UNIVERZITET U SARAJEVU
ELEKTROTEHNIČKI FAKULTET
ODSJEK ZA TELEKOMUNIKACIJE

Dizajn i implementacija WebRTC modela u NS-3 simulatoru

SIMULACIJA PROCESA U TELEKOMUNIKACIJSKIM MREŽAMA

- DRUGI CIKLUS STUDIJA -

Studenti:
Amila Čengić, Amina Omerčević, Džejlana Konjalić
Denin Mehanović, Emrah Dragolj

Profesor:
Vanr.prof.dr Miralem Mehić

Sarajevo, februar 2025.

Sadržaj

Poglavlje 1

Opis rješenja

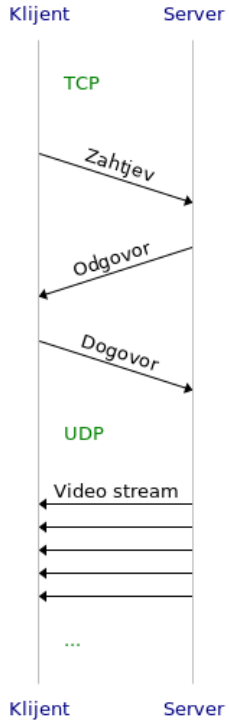
Kroz prvi dio projektnog zadatka objašnjeni su teorijski koncepti komunikacije u realnom vremenu putem *WebRTC-a (Web Real-Time Communication)*, analizirani su izazovi i standardi vezani za ovu tehnologiju, te su razmotrene mogućnosti njene implementacije u NS-3 mrežnom simulatoru. WebRTC je predstavljen kao ključna tehnologija koja omogućava *peer-to-peer (P2P)* razmjenu audio, video i podatkovnog sadržaja putem internetskih preglednika bez potrebe za dodatnim softverom.

Tokom istraživanja i analize dostupnih resursa, nisu pronađena postojeća rješenja za implementaciju predloženog WebRTC modela unutar trenutno dostupnih mrežnih simulatora. Nije pronađena niti dokumentacija niti unaprijed definisani moduli koji bi omogućili direktnu integraciju funkcionalnosti WebRTC-a, što uključuje signalizaciju, uspostavljanje peer-to-peer konekcija, prenos medijskih podataka putem RTP-a, kao ni podršku za sigurnosne protokole, što je dovelo do potrebe za prilagođenim modeliranjem. U ovom dijelu rada, fokus je na razvoju WebRTC modela u NS-3, pri čemu su implementirane ključne funkcionalnosti poput uspostavljanja peer-to-peer konekcije i prenosa podataka koristeći RTP preko UDP-a.

Implementirani model sastoji se od **dva čvora – klijenta i servera**, povezanih putem *peer-to-peer* veze. Komunikacija se inicijalno uspostavlja *TCP three-way handshakeom*, nakon čega slijedi prenos RTP paketa korištenjem UDP protokola.

Model implementiran u NS-3 omogućava evaluaciju različitih parametara WebRTC komunikacije, uključujući gubitak paketa i kašnjenje. Prikaz rezultata simulacija omogućen je kroz različite dijagrame i histogram, kako bi se omogućila detaljna analiza performansi predloženog rješenja.

Konačno, simulacija je pružila dublji uvid u ponašanje WebRTC tehnologije u različitim mrežnim uslovima, omogućila prepoznavanje mogućih izazova i doprinijela poboljšanju njenih performansi. Zaključeno je da NS-3 pruža optimalno okruženje za testiranje WebRTC modela, te da je njegov razvoj koristan za buduće implementacije i poboljšanja ove tehnologije u realnim aplikacijama.



Slika 1.1: TCP three-way handshake dijagram

1.1 Dizajn

U sklopu korištenih skripti *sptm-client.cc* i *sptm-sink.cc* definisana je struktura ***RtpHeader*** koja definiše ključna polja RTP zaglavlja, što je neophodno za pravilno upravljanje, isporuku i sinhronizaciju podataka u aplikacijama u realnom vremenu. Također, unutar *sptm-client.cc* definisan je konstruktor *RtpHeader()* koji inicijalizira sve članove strukture sa zadanim vrijednostima. *SptmClient* šalje pakete, dok *SptmSink* prima te pakete, procesuira ih i ispisuje detalje o prijemu, čime omogućavaju testiranje prilagođenih protokola u simuliranoj mreži.

