkovni sloj dodaje dodatno zaglavlje okviru kako bi se definirali pošiljatelj i primatelj okvira, a u njemu se nalazi hardverska ili MAC adresa koja je jedinstvena za svaki uređaj kako bi se osiguralo da svaki uređaj dobi podatke namijenjene za njega.
- Kontrola toka – ako je brzina kojom primatelj može prihvatiti podatke manja od one kojom pošiljatelj šalje te podatke podatkovni sloj koristi mehanizam za kontrolu toka kako ne bi došlo do preplavljivanja primatelja.
- Kontrola pogreške – koristi mehanizme za detekciju i retransmisiju oštećenih okvira te prepoznavanje duplih okvira.
- Kontrola pristupa – kada su dva ili više uređaja spojena na istu vezu, protokoli podatkovnog sloja imaju zadaću utvrditi koji uređaj ima kontrolu nad tom vezom u koje vrijeme.

2.2.3 Mrežni sloj

Na mrežni sloj može se gledati kao na onaj koji je zadužen za isporuku paketa između različitih mreža, dok podatkovni sloj omogućava isporuku paketa odnosno raspoznavanje između uređaja na istoj mreži [2]. Funkcije mrežnog sloja prema [2] i [3] su:

- Logičko adresiranje – logička adresa dobiva se od strane protokola mrežnoga sloja, npr. IP (*Internet Protocol*) adresa. To znači da će paket koji dođe do mrežnog sloja dobiti zaglavlje s logičkom adresom pošiljatelja i primatelja, te tako kreirati datagram.
- Usmjeravanje paketa se izvodi na ovome sloju, uređaji koji obavljaju usmjeravanje koriste funkcionalnosti mrežnoga sloja.
- Fragmentiranje paketa – zbog određenih ograničenja podatkovnoga sloja pakete koji su preveliki je potrebno je razdvojiti, proslijediti svaki dio podatkovnome sloju te na primateljevoj strani ponovno sastavi fragmentirani paket.

2.2.4 Transportni sloj

Glavna značajka transportnog sloja je omogućavanje računalu da dođe do podataka. Odnosno predstavlja neku vrstu veze između nižih slojeva koji se bave prijenosom samih podataka od mreže do mreže i računala do računala i viših slojeva čija je zadaća više apstraktne prirode, gdje se podatci čitaju i interpretiraju. Kod prijenosa, transportni sloj prati od koje aplikacije dolaze koji podatci te ih spaja u jedan tok koji prosljeđuje nižim slojevima, na strani primatelja ti procesi su obrnuti, podatci se dijele te usmjeravaju odgovarajućem primatelju [3]. Funkcije transportnog sloja prema [2] i [3] su:

- Adresiranje portova – već prije spomenuto odvajanje podataka generiranih od strane različitih aplikacija izvodi se na način da se unutar zaglavlja transportnog sloja nalaze informacije o portu primatelja i pošiljatelja. Nakon što mrežni sloj isporuči paket ispravnom računalu u mreži, transportni sloj pomoću porta određuje proces ili aplikaciju za koju je ta poruka namijenjena.
- Segmentacija – predugačke poruke razdvajaju se u segmente koje je moguće prenijeti, te se uz svaki segment dodaje redni broj koji omogućava drugoj strani pri primanju takve poruke njeno ponovno sastavljanje.
- Kontrola veze – transportni sloj može biti konekcijski ili beskonekcijski ovisno o korištenome protokolu, kod konekcijskog se prije slanja paketa stvara veza sa odredištem te se nakon što su svi paketi prenijeti ta se veza raskida, dok je kod beskonekcijskog svaki paket tretiran individualno.
- Kontrola toka – protokoli transportnog sloja koji podržavaju pouzdanu dostavu često omogućavaju i kontrolu toka. Kako ne bi došlo do preplavljivanja primatelja na strani pošiljatelja se regulira brzina slanja. Razlika između ove kontrole toke i one na podatkovnome sloju je što se ovdje radi o *end-to-end* kontroli toka za razliku od kontrole na razini jedne veze kao što je slučaj kod podatkovnog sloja.

2.2.5 Sloj sesije

Kao što mu i ime kaže sloj sesije služi za kontrolu komunikacije, omogućava uređajima da uspostave i upravljaju sesijom. Sama sesija označava stalnu logičku vezu između dva procesa aplikacije kako bi im omogućila razmjenu podataka tijekom nekog vremena. Glavna zadaća ovoga sloja je kreiranja, uspostava te raskidanje sesija, te se mogućnosti ovoga sloja često nude kroz API (*Application Program Interfaces*) kojima se služe programeri kako bi svojim aplikacijama dodali mogućnost korištenja komunikacije putem TCP/IP protokola bez potrebe da znaju na koji način radi npr. korištenje mrežnih utičnica (eng. *socket*) [3]. Upravljanje dijalogom spada pod funkcije sloja sesija, ono omogućava dvaju sustavima da pokrenu dijalog u *half-duplex* ili *full-duplex* načinu rada. Budući da neke aplikacije rade samo u *half-duplex*-u kod ISO protokola postoji *data token* koji se izmjenjuje između dvaju sudionika te omogućava govor sudioniku samo kada posjeduje *data token*.

Još jedna važna značajka sloja sesije je sinkronizacija. Budući da je transportni sloj u mogućnosti ispraviti samo greške u komunikaciji, sinkronizacija se brine o ispravljanju greški na višim slojevima. Primjer bi bio prijenos određene datoteke, ako dođe do greške u zapisu na disk i ta greška se dogodi na 70% prijenosa morala bi se izvoditi ponovna retransmisija te datoteke. Uz sinkronizaciju moguće je tok podataka datoteke rastaviti na stranice te dodati sinkronizacijske točke između tih stranica, u tome slučaju ako dođe do greške retransmisija će se izvesti samo od sinkronizacijske točke umjesto cijele datoteke [2] [4].

2.2.6 Prezentacijski sloj

Prezentacijski sloj predstavlja sloj kojemu mu je kao što mu i ime govori glavna funkcija prezentacija podataka, izvori [2] i [3] navode sljedeće funkcije:

- Prijevod podataka – budući da dva različita sustava mogu koristiti različite tipove kodiranja podataka bitno je imati sloj koji omogućava interoperabilnost između takvih sustava. Prezentacijski sloj mijenja podatke iz formata koji koristi pošiljatelj u uobičajen format, nakon toga se na strani primatelja podatci ponovno mijenjaju u format korišten od strane sustava koji koristi primatelj.
- Enkripcija – određene vrste enkripcije vrše se na sloju prezentacije npr. SSL (*Secure Sockets Layer*), kako bi se osigurala sigurnost podataka koji putuje prema nižim slojevima.
- Kompresija – kompresiju je moguće odraditi na prezentacijskom sloju kako bi se povećala propusnost podataka.

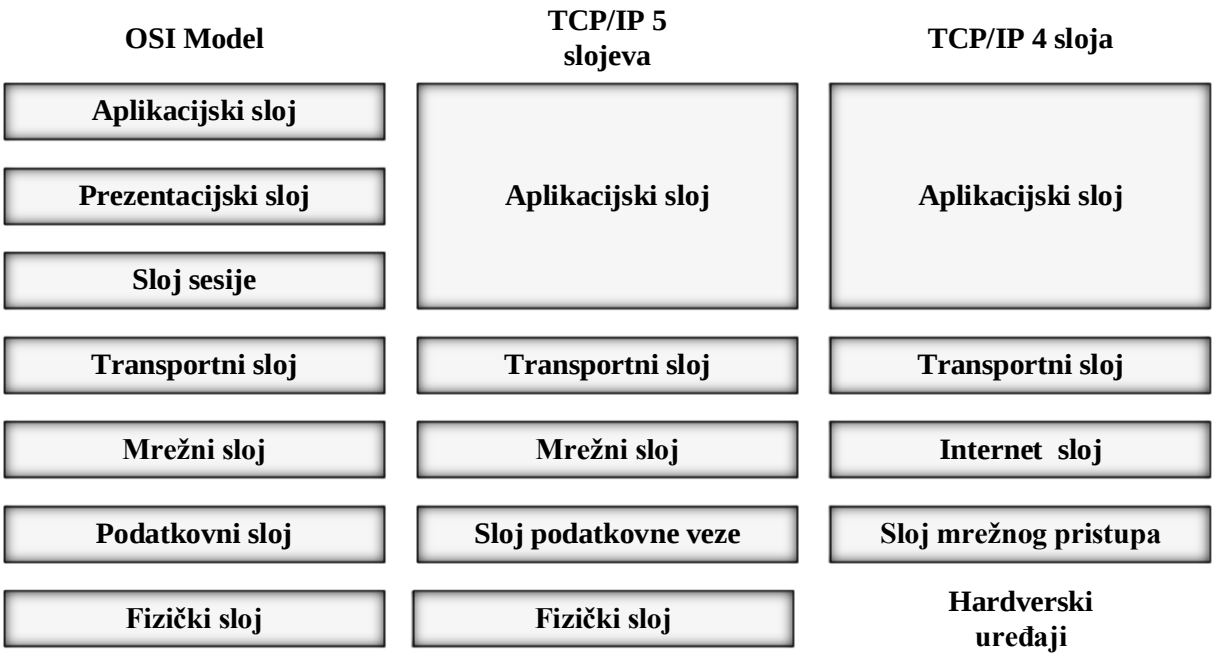
Budući da navedene funkcije nisu uvijek potrebne moguće je ne koristiti prezentacijski sloj. U tom slučaju protokoli sloja sesije i aplikacijski sloj komuniciraju izravno.

2.2.7 Aplikacijski sloj

Aplikacijski sloj omogućava korisniku, čovjeku ili softwareu komunikaciju putem mreže [2]. Na početku je bitno napraviti distinkciju što spada aplikacijski sloj, budući da se ovaj sloj naziva aplikacijski sloj moglo bi se zaključiti da su same aplikacije koje koristi krajnji korisnik dio ovoga sloja, iako tome nije tako. Svaka aplikacija namijenjena za komunikaciju mrežom koristi funkcionalnosti aplikacijskog sloja, iako ona sama nije dio istoga. Uz same aplikacije funkcionalnosti ovoga sloja koristi i sam operacijski sustav te njegove komponente. Protokoli koji se nalaze na aplikacijskom sloju većinom omogućavaju funkcionalnosti aplikacija koje koristi krajnji korisnik npr. internet preglednik (HTTP - *Hypertext Transfer Protocol*), email klijent (SMTP - *Simple Mail Transfer Protocol*, POP3 – *Post Office Protocol 3*), prijenos datoteka (FTP - *File Transfer Protocol*), tekstualnu komunikaciju (IRC - *Internet Relay Chat*, Telnet), prijevod adresa (DNS - *Domain Name System*) itd.

2.3 TCP/IP protokolarni složaj

TCP/IP protokolarni složaj dobio je ime po dvama najznačajnijim protokolima koji su njegov dio [3]. Sve do popularizacije Interneta devedesetih godina OSI model se smatrao standardnim modelom komunikacijskih mreža, dok se danas sveo više na model pomoću kojega se uči način funkcioniranja mreža te uvod u funkcije TCP/IP protokolarnog složaja [5]. TCP/IP protokolarni složaj usvojen je kao vojni standard 1983.g. te je predstavljao preduvjet za spajanje ARPANET (*Advanced Research Projects Agency Network*), mrežu razvijenu od strane istraživačkog odijela ministarstva obrane SAD-a [6]. 1980-ih se sve više računala spajalo na rastuću ARPANET mrežu, što je dovelo do onoga što je danas Internet. Tako je i TCP/IP uz Internet postao glavni skup protokola [3]. Za razliku od OSI modela koji se sastoji od sedam slojeva TCP/IP je izvorno zamišljen kao model od četiri sloja, iako ga se ovisno o izvoru može naći i kao model sastavljen od pet slojeva (slika 2) [2].



Slika 2. Usporedba modela protokolarnih složaja [2]

U ovome radu TCP/IP protokolarni složaj bit će opisan kroz model koji koristi četiri sloja: sloj mrežnoga pristupa, Internet sloj, transportni sloj i aplikacijski sloj. Na modelu ne postoji fizički sloj zato što se sloj mrežnoga pristupa smatra točkom gdje se nalazi veza TCP/IP složaja s mrežnim hardverom.

2.3.1 Sloj mrežnoga pristupa

Sloj mrežnoga pristupa definira postupke potrebne za povezivanje s mrežnim hardverom i pristup prijenosnom mediju, odnosno funkcije fizičkog i podatkovnog sloja gledajući prema OSI modelu. Zadaća ovoga sloja je prijenos okvira između uređaja dok su ostali postupci nad podatcima zadaća viših slojeva [7]. Postoje i određene kontroverze oko ovoga sloja budući da se niti jedan glavni protokol TCP/IP složaja ne nalazi na njemu, dapače na dosta TCP/IP mreža uopće ne postoji pripadajući protokol ovoga složaja zato što nije niti potreban. Ako se koristi Ethernet u kombinaciji sa TCP/IP on će preuzeti funkcije ovoga sloja, odnosno fizičkog i podatkovnog sloja gledano prema OSI modelu [3].

2.3.1.1 Point-to-Point Protokol

PPP (*Point-to-Point Protokol*) je protokol sloja mrežnoga pristupa te se često koristi za uspostavu direktne fizičke veze između dva čvora [7]. Prema Cisco-u, PPP pruža standardnu metodu prijenosa datagrama s više protokola između dva čvora te se sastoji od tri glavne komponente [8]:

- Metode za enkapsuliranje datagrama s više protokola.
- LCP (*Link Control Protocol*) za uspostavljanje, konfiguriranje i testiranje podatkovne veze.
- NCP (*Network Control Protocol*) za uspostavljanje i konfiguriranje različitih protokola mrežnoga sloja.

2.3.1.2 Ethernet

Prva Ethernet specifikacija razvijena je 1980.g. od strane Digital Equipment Corporation, Intel-a i Xerox-a, te je nazvana DIX standard. Druga i finalna razvijena je 1982.g. popularno zvani Ethernet II [9]. U to vrijeme je i IEEE (*Institute of Electrical and Electronics Engineers)* razvijao otvoreni standard kroz radnu grupu 802.3 pod nazivom „IEEE 802.3 *Carrier Sense Multiple Access with Collision* Detection (CSMA/CD) *Access Method and Physical Layer Specifications*“. Ta tehnologija iako se zapravo naziva 802.3 CSMA/CD često je se referira kao Ethernet. DIX V2.0 i 802.3 mogu zajedno koegzistirati na mreži, ali između njih postoje određene razlike koje su vidljive u tablicama 1 i 2 [10].

Tablica 1. DIX Ethernet okvir [7]

Preamble	Destination Address	Source Address	Type	Data	CRC
8 bajta	6 bajta	6 bajta	2 bajta	do 1500 bajta	4 bajta

Tablica 2. IEE 802.3 Ethernet okvir [7]

Preamble	SFD	Destination Address	Source Address	Length	LLC	Data	Pad	FCS
7 bajta	1 bajt	6 bajta	6 bajta	2 bajta	do 1500 bajta			4 bajta

Prema izvoru [7] Ethernet okviri sastoje se od tri segmenta: zaglavlja , sadržaja i podnožja (*trailer*)

- Zaglavlje
  - *Preamble* - 7-bitna sekvenca koja služi za sinkronizaciju prijemnika s odašiljačem.
  - *Start Frame Delimiter* - 8-bitna sekvenca (10101011).
  - *Destination Address* – odredišna MAC adresa.
  - *Source Address* – izvorišna MAC adresa.
  - *Type* – označava protokol kojim se šalje okvir (samo DIX).
  - *Length* – duljina podatkovnog polja (*data*) (samo IEEE 802.3).
- Sadržaj (*Payload*)
  - LLC – kontrolira prikupljanjem podataka na sloju podatkovne veze.
  - *Pad* – dodaje bitove u podatkovno polje ako u njemu ima manje od 46 bitova.
- Podnožje (*Trailer*)
  - *Cyclical Redundancy Check* (CRC) – detekcija greške pri prijenosu (samo DIX).

- *Frame Check Sequence* (FCS) – detekcija greške pri prijenosu i pružanje QoS-a (*Quality of Service*) na kraju primatelja.

Glavne funkcije Etherneta na fizičkom sloju su slanje i primanje toka bitova preko korištenoga prijenosnog medija i detekcija kolizije. Funkcije na sloju podatkovne veze mogu se podijeliti na funkcije na MAC podsloju i LLC podsloju [9]. Na MAC podsloju je to komunikacija Ethernet – Ethernet gdje se obavlja razmjena okvira unutar kojih je enkapsuliran datagram Internet sloja. Kako bi znao gdje poslati okvir koristi MAC adresu. Radi se o 48 bitnoj adresi koja je globalno jedinstvena te ju većinom dodjeljuje sam proizvođač mrežne opreme. Dio MAC adrese je identifikator proizvođača koji je dodijeljen od strane IEEE, dok preostali dio dodjeljuje sam proizvođač. Svaki uređaj koji je spojen na Ethernet LAN mora imati MAC adresu, to uključuje računala, usmjerivače, mrežne printere itd. Druga funkcija na ovome sloju odnosni na pristup prijenosnom mediju, koristeći već spomenutu CSMA/CD funkciju koja uređuje način na koji računala na mreži dijele kanal. CSMA/CD radi na temelju sljedećih pretpostavki [7]:

- Ethernet uređaj moći će prepoznati ako drugi uređaj koristi prijenosni medij tako da detektira njegov noseći signal.
- Dva Ethernet uređaja mogu nehotice koristiti prijenosni medij istodobno, uzrokujući koliziju okvira koja zahtjeva retransmisiju izgubljenih okvira. Ethernet uređaj bit će u mogućnosti detektirati tu koliziju okvira.

Kada je računalo na Ethernet LAN mreži dužno poslati informacije koristi sljedeći algoritam (prema [7]):

- Glavna procedura
  1. Ethernet okvir spreman je za prijenos.
  2. Ethernet uređaj osluškuje postoji li još koji uređaj koji koristi isti kanal. Ako medij nije u stanju mirovanja čeka periodični poziv koji se naziva *inter-frame-gap.*
  3. Ethernet uređaj započinje prijenos.
  4. Ako je došlo do kolizije okvira poziva se procedura koja je za to namijenjena (objašnjeno ispod).
  5. Prijenos okvira uspješno završen.
- Procedura pri detekciji kolizije
  1. Nastavlja se prijenos dok se ne dosegne minimalno vrijeme paketa kako bi se osiguralo da su svi primatelji detektirali koliziju.
  2. Ako je dosegnut maksimalan broj pokušaja prijenosa prekida se prijenos.
  3. Izračunava se i čeka slučajan *back-off* period.
  4. Ponovno se pokušava izvršiti glavna procedura.

Funkcije na LLC podsloju su osiguravanje pouzdanosti podataka i opskrba kanala viših slojeva podatcima.

2.3.2 Internet sloj

Ovaj sloj predstavlja ekvivalent mrežnome sloju OSI modela. Budući da se niži slojevi primarno bave spajanjem uređaja na mrežu, slanjem signala, lokalnom razmjenom podatka, ovaj sloj predstavlja prvi sloj koji se ne bavi hardverskim pitanjima. Kao što mu i ime sugerira ovaj sloj je najbitniji kada je mrežna komunikacija u pitanju [3]. Funkcije su: definiranje datagrama koji predstavlja osnovnu jedinicu prijenosa podataka na Internetu, usmjerivanje tih datagrama mrežom, ponovno sastavljanje onih paketa koji nisu došli valjanim redom, fragmentacija prevelikih paketa te njihovo ponovno sastavljanje na odredištu. Protokoli koji se koriste na ovome sloju su: IP, ARP (*Address Resolution Protocol*), RARP (*Reverse Address Resolution Protocol*), ICMP (*Internet Control Message Protocol*), IGMP (*Internet Group Message Protocol*) [11].

