Analiza i implementacija adaptacijskih algoritama korištenih u adaptivnom prijenosu video signala koji u obzir uzimaju veličinu segmenta

Grabić, Dominik

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

Sveučilišni studij

ANALIZA I IMPLEMENTACIJA ADAPTACIJSKIH ALGORITAMA KORIŠTENIH U ADAPTIVNOM PRIJENOSU VIDEO SIGNALA KOJI U OBZIR UZIMAJU VELIČINU SEGMENTA

Diplomski rad

Dominik Grabić

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 25.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Dominik Grabić
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D876 R, 20.09.2017.
OIB studenta:	94582666526
Mentor:	Prof.dr.sc. Drago Žagar
Sumentor:	Jelena Vlaović
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. Krešimir Grgić
Član Povjerenstva:	Doc. dr. sc. Višnja Križanović
Naslov diplomskog rada:	Analiza i implementacija adaptacijskih algoritama korištenih u adaptivnom prijenosu video signala koji u obzir uzimaju veličinu segmenta
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	Kako bi se krajnjem korisniku pružila najbolja moguća kvaliteta sadržaja, predložen je standard za prijenos multimedijskog signala zasnovan na HTTP protokolu koji se koristi za prijenos vezanih i nevezanih prijenosnih tokova multimedijskih podataka. MPEG DASH definira način obavještavanja korisnika o nizu prijenosnih tokova različite kvalitete, zajedno s informacijama potrebnim za odabir odgovarajućeg prijenosnog toka. Također definira formate medijskih datoteka pogodnih za adaptivni prijenos. Postojanje različitih formata datoteka omogućuje učinkovito i neprimjetno prebacivanje između prijenosnih tokova različite kvalitete,što omogućuje prilagodbu promjenjivim uvjetima u mreži bez zaustavljanja prikaza video zapisa. Algoritmi za
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: 2 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	25.09.2018.

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, 26.09.2018.

Ime i prezime studenta:	Dominik Grabić
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D876 R, 20.09.2017.
Ephorus podudaranje [%]:	5

Ovom izjavom izjavljujem da je rad pod nazivom**: Analiza i implementacija adaptacijskih algoritama korištenih u adaptivnom prijenosu video signala koji u obzir uzimaju veličinu segmenta**

