Mjerenje QoS kod IP DTV uređaja

Ivošević, Milan

Master's thesis / Diplomski rad

2017

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Josip Juraj Strossmayer University of Osijek, Faculty of Electrical Engineering, Computer Science and Information Technology Osijek / Sveučilište Josipa Jurja Strossmayera u Osijeku, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:200:774726

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-03-29**

Repository / Repozitorij:

Faculty of Electrical Engineering, Computer Science and Information Technology Osijek

zir.nsk.hr

Image not found or type unknown

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

Sveučilišni studij

MJERENJE QOS KOD IP DTV UREĐAJA

Diplomski rad

Milan Ivošević

Osijek, 2017.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 21.09.2017.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Milan Ivošević
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 798 R, 12.10.2015.
OIB studenta:	88775827851
Mentor:	Doc.dr.sc. Mario Vranješ
Sumentor:	
Sumentor iz tvrtke:	Matija Pul
Predsjednik Povjerenstva:	Doc.dr.sc. Ratko Grbić
Član Povjerenstva:	Izv. prof. dr. sc. Marijan Herceg
Naslov diplomskog rada:	Mjerenje QoS kod IP DTV uređaja
Znanstvena grana rada:	**Informacijski sustavi (zn. polje računarstvo)**
Zadatak diplomskog rada:	U suvremenim DTV (engl. Digital Television) uređajima IP (engl. Internet Protocol) sadržaji postaju ravnopravni sa standardnim DVB (engl. Digital Video Broadcasting) sadržajima. Kod testiranja rada DTV uređaja s IP sadržajima, vrlo je bitno konstantno praćenje kvalitete IP usluge. Cilj rada je pregled postojećih metoda analize QoS (engl. Quality of Service) kod IP mreža, kao i prilagođenje i integracija jedne od metoda u BBT (engl. Black Box Testing) okruženje. (sumentor Matija Pul, Institut RT-RK Osijek, Cara Hadrijana 10b)
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	21.09.2017.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA
I INFORMACIJSKIH TEHNOLOGIJA **OSIJEK**

IZJAVA O ORIGINALNOSTI RADA

Osijek, 27.09.2017.

Ime i prezime studenta:	Milan Ivošević
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 798 R, 12.10.2015.
Ephorus podudaranje [%]:	1

Ovom izjavom izjavljujem da je rad pod nazivom**: Mjerenje QoS kod IP DTV uređaja**

izrađen pod vodstvom mentora Doc.dr.sc. Mario Vranješ

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

Sadržaj

1. UVOD 1
2. IP DTV USLUGE – STANDARDI, PROTOKOLI, QOS 3
2.1. DTV preko IP protokola 3
2.2. Specifikacije protokola 9
2.3. Predložene metrike za QoS 18
3. POSTOJEĆA RJEŠENJA ZA MJERENJE QOS 21
3.1. Kriterij odabira postojećih rješenja 21
3.2. Testiranje postojećih rješenja 22
4. IMPLEMENTACIJA APLIKACIJE ZA MJERENJE QOS 33
4.1. Arhitektura sustava mjerenja 33
4.2. Implementacija metrika 34
4.3. Integracija s okolinom 37
5. ANALIZA ISPRAVNOSTI IMPLEMENTIRANE APLIKACIJE 39
5.1. Testna okolina i testni slučajevi 39
5.2. Prikaz rezultata 41
6. ZAKLJUČAK 45

1. UVOD

Proširenje standardnih zemaljskih i satelitskih DTV (engl. *Digital Television*) usluga korištenjem IP-a (engl. *Internet Protocol*) privlačno je jer omogućuje pružanje širokog spektra dodatnih (novih) usluga. Neke od aplikacija omogućene ovakvim sustavima su pristup *web* aplikacijama uz standardni TV sadržaj, prikaz nelinearnog TV sadržaja (vremenski pomaknut, snimljeni sadržaj) i pristup sadržaju na zahtjev. Navedene aplikacije mogu znatno poboljšati korisničko iskustvo gledanja televizije. Velika prednost implementacije DTV-a putem IP mreža je postojeća robusna infrastruktura i standardiziran protokolarni stog koji podržava širok spektar aplikacija. Uz visoku dostupnost širokopojasnog pristupa Internetu po relativno niskim cijenama, te postojanje fizičke infrastrukture visoke propusnosti, u mnogim zemljama prijenos TV sadržaja preko IP-a postao je brzo rastući segment tržišta.

Ozbiljno ograničenje prijenosa TV sadržaja preko IP-a je zahtjev za visokom propusnošću mreže, koju TV sadržaj zahtjeva. Uz konstante zahtjeve za povećanjem rezolucije i broja sličica po sekundi TV sadržaja, potrebno je korisniku pružiti i pristup popratnim uslugama kao što su: EPG (engl. *Electronic Program Guide*), interaktivne popratne aplikacije i igre, te snimanje i gledanje sadržaja na zahtjev. Kako bi korisničko iskustvo bilo zadovoljavajuće, pružatelj usluge mora osigurati određenu razinu QoS (engl. *Quality of Service* - kvaliteta usluge) u distribucijskoj mreži. Na današnjem tržištu postoje dva pristupa prijenosu TV sadržaja preko IP-a. Internet TV servisi korisnicima dostavljaju sadržaj kroz otvoreni Internet. Stoga imaju ograničenu kontrolu nad aspektima kvalitete usluge, dok IPTV (engl. *Internet Protocol TV*) operateri sadržaj distribuiraju izravno pretplatnicima kroz zatvorene i kontrolirane distribucijske mreže te mogu garantirati razinu QoS vlastitih usluga [1]. Širokopojasni internetski pristup iskorištavaju i hibridni TV standardi. Tako HbbTV (engl. *Hybrid Broadcast Broadband TV*)[2] definira model u kojem se linearni sadržaji prenose emisijskim putem, dok se za prijenos nelinearnih sadržaja i interaktivnih aplikacija koristi IP. U svim navedenim slučajevima kvaliteta dostavljenog sadržaja ovisi o razini kvalitete usluge prijenosne mreže.

Cilj ovog rada je istražiti koji su QoS parametri relevantni za prijenos TV sadržaja putem IP protokola, pronalaženje načina mjerenja njihovih vrijednosti s gledišta lokalne mreže klijenta i integriranje tih mjerenja u postojeći BBT (engl. *Black Box Testing*) sustav testiranja STB (engl. *Set Top Box*) uređaja. Prikupljanjem i analizom mrežnih metrika sa strane klijenta bave se programi kao što su tcpdump [3], Wireshark [4], Ettercap [5], Etherape [6]. Mogućnosti ovih programa, metrike koje pružaju i mogućnost njihove integracije u testno okruženje razmatraju se ovim radom. Također je razmotrena mogućnost prikupljanja mrežnih metrika preuzimanjem

paketa od upravljačkog programa mrežne kartice i analizom zaglavlja komunikacijskih protokola. Programske biblioteke za preuzimanje i analizu mrežnih paketa razmotrene ovim radom su libpcap [7] i libtins [8].

U drugom poglavlju ovog rada predstavljeni su standardi za prijenos DTV signala putem IP protokola. Opisani su mehanizmi prijenosa podataka TCP (engl. *Transmission Control Protocol*) i RTP (engl. *Realtime Transport Protocol*) preko UDP-a (engl. *User Datagram Protocol*) protokolima koji čine osnovu pružanja TV usluga putem IP-a, definirane su moguće greške u prijenosu podataka i mehanizmi upravljanja greškama. Definirane su metrike kvalitete usluge i njihov utjecaj na kvalitetu TV sadržaja. U trećem poglavlju predstavljena su postojeća rješenja za nadziranje parametara mreže s klijentske strane. Definirani su kriteriji prihvatljivosti rješenja za integraciju u testnu okolinu. Navedena rješenja su testirana i ocijenjena prema definiranim kriterijima. Odabrano je najprihvatljivije rješenje, te su pomoću njega implementirane metrike definirane u drugom poglavlju. Implementacija aplikacije za mjerenje opisana je u četvrtom poglavlju. U petom poglavlju prikazana je usporedba rezultata dobivenih implementiranim mjerenjima s rezultatima referentnih mjerenja. Šesto poglavlje donosi zaključke rada.

2. IP DTV USLUGE – STANDARDI, PROTOKOLI, QOS

U ovom poglavlju predstavljeni su komunikacijski protokoli korišteni u prijenosu TV sadržaja putem Interneta. Internet TV i IPTV standardi razlikuju se u implementaciji prijenosnog i sesijskog sloja ISO/OSI modela. Više slojeve protokolarnog stoga čine kodirani video i audio paketi, te metapodaci enkapsulirani u prijenosni format kao što je MPEG-2 prijenosni tok. Protokoli prijenosnog sloja korišteni za prijenos TV sadržaja razmatrani u ovom radu su TCP i RTP. U nastavku su opisani mehanizmi funkcioniranja navedenih protokola. Također su opisane metrike kvalitete usluge IP mreže i njihov utjecaj na prijenos TV sadržaja.

2.1. DTV preko IP protokola

U digitalnoj televiziji korisnicima se putem prijenosnog medija dostavlja digitalno uzorkovan i kodiran TV sadržaj. Elementarne komponente TV signala (video i audio, podnaslovi, teletekst) zasebno su enkapsulirane u paketizirane elementarne tokove (engl. *Packetized Elementary Stream* - PES). Elementarne komponente čiji sadržaj čini jedinstvenu cjelinu (TV program) povezane su u isti programski tok (engl. *Program stream* - PS). Paketi elementarnih komponenti jednog ili više programskih tokova multipleksirani su u jedinstven prijenosni tok (engl. *Transport Stream* - TS). Ovisno o prijenosnom mediju, prijenosni tok se digitalno modulira i šalje korisnicima. Korisnički prijemni uređaj odabire odgovarajući komunikacijski kanal (npr. frekvencijski opseg kod emisijskog prijenosa) i iz njega izdvaja pakete prijenosnog toka te obavlja postupak demultipleksiranja elementarnih tokova, dekodiranje i reprodukciju elementarnih komponenti. Postupak paketizacije i multipleksiranja TV sadržaja prikazan na slici 2.1. zajednički je svim DTV uslugama, neovisno o prijenosnom mediju.

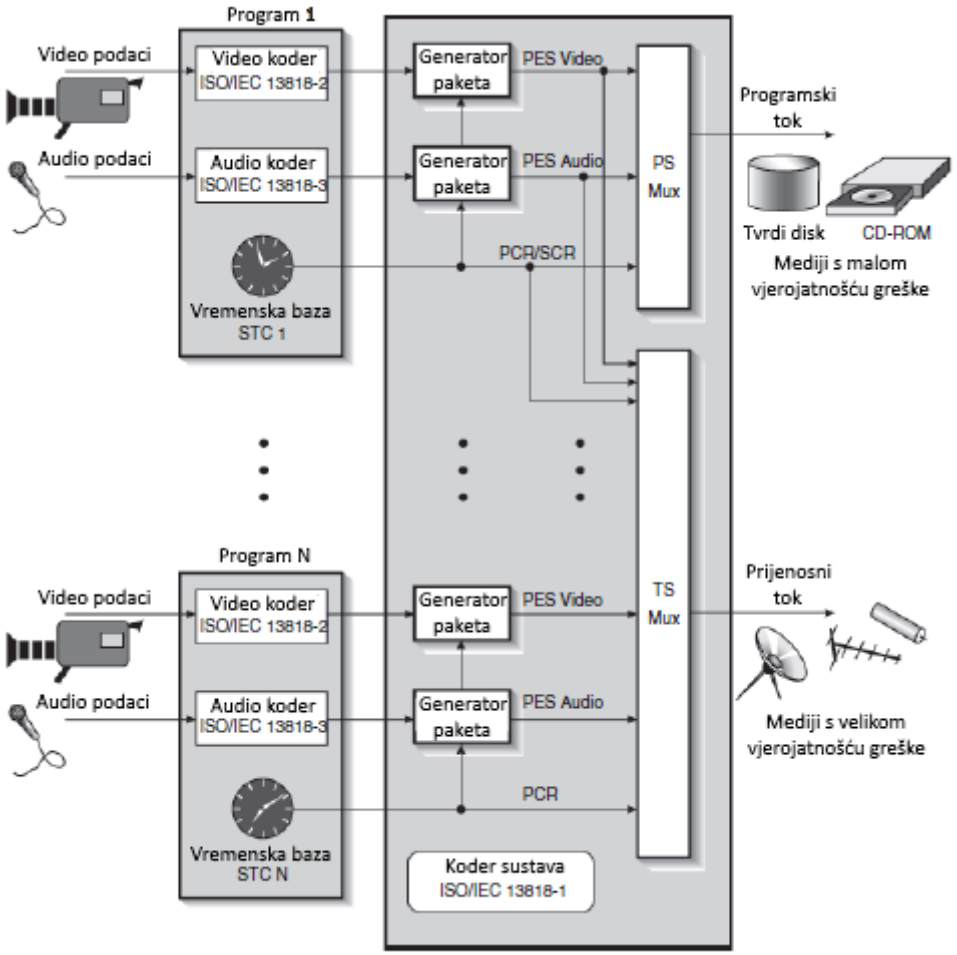
2.1.1. Specifičnosti IP prijenosa

Prijenos DTV preko IP protokola dijeli navedene karakteristike s emisijskim, satelitskim i kabelskim prijenosom. Mrežni protokolarni stog zasnovan na IP protokolu korišten kao prijenosni medij čini temeljnu razliku. TS paketi enkapsulirani su u pakete protokola prijenosnog ili aplikacijskog sloja TCP/IP mrežnog stoga. Specifičnosti mrežnog stoga kao prijenosnog medija su sljedeće:

- Nisu potrebni postupci modulacije i zaštitnog kodiranja prijenosnog toka. Ovi postupci su najčešće implementirani protokolima fizičke razine mrežnog stoga.
- Krajnje točke komunikacije proizvođača i potrošača sadržaja implementirane su programski. Kod emisijskog prijenosa, krajnju točku potrošača predstavlja

demodulator, sklopovska komponenta koja digitalno modulirani signal pretvara u pakete prijenosnog toka. Kod IP prijenosa, krajnja točka potrošača TV sadržaja je upravljački program mrežnog sučelja i protokolarni stog operacijskog sustava, koji prijenosni tok izdvaja iz mrežnih paketa.

- Prijenosni medij omogućuje dvosmjernu komunikaciju. Korisnik može zatražiti uslugu (npr. TV program) slanjem povratne poruke. Pružatelj usluge ne razašilje svoje program svim korisnicima odjednom, već samo onima koji su je zatražili, čime se postiže efikasnije korištenje komunikacijskog kanala.
- Ograničena propusnost prijenosnog medija. Ne mora postojati poseban distribucijski kanal u kojem poslužitelj neometano dostavlja sadržaj korisniku. Usko grlo predstavlja pristupna točka korisničke lokalne mreže, čija je propusnost raspodijeljena na velik broj nezavisnih usluga kao što su razmjena datoteka, video igre, *cloud* servisi, itd.



Slika 2.1. *Paketizacija i multipleksiranje komponenti DTV signala* [9, str. 82]

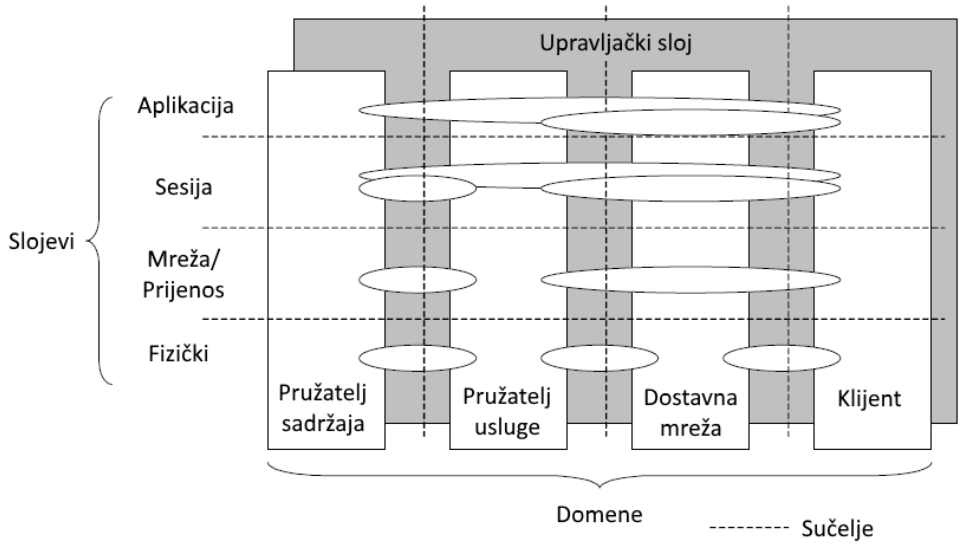
Digitalni TV sadržaj dostavlja se korisnicima kroz distribucijske mreže zasnovane na skupu Internet protokola definiranom u [10]. Skup protokola aplikacijskog sloja proširen je protokolima koji podržavaju razmjenu metapodataka i kontrolnih informacija vezanih uz usluge IP DTV poslužitelja.

2.1.2. Arhitektura IP DTV usluge

Na slici 2.2. prikazan je konceptualni model slojeva DVB IP usluga. Ovaj model općenito prikazuje domene svih IP DTV usluga te sučelja između njih. Uloge svake od ovih domena definirane su prema [11]:

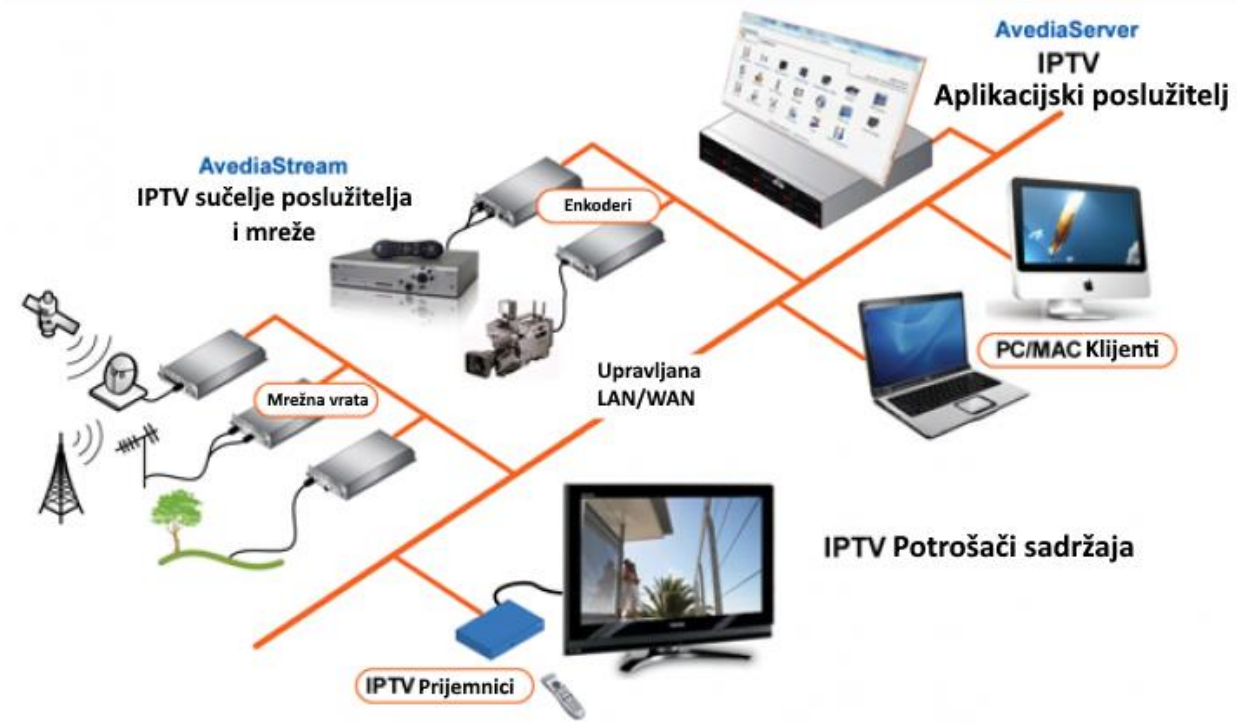
- Pružatelj sadržaja posjeduje ili je licenciran da prodaje televizijski sadržaj.
- Pružatelj usluge dobavlja sadržaj i pakira ga u usluge koje nudi korisniku.
- Distribucijska mreža povezuje korisnike i pružatelje usluge.
- Korisnik konzumira pružane sadržaje i usluge.

