



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА



УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЗАВРШНИ (BACHELOR) РАД

Кандидат: Милошевић Немања
Број индекса: Е12684

Тема рада: Реализација модула за процјену оптималне брзине преноса
видео тока података

Ментор рада: доц. др Илија Башичевић

Нови Сад, јун, 2012



КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, РБР:			
Идентификациони број, ИБР:			
Тип документације, ТД:	Монографска документација		
Тип записа, ТЗ:	Текстуални штампани материјал		
Врста рада, ВР:	Завршни (Bachelor) рад		
Аутор, АУ:	Немања Милошевић		
Ментор, МН:	доц. др Илија Башичевић		
Наслов рада, НР:	Реализација модула за процјену оптималне брзине преноса видео тока података		
Језик публикације, ЈП:	Српски / латиница		
Језик извода, ЈИ:	Српски		
Земља публиковања, ЗП:	Република Србија		
Уже географско подручје, УГП:	Војводина		
Година, ГО:	2012		
Издавач, ИЗ:	Ауторски репринт		
Место и адреса, МА:	Нови Сад; трг Доситеја Обрадовића 6		
Физички опис рада, ФО: (поглавља/страница/цитата/табела/ слика/графика/прилога)	7/36/0/0/10/0/0		
Научна област, НО:	Електротехника и рачунарство		
Научна дисциплина, НД:	Рачунарска техника		
Предметна одредница/Кључне речи, ПО:	HTTP Live Streaming протокол, процјена оптималне брзине преноса видео тока података		
УДК			
Чува се, ЧУ:	У библиотеци Факултета техничких наука, Нови Сад		
Важна напомена, ВН:			
Извод, ИЗ:	У овом раду приказана је и описана реализација једног од алгоритама за процјену оптималне брзине преноса видео тока података у комбинацији са HTTP Live Streaming протоколом.		
Датум прихватања теме, ДП:			
Датум одbrane, ДО:			
Чланови комисије, КО:	Председник:	проф. др Мирослав Поповић	
	Члан:	проф. др Јелена Ковачевић	Потпис ментора
	Члан, ментор:	доц. др Илија Башичевић	



KEY WORDS DOCUMENTATION

Accession number, ANO:		
Identification number, INO:		
Document type, DT:	Monographic publication	
Type of record, TR:	Textual printed material	
Contents code, CC:	Bachelor Thesis	
Author, AU:	Nemanja Milošević	
Mentor, MN:	PhD Ilija Bašičević	
Title, TI:	Implementation of module for evaluation of optimal transmission rate for video data stream	
Language of text, LT:	Serbian	
Language of abstract, LA:	Serbian	
Country of publication, CP:	Republic of Serbia	
Locality of publication, LP:	Vojvodina	
Publication year, PY:	2012	
Publisher, PB:	Author's reprint	
Publication place, PP:	Novi Sad, Dositeja Obradovica sq. 6	
Physical description, PD: (chapters/pages/ref./tables/pictures/graphs/appendices)	7/36/0/0/10/0/0	
Scientific field, SF:	Electrical Engineering	
Scientific discipline, SD:	Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, S/KW:	HTTP live Streaming Protocol, estimation of optimal transmission rate for video data stream	
UC		
Holding data, HD:	The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, N:		
Abstract, AB:	In this work is presented one algorithm for estimation of optimal transmission data rate for video data stream, in combination with HTTP Live Streaming Protocol	
Accepted by the Scientific Board on, ASB:		
Defended on, DE:		
Defended Board, DB:	President:	PhD Miroslav Popović
	Member:	PhD Jelena Kovačević
	Member, Mentor:	PhD Ilija Bašičević
		Menthor's sign

SADRŽAJ

1.	Uvod.....	1
2.	Teorijske osnove	2
2.1	Počeci slanja tokova podataka preko interneta (eng. <i>streaming</i>)	2
2.2	Uživo slanje toka i slanje toka na zahtjev korisnika	3
2.3	Neka popularna rješenja	4
2.3.1	RTP/RTSP	4
2.3.2	Adobe Flash Media Streaming Server	5
2.4	Slanje toka podataka uz prilagođavanje trenutnoj bitskoj brzini (<i>Adaptive bitrate streaming - ABS</i>)	6
2.5	Dinamičko adaptivno slanje toka podataka - DASH	8
2.6	HTTP Live Streaming (HLS)	8
2.6.1	HLS arhitektura	9
2.6.2	Komponenta na strani poslužioca	10
2.6.3	Sistem za distribuciju.....	12
2.6.4	Programska podrška na strani klijenta	12
2.7	Korišćenje HLS protokola.....	12
3.	Analiza zahtjeva.....	15
3.1	Opis zadatka	15
3.2	FFmpeg biblioteka.....	15
4.	Specifikacija rješenja	17
4.1	Opis algoritma	17
4.2	Opis korišćenjih funkcija i programske strukture	19
4.2.1	Struktura koja predstavlja cirkularni bafer (struct circularBuffer)	19

4.2.2	Funkcija za stvaranje novog bafera zadate veličine (CircularBufferCreate)	20
4.2.3	Funkcija za inicijalizaciju bafera (CircularBufferInit)	20
4.2.4	Funkcija za dodavanje novog elemanta u bafer (CircularBufferPut)	21
4.2.5	Funkcija za uništavanje bafera i slobađanje memorije (CircularBufferDestroy)	
	21	
4.2.6	Funkcija koja služi za pronalaženje toka podataka koji nam najviše odgovara (find_best_suited_stream).....	21
4.2.7	Funkcija za izdvajanje dijela URL adrese koji se odnosi na server (get_ready_url).....	22
4.2.8	Funkcija za obradu odgovora od servera na PING poruku (pr_pack)	22
4.2.9	Funkcija za slanje PING poruka (pinger)	23
5.	Rezultati	24
6.	Zaključak	28
7.	Literatura.....	29

SPISAK SLIKA

Slika 2.1 – Slanje toka podataka na zahtjev (eng. <i>on-demand streaming</i>)	3
Slika 2.2 – Slanje toka podataka uživo (<i>live streaming</i>).....	4
Slika 2.3 - Prilagođavanje emitovanja toka podataka trenutnoj bitskoj brzini (eng. <i>Adaptive BitRate Streaming</i>)	6
Slika 2.4 – Korišćenje standardnih HTTP poslužilaca.....	7
Slika 2.5 - HTTP Live Streaming arhitektura	9
Slika 2.6 - Mehanizam klizajućeg prozora (eng. <i>sliding window</i>).....	13
Slika 2.7 - Struktura index datoteke koja sadrži reference ka alternativnim tokovima podataka.....	14
Slika 5.1 – Prikaz vrijednosti vremena RTT u realnim uslovima	25
Slika 5.2 – Ponašanje sistema u realnim uslovima	26
Slika 5.3 – Prikaz posledice korišćenja rezultata koji ne prikazuju pravo stanje	27

SKRAĆENICE

RTP	– <i>Real-time Transfer Protocol</i> , Protokol za slanje podataka u realnom vremenu
RTSP	– <i>Real Time Streaming Protocol</i> , Protokol za kontrolu slanja toka podataka u realnom vremenu
RTMP	– <i>Real Time Messaging Protocol</i> , Protokol za slanje toka podataka
HTTP	– <i>Hyper Text Transfer Protocol</i> , Osnovni internet protokol za slanje datoteka
CDN	– <i>Content Delivery Network</i> , mreža za dostavljanje podataka
VCR	– <i>Video Cassete Recorder</i> , Video rekorder
RFC	– <i>Request For Comments</i> , predstavlja zvaničnu specifikaciju internet standarda
FMS	– <i>Flash Media Server</i> , Poslužilac za slanje toka podataka
RTT	– <i>Round Trip Time</i> , Vrijeme koje protekne od kada se neka poruka pošalje nekom poslužiocu do kad ne stigne odgovor od tog poslužioca
HLS	– <i>HTTP Live Streamings</i> , HTTP bazirano slanje toka podataka
DASH	– <i>Dynamic Adaptive Streaming over HTTP</i> , Dinamičko adaptivno slanje toka podataka
IETF	– <i>Internet Engineering Task Force</i> , Udruženje koje se bavi internet standardima
URL	– <i>Uniform Resource Locator</i> , Web adresa
VOD	– <i>Video On Demand</i> , Video koji se šalje korisniku na njegov zahtjev
MPEG2 TS	– <i>MPEG2 Transport Stream</i> , MPEG2 digitalni tok podataka

1. Uvod

U ovom radu prikazana je i opisana realizacija jednog od algoritama za procjenu optimalne brzine prenosa video toka podataka. Rješenje je realizovano za PC Linux, mada se uz minimalne izmjene može upotrijebiti i na Windows operativnom sistemu.

Realizovani algoritam treba da posluži kao podrška za HLS (eng. *HTTP Live Streaming*) protokol kako bi se omogućilo da korisnik u svakom trenutku dobija video najboljeg mogućeg kvaliteta u zavisnosti od karakteristika mreže, odnosno internet veze. Pošto su ove dvije stvari vezane, detaljno je objašnjen i sam HLS protokol kako bi se ukazalo na prednosti koje ovaj ili bilo koji drugi algoritam za procjenu dostupnog propusnog opsega ima na kvalitet usluge.

Ovaj rad sadrži 7 poglavlja.

U prvom poglavlju opisan je sadržaj rada.

Drugo poglavlje opisuje tehnike slanja toka podataka, od toga kako su se pojavile pa do danas. Nabrojane su neke od danas najkorišćenijih tehnika, a HTTP Live Streaming protokol je detaljno opisan jer je sama tema ovog rada vezana za taj protokol.

U trećem poglavlju detaljno je opisan algoritam i opisano je okruženje za koje je algoritam namjenjen.

Četvrto poglavlje sadrži opis same realizacije algoritma, kako je i na koji način algoritam implementiran, pri čemu je osnovni algoritam proširen sa jednim dijelom koji je dodat.

Peto poglavlje predstavlja osvrt na praktične rezultate primjene algoritma, kao i neke probleme koji su se javili.

Šesto poglavlje daje kratak osvrt na ono što je rađeno.

Konačno, sedmo poglavlje predstavlja spisak korišćene literature.

2. Teorijske osnove

2.1 Počeci slanja tokova podataka preko interneta (eng. *streaming*)

U ranim 1990-im pojavili su se komercijalni računari namjenjeni za ličnu upotrebu (eng. *personel consumer-grade computers*) koji su bili dovoljno moćni da prikažu video odnosno da reprodukuju audio signal. Ovi, rani oblici multimedije za računar isporučivani su preko prenosnih medijuma koji nisu bili namjenjeni za slanje tokova podataka, kao što su CD-ROM-ovi, ili su preuzimani putem interneta u obliku digitalnih datoteka sa udaljenih web-poslužilaca (eng. *server*) koje su zatim skladištene u spoljnoj memoriji korisničkog računara kako bi se kasnije reprodukovale. Ovaj postupak bi se mogao označiti kao koncept “prvo preuzmi pa tek onda reprodukuj” kao nešto što je suprotno pojmu slanja toka podataka. U to doba glavni izazov za prenos multimedijalnih datoteka preko računarske mreže bio je konflikt između veličine multimedijalne datoteke i ograničenosti propusnog opsega u mreži.