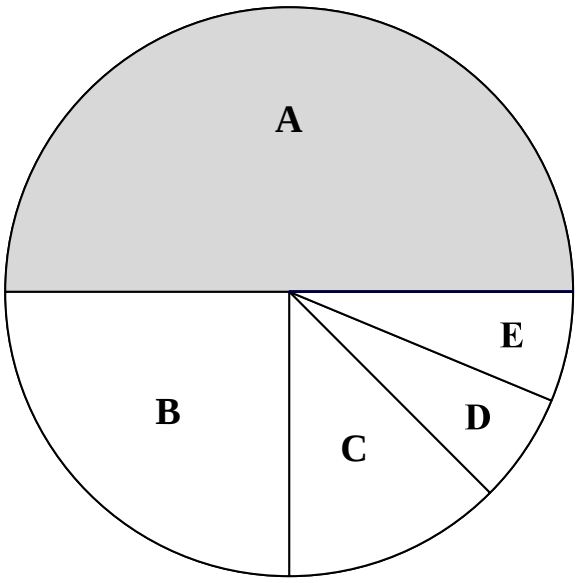
2.3.2.1 Internet Protokol

IP predstavlja „glavni“ protokol TCP/IP složaja, svi podatci TCP, UDP, ICMP, IGMP protokola prenose se IP datagramima [12]. Glavna zadaća IP-a je prijenos podataka između mreža, odnosno između uređaja koji se nalaze na različitim mrežama. Radi se o bezkonekcijskom protokolu, kako sve aplikacije ne zahtijevaju funkcije kao što su garancija isporuke, provjera grešaka. Te se funkcije ostavljaju protokolima viših slojeva kako ne bi nepotrebno dolazilo do gubitka performansi koje te funkcije uzrokuju iako nisu uvijek potrebne [12]. Ako dođe do problema sa dostavom datagrama IP se oslanja na funkcionalnosti ICMP protokola, datagram se uništava što stvara ICMP poruku koja biva poslana izvorišnom poslužitelju sa opisom problema [13] (detaljnije u poglavlju 2.3.2.2). Trenutno su u uporabi dvije inačice IP protokola IPv4 (*Internet Protocol Version 4*) i IPv6 (*Internet Protocol version 6*).

2.3.2.1.1 Internet Protocol Version 4

IP adresa predstavlja identifikator uređaja spojenog na Internet. IPv4 je 32 bitna adresa koja jedinstveno i univerzalno definira vezu poslužitelja ili usmjerivača na Internet, odnosno ona predstavlja adresu uređaja kojim se spaja na Internet. Svaka adresa definira jedan i samo jedan uređaj spojen na Internet, dva uređaja nikada ne mogu imati istu adresu u isto vrijeme. Adresni prostor odnosno broj adresa korištenih od strane protokola iznosi 2^b gdje *b* predstavlja broj bitova adresnog prostora. Kod IPv4 protokola taj broj bitova iznosi 32 bita, koja kada se ubace u ranije navedenu formulu daju 2^{32} ili 4 294 967 296 različitih adresa [2]. Za zapis samih adresa većinom se koristi decimalni zapis s točkama, rjeđe binarni ili heksadekadski.

Kod IPv4 protokola postoje klase koje su u početku bile zamišljene kako bi se olakšao posao adresiranja. Radi se pet klasa, klase A, B i C zauzimaju većinu adresnoga prostora što je vidljivo na grafikonu 1. Te se klase koriste za *unicast* adresiranje, odnosno slanje poruke samo jednome mrežnome sučelju.



Klasa A: 2^{31} = 2,147,483,648 adresa, 50%

Klasa B: 2^{30} = 1,073,741,824 adresa, 25%

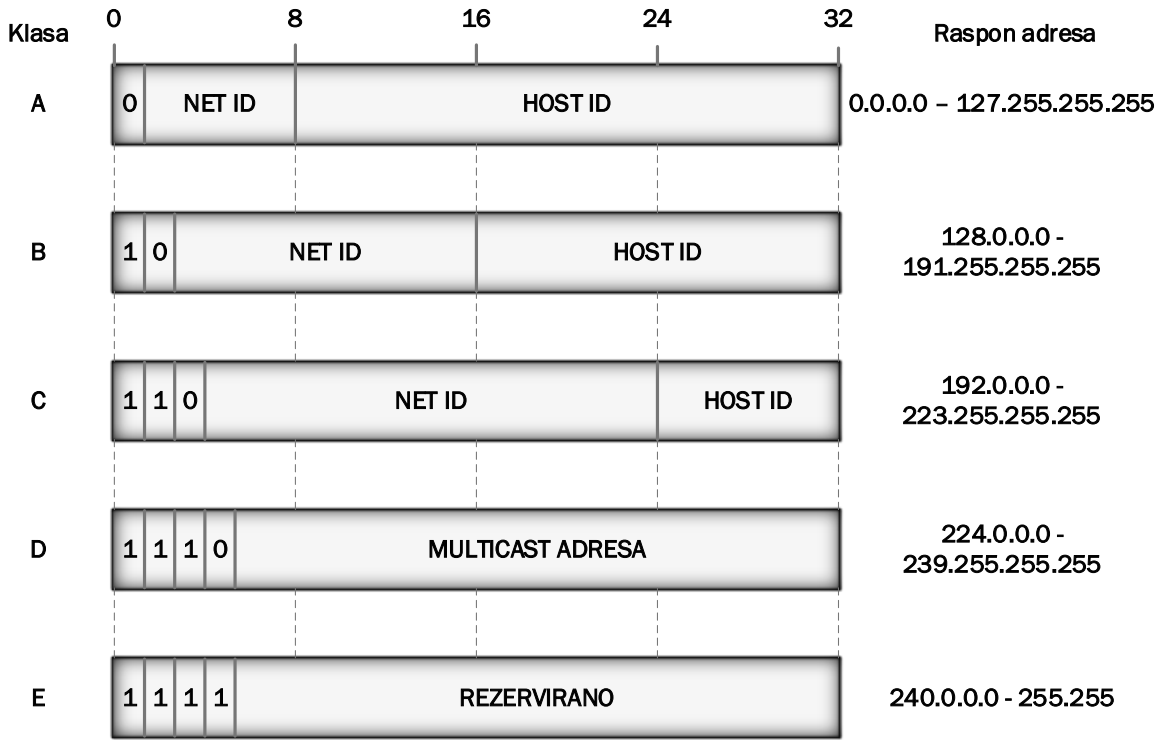
Klasa C: 2^{29} = 536,870,912 adresa, 12.5%

Klasa D: 2^{28} = 268,435,456 adresa, 6.25%

Klasa E: 2^{28} = 268,435,456 adresa, 6.25%

Grafikon 1. IPv4 raspodjela klasa [2]

Budući da je Internet bio zamišljen kao mreža skupa mreža klase imaju ovakav oblik. Ovisno o veličini organizacije odnosno broju računala/poslužitelja koji su se nalazili na toj mreži bile su dodjeljivane određene klase. Ako se pogleda struktura klasa na slici 3 može se vidjeti različiti broj bitova dodijeljen Net ID-u i Host-ID-u (klase A, B i C). Kod klasa A vidljivo je da nakon što se izuzme prvi bit koji služi za klasifikaciju klase ostaje 7 bitova, što znači da je moguće kreirati 2^7=128 blokova, odnosno moguće je kreirati 128 mreža koje će koristiti klasu A adresa gdje će svaki blok sadržavati 2^{24}= 16 777 216 adresa (potrebno je naglasiti da se prva i zadnja adresa ne mogu koristi jer su rezervirane uz još nekolicinu adresa za posebnu uporabu). Na istom principu rade i B i C adrese uz naravno veći broj blokova i manji broj adresa kao što je i vidljivo na slici 3. Ovo je izrazito jednostavan način usmjeravanja budući da usmjerivač na temelju prvih par bitova određuje klasu IP adrese koja ima fiksan Net i Host ID te zna točno gdje odvojiti ta dva dijela ovisno o klasi adrese.



Slika 3. IPv4 struktura klasa [2]

Sve veći broj uređaja koji imaju mogućnost spajanja na Internet doveo je do novoga načina usmjeravanja. Glavni problem klasa bio je taj što je nedostajalo IP adresa, primarno onih B klase. Srednje velikim organizacijama adresni prostor klase C bio je premali dok je korištenje klase B značilo veliki broj neiskorištenih adresa. Uz to došlo je i do značajnoga rasta tablica usmjeravanja te u konačnici mogućnošću iscrpljivanja IPv4 adresnoga prostora. Rješenja za prva dva problema je bio odbacivanje sustava klasa u korist CIDR-a (*Classless Inter-Domain Routing*) [14]. CIDR je baziran na varijabilnoj duljini subnet maske, to mu omogućava kreiranje prefiksa (Net ID) varijabilne duljine. CIDR IP adrese se sastoje od dvije komponente prefiksa i sufiksa (Host ID). Prefiks je oblika kao normalna IP adresa npr. 192.255.255.255 dok se sufiks piše iza oznake *slash* (npr. /12), čime se dobiva adresa koja u ovome slučaja izgleda ovako : 192.255.255.255/12 . Budući da kod CIDR-a ne postoje klase, sufiks označava na kojemu bitu se odvaja Host ID od Net ID-a. U ovome primjeru bi to značilo da prvih 12 bitova predstavlja identifikaciju mreže (Net ID) dok preostalih 20 bitova označava poslužitelja (Host ID) [15]. Budući da IP adrese na Internetu moraju biti jedinstvene CIDR blokove dodjeljuje IANA (*Internet Assigned Numbers Authority*) odnosno njena regionalna tijela (RIR - *Regional Internet Registries).* Sa sjedištem u Amsterdamu to je RIPE NCC (*Réseaux IP Européens Network Coordination Centre*) koji je zadužen za područje Europe, Bliskog Istoka i Središnje Azije. Ostala regionalna tijela i njihova područja djelovanja vidljiva su na slici 4 [16].

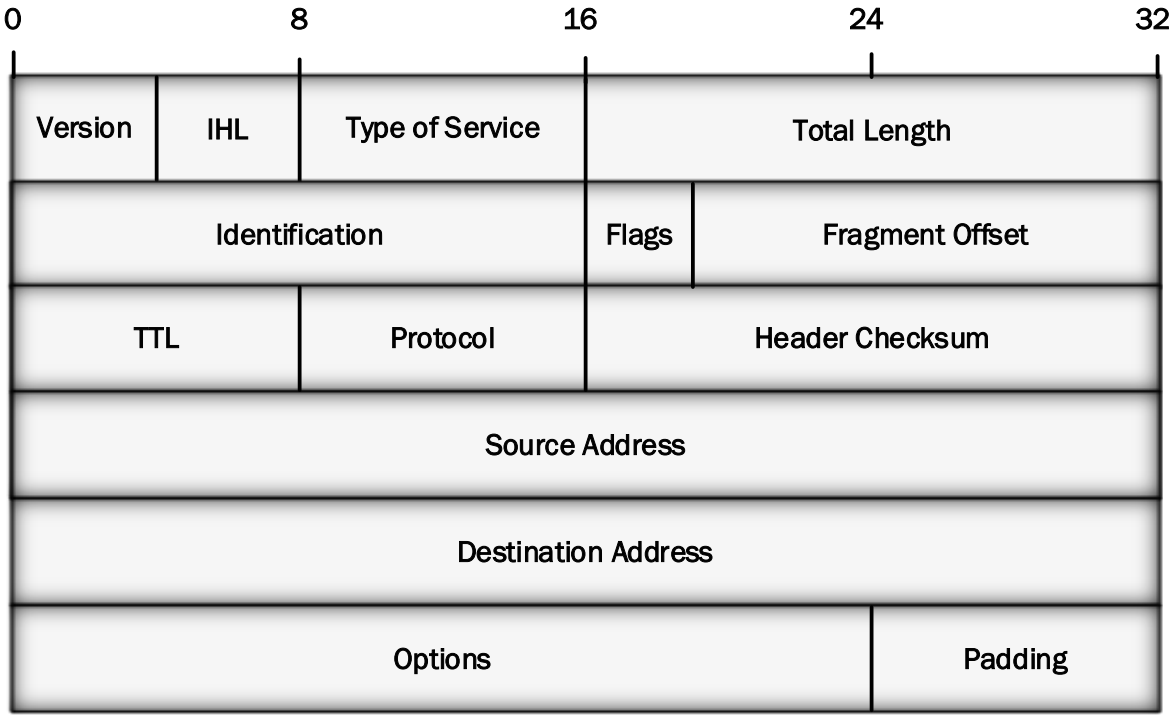
Slika 4. Geografski raspored pet RIR tijela [16]

Prema [13] i [17], zaglavlje IPv4 paketa (slika 5) sastoji se od:

- *Version* (4 bita) – označava verziju IP protokola, budući da se radi o zaglavlju IPv4 protokola ova vrijednost uvijek će biti 4 .
- IHL (4 bita) - *Internet Header Length* označava broj 32 bitnih riječi u zaglavlju, budući da IPv4 zaglavlje može biti varijabilne veličine potrebno je imati ovu informaciju.
- *Type of Service* (8 bita) – označava zahtijevani QoS, paketi s različitim ToS-om (*Type of Service*) bit će različito usmjeravani mrežom. Ako ToS nije implementiran vrijednost će biti 0.
- *Total Length* (16 bita) – potpuna duljina IP paketa, uključujući zaglavlje i sadržaj paketa.
- *Identification* (16 bita) – sadrži ID kojim se određuje kojemu strujanju (*streamu*) pripadaju fragmentirani datagrami.
- *Flags* (3 bita) – zastavice se koriste za kontrolu ili identifikaciju fragmenata, sastoje se od 3 bita koji označavaju sljedeće:
    - Bit 0 – rezerviran, uvijek mora biti vrijednosti 0.
    - Bit 1 – može sadržavati dvije vrijednosti, 0 = moguće fragmentiranje i 1 = fragmentiranje nije moguće.
    - Bit 2 – također može sadržavati dvije vrijednosti, 0 = radi se zadnjem fragmentu ili 1 = postoji još fragmenata nakon njega.
- *Fragment Offset* (13 bita) – služi za označavanje poredaka fragmenata, prvi fragment uvijek ima vrijednost pomaka 0 dok se naknadni određuju prema vrijednosti MTU-a (*Maximum transmission unit*).
- TTL (8 bita) - *Time To Live* označava vrijeme u sekundama koje datagram može provesti unutar mreže, svaki čvor koji primi datagram mora mu smanjiti vrijednost

barem za 1 bez obzira je li vrijeme procesiranje datagrama bilo i manje od jedne sekunde. Čvor koji postavi vrijednosti TTL-a na 0 obavezan je i odbaciti taj datagram. Maksimalna vrijednost koju TTL može imati iznosi 255 (sekundi).

- *Protocol* (8 bita) – vrijednost koja specificira protokol koji će biti korišten na višem sloju, vrijednosti za TCP su 06 u dekadskom obliku ili 0x06 u heksadekadskom i 17 za UDP za dekadskom odnosno 0x11 u heksadekadskom.
- *Header Checksum* (16 bita) – kontrolna suma zaglavlja osigurava da se bitovima nije ništa dogodilo, budući da se vrijednosti kao TTL mijenjaju potrebna je nanovo računati vrijednost na svakome čvoru. Ako vrijednosti kontrolne sume ne poklapaju na odredištu datagram će se odbaciti.
- *Source Address* (32 bita) – logička (IP) adresa pošiljatelja
- *Destination Address* (32 bita) - logička (IP) adresa primatelja
- *Options* (varijabilne duljine) - opcionalno polje koje je moguće koristi za implementaciju dodatnih parametara, ako ga se koristi svi poslužitelji i mrežni uređaji moraju podržavati implementaciju.
- *Padding* (varijabilne duljine) – koristi se samo kada se koristi polje *Options* kako bi se osiguralo da zaglavlje završava 32 bitnom granicom.



Slika 5. IPv4 zaglavlje [13]

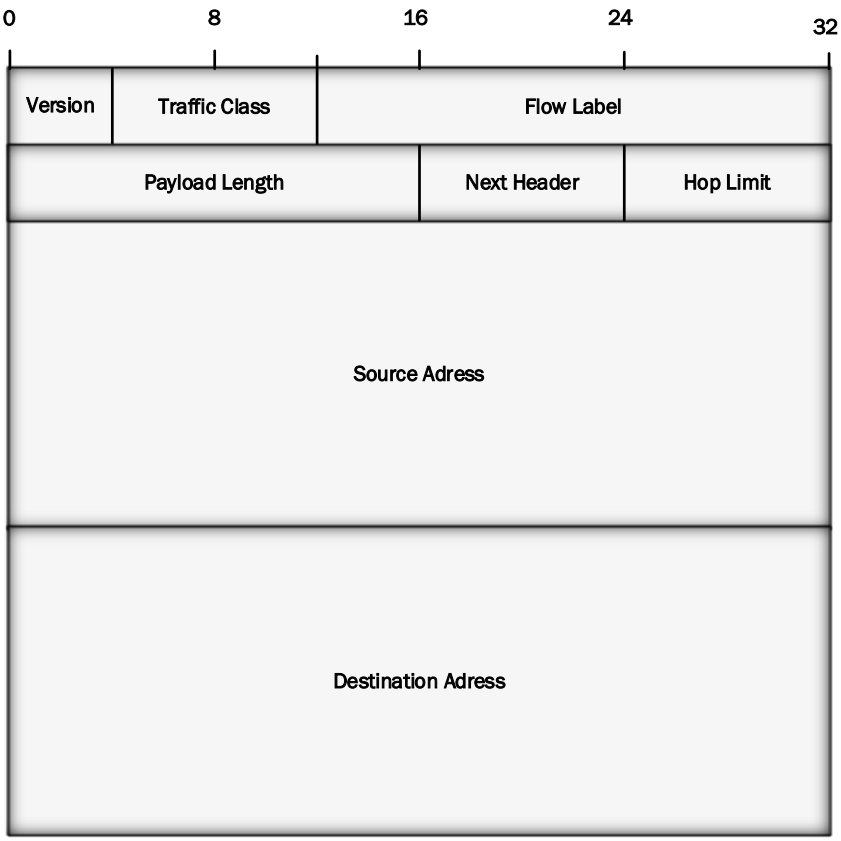
2.3.2.1.2 Internet Protocol Version 6

Iako je IPv4 imao velik broj ažuriranja i nadogradnji u obliku NAT-a (*Network address translation*), IPsec-a (*Internet Protocol Security*), MPLS-a (*Multiprotocol Label Switching), Tunellinga*, kako bi mu se povećao životni vijek i dalje je ostao ograničen u određenim mogućnostima koje IPv6 rješava. Adresni prostor IPv6 protokola iznosi 128 bitova naspram

32 bita kod IPv4, što znači mogućnost adresiranja 2^{128} različitih adresa u odnosu na 2^{32} kod IPv4 [18]. Sama adresa je u heksadekadskom zapisu. Prema izvoru [19] zaglavlje je kod IPv6 (slika 6) kraće i iznosi 40 bajta, te se sastoji od:

- *Version* (4 bita) – isto kao kod IPv4 označava verziju protokola, u ovome slučaju vrijednost iznosi 6.
- *Traffic class* (8 bita) – omogućava aplikacijama da odrede prioritet prometa koji će generirati, odnosno određuje klase usluga.
- *Flow label* (20 bita) – označava tok (*flow*) kojemu paket pripada, označava na koji će se način rukovati paketima na putu od izvorišta do odredišta od strane usmjerivača na ruti. Svi paketi koji pripadaju istome toku moraju biti poslani s istom izvorišnom adresom, istom odredišnom adresom te istom vrijednosti *flow labela*. Ako paket ne pripada niti jednome toku vrijednost *flow labela* iznosi 0.
- *Payload length* (16 bita) – oznaka duljine paketa u bajtovima ne uključujući zaglavlje.
- *Next header* (8 bita) – označava sljedeće zaglavlje koje dolazi, može se raditi o IP opcionalnom zaglavlju ili zaglavlju protokola višega sloja. Ovo polje označava i prisutnost *extension headera*.
- *Hop limit* (8 bita) – ima sličnu funkciju kao TTL polje kod IPv4, s razlikom da se ovdje mjeri u skokovima umjesto u sekundama.
- *Source address* (128 bita) – logička (IP) adresa pošiljatelja.
- *Destination address* (128 bita) – logička (IP) adresa primatelja.