Dodatno su slikom 2.2. prikazana sučelja definiranih entiteta. S gledišta korisnika entiteti distribucijske mreže i pružatelja usluge potpuno su prozirni na aplikacijskom i sesijskom nivou. Korisnik se pretplaćuje i konzumira sadržaj pružatelja sadržaja. Na sesijskom nivou postoji sučelje korisnika i pružatelja usluge, budući da je korisnik pretplaćen na paket usluga, koji može uključivati sadržaj većeg broja proizvođača. Entitet distribucijske mreže transparentan je za korisnika na nivou. Za prijenos zahtijevanog sadržaja dovoljno je poznavanje krajnje točke korisnika i pružatelja usluge.



Slika 2.2. *Slojeviti model IP DTV usluga* [11, str. 32]

Na slici 2.3. prikazana je shema IPTV sustava s komponentama koje predstavljaju konkretizaciju entiteta prikazanih na slici 2.2. IPTV sučelje poslužitelja i mreže (engl. *head-end*) predstavlja spregu pružatelja sadržaja i pružatelja usluge prema IP mreži. Dodatna komponenta entiteta pružatelja usluge je i IPTV aplikacijski poslužitelj čija je svrha pružanje dodatnih usluga koje nisu izravno povezane s TV sadržajem. Entitet distribucijske mreže predstavljen je upravljanom LAN/WAB mrežom. U slučaju IPTV usluge ova komponenta je zatvorena mreža u vlasništvu ili zakupljena od strane pružatelja usluge. Zatvorena distribucijska mreža omogućuje alokaciju i kontrolu propusnog opsega, kao i garantiranje određenog stupnja kvalitete usluge. Zasebni IPTV prijemnici, kao i drugi klijentski uređaji (računala, Smart TV, mobilni uređaji) su komponente korisničkog entiteta.



Slika 2.3. *Shema IPTV sustava* [12]

Pružatelj usluge u velikom broju slučajeva od pružatelja sadržaja dobiva unaprijed snimljen i pripremljen sadržaj koji dostavlja pretplatnicima (izuzev u slučaju prijenosa „u živo“). Ova dva entiteta čine zatvoren sustav u kojem je mala vjerojatnost pojavljivanja grešaka te nemaju znatan utjecaj na kvalitetu usluge sadržaja. Dakle, domena distribucijske mreže najvažnija je s gledišta kvalitete usluge. Ovaj entitet najčešće čini skup pristupnih i jezgrenih mreža koje koriste velik broj različitih mrežnih tehnologija. Distribucijska mreža je entitet unutar kojega se u prijenosni tok mogu inducirati greške kašnjenja ili gubitka, relevantne za kvalitetu prenošenih audio i video sadržaja.

2.1.3. Standardi i protokoli u IP televiziji

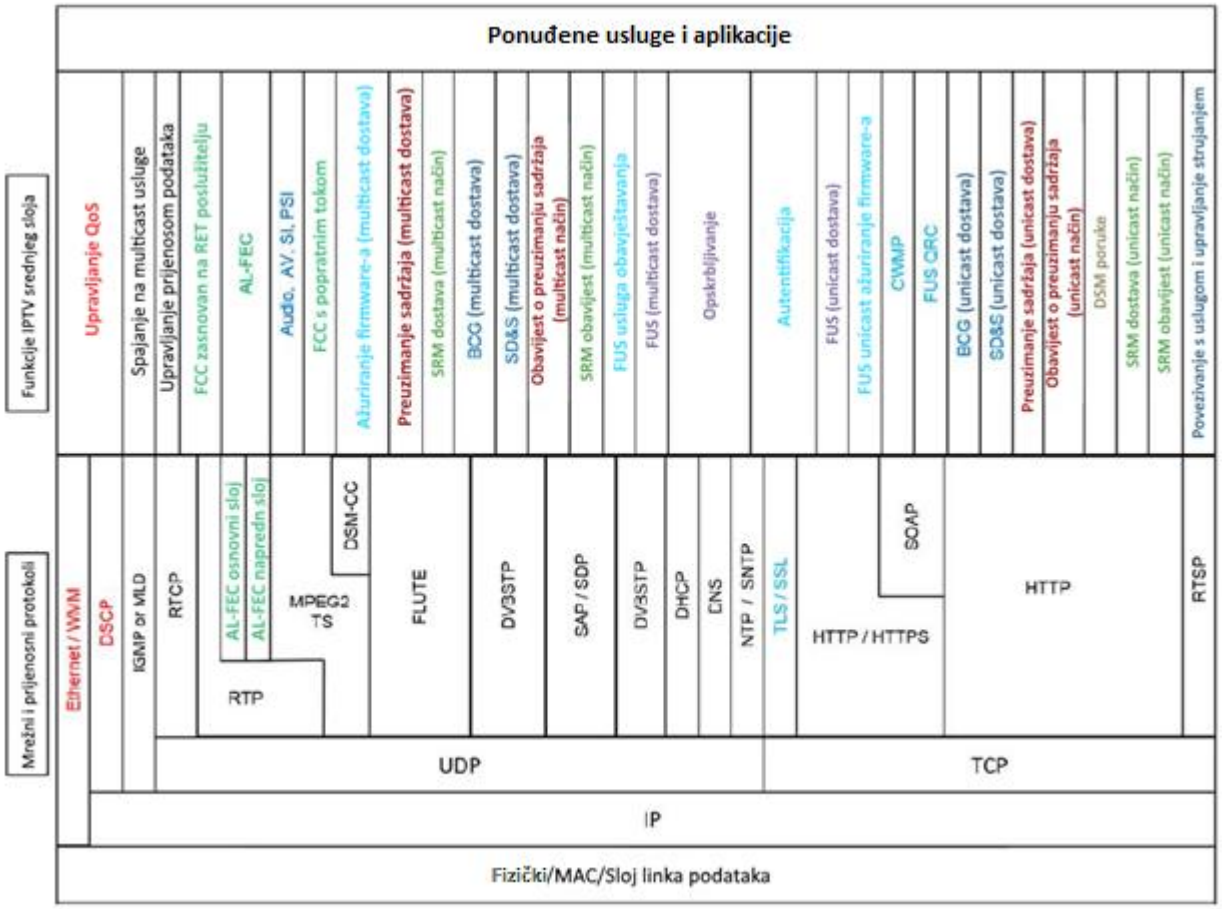
Kao što je već prikazano, domena distribucijske mreže nevidljiva je za klijenta na višim nivoima mrežnog modela. Međutim, na prijenosnom nivou klijentu, osim krajnje točke (adrese) poslužitelja, mora biti poznat komunikacijski protokol. Komunikacijski protokoli najčešće implementiraju algoritme oporavka od grešaka i zajedno sa sadržajem prenose metapodatke koje koriste ti algoritmi. Pomoću metapodataka koje pruža komunikacijski protokol mogu se identificirati greške nastale u domeni distribucijske mreže. Stoga je vrlo bitno identificirati koji komunikacijski protokoli su standardizirani u IP DTV uslugama. Na slici 2.4. prikazan je cjelokupni protokolarni stog definiran DVB-IPTV standardom. Različite funkcionalnosti koje čine DVB-IPTV uslugu raspoređene su prema protokolima na koje se oslanjaju. Bojama su povezane grupe funkcija koje čine pojedine segmente usluge [11, str. 36]:

- Upravljanje kvalitetom usluge – crvena
- Upravljanje *multicast* vezama i grupama – crna
- Ispravak grešaka/mehanizam brze promjene kanala – svijetlo zelena
- AV sadržaj i metapodaci – tamno plava
- Udaljeno upravljanje i ažuriranje *firmware-a* - svijetlo plava
- Usluga preuzimanja sadržaja – tamno crvena

Segment usluge najvažniji za korisnika je odabir i dostavljanje TV sadržaja. Stoga je u sklopu ovog rada istražena mogućnost mjerenja kvalitete usluge onih protokola prijenosnog nivoa na kojima su zasnovane ove usluge. Dokument [13] definira tri različita profila za usluge audio i video sadržaja. Ovi profili specificirani su prema grupama protokola koje koriste, a specifikacija profila dana je u tablici 2.1.

Interes ovog rada usmjeren je na QoS reprodukcije u realnom vremenu, odnosno strujanja (engl. *streaming*) TV sadržaja. Profili reprodukcije sadržaja u živo i na zahtjev definiraju enkapsulaciju MPEG prijenosnog toka u RTP protokol, odnosno UDP *datagrame* na prijenosnom nivou. UDP je dizajniran kao prijenosni protokol s vrlo niskom razinom komunikacijskog troška. Stoga ne pruža nikakve informacije o stanju prijenosne mreže između krajnjih točaka komunikacije. RTP kao protokol za prijenos i reprodukciju sadržaja u realnom vremenu pruža informacije o QoS metrikama tog sadržaja. Profil preuzimanja sadržaja specificiran je za CDS (engl. *Content Download Service*) uslugu definiranu u [14], koja ne predviđa reprodukciju sadržaja u realnom vremenu. Stoga u ovom radu nije istražena mogućnost dobivanja metrika QoS iz FLUTE (engl. *File Delivery over Unidirectional Transport*) protokola. U sklopu ovog rada također

je podržano mjerenje QoS za IP prijenos sadržaja kod hibridnih uređaja. Prema HbbTV specifikaciji [2, str. 65,66], strujanje sadržaja na zahtjev, kao i preuzimanje sadržaja odvija se preko HTTP protokola. HTTP je tekstualni protokol aplikacijskog nivoa, te sam ne pruža nikakve informacije o razini QoS. Međutim, sve HTTP poruke enkapsulirane su u TCP segmente na prijenosnom nivou.



Slika 2.4. *Protokoli DVB-IPTV standarda* [11, str. 37]

Tablica 2.1. *Specifikacije DVB-IPTV profila* [13]

Profil	**Modul**		
	Prijenos	Veza	Otkrivanje usluge
Reprodukcija sadržaja u živo	UDP, RTP/UDP	IGMP (engl. *Internet Group Management Protocol*), MLD (engl. *Multicast listener discovery*)	SD&S (engl. *Service Discovery and Selection*)
Reprodukcija sadržaja na zahtjev	UDP, RTP/UDP	IGMP, MLD	SD&S, BCG (engl. *Broadband Content Guide*)
Preuzimanje sadržaja	HTTP, FLUTE	HTTP, IGMP, MLD	SD&S, BCG

Iz razmotrenih standarda i specifikacija zaključeno je da će u sklopu ovog rada biti istražene mogućnosti dobivanja QoS metrika iz sljedećih komunikacijskih protokola:

- RTP/UDP – protokol na kojem je zasnovan prijenos A/V (audio/video) sadržaja kod IPTV sustava prema DVB-IPTV standardu;
- TCP – prijenosni protokol za HTTP strujanje A/V sadržaja u hibridnim sustavima definiran prema HbbTV standardu.

2.2. Specifikacije protokola

U funkcije mrežnih komunikacijskih protokola između ostalih spadaju: identifikacija krajnjih točaka komunikacije, uspostavljanje komunikacijskog kanala, prijenos podataka bez grešaka i upravljanje zagušenjem mreže. U IPTV i hibridnim DTV sustavima IP se koristi kao robustan i globalno podržan protokol za adresiranje mrežnih uređaja i pronalaženje komunikacijskih puteva. Protokoli koji se prema TCP/IP mrežnom modelu nalaze ispod IP-a nisu specificirani standardima. IP DTV sustavi funkcioniraju nezavisno od njih. Protokoli prijenosnog sloja implementiraju mehanizme upravljanja zagušenjem mreže i oporavka od grešaka. Protokoli aplikacijskog sloja koriste se za prijenos metapodataka i upravljanje korisničkim sesijama.

Upravo mehanizmi identifikacije grešaka i zagušenja mreže kod prijenosnih protokola iskorišteni su u sklopu ovog rada za mjerenje parametara QoS. Osnovna jedinica podataka nekog protokola je PDU (engl. *Protocol Data Unit*). PDU čine zaglavlje, koje nosi metapodatke samog protokola, i prenošeni podaci. Upravo analizom zaglavlja TCP i RTP protokola nastoje se dobiti što preciznije QoS metrike za promatranu komunikaciju, pri čemu se analizom IP zaglavlja utvrđuju krajnje točke i smjer komunikacije.

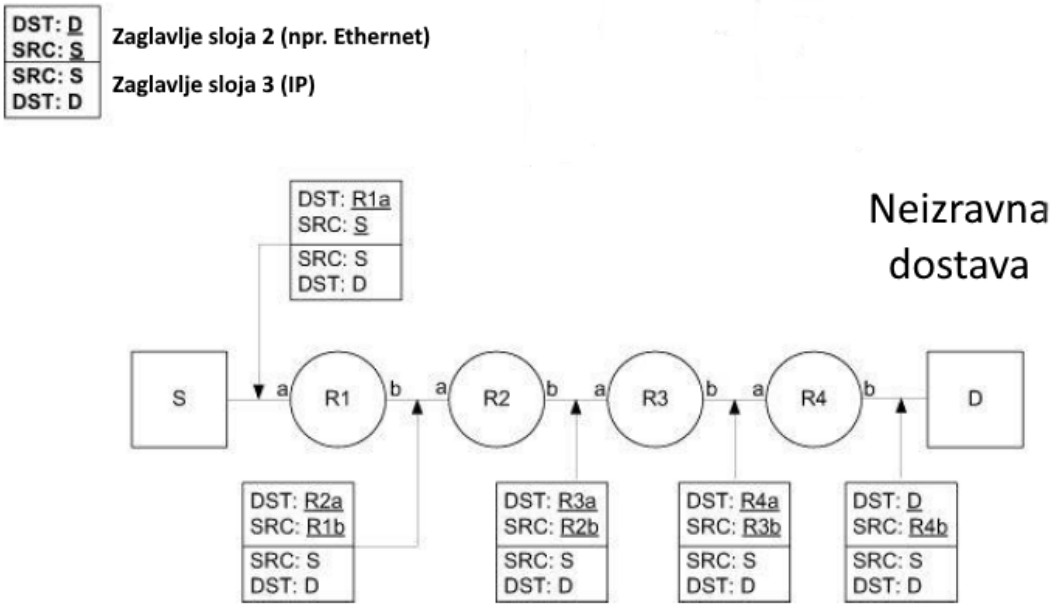
2.2.1. Internet Protocol i Internet Control Message Protocol

IP pruža negarantiran, bezkonekcijski prijenos *datagrama*. Negarantiran znači da IP ne može jamčiti uspješnu dostavu poslanog *datagrama* na njegovo odredište. Politika upravljanja greškama i zagušenjem IP-a je odbacivanje podataka ako su nepravilni, ili ako je međuspremnik mrežnog uređaja pun. Bezkonekcijski znači da IP ne održava podatke o stanju kon-ekcije unutar mrežnih čvorova, već se svaki IP *datagram* usmjerava kroz mrežu neovisno o svim ostalima. Ovo znači da *datagrami* uzastopno poslani na isto odredište mogu biti usmjereni kroz različite puteve, te na odredište stići u krivom redoslijedu. Također je moguća duplikacija *datagrama* na putu do odredišta [15, str. 196].

Osnovna funkcija IP-a je adresiranje mrežnih čvorova i usmjeravanje podataka kroz put sačinjen od niza mrežnih elemenata koji povezuju izvor i odredište. Postupak usmjeravanja opisan u [16] odvija se na sljedeći način:

1. IP modul izvora dobiva podatke za slanje, adresu odredišta i ostale potrebne parametre. Zatim ih enkapsulira u IP *datagram* i prosljeđuje ih sučelju lokalne mreže. Sučelje lokalne mreže prosljeđuje ovaj *datagram* uređaju koji djeluje kao pristupna točka lokalne mreže.
2. Pristupna točka, na osnovu IP adrese odredišta određuje kojem će lokalnom mrežnom sučelju proslijediti *datagram*
3. Svaki usmjerivač koji primi *datagram* donosi odluku iz točke 2. sve dok *datagram* ne stigne na lokalno mrežno sučelje odredišnog uređaja.

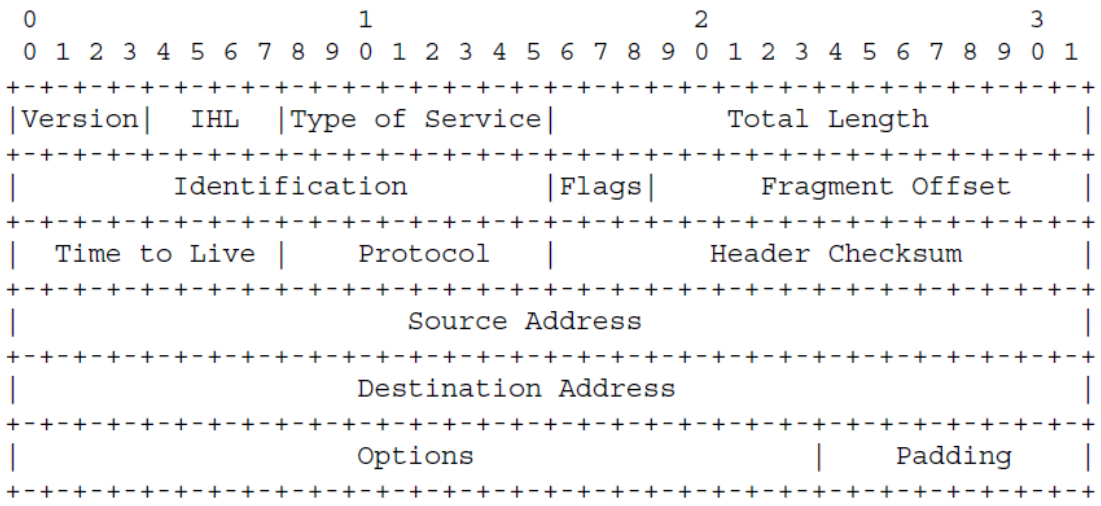
Opisani postupak prikazan je a slici 2.5. Odluke o prosljeđivanju paketa usmjerivači donose na osnovu tablica usmjeravanja mrežnih čvorova. Ova tablica za svako poznato odredište sadrži adresu sljedećeg mrežnog čvora na najkraćem putu, te mrežno sučelje na koje treba proslijediti *datagram*.



Slika 2.5. *Ilustracija IP usmjeravanja* [15, str. 223]

Na slici 2.6. prikazano je zaglavlje IP *datagrama*. Polje *Version* označava verziju protokola: IPv4 ili IPv6. U sklopu ovog rada razmotren je samo IPv4, budući da IPv6 još uvijek nije u potpunosti globalno podržan. *Type of Service* polje u novije vrijeme naziva se DS (engl. *Differential Services*) i predstavlja prioritet *datagrama*, odnosno zahtijevanu razinu QoS. IHL

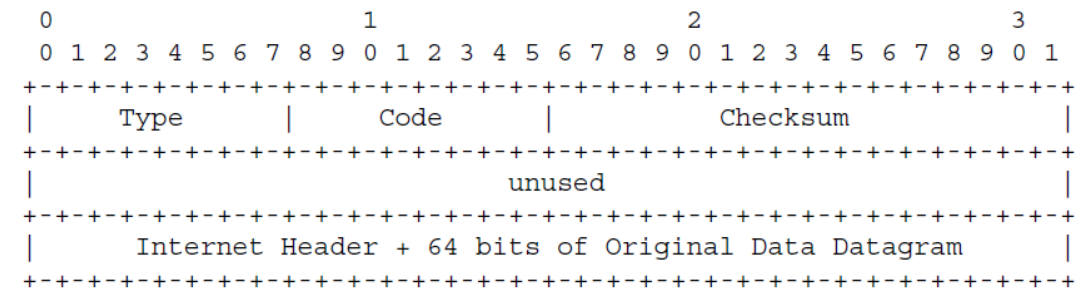
(engl. *Internet Header Length*) označava broj 32-bitnih riječi IP zaglavlja, dok *Total Length* označava cjelokupan broj okteta IP *datagrama*, uključujući zaglavlje. *Identification* i *Fragment offset* polja podržavaju fragmentaciju IP *datagrama*. *Time to live* polje označava maksimalan broj čvorova kroz koje *datagram* može biti usmjeren prije odbacivanja.