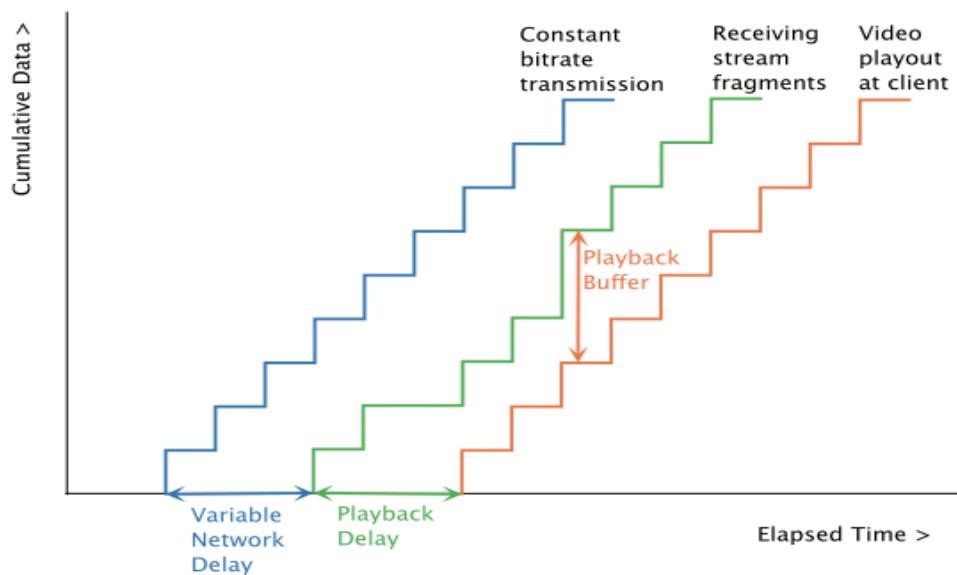
Pocetkom 2000-ih u internetu je doslo do naglog povećanja propusnog opsega. Zajedno sa boljim algoritmima za kompresiju multimedijalnih sadržaja (prije svega video sadržaja) i sve moćnijim personalnim računarima, dostavljanje multimedijalnih sadržaja korišćenjem tehnike slanja toka podataka je postalo moguće. Pojam slanja toka podataka (eng. *streaming*) se koristi da se opiše metod prenosa podataka preko računarske mreže u vidu stabilnog neprekidnog niza podataka dozvoljavajući reprodukciju već primljenih podataka dok se ostatak podataka koji još nisu primljeni nastavlja istovremeno da prima. Ovo je u suprotnosti sa konceptom “prvo preuzmi pa tek onda reprodukuj” gdje reprodukcija može da počne tek kada su preuzeti svi podaci koji čine neku multimedijalnu datoteku. Trenutna reprodukcija sadržaja je glavna prednost dostavljanja podataka putem slanja toka podataka, jer korisnik više ne mora da čeka dok se ne

preuzmu i skladište svi podaci koji su dio neke multimedijalne datoteke sto zahtjeva puno vremena ukoliko je internet veza loša.

2.2 Uživo slanje toka i slanje toka na zahtjev korisnika

Postoje dvije klase slanja toka podataka multimedijalnog sadržaja: slanje sadržaja na zahtjev korisnika (eng. *on-demand streaming*) i slanje sadržaja koji nastaje uživo (eng. *live streaming*). U prvom slučaju multimedijalni sadržaj je prethodno snimljen i kompresovan. Multimedijalni sadržaj se skladišti na serveru i dostavlja se do jednog ili do više korisnika na njihov zahtjev(eng. *on demand*). Danas na hiljade sajtova omogućava slanje skladištenog audio i video sadržaja uključujući Microsoft, You Tube,Vimeo, CNN i mnoge druge. Sa druge strane kod slanja sadržaja koji nastaje uživo, multimedijalni sadržaj se snima, kompresuje i šalje prema korisnicima u realnom vremenu. Slanje sadržaja koji nastaje uživo zahtjeva značajne računarske resurse i veoma često specijalnu fizičku arhitekturu.

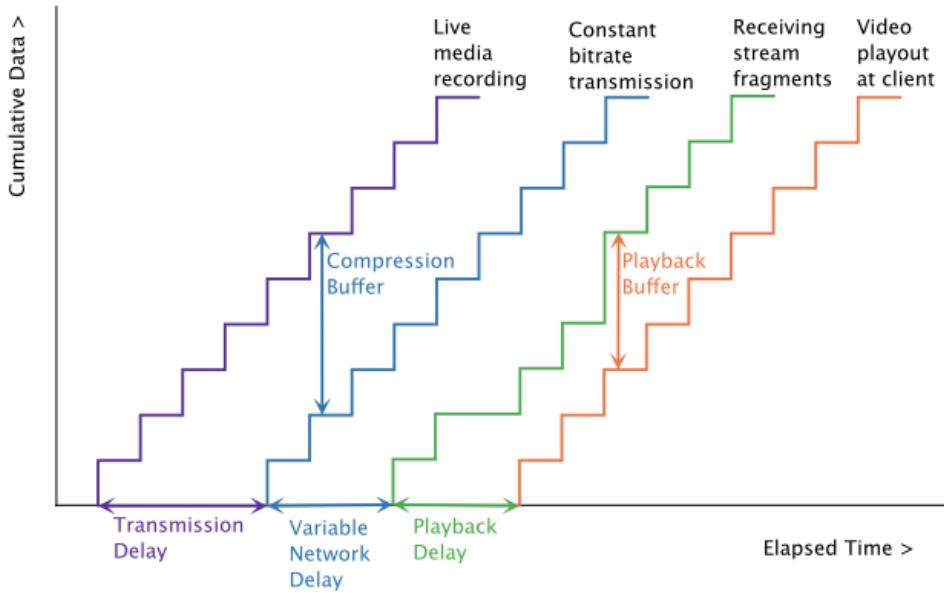
Slanje toka multimedijalnog sadržaja pruža korisnicima istu funkcionalnost kao i tradicionalni uređaji za reprodukciju multimedijalnog sadržaja: zaustavljanje, vraćanje unazad, ubrzano premotavanje i prebacivanje na osnovu sadržaja (na primjer film je podjeljen na više cjelina i moguće je da se direktono skoči na neku od cjelina bez potrebe za premotavanjem i gledanjem da li se došlo do željenog dijela).



Slika 2.1 – Slanje toka podataka na zahtjev (eng. *on-demand streaming*)

Slika 2.1 prikazuje slanje toka već snimljenog i skladištenog video sadržaja. Vertikalna osa predstavlja ukupnu količinu podataka koja se šalje dok horizontalna osa predstavlja proteklo vrijeme. Skroz lijeva stepeničasta linija (plava linija) predstavlja prenos podataka na konstantnoj

bitskoj brzini. Zbog malih oscilacija u kašnjenjima kroz računarsku mrežu (eng. *network jitter*) poslati paketi (odnosno dijelovi video toka koji se šalje) stižu do klijenta (korisnika) u nejednakim vremenskim intervalima. Da bi se uklonila ova neugodna posljedica promjenjivog mrežnog kašnjenja na prijemnoj strani se koristi prijemni bafer i samim tim se i reprodukcija odlaže za kratko vrijeme.



Slika 2.2 – Slanje toka podataka uživo (*live streaming*)

Slika slanja toka video sadržaja koji nastaje uživo (slika 2.2) ima jednu dodatnu stepeničastu liniju (ljubičasta linija) koja predstavlja samo snimanje događaja (npr. video kamerom). Taj snimljeni audio/video sadržaj mora da se kompresuje u realnom vremenu i da se pošalje prema korisnicima. Zbog potrebe da se audio/video sadržaj kompresuje koristi se bafer na predajnoj strani što prouzrokuje dodatno kašnjenje u prenosu. Sve ostalo je isto kao i kod emitovanja toka već snimljenog i skladištenog sadržaja.

2.3 Neka popularna rješenja

Danas postoje razna rješenja u oblasti emitovanja toka podataka ali među njima postoje dva koja su široko rasprostranjena i koja se najviše koriste a to su RTP/RTSP protokol i Adobe Flash Media Streaming Server.

2.3.1 RTP/RTSP

Real Time Streaming Protocol (protokol za slanje toka podataka u realnom vremenu) je razvijen od strane firmi Netscape i Real u kasnim 90-im i predstavlja kontrolni mrežni protokol

koji služi da bi se kontrolisali specijalni poslužioci koji služe za slanje toka multimedijalnih podataka. Ovaj protokol se koristi kako bi se ostvarila i kontrolisala sesija odnosno razmjena podataka između dvije krajnje tačke (između poslužioca i klijenta). Klijenti koriste komande koje su slične kao i standardne komande za videorekorder (eng. *VCR-like commands*) kao što su započeti reprodukciju i pauziraj reprodukciju, kako bi se olakšala kontrola reprodukcije multimedijalnih datoteka sa poslužioca u realnom vremenu. RTSP protokol je objavljen kao RFC 2326 1998. godine.

Sam prenos podataka koji se šalje nije definisan od strane RTSP protokola. Većina RTSP poslužilaca koristi RTP protokol (protokol za dostavljanje podataka u realnom vremenu) za dostavljanje multimedijalnog toka podataka, međutim neki proizvođači koriste i svoje, komercijalne protokole za prenos podataka.

Jedan od nedostataka RTSP protokola je to što ovaj protokol koristi prolaz 554 (eng. *port 554*) koji je često blokiran od strane zaštitnih barijera (eng. *firewall*).

2.3.2 Adobe Flash Media Streaming Server

Flash Media Server (FMS) je komercijalni proizvod i u vlasništvu je firme Adobe Systems (originalno proizvod firme Macromedia). Može da se koristi za slanje video toka podataka i uživo i na zahtjev. Najveći video sajt na svijetu, You Tube koristi ovaj sistem za slanje video toka podataka na zahtjev.

Flash Media Server radi kao centralna tačka u kombinaciji sa odgovarajućom klijentskom aplikacijom sa kojom se povezuje koristeći RTMP protokol. Poslužilac može da šalje i prima podatke od korisnika sa kojim je povezan i koji ima instaliranu Flash Player aplikaciju.