IPv6 paketi sadržavaju osnovno zaglavlje koje u većini slučajeva predstavlja i jedino zaglavlje potrebno za dostavu paketa. Ako je potrebno *extension header* se koristi za dodatne informacije, ovo polje zamjenjuje funkcionalnosti *Options* polja kod IPv4 [19]. Ostala važna poboljšanja kod IPv6 donijela su polja *Flow label* i *Traffic class*. Kod IPv6 se obavlja klasifikacija prometa po zahtijevanim parametrima, budući da različite aplikacije imaju različite zahtjeve za parametrima kao što su *jitter*, propusnost, kašnjenje. Eliminacija NAT-a s obzirom da IPv6 posjeduje dovoljan adresni prostor te nema potrebe za privatnim adresama iza NAT-a i njihovim prevođenjem u javne. Poboljšana sigurnost, IPv6 pruža cjelovitu podršku za mjere provjere autentičnosti, privatnosti i integriteta podataka, zahtijevajući da sve implementacije podržavaju navedene značajke [18].



Slika 6. IPv6 zaglavlje [19]

2.3.2.2 Internet Control Message Protocol (ICMP)

Budući da IP protokol ne posjeduje mehanizme ispravljanja i obavještavanja o pogreškama te mehanizme za komunikaciju između izvorišnog i odredišnog poslužitelja, navedene funkcije obavlja ICMP. On koristi podršku IP protokola kao da se radi o protokolu višega sloja iako je zapravo sastavni dio IP-a i mora biti implementiran u svaki IP modul [20]. Postoje dvije kategorije ICMP poruka, one koje izvještavaju o greškama kao što su nemogućnost dohvaćanja odredišnog poslužitelja, premašeno vrijeme, i poruke kojima se šalju upiti uređajima u mreži [21]. Kako bi se izbjegla beskonačna regresija poruka o porukama, ICMP poruke ne šalju informacije (poruke) o drugim ICMP porukama. Također ICMP poruke se šalju samo o pogreškama pri rukovanju isključivo nultim fragmentom ako se radi o fragmentiranoj poruci [20].

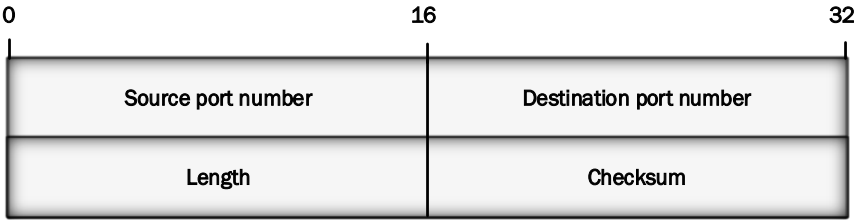
2.3.3 Transportni sloj

Transportni sloj služi za komunikaciju s kraja-na-kraj te logičke veze između aplikacija na različitim poslužiteljima. Gledano prema OSI modelu primarno obuhvaća transportni sloj toga modela ali i neke elemente sloja sesije. Primjer toga bi bio duže uspostavljena TCP konekcija koja bi poprimila osobine sesije [3]. Glavna funkcija transportnog sloja je omogućavanje komunikacije između dva mrežna mjesta. Dva najkorištenija protokola ovoga sloja su beskonekcijski UDP i konekcijski orijentirani TCP.

2.3.3.1 User Datagram Protocol (UDP)

UDP je beskonekcijski protokol transportnoga sloja, što znači da je svaki datagram neovisan o ostalima. Ne pruža nikakve mehanizme ispravljanja pogrešaka, očuvanja redoslijeda i dupliciranja paketa, kontrole toka te kontrole zagušenja. Postoji mogućnost detekcije grešaka i posjeduje mogućnost provjere kontrolne sume (*checksuma*). Ovakvo minimalno korištenje mehanizama od strane ovoga protokola ostavlja više mjesta samim aplikacijama za kontrolu paketa. Glavna prednost UDP-a naspram TCP-a je njegov mali *overhead* koji omogućava veću brzinu što znači da će ga koristiti aplikacije kojima ranije navedeni mehanizmi nisu potrebni dok je sama brzina obrade bitna. Ovdje se mogu uključiti i aplikacije koje imaju već ugrađene potrebne mehanizme te im oni nisu potrebni na razini protokola. Zaglavlje UDP protokola (slika 7) dolazi nakon IP zaglavlje i duljine je 8 bajta te se prema izvorima [2] i [12] sastoji se od sljedeća četiri elementa:

- *Source port number* (16 bita) – predstavlja broj porta koji koristi proces na izvorišnome poslužitelju. Izvorišni port nije obavezan te njegova vrijednost može biti 0 ako pošiljatelj datagrama ne zahtijeva odgovor.
- *Destination port number* (16 bita) – predstavlja broj porta koji koristi proces na odredišnome poslužitelju.
- *Length* (16 bita) – definira ukupnu duljinu datagrama, zaglavlje i sadržaj. Ovo je isto opcionalno polje budući da je datagram enkapsuliran unutar IP paketa koji posjeduje polja IHL i *Total Length* čijim oduzimanjem se dobiva duljina UDP datagrama ili kombinacija *Payload Lengtha* i fiksne duljine zaglavlja od 40 bajta kod IPv6.
- *Checksum* (16 bita) – služi za detekciju pogrešaka kod cijeloga datagrama uključujući zaglavlje i sadržaj.

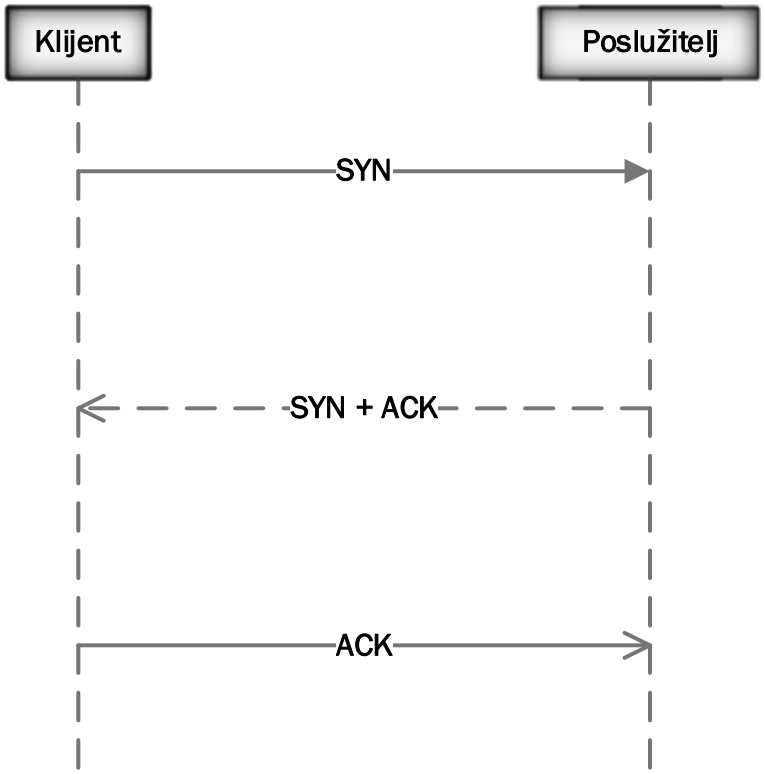


Slika 7. UDP zaglavlje [2] [12]

2.3.3.2 Transmission Control Protocol (TCP)

Prije nego započne razmjena podataka kod TCP-a prvo je potrebno uspostaviti logičku vezu između dva kraja. To se radi korištenjem *3-way handshake* mehanizma koji je vidljiv na slici 8. Kada klijent želi uspostaviti vezu s poslužiteljem šalje prvi segment prema poslužitelju, taj segment u sebi sadrži samo SYN zastavicu te služi za sinkronizaciju rednih brojeva. Klijent odabire slučajni broj te ga postavlja kao prvi redni broj i šalje prema poslužitelju, taj se broj naziva ISN (*initial sequence number*). SYN je kontrolni segment te on ne sadržava nikakve podatke, ali koristi jedan redni broj te nakon što prijenos podataka počne ISN se povećava za 1. Nakon što je poslužitelj primio SYN segment šalje odgovor odnosno drugi segment koji sadrži zastavice SYN i ACK te ima dvostruku funkciju. Prva je SYN segment od strane poslužitelja za komunikaciju prema drugoj strani odnosno klijentu. Poslužitelj ga koristi kako

bi inicijalizirao broj za numeriranje bajtova poslanih od strane poslužitelja ka klijentu. Druga funkcija postavljanje ACK zastavice kojom se potvrđuje da SYN segment poslan od strane klijenta zaprimljen. Nakon toga klijent šalje svoj drugi segment, ACK segment. Njime potvrđuje da je primio segment od strane poslužitelja koji je sadržavao ACK zastavicu [2]. Nakon što je veza uspostavljena za svaki segment koji prenosi podatke u jednome smjeru šalje se jedan ACK u suprotnom smjeru. Koristeći redne brojeve primatelj odbacuje duple segmente te preslaguje segmente koji su došli krivim redoslijedom [12].

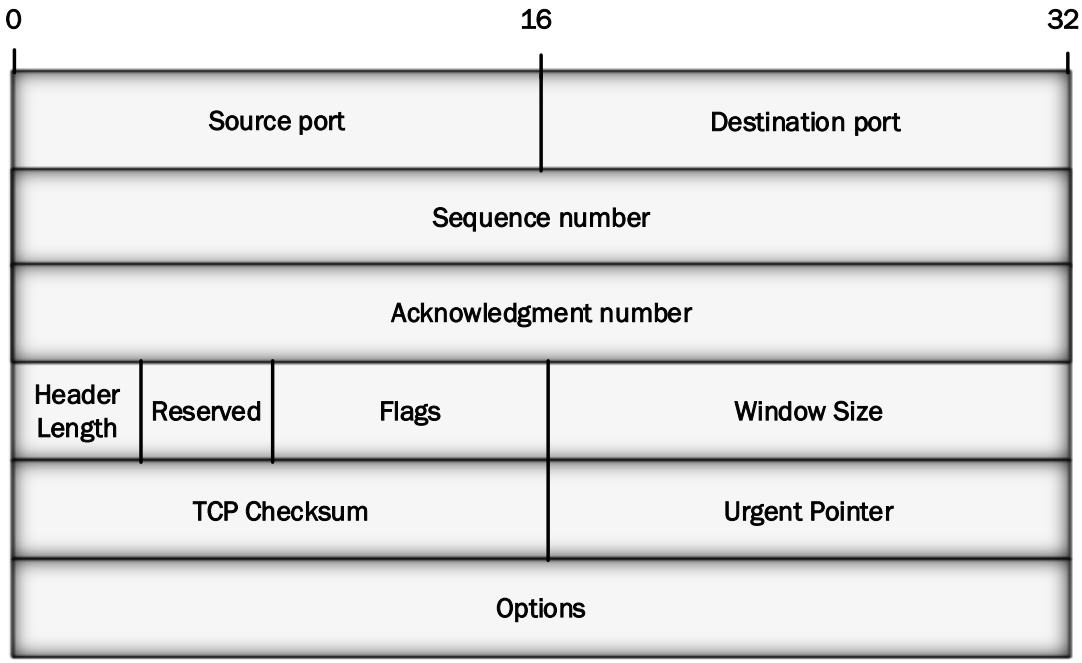


Slika 8. Uspostavljanje veze korištenjem *3-way handshake* mehanizma [12].

Format TCP zaglavlja vidljiv je na slici 9. Duljine je 20 bajtova ako se ne koristi polje *Options*, te do 60 bajtova ako se koristi. Prema izvorima [12] i [2] sadrži sljedeće elemente:

- *Soruce port* (16 bita) – izvorišni port zajedno sa izvorišnom IP adresom čini *socket* jedinstveno identificirajući svaku TCP vezu.
- *Destination port* (16 bita) – odredišni port zajedno sa odredišnom IP adresom čini *socket* jedinstveno identificirajući svaku TCP vezu.
- *Sequence number* (32 bita) – ovo polje služi za identifikaciju određenoga bajta unutar strujanja podataka pridjeljujući svakome bajtu unutar strujanja redni broj. Ti redni brojevi su 32 bitni nepotpisani (*unsigned*) brojevi čije numeriranje kreće opet od nule nakon što se dosegne vrijednost (2^{32})-1.

- *Acknowledgment number* (32 bita) – budući da je svaki bajt numeriran rednim brojem ovo polje sadrži vrijednost sljedećeg rednoga broja koji se očekuje, vrijednost ovog polja jednaka je vrijednosti rednoga broja zadnjeg uspješno dostavljenog bita + 1 . ACK zastavica treba biti aktivna kako bi ovo polje bilo valjano.
- *Header length* (4 bita) – predstavlja duljinu samog zaglavlja, bitno je zato što je polje *Options* varijabilne duljine što i samu duljinu zaglavlja čini varijabilnome.
- *Reserved* (4 bita) – rezervirano za buduću uporabu.
- *Flags* (8 bita) – definira 8 različitih kontrolnih bitova ili zastavica, može biti aktivna jedna ili više zastavica u isto vrijeme. Postoje sljedeće zastavice te su postavljene sljedećim redoslijedom:
    - CWR – Smanjen *Congestion Window* (pošiljatelj je smanjio brzinu slanja).
    - ECE – ECN *Echo* (pošiljatelj je primio raniju obavijest o zagušenju).
    - URG - *Urgent pointer* je valjan.
    - ACK – *Acknowledgment* (*Acknowledgment Number* polje je valjano – uvijek aktivno nakon uspostavljanja veze).
    - PSH – *Push* (primatelj bi trebao podatke proslijediti aplikaciji čim prije moguće - nije pouzdano implementirano ili korišteno).
    - RST – Resetiraj vezu (veza se prekida, obično zbog pogreške).
    - SYN – Sinkroniziraj redne brojeve.
    - FIN – Prekini vezu.
- *Window size* (16 bita) – broj bajtova počevši s onim specificiranim od strane ACK broja, koji je primatelj spreman prihvatiti. Budući da se radi o 16 bitnom polju maksimalna veličina prozora iznosi 65 535 bajtova. Ova vrijednost se većinom naziva *receiving window* (rwnd) i određuje je primatelj.
- *Checksum* (16 bita) – isto kao kod UDP-a, služi za detekciju pogrešaka kod cijeloga datagrama uključujući zaglavlje i sadržaj. Uz jednu iznimku kod TCP-a je ovo polje obavezno.
- *Uregent pointer* (16 bita) – vrijedi samo ako je URG zastavica aktivna, Definira vrijednost koja se mora dodati rednome broju kako bi se dobio broj zadnjeg hitnog bajta u podatkovnome dijelu segmenta.
- *Options* (varijabilne duljine) – mogućnost zapisa do 40 bajtova dodatnih podatka, ovo polje je neobavezno.



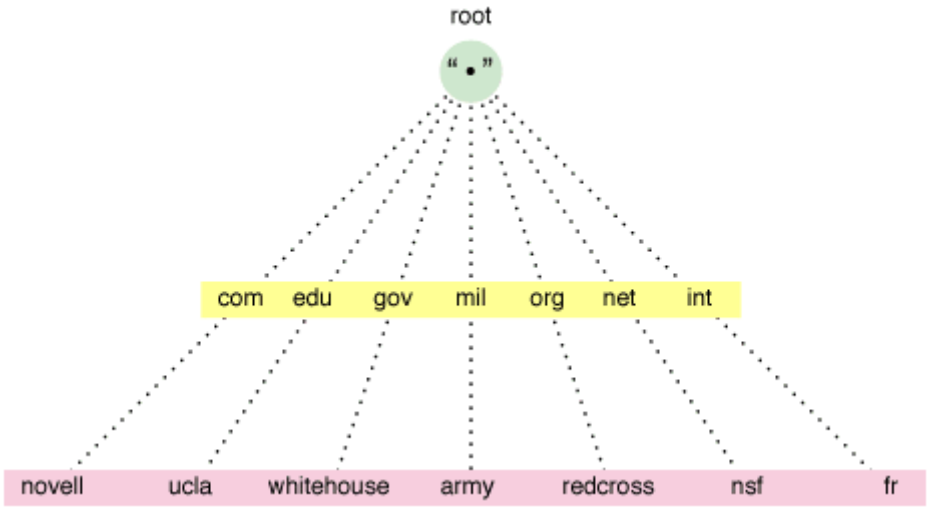
Slika 9. TCP zaglavlje [2] [12]

2.3.4 Aplikacijski sloj

Aplikacijski sloj je najbliži krajnjemu korisnik te omogućava krajnjem korisniku odnosno aplikacijama koje koristi slanje te primanje podataka. Aplikacijski sloj se ponaša kao sučelje između aplikacija i mrežnog dijela ispod. Kod TCP/IP složaja aplikacijski sloj odrađuje funkcije tri gornja sloja OSI modela, aplikacijskog, prezentacijskog i sesijskog [22]. Neki od bitnijih protokola na ovome sloju su: DHCP (*Dynamic Host Configuration Protocol*), DNS, HTTP, SMTP.

2.3.4.1 DNS

Kako su ljudima riječi razumljivije od brojeva kada se nešto označava, IP adrese poslužitelja potrebno je prikazati u tekstualnome obliku. DNS se koristi za prevođenje imena domena u IP adresu. Budući da aplikacija treba IP adresu kako bi otvorila TCP vezu ili poslala UDP datagram potrebno je dobiti IP adresu, a to se radi na sljedeći način: zahtjev za IP adresom određene domene prvo se provjerava unutar predmemorije (*cachea*) na računalu. Ako na računalu nije dostupna adresa šalje se upit prema *resolveru* koji većinom kontrolira ISP. Ako niti tamo nije dostupna traži se kroz DNS hijerarhiju koja je vidljiva na slici 10.



Slika 10. DNS hijerarhija [23]

DNS za distribuciju koristi hijerarhiju koja se još naziva i *domain name space*. DNS stablo (slika 10) ima jednu domenu na vrhu koja se naziva *root* domenom te se označava točkom. Ispod nje se nalaze domene najviše razine. Domene ispod tih domena označavaju određene organizacije ili entitete te i one mogu još biti podijeljene u poddomene [23].

2.3.4.2 DHCP

DHCP je klijent-server protokol koji pruža konfiguracijske parametre Internet poslužiteljima. Sastoji se od dvije komponente, protokola koji služi za prijenos konfiguracijskih parametara od strane DHCP poslužitelja prema poslužitelju i mehanizma za alokaciju adresa poslužiteljima [24]. Postoji osam vrsta poruka kod DHCP-a:

- DHCPDISCOVER – *broadcast* od strane klijenta kako bi našao dostupne DHCP poslužitelje
- DHCPOFFER – odgovor poslužitelja na DHCPDISCOVER poruku te ponuda IP adrese i ostalih parametara
- DHCPREQUEST – poruka od klijenta prema poslužitelju koja obavlja jednu od sljedećih funkcija:
    - Zahtjev za parametrima ponuđenim od strane jednoga poslužitelja i odbijanje ostalih ponuda
    - Provjera prethodno alociranih adresa nakon promjene sustava ili mreže
    - Zahtjev za produljenjem najma određene adrese
- DHCPACK – potvrda od strane poslužitelja klijentu uključujući IP adresu
- DHCPNACK – negativna potvrda od poslužitelja klijentu naznačujući da je najam klijenta istekao ili da je zatražena IP adresa netočna
- DHCPDECLINE – poruka od klijenta prema poslužitelju naznačujući da se ponuđena adresa već koristi
- DHCPRELEASE - poruka od klijenta prema poslužitelju kojom se otkazuje ostatak najma i odustaje od tre mrežne adrese

- DHCPINFORM – poruka od klijenta da već posjeduje IP adresu te traži ostale konfiguracijske parametre od DHCP poslužitelja

Način na koji DHCP radi je da klijent šalje DHCPDISCOVER *broadcast* poruku na svojoj lokalnoj fizičkoj podmreži te takva poruka još može sadržavati neke dodatne mogućnosti kao što su prijedlog IP adrese ili duljinu najma. Svaki poslužitelj može poslati odgovor u obliku DHCPOFFER poruke koja sadržava dostupne adrese i ostale konfiguracijske parametre. Isto tako svaki poslužitelj će ubilježiti koju je adresu ponudio klijentu kako ne bi došlo do nuđenja iste adrese ostalim klijentima. Ako klijent primi jednu ili više ponuda od strane poslužitelja bira jednu na osnovu konfiguracijskih parametara. Nakon toga opet šalje *broadcast* poruku ali ovaj puta DHCPREQUEST koja sadrži identifikator poslužitelja kako bi se označilo čiju je ponudu odabrao. U slučaju da nije bilo niti jedne ponude klijent može koristi prethodnu adresu ako ima saznanja te je najam i dalje valjan [25].