Slika 2.6. *Zaglavlje IP datagrama* [16]

Protocol polje označava enkapsulirani protokol, najčešće TCP, UDP, ICMP (engl. *Internet Control Message Protocol*) ili IGMP. *Header checksum* je kontrolna riječ koja označava validnost svih polja IP zaglavlja. Najveći značaj imaju dvije 32-bitne riječi koje označavaju IP adresu izvora, odnosno odredišta. IP opcije ne koriste se u normalnoj IP komunikaciji i nisu pokrivene ovim radom.

Popratni protokol IP-a namijenjen razmjeni kontrolnih poruka i izvještavanju o greškama je ICMP [17]. Kontrolne poruke i izvještaji ICMP-a enkapsulirani su u IP *datagramima*. Zaglavlje ICMP poruke prikazano je na slici 2.7. Poljima *Type* i *Code* definirana je semantika ICMP poruke. *Checksum* je kontrolna suma koja pokriva polja ICMP zaglavlja.



Slika 2.7. *Zaglavlje ICMP poruke* [17]

U sklopu ovog rada ICMP je korišten za provjeru dostupnosti uređaja na ciljanoj IP adresi i mjerenje vremenskog kašnjenja između dviju točkaka komunikacije. Tipovi poruka korištenih u ovu svrhu prikazani su tablicom 2.2.

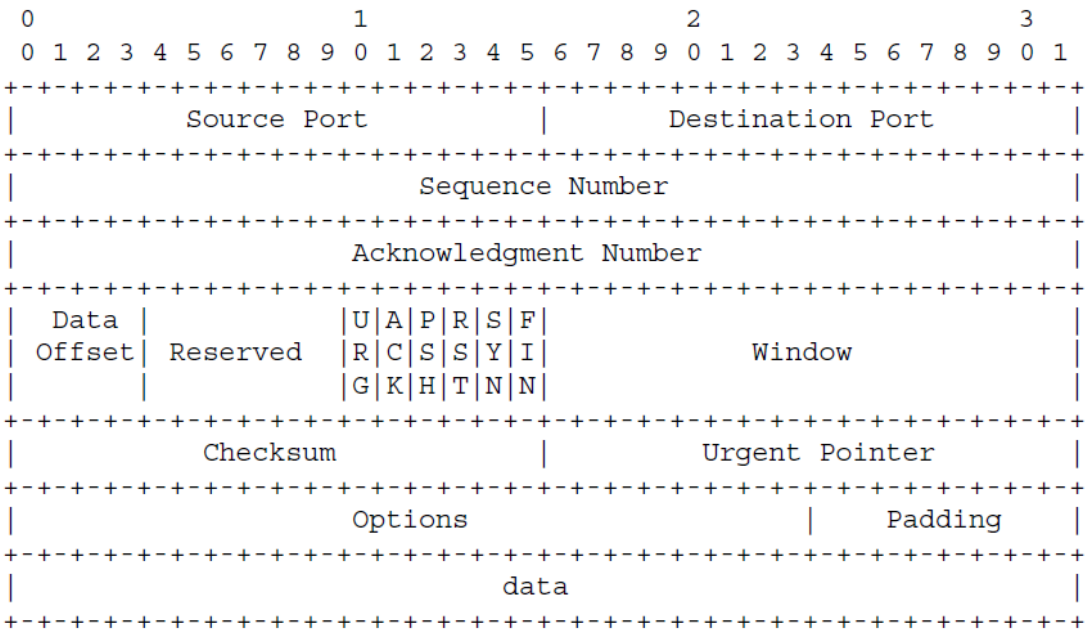
Tablica 2.2. *Tipovi ICMP poruka* [15, str. 325]

Tip	Ime	Opis
0	*Echo Reply*	Odgovor na *echo* zahtjev, sadrži kopiju podataka poslanih u zahtjevu.
3	*Destination Unreachable*	Ciljana mreža, uređaj ili protokol nije dostupan.
8	*Echo*	*Echo* zahtjev, može sadržavati podatke.

2.2.2. Transmission Control Protocol

TCP pruža konekcijski, pouzdan prijenos toka podataka. TCP je konekcijski orijentiran protokol, što znači da krajnje točke, prije razmjene podataka, postupkom rukovanja (engl. *handshake*) iniciraju međusobnu konekciju i tijekom razmjene održavaju stanje te konekcije dok ona ne bude prekinuta (također rukovanjem). Pouzdani prijenos znači da TCP implementira mehanizam automatske retransmisije podataka za koje se pretpostavlja da su izgubljeni. Na posljetku, TCP je protokol koji rukuje tokom podataka, tj. svi segmenti (paketi) iste konekcije čine kontinuiran tok podataka. Unutar tog toka TCP ne razlikuje granice pojedinačnih poruka koje šalju protokoli viših instanci [15, str. 561].

Na slici 2.8. prikazano je zaglavlje TCP segmenta. Izvorišni i odredišni *port* su 16-bitne riječi koje jednoznačno identificiraju aplikacije koje sudjeluju u TCP konekciji.



Slika 8. *Zaglavlje TCP segmenta* [18]

Sequence number polje (SEQ) je jedinstveni identifikator TCP segmenta. Tijekom uspostave konekcije, obje strane nasumično biraju vlastite početne brojeve sekvence. Za svaki poslani oktet u TCP toku broj sekvence se uvećava za jedan. Tako broj sekvence poslanog segmenta označava broj prvog poslanog okteta podataka. *Acknowledgement number* polje (ACK) koristi se za potvrdu prijema podataka. Prijemna strana šalje potvrdu prijema pošiljatelju, gdje ovo polje ima vrijednost sekvence zadnjeg uspješno primljenog okteta podataka uvećanu za jedan. Ova vrijednost naznačuje pošiljatelju koju sljedeću sekvencu očekuje primatelj.

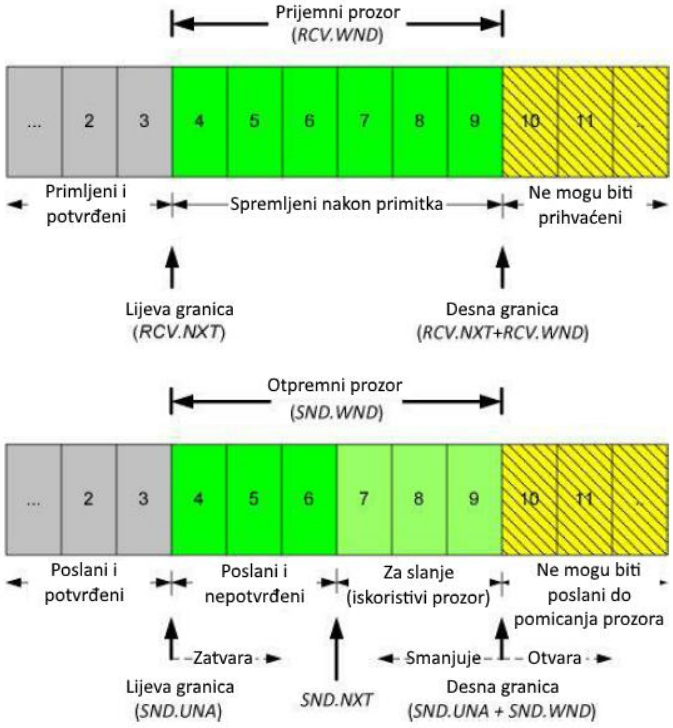
Data offset polje označava broj 32-bitnih riječi u TCP zaglavlju. Nakon šest bitova rezerviranog polja nalaze se zastavice TCP zaglavlja. Značenja zastavica definirano u [18] je:

- URG – polje *Urgent pointer* je validno.
- ACK – polje ACK je validno (uvijek postavljena nakon uspostavljanja konekcije)
- PSH – *Push* funkcija (naznačuje prijemnom TCP modulu da prijemni međuspremnik treba što prije proslijediti korisniku)
- RST – resetiraj (prekini) konekciju
- SYN – sinkronizraj brojeve sekvence (koristi se kod inicijalizacije konekcije
- FIN – pošiljatelj nema više podataka (koristi se pri zatvaranju konekcije)

Window polje označava maksimalni broj okteta koje je prijemnik spreman prihvatiti. Paketi sa sekvencom većom od zadnjeg poslanog ACK-a uvećanog za *Window* bit će odbačeni od strane prijemnika. *Checksum* polje je kontrolna suma kojom se potvrđuje validnost TCP zaglavlja. Kontrolna suma, također pokriva pseudo-zaglavlje koje se sastoji iz izvorišne i odredišne adrese i oznake protokola IP zaglavlja, te ukupne dužine TCP segmenta. *Urgent pointer* polje predstavlja pomak od broja sekvence segmenta do prvog okteta iza podataka koji su označeni kao hitni.

Zaglavlje prate TCP opcije varijabilne dužine. TCP segment može nositi više opcija. Svaka opcija sastoji se iz okteta koji označava tip opcije, ili okteta tipa praćenog oktetom ukupne dužine opcije i podacima opcije. Ovim radom pokrivena je opcija *Selective Acknowledgement* (SACK) korištena u TCP mehanizmu retransmisije.

TCP protokol pomoću SEQ i ACK brojeva implementira mehanizam automatske retransmisije. Temeljno načelo ovog mehanizma nalaže da prijemnik treba ACK-om potvrditi sve uspješno primljene pakete u ispravnom redoslijedu. Ovo načelo usko je povezano uz koncept klizećeg prozora (engl. *sliding window*) implementiran *Window* poljem. Koncept klizećeg prozora gledan s prijemne i otpremne strane prikazan je na slici 2.9.



Slika 2.9. *Prijemni i otpremni TCP klizeći prozor* [15, str. 664,665]

TCP tok sastoji se od kontinuiranog niza segmenata. Ovaj tok prijemnik i pošiljatelj „vide“ kroz prozor veličine *Window* okteta. Lijeva granica prozora je posljednja potvrđena sekvenca, desna granica prozora ovisi o veličini *Window* polja i naznačuje maksimalnu sekvencu koju prijemnik može primiti. Slanjem novog ACK-a prijemnik pomiče lijevu granicu prozora, tj. prozor „klizi“ naprijed u TCP toku.

Mehanizam retransmisije sastoji se u tome da pošiljatelj privremeno čuva poslane segmente dok ne primi ACK koji pomiče prozor prema naprijed. Pošiljatelj procjenjuje RTO (engl. *Retransmission timeout*), maksimalno vrijeme čekanja na prijem ACK segmenta. Ukoliko ACK ne stigne unutar RTO intervala, pošiljatelj ponovno šalje najstariji sačuvani segment, poveća RTO interval, te ponovo čeka ACK. Pogrešno procijenjen RTO može uzrokovati slanje duplikata ako istekne prije nego pošiljatelj primi ACK, ili predugo čekanje na retransmisiju ako ACK ne dolazi [15, str. 615].

Ovaj algoritam može uzrokovati duga čekanja u okolini s većim brojem izgubljenih segmenata. Stoga je definiran algoritam brze retransmisije (engl. *Fast retransmit*) opisan u [15, str. 631]. Prijemnik ima mogućnost slanja dupliciranih ACK-ova. Duplicirani ACK označava zadnji primljeni segment u ispravnom redoslijedu. Prijemnik šalje duplicirani ACK ako primi SEQ koji je unutar prozora, ali ne odgovara očekivanom, što može biti uzrokovano izgubljenim

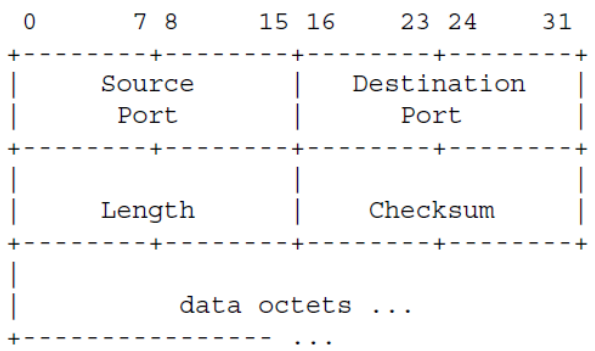
segmentom ili promjenom redoslijeda kojim segmenti stižu na odredište. Postoji minimalni broj dupliciranih ACK-ova koje pošiljatelj mora primiti prije retransmisije podataka (tzv. *DupThresh*). Ako pošiljatelj primi *DupThresh* dupliciranih ACK-ova prije isteka RTO intervala, on odgovara prijevremenom retransmisijom segmenta čiji je SEQ jednak dupliciranom ACK-u. Dodatno, korištenjem SACK opcije prijemnik može naznačiti pošiljatelju koji su točno segmenti uspješno primljeni. Ova opcija sadrži granice (najniži i najviši SEQ broj) uspješno primljenih blokova podataka. „Rupe“ između ovih blokova označavaju izgubljene pakete. Tako pošiljatelj može identificirati i retransmitirati više izgubljenih paketa odjednom.

2.2.3. User Datagram Protocol i Realtime Transport Protocol

UDP je jednostavan, *datagram*-orijentiran protokol. UDP protokol održava granice poruka viših protokola. Svaki UDP *datagram* sadržava kompletnu poruku. UDP protokol ne implementira mehanizme oporavka od grešaka i upravljanja zagušenjem. Ovaj protokol dizajniran je kako bi aplikacijama pružio minimalnu razinu kompleksnosti i minimalnu količinu kontrolnih podataka [15, str. 459].

Preferirani način distribucije sadržaja IPTV usluge je *multicast* (istovremeno razašiljanje sadržaja grupi korisnika). Prijenosni tok sa sadržajem nekog programa dostavlja se odjednom grupi korisnika koji su izrazili interes. UDP protokol pogodan je za *multicast* komunikaciju jer ne zahtjeva uspostavljanje konekcije između prijemnika i pošiljatelja. U slučaju TCP-a pošiljatelj bi morao održavati posebnu konekciju za svakog prijemnika, dok UDP nema ove zahtjeve što ga čini puno fleksibilnijim.

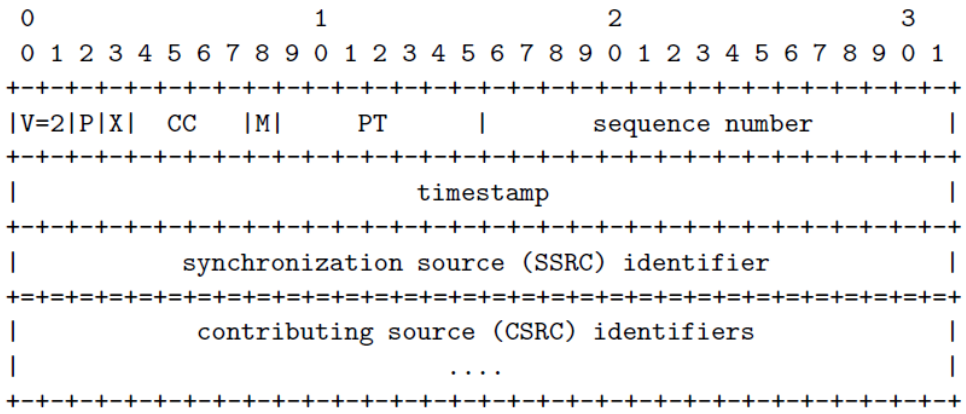
Veličina UDP zaglavlja je uvijek osam okteta i njegova struktura prikazana je na slici 2.10. Izvorišni i odredišni *port* jedinstveno označavaju aplikacije koje razmjenjuju UDP *datagrame*. *Length* polje označava ukupnu dužinu *datagrama*. *Checksum* polje je kontrolna suma koja pokriva polja UDP zaglavlja, kao i pseudo-zaglavlje opisano kod TCP protokola.



Slika 2.10. *Zaglavlje UDP datagrama* [19]

Aplikacije koje koriste UDP protokol moraju samostalno implementirati mehanizme upravljanja greškama. DVB-IPTV specifikacija definira da MPEG-2 prijenosni tok može biti enkapsuliran direktno u UDP ako distribucijska mreža može garantirati dostavljanje *datagrama* na odredište u ispravnom redoslijedu. U suprotnom potrebno je enkapsulirati prijenosni tok u RTP pakete koji se prenose unutar UDP *datagrama* [11, str. 142].

RTP protokol pruža funkcionalnost dostavljanja s kraja na kraj podataka u realnom vremenu kao što su audio i video sadržaji. Ova funkcionalnost uključuje identificiranje tipa podataka, brojanje sekvenci i vremensko označavanje podataka. RTP podržava prijenos podataka na više odredišta *multicast* distribucijom [20].



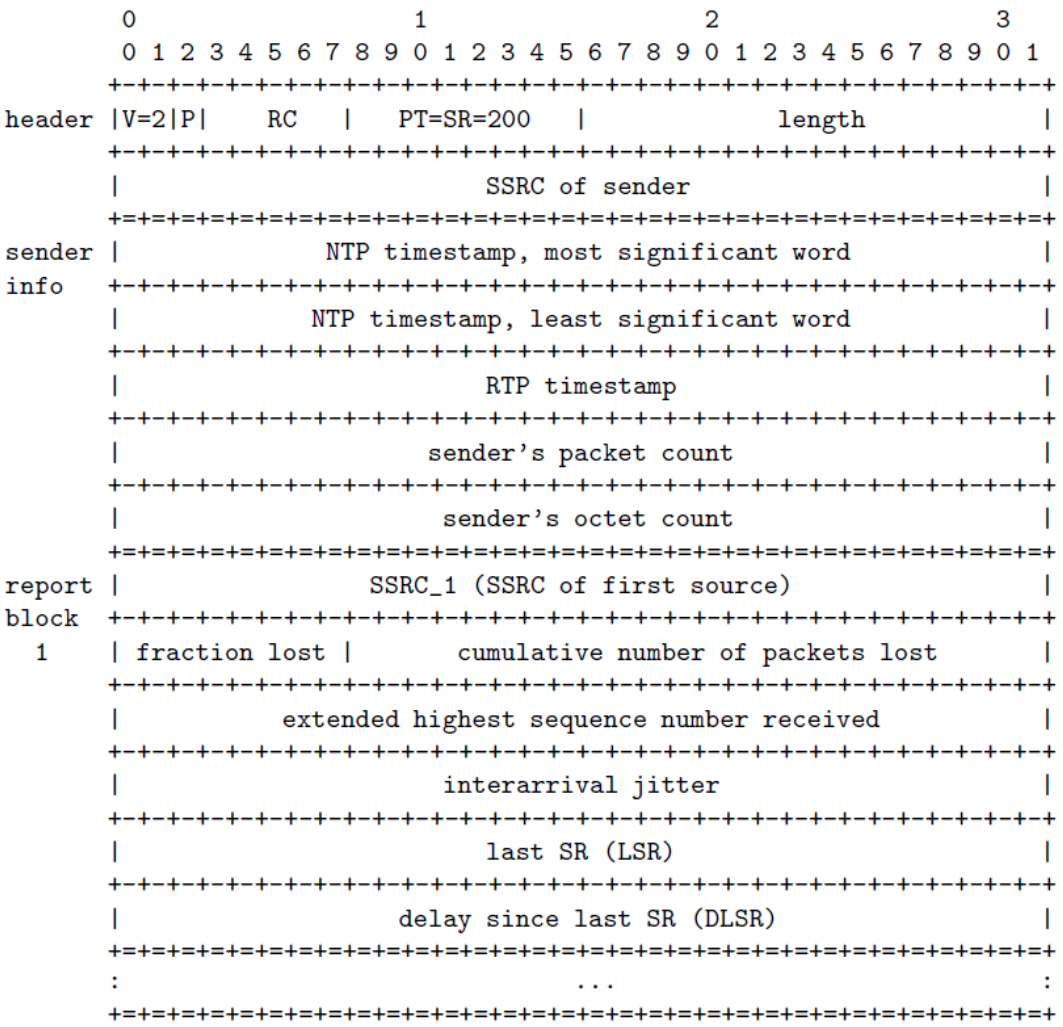
Slika 2.11. *Zaglavlje RTP paketa* [20]

Version (V) polje označava verziju RTP-a, trenutna vrijednost je 2. *Padding* (P) bit označava postojanje *padding* okteta nakon RTP podataka, broj ovih okteta naznačen je zadnjim oktetom. *Extension* (X) bit označava postojanje zaglavlja proširenja iza RTP zaglavlja, prije samih podataka. *CSRC Count* (CC) označava broj 32-bitnih *Contributing source* polja. *Sequence number* polje povećava se za jedan za svaki poslani RTP paket. Prijemnik može koristiti ovo polje za detekciju gubitka ili duplikacije podataka, te uspostavljanje pravilnog redoslijeda paketa. Početna vrijednost ovog polja treba biti nasumično odabrana. *Synchronization source ideentifier* (SSRC) označava izvor toka RTP paketa koji maju zajedničke domene sekvenci i vremenskih oznaka. Unutar RTP medijske sesije dopušteno je slanje više različitih tokova podataka s odvojenim sadržajem. RTP tokovi odvajaju se na prijemnoj strani na osnovu SSRC vrijednosti [20].