U mrežnim simulacijama koje se provode u sklopu ns-3 okvira, važno je simulirati realne uslove u kojima mreža može doživjeti različite vrste grešaka, kao što su gubitak paketa ili oštećenje podataka. Jedan od načina za postizanje ove simulacije jest korištenje modela grešaka, koji omogućava uvođenje takvih grešaka u prenos podataka. U ovom kodu, prikazano je korištenje dva različita modela grešaka: opšti *ErrorModel* i specifični *ListErrorModel*.

ErrorModel je osnovni model grešaka koji se koristi za simulaciju grešaka prilikom prijema paketa na mrežnim uređajima. U ovom kodu, korištenjem objekta *ObjectFactory*, inicijaliziran je model greške na temelju unaprijed definisanog tipa (koji se definiše pomoću *errorModelType*). Ovaj tip može biti bilo koji model greške, a odabir tipa ovisi o specifičnim zahtjevima simulacije. Kroz metodu *SetTypeId* objektu factory postavlja se tip modela, a zatim se pomoću funkcije *Create<ErrorModel>()* stvara konkretan objekat modela greške. Nakon što je model greške kreiran, on se postavlja putem metode *SetAttribute*. Atribut koji se postavlja je *"ReceiveErrorModel"*, a njegovu vrijednost čini pokazivač na kreirani model greške, što znači da će mrežni uređaj prilikom prijema paketa koristiti ovaj model za simulaciju grešaka.

S druge strane, *ListErrorModel* predstavlja napredniji tip modela greške koji omogućava eksplicitno definisanje paketa koji će biti odbačeni tokom simulacije. Ovdje se prvo generiše popis ID-eva paketa koji će biti odbačeni, pomoću funkcije *getPacketIDsToLose(0, 50)*, koja vraća listu paketa s ID-evima između 0 i 50. Nakon što je lista paketa za odbacivanje pripremljena, stvara se instanca *ListErrorModel* objekta i pomoću metode *SetList* ta lista paketa postavlja se u model greške. Na taj način, *ListErrorModel* omogućava precizno kontrolisanje koji paketi će biti odbačeni tokom simulacije, što može biti korisno za testiranje mrežnih protokola u uvjetima gubitka paketa.

Funkcija ***getPacketIDsToLose*** je implementirana s ciljem da omogući odbacivanje paketa na temelju slučajnog odabira, na osnovu implementiranog Error modela koji je prethodno opisan. Funkcija koristi objekat *UniformRandomVariable* za generisanje nasumičnih brojeva po uniformnoj distribuciji, s minimalnim i maksimalnim vrijednostima koje su zadane kao argumenti funkcije. Na temelju tih parametara, generiše se nasumičan broj ponavljanja, koji označava koliko će puta biti odabrani nasumični ID-evi paketa. Ovaj broj ponavljanja se dobija pozivanjem metode *GetInteger()*, koja vraća nasumičan broj unutar zadanog opsega. Svaki generisani ID paketa se dodaje u listu koja na kraju sadrži sve odabrane ID-eve paketa koji će biti odbačeni u simulaciji.