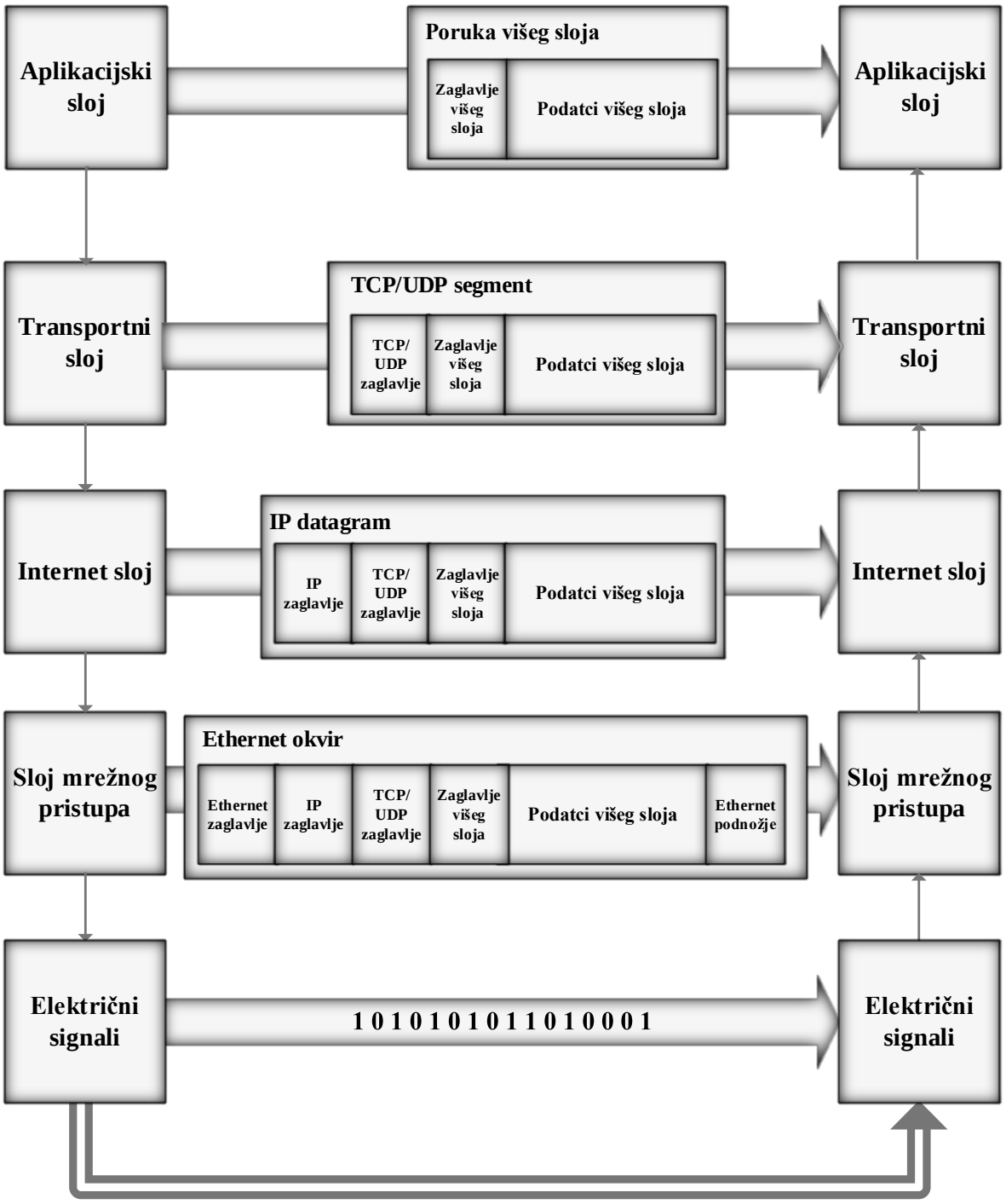
2.3.4.3 HTTP

HTTP je protokol aplikacijskog sloja koji omogućava dohvaćanje resursa. Radi se o klijent-poslužitelj protokolu što znači da su zahtjevi upućeni od strane primatelja, većinom Internet preglednika. Kompletni web dokument se sastoji od svih elemenata koji su dohvaćeni npr. tekst, video, slike, skripte itd. [26] Beskonekcijski je protokol što znači da svaki puta kada se dogodi komunikacija između klijenta i poslužitelja prekida se veza. Klijent i poslužitelj imaju znanje o drugoj strani isključivo tijekom razmjene podataka, što znači da se radi i o protokolu bez stanja (*stateless protocol*). HTTP je neovisan o mediju što znači da može poslati bilo koju vrstu podataka sve dok klijent i poslužitelj znaju rukovati tim tipom podatka [27].

2.3.5 Enkapsulacija paketa

Proces enkapsulacije i dekapsulacije unutar TCP/IP složaja prikazan je na slici 11. Prvo se podatci koje je potrebno prenijeti mrežom enkapsuliraju unutar korištenog protokola aplikacijskog sloja. Nakon toga se na transportnom sloju poruci višeg sloja dodaje zaglavlje protokola transportnog sloja, najčešće TCP ili UDP protokola što čini segment. Dolaskom na Internet sloj segmentu se dodaje zaglavlje IP protokola te ti podatci čine datagram. Ako je poruka prevelika tada se obavlja fragmentacija, te se poruka dijeli u više datagrama gdje svaki sadrži određeni dio. Takva poruka mora se ponovno sastaviti na odredištu. Datagram se opet spušta na nižu razinu, u ovom slučaju sloj podatkovne veze. Na tome sloju datagramu se dodaje okvir protokola korištenog na tome sloju. U ovome slučaju na slici 11 prikazano je enkapsuliranje unutar Ethernet protokola, ali IP datagram se može enkapsulirati i u mnoge druge protokole sloja podatkovne veze.

Nakon što je dobiven Ethernet okvir podatci su spremni za slanje prijenosnim medijem. Nakon dolaska na odredište obavlja se raspakiravanje odnosno dekapsulacija paketa. To se radi tako da svaki sloja uzima svoje zaglavlje čita podatke te prosljeđuje ostatak sve dok se ne dođe do aplikacijskog sloja, na kojemu se aplikacija uzima podatke koji su joj namijenjeni [3].

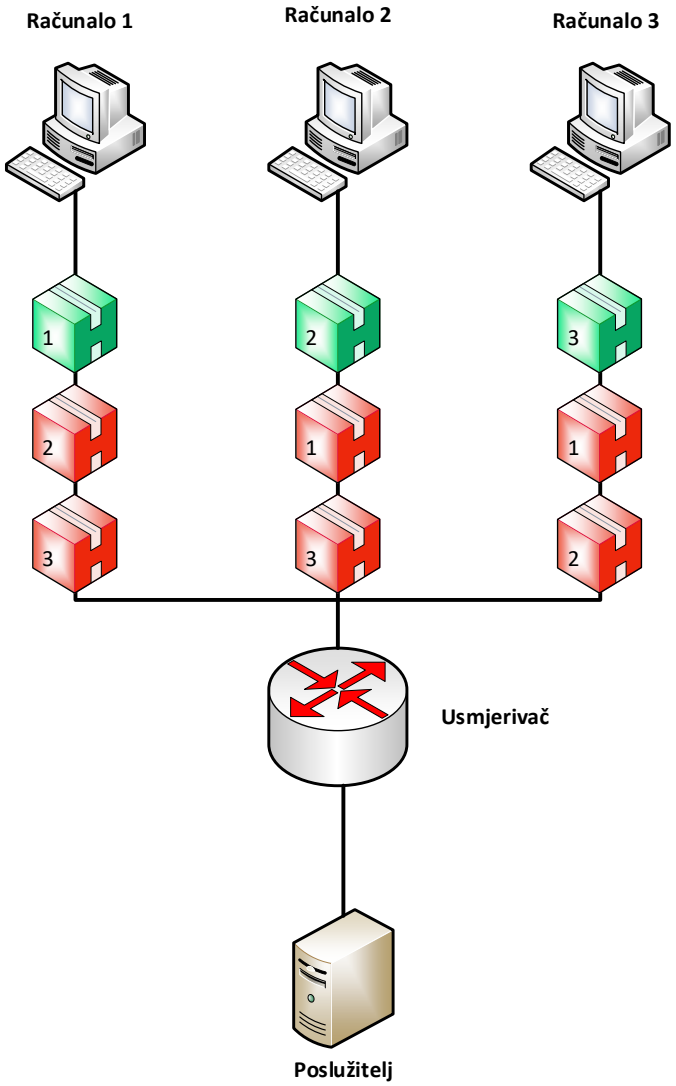


Slika 11. Proces enkapsulacije podataka [3]

Ovakav slojevit način rada gdje se informacije čitaju sloj po sloj koriste i aplikacije za analizu mrežnog prometa. One čitaju sadržaj na isti način kao i računalo, ali ga prezentiraju korisniku aplikacije u razumljivijem obliku od samog skupa nula i jedinica. Funkcija takvih aplikacija se višestruka, omogućavaju korisniku uvid u podatke koje čita računalo (ili neki drugi uređaj), kako bi provjerio sadržaj prometa i ustanovio postoji li kakvih anomalija ili nepoželjan sadržaja. Budući da paketi sadrže informacije o svojoj duljini moguće je i mjeriti promet ovim alatima, iako se rijetko rabe za tu funkciju. Dobivenim informacijama moguće je i kreirati zapise o prometu, itd.

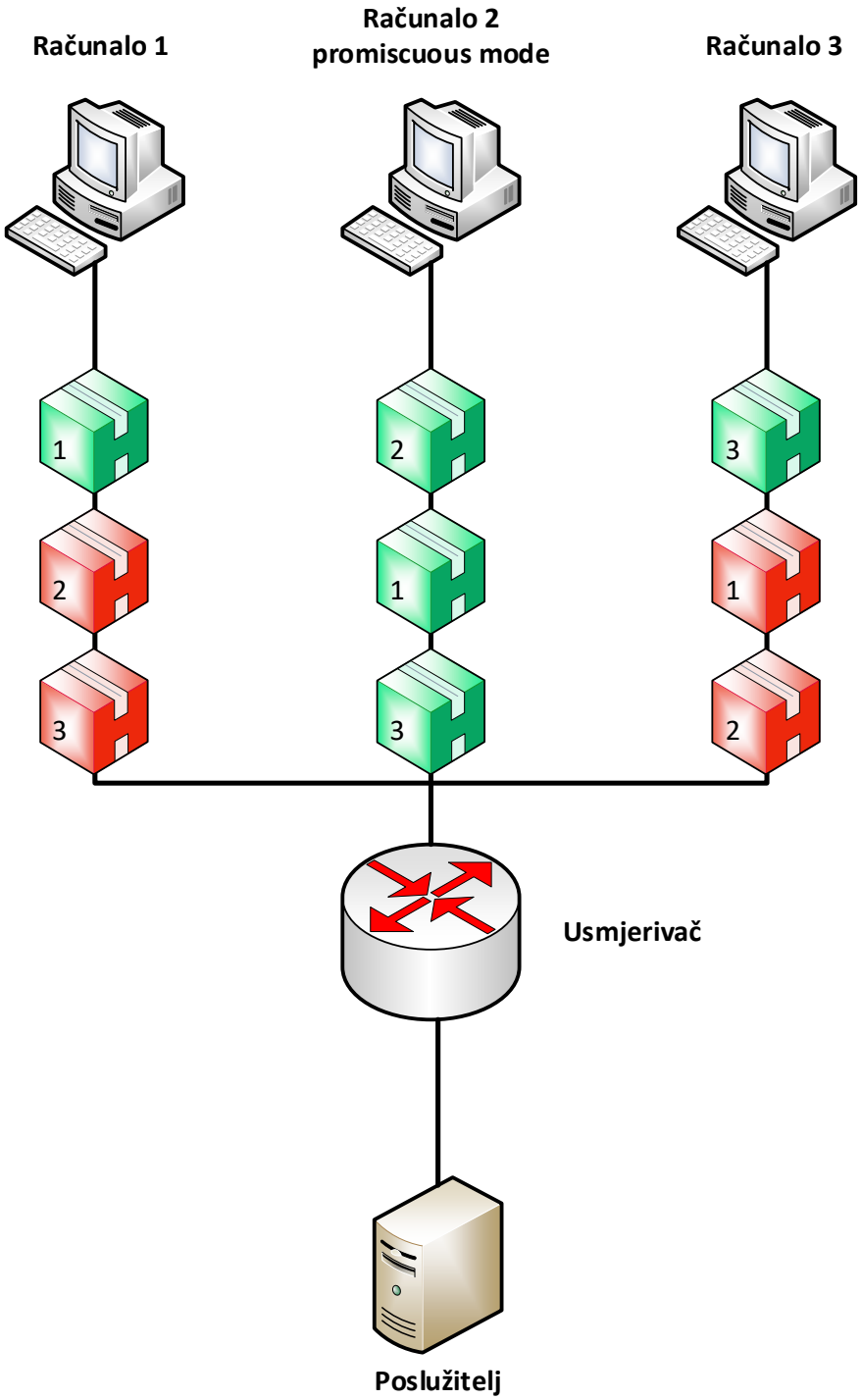
3. Funkcionalnosti aplikacija za analizu mrežnog prometa

Aplikacije za analizu mrežnog prometa popularno zvani *packet analyzer* ili *packet sniffer* su računalni programi, iako postoje i fizički uređaji za tu svrhu. Služe za presretanje i bilježenje prometa koji se kreće mrežom ili dijelom mreže. Funkcioniraju tako da iz toka podataka koji putuje mrežom „hvataju“ pakete. Oni se nakon toga dekodiraju kako bi se došlo do vrijednosti koja se nalaze u poljima paketa te se ih se analizira i prikazuje prema specifikacijama (RFC ili druge) [28]. Mrežna kartica računala prima više paketa s mreže od onih njoj namijenjenih, ali ignorira takve pakete. Takvo filtriranje se izvodi prema MAC adresi, kartica uspoređuje svoju MAC adresi s onom dolaznog paketa. Kako bi prikupile pakete namijenjene drugim uređajima u mreži, ovakve aplikacije iskorištavaju taj mehanizam. Budući da će aplikacija prikupiti samo one pakete koje mrežna kartica prihvati uobičajeno prikupljanje paketa bi izgledalo kao što je prikazano na slici 12. Aplikacija vidi samo primljene pakete označene zelenom dok se crveni koji nisu namijenjeni tome računalu odbacuju od strane mrežne kartice te ih aplikacija ne vidi [29] [30].



Slika 12. Normalan način rada mrežne kartice

Budući da je pri analizi prometa bitno vidjeti sav ili barem većinu prometa na mreži koristi se *promiscuous mode*. Ovakav način rada mrežne kartice omogućava prihvat svih paketa koji dolaze do mrežne kartice, odnosno ignoriranje MAC adrese dolaznih paketa i puštanje i onih čija je MAC adresa različita od one mrežne kartice (slika 13). Kada se paketi šalju od jednog čvora do drugog prolaze kroz mnoštvo međučvorova. Čvor čija je mrežna kartica postavljena u *promiscuous mode* može primiti pakete. Paketi koji dođu do mrežne kartice budu kopirani u memoriju te predani *kernel bufferu* od kuda će biti korišteni od strane aplikacije [30].



Slika 13. *Promiscuous mode*

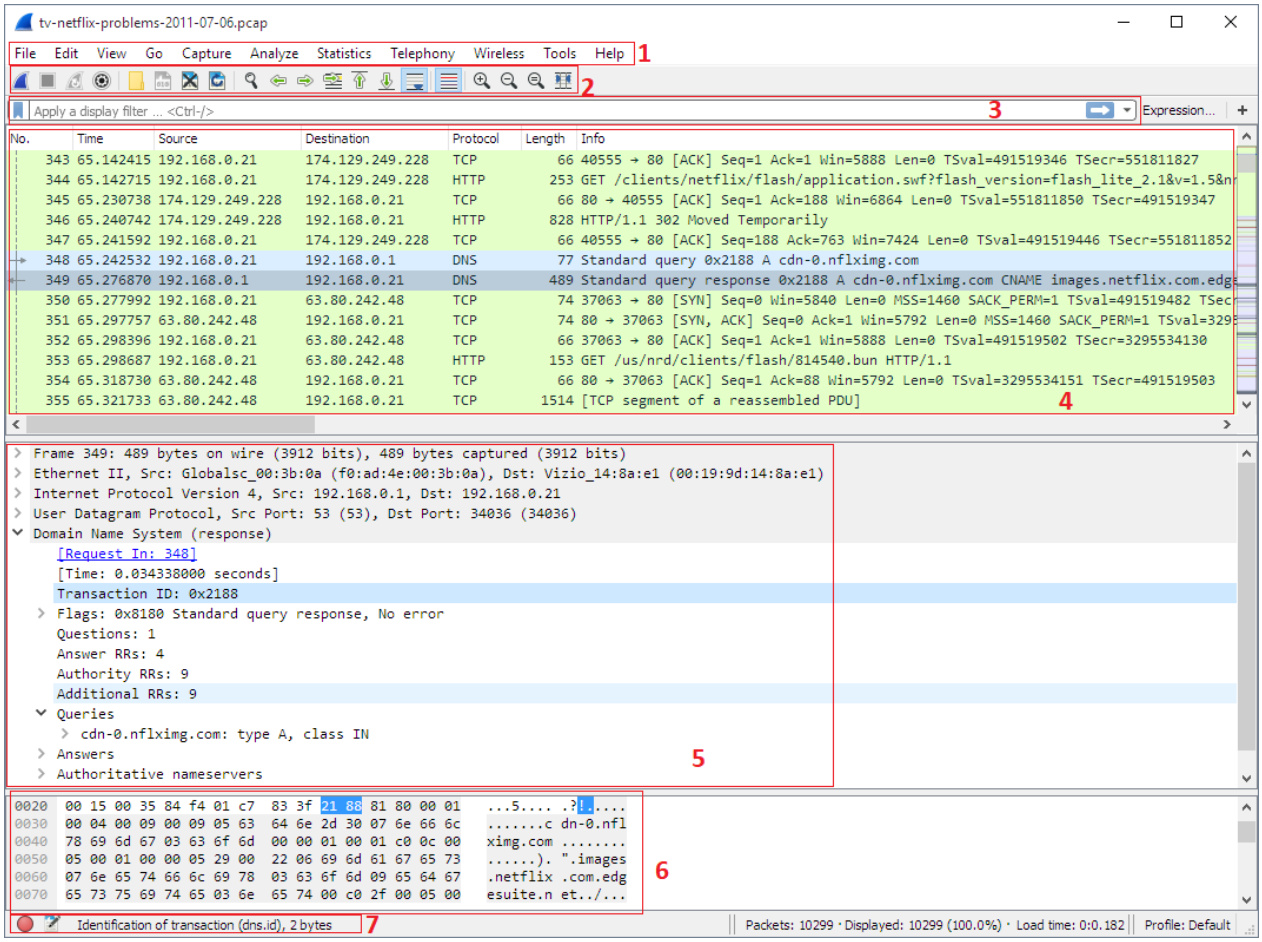
3.1 Wireshark

Wireshark je aplikacija za analizu mrežnog prometa, služi za detaljnu prezentaciju prikupljenih paketa. Radi se o besplatnom alatu otvorenog koda koji je donio veliku promjenu na to tržište budući da su alati prije njega bili izrazito skupi. Ethereal odnosno preteču Wiresharka razvio je Gerald Combs 1997. godine zbog vlastite potrebe za alatom kojim bi mogao pratiti mrežni promet te željom da nauči više o upravljanju i administriranju mreža. Izdan je pod GNU GPL licencom te se sav izvorni kod može besplatno preuzeti. Većina programa implementirana je u C programskom jeziku (C99) uz iznimku grafičkog sučelja za koje je korišten C++, ostali korišteni programski jezici su CMake, Python, PowerShell, Bash, Perl, Lua. Koristi pcap biblioteku koja je zapravo API koji pruža alatima koji ga koriste mehanizme za prikupljanje i filtriranje mrežnih paketa. Same pakete moguće je uživo prikupiti na mreži ili ih učitati iz pcap datoteke [31] [32]. Wireshark se koristi za rješavanje mrežnih problema, ispitivanje sigurnosnih problema, provjeru mrežnih aplikacija, otklanjanje pogrešaka pri implementaciji protokola, učenju rada mrežnih protokola i slično. Prema [32] glavne funkcionalnosti koje Wireharsk pruža su:

- Prikupljanje paketa s mrežnog sučelja u stvarnom vremenu.
- Čitanje datoteka koje posjeduje zapise o prikupljenim paketima generiranim Wiresharkom ili drugim alatom kompatibilnog formata.
- Uvoz podataka iz tekstualne datoteke koja sadržava *hex dump* podataka paketa.
- Prikaz paketa s izrazito detaljnim informacijama o protokolu.
- Spremanje prikupljanih paketa.
- Izvoz određenih ili svih paketa u velik broj formata.
- Mogućnost filtriranja paketa po različitim kriterijima.