Značenje ostalih polja u RTP zaglavlju u DV-IPTV usluzi definirano je u [21] i [11, str. 140]. *Marked* (M) bit je postavljen na 1 kad god vremenska oznaka RTP paketa nije u kontinuitetu s vremenskim oznakama prethodno poslanih paketa, što može biti uzrokovano promjenom izvora podataka na otpremnoj strani. *Payload type* (PT) polje ima vrijednost 33 koja označava MPEG-2

prijenosni tok [22]. *Timestamp* polje je 32-bitna vremenska oznaka frekvencije 90 kHz koja označava ciljano vrijeme transmisije paketa. *Contributing source identifier* (CSRC) polja označavaju izvore svih RTP tokova koji su mikserom spojeni u trenutni RTP tok. Ova polja su opcionalna, a DVB-IPTV specifikacija nalaže da trebaju biti zanemarena.

RTP specifikacija također definira RTCP (engl. *Realtime Transport Control Protocol*). Primarna funkcija ovog protokola je dostavljanje povratnih informacija o kvaliteti distribucije podataka pošiljateljima i prijemnicima [20]. U tu svrhu RTCP definira dva tipa paketa: *Sender Report* (SR) i *Receiver Report* (RR). Sudionici RTP konekcije koji šalju podatke u pravilnim vremenskim intervalima generiraju SR pakete, dok prijemnici generiraju RR pakete. Struktura SR paketa prikazana su na slici 2.12.



Slika 2.12. *RTCP SR paket*[20]

V polje i *P* bit imaju isto značenje kao kod RTP zaglavlja. *Reception report count* (RC) polje označava broj izvještaja o prijemu u SR paketu. SR paket može sadržavati ove izvještaje,

RTP medijske sesije su dvosmjerne pa svaki pošiljatelj istovremeno može biti i primatelj. *PT* polje označava tip RTCP paketa, njegova vrijednost je 200 za SR ili 201 za RR. *Length* označava veličinu RTCP paketa iskazanu brojem 32-bitnih riječi umanjenim za jedan. SSRC je identifikator pošiljatelja SR paketa. Sljedećih pet polja SR-a čine *sender info* blok. Prva dva polja ovog bloka predstavljaju 64-bitnu NTP (engl. *Network time protocol*) vremensku oznaku. Sljedeće polje je RTP vremenska oznaka generirana u istom trenutku kao i NTP oznaka. Vremenske oznake u *sender info* bloku omogućuju prijemniku da uspostavi vezu između vremenskih oznaka RTP paketa i stvarnog vremena, što omogućuje sinkronizaciju RTP tokova iz različitih izvora (npr. slika u slici). Zadnja dva polja ovog bloka pokazuju koliko je ukupno RTP paketa, odnosno okteta podataka poslano od početka transmisije. Ostatak SR paketa čine *report* blokovi. Oni se odašilju u situacijama kada je RTP komunikacija dvosmjerna, te pošiljatelj prima sadržaj i izvještava o primitku. Ovi slučajevi korištenja ne spadaju u domenu IP DTV-a. RTCP RR paketi imaju identičnu strukturu kao i SR paketi, s tim da ne sadrže *sender info* blokove.

DVB-IPTV specifikacijom definirano je da distributeri sadržaja trebaju generirati SR pakete, međutim, ovi paketi ne bi trebali sadržavati *receprion report* blokove. Interval slanja SR paketa ne smije biti veći od 10 s. Korisnički uređaji trebaju parsirati SR pakete. Njihova funkcija je sinkronizacija RTP tokova iz različitih izvora. Zbog skalabilnosti definirano je da korisnički uređaji neće generirati RR pakete, osim kod IPTV sustava gdje se pomoću povratnih RR paketa implementira mehanizam retransmisije[11, str. 140].

2.3. Predložene metrike za QoS

Osnovni parametri QoS svakog oblika mrežne komunikacije su širina pojasa, odnosno propusnost mreže (engl. *bandwidth*), komunikacijsko kašnjenje (engl. *latency*, *ping*) i varijacija kašnjenja (engl. *jitter*). IP protokol ne definira mehanizme garantirane i uređene dostave *datagrama*. Posljedice toga su mogući gubitak paketa, izmjena redosljeda kojim su paketi poslani i pojava duplikata unutar distribucijske mreže. Prijenosni protokoli opisani u prethodnom poglavlju sekvenciranjem paketa mogu ispraviti greške duplikacije i krivog redoslijeda. TCP implementira i mehanizam oporavka od izgubljenih paketa zasnovan na čekanju i retransmisiji, što uzrokuje dodatna kašnjenja u mreži. Uobičajeni slučajevi korištenja IPTV usluga opisani tablicom 2.1. podrazumijevaju strujanje DTV sadržaja u realnom vremenu s mogućnošću interakcije korisnika i usluge, uspostavljanjem povratne veze. Navedeni parametri QoS imaju različit utjecaj na kvalitetu sadržaja koji je dostavljen korisniku.

Propusnost mreže može predstavljati binarni prag za IP DTV uslugu. Budući da DTV sadržaji imaju konstantnu bitsku brzinu (engl. *bit-rate*) i trebaju biti reproducirani u realnom vremenu, nedostatak odgovarajuće propusnosti u potpunosti sprječava reprodukciju. Protokoli adaptivnog strujanja, poput MPEG-DASH [23] protokola nastoje ublažiti stroge zahtjeve na propusnost uvođenjem višestrukih instanci istog sadržaja različitih razina kvalitete koje korisnik može odabrati ovisno o dostupnoj propusnosti.

Komunikacijsko kašnjenje ima najviše utjecaja na povratnu vezu korisnika i DTV usluge. Pojava primjetnog kašnjenja može otežati osnovne slučajeve korištenja DTV usluge, poput inicijalizacije usluge, promjene programa itd. DTV-IPTV specifikacija preporučuje zahtjev na maksimalno kašnjenje kod priključivanja, odnosno napuštanja *multicast* tokova od 500 ms [11, str.143]. Varijacija kašnjenja može imati znatan utjecaj na kvalitetu sadržaja. Prijemni uređaj pohranjuje dostavljeni sadržaj u međuspremnike iz kojih se sadržaj dohvaća i dekodira u konstantnim intervalima. Varijacija u kašnjenju može uzrokovati dva tipa grešaka: ukoliko je kašnjenje povećano, prijemni međuspremnik neće biti popunjen što znači da nema dovoljno podataka za ispravno dekodiranje, te ukoliko je kašnjenje smanjeno, prijemni međuspremnik postaje popunjen pa se podaci pristigli prije oslobađanja memorije odbacuju. DVB-IPTV specifikacija postavlja strog zahtjev na maksimalni raspon varijacije kašnjenja od 40 ms [11, str.142].

Pojava izgubljenih paketa u slučaju kada ne postoje mehanizmi retransmisije i ispravka grešaka direktno utječe na kvalitetu DTV sadržaja. Izgubljeni paketi manifestiraju se pojavom artefakata pri reprodukciji koji su posljedica nedostatka podataka u postupku dekodiranja. Problem gubitka paketa uvećan je činjenicom da jedan IP *datagram* sadrži više paketa MPEG-2 prijenosnog toka kako bi se postigla bolja efikasnost prijenosa. Kako bi se izbjegla fragmentacija Ethernet MTU-a (engl. *Maximum transmission unit*) od 1500 okteta, IP *datagram* s UDP i RTP zaglavljima ne smije sadržavati više od sedam MPEG-2TS paketa dužine 188 okteta (maksimalna ukupna dužina *datagrama* 1356 okteta) [11, str. 139]. Kod HTTP strujanja TCP protokol obavlja retransmisiju izgubljenih paketa, međutim ovaj mehanizam unosi dodatno kašnjenje u mrežu, što je nepovoljno za reprodukciju u stvarnom vremenu. DVB-IPTV specifikacija preporučuje zahtjev na kvalitetu sadržaja: maksimalno jedan primjetni artefakt po satu. Broj IP grešaka koji odgovara ovom zahtjevu ovisi o prijenosnoj brzini. Za brzinu prijenosa 4 Mb/s, uz uvjet da IP *datagram* sadrži sedam MPEG-2TS paketa, ovaj broj iznosi 10^{-6} IP grešaka po sekundi [11, str. 142].

Duplicirani paketi i paketi u krivom redoslijedu imaju manji utjecaj na QoS. RTP i TCP označavaju pakete brojevima sekvence koji monotono rastu. Tako je vrlo lako obnoviti ispravan

redoslijed paketa i prepoznati duplikate. Međutim, učestala pojava ovih grešaka može značajno ograničiti kvalitetu sadržaja. Pojava velikog broja dupliciranih paketa ograničava propusnost mreže za normalni promet. Izmjena redoslijeda paketa zahtjeva duži period pohrane podataka u međuspremnike. Dodatno, kod TCP komunikacije, krivi redoslijed može biti interpretiran kao gubitak i uzrokovati neželjene retransmisije i duplikate.

Tablica 2.3. prikazuje metrike QoS pokrivene ovim radom, kao i način njihovog dobivanja analizom mrežnog prometa klijentskog uređaja.

Tablica 2.3. *QoS metrike*

Br.	Metrika	Opis
1.	Provjera dostupnosti IP adrese pomoću ICMP protokola	Slanje *Echo* poruke. Čekanje na odgovor, *host* je dostupan ako je odgovor *Echo reply*, nedostupan ako je odgovor *Destination unreachable*, ili ako istekne *timeout* interval.
2.	Provjera dostupnosti TCP porta na IP adresi pomoću TCP protokola	Slanje TCP SYN paketa. Port je dostupan ako pristigne SYN+ACK, nedostupan ako istekne *timeout* interval.
3.	Mjerenje kašnjenja, varijacije kašnjenja pomoću ICMP protokola	Slanje niza *Echo* poruka. Mjerenje vremenskih intervala u kojima stižu odgovori.
4.	Mjerenje kašnjenja, varijacije kašnjenja pomoću TCP protokola	Slanje niza TCP SYN paketa. Mjerenje vremenskih intervala u kojima stižu odgovori.
5.	Mjerenje količine prometa, tj. iskorištene propusnosti (primljeni, poslani i ukupni promet)	Praćenje koliko je paketa primljeno i poslano u periodu od jedne sekunde. Paketi mogu biti filtrirani samo prema IP adresama ili prema IP adresama i *port-ovima*.
6.	Broj izgubljenih TCP paketa, odnosno okteta	Klijent ili server javljaju izgubljene pakete slanjem dupliciranih ACK sekvenci. Iz ovih sekvenci računa se broj izgubljenih okteta, odnosno paketa.
7.	Broj TCP paketa u krivom redoslijedu	Paketi čija se sekvenca razlikuje od sljedeće očekivane.
8.	Broj dupliciranih TCP paketa	Paketi čija je sekvenca već uspješno primljena.
9.	Mjerenje kašnjenja, varijacije kašnjenja TCP konekcije	Mjerenje vremenskih intervala između poslanog paketa i ACK-a za taj paket.
10.	Broj izgubljenih RTP paketa	Provjera jesu li sekvence paketa u RTP toku preskočene i koji je broj preskočenih sekvenci.
11.	Broj RTP paketa u krivom redoslijedu	Paketi čija sekvenca je različita od očekivane.
12.	Broj dupliciranih RTP paketa	Paketi čija sekvenca je već primljena.
13.	Mjerenje kašnjenja, varijacije kašnjenja RTP sesije	Računanje razlike vremena prijema paketa i vremena zapisanog u RTP vremenskoj oznaci paketa (ovo mjerenje je validno jedino ako su satovi sustava klijenta i servera sinkronizirani).

3. POSTOJEĆA RJEŠENJA ZA MJERENJE QOS

U ovom poglavlju opisana su postojeća programska rješenja za mjerenje kvalitete usluge mreže s klijentske strane testirana u sklopu ovog rada. Za svako testirano rješenje predstavljene su podržane metrike i mogućnosti njihovog prikazivanja. Testirani programi evaluirani su prema postavljenim kriterijima i odabrano je najpogodnije rješenje za zadatak ovog diplomskog rada.

3.1. Kriterij odabira postojećih rješenja

Kriteriji prihvatljivosti postojećih programskih rješenja za nadziranje mreže postavljeni su prema metrikama predloženim u drugom poglavlju ovog rada te prema mogućnosti integracije s drugim sustavima.

Najvažniji kriterij predstavljaju metrike predložene ovim radom. Prihvatljivost nekog programskog rješenja određena je brojem metrika koje podržava. Također je bitan način na koji su prikupljeni podaci dijeljeni s vanjskom okolinom. Preferiran je standardiziran oblik podataka koji može biti jednostavno parsiran. U tom smislu, najbolja rješenja imaju mogućnost pružanja rezultata u prijenosnim formatima kao što su XML (engl. *Extensible markup language*) ili JSON (engl. *JavaScript object notation*), ili u binarnom formatu.

Dodatno, postavljeni su kriteriji vezani uz interoperabilnost rješenja s drugim sustavima. Ovi kriteriji uključuju: podržanu platformu (ili platforme), vrstu sučelja (API (engl. *Application Programming Interface*), komandna linija, grafičko sučelje) i način licenciranja programa (komercijalno, otvoreni kod, slobodno). Testirana programska rješenja trebaju podržavati MS *Windows* i *Linux* operacijske sustave, čime se osigurana visok stupanj kompatibilnosti s postojećim rješenjima. Sučelje programskog rješenja treba omogućiti programsko pozivanje funkcija i pružati rezultate u formatu koji je moguće programski parsirati. Najbolja rješenja pružaju API preko kojeg se mogu dobiti rezultati. Prihvatljiva su i rješenja koja imaju sučelje komandne linije, budući da je radom ovih programa moguće upravljati programski, korištenjem skripti, te omogućiti zapis rezultata u datoteke na tvrdom disku. Ipak, ovakva rješenja, najčešće su nepotrebno kompleksna.

Način licenciranja gotovog programskog rješenja vrlo je bitan kriteriji odabira rješenja za implementaciju i integraciju mjerenja QoS u postojeći sustav. Komercijalno licencirani programi uzeti su u obzir, ali je njihova upotreba odbačena zbog postojanja besplatno licenciranih rješenja sa sličnim mogućnostima. Programska rješenja otvorenog koda licencirana pod GNU GPL (engl. *GNU General Public Licence*) dozvoljavaju slobodnu upotrebu, pod uvjetom da proizvodi zasnovani na ovim rješenjima također budu otvorenog koda, što u slučaju ovog diplomskog rada

nije ideja te ovi programi nisu prihvatljivi. Na posljetku, u potpunosti su prihvatljivi programi licencirani pod uvjetima koji dozvoljavaju potpuno slobodnu upotrebu. Primjer ovih uvjeta su BSD (engl. *Berkeley software distribution*) i MIT (engl. *Massachusetts Institute of Technology*) licence.

Tablica 3.1. prikazuje opisane kriterije i stupnjeve prihvatljivosti programskih rješenja. Zelenom bojom označava se potpuna, žutom djelomična prihvatljivost i crvenom neprihvatljivost rješenja.

Tablica 3.1. *Kriteriji implementacije postojećih rješenja*

Kriterij	**Stavke**		
Podržane definirane metrike	U potpunosti	Djelomično	
Implementacija pojedine metrike	Samostalno	Potrebna dodatna analiza	
Format rezultata	Binarni zapis	XML, JSON	Tekstualni zapis
Platforma	Windows/Linux	Linux	Windows
Sučelje	API	Komandna linija	Samo grafičko sučelje
Uvjeti licence	Slobodno korištenje	Otvoreni kod	Komercijalna licenca

3.2. Testiranje postojećih rješenja

U sklopu ovog rada testirano je nekoliko programa i programskih biblioteka. Svi testirani programi imaju određeni skup funkcionalnosti vezan uz nadziranje mrežnog prometa. Testirani programi i biblioteke su:

- *Tcpdump* – Program za analizu, filtriranje i prikaz mrežnog prometa i jednostavnu disekciju mrežnih protokola
- *Ettercap* – Program s mogućnostima filtriranja i analize mrežnog prometa. Podržava izvođenje napada na razini lokalne mreže koji uzrokuju preusmjeravanje mrežnog prometa.
- *Etherape* – Program za praćenje mrežne aktivnosti između krajnjih točaka na razini IP protokola.
- *Wireshark* – Program s mogućnostima filtriranja i napredne analize prometa, te napredne disekcije paketa s podrškom za velik broj mrežnih protokola

- *Libpcap* – Upravljački program mrežnog sučelja i popratna biblioteka koja omogućuje preuzimanje paketa s mrežnog sučelja i injekciju paketa u mrežu.
- *Libtins* – Programska biblioteka koja funkcionalnosti *libpcap* biblioteke nadograđuje API-jem za disekciju mrežnih protokola.

3.2.1. Opis testiranih rješenja

Tcpdump ispisuje opis sadržaja paketa preuzetih s mrežnog sučelja koji odgovaraju zadanom filtru [3]. Ovaj program ima mogućnost pohrane preuzetih paketa u datoteke za kasniju analizu, kao i čitanje sadržaja ovih datoteka. Pokreće se kroz komandnu liniju. Program pruža osnovne funkcionalnosti ispisa informacija zaglavlja protokola mrežnog i prijenosnog sloja, te ispisa sadržaja paketa. Prikazani paketi također su upotpunjeni vremenskim oznakama, čime je omogućena osnovna analiza mrežnog prometa.

Tablica 3.2. *Slučajevi korištenja programa Tcpdump*

Slučaj 1: Čitanje paketa s mrežnog sučelja i ispis zaglavlja

`tcpdump -i wlan0 -nv "ip host drava.etfos.hr and icmp"`
`    ...`
`13:28:18.019098 IP (tos 0x0, ttl 64, id 47916, offset 0, flags [DF], proto ICMP (1), length 84)`
`    192.168.8.100 > 161.53.201.4: ICMP echo request, id 4148, seq 1, length 64`

Slučaj 2: Zapisivanje paketa u datoteku

`tcpdump -i wlan0 -nv -c 25 -w ./output.pcap "ip host drava.etfos.hr and tcp port 80"`
`    ...`
`25 packets captured`
`47 packets received by filter`
`0 packets dropped by kernel`

Slučaj 3: Čitanje paketa iz datoteke, ispis zaglavlja i podataka

`tcpdump -nv -r ./output.pcap`
`    ...`
`13:38:22.337443 IP (tos 0x0, ttl 64, id 31140, offset 0, flags [DF], proto TCP (6), length 405)`
`    192.168.8.100.45344 > 161.53.201.4.80: Flags [P.], cksum 0x6eb7 (correct), seq 1:354, ack 1,`
`win 229, options [nop,nop,TS val 1031056 ecr 349197178], length 353: HTTP, length: 353`
`        GET / HTTP/1.1`
`        Host: www.etfos.hr`
`        User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:53.0) Gecko/20100101 Firefox/53.0`
`        Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`
`        Accept-Language: en-US,en;q=0.5`
`        Accept-Encoding: gzip, deflate`
`        Referer: https://www.google.hr/`
`        Connection: keep-alive`
`        Upgrade-Insecure-Requests: 1`
`    ...`

Tablica 3.2. prikazuje moguće slučajeve korištenja programa *Tcpdump*. Parametri rada programa podešavaju se opcijama komandne linije. Opcijom *-n* specificira se ispis IP adrese i *port-a* nekog Internet *host-a* umjesto DNS (engl. *Domain Name System*) imena. Opcijom *-v* ispisuju se sva polja zaglavlja paketa. Opcijom *-i* odabire se mrežno sučelje s kojeg se čitaju paketi. Opcijom *-c* može se definirati broj očitanih paketa, nakon kojeg će program prekinuti s radom. Opcijama *-w* i *-r* specificira se ime datoteke u koju će paketi biti zapisani, odnosno iz koje će biti očitani. Ove datoteke zapisane su u *pcap* (engl. *Packet capture*) formatu definiranom prema [24]. Komandna linija ovog program završava *pcap* filtrom. Ovo je tekstualni izraz kojim se definira koji od paketa očitanih s mrežnog sučelja trebaju biti ispisani na ekran, odnosno zapisani u datoteku. Sintaksa *pcap* filtra definirana je u [25]. Tekstualni filtar prevodi se u program koji se izvodi na razini upravljačkog programa mrežnog uređaja i određuje koji paketi trebaju biti kopirani s mrežnog sučelja i dostavljeni pozivajućem programu.