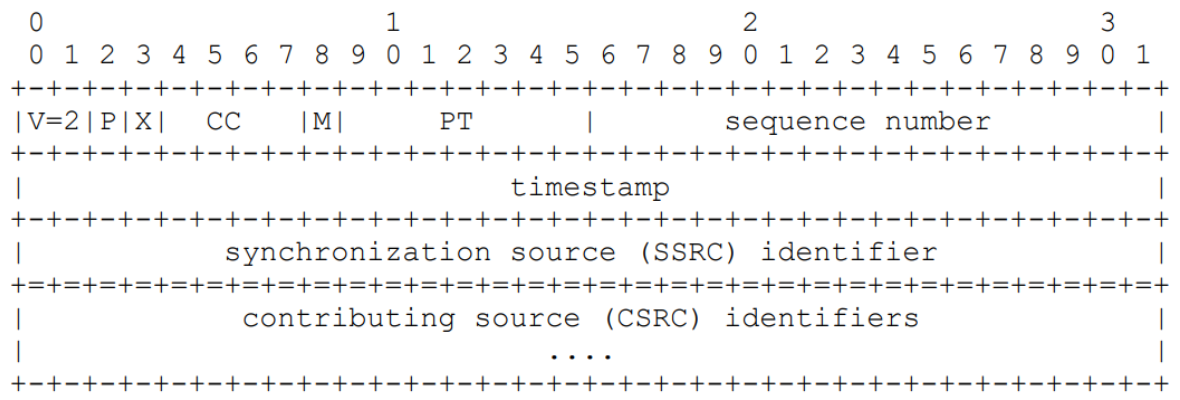
1.2 Three-way handshake

Three-way handshake je proces koji se koristi za uspostavljanje pouzdane veze između klijenta i servera u okviru *Transmission Control Protocol-a (TCP)*. Prvi korak uključuje slanje *SYN (synchronize)* paketa od klijenta ka serveru kako bi inicirao komunikaciju. Server zatim odgovara sa *SYN-ACK (synchronize-acknowledge)* paketom, potvrđujući prijem zahtjeva i predlažući vlastite parametre veze. Klijent potom šalje *ACK (acknowledge)* paket kao konačnu potvrdu, čime se uspostavlja veza i omogućava dvosmjerna komunikacija. Ovaj proces osigurava da su obje strane spremne za razmjenu podataka i da su usklađene u pogledu sekvencijskih brojeva.

1.3 Opis RTP zaglavlja

U simulaciji, podaci se prenose putem RTP (*Real-time Transport Protocol*) zaglavlja, koje omogućava sihronizaciju, identifikaciju i sekvenciranje medijskih paketa. RTP zaglavlje igra ključnu ulogu u pravilnom dekodiranju i rekonstrukciji video i audio podataka, osiguravajući da paketi budu isporučeni u ispravnom redoslijedu i sa minimalnim kašnjenjem. U nastavku je detaljno objašnjen format RTP zaglavlja, uz opis značenja svakog bita.

Standard [3] opisuje zaglavlje RTP-a. Prvih dvanaest okteta u RTP-u uvijek ostaje nepromijenjeno, dok je lista *CSRC* opciona (koristi se samo u slučaju kada se koristi mješač, što u ovoj implementaciji nije urađeno). Na slici 1.2 prikazan je izgled RTP zaglavlja.



Slika 1.2: Izgled RTP zaglavlja

Polja RTP zaglavlja, zajedno s njihovim opisima i funkcijama:

- **Verzija (V) - 2 bita** - Ovo polje označava verziju RTP protokola. Prema ovoj specifikaciji, trenutna verzija je 2. Vrijednost 1 korištena je u prvoj radnoj verziji RTP-a, dok je vrijednost 0 pripadala ranijem protokolu implementiranom u audio alatu "*vat*".
- ***Padding* (P) - 1 bit** - Kada je ovaj bit postavljen, na kraju paketa se nalaze dodatni bajtovi za *padding*, koji nisu dio stvarnog sadržaja. Posljednji bajt tog *padding-a* sadrži informaciju o tome koliko tih bajtova treba zanemariti, uključujući i njega samog. Ovo je potrebno kod određenih metoda enkripcije koje koriste fiksne veličine blokova ili kada se više RTP paketa prenosi unutar jedne jedinice podataka nižeg sloja.
- **Proširenje (X) 1 bit** - Ako je ovaj bit aktiviran, nakon fiksnog zaglavlja mora slijediti tačno jedno proširenje zaglavlja.
- **CSRC brojač (CC) 4 bita** – Ovo polje označava broj CSRC identifikatora koji dolaze nakon fiksnog zaglavlja.
- **Oznaka (M) 1 bit** – Značenje ovog bita određuje profil. Koristi se za označavanje važnih događaja, poput granica kadrova, unutar toka paketa. Profil može dopustiti dodatne oznake ili odrediti da ovaj bit nije prisutan tako što će promijeniti broj bitova u polju tipa korisnog tereta.
- **Tip korisnog tereta (PT) 7 bita** – Polje tipa korisnog tereta određuje format RTP korisnog tereta i način na koji ga aplikacija interpretira. Profil može definisati zadano mapiranje kodova tipa korisnog tereta na njihove odgovarajuće formate. Također, novi kodovi tipa korisnog tereta mogu se dinamički dodavati putem vanjskih mehanizama koji nisu dio RTP-a. RTP izvor može promijeniti tip korisnog tereta unutar iste sesije, no ovo polje se ne bi smjelo koristiti za multipleksiranje različitih medijskih tokova. Prijemnik je obavezan ignorisati pakete s tipovima korisnog tereta koje ne prepoznaje.
- **Redni broj 16 bitova** – Ovaj broj se povećava za jedan sa svakim novim RTP paketom koji se šalje i može pomoći prijemniku da prepozna izgubljene pakete i ponovno uspostavi pravilan redoslijed. Početna vrijednost rednog broja trebala bi biti nasumično odabrana kako bi se otežali napadi temeljeni na poznatim uzorcima otvorenog teksta. Ovo je važno čak i ako izvor sam ne primjenjuje enkripciju, jer paketi mogu prolaziti kroz prevoditelje koji enkripciju koriste.