Ovo rješenje ima i nekoliko mana:

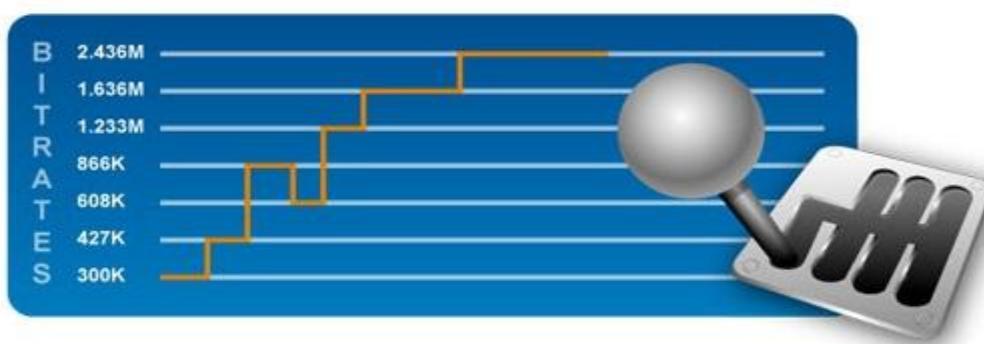
- Flash Media Streaming Server je komercijalni proizvod, dakle naplaćuje se
- Korisnici moraju da imaju instaliranu Flash Player aplikaciju. Ovo danas ne predstavlja problem posto je većina internet korisnika tokom vremena preuzeo i instalirala ovu aplikaciju.
- Flash Player aplikacija nije dobro implementirana na nekim platformama. Pogotovo na Linuxu i na MAC operativnom sistemu loša implementacija prouzrokuje veliko opterećenje procesora što utiče na brže pražnjenje baterija. Ovo je ujedno i glavni razlog zašto je Apple izbacio iz upotrebe Flash Player aplikaciju na svojim iPhone i iPod touch uređajima.

2.4 Slanje toka podataka uz prilagođavanje trenutnoj bitskoj brzini (*Adaptive bitrate streaming - ABS*)

ABS je tehnika koja se koristi za prenos multimedijalnih podataka preko računarskih mreža. Dok je u prošlosti većina tehnika za slanje video toka podataka koristila protokole kao što su RTP/RTSP, današnje tehnologije adaptivnog slanja toka podataka su gotovo u svim slučajevima bazirane na HTTP protokolu i projektovane su da rade efikasno preko velikih distribuiranih mreža koje podržavaju HTTP protokol kao što je internet.

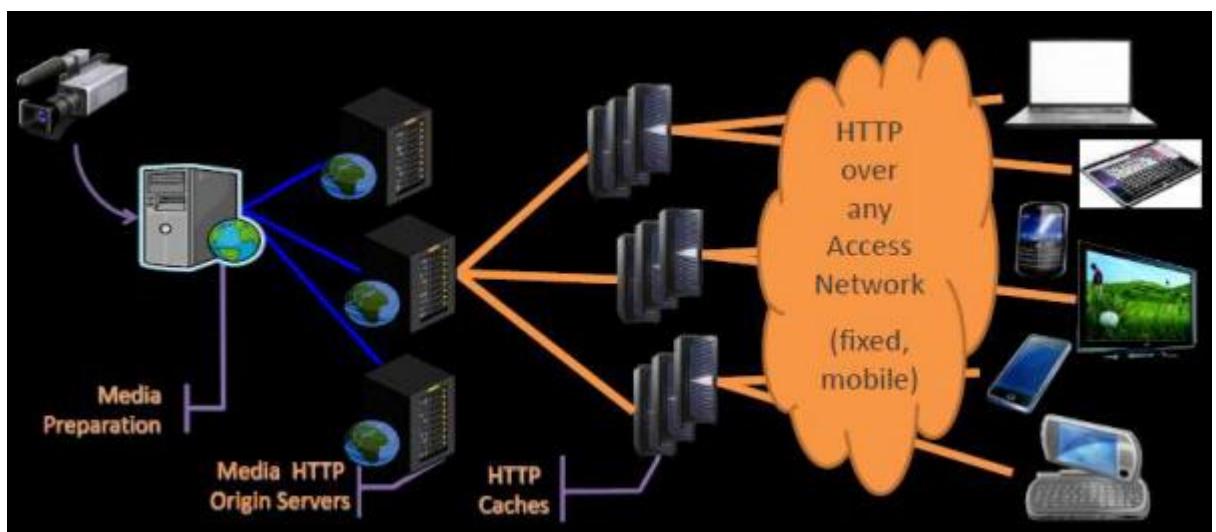
Ova tehnika radi tako što se detektuje korisnički (klijentski) propusni opseg i procesorski kapacitet (od procesorske moci tj. brzine zavisi da li će podaci uspjeti da se obrade dovoljno brzo) u realnom vremenu, i na osnovu tih podataka prilagođava kvalitet video toka podataka (ovo podrazumijeva korišćenje enkodera koji može da enkoduje jedan izvorni video na više različitih bitskih brzina). Korisnička aplikacija za reprodukciju video sadržaja se na osnovu raspoloživih resursa (tj. verzija istog video materijala ali na različitim bitskim brzinama) i u zavisnosti od propusnog opsega prebacuje sa jednog toka podataka na drugi. Kao rezultat svega ovoga dobijamo veoma malo baferovanje, brz početak reprodukcije i dobar doživljaj sa korisničke strane, bilo da korisnici imaju dobru ili pak lošiju internet vezu.

Razne produkcijske kuće i mreže za isporuku podataka (*CDN – Content Delivery Networks*) koriste ABS tehnologiju kako bi korisnicima obezbijedili što kvalitetniji video sadržaj i što bolji doživljaj prilikom gledanja, koristeći pri tome što manje energije i resursa. Sa te tačke gledišta stvaranje višestrukih verzija video materijala pogotovo za ABS tehnologiju daje dosta na značaju što se tiče korisničke strane tj. kvaliteta usluge (eng. *QoS - Quality of Service*). Ukoliko ova tehnologija zaista radi onako kako je i zamišljena onda krajnji korisnik odnosno konzumer treba da bude potpuno nesvjestan da ona postoji. Stoga, iako medejske kompanije već dugo vremena aktivno koriste ABS tehnologiju većina korisnika je relativno neupućena o njenoj važnosti.



Slika 2.3 - Prilagođavanje emitovanja toka podataka trenutnoj bitskoj brzini (eng. *Adaptive BitRate Streaming*)

Korisnici čiji uređaji koriste tehniku slanja toka multimedijalnih podataka imaju najbolji doživljaj odnosno dobijaju najbolji video kvalitet ukoliko se koristi ABS tehnologija zbog toga što se u svakom trenutku sve prilagođava uslovima na korisničkoj mreži i uslovima reprodukcije. Najviše koristi od ove tehnologije imaju industrija zabave i multimedijalna industrija. Kako se količina video podataka eksponencijalno povećava, video provajderi i mreže za isporuku podataka mogu da obezbijede svojim korisnicima još bolji doživljaj prilikom gledanja video sadržaja odnosno još bolji kavalitet usluge. CDN dobija originalni tok video podataka od nekog poslužioca koji se zatim umnožava i kopije se šalju do nekih određenih ili do svih krajnjih poslužilaca (eng. *edge servers*) koji direktno komuniciraju sa korisnicima i dostavljaju im podatke. Krajnji korisnik zahtjeva neki video tok i biva preusmjeren do najbližeg krajnjeg poslužioca u CDN mreži sa kojim dalje direktno komunicira. Korišćenje tehnologije adaptivnog prenosa toka podataka baziranog na HTTP protokolu dozvoljava krajnjim CDN poslužiocima da koriste jednostavnu programsku podršku za HTTP poslužioce čija licenca je jeftina ili čak besplatna smanjujući pri tome cijenu licenciranja u odnosu na komercijalne pakete specijalizovane programske podrške za multimedijalne poslužioce.



Slika 2.4 – Korišćenje standardnih HTTP poslužilaca

Slanje toka podataka koje se oslanja na HTTP protokol ima tu prednost što ne zahtjeva otvaranje nikakvih posebnih prolaza (portova) koji bi mogli da budu blokirani od strane zaštitnih barijera osim standardnih prolaza koje koriste web pretraživaci. HTTP bazirani prenos toka podataka takođe dozvoljava dijelovima video sadržaja da budu sačuvani za kasniju upotrebu (eng. *cashed*) od strane web pretraživaca, poslužilaca posrednika (eng. *proxy server*) i CDN poslužilaca pri tome drastično smanjujući opterećenje na poslužiocu sa koga originalno potice video materijal.

Danas postoji nekoliko rješenja koja podržavaju ABS tehnologiju, to su:

- HTTP Live Streaming (Apple),

- Smooth Streaming (Microsoft) i
- HTTP Dynamic Streaming (Adobe).

Tehnologija pod nazivom HTTP Live Streaming (HLS) omogućava slanje toka podataka uživo ili na zahtjev, do uređaja kao što su Apple-ovi uređaji iPhone, iPod i iPad bez potrebe za nekim posebnim poslužiocem za slanje toka podataka jer bilo koji HTTP web poslužilac može da se upotrijebi umjesto. Osim HLS tehnologije koju je razvila kompanija Apple ostale dvije tehnologije su komercijalni proizvodi koji se naplaćuju. Pored toga jedino je Apple svoju tehnologiju predstavio pred IETF (*Internet Engineering Task Force*) organizacijom u obliku predloga za standardizaciju (*Internet Draft*).

2.5 Dinamičko adaptivno slanje toka podataka - DASH

DASH (eng. *Dynamic Adaptive Streaming over HTTP*) je tehnologija koja predstavlja poseban oblik ABS tehnologije, gdje se multimedijalna datoteka dijeli na više dijelova (segmenata) koji se zatim koristeći HTTP protokol dostavljaju do klijenta. Pored samih segmenata video datoteke prenose se i podaci koji nose informacije o samim segmentima (Media Presentation Description - MDP) kao što su vrijeme reprodukcije, URL adresa samog segmenta, zatim i neke karakteristike kao što su video rezolucija i bitska brzina. Segmenti (eng. *MFS-Media Segment File*) mogu da sadrže bilo koju vrstu multimedijalnih podataka u bilo kom formatu, međutim preporuka je da se koristi MPEG4 format datoteka, odnosno MPEG2 transportni tok za prenos tih datoteka.

Obično je dostupno više verzija multimedijalne datoteke (verzije različite rezolucije, bitske brzine) i mi možemo da biramo između različitih verzija na osnovu uslova u mreži, mogućnosti samog uređaja na kojem se vrši reprodukcija i korisničkih podešavanja omogućavajući time slanje toka podataka uz prilagođavanje trenutnoj bitskoj brzini (Adaptive Bitrate Streaming).