Glavni prozor Wiresharka vidljiv je na slici 14, sastoji se od sljedećih dijelova:

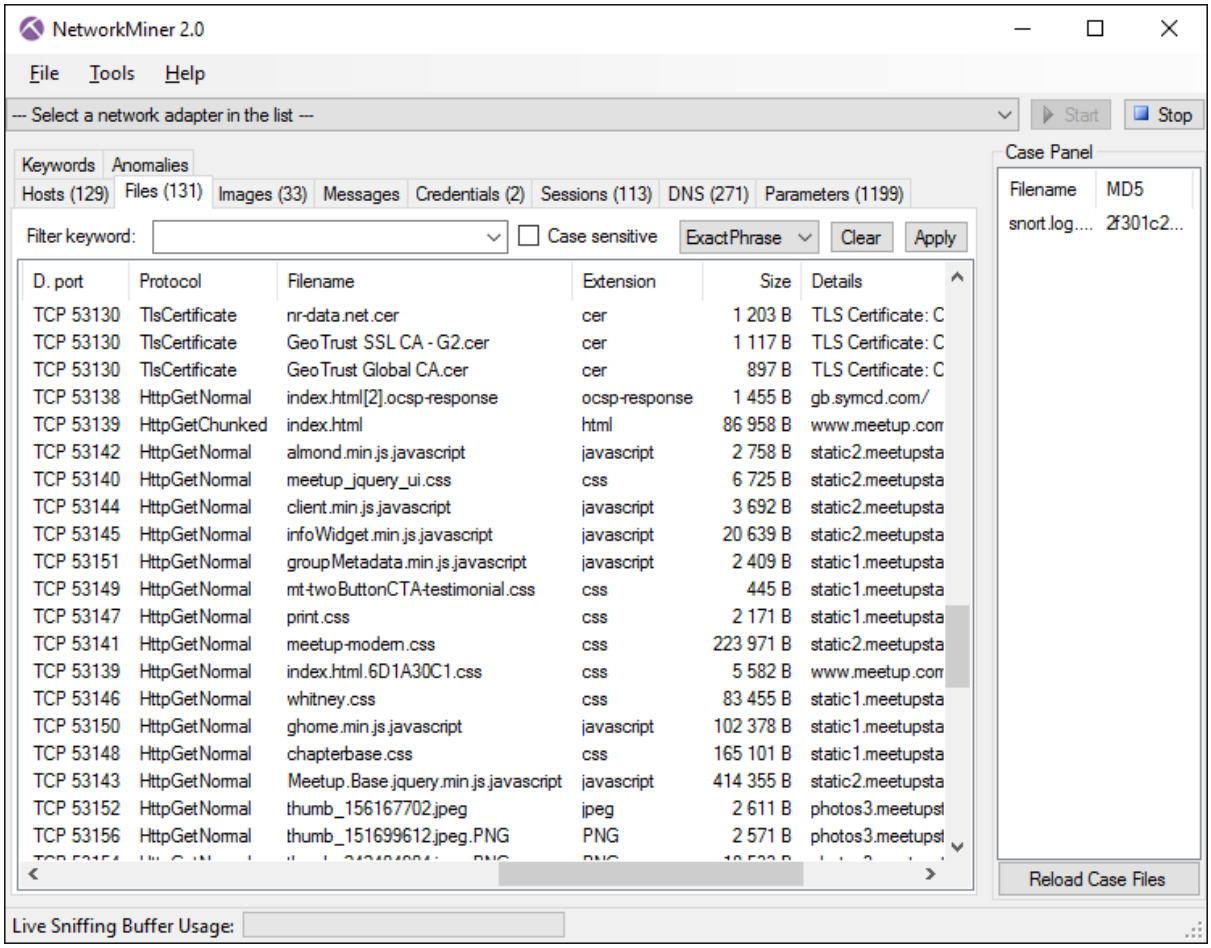
1. Izbornik za pokretanje radnji.
2. Glavna alatna traka koja omogućava brzi pristup često korištenim funkcijama.
3. Alatna traka filtera omogućava postavljanje i prikaz filtera po kojima će paketi biti filtrirani.
4. Popis paketa, prikazuje sažetak informacija o paketima te je klikom na određeni paket moguće vidjeti detaljne informacije u poljima ispod.
5. Prikaz detalja odabranog paketa.
6. Prikaz podataka samog paketa.
7. Statusna traka, prikazuje detaljnije informacije o trenutnom stanju programa i prikupljenih podataka.



Slika 14. Glavni prozor Wiresharka [32]

3.2 NetworkMiner

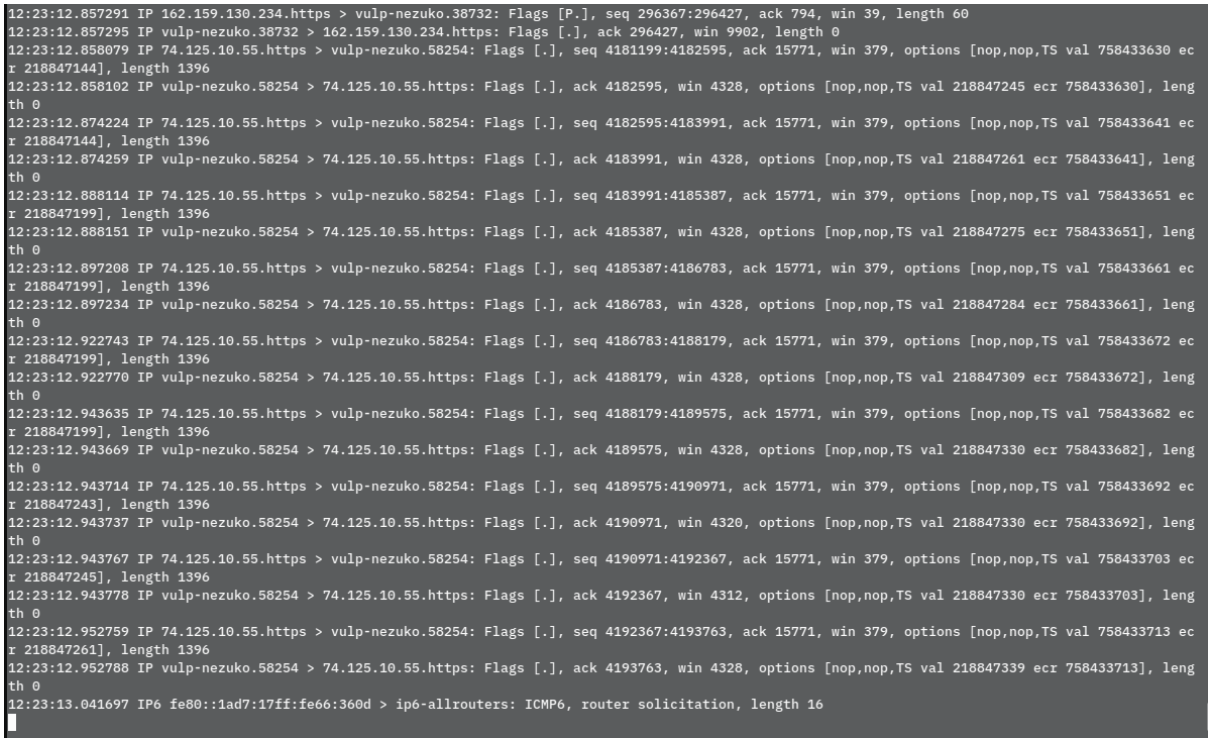
NetworkMiner je alat otvorenog koda koji služi za forenzičku analizu mreže. Radi primarno na Windows operacijskom sustavu, ali može biti korišten i na Linux, Mac OS X i FreeBSD operativnim sustavima. Može ga se koristiti kao pasivni alat kako bi se otkrili operativni sustavi, sesije, imena uređaja na mreži, otvoreni portovi itd. bez potrebe za ikakvim prometom u mreži. NetworkMiner ima mogućnost prikupljanja mrežnog prometa u stvarnom vremenu te uvoz pcap datoteka. Glavna prednost ovoga alata je to što on izvlači veliku količinu korisnih podataka automatski, za razliku od drugih alata gdje je to potrebno ručno raditi, što je izrazito dugotrajan proces. Može prikazati podatke o uređajima na mreži, datoteke, slike, poruke, podatke za akreditaciju (za podržane protokole), sesije, DNS, parametre te je sve te podatke moguće pretraživati koristeći ključne riječi ili niz bajtova (slika 15) [33].



Slika 15. Prikaz glavnog prozora NetworMinera [33]

3.3 Tcpdump

Tcpdump je alat otvorenog koda za analizu paketa koji se izvodi preko komandne linije. Dolazi pred instaliran na većini unix baziranih sustava za koje je i napravljen iako postoji i inačica za Windows operativne sustave naziva WinDump koja korisit WinPcap odnosno Windows verziju libpcapa. Kao i svi ovakvi alati ima mogućnost prikupljanja podataka te prikaz istih uz korištenje ugrađenih filtera (slika 16). Osim prikaza podataka ima i mogućnost spremanja u datoteku što se radi sa -w komandom, te ih se kasnije može pročitati koristeći -r komandu. Glavni nedostatak u odnosu na većinu alata ovog tipa predstavlja nemogućnost čitanja pcap datoteka [34] [35].

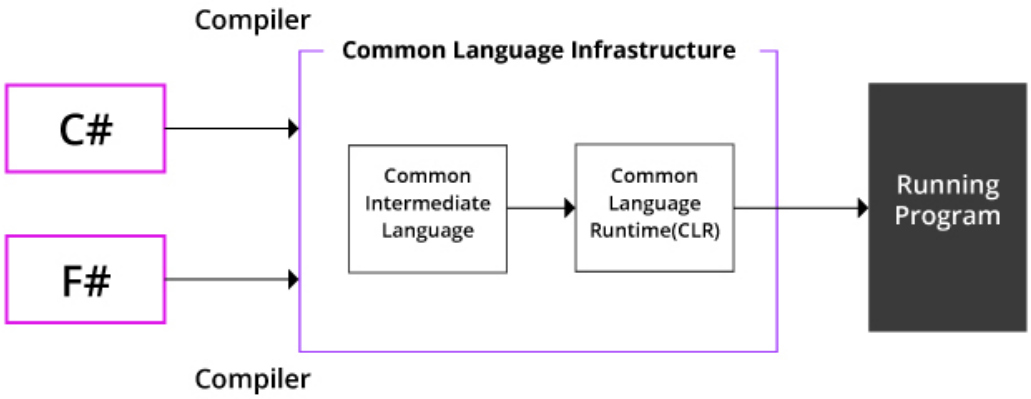


Slika 16. Ispis podataka korištenjem tcpdump alata [35]

Iz svih navedenih alata je vidljivo da je neizostavna stavka svakog od njih filter, kako bi se moglo izabrati koji će promet biti prikazan. Većina alata ima i naprednije funkcije kao što su automatski prikaz sadržaja paketa aplikacijskog sloja, praćenje TCP toka kroz mogućnosti same aplikacije (iako je isto moguće dobiti samim filtriranjem paketa), pa do sastavljanja cijelih datoteka iz prikupljenih paketa kao što je slučaj kod NetworkMInera. Aplikacija izrađena za potrebe ovog rada posjeduje samo osnovne funkcije kao primjerice tcpdump, uz ograničenje prikaza samog IP, UDP, TCP paketa te dodatak grafičkog sučelja i mogućnost čitanja pcap datoteka.

4. Izrada aplikacije za analizu toka paketa

Aplikacija za analizu toka paketa koja je napravljena u sklopu ovog diplomskog rada, u potpunosti je napisana u C# programskom jeziku. Radi se o objektno orijentiranom programskom jeziku opće namjene razvijenom od strane Microsofta te je odobren od strane ECMA (*European Computer Manufacturers Association*) i ISO-a, kao autori jezika se navode Anders Hejlsberg i njegov tim koji su ga razvili tijekom razvoja .Net Frameworka [36]. Microsoft C# opisuje kao moderan, objektno orijentiran te *type safe* (mjera koja obeshrabruje ili sprječava pogreške pri pisanju koda) jezik. Koristi se za izradu aplikacija unutar .NET ekosustava te je sličan C, C++, Java i JavaScript jezicima [37]. C# je dizajniran za CLI (*Common Language Infrastructure*) koji opisuje izvršni kôd i *runtime* okruženje, što omogućava upotrebu više jezika visoke razine na različitim platformama i arhitekturama. CLR (*Common Language Runtime*) je definiran unutar CLI-a. Predstavlja komponentu virtualnog stroja koji upravlja izvršavanjem programa napisanih jezicima koji koriste .NET Framework kao što je npr. C#. Izvorni .NET kôd kompajliran je unutar CLI-a, te se izvršava nakon što se CIL pretvori u izvorni kôd što se radi pomoću JIT (*Just-In-Compiler*) kao što je prikazano na slici 17 [38].



Slika 17. Prikaz CLI infrastrukture [38]

Glavne prednosti C# predstavljaju *Garbage collection* koji automatski vraća memoriju zauzetu od strane nedostupnih i neiskorištenih objekata. Mogućnost definiranja klase unutar druge klase odnosno ugniježđene klase. Postoji manja mogućnost da dođe do greške budući da varijable koje nisu *Boolean* tipa nije moguće koristiti kao uvjete. Nije potrebno deklarirati funkcije i klase te ih je moguće definirati u bilo kojim redoslijedom. Sve varijable automatski se inicijaliziraju na svoju početnu vrijednost prije nego budu korištene. *Exception handling* pruža strukturiran i proširiv pristup otkrivanju pogrešaka te oporavku. *Lambda expressions* podržavaju funkcionalne tehnike programiranja, dok LINQ (*Language Integrated Query*) stvara zajednički obrazac za rad s podatcima iz bilo kojega izvora [37], [38]. Nedostatci s druge

strane su većinom vezani uz .NET Framework za čije je izvršavanje potrebno Windows okruženje, isto tako podrška za starije verzije .NET Frameworka ne postoji [38].

4.1 SharpPcap i PacketDotNet biblioteke

Biblioteka unutar programskog jezika predstavlja skup nepromjenjivih izvora koje je moguće koristi. Ova aplikacija oslanja se na dvije vanjske biblioteke koje nisu dio standardnih biblioteka. Moguće ih je preuzeti koristeći NuGet *package manager* čijem korisničkom sučelju se pristupa kroz Visual Studio, gdje je moguće dodati, maknuti ili ažurirati određene pakete na projektu. Točne korištene verzije u ovome radu su SharpPcap 6.0.0 i PacketDotNet 1.2.0.

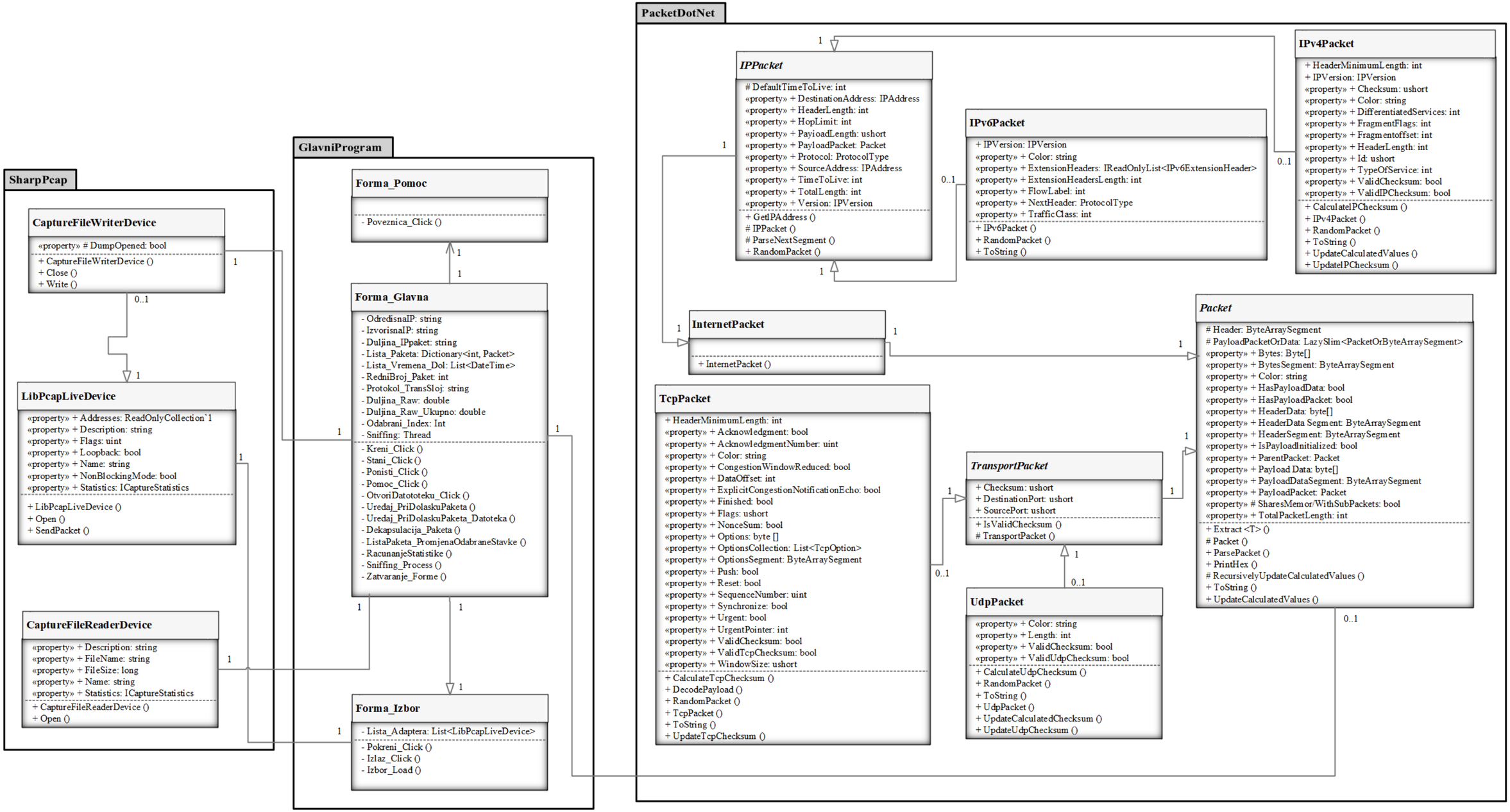
Glavna zadaća SharpPcapa je omogućiti hvatanje, mijenjanje te analizu mrežnih paketa za .NET aplikacije. Radi se o *fully managed* .NET biblioteci koja radi na više platformi (Windows, Linux, Mac). Omogućava prikaz liste mrežnih adaptera na računalu na kojemu se pokreće (*Live device lists*), prikupljanje određenih statističkih podataka o mrežnome sučelju (kartici) preko kojega se prikuplja promet. Glavna funkcija ove biblioteke jest čitanje paketa koji dolaze na odabrano mrežno sučelje. Pakete je moguće čitati i *offline*, odnosno iz *capture* (.pcap) datoteke te je prikupljene pakete isto tako moguće zapisati unutar istog formata datoteke. Filtriranje paketa moguće je kroz *Berkeley Packet Filter*. SharpPcap ima slojevitu arhitekturu te se na najvišoj razini nalaze klase koje rade na svim uređajima [39] [40].