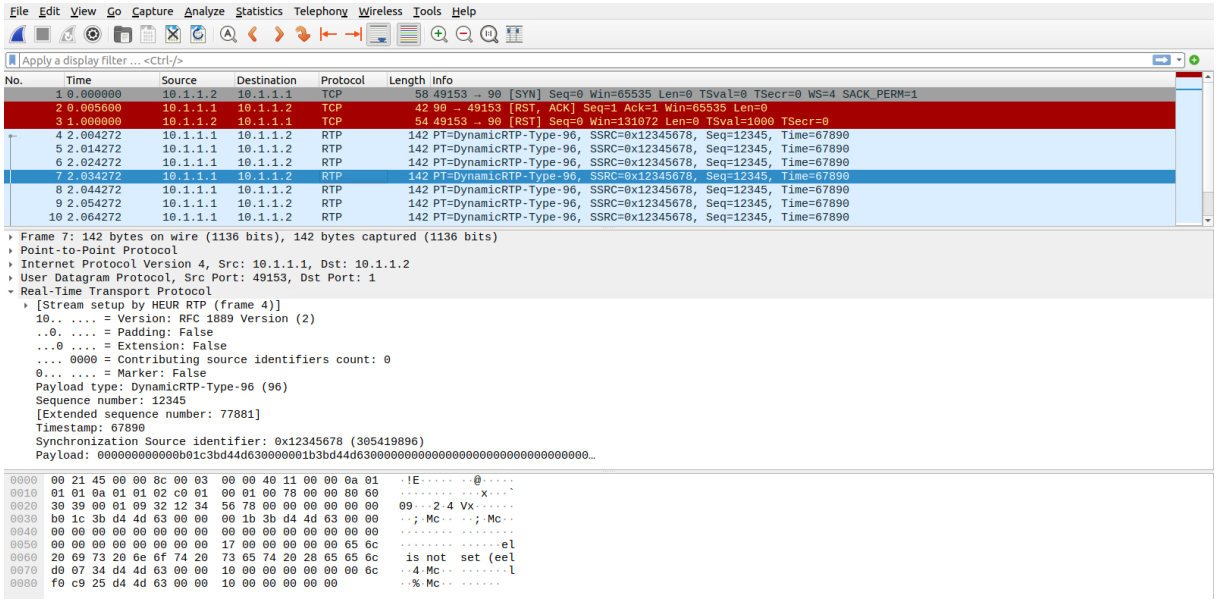
- **Timestamp 32 bita** - Ova oznaka pdražava trenutak uzorkovanja prvog okteta u RTP paketu podataka. Vrijeme uzorkovanja mora biti izvedeno iz *clock-a* koji se povećava monotono i linearno tokom vremena kako bi omogućio sinhronizaciju i izračun kašnjenja paketa (eng. *jitter*). Rezolucija *clock-a* mora biti dovoljna za željenu preciznost sinhronizacije i mjerenje kašnjenja paketa (jedan impuls po video kadru obično nije dovoljan). Frekvencija *clock-a* ovisi o formatu podataka koji se prenosi kao korisni teret i može biti definisana statički u specifikaciji profila ili formata korisnog tereta, ili može biti definisana dinamički putem mehanizama koji nisu dio RTP-a. Ako se RTP paketi generišu periodično, treba koristiti nominalno vrijeme uzorkovanja određeno iz *clock-a* uzorkovanja, a ne trenutno očitanje sistemskog *clock-a*.
- **SSRC 32 bita** - Ovo polje predstavlja identifikator izvora sinhronizacije i treba biti odabran nasumično kako bi se osiguralo da niti jedan od dva izvora unutar iste RTP sesije nemaju isti identifikator. Iako je vjerovatnoća da više izvora odabere isti identifikator mala, sistemi moraju biti sposobni prepoznati i riješiti takve situacije. Također, ako izvor promijeni svoju transportnu adresu, mora odabrati novi identifikator kako bi se izbjeglo pogrešno tumačenje kao ponovljeni izvor. Ove identifikatore umeću mikseri, koristeći SSRC identifikatore izvora koji su doprinijeli.

1.4 Verifikacija ispravnosti implementacije protokola

Na slici 1.3 je prikazan snimak paketa iz Wiresharka, gdje su vidljivi TCP i RTP paketi, a njihova analiza može pružiti uvid u način na koji su podaci preneseni i procesuirani.