2.6 HTTP Live Streaming (HLS)

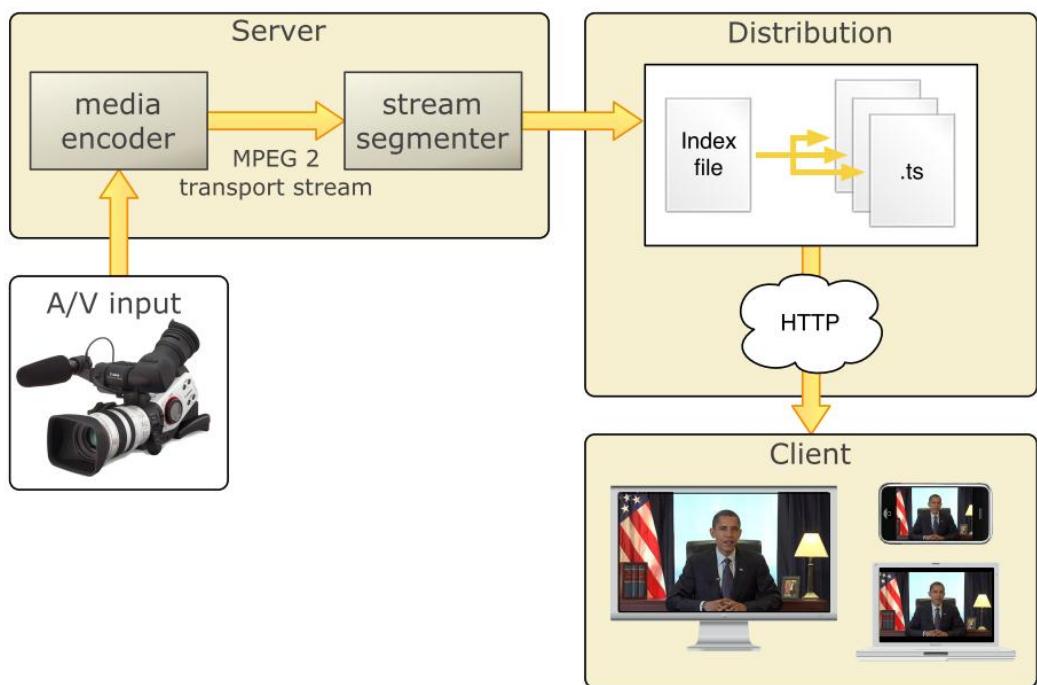
Kompanija Apple je razvila HLS tehnologiju kako bi omogućila provajderima sadržaja da šalju uživo ili na zahtjev, audio i video podatke do korisnika Apple-ovih uređaja koristeći obične web poslužioce, da bi tek kasnije istu funkcionalnost prenijeli i omogućili i na svojim desktop računarima. Kao korak dalje ka standardizaciji Apple je svoje rješenje dostavio i do IETF organizacije u cilju standardizacije ove tehnologije tako da je sada ova tehnologija dostupna svima.

2.6.1 HLS arhitektura

Konceptualno HLS se sastoji od tri dijela: komponente na strani poslužioca, komponente za distribuciju i programske podrške na strani klijenta:

- Komponenta na strani poslužioca je zadužena za uzimanje ulaznih tokova (eng. *input streams*) multimedijanih podataka, njihovo enkodovanje (enkapsulirajući ih u format koji je pogodan za prenos), i pripremanje tih podataka za distribuciju.
- Sistem za distribuciju se sastoji od standardnih web poslužilaca. Oni su odgovorni za prihvatanje klijentskih zahtjeva i dostavljanje pripremljenih podataka do klijenata. Ukoliko se radi o distribuciji širih razmjera mogu se koristiti i specijalne mreže za isporuku podataka (*CDN*).
- Programska podrška na strani klijenta je odgovorna za slanje zahtjeva za određenim audio/video sadržajem, preuzimanje tih sadržaja i njihovu reprodukciju na korisničkom uređaju.

U tipičnoj konfiguraciji hardverski dekoder uzima audio/video ulaz i konvertuje ga u MPEG-2 transportni tok podataka (eng. *MPEG-2 transport stream*), koji se zatim od strane programa za segmentaciju toka podataka (eng. *stream segmenter*) razlaže u niz kratkih multimedijalnih datoteka odnosno multimedijalnih segmenata. Tako dobijeni segmenti se zatim smještaju na web poslužilac.



Slika 2.5 - HTTP Live Streaming arhitektura

Program za segmentaciju toka podataka takođe stvara i održava tzv. index datoteku (eng. *index file*) koja sadrži spisak multimedijalnih segmenata koji su nastali u procesu segmentiranja. URL adresa index datoteke se postavlja na web poslužioc kako bi klijenti mogli da joj pristupe. Klijenti na osnovu URL adrese (koju prethodno moraju da znaju) dobavljaju index datoteku, obrađuju je i prolaze kroz listu segmenata tražeći od poslužioca da im dostavi segmente kako bi ih na kraju reprodukovali i to bez prekida između pojedinačnih segmenata.

Ulagani audio/video podaci mogu da prestavljaju sadržaj koji nastaje uživo ili to može biti nešto što je već snimljeno i skladišteno. Ovi podaci se obično enkoduju u MPEG-2 transportni tok podataka od strane posebne fizičke arhitekture namjenjene za enkodovanje ovakvih podataka. MPEG-2 tok podataka se zadim dijeli na segmente koji se čuvaju kao niz od jedne ili više multimedijalnih datoteka sa ekstenzijom .ts (eng. *transport stream*). Segmentiranje se za razliku od enkodovanje ne obavlja pomoću specijalne fizičke arhitekture već pomoću programske podrške u obliku programa za segmentaciju.

Program za segmentaciju toka podataka kreira index datoteku. Index datoteka sadrži listu multimedijalnih segmenata i još neke dodatne podatke vezane za segmente kao i za sam tok podataka. Index datoteka se nalazi u .M3U formatu i ima istu ekstenziju. URL adresa index datoteke je dostupna klijentima koji prvo dobavljaju samu index datoteku a zatim redom šalju serveru zahtjeve za .ts datotekama (multimedijalnim segmentima) navedenim u samoj index datoteci.

2.6.2 Komponenta na strani poslužioca

Poslužilac mora da ima multimedijalni enkoder, koji može da bude posebno projektovana fizička arhitektura baš za tu namjenu, zatim mora da postoji način da se enkodovani podaci podijele na segmente i da se sačuvaju kao posebne datoteke, pri čemu ovaj dio može da se obavlja i uz pomoć programske podrške kao što su programi za segmentiranje toka podataka. Multimedijalni enkoder uzima signal u realnom vremenu sa izlaza audio/video uređaja, enkoduje multimedijalne podatke i enkapsulira ih u format pogodan za slanje. Trenutno podržani format je MPEG-2 transportni tok podataka (*MPEG-2 transport stream*) za audio/video podatke, ili MPEG elementarni tok podataka za audio podatke. Multimedijalni enkoder zatim dostavlja MPEG-2 transportni tok podataka do programa za segmentiranje toka podataka.

Specifikacija HLS protokola omogućava korišćenje i drugih formata , ali jedino su MPEG-2 video tok podataka (H.264 format za video podatke i AAC za audio podatke) i MPEG elementarni tok audio podataka (AAC format ili MP3 format) trenutno podržani. Ono što je važno da primjetimo ovdje jeste da video enkoder ne smije da promjeni podešavanja vezana za tok podataka (kao na primjer kodek koji se koristi...) usred operacije enkodovanja toka podataka.

Segmentiranje toka podataka predstavlja proces, koji se obično obavlja uz pomoć programske podrške, koja prima transportni tok podataka sa lokalne mreže i dijeli ga u niz malih segmenata, multimedijalnih datoteka, jednake dužine. Iako se svaki segment nalazi u posebnoj datoteci, video datoteke su nastale iz jednog kontinualnog toka podataka koji se može bez problema rekonstruisati nakon što se dostavi do klijenta.

Segmenter toka podataka takođe pravi i index datoteku koja sadrži pokazivače do svih segmenata. Svaki put kada segmenter napravi novi segment index datoteka se ažurira. Index datoteka se koristi da bi imali informaciju o dostupnosti i lokaciji multimedijalnih segmenata. U toku procesa segmentiranja segmenter može i da enkriptuje svaki od segmenata i u tom slučaju mora da napravi i datoteku u kojoj će se nalaziti ključ za dekriptovanje.

Multimedijalni segmenti se snimaju u formatu sa ekstenzijom .ts (MPEG-2 tok podataka) a index datoteka se snima u formatu sa ekstenzijom .M3U8 koja predstavlja proširenje osnovnog formata sa ekstenzijom .m3u.

Slijedi jedan jednostavan primjer .M3U datoteke (*index file*) koja je nastala u procesu segmentiranja toka podataka (pri čemu nije vršena enkripcija) čiji rezultat su tri segmenta trajanja po 10 sekundi:

```
#EXTM3U
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-TARGETDURATION:10
#EXTINF:10,
http://media.example.com/segment1.ts
#EXTINF:10,
http://media.example.com/segment2.ts
#EXTINF:10,
http://media.example.com/segment3.ts
#EXT-X-ENDLIST
```

Index datoteka takođe može da sadrži i URL adrese do datoteka koje sadrže ključeve koji su potrebni za dekriptovanje (naravno ukoliko je enkripcija izvršena) ili do drugih, alternativnih index datoteka za drugačiji propusni opseg odnosno bitsku brzinu.

U krajnjem slučaju moguće je i da se index datoteka i multimedijalni segmenti naprave odvojeno jedni od drugih, ali pod uslovom da su u skladu sa sprecifikacijom. Na primjer za emitovanje audio toka podataka koguće je da se napravi index datoteka korišćenjem jednostavnog uređivača teksta (eng. *text editor*) tako što ćemo u njoj jednostavno samo da navedemo imena postojećih audio datoteka.

2.6.3 Sistem za distribuciju

Sistem za distribuciju se sastoji od web poslužilaca koji dostavljaju index datoteke i multimedijalne segmente do korisnika putem HTTP protokola tako da nema potrebe ni za kakvim posebnim poslužiocima i komponentama da bi se podaci dostavili do klijenta. Kao što je i ranije rečeno u ovu svrhu mogu se iskoristiti i CDN mreže za isporuku podataka koje imaju veliki broj specijalizovanih poslužilaca posebno za tu namjenu.

2.6.4 Programska podrška na strani klijenta

Klijent prvo od servera dobavlja index datoteku, na osnovu URL adrese koja identificuje tok podataka. Sa druge strane index datoteka identificuje lokacije multimedijalnih datoteka (segmenata), datoteka koje sadrže ključeve za dekriptovanje, ili alternativnih tokova podataka ikoliko ih ima. Za odabrani tok podataka, klijent od servera dobavlja redom sve segmente, pri čemu svaki naredni segment sadrži dio toka podataka koji sledeći treba da se reprodukuje. Jednom kada klijent dobavi dovoljnu količinu multimedijalnih podataka, on rekonstruiše tok podataka i počinje da ga reprodukuje. Klijent je zadužen i za dobavljanje ključeva potrebnih za dekripciju.