- *CaptureDeviceList* – Dohvaća popis svih uređaja za prikupljanje
- *ICaptureDevice* – Svi uređaji za prikupljanje posjeduju *ICaptureDevice* sučelja

Hijerarhijski raspored *namespacea*:

- LibPcap
    - *LibPcapLiveDevice* – *IcaptureDevice* za *LibPcap*
    - *LibPcapLiveDeviceList* – Dohvaća popis *LibPcapLiveDevice* uređaja uključujući i winpcap i aripcap uređaje
    - *CaptureFileReaderDevice* – Uređaj koji čita .pcap datoteku
    - *CaptureFileWriterDevice* - Uređaj koji stvara .pcap datoteku odnosno zapisuje pakete unutar te datoteke

U starijim inačicama postojali su još WinPcap (kasnije Npcap) te AirPcap koji su se koristili za prikupljanje paketa unutar Windowsa odnosno bežičnih uređaja u slučaju AirPcapa. U novijim inačicama se sve izvodi preko LibPcapa. Na slici 18 unutar paketa SharpPcap vidljive su klase korišten u ovoj aplikaciji te njihovi odnosi koji se odnose na SharPcap biblioteku. *CaptureFileWriterDevice* klasa povezana je sa *LibPcapLiveDevice* klasom agregacijom što znači da je prvo potrebno imati aktivan uređaj odnosno sučelje kako bi bilo moguće koristiti funkciju zapisivanja u datoteku. S druge strane takvi preduvjeti nisu potrebni ako je čita iz datoteke.



Slika 18. Dijagram klasa

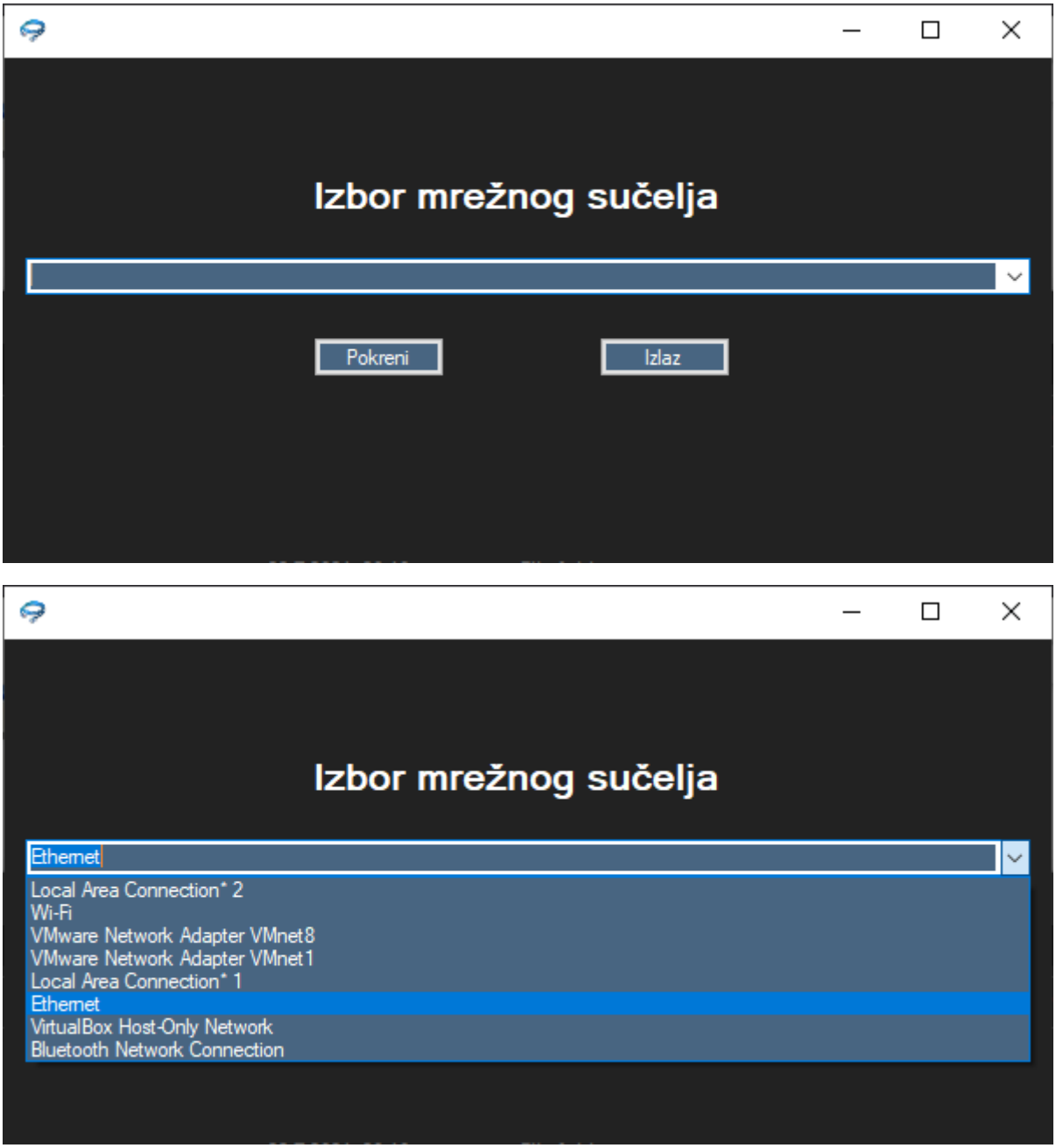
Nakon što se paketi prikupe potrebno ih je i interpretirati i prikazati, za to se koristi Packet.Net biblioteka. Radi se o *assemblyju* visokih performansi za seciranje i konstruiranje mrežnih paketa. Dizajniran je s naglaskom na što veće performanse te je cilj da izvršava minimalnu potrebnu količinu procesiranja podataka kako bi utvrdio ugniježđene datagrame. Na primjer TCP paket bi bio raščlanjen na niz povezanih objekata Ethernet -> IPv4 -> TCP ali se ne bi obavljala daljnja obrada podataka sve dok ne bi bilo potrebe za pristupom određenom polju. Podržava sve poznatije formate paketa kao što su:

- Ethernet,
- IPv4 / IPv6,
- TCP,
- UDP,
- ICMP,
- IGMP,
- OSPF,
- IEEE 802.11,
- ARP,
- SSL,
- PPP.

Na slici 18 unutar PacketDotNet paketa vidljive su klase korištene unutar aplikacije. Glavna iz koje proizlaze sve druge klase je *Packet*. Radi se o ugniježđenoj klasi kao i slučaju *InternetPacket*, *TransportPacket* te *IPPacket*. Cilj tih klasa je grupirati zajedno klase koje pripadaju određenoj cjelini. U slučaju *TransportPacket* klase radi se o paketima transportnog sloja prema TCP/IP složaju, UDP i TCP. *InternetPacket* objedinjuje pakete Internet sloja prema TCP/IP složaju te je unutar njega još jedna ugniježđena klasa *IPPacket* koja sadrži pakete odnosno klase paketa IPv4 i IPv6 protokola.

4.2 Analiza kôda i grafičkog sučelja

Pri otvaranju aplikacije korisnika će dočekati prozor s mogućnošću odabira mrežnog sučelja kao što je vidljivo na slici 19. Prilikom učitavanja forme deklarira se varijabla *LibPcapLiveDeviceList* kao lista uređaja s imenom *devices*. Kako bi sva sučelja dostupna za izbor bila funkcionalna potrebno je izvršiti određene provjere. Slika 20 prikazuje *foreeach* petlju koja obavlja tu provjeru. Za svako mrežno sučelje unutar ranije deklarirane liste provjerava se *lambda expression*om (linija 3, unutar zagrada) postoji li adresa, odnosno IP adresa tog sučelja te se ta dobivena *Bool* vrijednost okreće koristeći logičko NE na početku, budući da je potrebna istina kako bi se izvršila naredba *continue* u slučaju neispravnog sučelja te se preskočilo dodavanje takvoga sučelja na listu.

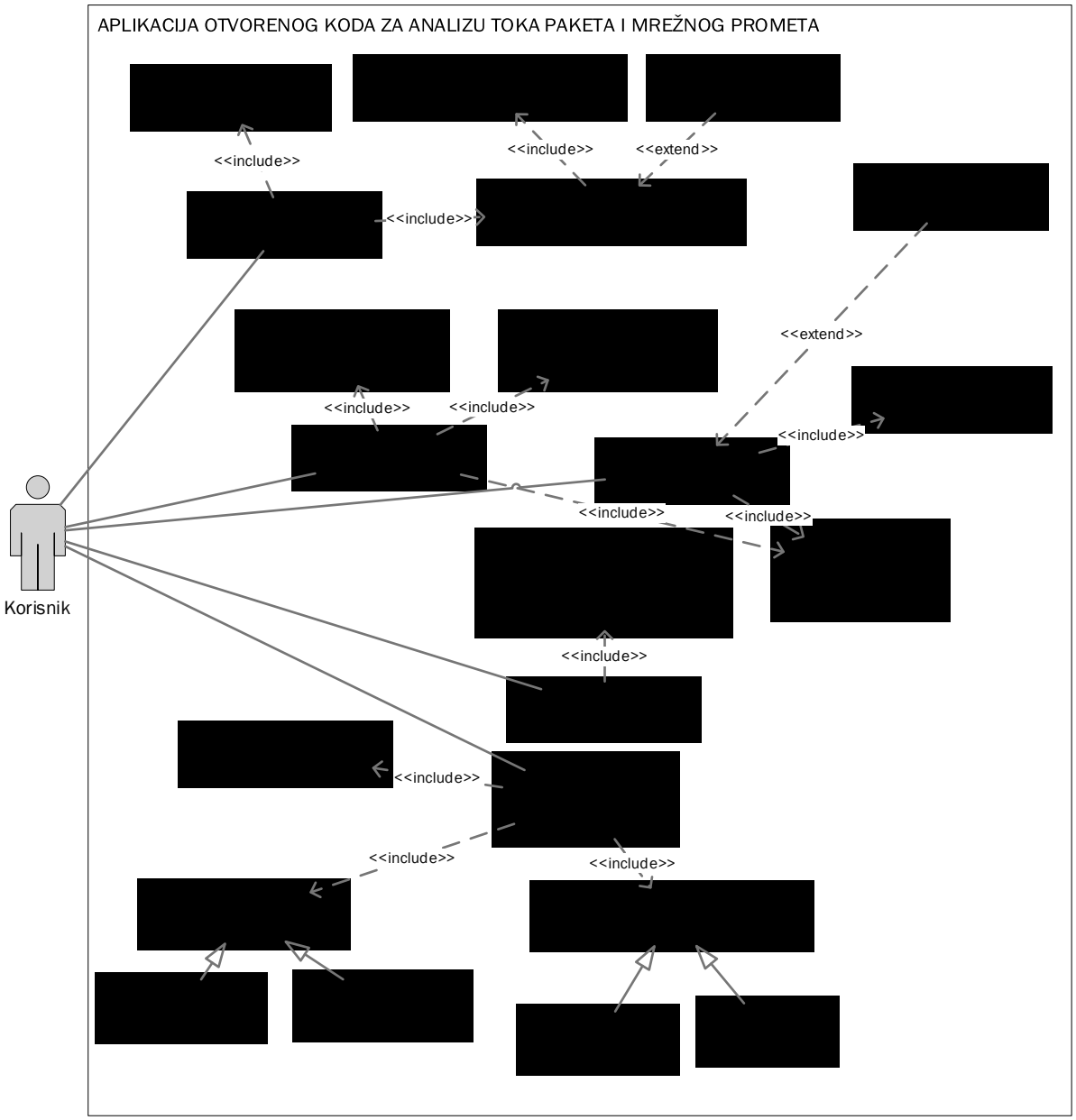


Slika 19. Početna forma pri otvaranju aplikacije

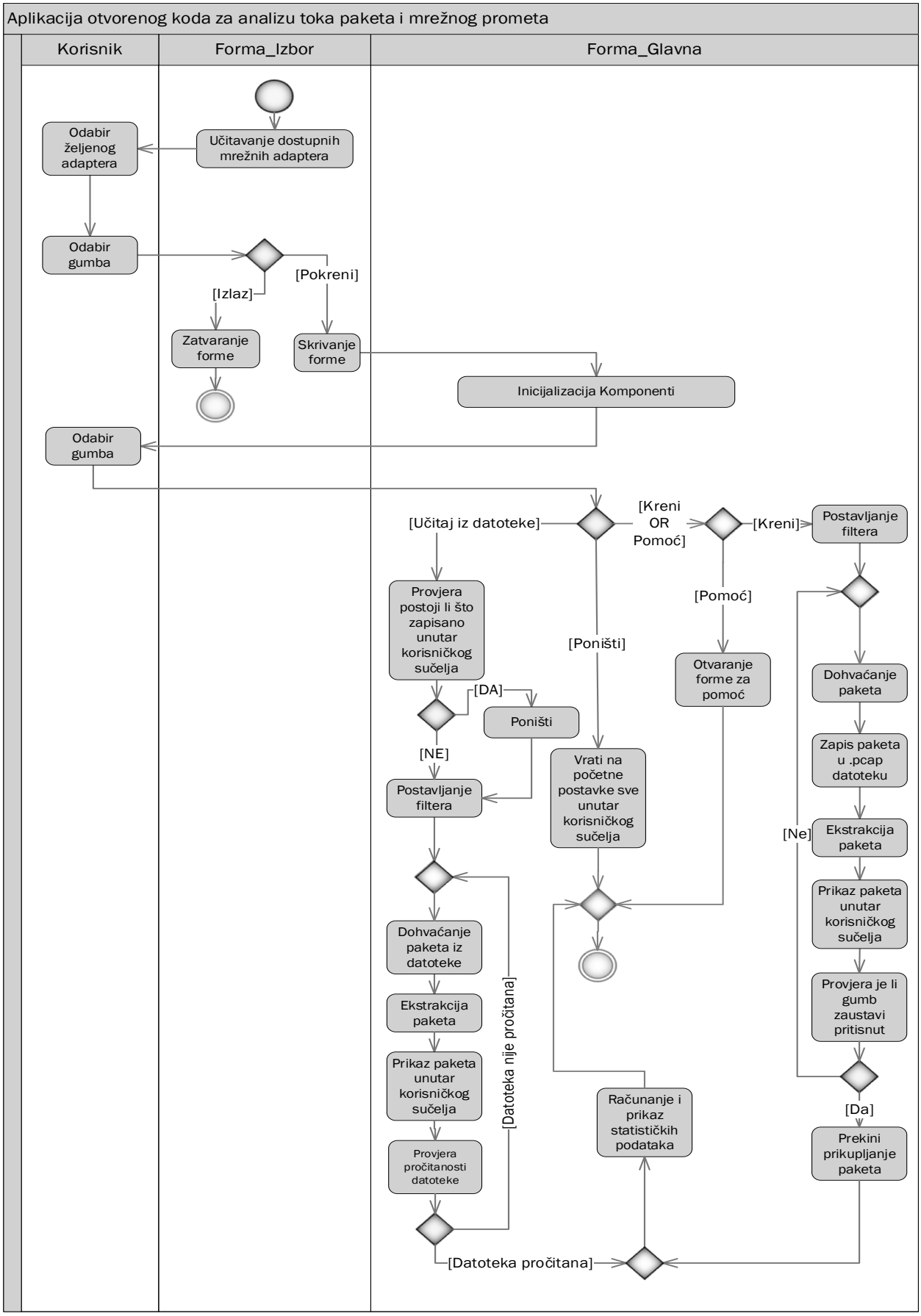
```
1.  foreach (LibPcapLiveDevice device in devices)
2.  {
3.      if (!device.Interface.Addresses.Exists(a => a != null && a.Addr != null && a.Ad
    dr.ipAddress != null)) continue;
4.      var devInterface = device.Interface;
5.      var friendlyName = devInterface.FriendlyName;
6.      var description = devInterface.Description;
7.
8.      lista_adaptera.Add(device);
9.      izborCombo.Items.Add(friendlyName);
```

Slika 20. Popunjavanje liste mrežnih sučelja

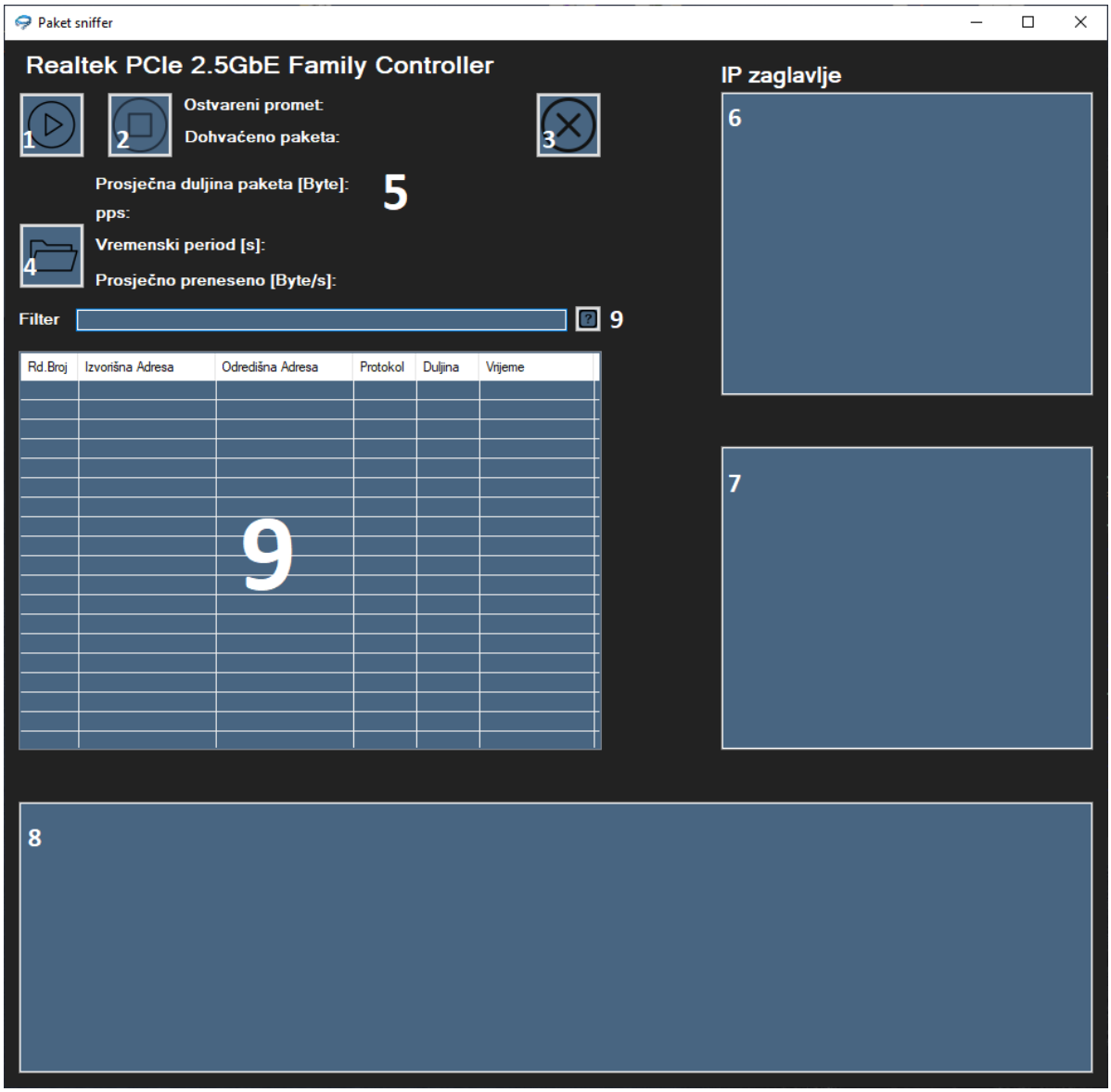
Nakon pritiska na gumb „Pokreni“ vrijednost izbora iz liste se prenosi na glavnu formu te se ona otvara dok se trenutna forma skriva. Cjelokupan način rada aplikacije prikazan je dijagramom aktivnosti na slici 22. Dok su funkcije obuhvaćene unutar glavne forme prikazane dijagramom slučaja uporabe na slici 21.