Proces započinje zahtjevom za uspostavu konekcije te konsekventnom razmjenom informacija o parametrima konekcije i tipu sadržaja koji se zahtijeva. Prenos se odvija putem TCP protokla. Na slici 1.3 može se uočiti da paket dolazi sa IP adrese 10.1.1.1 i upućen je ka 10.1.1.2, koristeći UDP protokol. Koristi se RTP verzija 2, bez dodatnog *padding-a* i ekstenzija. Sekvencijski broj paketa je 12345, dok vremenska oznaka koja omogućava sinhronizaciju reprodukcije 67890. Također, SSRC identifikator 0x12345678 osigurava prepoznavanje izvora RTP toka. Vidljivo je također na osnovu snimljenog saobraćaja da three-way-handshake nije sinhronizovan, ali to ne utiče na ishod simulacije obzirom da su nam od važnosti isključivo vremena uspostavljanja konekcije.

Payload tip je *DynamicRTP-Type-96*, koji se koristi za prenos multimedijalnih podataka, poput audio ili video sadržaja u prilagodljivim formatima. Donji dio prikaza sadrži heksadecimalni zapis paketa, koji predstavlja enkodirane audio ili video podatke. Ova analiza potvrđuje da je RTP paket ispravno formiran i prenesen, što ukazuje na pravilnu implementaciju RTP protokola.



Slika 1.3: Prikaz u Wireshark okruženju

Poglavlje 2

Opis testiranih scenarija i rezultati simulacija

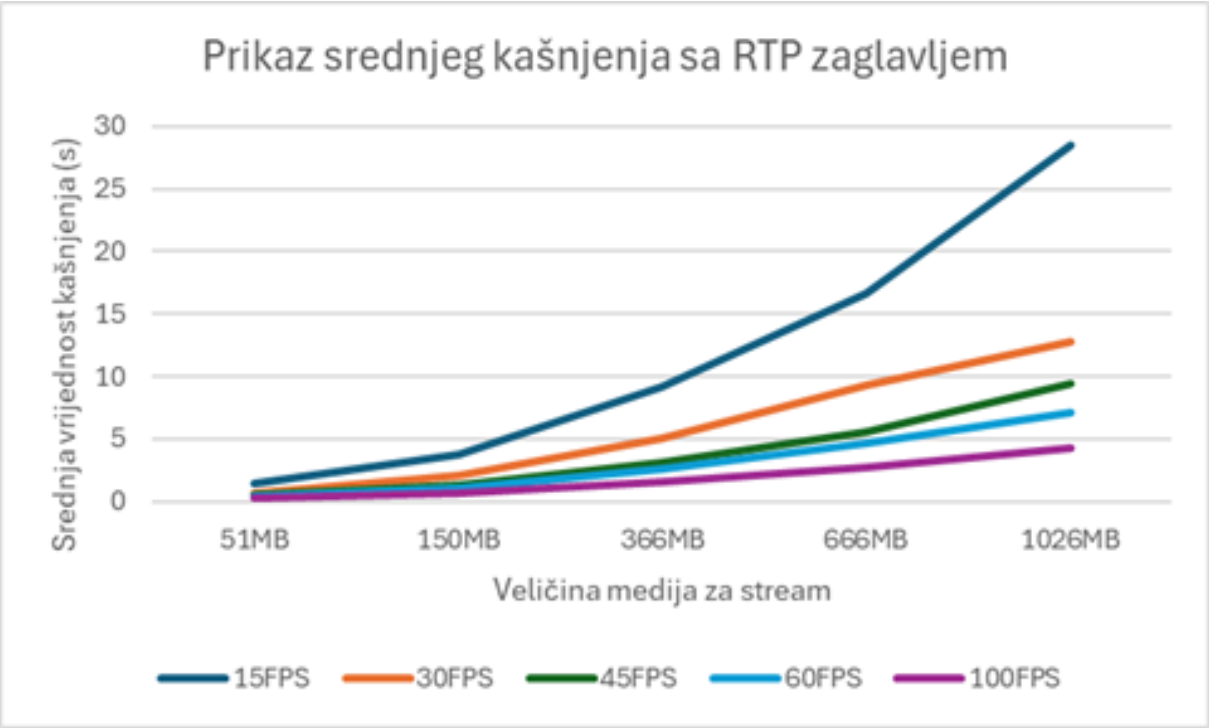
Kako bi izvršili adekvatnu simulaciju, kreiran je payload paket veličine 1200 bajta, jer ova veličina predstavlja optimalnu ravnotežu između efikasnosti pakovanja i performansi kod MPEG-4 enkodera, konkretno H.264 (AVC) kodeka, prema istraživanjima u literaturi [1]. Odabrana veličina paketa omogućava bolje upravljanje protokom podataka u odnosu na druge metode, poput MPEG-2 TS, te doprinosi smanjenju latencije u sistemu. Također, prisutne su značajne razlike u MTU (Maximum Transmission Unit) veličinama različitih mrežnih okruženja. Maksimalna Ethernet MTU 1500 bajta, nakon oduzimanja IP (20 bajta) i UDP/RTP zaglavlja (28 bajta), preostaje otprilike 1452 bajta za korisničke podatke. Da bi se izbjeglo fragmentiranje i poboljšala efikasnost mreže, izabrana je veličina 1200 bajta, jer se sigurno uklapa unutar Ethernet MTU čak i uz dodatna RTP/UDP/IP zaglavlja.