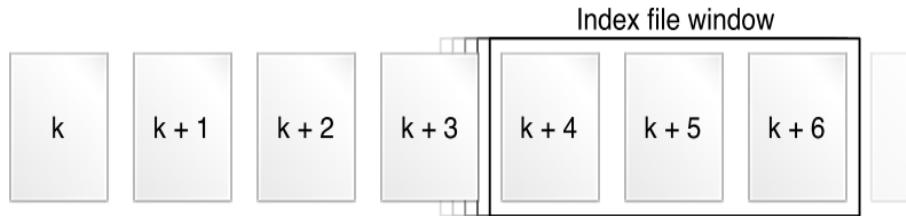
Ovaj proces dobavljanja segmenata se nastavlja sve dok klijent ne nađe na direktivu `#EXT-X-ENDLIST` u index datoteci. Ukoliko se u index datoteci na nalazi ova direktiva to znači da se radi o toku podataka koji se šalje uživo. Klijent periodično dobavlja novu verziju index datoteke, i ako je datoteka promjenjena (znači radi se o slanju toka podataka uživo), onda klijent u index datoteci traži podatke o novododatim segmentima i njihovim URL adresama, i ukoliko ima novih segmenata njihove URL adrese se dodaju u red za dobavljanje podataka.

2.7 Korišćenje HLS protokola

HLS protokol podržava i slanje toka podataka uživo (eng. *live broadcast session*) kao i slanje toka podataka na zahtjev korisnika (eng. *VOD – Video On Demand*).

U slučaju slanja toka podataka uživo čim se proizvedu novi segmenti (multimedijalne datoteke) i čim postanu dostupni na serveru, index datoteka se odmah ažurira. Index datoteka koja je ažurirana sadrži pokazivače (reference) na te nove segmente, dok se stariji segmenti odnosno reference na starije segmente obično uklanjaju. Ažurirana index datoteka može se posmatrati i kao klizajući prozor (eng. *sliding window*) unutar neprekidnog toka podataka. Na slici 2.6 prikazan je mehanizam klizajućeg prozora. Čim neki novi segment postane spreman (dostupan na poslužiocu) index datoteka se ažurira tako da sadrži taj najnoviji segment, dok se najstariji segment uklanja iz datoteke. Na ovaj način index datoteka uvijek sadrži reference na poslednjih x segmenata. Apple preporučuje da x treba da bude veći od 3 tako da klijent može da

pauzira reprodukciju, nastavi reprodukciju i premota unazad za bar dužinu od 3 segmenta (to je obično 30 sekundi jer 1 segment obično traje 10 sekundi).

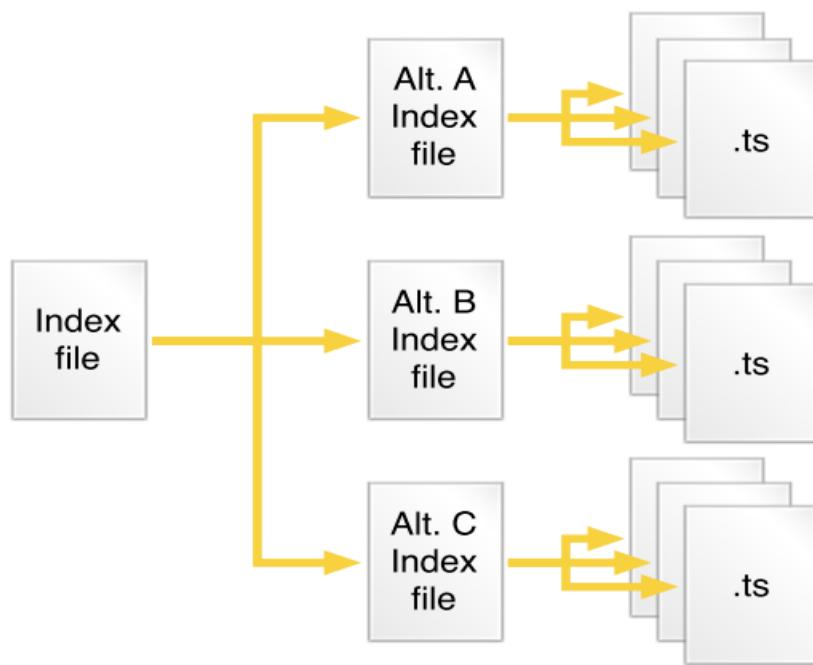


Slika 2.6 - Mehanizam klizajućeg prozora (eng. *sliding window*)

U slučaju slanja video toka podataka na zahtjev svi segmenti su dostupni u svakom trenutku. Index datoteka se ne mijenja i sadrži reference do svih segmenata koji čine neki video tok. Moguće je i da se od toka podataka uživo napravi tok podataka na zahtjev koji je odmah, trenutno dostupan. Da bismo ovo postigli potrebno je samo da u toku slanja toka podataka uživo iz index datoteke ne uklanjamo stare segmente već da ih sve čuvamo, odnosno nove segmente samo dodajemo na kraj liste, i na samom kraju kada više nema novih segmenata dodamo #EXT-X-ENDLIST direktivu. Ovo omogućava korisnicima da čak i kada zakasne sa početkom gledanja nekog video sadržaja ipak mogu da ga odgledaju od početka.

HLS protokol ima i dodatne pogodnosti, kao što su mogućnost enkripcije i dinamičko prebacivanje sa jednog video toka na drugi video tok, sa drugačijom brzinom prenosa podataka, kao odgovor na promjenjive uslove internet veze. Što se tiče enkodovanja trenutno je podržana AES-128 (eng. *Advanced Encryption System*) enkripcija. Ukoliko se koristi enkripcija, u index datoteci se dodaje referenca na datoteku koja sadrži ključ koji je potreban da bi se podaci dekriptovali. Moguće je da se čitav tok podataka, odnosno svi segmenti enkriptuju korišćenjem jednog ključa ili da se n segmenata enkriptuje pomoću jednog ključa, nakon čega se sledećih n segmenata enkriptuje korišćenjem drugog ključa itd. Ovaj zadnji metod enkriptovanja podataka poznat je kao metod rotacije ključeva (eng. *key rotation*). Teoretski je moguće da se za svaki segment koristi drugi ključ ali pošto se na ovaj način dodatno opterećuje internet saobraćaj, pošto je sad za svaki segment potrebno dobaviti ključ to se izbjegava te se najčešće koristi metod rotacije ključeva gdje se ključ mijenja na svakih n ($n > 1$) segmenata.

Index datoteka može da ima i reference ka alternativnim tokovima podataka koji sadrže isti video ali recimo na različitim bitskim brzinama i različitog kvaliteta. Programska podrška na klijentskoj strani na osnovu nekog algoritma za izračunavanje propusnog opsega odlučuje kada će i da li će da se prebaci na neki drugi tok podataka. Na sledećoj slici prikazana je struktura koja prikazuje index datoteku koja sadrži reference ka alternativnim tokovima podataka.



Slika 2.7 - Struktura index datoteke koja sadrži reference ka alternativnim tokovima podataka

Osim što se alternativni tokovi podataka mogu koristiti da bi se poboljšao kvalitet usluge odnosno doživljaj prilikom gledanja video sadržaja, alternativni tokovi podataka mogu da posluže i kao izvjesna vrsta zaštite ukoliko je recimo neki video tok podataka iznenada postao nedostupan na poslužiocu a i dalje u index datoteci postoji referenca na njega. U tom slučaju moguće je detektovati da je traženi tok nedostupan (prilikom zahtjeva koji se šalje poslužiocu) jer bi poslužilac u tom slučaju vratio poruku o grešci, i bio bi zadržan jedan od tokova podataka koji je dostupan. Nažalost, ova funkcionalnost u korišćenoj verziji HLS protokola u sklopu FFmpeg biblioteke nije realizovana.

3. Analiza zahtjeva

3.1 Opis zadatka

Potrebno je proširiti PC Player aplikaciju (koja je zasnovanu na FFmpeg biblioteci) modulom za procjenu optimalne brzine prenosa toka HTTP Live video podataka sa poslužioca. Kada se promjene uslovi prenosa na mreži (obzirom na kašnjenje i propusnost), klijent automatski treba da izvrši procjenu nove optimalne brzine prenosa i zahtjeva od poslužioca da pređe na tok koji podržava izračunatu brzinu prenosa.

Algoritam za proračunavanje propusnog opsega je zadat i njegov opis slijedi:
Šalju se dva paketa za ispitivanje (PK1 i PK2) jedan za drugim (eng. *back-to-back*). Za svaki se izračuna RTT (to su RTT_1 i RTT_2 respektivno). Mjerenje se ponavlja n puta. Ako je L veličina paketa, kapacitet veze se može izračunati kao $C = L/D$, gde je D minimalna disperzija data sa:

$$D = \min[RTT_2(i, i=1,..,n)] - \min[RTT_1(i, i=1,..,n)]$$

Dostupni propusni opseg, A je dat sa:

$$A = \frac{L}{\frac{L}{C} + d_r}$$

Pri čemu je:

$$d_r = \text{avg}[RTT_1(i, i=1,..,n)] - \min[RTT_1(i, i=1,..,n)]$$

Kao paketi za ispitivanje se mogu koristiti ICMP ping ili UDP paketi.

3.2 FFmpeg biblioteka

FFmpeg je softverski projekat koji je slobodan (besplatan) za korišćenje i čiji su rezultat odnosno proizvod programske biblioteke i programi za rad sa multimedijalnim podacima. FFmpeg predstavlja vodeće programsko radno okruženje za rad sa multimedijom koje omogućava enkodovanje, dekodovanje, transkodovanje, multipleksiranje, demultipleksiranje,

slanje toka podataka, filtriranje i reprodukciju skoro svega u smislu multimedijalnog sadržaja. Podržava skoro sve audio/video formate od najstarijih pa sve do onih koji tek ulaze u upotrebu, bez obzira na to ko ih je napravio.

FFmpeg obezbjeđuje razne alate:

- *ffmpeg* – program za konverziju multimedijalnih datoteka iz jednog formata u drugi
- *ffserver* – program koji predstavlja multimedijalni server za emitovanje toka podataka uživo
- *ffplay* – jednostavna aplikacija za reprodukciju multimedijalnog sadržaja
- *ffprobe* – aplikacija koja služi za analizu multimedijalnih tokova podataka

kao i razne programske biblioteke za razvoj programske podrške:

- *libavutil*
- *libavcodec*
- *libavformat*
- *libavdevice*
- *libavfilter*
- *libswscale* i
- *libswresample*

ffplay je aplikacija koja se konkretno koristi u našem zadatku. Ova aplikacija već omogućava reprodukciju HTTP Live toka podataka sa poslužioca ali nema realizovanu podršku za automatsko prebacivanje sa jednog toka podataka na drugi u slučaju da je recimo mreža zagušena pa je protok smanjen ili u obrnutom slučaju kada je omogućen veći protok podataka. Naš zadatak je da u okviru ove aplikacije napravimo i dodamo modul koji treba da vrši procjenu optimalne brzine prenosa odnosno dostupnog propusnog opsega, i da u slučaju da se optimalna brzina prenosa promijenila, od poslužioca zahtjevamo da pređe na drugi tok. Međutim zbog toga što poslužilac posjeduje ograničen broj različitih varijanti nekog video zapisa na različitim bitskim brzinama on nije uvijek u mogućnosti da udovolji zahtjevima klijenata tako da je, pošto znamo koje sve bitske brzine poslužilac podržava, potrebno da izvršimo proračun koji će da nam kaže koji od mogućih tokova nam najviše odgovara.