Tcpdump pruža platformu na kojoj je moguće izgraditi aplikaciju za prikupljanje QoS metrika. Njegov glavni nedostatak je što nema mogućnost samostalnog prikupljanja metrika. Ovaj program može se podesiti da automatski pohranjuje filtrirane pakete, nakon čega je potrebna posebna aplikacija za analizu i izdvajanje željenih metrika. Prednosti ovog programa su podrška za sve tražene platforme, te slobodni uvjeti korištenja.

Ettercap je program za prikupljanje i analizu paketa na lokalnoj mreži koji dodatno pruža skup alata za izvođenje MitM (engl. *Man in the middle*) napada [5]. Glavne prednosti ovog programa su mogućnost injekcije paketa na mrežno sučelje, te mogućnost uvođenja vlastite politike prosljeđivanja paketa koja zaobilazi operacijski sustav. *Ettercap* ima slične mogućnosti analize mrežnog prometa kao i *Tcpdump*. Slučajevi korištenja ovog programa prikazani su tablicama 3.3. i 3.4.

Tablica 3.3. *Slučajevi korištenja programa Ettercap*

<table>
<tr><th colspan="2">Slučaj 1: Čitanje paketa s mrežnog sučelja, zapisivanje u pcap datoteku, istovremeno log-anje paketa</th></tr>
<tr><td colspan="2">ettercap -Tzq -w ./ettercap_logs/packetdump.pcap -L ./ettercap_logs/log //161.53.201.4/80</td></tr>
<tr><th colspan="2">Slučaj 2: Analiza packet log-a</th></tr>
<tr><td>etterlog -c ./ettercap_logs/log.ecp
...
Creating the connection table...
Found 2 connection...

TCP: 192.168.8.100:44270 <--> 161.53.201.4:80
TCP: 192.168.8.100:44268 <--> 161.53.201.4:80</td><td>etterlog -a ./ettercap_logs/log.ecp
...
Log file size (uncompressed) : 59515
...
Effective payload size : 51627
...
Number of packets : 98
Average size per packet : 526</td></tr>
</table>

Tablica 3.4. *Slučajevi korištenja programa Ettercap (nastavak)*

Slučaj 3: Ispis paketa *packet log-a* za zadanu konekciju

`etterlog -T -F TCP:192.168.8.100:44268:161.53.201.4:80 ./ettercap_logs/log.ecp`
`    ...`
`Sun Aug 13 17:07:25 2017 [545514]`
`TCP  192.168.8.100:44268 --> 161.53.201.4:80 | S (0)`
 	
`Sun Aug 13 17:07:25 2017 [570224]`
`TCP  161.53.201.4:80 --> 192.168.8.100:44268 | SA (0)`
 	
`Sun Aug 13 17:07:25 2017 [570242]`
`TCP  192.168.8.100:44268 --> 161.53.201.4:80 | A (0)`
 	
`Sun Aug 13 17:07:25 2017 [570323]`
`TCP  192.168.8.100:44268 --> 161.53.201.4:80 | AP (320)`
`GET / HTTP/1.1`
`Host: www.etfos.hr`
`User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:53.0) Gecko/20100101 Firefox/53.0`
`Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8`
`Accept-Language: en-US,en;q=0.5`
`Accept-Encoding: gzip, deflate`
`Connection: keep-alive`
`Upgrade-Insecure-Requests: 1`
`    ...`

Iz tablice 3.3. vidljivo je da *Ettercap* ima naprednije mehanizme spremanja paketa od *Tcpdump-a*. Paketi su spremljeni u binarne datoteke s ekstenzijom *.ecp* koji mogu biti parsirani programom *Etterlog* [26]. Ovaj program omogućava statističku analizu *log* datoteka, prepoznavanje i izdvajanje TCP konekcija i napredno filtriranje paketa.

Ettercap programom se ne mogu samostalno dobiti definirane metrike, već je potrebna dodatna analiza datoteka i rezultata koje može pružiti ovaj program. Ovaj program posjeduje komandnu liniju i grafičko sučelje. Kompatibilan je s *Linux* operacijskim sustavima. Njegovo korištenje uvjetovano je licencom otvorenog koda, što ga čini neprihvatljivim.

Etherape je preglednik mrežnog prometa koji grafičkim sučeljem prikazuje aktivnosti mrežnih *host-ova* [27]. Ovaj program omogućava identifikaciju krajnjih točaka komunikacije i razvrstavanje mrežnog prometa prema izvoru ili odredištu, te prema protokolima fizičkog, mrežnog, prijenosnog i aplikacijskog sloja. Programom je moguće upravljati iz komandne linije slanjem signala. Konkretno, omogućeno je spremanje trenutnog stanja mrežnog prometa u XML formatu koji je prikazan tablicom 3.5. Ime čvora (element <*name*>) sastoji se iz DNS imena i IP adrese. Za svaki poznati mrežni čvor spremljene su statistike prometa i razvrstane prema komunikacijskim protokolima

Tablica 3.5. *Format mrežnih statistika programa Etherape*

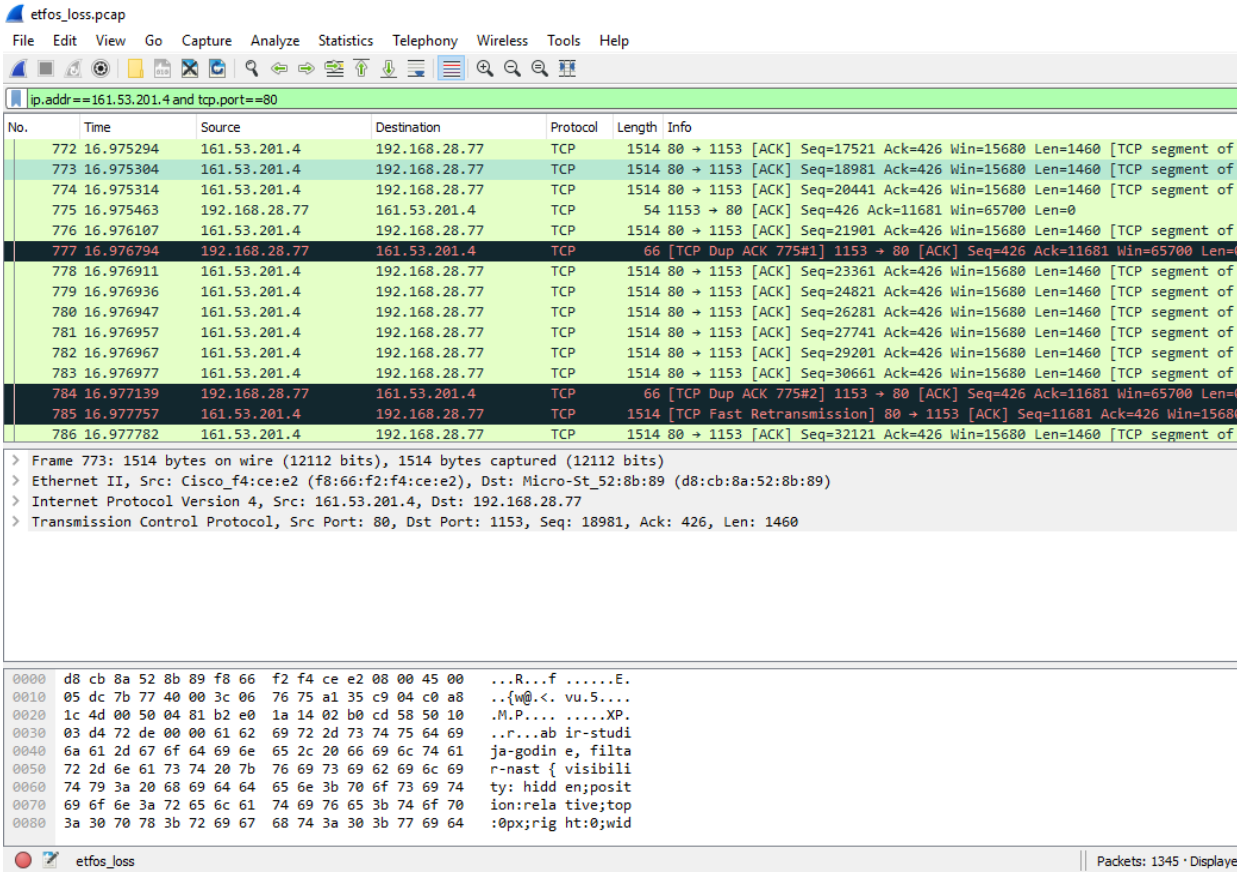
Globalna struktura	**Element *<traffic_stats>***
<etherape> <header> <capture_device> wlan0 </capture_device> <timestamp> 2017-08-13 14:22:41 +0200 </timestamp> </header> <nodes> <node> <name>...</name> <traffic_stats> ... </traffic_stats> </node> ... </nodes> </etherape>	<traffic_stats> <active_packets>0</active_packets> <in>...</in> <out>...</out> <tot>...</tot> <protocols> ... <protocol> <level>2</level> <key>IP</key> <stats> <avg>0</avg> <total>113292</total> <avg_size>515</avg_size> <packets>220</packets> <last_heard> 19.096227 </last_heard> </stats> <name>...</name> </protocol> ... </protocols> </traffic_stats>

Etherape pruža statistiku mrežnog prometa u XML formatu. Uz pomoć ovog programa moguće je, parsiranjem XML-a dobiti metriku količine mrežnog prometa. Međutim, ovaj program uopće ne podržava ostale definirane metrike. *Etherape* je kompatibilan s *Linux* operacijskim sustavima. Njegovo korištenje uvjetovano je licencom otvorenog koda, što ga čini neprihvatljivim.

Wireshark je program za naprednu analizu mrežnog prometa [4]. U osnovi *Wireshark* pruža funkcionalnost čitanja paketa sa zadanog mrežnog sučelja u memoriju ili na tvrdi disk. Međutim, ovaj program ima mogućnost disekcije velikog broja mrežnih protokola, kao i algoritme za naprednu analizu često korištenih protokola. Tako su neki od standardnih *Wireshark* modula alati za analizu TCP i RTP tokova. Ovi alati mogu samostalno pružiti uvid u neke od metrika definiranih ovim radom. *Wireshark* posjeduje grafičko sučelje koje čini glavni dio ovog programa, i komandnu liniju, kojom se veći dio podataka dobivenih grafički može dobiti u tekstualnom obliku.

Grafičko sučelje programa *Wireshark* prikazano na slici 3.1. sastoji se iz tri glavna odjeljka. Unutar prvog odjeljka prikazani su paketi odabrani filtrom na vrhu odjeljka. *Wireshark* filtar napredniji je od *pcap* filtra, te podržava filtriranje paketa prema poljima protokola (npr. *ip.addr*) i prema svojstvima koje sam program generira analizom (npr. *tcp.analysis.out_of_order*). Prilikom čitanja paketa s mrežnog sučelja, svi očitani paketi spremaju se u memoriju, filtrom se određuje koji paketi će biti prikazani. Prikazani paketi označeni su različitim bojama zbog jednostavnije

vizualne analize. Tako na slici 3.1. zeleno obojeni paketi označavaju normalan TCP promet, dok crno obojeni paketi predstavljaju posebne situacije (retransmisije, izgubljene pakete i krivi redoslijed). Drugi odjeljak grafičkog sučelja prikazuje disekciju mrežnih protokola odabranog paketa i analitičke podatke koje generira *Wireshark* (npr. procijenjeni RTT – engl. *Round-Trip Time*). Unutar trećeg odjeljka vidljivi su heksadecimalni i tekstualni prikaz sadržaja odabranog paketa.



Slika 3.1. *Grafičko sučelje programa Wireshark*

Ispis paketa moguće je dobiti u tekstualnom obliku programom komandne linije *Tshark* [28]. Ovaj program ispisuje sve relevantne informacije vidljive u grafičkom sučelju. Parametrima komandne linije moguće je podesiti što će biti ispisano, kao i format ispisa. Informacije paketa mogu biti ispisane ukratko (kako su prikazani u gornjem dijelu grafičkog sučelja) ili detaljno (kako su prikazani u srednjem dijelu sučelja). Detaljan ispis podatak omogućen je komandom *-T* kao običan tekst (*-T text*), u XML (*-T pdml* za detaljan ispis ili *-T psml* za sumarni ispis) ili JSON (*-T json*) formatu. Moguć je i ispis polja paketa po želji u TSV formatu (engl. *Tab Separated Values*) komandom *-T fields*. *Tshark* omogućava i ispis statistika prometa, poput količine prometa u nekom vremenskom intervalu, statistike detektiranih TCP konekcija i RTP tokova. Neki slučajevi korištenja alata *Tshark* prikazani su u tablici 3.6. i 3.7.

Tablica 3.6. *Slučajevi korištenja programa Tshark*

Slučaj 1: Ispis kratkog opisa paketa

```
tshark -r etfos_loss.pcap "frame.number==759"

  759  16.969422 161.53.201.4 → 192.168.28.77 TCP 1514 HTTP/1.1 200 OK  [TCP segment of a
reassembled PDU]
```

Slučaj 2: Ispis kratkog opisa paketa u XML formatu

```
tshark -r etfos_loss.pcap -T psml
"frame.number==759"

<?xml version="1.0" encoding="utf-8"?>
<psml version="0" creator="wireshark/2.4.0">
<structure>
<section>No.</section>
<section>Time</section>
<section>Source</section>
<section>Destination</section>
<section>Protocol</section>
<section>Length</section>
<section>Info</section>
</structure>
```

```
<packet>
<section>759</section>
<section>16.969422</section>
<section>161.53.201.4</section>
<section>192.168.28.77</section>
<section>TCP</section>
<section>1514</section>
<section>
  HTTP/1.1 200 OK  [TCP segment of a
reassembled PDU]
</section>
</packet>

</psml>
```

Slučaj 3: Ispis paketa u TSV formatu

```
tshark -r etfos_loss.pcap -e frame.number -e frame.time -e ip.src -e ip.dst -e ip.proto -e
ip.len -E header=y -T fields "frame.number==759"
frame.number    frame.time      ip.src  ip.dst  ip.proto        ip.len
759     Mar 20, 2017 07:50:05.734240000 Central European Standard Time  161.53.201.4
192.168.28.77   6       1500
```

Slučaj 4: Ispis statistika prometa

```
tshark -r etfos_loss.pcap -q -z
"io,stat,1,ip.addr==161.53.201.4 &&
tcp.port==1153"


================================================
====
| IO Statistics                                |
|                                              |
| Duration: 25.634567 secs                     |
| Interval:  1 secs                            |
|                                              |
| Col 1: ip.addr==161.53.201.4 &&
tcp.port==1153 |
```

```
|----------------------------------------------|
|          |1                |                 |
| Interval | Frames |  Bytes |                 |
|--------------------------|                   |
|  0 <>  1 |      0 |      0 |                 |
   ...
| 15 <> 16 |      0 |      0 |                 |
| 16 <> 17 |     63 |  58069 |                 |
| 17 <> 18 |    174 | 169398 |                 |
   ...
| 25 <> Dur|      0 |      0 |                 |
================================================
====
```

Slučaj 5: Ispis statistika RTP tokova

```
tshark -r rtp_stream.pcap -qz rtp,streams
======================== RTP Streams ========================
    Src IP addr  Port    Dest IP addr  Port       SSRC          Payload  Pkts
      Lost   Max Delta(ms)  Max Jitter(ms) Mean Jitter(ms) Problems?
  192.168.0.124 56027       239.0.0.1  5004 0xC7FD9C1C MPEG-II transport streams
   287   350 (54.9%)            0.00            0.00            0.00 X
```

Tablica 3.7. *Slučajevi korištenja programa Tshark (nastavak)*

Slučaj 6: Ispis statistika TCP konekcija
tshark -r etfos_loss.pcap -q -z "conv,tcp" == TCP Conversations Filter:<No Filter> | <- | | -> | | Total | Relative | Duration | | Frames Bytes | | Frames Bytes | | Frames Bytes | Start | | 192.168.28.77:1128 <-> 216.58.209.195:443 153 171767 91 7749 244 179516 8.893831000 10.0986 192.168.28.77:1153 <-> 161.53.201.4:80 150 221714 89 5867 239 227581 16.816224000 5.4505

Wireshark pruža napredne mogućnosti analize mrežnog prometa. Neke od definiranih metrika QoS moguće je dobiti ovim programom bez potrebe za dodatnom analizom. Željene informacije mogu biti formatirane u XML, JSON ili TSV prijenosne formate, čime je olakšano njihovo parsiranje. Nedostatak ovog programa, kao i svih dosad prikazanih jest što se informacije ne mogu dobiti „na licu mjesta“, prilikom nadgledanja aktivne komunikacije, već isključivo naknadnom analizom spremljenih datoteka koje prikazuju sadržaj komunikacije. Prikazani programi također ne podržavaju proizvoljnu injekciju paketa u mrežu, koja je neophodna za implementaciju nekih definiranih metrika. *Wireshark* podržava *Windows* i *Linux* platforme. Njegovo korištenje uvjetovano je licencom otvorenog koda.

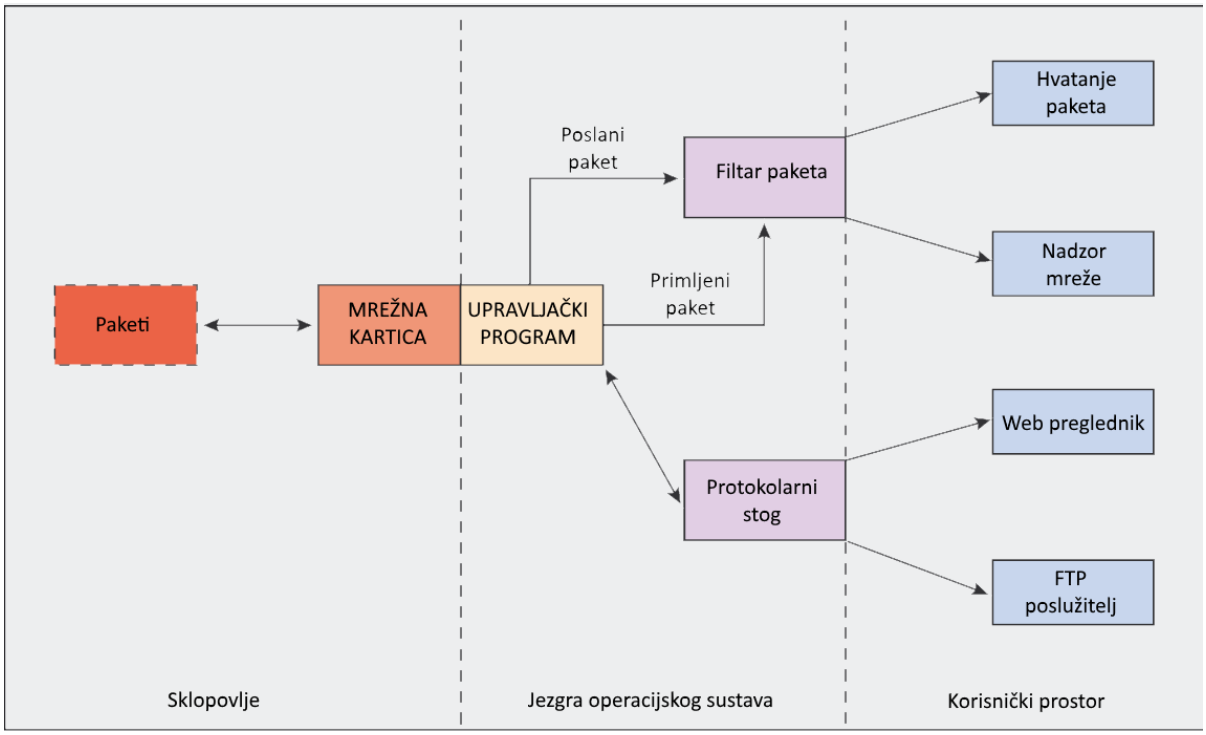
Libpcap je programska biblioteka pisana u C programskom jeziku koja pruža mehanizam za „hvatanje“ paketa s mrežnog sučelja [7]. Također je podržano spremanje i čitanje paketa iz *pcap* datoteka. Ova biblioteka predstavlja osnovu svih programa za analizu mreže prikazanih u sklopu ovog rada. Njezina funkcionalnost zasnovana je na mehanizmu upravljačkih rutina mrežnog sučelja – filtara koji pristigle pakete evaluiraju prema postavljenom filtar-izrazu i kopiraju iz domene operacijskog sustava u korisničku domenu, gdje ih prosljeđuju bibliotečnim funkcijama. Pomoću bibliotečnih funkcija, korisnički programi mogu dohvatiti sadržaj filtriranih paketa. Ovaj postupak filtriranja prikazan je na slici 3.2.