Za kašnjenje je postavljen prag ispod 150 ms, jer se prema ITU-T Rec. G.114 [2] preporukama smatra da većina aplikacija neće biti značajno pogođena kašnjenjima ispod ovog praga. Na osnovu testova dokumentovanih u Annexu B ITU-T Rec. G.114, interaktivne aplikacije, kao što su govor i video konferencije, mogu imati problema sa kašnjenjima iznad 100 ms, dok kašnjenje ispod 150 ms omogućava stabilnu komunikaciju za većinu aplikacija. Također, u svrhu planiranja mreže, ITU-T Rec. G.114 postavlja gornju granicu kašnjenja od 400 ms.

Eksperiment iz rada [1] pokazao je da latencija ostaje stabilna do 180 korisnika, sa prosječnom vrijednošću latencije od 145,20 ms i standardnom devijacijom od 20,48 ms, što potvrđuje da se kašnjenje unutar ovog opsega smatra prihvatljivim za real-time video komunikaciju. Međutim, iznad ovog broja korisnika, latencija se brzo povećava, što znači da je sistem dosegao svoj gornji limit performansi.

2.1 Prikaz srednjeg kašnjena sa RTP zaglavljem

Na slici 2.1 vidi se prikaz srednjeg kašnjenja sa RTP zaglavljem. Grafik sadrži pet krivulja, od kojih svaka predstavlja različitu vrijednost FPS-a: 15, 30, 45, 60 i 100 FPS.



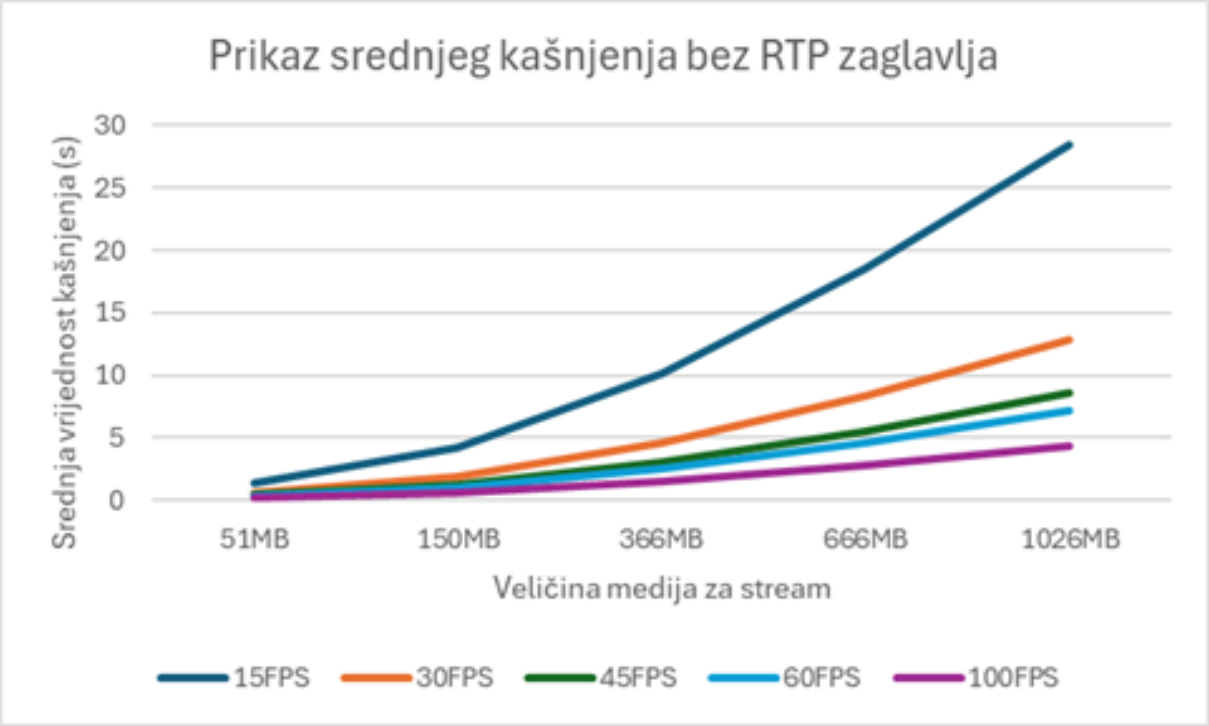
Slika 2.1: Prikaz srednjeg kašnjenja sa RTP zaglavljem