4. Specifikacija rješenja

U konkretnom slučaju korišćena je verzija 0.10.4 FFmpeg programske biblioteke. U okviru ove biblioteke nalazi se i aplikacija *ffplay* u sklopu koje treba da se doda modul za procjenu optimalne brzine prenosa. Sama *ffplay* aplikacija već podržava HLS (*HTTP Live Streaming*) protokol tako da je traženi zadatak da se doda modul za procjenu optimalne brzine prenosa i da se na osnovu procjene optimalne brzine prenosa podataka ukoliko je to potrebno izvrši prebacivanje na tok podataka koji najbolje odgovara izračunatoj vrijednosti.

4.1 Opis algoritma

Kada se pokrene aplikacija *ffplay* još uvijek se ne zna da li se radi o HLS video toku podataka. Zbog toga se algoritam za procjenu optimalne brzine ne aktivira odmah. Tek kada se sazna da se radi o HLS video toku podataka tek u tom slučaju se aktivira algoritam za procjenu optimalne brzine prenosa video toka podataka. Algoritam se pokreće tako što se aktivira nit `calculate_bandwidth_thread`. Citav algoritam je realizovan u ovoj niti. Da bi bilo moguće slanje pakete za ispitivanje (PING pakete) prvo mora da se zna adresa poslužioca sa koga se vrši slanje video tok podataka. Prilikom pokretanja aplikacije zadaje se URL adresa index datoteke, međutim iz te adrese potreban je samo dio koji se odnosi na server. Za HTTP protokol URL adresa ima sledeći oblik: [http://racunar\[:porta\];putanja\[;parametri\]\[?upit\]](http://racunar[:porta];putanja[;parametri][?upit]) tako da je potrebno da se iz ove adrese izdvoji samo dio koji se odnosi na računar tj. poslužilac. Za potrebe izdvajanja dijela URL adrese koji je potreban realizovana je funkcija `get_ready_url`.

Nakon što se obezbijede uslove za slanje PING poruka može se početi sa njihovim slanjem. Za slanje ping poruka koristi se funkcija `pinger`. Pošto se s vremenom na vrijeme dešava da se ne primi odgovor na neku od PING poruka, njihovo slanje je izmješteno u posebnu nit jer se u suprotnom može desiti da se čeka na odgovor koji nikad neće stići i u tom slučaju će se zaustaviti algoritam sve dok se ne primi odgovor što se neće nikada desiti. Da bi se to spriječilo

slanje PING poruka je izmješteno u posebnu nit, pri čemu se poruke šalju periodično. U ovom slučaju čak i da se neka od poruka izgubi tj. da ne primimo odgovor algoritam će nastaviti da radi jer će se u međuvremenu poslati neka druga PING poruka na koju će server da pošalje odgovor koji će se ispravno primiti te će algoritam nastaviti da radi normalno.

U konkretnom slučaju PING poruke se šalju u paru, tj. po dvije poruke se salju jedna za drugom i to periodično na svaki sekund (neka su to poruke PING1 i PING2). Kada se primi odgovor na PING poruku računa se vrijeme koje je proteklo od njenog slanja (eng. *RTT – Round Trip Time*). Ovo se radi tako što se prilikom slanja PING poruke zapamti vrijeme kada je poruka poslata, i kada se primi odgovor na tu poruku zapamti se i vrijeme kada je poruka primljena. Razlika ova dva vremena predstavlja ukupno vrijeme koje je bilo potrebno da se poruka pošalje do poslužioca i da se pošalje odgovor od poslužioca nazad. Ovo vrijeme je u engleskoj literaturi poznato kao Round Trip Time.

Algoritam za računanje optimalne brzine prenosa radi na sledeće način:

Svake sekunde šalju se dva paketa za ispitivanje (dvije PING poruke, PING1 i PING2) i to odmah jedna za drugom. Kao što je i ranije rečeno ovo slanje se obavlja iz posebne niti. U drugoj niti čeka se na odgovor od servera na PING poruke. Kada se odgovor primi računa se RTT vrijeme koje se zatim smješta u cirkularni bafer. Postoje dva ovakva bafera, pošto postoje i dvije poruke za ispitivanje koje se šalju (neka se zovu BAFER1 i BAFER2). Ukoliko je sve u redu približno svake sekunde se primaju po dva odgovora od servera (po jedan odgovor na svaku PING poruku). Kada se primljene poruke obrade dobijaju se RTT1 i RTT2 vremena, koja se zatim smeštaju u BAFER1 i BAFER2 respektivno. Međutim algoritam za računanje optimalne brzine prenosa još uvijek ne počinje da radi već se čeka da se cirkularni baferi koji sadrže RTT1 i RTT2 (BAFER1 i BAFER2) ne popune. Veličina cirkularnog bafera predstavlja promjenjivi dio algoritma koji može da se mijenja. Nakon što su se oba bafera popunila tek tada algoritam počinje da radi. Sada svaka nova poruka koja se primi, odnosno RTT1 i RTT2 se upisuju u cirkularni bafer i to se radi tako što se prepisuje najstarija vrijednost u baferu. Na ovaj način u baferima u svakom trenutku imamo n poslednjih vremena (BAFER1 sadrži RTT1 vremena, a BAFER2 sadrži RTT2 vremena). Pošto novu poruku primamo približno svake sekunde algoritam za računanje optimalne brzine prenosa takođe svake sekunde daje novi proračun (to jest nakon što se BAFER1 I BAFER2 popune, prije toga ne dobijaju se nikakvi rezultati).

Pošto izračunata optimalna brzina prenosa (A) dosta varira nije zgodno da se svake sekunde vrši prelazak na novi tok podataka. Zbog toga je algoritam dorađen tako i za izračunatu vrijednost optimalne brzine prenosa postoji jedan cirkularni bafer određene veličine (BAFER_A). I u ovom slučaju se čeka da se bafer prvo napuni. Nakon što se BAFER_A napuni računa se prosječna vrijednost optimalne brzine prenosa (AVR_A), što se radi i svaki sledeći put

kada se u bafer doda nova vrijednost. Posle svakih x dodatih vrijednosti A-optimalne brzine prenosa, u bafer, ukoliko je to potrebno, i moguće, vrši se promjena toka podataka. Naravno, u ovom slučaju se koristi srednja vrijednost optimalne brzine prenosa-AVR_A. Na ovaj način se smanjuje uticaj čestih varijacija izračunate optimalne brzine prenosa. Takođe, korišćenje srednje vrijednosti je bolje i zbog toka što u jednom trenutku optimalna brzina prenosa može naglo da se poveća i odmah u sledećem trenutku da se vратi na staru vrijednost. Ukoliko bi se na svaku promjenu vrijednosti vršila promjenu toka podataka moralo bi u vrlo kratkom roku dva puta da se prebacuje sa jednog toka podataka na drugi što nije dobro. Korišćenjem srednje vrijednosti-AVR_A ovi nagli skokovi, ili uopšte nagle i kratkotrajne promjene neće u tolikoj mjeri dolaziti do izražaja.

Pošto poslužilac posjeduje ograničen broj alternativnih tokova video podataka potrebno je da postoji i algoritam koji će da kaže koji od ponuđenih alternativnih tokova podataka najbolje odgovara i to na osnovu izračunate srednje vrijednosti optimalne brzine prenosa-AVR_A. Ovaj algoritam radi tako što se izabira onaj tok podataka sa najmanjom izračunatom pozitivnom razlikom između AVR_A-srednje vrijednosti optimalne brzine prenosa, i bitske brzine tog toka podataka. To znači da će biti izbran onaj tok podataka čija je bitska brzina po vrijednosti odmah ispod izračunate srednje vrijednosti optimalne bitske brzine. U slučaju da je izračunata bitska brzina veća od bitskih brzina svih alternativnih tokova podataka onda se izabira onaj tok podataka sa najmanjom bitskom brzinom.

Treba napomenuti i to da je program, nakon što je u njega dodat algoritam za računanje optimalne brzine prenosa, tačnije nakon što je dodat dio koji vrši slanje PING potuka, potrebno pokretati sa administratorskim pravima. Ovo je potrebno zbog toka što se za slanje PING poruka mora otvoriti tzv. *RAW socket* što zahtjeva administratorska prava.

4.2 Opis korišćenjih funkcija i programske strukture

U ovom dijelu dat je opis nekih od korišćenih struktura i funkcija koje su dodate kako bi se realizovao traženi algoritam.

4.2.1 Struktura koja predstavlja cirkularni bafer (struct circularBuffer)

```
typedef struct _circularBuffer
{
    double *head;
    double *tail;
    double *buffer;
    int noel;
    int size;
    double min;
    double max;
```

```

    double avr;
}circularBuffer;

```

Polja:

- `head`: pokazivač na prvi element u baferu
- `tail`: pokazivač na zadnji element u baferu
- `buffer`: pokazivač na memoriju bafera
- `noel`: broj elementa u baferu
- `size`: maksimalan broj elemenata
- `min`: vrijednost najmanjeg elementa
- `max`: vrijednost najvećeg elementa
- `avr`: prosječna vrijednost svih elemenata u baferu

4.2.2 Funkcija za stvaranje novog bafera zadate veličine (CircularBufferCreate)

Opis funkcionalnosti:

Stvara novi bafer tj. zauzima memoriju za samu strukturu kao i za zadati broj elemenata bafera.

Zaglavlje funkcije:

```
circularBuffer * CircularBufferCreate(int n);
```

Parametri funkcije:

- `n`: veličina bafera koji se kreira

Povratna vrijednost funkcije:

- pokazivač na strukrutu novostvorenog bafera

4.2.3 Funkcija za inicijalizaciju bafera (CircularBufferInit)

Opis funkcionalnosti:

Inicijalizuje vrijednosti svih elemenata u baferu na nulu.

Zaglavlje funkcije:

```
void CircularBufferInit(circularBuffer *buf);
```

Parametri funkcije:

- `buf`: pokazivač na strukturu koja predstavlja bafer

Povratna vrijednost funkcije:

- nema povratnu vrijednost

4.2.4 Funkcija za dodavanje novog elemanta u bafer (CircularBufferPut)

Opis funkcionalnosti:

Ova funkcija dodaje novi element u bafer i istovremeno ažirira vrijednosti najvećeg i najmanjeg elementa u baferu i izračunava novu prosječnu vrijednost svih elemenata u baferu.

Zaglavlje funkcije:

```
int CircularBufferPut(circularBuffer *buf, double n);
```

Parametri funkcije:

- **buf**: pokazivač na strukturu koja predstavlja bafer
- **n**: nova vrijednost koja se dodaje (novi element)

Povratna vrijednost funkcije:

- **CB_NO_ERR**: element uspješno dodat u bafer
- **CB_ERR_OVERFLOW**: element nije uspješno dodat u bafer – nema mesta u baferu (bafer omogućava zaštitu ukoliko je pun ne dozvoljava novi upis, u našem slučaju zaštita za tzv. overflow nije uključena tako da je uvijek moguće upisati novi element u bafer ali prepisivaljem preko najstarijeg elementa u baferu)

4.2.5 Funkcija za uništavanje bafera i slobađanje memorije

(CircularBufferDestroy)

Opis funkcionalnosti:

Oslobađa svu zauzetu memoriju.