API koji pruža *Libpcap* biblioteka sadrži funkcije za prepoznavanje mrežnih sučelja, prevođenje tekstualnih filtar-izraza u rutine, inicijalizaciju sučelja za prikupljanje paketa, dohvaćanje paketa i deinicijalizaciju. Sekvenca bibliotečnih poziva kod čitanja s mrežnog sučelja prikazan je na slici 3.3. Rad ovih funkcija detaljno je opisan u [7]. Lista dostupnih mrežnih sučelja dohvaća se funkcijom *pcap_findalldevs,* dok se funkcijom *pcap_lookupdev* automatski dohvaća

prvo dostupno sučelje. Funkcijom *pcap_create* kreira se rukovatelj za čitanje paketa s mrežnog sučelja. Funkcijom *pcap_compile*, tekstualni filtar prevodi se u rutinu upravljačkog programa mrežnog sučelja koja se dodjeljuje rukovatelju funkcijom *pcap_setfilter*. Postoje tri funkcije koje vraćaju očitane pakete:

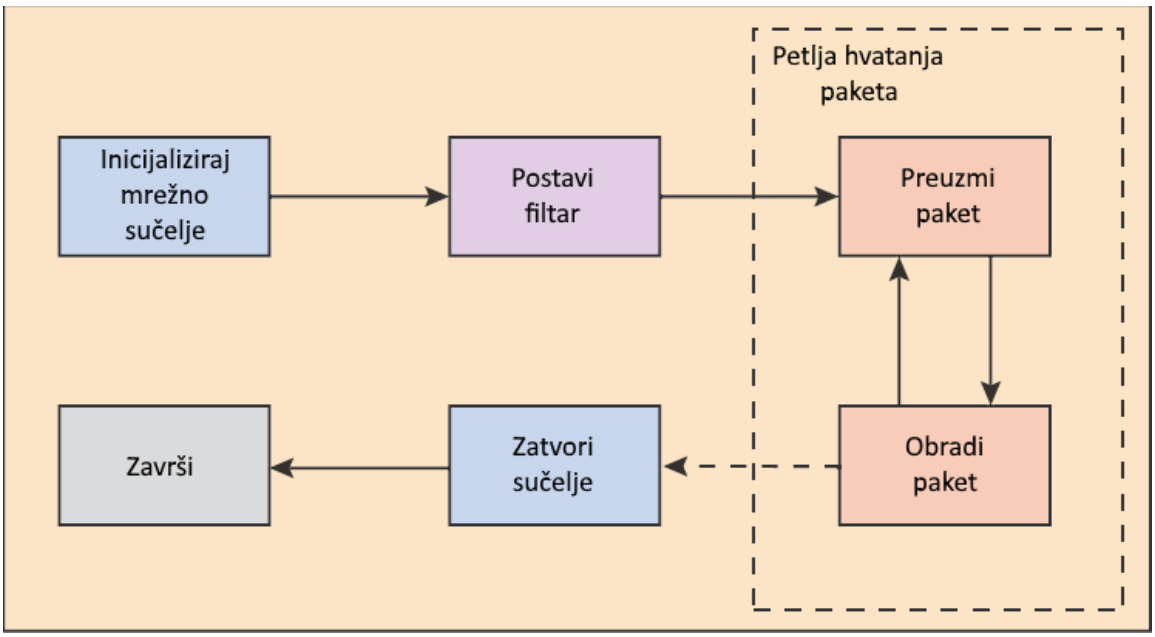
- *pcap_next* – čeka na sljedeći paket i vraća njegov sadržaj.
- *pcap_loop* – očitava pakete s mrežnog sučelja do pojave greške ili prekida. Očitani paketi prosljeđuju se *callback* funkciji zadanoj od strane korisnika.
- *pcap_dispatch* - očitava pakete s mrežnog sučelja do pojave greške ili prekida, ili do isteka zadanog vremenskog intervala. Očitani paketi prosljeđuju se *callback* funkciji zadanoj od strane korisnika.

Deinicijalizacija rukovatelja za čitanje paketa obavlja se funkcijom *pcap_close*.



Slika 3.2. *Prikaz mehanizma filtriranja paketa*[29]

Libpcap biblioteka omogućuje pristup mrežnim paketima na fizičkom nivou. Najveća prednost ove biblioteke je što pruža programski API koji je vrlo lako integrirati s postojećim sustavom kao novi modul. Nedostatak je što ova biblioteka nema funkcionalnosti disekcije mrežnih protokola. *Libpcap* podržava *Linux* i *Windows* platforme. Korištenje je uvjetovano slobodnom licencom.



Slika 3.3. *Rad Libpcap biblioteke*[29]

Libtins je programska biblioteka za čitanje, sastavljanje i manipulaciju mrežnim paketima pisana u C++ programskom jeziku. Neke od funkcionalnosti koje ova biblioteka podržava su: čitanje paketa s mrežnog sučelja, disekcija komunikacijskih protokola fizičkog, Internet i prijenosnog nivoa, sastavljanje paketa i injekcija u mrežu, čitanje i spremanje *pcap* datoteka [8].

Ova biblioteka predstavlja nadogradnju *Libpcap-a* na kome je i zasnovana. Zadržane su sve osnovne funkcionalnosti, a istovremeno dodane nove. Najznačajnije među njima su disekcija mrežnih protokola i mogućnost sastavljanja i slanja paketa. *Libtins* je objektno orijentirana biblioteka. Za prikupljanje mrežnih paketa zadužen je objekt klase *Sniffer*. Ovaj objekt obuhvaća funkcionalnosti inicijalizacije sučelja, filtriranja i čitanja paketa. Filtrirani paketi dostavljaju se korisničkoj aplikaciji posredstvom *callback* funkcije koja je predana *Sniffer* objektu. Velika prednost ove biblioteke u odnosu na *Libpcap* je što su očitani paketi dostavljeni korisniku kao instance klasa koje predstavljaju mrežne protokole. Poljima zaglavlja paketa moguće je pristupiti kao svojstvima ovih objekata. Popis podržanih protokola dostupan je u [8]. Na posljetku, *Libtins* omogućuje korisniku instanciranje vlastitih paketa i postavljanje njihovih zaglavlja, te slanje paketa posredstvom klase *Sender*. Biblioteka također ima podršku za proširenje mehanizma disekcije dodavanjem novih protokola.

Libtins biblioteka proširuje API za čitanje mrežnih paketa *Libpcap-a* mogućnostima disekcije protokola i injekcije paketa. Za dobivanje definiranih metrika potrebna je dodatna analiza pročitanih paketa, no ova analiza olakšana je činjenicom da biblioteka manipulira objektima, a ne

binarnim zapisom paketa (*Libpcap*) ili tekstualnim zapisima različitih formata (ostali testirani programi).

3.2.2. Rezultati testiranja postojećih rješenja

Tablicom 3.8. sumarno su prikazani rezultati i zaključci testiranja opisanog u prethodnom poglavlju. Definirane metrike označene su znakom „*“ kod aplikacija gdje je za njihovo dobivanje potrebna dodatna analiza ili pomoć drugog programa.

Tablica 3.8. *Sumarni prikaz analize postojećih rješenja*

Progam		Tcpdump	Ettercap	Etherape	Wireshark	Libpcap	Libtins
Metrike	1.	*	*	*	*	*	Da
	2.	*	*	*	*	*	Da
	3.	*	*	*	*	*	Da
	4.	*	*	*	*	*	Da
	5.	*	*	Da	Da	Da	Da
	6.	*	*	Ne	*	*	*
	7.	*	*	Ne	*	*	*
	8.	*	*	Ne	*	*	*
	9.	*	*	Ne	*	*	*
	10.	Ne	Ne	Ne	*	*	*
	11.	Ne	Ne	Ne	*	*	*
	12.	Ne	Ne	Ne	*	*	*
	13.	Ne	Ne	Ne	*	*	*
Sučelje		Komandna linija	Komandna linija	Grafičko, komandna linija	Grafičko, komandna linija	API	API
Format		Tekst	Tekst	XML	Tekst, XML, JSON, TSV	Binarni	Binarni
Platforma		Linux, Windows	Linux	Linux	Linux, Windows	Linux, Windows	Linux, Windows
Licenca		Slobodna	Otvoreni kod	Otvoreni kod	Otvoreni kod	Slobodna	Slobodna

Na osnovu prikazanih rezultata, *Libtins* programska biblioteka smatra se najpogodnijom platformom za razvoj aplikacije koja će analizom mrežnih protokola prikupljati QoS metrike. Ova biblioteka pruža API za prikupljanje paketa i disekciju protokola pomoću kojeg je moguće implementirati QoS modul koji će biti integriran u postojeći sustav.

4. IMPLEMENTACIJA APLIKACIJE ZA MJERENJE QOS

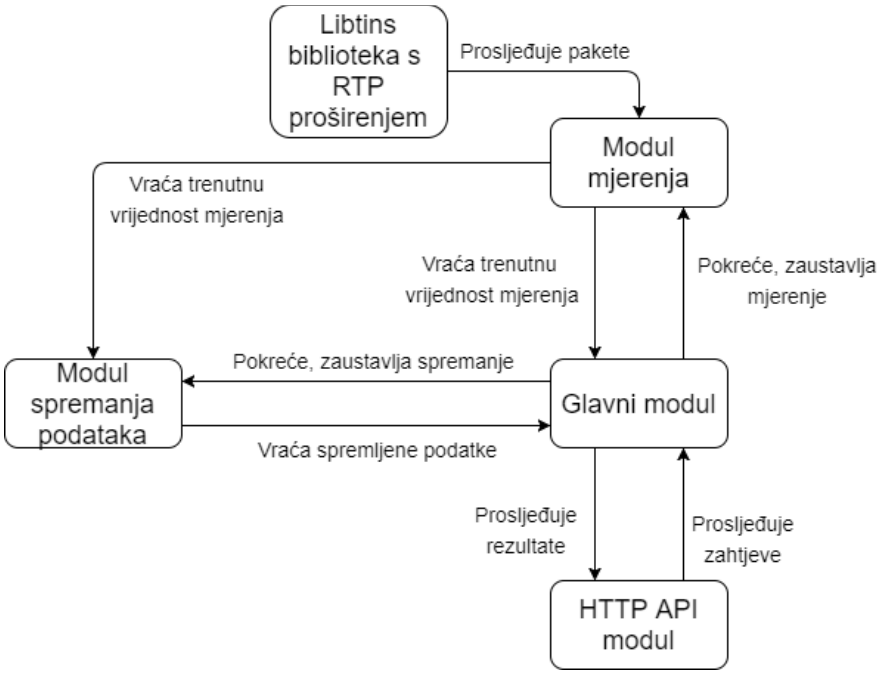
U ovom poglavlju opisana je implementirana aplikacija za prikupljanje metrika QoS. Osnova ove implementacije je *Libtins* biblioteka opisana u prethodnom poglavlju. Prikazana je arhitektura i sastavni dijelovi aplikacije. Opisani su algoritmi prikupljanja metrika. Definiran je API pomoću kojeg aplikacija komunicira s okolinom.

4.1. Arhitektura sustava mjerenja

Aplikacija za mjerenje razine QoS kod IP DTV usluga osmišljena je da podržava sljedeće funkcionalnosti:

- Mjerenje QoS – Aplikacija ima mogućnost prikupljanja svih metrika definiranih u poglavlju 2. ovog rada.
- Pohrana podataka – Aplikacija trajno pohranjuje prikupljene podatke, čime je omogućeno naknadno pristupanje i analiza.
- Komunikacija s okolinom – Aplikacija ima API kroz koji klijenti mogu pristupati metrikama.

Funkcionalnosti aplikacije implementirane su kroz rad više modula. Konceptualni model na slici 4.1. prikazuje module aplikacije i sučelja između njih.



Slika 4.1. *Konceptualni model QoS aplikacije*

Kontrolni, tj. glavni modul aplikacije je zadužen za obradu zahtjeva poslanih od strane klijenta i pozivanje drugih modula aplikacije. Na osnovu podataka očitanih iz korisničkog zahtjeva, određuje se potrebna akcija (pokretanje mjerenja, dohvaćanje metrike) i tip zahtijevane metrike. Kada klijent zahtjeva početak mjerenja, glavni modul kreira instancu modula mjerenja. Svakoj instanci mjerenja dodjeljuje se jedinstveni identifikacijski broj (ID). Kada klijent zahtjeva vrijednost mjerenja, glavni modul, prema ID-u predanom u zahtjevu određuje kojoj instanci mjerenja treba pristupiti. Također, modulu za pohranu podataka predaje se instanca mjerenja i njezin ID. Modul za pohranu u pravilnim vremenskim intervalima uzorkuje i sprema trenutnu vrijednost instance mjerenja. Klijent može specificirati vremenski raspon kako bi pristupio nizu vrijednosti neke metrike. U tom slučaju, glavni modul poziva modul za pohranu podataka koji iz datoteke očitava vrijednosti mjerenja dohvaćene unutar zadanog vremenskog raspona. Kada klijent zahtjeva zaustavljanje mjerenja, glavni modul nalaže modulu za pohranu da zaustavi zapis. Zatim oslobađa instancu mjerenja.

Modul za pohranu podataka rukuje zapisima (engl. *log*) u kojima su pohranjeni podaci QoS mjerenja. Za svaku instancu mjerenja kreiran je poseban zapis, imenovan prema ID-u mjerenja. U zapis su pohranjene sve metrike koje pruža pojedina instanca mjerenja, zajedno s vremenskim oznakama koje pokazuju kad su podaci dohvaćeni. Podaci su zapisani u tekstualnom CSV (engl. *Comma separated values*) formatu. Ovaj format jednostavan je za programsko parsiranje, dok je istovremeno čitljiv golim okom, čime je olakšana provjera ispravnosti zapisa i pronalazak grešaka. Budući da QoS aplikacija omogućuje dohvaćanje metrika u različitim mjernim jedinicama, vrijednosti metrika u zapisu imaju mjernu jedinicu najveće rezolucije. Tako su metrike mrežnog protoka zapisane kao broj okteta, odnosno broj paketa po sekundi, dok su vremenske metrike zapisane u mikrosekundama. Vremenske oznake podataka zapisane su kao broj sekundi od početka *Unix* epohe (1.1.1970. 00:00:00).

Modul mjerenja prikuplja QoS metrike, dok HTTP modul komunicira s klijentskim sustavima. Pojedinosti ovih modula opisane su u sljedećim potpoglavljima.

4.2. Implementacija metrika

Modul mjerenja sastoji se iz programske biblioteke *QoSMonitoring*, pisane u C++ programskom jeziku. Biblioteka je objektno orijentirana i definira klase kojima je implementirana analiza mrežnih protokola i prikupljanje metrika. Pri tome je za prikupljanje mrežnih paketa i disekciju protokola korištena *Libtins* biblioteka. Mrežni paketi koje aplikacija analizira moraju biti prikupljeni s nekog od mrežnih sučelja uređaja na kojem je aplikacija pokrenuta. Stoga, aplikacija

za mjerenje mora biti pokrenuta na uređaju, odnosno mrežnom sučelju uređaja kroz koje će biti proslijeđeni paketi koji pripadaju promatranoj mrežnoj komunikaciji.

Budući da *Libtins* ne podržava disekciju RTP paketa, u sklopu ovog rada implementirano je proširenje ove biblioteke. Proširenja mehanizma disekcije protokola *Libtins* biblioteke omogućena su nasljeđivanjem apstraktne klase *PDU*. Primjer ovog postupka dan je u [30]. U sklopu ovog rada implementirana je disekcija RTP protokola, te SR i RR paketa RTCP protokola. Ova funkcionalnost implementirana je kroz posebni modul *LibtinsRTPExtension* koji se dinamički povezuje s aplikacijom koja ga koristi (u ovom slučaju *QoSMonitoring* biblioteka).

Instance modula mjerenja, nazvane *monitor-i*, implementiraju algoritme za prikupljanje različitih metrika, ali imaju zajednički mehanizam djelovanja. Prilikom instanciranja, *monitor-u* se prosljeđuje ime mrežnog sučelja s kojeg će prikupljati pakete. Svaki *monitor* posjeduje *start* funkciju kojoj se prosljeđuju parametri nadgledane komunikacije (npr. krajnje točke i protokol). Na osnovu ovih parametara kreiran je *pcap* filtar koji određuje koji paketi će s mrežnog sučelja biti proslijeđeni *monitoru*. Također, *monitor* posjeduje *stop* funkciju kojom se prekida prikupljanje paketa. *Monitori-i* i metrike koje prikupljaju prikazane su tablicoma 4.1.

Tablica 4.1. *Instance modula mjerenja i podržane metrike*

Monitor prometa	**TCP monitor**	**RTP monitor**	**Monitor dostupnosti**
Količina mrežnog prometa	TCP izgubljeni paketi	RTP izgubljeni paketi	ICMP dostupnost *host-a*
	TCP duplicirani paketi	RTP duplicirani paketi	TCP dostupnost *host-a*
	TCP paketi krivog redosljeda	RTP paketi krivog redosljeda	ICMP kašnjenje
	TCP kašnjenje	RTP kašnjenje	TCP kašnjenje
	TCP varijacija kašnjenja	RTP varijacija kašnjenja	

Monitor prometa provodi jednostavan algoritam kojim se dobiva metrika količine mrežnog prometa. On sumira broj i ukupnu veličinu svih paketa dobivenih u vremenskom intervalu mjerenja od jedne sekunde. Pri tome su izvorišna i odredišna adresa paketa, pročitane iz zaglavlja IP protokola, korištene za razdvajanje prijemnog i otpremnog prometa. Evidencija količine prometa ažurirana je na kraju svakog mjernog intervala. Ona se sastoji od vrijednosti prijemnog, otpremnog i ukupnog prometa, zapisanih u broju paketa i broju okteta po sekundi.