Slika 21. Funkcije glavne forme



Slika 22. Prikaz načina rada aplikacije



Slika 23. Glavna forma

Glavna forma sastavljena je od nekoliko elemenat čije su funkcije opisane ranije dijagram slučaja uporabe na slici 21, dok detaljan opis svakog od njih slijedi redom kao što je označeno na slici 23:

1. Gumb Pokreni –Prvo se inicijalizira metoda koja upravlja paketima pri dolasku. Zatim se pokreće *thread* koji upravlja uređajima za prikupljanje i zapis paketa te on pokreće proces prikupljanja. Uređaj odnosno sučelje za prikupljanje se uvijek postavlja u *promiscuous* način rada koji je detaljnije opisan u poglavlju 3. Nakon toga se unutar prve metode uzimaju podatci o tome kada je pojedini paket bio prihvaćen od strane upravljačkog programa (*drivera*). Duljina podatkovnog dijela paketa se uzima iz *RawCapture* čiji je format u obliku:
   `[RawCapture:LinkLayerType=Ethernet,Timeval=1627084398.926421s]`

Na temelju *RawCapturea* radi se daljnja ekstrakcija paketa u *IPPacket* te uzimanje potrebnih informacija (slika 24). Ako se ne radi o IP paketu na Internet sloju ekstrakcija takvog paketa bit će zanemarena odnosno preskočena.

```
1.              Packet packet1 = Packet.ParsePacket(rawpaket.LinkLayerType,
    rawpaket.Data);
2.              var ipPacket = packet1.Extract<PacketDotNet.IPPacket>();
3.              if (ipPacket != null)
4.              {
5.  System.Net.IPAddress srcIp = ipPacket.SourceAddress;
6.  System.Net.IPAddress dstIp = ipPacket.DestinationAddress;
7.  protocol_type = ipPacket.Protocol.ToString();
8.  sourceIP = srcIp.ToString();
9.  destinationIP = dstIp.ToString();
10. length = ipPacket.TotalPacketLength.ToString();
11. lista_paketa.Add(packetNumber, packet1);
12. lista_vremena.Add(rawpaket.Timeval.Date);
13. string datum = rawpaket.Timeval.Date.ToString("HH:mm:ss:ff");
```

Slika 24. Ekstrakcija IP paketa

2. Gumb Prekini – prekida i zatvara sve operacije koje su bile aktivne prilikom prikupljanja paketa kao što je otvoreni *thread*, aktivno sučelje te uređaj za zapis paketa u datoteku. Nakon što su operacije prekinute poziva se metoda za računanje statističkih podataka i ponovno omogućava korištenje ostalih gumbi.
3. Gumb Poništi – obavlja čišćenje *listview* elementa od podataka koji se u njemu nalaze, čisti liste koje služe za pohranu paketa i vremena dolazaka te vraća sve oznake (*label*) i brojač paketa na početno stanje.
4. Gumb Otvori datoteku – omogućava otvaranja datoteke koja sadrži pohranjene pakete. Podržane su isključivo .pcap datoteke koje mogu biti generirane od strane ove aplikacije ili neke druge koja podržava kreiranje takvih datoteka (npr. Wireshark). Postupak čitanje datoteke je sličan onome opisanome kod gumba „Pokreni“ uz razliku da ovdje ne postoji dodatni *thread* i uređaj za zapisivanje paketa u datoteku te se paketi čitaju uređajem za čitanje datoteka a ne onime za čitanje paketa s mrežnog uređaja u stvarnome vremenu.
5. Statističke oznake – označavaju statističke pokazatelje prikupljenih paketa:
    a. Ostvareni promet – računa se tako da se zbrajaju duljine *RawCapturea* ali samo za podržane pakete. To znači da svaki paket niže razine koji ne enkapsulira TCP ili UDP pakete neće biti ubrojen u ovome statističkom pokazatelju.
    b. Dohvaćeno paketa – predstavlja broj dohvaćenih paketa, isto kao i u prvome slučaju broje se samo podržani paketi.
    c. Prosječna duljina paketa – predstavlja omjer duljine svih podržanih paketa s brojem paketa
    d. pps - predstavlja broj dohvaćenih paketa po sekundi, zaokruženo na cijeli broj
    e. Vremenski period – period prikupljanja, vremenska razlika između prvoga paketa prikazanog na listi i zadnjeg. Mjereno u sekundama.

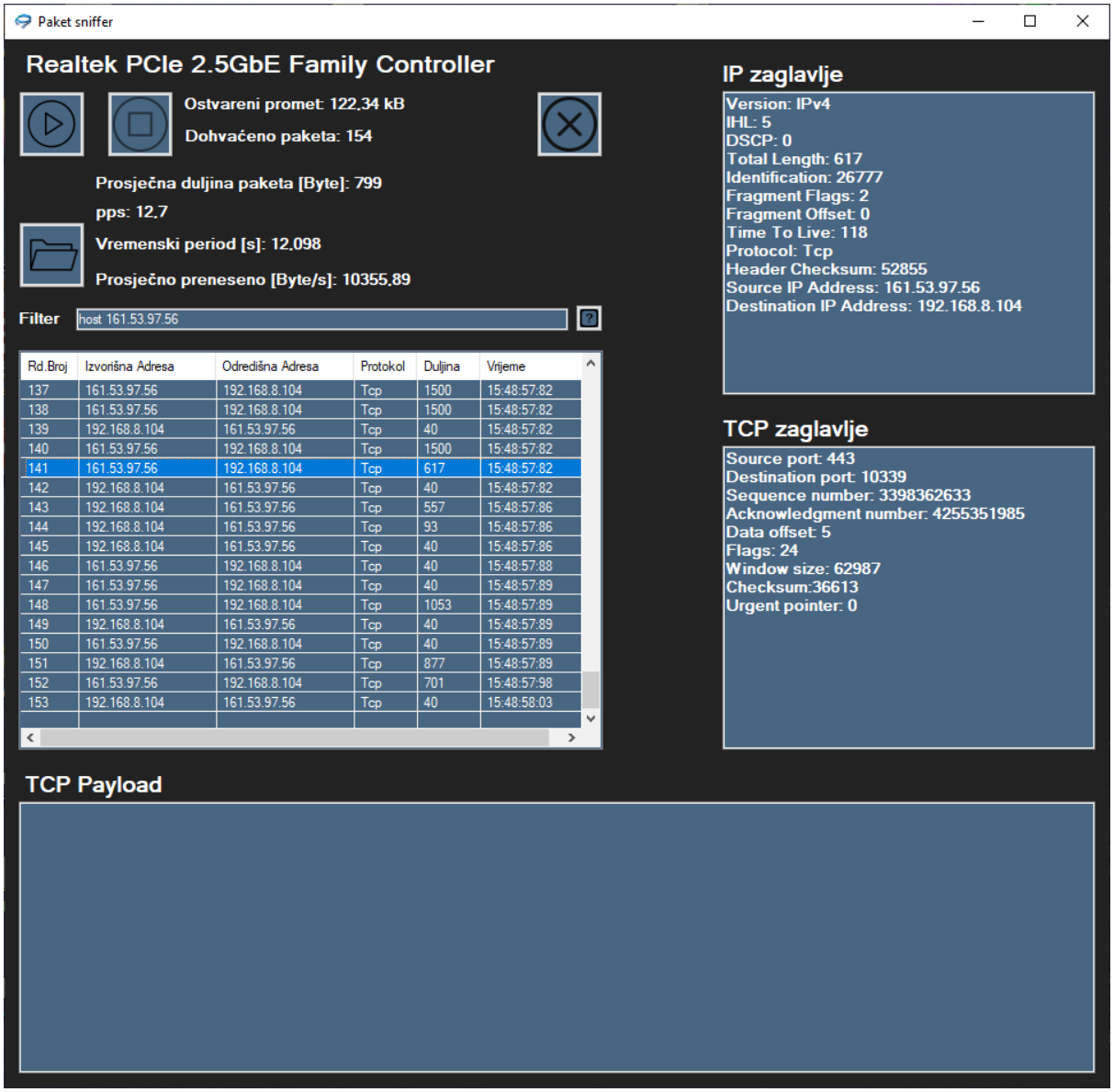
    f. Prosječno preneseno – prosječna brzina pristizanja podataka, računa se kao ostvareni promet kroz vremenski period prikupljanja. Izraženo u [Byte/s]
6. Zaglavlje IP protokola – ispisuje vrijednosti koje se nalaze unutar IP zaglavlja. Ovisno o protokolu mijenja se koja će polja biti ispisana, podržani su IPv4 i IPv6 protokoli.
7. Zaglavlje transportnog sloja – Ispisuje vrijednosti koje se nalaze u poljima protokola transportnog sloja ovisno o tome radi li se o UDP ili TCP protokolu
8. Sadržaj paketa – prikazuje sadržaj (*payload*) UDP ili TCP koji je pretvoren iz polja bajtova u ASCII format kako bi bio razumljiviji čovjeku
9. Filter – postavlja se prije početka prikupljanja paketa neovisno dali se radi prikupljanje u stvarnom vremenu ili iz datoteke. Radi se o *Berkeley Packet Filteru,* dok se najkorišteniji izrazi mogu se vidjeti klikom na gumb pored elementa za unos teksta filtera.
10. Lista paketa – centralno mjesto unutar aplikacije gdje su prikazani svi prikupljeni paketi i njihove vrijednosti: redni broj, izvorišna i odredišna adresa, protokol transportnog sloja, duljina IP paketa i vrijeme dolaska paketa.

5. Analiza toka paketa

Analiza toka paketa predstavlja proces prikupljanja i interpretacije podataka koji se kreću mrežom s ciljem boljeg razumijevanja onoga što je događa unutar mreže. Takve analize se većinom izvode mrežnim alatima koji se nazivaju *packet snifferima* te služe za prikupljanje neobrađenih podataka koji se kreću transportnim medijem. Analiza toka paketa može pomoći pri razumijevanju karakteristika prometa, otkrivanju tko se sve nalazi na mreži te koliko se prometa generira i prema komu, identifikacija vršnog vremena korištenja (vremenski period u kojemu se generira najviše prometa), identifikaciju malicioznih aktivnosti na mreži, pronalazak aplikacija koje predstavljaju sigurnosni rizik ili generiraju nepotreban promet i stvaraju „smeće“ unutar mreže itd.

Ova aplikacija je napravljena kao vrlo jednostavno rješenje za analizu mrežnog prometa. Takva jednostavnost znači da je vrlo jednostavno shvatiti način njenog rada, ima jako mali broj opcija te u suštini samo interpretira mrežni promet i prikazuje ga na korisniku razumljiv način. Iako nedostaje velik broj opcija počevši od podrške za samo Ethernet protokol na fizičkom sloju odnosno IP protokol na mrežnog sloju i TCP i UDP protokol na transportnom sloju, nemogućnost praćenja TCP veze i mnoštvo drugih, i dalje je moguće napraviti osnovnu analizu mrežnog prometa.

Mogućnost filtriranja prometa prema IP adresi omogućava izoliranje i analizu prometa prema određenom uređaju/domeni. Analizom domene www.fpz.unizg.hr koja se nalazi iza IP adrese 161.53.97.56 može se postaviti filter u obliku „host IPadresa“ te vidjeti sav promet u kojem je sudjelovala ta domeni, bilo kao odredište ili izvor (slika 25).



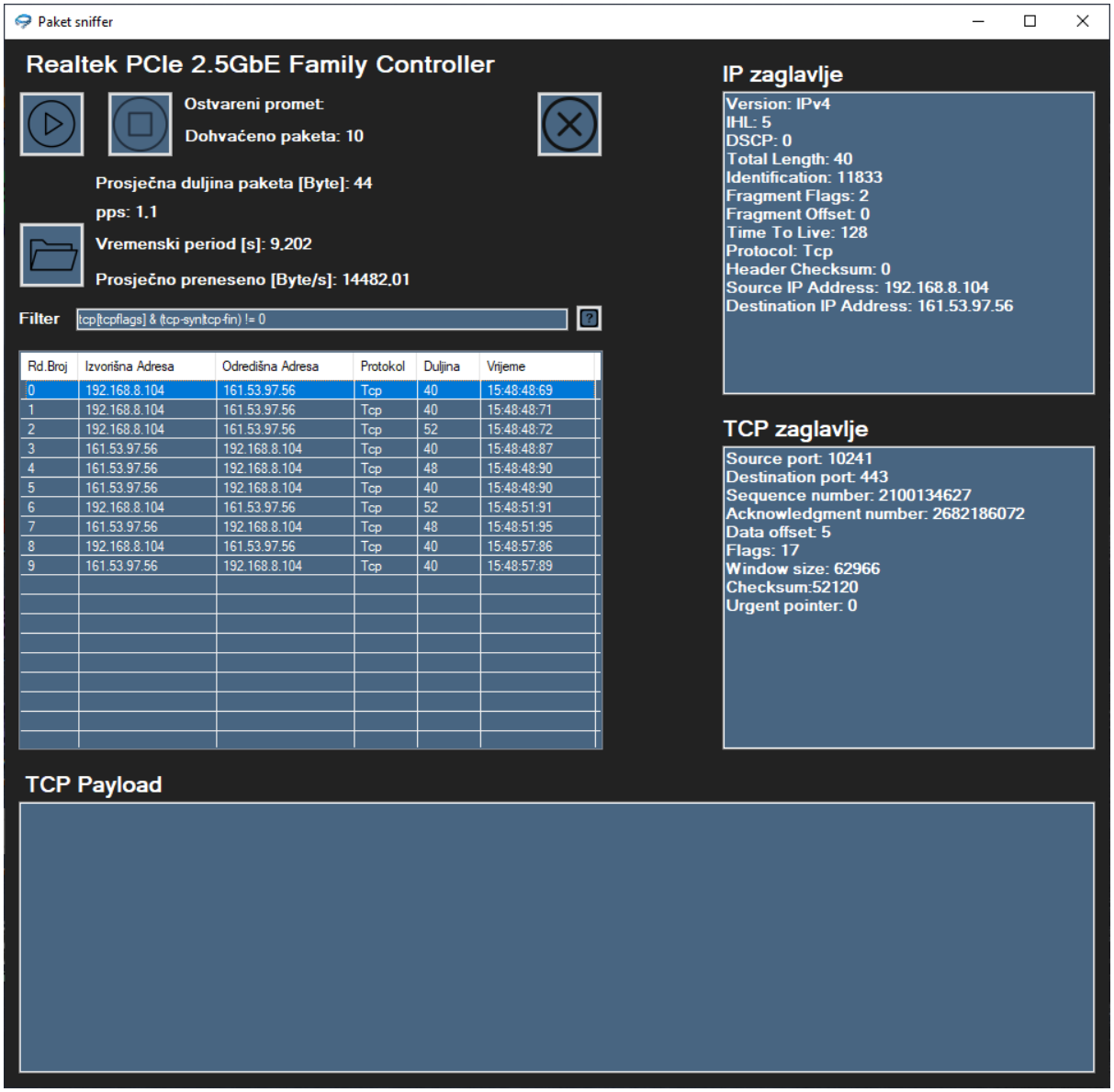
Slika 25. Promet generirani od strane mrežne stranice www.fpz.unizg.hr

Način na koji se izvodi filtriranje na temelju određenih vrijednosti unutar zaglavlja protokola je da se prvo upiše naziv protokola u obliku protokol[ ], zatim se unutar zagrada postavlja broj pomaka bitova za vrijednost koja se filtrira, dok se za operatore koriste standardni logički operateri and[&&], or[||], not[!] itd. Moguće je koristiti i tekstualne i znakovne operatore. Primjer filtriranja zastavica TCP protokola bio bi tcp[13] = 8, na tablici 3 vidljivo je da se zastavice nalaze na 13 pomaku. Vrijednost je postavljena na 8 prema decimalnom sustavu što odgovara vrijednosti PSH zastavice. Ista stvar se može postići i korištenjem samog naziva polja zaglavlja npr. tcp[tcpflags] = (tcp-psh) [41] [42].

Tablica 3. Prikaz pomaka bajtova [41]

TCP HEADER - RFC 793																													
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Offset 0								Offset 1								Offset 2								Offset 3							
Source Port Number																Destination Port Number															
Offset 4								Offset 5								Offset 6								Offset 7							
Sequence Number																															
Offset 8								Offset 9								Offset 10								Offset 11							
Acknowledgement Number																															
Offset 12								Offset 13								Offset 14								Offset 15							
Hacker Length				Reserved				CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
Offset 16								Offset 17								Offset 18								Offset 19							
Checksum																Urgent Pointer															
Offset 20								Offset 21								Offset 22								Offset 23							
TCP Options																															
Data																															

Vizualnom analizom je vidljivo je da sav promet sadrži TCP pakete, to se naravno može i provjeriti filtriranjem UDP paketa uz filter proto \udp. Pregledom porta u zaglavlju TCP paketa može se ustanoviti da se radi o HTTPS protokolu koji se nalazi iza porta 443. Iako je ranije navedeno da ova aplikacija ne podržava praćenje TCP veze, moguće je filtrirati TCP prema zastavicama koje sadržavaju. Tako se filterom tcp[tcpflags] & (tcp-syn|tcp-fin) != 0 prikazuju se samo oni paketi koji sadržavaju SYN i FIN (slika 26), odnosno paketi kojima započinje i završava TCP veza. Na slici 23 zaglavlje TCP paketa prikazuje vrijednost 17 pod poljem zastavica, to znači da taj paket sadrži FIN zastavicu čija je vrijednost 1 te uz nju i ACK zastavicu vrijednosti 16. Tako je moguće sastaviti TCP vezu prateći *Sequence* i *Acknowledgment* brojeve.



Slika 26. Prikaz TCP paketa sa SYN i FIN zastavicama

6. Analiza mrežnog prometa

Analiza mrežnog prometa obavlja se preko statističkih pokazatelja koji se računaju unutar aplikacije, gdje se prate već ranije navedeni pokazatelji. Prema [1] intenzitet dolazaka paketa može se izraziti Littelovim teoremom (1):

$$\lambda = \frac{N_{PAK}}{T[s]} \ [\frac{pak}{s}] \tag{1}$$

gdje je značenje oznaka sljedeće:

- λ – Intenzitet dolazaka paketa izražen u paketima po sekundi,
- N_{PAK} – ukupan broj paketa u sustavu,
- T – vremenski period prikupljanja uzorka.

Prosječna duljina paketa izražena je formulom (2):

$$\bar{p}[pak] = \frac{\sum p_1 + p_2 + \cdots + p_{N_{PAK}}}{N_{PAK}} \ [Byte] \tag{2}$$

gdje su značenja oznaka sljedeća:

- $\bar{p}[pak]$ – prosječna vrijednost duljine paketa,
- $\sum p_1 + p_2 + \cdots + p_{N_{PAK}}$ – zbroj duljina svih prikupljenih paketa,
- N_{PAK} – broj prikupljenih paketa.

S ta dva podatka moguće je izračunati ponuđeni promet koji prema [1] glasi (3):

$$a = \lambda \cdot \frac{p}{C} [bit/s] \tag{3}$$

dok je značenje oznaka sljedeće:

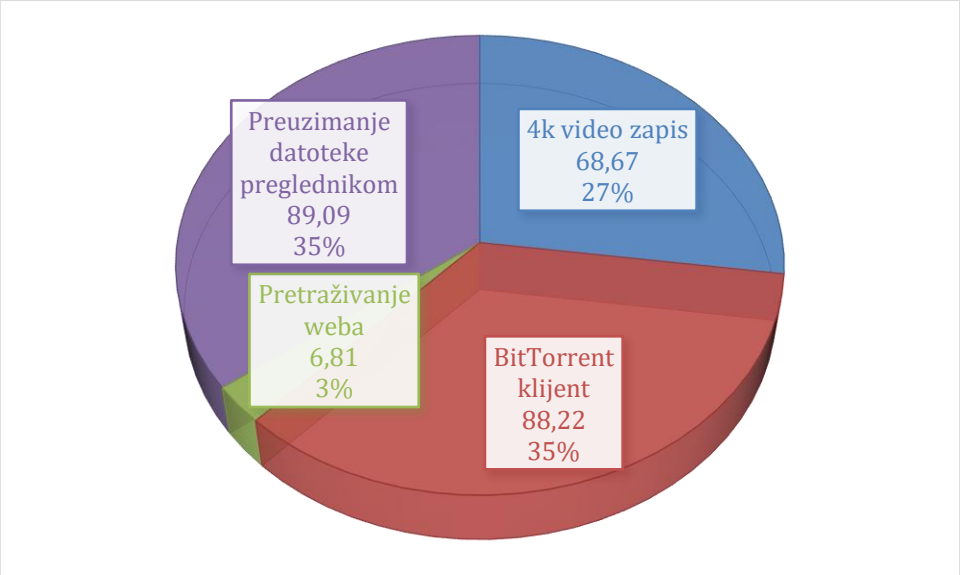
- a – ponuđeni promet,
- λ – Intenzitet dolazaka paketa,
- p – duljina paketa,
- C – kapacitet linka.