Može se primijetiti da s povećanjem veličine medija (od 51 MB do 1026 MB), dolazi do povećanja kašnjenja. Međutim, brzina povećanja zavisi od FPS-a. Krivulja za 15 FPS pokazuje najveći rast kašnjenja, što znači da pri nižim FPS vrijednostima povećanje veličine medija značajno utječe na povećanje kašnjenja. S druge strane, krivulja za 100 FPS pokazuje najmanji rast, što ukazuje da pri većim FPS vrijednostima povećanje veličine medija uzrokuje manji rast kašnjenja.

Još jedan ključni uvid je da, bez obzira na FPS, kašnjenje je minimalno za manje vrijednosti veličine medija (npr. 51 MB i 150 MB), međutim, kako se ta veličina povećava (npr. 666 MB i 1026 MB), razlike između FPS vrijednosti postaju izraženije, s tim da niži FPS rezultira značajnijim kašnjenjem.

2.2 Prikaz srednjeg kašnjena bez RTP zaglavlja

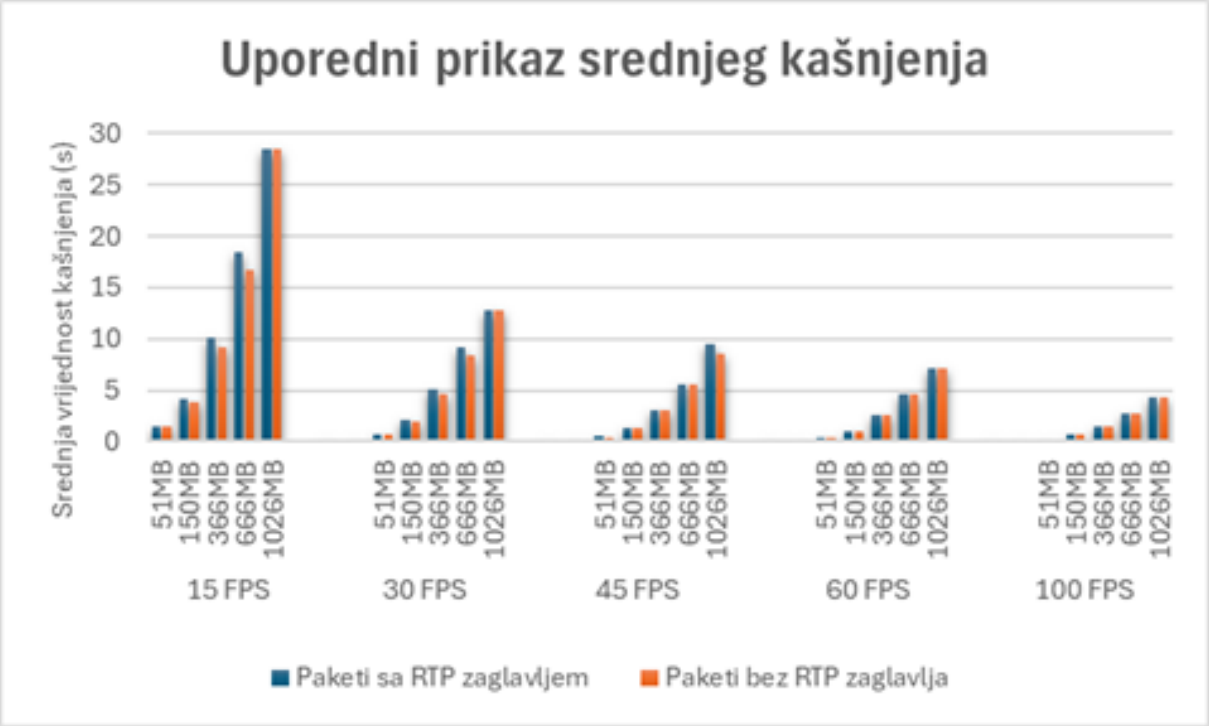
Parametri na osnovu kojih se vrši analiza su isti kao u prethodnom slučaju. Grafik prikazuje srednje kašnjenje u zavisnosti od veličine medija i različitih FPS vrijednosti. Na x-osi su prikazane veličine paketa (51MB, 150MB, 366MB, 666MB, 1026MB), dok y-osa prikazuje srednju vrijednost kašnjenja u sekundama, koja označava vrijeme potrebno za prenos podataka. Linije različitih boja predstavljaju različite FPS vrijednosti (15FPS, 30FPS, 45FPS, 60FPS, 100FPS), a iz grafika je jasno da povećanje veličine paketa povećava kašnjenje, dok povećanje FPS-a smanjuje kašnjenje. Razlika u ovim graficima leži u tome što pri izvođenju testova ove simulacije poslan je paket bez RTP zaglavlja. U skladu s tim vrijednosti srednjeg kašnjenja su umanjenje proprocionalno vrijednosti RTP zaglavlja.