TCP *monitor* prikuplja podatke o broju grešaka TCP konekcije. Greške uključuju izgubljene, duplicirane i pakete u krivom redoslijedu. Za detekciju izgubljenih podataka koristi se kombinacija TCP algoritma za brzu retransmisiju [31] i TCP SACK opcije [32]. Analizom ACK

sekvenci utvrđuje se kada prijemna strana konekcije šalje duplicirane ACK sekvence. Kada broj dupliciranih ACK sekvenci dostigne *DupThresh* (ova vrijednost je najčešće 3 [15, str. 631]), pošiljatelj smatra da je paket označen ponovljenim ACK-om izgubljen. TCP *monitor* detektira gubitak paketa na isti način. Dodatno, ukoliko prijemnik i pošiljatelj podržavaju SACK, ova opcija koristi se za detekciju više izgubljenih paketa kod istog dupliciranog ACK-a. Kao duplicirani, označavaju se dolazni paketi čiji je SEQ manji od zadnjeg poslanog ACK-a (ovi paketi su već primljeni i potvrđeni), ili odlazni paketi čiji je SEQ jednak zadnjem poslanom kada su u prethodnom paketu poslani podaci (SEQ broj je trebao biti povećan). Paketi u krivom redosljedu su oni dolazni paketi čiji je SEQ veći od zadnjeg poslanog ACK-a, ili odlazni čiji je SEQ broj manji od zadnjeg poslanog SEQ-a. Problem kod mjerenja broja izgubljenih, dupliciranih, i paketa u krivom redoslijedu je što TCP protokol barata s kontinuiranim tokom okteta podataka, gdje granice paketa nisu važne. Stoga TCP *monitor* nastoji preslikati broj okteta na broj paketa korištenjem MSS (engl. *Maximum segment size*) opcije koja određuje maksimalnu veličinu TCP segmenta za neku konekciju. Ova opcija postavljena je pri uspostavljanju konekcije (slanju SYN segmenata). Ukoliko mjerenjem nisu detektirani SYN segmenti, pretpostavlja se da MSS odgovara veličini *Ethernet* MTU-a od 1500 okteta, umanjenoj za veličinu IP i TCP zaglavlja (što iznosi 1460 okteta). Za procjenu kašnjenja identificiraju se komunikacijske točke konekcije kao klijent i poslužitelj. Pretpostavljeno je da se klijent nalazi u lokalnoj mreži mjernog uređaja. Stoga je kašnjenje paketa između klijenta i mjernog uređaja zanemareno. TCP *monitor* procjenjuje kašnjenje kao vrijeme između detekcije paketa kojim klijent šalje podatke i ACK-a kojim poslužitelj potvrđuje primitak istih podataka.

RTP *monitor* prikuplja podatke o broju grešaka RTP konekcije. Kao kod TCP-a, identificirani su izgubljeni, duplicirani i paketi u krivom redoslijedu. Greške su detektirane analizom RTP brojeva sekvence. Budući da sekvenca RTP paketa raste za jedan za svaki poslani paket, detekcija preskočenih sekvenci u RTP toku interpretira se kao gubitak paketa. RTP *monitor* privremeno sprema primljene sekvence u svrhu detekcije duplikata i krivog redosljeda. Svaka primljena sekvenca, manja od najviše primljene, uspoređuje se s međuspremnikom svih primljenih sekvenci. Ukoliko je sekvenca već spremljena, paket je duplikat, inače je primljen u krivom redoslijedu. Kašnjenje između RTP poslužitelja i klijenta računa se kao razlika između vremena primitka paketa i vremena sadržanog u RTP vremenskoj oznaci paketa. Vrijednost RTP vremenske oznake povezuje se sa stvarnim vremenom uz pomoć NTP vremenske oznake očitane iz RTCP SR paketa. Kod ovog rješenja javlja se problem sinkronizacije satova poslužitelja i klijenta. Odstupanje ovih satova uzrokuje pogrešnu vrijednost kašnjenja, koja može biti i negativna,

ukoliko poslužiteljev sat prednjači u odnosu na klijentov za vrijednost veću od stvarnog kašnjenja. Vrijednost varijacije kašnjenja (*jitter*) računa se iz RTP vremenskih oznaka prema postupku opisanom u [20]. Utjecaj na izračune kašnjenja ima i činjenica da je vrijeme primitka svakog paketa uzorkovano u trenutku kad je taj paket predan aplikaciji za mjerenje. Stoga u izračun kašnjenja ulazi vrijeme obrade i držanja paketa u međuspremniku na nivou jezgre operacijskog sustava. Dodatno, jezgra operacijskog sustava najčešće predaje aplikaciji sve pakete iz međuspremnika nakon što je popunjen [7]. Ovo uzrokuje grešku u mjerenju varijacije kašnjenja, Budući da grupa paketa iz međuspramnika stiže na obradu odjednom, a obrađeni su slijedno, varijacija kašnjenja jednaka je vremenu obrade.

Monitor dostupnosti pruža funkcionalnost provjere dostupnosti i mjerenja kašnjenja prema udaljenom mrežnom poslužitelju. Provjera dostupnosti implementirana je kreiranjem i slanjem paketa kojima se zahtijeva odgovor mrežnog poslužitelja. Korištene su dvije vrste paketa: ICMP *Echo* zahtjevi i TCP SYN segmenti. Slanjem TCP SYN segmenta pokušava se inicirati TCP konekcija s poslužiteljem. Na odgovor poslužitelja čeka se unutar vremenskog intervala koji specificira korisnik. Ukoliko odgovor ne stigne, poslužitelj se smatra nedostupnim. Kod slanja ICMP zahtjeva, mrežni uređaji na putu do poslužitelja odgovaraju greškom ukoliko zahtjev nije mogao biti usmjeren do poslužitelja. Nedostatak korištenja ICMP-a je u tome što su ovi paketi često filtrirani vatrozidima. Ukoliko je od poslužitelja stigao odgovor, utvrđuje se kašnjenje između poslanog zahtjeva i primljenog odgovora. Slanjem više uzastopnih zahtjeva prema istom poslužitelju utvrđuje se varijacija kašnjenja.

4.3. Integracija s okolinom

Komunikacija aplikacije za mjerenje i vanjske okoline implementirana je HTTP API-jem. Svaka metrika predstavljena je kao mrežni resurs s vlastitim URI-jem (engl. *Uniform Resource Identifier*). Klijenstke aplikacije pristupaju metrikama pomoću POST, GET i DELETE metoda. POST metodom klijent inicira mjerenje i specificira parametre potrebne za filtriranje paketa koji će biti analizirani. Ako je uspješno kreirana nova instanca mjerenja i odgovarajući zapis, na POST poruku aplikacija odgovara ID-em instance mjerenja. DELETE metodom zaustavlja se mjerenje i briše instanca mjerenja. Vrijednosti metrika ostaju pohranjena u datoteci zapisa, te ih je moguće dohvatiti i nakon pozivanja DELETE metode.

Kreiranom mjerenju moguće je pristupiti preko URI-ja koji sadrži ID mjerenja. GET metodom klijent dohvaća vrijednosti metrike specificirane URI-jem. Parametrima GET metode moguće je specificirati željenu mjernu jedinicu i vremenski raspon kako bi se dobio niz vrijednosti.

Rezultat GET metode je uniformni JSON format, neovisan o tipu mjerenja. Ovim formatom prikazan je ID *monitor* objekta, ime metrike, mjerna jedinica, vrijeme uzorkovanja i vrijednost. Ukoliko parametrima GET metode nije specificiran vremenski raspon, povratna vrijednost odgovara zadnjoj uzorkovanoj vrijednosti metrike. Inače, vraćen je niz uzoraka metrike, čija su vremena uzorkovanja unutar zadanog raspona.

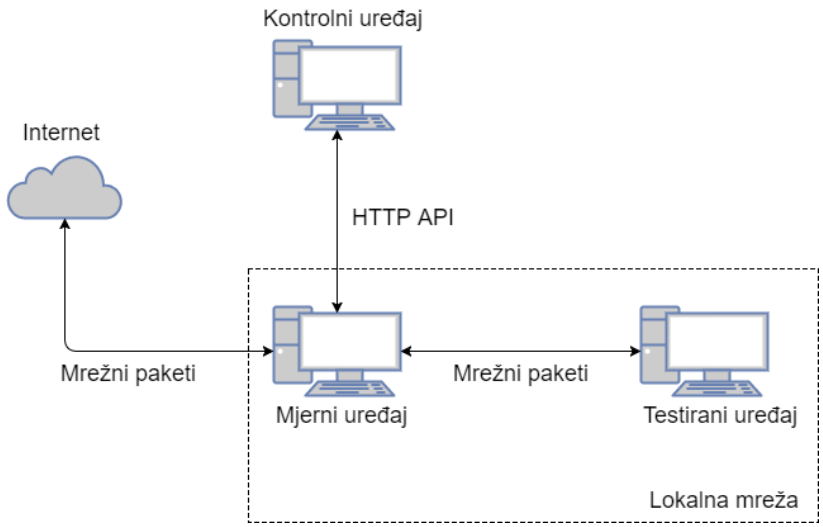
5. ANALIZA ISPRAVNOSTI IMPLEMENTIRANE APLIKACIJE

U ovom poglavlju analizirani su rezultati koje pruža implementirani sustav mjerenja. Predstavljeno je testno okruženje sustava i način provođenja testova. Prikazani su rezultati mjerenja i analizirana je njihova preciznost u odnosu na referentna mjerenja.

5.1. Testna okolina i testni slučajevi

Cilj testiranja provedenog u sklopu ovog rada je utvrđivanje ispravnosti aplikacije za mjerenje usporedbom rezultata ove aplikacije i korištene referentne aplikacije. Za referencu je korišten *Wireshark* mrežni analizator. Rezultati su prikupljeni u kontroliranoj testnoj okolini. Testna okolina sustava mjerenja, prikazana na slici 5.1., sastoji se iz tri glavne komponente:

- Testirani uređaj – uređaj čija se komunikacija analizira radi dobivanja QoS metrika
- Mjerni uređaj – prosljeđuje pakete između vanjske mreže i testiranog uređaja, pri tome filtrirane pakete dostavlja aplikaciji za mjerenje.
- Kontrolni uređaj – uređaj pomoću kojeg korisnik upravlja aplikacijom. Koristi API definiran u poglavlju 4. za slanje naredbi i dohvaćanje rezultata.



Slika 5.1. *Testna okolina aplikacije z mjerenje*

Slika 5.1. predstavlja konceptualni prikaz okoline sustava mjerenje. Kod konkretne implementacije sustava, prikazane komponente ne moraju biti fizički uređaji. Kod testiranja obavljenog u sklopu ovog rada, mjerni uređaj je virtualni uređaj s *Ubuntu Server* operacijskim sustavom i dva mrežna sučelja: lokalno, prema testiranom uređaju, i vanjsko, prema pristupnoj točki Interneta. Ovaj virtualni uređaj konfiguriran je tako da prosljeđuje mrežni promet s lokalnog sučelja na vanjsko i obrnuto. Tako je testiranom uređaju omogućen pristup vanjskoj mreži, dok

mjerni uređaj ima uvid u sav promet razmijenjen između testiranog uređaja i mreže. Računalo na kojem je pokrenut virtualni mjerni uređaj ima ulogu kontrolnog uređaja. Za komunikacija s aplikacijom pokrenutom na mjernom uređaju korištena je *Curl* aplikacija za kreiranje HTTP poruka [33]. Kontrolno računalo također ima ulogu izvora testnih A/V sadržaja.

Aplikacija za mjerenje testirana je tako da je s kontrolnog računala, pomoću *VLC Media Player* aplikacije propušten testni sadržaj na testirani uređaj. Testni sadržaj sastoji se iz video sekvence kodirane u H.264 formatu, rezolucije 1280 x 720 elemenata slike, brzine 25 sličica u sekundi, te audio sekvence MP3 formata, bitske brzine 320 *kb/s*. Ukupna bitska brzina testnog sadržaja iznosi 3325 *kb/s*. Rezolucija reproduciranog videa skalirana je na polovinu osnovne rezolucije. Ovim je smanjen zahtjev na propusnost mreže kako bi bila izbjegnuta pojava spontanih grešaka pri slanju. *VLC Media Player* podržava više načina mrežnog strujanja sadržaja. U sklopu ovog rada korišteno je RTP/UDP strujanje za test RTP metrika, te HTTP strujanje za test TCP metrika.

Mjerni uređaj, postavljen u sredini konekcije između kontrolnog i testiranog uređaja, prikuplja i analizira propuštene pakete. Kontrolni uređaj prikuplja metrike QoS-a od mjernog uređaja nakon što mjerenje završi. Istovremeno, na testiranom uređaju pokrenut je *Wireshark* mrežni analizator. Ovaj program prikuplja pakete pristigle na testirani uređaj, nakon čega se analiziraju QoS metrike. Metrike dobivene analizom programa *Wireshark* korištene su kao referenca pri testiranju ispravnosti aplikacije za mjerenje izrađene u sklopu ovog rada.

Induciranje grešaka u mreži postignuto je korištenjem *Clumsy* [34] aplikacije koja, posredstvom *Windivert* [35] biblioteke omogućuje manipulaciju mrežnim prometom. Ova aplikacija podržava odbacivanje, dupliciranje i promjenu redosljeda paketa, te induciranje kašnjenja i prepisivanje sadržaja paketa. Odbacivanje, duplikacija i promjena redoslijeda paketa obavlja se na kontrolnom računalu, pri slanju testnog sadržaja. *Clumsy* aplikacija omogućava postavljanje tekstualnog filtra koji definira ciljane paketa, te vjerojatnosti nastanka greške.

Testiranje provedeno u sklopu ovog rada obuhvaća sljedeće testne slučajeve:

- Mjerenje količine prometa
- Mjerenje izgubljenih RTP paketa
- Mjerenje dupliciranih RTP paketa
- Mjerenje RTP paketa u krivom redosljedu
- Mjerenje RTP varijacije kašnjenja
- Mjerenje izgubljenih TCP paketa

- Mjerenje dupliciranih TCP paketa
- Mjerenje TCP paketa u krivom redosljedu

Svaki od testnih slučajeva provodi se sljedećim postupkom:

1. Slanjem HTTP POST poruke s kontrolnog računala nalaže se aplikaciji da započne mjerenje vezano uz konkretan testni slučaj. Istovremeno se na mjernom računalu pokreće prikupljanje paketa *Wireshark* analizatorom.
2. S kontrolnog računala se započne strujanje testnog video sadržaja na testirano računalo.
3. Ovisno o testnom slučaju, u testni komunikacijski tok se induciraju greške korištenjem *Clumsy* aplikacije. Vjerojatnost nastanka greške postavljena je na 10% za sve testne slučajeve.
4. Nakon završetka reprodukcije testnog sadržaja, slanjem HTTP DELETE poruke s kontrolnog računala zaustavlja se pokrenuto mjerenje. Na testiranom računalu se zaustavlja *Wireshark* analizator i prikupljeni paketi se pohranjuju.
5. Slanjem GET poruka sa zadanim rasponom trajanja mjerenja, dohvaćaju se metrike prikupljene aplikacijom.

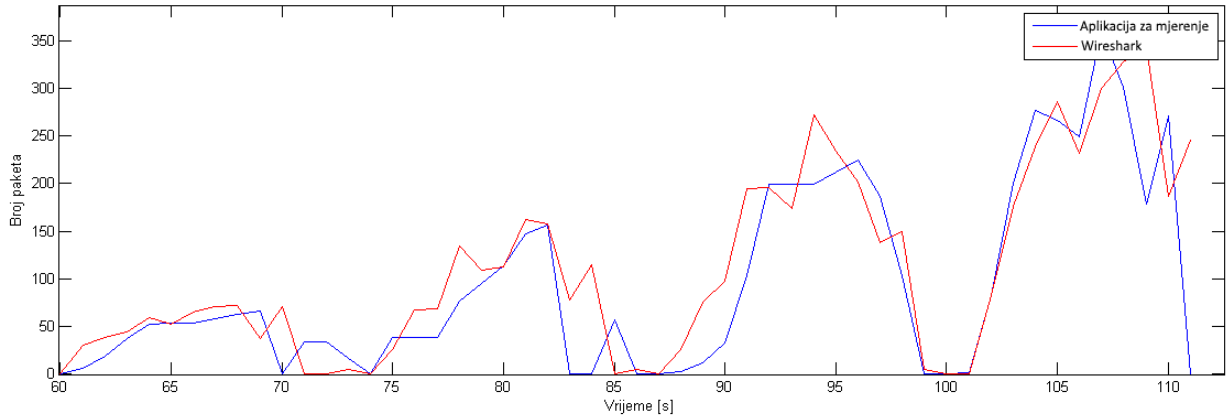
Od navedenog postupka postoje iznimke. Kod mjerenja količine prometa ne stvaraju se namjerne greške u mreži. Kod testnih slučajeva gdje je mjerena TCP konekcija, strujanje je potrebno započeti prije samog mjerenja, te je potrebno odrediti komunikacijski *port* koji je odabralo klijentsko računalo. Sam početak TCP konekcije, vrlo je teško „uhvatiti“ mjerenjem, budući da klijent nasumično bira komunikacijski *port*.

5.2. Prikaz rezultata

Testni slučaj mjerenja količine prometa sastoji se iz uzastopnog mrežnog strujanja video sadržaja različitih rezolucija, koji generiraju različite razine prometa u mreži. Testni video sadržaj uzastopno je skaliran na 25%, 50% i 75% osnovne rezolucije, te na osnovnu rezoluciju i poslan testiranom uređaju. Mjerenja količine poslanih paketa prikupljena mjernom aplikacijom uspoređena su sa statistikama *Wireshark* analizatora na slici 5.2.

Vidljivo je da se krivulje koje prikazuju mjerenja približno podudaraju. Jedan od razloga odstupanja je činjenica da mjerna aplikacija i *Wireshark* uzorkuju mrežni promet u različitim trenucima. Dok *Wireshark* ima mogućnost prikaza statistika prometa s rezolucijom na razini mikrosekundi, mjerna aplikacija prikuplja i pohranjuje vrijednosti mjerenja tek svake sekunde.

Aplikacija je implementirana s ovim ograničenjem zbog toga što pohranjene podatke pruža preko mrežnog API-ja. Stoga se postavljanjem perioda uzorkovanja na jednu sekundu nastoj smanjiti količina razmjenjenih podataka, uz održavanje preciznosti mjerenja.



Slika 5.2. *Mjerenje količine prometa aplikacijom za mjerenje QoS (plavo) i Wireshark analizatorom (crveno)*

Sljedećim testnim slučajevima provjerena je ispravnost mjerenja RTP grešaka tj. izgubljenih, dupliciranih, i paketa u krivom redoslijedu. *Wireshrk* ima mogućnost prikupljanja ovih metrika, međutim one su prikazane sumarno za analiziranu RTP sesiju. Metrike se mogu prikazati u *Wireshark-u* pod karticom *Telephony->RTP->Stream Analysis.* Aplikacija za mjerenje može ove metrike prikazati sumarno, ili kao niz vrijednosti uzorkovanih s periodom od jedne sekunde. Tablica 5.1. prikazuje usporedbu sumarnih metrika *Wireshark-a* i aplikacije za mjerenje.

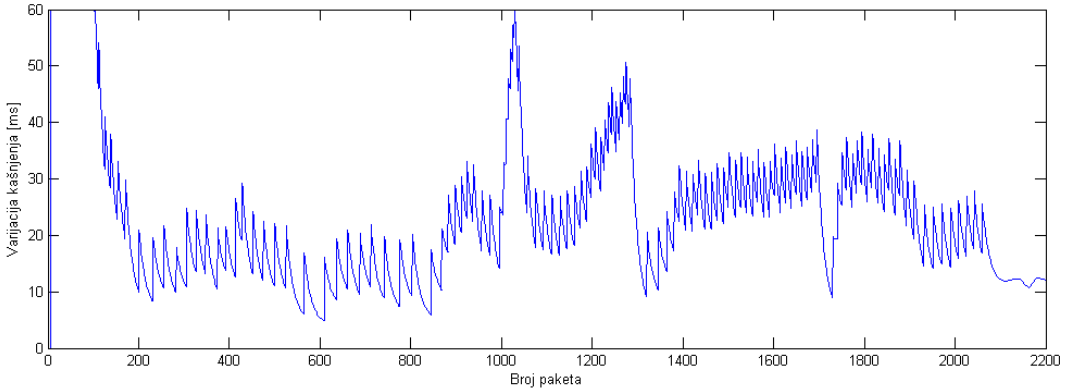
Tablica 5.1. *Sumarni prikaz mjerenja RTP grešaka*

Metrika	Aplikacija za mjerenje QoS	Wireshark	
		Ime analiziranog podatka	Vrijednost
Izgubljeni RTP paketi	163	Lost	163
Duplicirani RTP paketi	155	Lost	-155
RTP paketi krivog redosljeda	140	Seq Errs	280

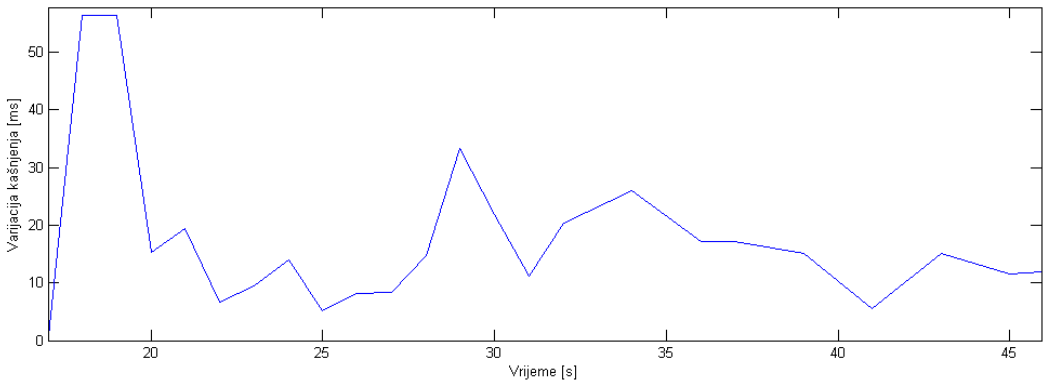
Iz tablice 5.1. vidljivo je da metrike dobivene aplikacijom odgovaraju vrijednostima *Wireshark* analize. Broj izgubljenih paketa, odnosno broj preskočenih RTP sekvenci odgovara razlici očekivanog broja paketa i broja primljenih koju računa *Wireshark*. Broj dupliciranih paketa odgovara negativnom broju izgubljenih kod *Wireshark-a* (negativan broj izgubljenih označava da je pristiglo više paketa nego što je potrebno). Broj paketa u krivom redoslijedu odgovara polovini broja grešaka sekvence koje detektira *Wireshark*. Svaki paket u krivom redoslijedu uzrokuje dvije greške sekvence: jednu sekvencu veću od očekivane, i jednu manju.