Promet je generiran za četiri slučaja prikazana unutar tablice 4, u svim slučajevima prikupljanje je trajalo 20 sekundi. Pri računanju ponuđenog prometa za svaki od primjera korišten je kapacitet linka od C=1000 Mbit.

Tablica 4. Promet generiran u primjerima

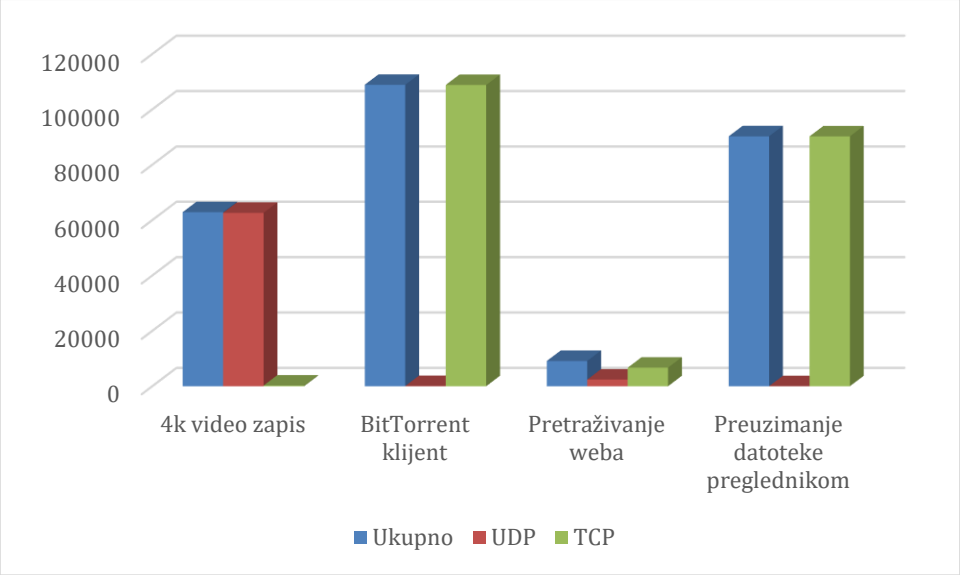
	4k video zapis	BitTorrent klijent	Pretraživanje weba	Preuzimanje datoteke preglednikom
N_{PAK}	62874	108882	9126	90275
$T[s]$	20	20	20	20
λ $[pak/s]$	3143,7	5444,1	456,3	4513,75
$\bar{p}$ $[Byte]$	1131	836	768	1021
a $[bit/s]$	28,44	36,41	2,8	36,87

Iz tablice 4 je vidljivo da je najveći ponuđeni promet dobiven pri preuzimanju datoteke preglednikom, dok je nešto manja vrijednost dobivena BitTorrent klijentom te nakon njega slijedi pregled video zapisa i u konačnici pretraživanje weba gdje je dobiven značajno manji promet nego u ova tri slučaja. Ovakvi omjeri odgovaraju količini ostvarenog prometa koji je izmjeren aplikacijom i prikazan na grafikonu 2 (vrijednosti izražene u megabajtima).



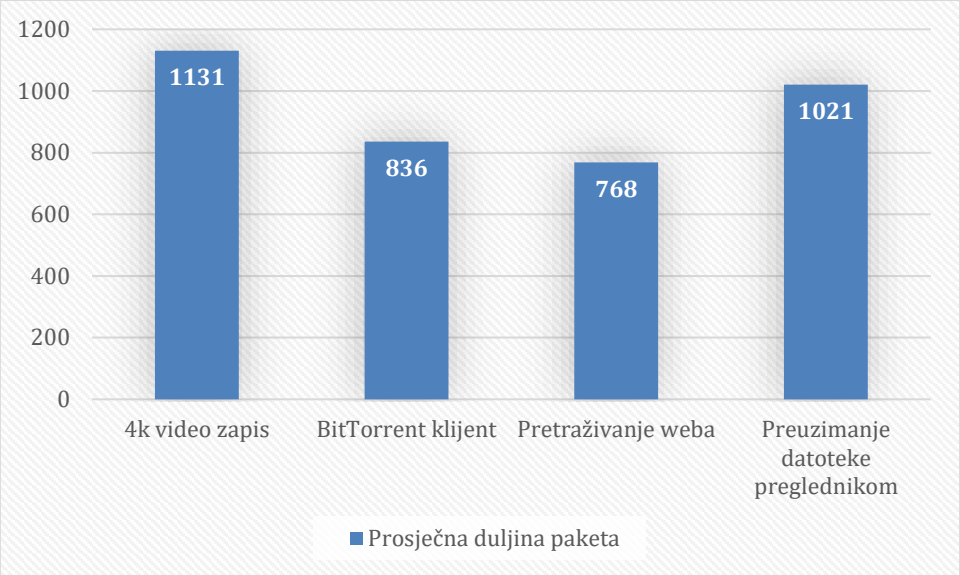
Grafikon 2. Ostvareni promet izmjeren aplikacijom

Ako se pogleda količina i vrsta paketa generirana u svakom od primjera na grafikonu 3 vidljivo je u onim slučajevima kada je dobiven najveći promet generirano i najviše paketa. U oba slučaja kada je generirano najviše paketa ujedno su skoro u potpunosti generirani samo TCP paketi, dok je u slučaju video zapisa trend obrnut i većinom su generirani UDP paketi. Pretraživanje weba imalo je nešto ujednačenije rezultate iako je 2/3 generiranih paketa TCP.



Grafikon 3. Raspodjela prikupljenih paketa

Kod prosječne duljine pakete (grafikon 4) vrijednosti su nešto drugačije. Ovdje je najveća duljina paketa u slučaju pregleda video zapisa, nešto manja kod preuzimanja datoteke dok je u slučaju korištenja BitTorrent klijenta svega malo veća nego kod pretraživanja weba.



Grafikon 4. Prosječna duljina paketa

7. Zaključak

Život bez Interneta većini je postao nezamisliv, iako većina ne razmišlja previše o načinu njegova funkcioniranja. U pozadini se odvija velik broj procesa kako bi primjerice na pametnom telefonu bila prikazana samo jedna slika koju je netko poslao, ili bi npr. bile prikazane stranice s vijestima. Iza svega toga nalaze se procesi koji omogućavaju kretanje podataka mrežom i njihovu interpretaciju na uređaju. Povećanje mrežnoga promet i uređaja koji ga generiraju, te sve veća prisutnosti poslovanja, obrazovanja, zabave i drugih sfera društva na Internetu dovela je i do sve veće potrebe za analizom podataka koji se kreću mrežama.

Cilj ovog rada bio je upoznati se obradom i interpretacijom podataka koji prolaze mrežom, te analizom prikazanih podataka. Vidjeti kako u praksi izgleda interpretacija podataka odnosno kako izgledaju podatci koje računalo zaprimi, način na koji se čitaju informacije iz zaglavlja paketa te na koji se način interpretira njihov sadržaj. Kako unutar te mase podataka pronaći i prikazati ono traženo.

U radu je prikazan način rada slojevite arhitekture, detaljno su objašnjene zadaće svih slojeva i međusobna komunikacija s ostalim slojevima. Unutar svakog sloja navedeno je i objašnjeno nekoliko najkorištenijih protokola i način na koji svi ti protokoli zajedno funkcioniraju kao cjelina. Omogućavajući tako pripremu podataka na prijenos mrežom i njihova ponovna sastavljanje i čitanje kada dođu do svoga odredišta.

Tijekom izrade same aplikacije pokazala su se određena ograničenja korištenih biblioteka, poput nemogućnosti dolaska do informacija koje postoje, ali čiji dohvat autor biblioteke nije predvidio. Skoro nepostojeća dokumentacija u kombinaciji s velikim promjenama od inačice do inačice same biblioteke dosta je otežala cijeli postupak. Sam C# pokazao se dosta dobar za ovu primjenu, iako je po jezicima korištenim za ovakvu vrstu aplikacija vidljivo da je za nešto kompleksnije aplikacije ipak bolje koristiti druga rješenja.

U konačnici rezultat je jednostavna aplikacija dosta ograničenih mogućnosti, ali koja ipak ispunjava svoju glavnu zadaću prikaza i interpretacije prometa koji se kreće mrežom. Korištenjem ugrađenih filtera moguće je izrazito detaljno filtrirati željeni promet dok uz statističke informacije korisnik dobiva informacije o paketima unutar mreže.

POPIS KRATICA

OSI - Open Systems Interconnection

ISO - International Organization for Standardization

ITU-T - International Telecommunication Union Telecommunication Standardization Sector

LLC - Logical Link Control

MAC - Media Access Control

IP - Internet Protocol

API - Application Program Interfaces

SSL - Secure Sockets Layer

HTTP - Hypertext Transfer Protocol

SMTP - Simple Mail Transfer Protocol

POP3 - Post Office Protocol 3

FTP - File Transfer Protocol

IRC - Internet Relay Chat

IOT - Internet of things

DNS - Domain Name System

ARPANET - Advanced Research Projects Agency Network

PPP - Point-to-Point Protocol

LCP - Link Control Protocol

NCP - Network Control Protocol

IEEE - Institute of Electrical and Electronics Engineers

CRC - Cyclical Redundancy Check

FCS - Frame Check Sequence

ARP - Address Resolution Protocol

RARP - Reverse Address Resolution Protocol

ICMP - Internet Control Message Protocol

IGMP - Internet Group Message Protocol

IPv4 - Internet Protocol Version 4

IPv6 - Internet Protocol version 6

CIDR - Classless Inter-Domain Routing

IANA - Internet Assigned Numbers Authority

RIR - Regional Internet Registries

RIPE NCC - Réseaux IP Européens Network Coordination Centre

IHL - Internet Header Length

QOS - Quality of Service

TOS - Type of Service

MTU - Maximum transmission unit

TTL - Time To Live

NAT - Network address translation

IPsec - Internet Protocol Security

MPLS - Multiprotocol Label Switching

UDP - User Datagram Protocol

TCP - Transmission Control Protocol

ISN - Initial sequence number

DHCP - Dynamic Host Configuration Protocol

DNS - Domain Name System

ECMA - European Computer Manufacturers Association

CLI - Common Language Infrastructure

CLR - Common Language Runtime

JIT - Just-In-Compiler

LINQ - Language Integrated Query

POPIS LITERATURE

[1] Mrvelj Š. *Autorizirana predavanja*. Fakultet prometnih znanosti. 2021.

[2] Forouzan B.A. *TCP/IP protocol suite / Behrouz A. Forouzan.—4th ed*. New York: Raghothaman Srinivasan; 2010.

[3] Kozierok C.M. *The TCP/IP Guide*; 2005.

[4] Tezupur University. *Session Layer.* http://www.tezu.ernet.in/~utpal/course_mat/wpi_sess_leR.html [Pristupljeno 26 travanj 2021].

[5] UNICEN. *Protocols and Layers.* https://users.exa.unicen.edu.ar/catedras/comdat1/material/TP1-Ejercicio5-ingles.pdf [Pristupljeno 27 travanj 2021].

[6] PMF. *TCP/IP model.* http://www.phy.pmf.unizg.hr/~dandroic/nastava/ramr/poglavlje_3.pdf [Pristupljeno 27 travanj 2021].

[7] Fujitsu. *TCP/IP Protocol Suite Tutorial;* 2006.

[8] CISCO. *Point-to-Point Protocol*. https://www.cisco.com/c/en/us/tech/wan/point-to-point-protocol-ppp/index.html [Pristupljeno 28 travanj 2021].

[9] Phoenix Contact. *Ethernet Basics Rev.02*. https://www.mouser.com/pdfdocs/Ethernet_Basics_rev2_en.pdf [Pristupljeno 28 travanj 2021].

[10]. IEB media. INTRODUCTION TO ETHERNET. *The Industrial Ethernet Book*. 1999; 1(3): 6.

[11] Ram, V. *The Internet Layer in the TCP/IP Model.* https://www.tutorialspoint.com/The-Internet-Layer-in-the-TCP-IP-Model [Pristupljeno 29 travanj 2021].

[12] Fall K., Stevens R. *TCP/IP Illustrated, Volume 1: The Protocols 2nd Edition*; Arizona: Addison-Wesley Professional; 2011.

[13] Pearson. *NETWORK LAYER/INTERNET PROTOCOLS.* https://www.pearsonhighered.com/assets/samplechapter/0/6/7/2/0672322080.pdf [Pristupljeno 1 svibanj 2021].

[14] Fuller V. *Classless Inter-domain Routing (CIDR).* https://datatracker.ietf.org/doc/html/rfc4632 [Pristupljeno 16 svibanj 2021].

[15] keycdn. *What Is CIDR (Classless Inter-Domain Routing)?*. https://www.keycdn.com/support/what-is-cidr [Pristupljeno 16 svibanj 2021].

[16] CARNet Cert u suradnji s LS&S. *Dodjeljivanje IP adresa*. Zagreb: CCERT; 2007. https://www.cis.hr/www.edicija/LinkedDocuments/CCERT-PUBDOC-2007-09-203.pdf [Pristupljeno 16 svibanj 2021].

[17] Pearson. *Anatomy of an IPv4 Packet.* https://www.pearsonitcertification.com/articles/article.aspx?p=1843887 [Pristupljeno 19 svibanj 2021].

[18] Electronic Communications Committee. *IPV6 – THE NEXT GENERATION INTERNET PROTOCOL*. ECC. Broj izvješća: 78, 2006.

[19] Sangam Racherla JD. *IPv6 Introduction and Configuration*: An IBM Redpaper publication. 2020.

[20] Postel J. *INTERNET CONTROL MESSAGE PROTOCOL* https://datatracker.ietf.org/doc/html/rfc792 [Pristupljeno 1 lipanj 2021].

[21] Villanova University. *Internet Control Message Protocol* http://www.csc.villanova.edu/~mdamian/Past/networksfa12/Notes/ICMP.pdf [Pristupljeno 1 lipanj 2021].

[22] Academy, Cisco Networking. *Chapter 10: Application Layer* http://mars.tekkom.dk/mediawiki/images/5/53/ITNv51_InstructorPPT_CH10.pdf [Pristupljeno 12 lipanj 2021].

[23] Novell. *DNS/DHCP Services* http://www.novell.com/documentation/dns_dhcp/pdfdoc/dhcp_enu/dhcp_enu.pdf. [Pristupljeno 12 lipanj 2021].

[24] Bucknell University. *Dynamic Host Configuration Protocol* https://datatracker.ietf.org/doc/html/rfc2131#section-1 [Pristupljeno 12 lipanj 2021].

[25] Zarki, Magda El. *Ch 6: Networking Services: NAT, DHCP, DNS, Multicasting* https://www.ics.uci.edu/~magda/cs620/ch6.pdf [Pristupljeno 13 lipanj 2021].

[26] Mozilla. *An overview of HTTP* https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview [Pristupljeno 13 lipanj 2021].

[27] Tutorialspoint. *HTTP - Overview* https://www.tutorialspoint.com/http/http_overview.htm [Pristupljeno 13 lipanj 2021].

[28] Networx Security. *Packet analyzer* https://www.networxsecurity.org/members-area/glossary/p/packet-sniffing.html [Pristupljeno 17 lipanj 2021].

[29] DNSstuff. *Best 10 Packet Sniffer and Capture Tools* https://www.dnsstuff.com/packet-sniffers [Pristupljeno 17 lipanj 2021].

[30] Qadeer M.A, Siddiqui M.R. Network Traffic Analysis and Intrusion Detection Using Packet Sniffer. *Communication Software and Networks, International Conference on*. 2010; 0: 313-317.

[31] CARNet Cert u suradnji s LS&S. *Analiza alata Wireshark* https://www.cis.hr/www.edicija/LinkedDocuments/NCERT-PUBDOC-2010-09-312.pdf [Pristupljeno 18 lipanj 2021].

[32] Sharpe R., Warnicke E., Lamping U. *Wireshark User's Guide* https://www.wireshark.org/docs/wsug_html_chunked/ [Pristupljeno 18 lipanj 2021].

[33] Netresec. *NetworkMiner* https://www.netresec.com/?page=networkminer [Pristupljeno 19 lipanj 2021].

[34] Network Analyzers. *Why Choose TCPDUMP* https://www.networkanalyzers.co/why-choose-tcpdump/ [Pristupljeno 19 lipanj 2021].

[35] Wikipedia. *tcpdump* https://en.wikipedia.org/wiki/Tcpdump [Pristupljeno 19 lipanj 2021].

[36] Tutorialspoint. *C# - Overview* https://www.tutorialspoint.com/csharp/csharp_overview.htm [Pristupljeno 21 srpanj 2021].

[37] Microsoft. *A tour of the C# language* https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [Pristupljeno 21 srpanj 2021].

[38] Knowledgehut. *C# Introduction* https://www.knowledgehut.com/tutorials/csharp/csharp-introduction [Pristupljeno 21 srpanj 2021].

[39] Gal T., Morgan C. *SharpPcap - A Packet Capture Framework for .NET* https://www.codeproject.com/Articles/12458/SharpPcap-A-Packet-Capture-Framework-for-NET [Pristupljeno 22 srpanj 2021].

[40] Morgan C. *sharppcap* https://github.com/chmorgan/sharppcap#readme [Pristupljeno 22 srpanj 2021].

[41] Gigamon. *BPF Reference Guide* https://www.gigamon.com/content/dam/resource-library/english/guide---cookbook/gu-bpf-reference-guide-gigamon-insight.pdf [Pristupljeno 26 srpanj 2021].

[42] IBM. Berkeley packet filters https://www.ibm.com/docs/en/qsip/7.4?topic=queries-berkeley-packet-filters [Pristupljeno 26 srpanj 2021].

POPIS KORIŠTENIH SLIKA

POPIS KORIŠTENIH TABLICA

POPIS KORIŠTENIH GRAFIKONA

Sveučilište u Zagrebu
Fakultet prometnih znanosti
10000 Zagreb
Vukelićeva 4

IZJAVA O AKADEMSKOJ ČESTITOSTI I SUGLASNOST

Izjavljujem i svojim potpisom potvrđujem kako je ovaj diplomski rad isključivo rezultat mog vlastitog rada koji se temelji na mojim istraživanjima i oslanja se na objavljenu literaturu što pokazuju korištene bilješke i bibliografija.

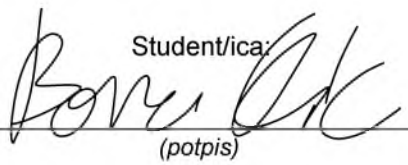
Izjavljujem kako nijedan dio rada nije napisan na nedozvoljen način, niti je prepisan iz necitiranog rada, te nijedan dio rada ne krši bilo čija autorska prava.

Izjavljujem također, kako nijedan dio rada nije iskorišten za bilo koji drugi rad u bilo kojoj drugoj visokoškolskoj, znanstvenoj ili obrazovnoj ustanovi.

Svojim potpisom potvrđujem i dajem suglasnost za javnu objavu diplomskog rada pod naslovom **RAZVOJ I PRIMJENA APLIKACIJE OTVORENOG KODA ZA ANALIZU TOKA PAKETA I MREŽNOG PROMETA** na internetskim stranicama i repozitoriju Fakulteta prometnih znanosti, Digitalnom akademskom repozitoriju (DAR) pri Nacionalnoj i sveučilišnoj knjižnici u Zagrebu.

U Zagrebu, 31.8.2021

Student/ica:



(potpis)