Slika 2.2: Prikaz srednjeg kašnjenja sa RTP zaglavljem

2.3 Uporedni prikaz srednjeg kašnjenja

Na grafiku 2.3 dat je prikaz uporedne analize prethodna dva grafika, što je omogućilo dublje razumijevanje srednjih kašnjenja. Uočava se vrlo mala razlika u vrijednostima srednjih kašnjenja, što je posljedica minimalnog utjecaja zaglavlja RTP paketa od 12B. Iako je ta razlika zanemariva pri manjem broju FPS, s porastom broja FPS-a (posebno kod viših vrijednosti poput 60FPS i 100FPS) ova razlika nestaje. To ukazuje da se s povećanjem FPS-a većina kašnjenja izazvana zaglavljem RTP paketa gotovo potpuno kompenzira, a ukupno kašnjenje postaje dominantno povezano s veličinom paketa i brzinom prijenosa podataka.



Slika 2.3: Uporedni prikaz srednjg kašnjenja

Zaključak

Kroz drugi dio projektnog zadataka, cilj je bio praktično primijeniti teorijska znanja iz oblasti WebRTC-a u pogledu sprovođenja simulacija i analize rezultata specifičnih scenarija. S tim ciljem izvršeno je mijenjanje intervala generisanja paketa koje server šalje ka klijentu na osnovu vrijednosti od minimalno 15 do maksimalno 100 FPS-ova, dobijena su prethodno opisana dva scenarija.

Na osnovu provedenih simulacija i prikazanih rezultata, može se zaključiti da veličina paketa, FPS vrijednosti i prisustvo RTP zaglavlja imaju utjecaj na kašnjenje u mreži. Odabrana veličina paketa od 1200 bajta pokazala se optimalnom, jer omogućava bolju efikasnost u prenosu podataka i smanjuje rizik od fragmentacije, s obzirom na Ethernet MTU od 1500 bajta.

Kašnjenje je generalno povećano s povećanjem veličine medija, ali brzina povećanja ovisi o FPS vrijednosti. Za niže FPS vrijednosti, povećanje veličine medija dovodi do većeg porasta kašnjenja, dok veće FPS vrijednosti smanjuju taj rast. Također, prisustvo RTP zaglavlja dodatno povećava kašnjenje, dok njegovo izostavljanje smanjuje isto, što je potvrđeno u simulacijama s i bez RTP zaglavlja.

Obzirom na jednostavno implementiranu topologiju, prostor za varijaciju parametara bio je ograničen, što je otežalo planiranje scenarija za testiranje. Ovaj limitirani prostor nije omogućio dovoljno fleksibilnosti u ispitivanju različitih kombinacija faktora, što je rezultiralo izazovima u procesu kreiranja i optimizacije testnih scenarija. Zbog toga su neki aspekti testiranja morali biti pojednostavljeni, a preciznost rezultata mogla je biti smanjena, jer se nisu mogli u potpunosti istražiti svi potencijalni utjecaji varijacija parametara na performanse sistema.

Skripte korištene u implementaciji modela nalaze se na Github repozitoriju.

Popis slika

Literatura

[1] T. Uhl, K. Nowicki, J. H. Klink, and C. Hoppe. Comparison study of h.264/avc, h.265/hevc and vp9-coded video streams for the service iptv. *Faculty of Transport Engineering and Economics, Maritime University of Szczecin, Szczecin, Poland*, 2021.

[2] International Telecommunication Union. Itu-t g.114: One-way transmission time. ITU-T Recommendation G.114, Telecommunication Standardization Sector of ITU, May 2003.

[3] H. R. Zimmerman et al. *RTP: A Transport Protocol for Real-Time Applications*, 2003.