Zaglavlje funkcije:

```
void CircularBufferDestroy(circularBuffer *buf);
```

Parametri funkcije:

- **buf**: pokazivač na strukturu koja predstavlja bafer

Povratna vrijednost funkcije:

- nema povratnu vrijednost

4.2.6 Funkcija koja služi za pronalaženje toka podataka koji nam najviše odgovara (find_best_suited_stream)

Opis funkcionalnosti:

Ova funkcija nam govori koji od ponuđenih alternativnih tokova podataka nam najbolje odgovara na osnovu izračunate strednje vrijednosti optimalne brzine prenosa. Funkcija radi tako što se izabira onaj tok podataka sa najmanjom izračunatom pozitivnom razlikom između izračunate optimalne brzine prenosa (odnosno njene srednje vrijednosti) i bitske brzine tog toka podataka.

Zaglavlje funkcije:

```
int find_best_suited_stream(int optimal_bandwidth, VideoState *vs);
```

Parametri funkcije:

- **optimal_bandwidth**: optimalna bitska brzina prenosa koju je izračunao naš algoritam
- **vs**: pokazivač na strukturu **videostate** koja sadrži informacije o svim raspoloživim tokovima podataka (između ostalog i o njihovoj bitskoj brzini)

Povratna vrijednost funkcije:

- indeks izabranog toka podataka u strukturi **videostate**

4.2.7 Funkcija za izdvajanje dijela URL adrese koji se odnosi na server (get_ready_url)

Opis funkcionalnosti:

Ova funkcija izdvaja iz URL adrese dio koji se odnosi na računar (server) kako bi se omogućilo slanje PING poruka

Zaglavlje funkcije:

```
char * get_ready_url(char *url);
```

Parametri funkcije:

- **url**: URL adresa

Povratna vrijednost funkcije:

- pokazivač na string u kome se nalazi izdvojeni dio adrese

4.2.8 Funkcija za obradu odgovora od servera na PING poruku (pr_pack)

Opis funkcionalnosti:

Ova funkcija vrši obradu odgovora na PING poruku, između ostalog ovdje se računa RTT vrijeme koje se dodaje u dgovarajući bafer

Zaglavlje funkcije:

```
void pr_pack(char *buf, int cc, struct sockaddr_in *from);
```

Parametri funkcije:

- **buf**: bafer u kome se nalaze primljeni podaci (kao odgovor servera na PING poruku)
- **cc**: količina primljenih podataka u bajtima
- **from**: struktura koja opisuje server tj. onog od koga je poruka primljena

Povratna vrijednost funkcije:

- ne vraća ništa

4.2.9 Funkcija za slanje PING poruka (pinger)

Opis funkcionalnosti:

Ova funkcija vrši slanje PING poruka.

Zagлавlje funkcije:

```
void pinger(int num);
```

Parametri funkcije:

- `num`: ovaj parametar nam služi da razlikujemo dvije PING poruke tako da ima u našem slučaju imao vrijednost ili 1 ili 2.

Povratna vrijednost funkcije:

- ne vraća ništa

5. Rezultati

Realizovani algoritam je isprobao uz pomoć postojećeg Apple-ovog toka podataka za ispitivanje HTTP Live Streaming-a. Kada se pokrene *ffplay* aplikacija kojoj se kao ulaz zada URL adresa index datoteke koja se odnosi na testni tok podataka prvo se provjerava o kakvom je ulazu riječ. Tek kada se utvrdi da se radi o HTTP Live Streaming-u aktivira se nit koja vrši procjenu optimalne brizine video toka podataka. Međutim, djelovanje realizovanog algoritma se ne primjećuje odmah po pokretanju jer je potrebno da se popune baferi sa potrebnim podacima kako bi proračun bio potpun.

U zavisnosti od veličine bafera kao i od toga koliko često se prema poslužiocu šalju poruke za ispitivanje (PING poruke) zavisi koliko dugo vremena će biti potrebno da se prikupe svi podaci da bi algoritam mogao da počne da radi. Svi ovi parametri su podešivi (veličina bafera kao i učestalost slanja PING poruka) tako da ih je moguće prilagoditi što na kraju utiče na to koliko često može da se dešava promjena sa jednog toka podataka na drugi. Praktično je probano više kombinacija ovih parametara i utvrđeno je da česte promjene nisu dobre jer sam prelaz sa jednog toka podataka na drugi nije neprimjetan tako da ako to radimo često postaje naporno.

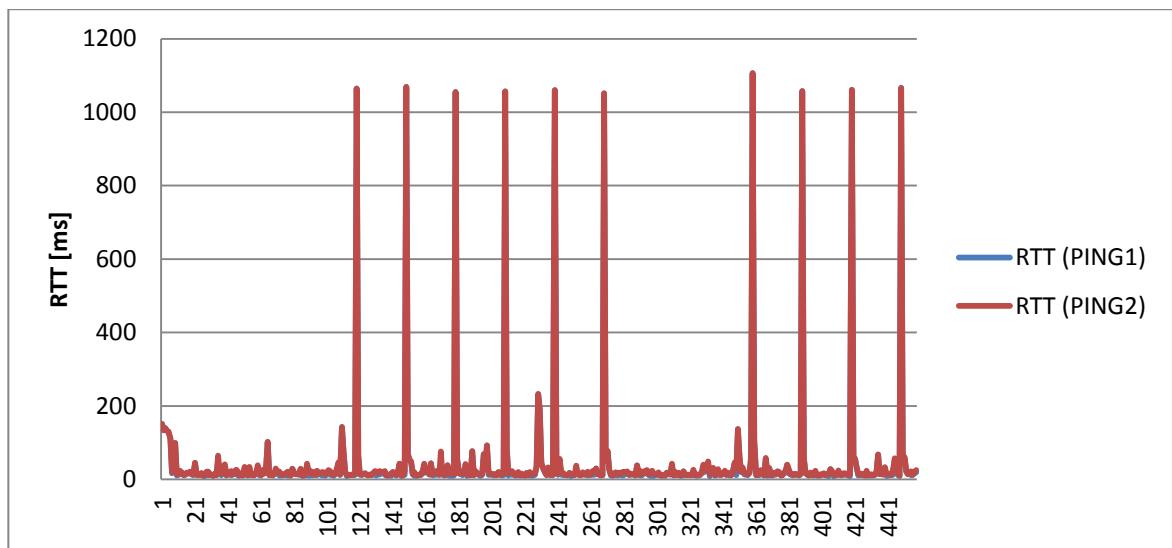
Prije nego se pređe na prikaz rezultata dobijenih u realnim uslovima treba napomenuti da trenutna realizacija HLS protokola u FFmpeg biblioteci radi tako da u svakom trenutku ima u memoriji tri segmenta unaprijed. Ovo je bitno zbog toga što se prilikom prelaska sa jednog toka podataka na drugi tok podataka dešava da se odjednom dobavljuju tri segmenta, i tek nakon toga se na svakih 10 sekundi dobavlja po još jedan segment. Na ovaj način u svakom trenutku će u memoriji da se nalaze 3 segmenta unaprijed.

Testirani tok podataka se sastoji od ukupno 8 tokova (osnovni tok podataka zajedno sa svim alternativnim tokovima) koji su enkodovani na različitim bitskim brzinama:

- Tok 0 enkodovan na 688 301 [bits/s],

- Tok 1 enkodovan na 165 135 [bits/s],
- Tok 2 enkodovan na 262 346 [bits/s],
- Tok 3 enkodovan na 481 677 [bits/s],
- Tok 4 enkodovan na 1 308 077 [bits/s],
- Tok 5 enkodovan na 1 927 853 [bits/s],
- Tok 6 enkodovan na 2 650 941 [bits/s],
- Tok 7 enkodovan na 3 477 293 [bits/s].

Na sledećoj slici prikazano je izmjereno vrijeme (RTT) za PING poruke, pri čemu su po dvije PING poruke slate svake sekunde:

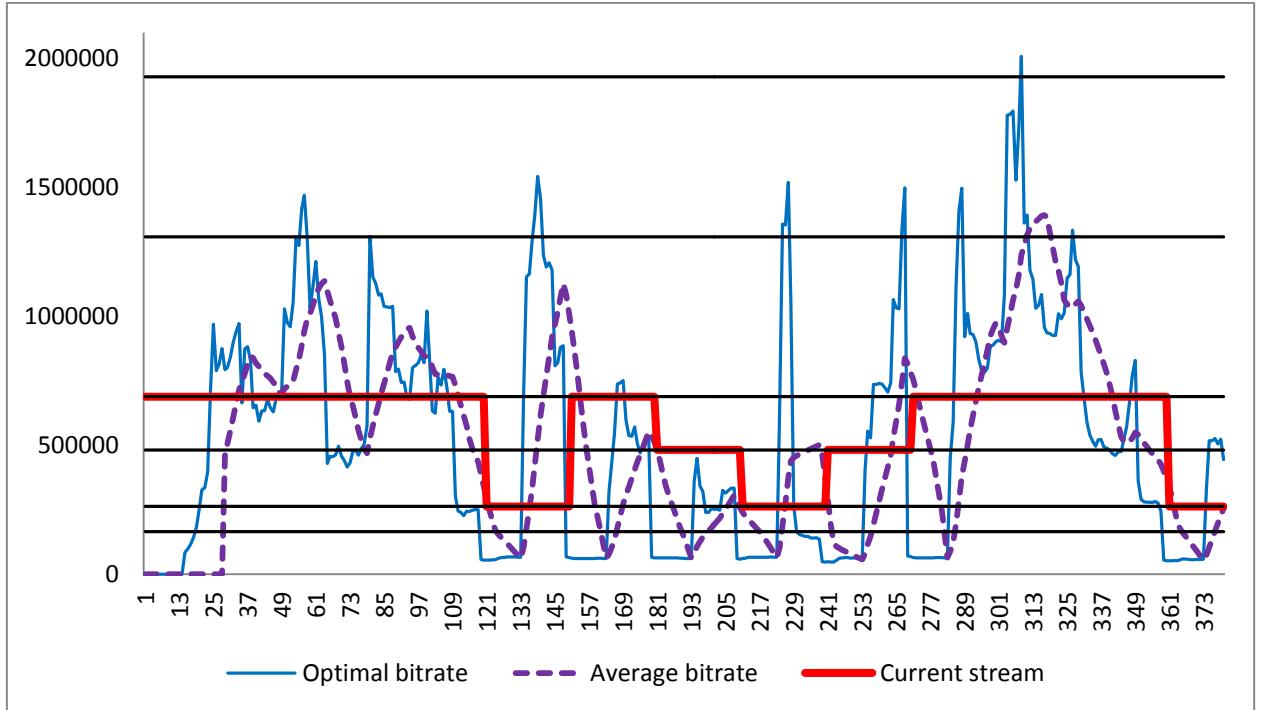


Slika 5.1 – Prikaz vrijednosti vremena RTT u realnim uslovima

Kao što može da se vidi sa slike, vrijeme RTT u određenim momentima naglo skače. To se dešava nakon što se izvrši prelazak na novi tok podataka. Kao što je ranije napomenuto, u tom trenutku poslužiocu se šalju zahtjevi za 3 naredna segmenta i to je glavni uzrok zašto se dešava taj nagli skok. U tom trenutku poslužilac je zauzet i nije u stanju da odmah odgovori na PING poruke što se i vidi na slici. Pored ovih velikih promjena mogu se primjetiti i promjene koje nisu toliko velike ali su ipak primjetne. Jedna od takvih promjena je recimo i promjena odmah prije prvog velikog skoka (gdje RTT dostiže vrijednost preko 1000 ms). I ove manje promjene vrijednosti RTT su posledica dobavljanja pojedinih segmenata. Jasno je da na vrijednost RTT utiče to da li je od poslužioca zahtijevano da dostavi neki segment, kao i da li se radi o jednom ili o više segmenata (tri u ovom konkretnom slučaju). U suštini vremena RTT predstavljaju zauzetost poslužioca i prenosnog puta.