Tijekom testiranja utvrđen je i utjecaj grešaka u prijenosu na reproducirani sadržaj. Budući da RTP protokol nema mehanizam oporavka od izgubljenih podataka, pojava izgubljenih paketa uzrokuje greške pri dekodiranju video sadržaja i pojavu *blocking* artefakata tijekom reprodukcije. Pojava dupliciranih paketa nema utjecaj na sadržaj, ovi paketi se jednostavno odbacuju na prijemnoj strani. Krivi redoslijed paketa ima za posljedicu duže zadržavanje podataka u prijemnom međuspremniku, što uzrokuje povremeno zamrzavanje reproduciranog videa.

Testnim slučajem RTP varijacije kašnjenja provjerena je ispravnost vremenskih metrika aplikacije za mjerenje. Prilikom strujanja testnog sadržaja nisu inducirane pogreške, promatrana je varijacija kašnjenja RTP paketa između kontrolnog i testnog računala. Rezultati su uspoređeni s *Wireshark* analizom na slici 5.3.



(a)



(b)

Slika 5.3. *Usporedba rezultata (a) Wireshark analize i (b) analize stvorenom aplikacijom*

Vidljiv je određeni stupanj korelacije između izmjerenih i referentnih mjerenja. Postoje čimbenici koji uzrokuju odstupanje. Domene grafova se razlikuju: kod referentnog mjerenja domenu čine redni brojevi pristiglih paketa, dok je mjerenje aplikacije prikazano u vremenskoj domeni. Između broja paketa i vremena postoji korelacija, međutim oni nisu linearno ovisni

(upravo zbog varijacije kašnjenja). Drugi čimbenik je rezolucija kojom su uzorkovani podaci. Dok *Wireshark* prikazuje varijaciju kašnjenja za svaki paket, aplikacija za mjerenje prikazuje vrijednosti uzorkovane s periodom jedne sekunde.

Sljedećim testnim slučajevima provjerena je ispravnost mjerenja prethodno prikazanih grešaka kod TCP komunikacije. *Wireshark* posjeduje algoritme prema kojima detektira izgubljene TCP pakete, TCP retransmisije (duplikate) i krivi redoslijed paketa. Broj ovih grešaka za promatrani TCP tok moguće je vidjeti postavljanjem *tcp.analysis* filtara prikaza. Usporedba sumarnih metrika *Wireshark-a* i aplikacije za mjerenje nalazi se u tablici 5.2.

Tablica 5.2. *Sumarni prikaz mjerenja TCP grešaka*

Metrika	Aplikacija za mjerenje QoS	Wireshark	
		Ime analiziranog podatka	Vrijednost
Izgubljeni TCP paketi	76	tcp.analysis.lost_segment	69
Duplicirani TCP paketi	92	tcp.analysis.retransmission	92
TCP paketi krivog redosljeda	108	tcp.analysis.out_of_order	108

Kod analize TCP protokola dolazi do odstupanja zbog korištenja različitih algoritama analize. *Wireshark* alat za analizu TCP paketa [36] ima vrlo jednostavan algoritam detekcije izgubljenih TCP paketa: svaka sekvenca veća od očekivane znači izgubljeni paket. Postoji mogućnost da preskočeni paket nije izgubljen, već je došlo do promjene redoslijeda paketa. Dodatno, preskočena sekvenca može značiti više od jednog izgubljenog paketa. Kako je već opisano, aplikacija za mjerenje koristi najčešće TCP mehanizme signalizacije izgubljenih podataka (duplicirani ACK i SACK) za detekciju izgubljenih paketa. Ovaj algoritam osigurava da aplikacija za mjerenje detektira gubitak podataka na isti način kao i TCP modul prijemnika. Prilikom testa mjerenja broja dupliciranih paketa i paketa u krivom redosljedu, rezultati dobiveni aplikacijom za mjerenje i *Wireshark* analizom se podudaraju.

Pregledom reproduciranog u zadnja tri testna slučaja utvrđen je utjecaj TCP pogrešaka na kvalitetu sadržaja. TCP mehanizam retransmisije osigurao je oporavak konekcije od gubitka paketa. Međutim, oporavak od retransmisije uzrokovao je krivi redoslijed paketa i povećano kašnjenje. Stoga, kod većeg broja izgubljenih paketa (provedeni su testovi gdje je vjerojatnost odbacivanja paketa povećana s 10% na 15%), dolazi do zamrzavanja videa zbog dužeg perioda popunjavanja međuspremnika za dekodiranje. Duplicirani paketi nemaju značajan utjecaj, dok krivi redosljed može imati slične posljedice kao i gubitak paketa.

6. ZAKLJUČAK

Ovim radom obrađen je problem mjerenja kvalitete usluge prilikom prijenosa digitalnih televizijskih sadržaja putem IP mreže. Cilj rada bio je predložiti metode mjerenja parametara QoS IP DTV usluge, implementirati i testirati predložene metode, te omogućiti integraciju sustava mjerenja s drugim automatiziranim sustavima. Glavno ograničenje sustava mjerenja je što sustav ima ulogu klijenta u mreži pružatelja IP DTV usluge. Stoga je potrebno procijeniti QoS parametre mreža na osnovu analize mrežnog prometa na sučelju IP DTV klijenta. Ova analiza mrežnog prometa sastoji se u očitavanju i interpretaciji podataka koji se nalaze u zaglavljima protokola mrežnih paketa.

Ispitani su prihvaćeni standardi na području IP DTV usluga i identificirani komunikacijski protokoli korišteni za prijenos digitalnih audio i video sadržaja. Zaključeno je da su najznačajniji komunikacijski protokoli u prijenosu DTV sadržaja: RTP preko UDP, i TCP. Ovi protokoli koriste usluge usmjeravanja podataka IP protokola. Istovremeno implementiraju mehanizme identificiranja i ispravljanja grešaka nastalih pri usmjeravanju paketa. Analizom podataka koje daju ovi mehanizmi identifikacije grešaka, aplikacija za mjerenje implementirana u sklopu ovog rada procjenjuje QoS parametre mreže. Stoga su ovim radom ispitani podaci sadržani u zaglavljima ovih protokola i njihova funkcija. Također su predstavljeni najvažniji QoS parametri IP mreža: propusnost, kašnjenje, varijacija kašnjenja, i greške (gubitak, duplikacija i krivi redoslijed paketa), te je prikazan njihov utjecaj na prijenos DTV sadržaja. Na osnovu prikazanih QoS parametara i analize funkcioniranja komunikacijskih protokola definirane su ciljane metrike koje implementirani sustav mjerenja treba pružiti.

Ispitana je mogućnost dizajniranja sustava mjerenja korištenjem nekog od postojećih programa za analizu mrežnog prometa. U svrhu odabira najpogodnijeg rješenja predstavljen je niz programa i programskih biblioteka za prikupljanje i analizu mrežnog prometa, te su postavljeni kriteriji testiranja. Kriteriji uključuju: ciljane metrike, sučelje i format podataka, podržane platforme i licencu za korištenje. Predstavljena rješenja su testirana i ocijenjena prema zadanim kriterijima. Kao najpogodnije rješenje na kojem je zasnovan sustav mjerenja odabrana je *Libtins* programska biblioteka za prikupljanje i disekciju mrežnih paketa.

Rad sustava mjerenja implementiran je kroz rad tri modula. Modul za mjerenje zasnovan je na *Libtins* biblioteci. Ovaj modul prikuplja mrežne pakete, te koristi algoritme analize prikazane ovim radom u svrhu procjene parametara QoS mreže. Modul za pohranu podataka uzorkuje mjerenja u pravilnim intervalima i sprema očitane rezultate na tvrdi disk. HTTP API modul

komunicira s vanjskim sustavima pomoću unaprijed definiranog API-ja zasnovanog na razmjeni HTTP poruka. Na ovaj način korisnik može zadavati naredbe sustavu mjerenja, te dohvaćati izmjerene vrijednosti. Tipovi razmjenjenih poruka i njihova semantika predstavljene su u radu.

Na posljetku provedeno je testiranje ispravnosti aplikacije za mjerenje tako što su izmjerene vrijednosti parametara QoS uspoređene s referentnim mjerenjima. Za prikupljanje referentnih mjerenja korišten je *Wireshark* mrežni analizator, kao jedan od najšire prihvaćenih programa za prikupljanje i analizu mrežnog prometa. Postavljena je testna okolina unutar koje je simuliran slučaj korištenja IP DTV usluge, strujanjem sadržaja s računala koje predstavlja vanjsku mrežu na računalo u lokalnoj mreži. Pri tome, sustav mjerenja je postavljen između ova dva računala, te ima pristup svom mrežnom prometu između njih. Strujanje je izvedeno korištenjem RTP i TCP protokola. Prilikom strujanja, u mrežni promet inducirane su greške. Usporedbom rezultata mjerenja s referentnim, u većini provedenih testova utvrđena je podudarnost. Kod određenih metrika postoji odstupanje zbog razlike algoritama analize, gdje se implementirani algoritam pokazao kao bolje rješenje.

Testiranjem je dokazana je ispravnost rada aplikacije. Međutim, sustav mjerenja u trenutnom stanju moguće je proširiti podrškom za nove protokole, kao i poboljšanjem algoritama analize postojećih protokola. Primjer standardiziranog protokola koji nije pokriven ovim radom je FLUTE protokol. Algoritam analize TCP protokola moguće je unaprijediti implementacijom algoritma koji detektira gubitak na osnovu kašnjenja ACK paketa. Trenutni algoritam za prepoznavanje izgubljenih paketa zasnovan je na mehanizmu dupliciranih ACK segmenata. Algoritmi analize kašnjenja i varijacije kašnjenja mogu biti unaprijeđeni uzorkovanjem vremena paketa na nivou jezgre operacijskog sustava (moguće kod *Wnipcap* biblioteke)

LITERATURA

[1] „What Is the Difference Between Internet TV and Internet Protocol Television (IPTV)?“, *eBay*, [Na internetu]. Dostupno na: http://www.ebay.com/gds/What-Is-the-Difference-Between-Internet-TV-and-Internet-Protocol-Television-IPTV-/10000000177630834/g.html. [Pristupljeno: 16-kol-2017].

[2] „Hybrid Broadcast Broadband TV“, ETSI, TS 102 796, kol. 2016.

[3] „Manpage of TCPDUMP“. [Na internetu]. Dostupno na: http://www.tcpdump.org/tcpdump_man.html. [Pristupljeno: 16-kol-2017].

[4] „Wireshark User’s Guide“. [Na internetu]. Dostupno na: https://www.wireshark.org/docs/wsug_html_chunked/. [Pristupljeno: 16-kol-2017].

[5] „ettercap(8) - Linux man page“. [Na internetu]. Dostupno na: https://linux.die.net/man/8/ettercap. [Pristupljeno: 16-kol-2017].

[6] „EtherApe, a graphical network monitor“. [Na internetu]. Dostupno na: http://etherape.sourceforge.net/. [Pristupljeno: 16-kol-2017].

[7] „Manpage of PCAP“. [Na internetu]. Dostupno na: http://www.tcpdump.org/manpages/pcap.3pcap.html. [Pristupljeno: 16-kol-2017].

[8] „C++ packet sniffing and crafting library“. [Na internetu]. Dostupno na: http://libtins.github.io/. [Pristupljeno: 16-kol-2017].

[9] H. Benoit, *Digital Television Satellite, Cable, Terrestrial, IPTV, Mobile TV in the DVB Framework*, Elsevier, SAD, 2008.

[10] „Requirements for Internet Hosts -- Communication Layers“, IETF, RFC 1122, lis. 1989.

[11] „Digital Video Broadcasting (DVB); Transport of MPEG-2 TS Based DVB Services over IP Based Networks“, ETSI, TS 102 034, tra. 2016.

[12] „Detailed Information About How IPTV Works?“, *Calisia.net*, 11-velj-2016. [Na internetu]. Dostupno na: http://www.calisia.net/detailed-information-about-how-iptv-works/.

[13] „Digital Video Broadcasting (DVB); DVB-IPTV Profiles for TS 102 034“, ETSI, TS 102 826, tra. 2016.

[14] „Digital Video Broadcasting (DVB); Guidelines for the implementation of DVB-IPTV Phase 1 specifications; Part 5: Content Download Service (CDS)“, ETSI, TS 102 542-5, tra. 2016.

[15] W. R. Stevens i K. R. Fall, *TCP/IP Illustrated, Volume 1: The Protocols*, Addison-Wesley, SAD, 2011.

[16] „Internet Protocol“, IETF, RFC 791, ruj. 1981.

[17] „Internet Control Message Protocol, DARPA Internet Program, Protocol Specification“, IETF, RFC 792, ruj. 1981.

[18] „Transmission Control Protocol, DARPA Internet Program, Protocol Specification“, IETF, RFC 793, ruj. 1981.

[19] „User Datagram Protocol“, IETF, RFC 768, kol. 1980.

[20] „RTP: A Transport Protocol for Real-Time Applications“, IETF, RFC 3550, srp. 2003.

[21] „RTP Payload Format for MPEG1/MPEG2 Video“, IETF, RFC 2250, sij. 1998.

[22] „RTP Profile for Audio and Video Conferences with Minimal Control“, IETF, RFC 3551, srp. 2003.

[23] „Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 3: Implementation guidelines“, ISO/IEC, PDTR 23009-3, sij. 2013.

[24] „Development/LibpcapFileFormat - The Wireshark Wiki“, 16-kol-2017. [Na internetu]. Dostupno na: https://wiki.wireshark.org/Development/LibpcapFileFormat. [Pristupljeno: 16-kol-2017].

[25] „Manpage of PCAP-FILTER“, 16-kol-2017. [Na internetu]. Dostupno na: http://www.tcpdump.org/manpages/pcap-filter.7.html. [Pristupljeno: 16-kol-2017].

[26] „etterlog(8) - Linux man page“, 16-kol-2017. [Na internetu]. Dostupno na: https://linux.die.net/man/8/etterlog. [Pristupljeno: 16-kol-2017].

[27] „etherape(1): graphical network traffic browser - Linux man page“, 16-kol-2017. [Na internetu]. Dostupno na: https://linux.die.net/man/1/etherape. [Pristupljeno: 16-kol-2017].

[28] „tshark - The Wireshark Network Analyzer 2.4.1“, 16-kol-2017. [Na internetu]. Dostupno na: https://www.wireshark.org/docs/man-pages/tshark.html. [Pristupljeno: 16-kol-2017].

[29] L. M. Garcia, „Programming with Libpcap - Sniffing th Network From Our Own Application“, *HAKIN9*, sv. 3, izd. 2/2008, str. 38–46, velj-2008.

[30] „Tutorial: Adding new protocols“. [Na internetu]. Dostupno na: http://libtins.github.io/tutorial/new_protocols/. [Pristupljeno: 07-ruj-2017].

[31] „TCP Congestion Control“, IETF, RFC 5681, ruj. 2009.

[32] „A Conservative Loss Recovery Algorithm Based on Selective Acknowledgment (SACK) for TCP“, IETF, RFC 6675, kol. 2012.

[33] „curl - How To Use“. [Na internetu]. Dostupno na: https://curl.haxx.se/docs/manpage.html#--post301. [Pristupljeno: 26-kol-2017].

[34] „clumsy, an utility for simulating broken network for Windows Vista / Windows 7 and above“. [Na internetu]. Dostupno na: https://jagt.github.io/clumsy/. [Pristupljeno: 27-kol-2017].

[35] „WinDivert 1.1: Windows Packet Divert“. [Na internetu]. Dostupno na: https://reqrypt.org/windivert.html. [Pristupljeno: 27-kol-2017].

[36] „7.5. TCP Analysis“. [Na internetu]. Dostupno na: https://www.wireshark.org/docs/wsug_html_chunked/ChAdvTCPAnalysis.html. [Pristupljeno: 03-ruj-2017].

SAŽETAK

Ovim radom ispitan je problem mjerenja QoS u IP DTV sustavima s gledišta klijenta. Kao rješenje, predložena je i implementirana računalna aplikacija za mjerenje QoS parametara kroz analizu podataka koje pružaju mrežni protokoli. Predstavljeni su komunikacijski protokoli koji su dio standarda IP DTV usluga, ispitan je njihov rad i podaci koje sadržavaju. Predstavljen je utjecaj parametara QoS na rad IP DTV usluga. Definirane su metrike QoS koje treba pružiti aplikacija za mjerenja i način njihovog dobivanja analizom protokola. Postavljeni su kriteriji implementacije prema kojima je testirano nekoliko postojećih rješenja za prikupljanje i analizu mrežnih paketa, te je najpogodnije rješenje integrirano u aplikaciju za mjerenje. Mjerenja su implementirana algoritmima za analizu zasnovanim na radu mrežnih protokola. Definiran je HTTP API pomoću kojeg aplikacija komunicira s okolinom. Ispravnost mjerenja testirana je usporedbom rezultata mjerenja s rezultatima koje daje referentni program za analizu.

Ključne riječi: DTV, QoS, IPTV, DVB, mrežni protokoli,

QOS MEASUREMENT FOR IP DTV DEVICES

ABSTRACT

This work explores the problem of QoS measurement in IP DTV services, from the client standpoint. As a solution, a computer application for measurement of QoS parameters through analysis of data provided by network protocols, is proposed. Communication protocols, which are a part of the IP DTV services standard, are presented, and their function and contained data are explored. Influence of QoS parameters on the function of IP DTV services is shown. Metrics provided by the measurement application and methods of their extraction through protocol analysis are defined. Implementation criteria are defined, and multiple existing software solutions for collection and analysis of network packets are tested using this criteria. The most suitable solution is integrated into the measurement application. Measurements are implemented using analysis algorithms based on the function of network protocols. An HTTP API, through which the application communicates with the environment is defined. Measurement accuracy is tested by comparing the measurement results with results provided by the reference analysis software.

Keywords: DTV, QoS, IPTV, DVB, network protocols,

ŽIVOTOPIS

Milan Ivošević rođen je 28.06.1993. u Glini, Republika Hrvatska. Osnovnu školu završio je u Vukovaru, nakon čega je upisao opći smjer Gimnazije Vukovar. Godine 2012. upisao je preddiplomski studij računarstva na Elektrotehničkom fakultetu Sveučilišta J. J. Strossmayera u Osijeku. U svibnju 2014. Godine, kao student druge godine, dobio je povodom 36. godišnjice Elektrotehničkog fakulteta *priznanje za postignut uspjeh u studiranju*. Godine 2015. završio je preddiplomski studij izradom završnog rada na temu „Protokoli usmjeravanja u VANET mrežama“. Iste godine upsao je diplomski studij računarstva, smjer programsko inžinjerstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku.

_____

Milan Ivošević