Kao što će biti prikazano na sledećoj slici ovi nagli skokovi imaju veliki uticaj na izračunatu vrijednost optimalne brzine prenosa toka podataka. Uticaj tih velikih vrijednosti RTT ne nestaje odmah, već traje još određeno vrijeme. To se dešava zbog toga što se prilikom

računanja optimalne brzine prenosa u obzir uzima n poslednjih vrijednosti i sve dok ta velika vrijednost RTT nastala naglim skokom ne izade iz opsega računanja izračunata vrijednost optimalne brzine prenosa će biti jako mala.

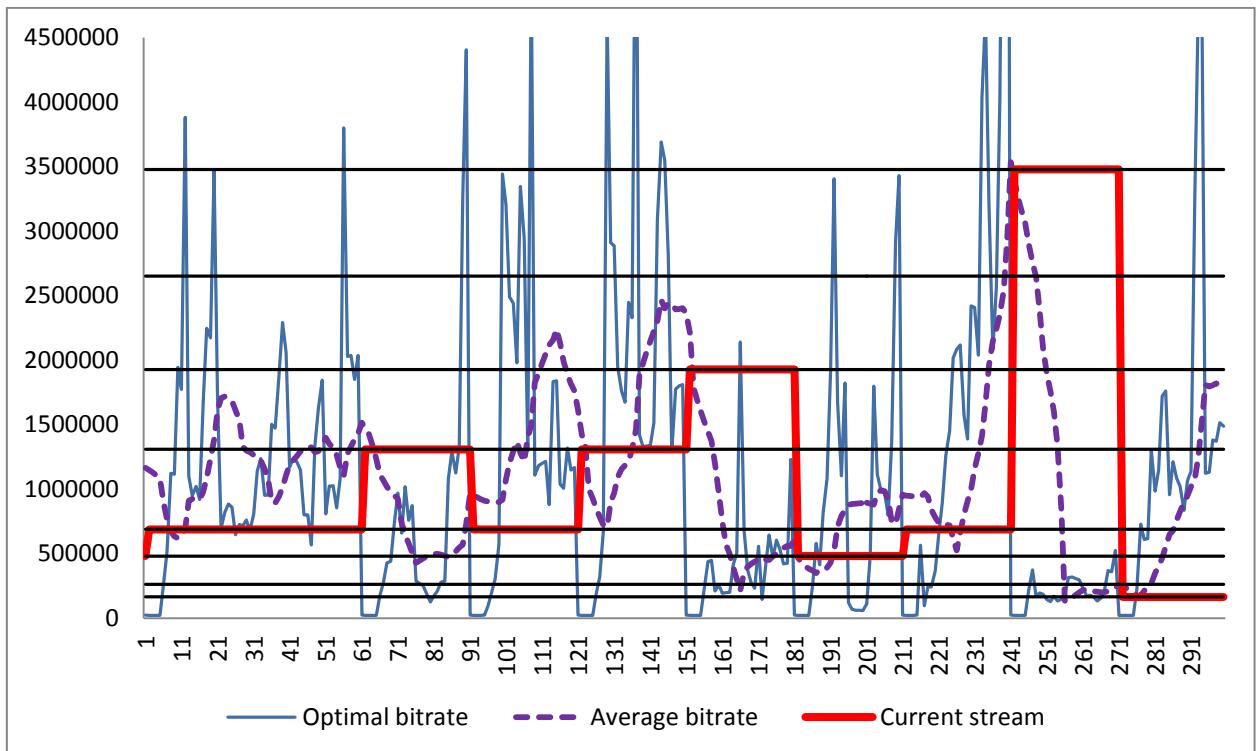


Slika 5.2 – Ponašanje sistema u realnim uslovima

Plava linija predstavlja optimalnu brzinu prenosa koja je dobijena po zadatom algoritmu. Primjećuje se da ona dosta varira, a jasno se vidi i da je nakon promjene toka podataka vrijednost optimalne brzine prenosa jako mala, što je upravo posljedica naglog skoka vremena RTT. Optimalna brzina prenosa ostaje jako mala sve dok ta velika vrijednost RTT ne izade iz opsega računanja. Pošto optimalna brzina prenosa dosta varira umjesto nje koristi se srednja vrijednost optimalne brzine prenosa koja predstavlja usrednjenu vrijednost zadnjih m izračunatih vrijednosti optimalne brzine prenosa. Pošto algoritam za procjenu optimalne brzine prenosa kao rezultat daje propusni opseg koji je dostupan u određenom trenutku nije od koristi posmatrati period u kome je došlo do naglog skoka vrijednosti RTT odnosno kada se od poslužioca primaju podaci. Razlog za to je što je u tom period određeni dio propusnog opsega zauzet (posljedica prijema podata od poslužioca) па се као rezultат алгоритма добија само преостали доступни propusni opseg. Управо због тога потенцијално пребацивање са једног тока података на други се не треба вршити сувише често јер се може догодити да се подаци и даље примају од послуžиоца, а да ми баš у том тренутку одлучимо да се пребачимо на други ток података, али због тога што је веза оптерећена ми не добијамо добре резултате јер у том тренутку не знамо колики је стварни propusni opseg већ само колики је преостали доступни propusni opseg у том тренутку. Као резултат овога

dobiće se mnogo manja vrijednost od stvarne i vjerovatno će se izvršiti prelazak na tok podataka lošijeg kvaliteta.

Spomenuta situacija prikazana je na slici 5.3:



Slika 5.3 – Prikaz posledice korišćenja rezultata koji ne prikazuju pravo stanje

U ovom slučaju broj vrijednosti RTT koje ulaze u proračun algoritma je smanjen sa 15 (prikazani slučaj na slici 5.2) na 5. To se može vidjeti po tome što se nakon promjene toka podataka linija koja predstavlja izračunatu optimalnu brzinu prenosa mnogo kraće zadržava na jako malim vrijednostima. Na ovaj način je pokušano da se smanji vjerovatnoća da će prilikom odabira toka podataka biti uzete u obzir netačne vrijednosti. Međutim, kao što se vidi sa grafika, oko 240. sekunde došlo je do promjene toka pri čemu se prešlo na tok enkodovan na jako velikoj bitskoj brzini (3 500 000 bita/s). Pošto se prešlo na novi tok podataka, bilo je potrebno dobaviti 3 segmenta ali ovaj put pošto su segmenti enkodovani na mnogo većoj bitskoj brzini, iako je vrijeme trajanja reprodukcije svih segmenta isto (10 sekundi) memorijski gledano, ova tri nova segmenta zauzimaju mnogo više memorije, pa je potrebno i mnogo više vremena da bi se ovi segmenti preuzeли od poslužioca. To je uticalo na to da je vrijeme koje je potrebno da se preuzmu ta tri segmenta postalo veliko, što je dalje uticalo na rezultate algoritma koji je kao rezultat davao da je dostupni propusni opseg jako mali. Kao posljedica ovoga čim je prošlo 30 sekundi (to je vrijeme nakon kogase provjerava da li je potrebno preći na neki drugi tok) prešlo se na tok sa najmanjom bitskom brzinom, iako je vjerovatno bilo moguće da se zadrži isti tok podataka ili da se eventualno pređe na tok na nešto manjoj bitskoj brzini.

6. Zaključak

U ovom radu prikazano je rješenje jednog algoritma za procjenu optimalne brzine prenosa video toka podataka u okviru HTTP Live Streaming protokola. Praktično je dokazano da algoritam radi ali sam algoritam se zasniva na PING porukama koje u svom osnovnom obliku nisu najsvrsishodnije u našem slučaju. Generalno gledano PING poruke (odnosno vrijeme RTT) govore koliko je kašnjenje do nekog računara na internetu (poslužioca u našem slučaju) a ne koliki je dostupni propusni opseg.

Međutim ukoliko se poveća veličina PING poruke (što je i urađeno u konkretnoj realizaciji) dobijaju se pouzdaniji rezultati. Ali i pored ovoga, ako prepostavimo da se dobijaju tačni rezultati javljaju se drugi problemi. Jedan od njih je to što nisu svi rezultati upotrebljivi.

Naime, ispostavlja se da su upotrebljivi samo oni rezultati koji su dobijeni kada server nije slao nikakve podatke, odnosno kada nije bilo saobraćaja koji uzrokuje sama aplikacija koja služi za reprodukciju. U tom smjeru algoritam je potrebno dodatno poboljšati i prilagoditi kako bi se izbjegao uticaj rezultata koji ne predstavljaju sliku pravog stanja.

7. Literatura

- [1] James F. Kurose, Keith W.Ross: *Computer Networking: A Top Down Approach*, 5th edition. Addison Wesley, Chapter 7, 2009
- [2] R.Fielding, UC Irvine, J.Gettys, J.Mogul, H.Frystyk, L.Masinter, P.Leach and T.Berners-Lee: *Hypertext Transfer Protocol – HTTP-1.1*, IETF Network Working Group, Request for Comments: 2616, April 1999
- [3] Tomas Stockhammer: *Dynamic Adaptive Streaming over HTTP – Design Principles and Standards*
- [4] Apple Inc.: *HTTP Live Streaming Overview*, 2009, Online PDF: <https://developer.apple.com/library/ios/#documentation/NetworkingInternet/Conceptual/StreamingMediaGuide>
- [5] Andrew Fecheyrs-Lippens: *A review of HTTP Live Streaming*
- [6] R.Pantos, *HTTP Live Streaming: draft-pantos-http-live-streaming-07*, 2009, Online PDF: <http://tools.ietf.org/pdf/draft-pantos-http-live-streaming-07.pdf>
- [7] A. Delphinanto, T. Koonen, F. den Hartog: *Real-time probing of available bandwidth in home networks*, IEEE Communications Magazine, June 2011, Vol 49, No 6