izrađen pod vodstvom mentora Prof.dr.sc. Drago Žagar

i sumentora Jelena Vlaović

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

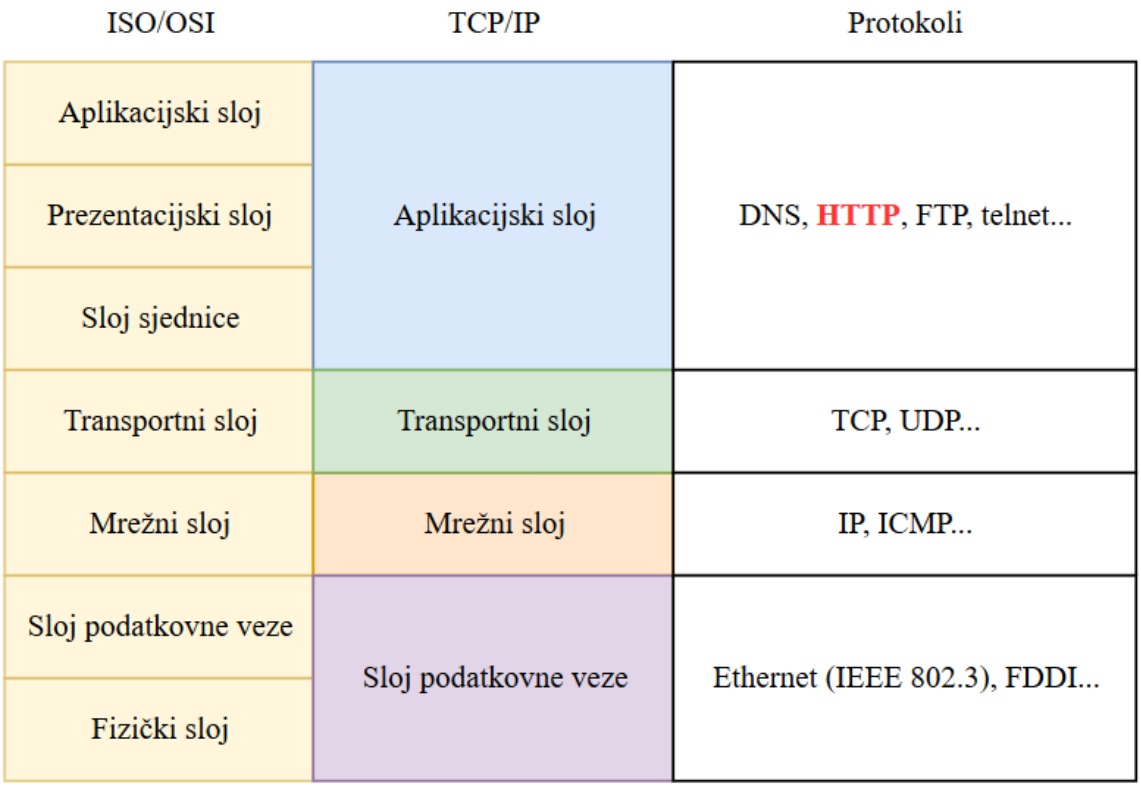
U ovom diplomskom radu opisani su i implementirani neki od adaptacijskih algoritama korištenih pri adaptivnom prijenosu video signala. Kako bi se krajnjem korisniku pružila najbolja moguća kvaliteta sadržaja, predložen je standard za prijenos multimedijskog signala zasnovan na HTTP protokolu (eng. *Hypertext Transfer Protocol*) koji se koristi za prijenos vezanih i nevezanih prijenosnih tokova multimedijskih podataka. MPEG-DASH standard (eng. *Moving Picture Experts Group Dynamic Adaptive Streaming over HTTP*) definira način obavještavanja korisnika o nizu prijenosnih tokova različite kvalitete s informacijama potrebnim za odabir odgovarajućeg prijenosnog toka. Također definira formate medijskih datoteka pogodnih za adaptivni prijenos. Postojanje različitih formata datoteka omogućuje učinkovito i neprimjetno prebacivanje između prijenosnih tokova različite kvalitete, što omogućuje prilagodbu promjenjivim uvjetima u mreži bez zaustavljanja prikaza video zapisa. Ovaj rad bavi se skupinom adaptacijskih algoritama koji uzimaju u obzir podatke o veličinama pojedinih segmenata pri izboru sljedećeg segmenta u svrhu poboljšanja kvalitete preuzetog video sadržaja.

Rad je podijeljen u tri veće cjeline. Najprije je opisana teorijska podloga potrebna za razumijevanje tematike. Tu se spominju korištene tehnologije i standardi kao i klijentski program na kojem se temelje algoritmi. Naglasak u tom poglavlju stavljen je na opis MPEG-DASH standarda. Druga cjelina bavi se analizom tri odabrana adaptacijska algoritma koji su implementirani u okviru ovog rada. Unutar ove cjeline detaljno su opisani principi rada svakog algoritma te su prikazani pripadajući dijagrami toka i pseudokodovi. U posljednjoj cjelini opisana je metoda testiranja implementiranih algoritama i prikazani su rezultati testiranja.

2. OPIS TEHNOLOGIJA

2.1. HTTP protokol

HTTP je protokol aplikacijskog sloja TCP/IP (eng. *Transmission Control Protocol / Internet Protocol*) mrežnog modela namijenjen prijenosu podataka na računalnoj mreži (eng. *WWW – World Wide Web*) kako je prikazano slikom 2.1. Temelji se na modelu klijent-poslužitelj, a standardni port za ostvarivanje veze je port 80. Klijent i poslužitelj komunikaciju ostvaruju nizom naredbi definiranih HTTP-om. Naredbe koje mogu biti zahtjevi ili odgovori izvršavaju se korištenjem usluga nižih slojeva mreže. HTTP se ne koristi isključivo za prijenos datoteka nego i za prijenos drugih resursa. U resurse spadaju svi dijelovi informacija čija se lokacija može opisati pomoću adrese (eng. *URL - Uniform Resource Locator*). U inačici HTTP/1.0 za svaku razmjenu zahtjev-odgovor bilo je potrebno otvoriti novu vezu, dok se u inačici HTTP/1.1 jedna veza može koristiti za više transakcija čime se optimizira upotreba resursa i smanjuje mogućnost zagušenja mreže [1].



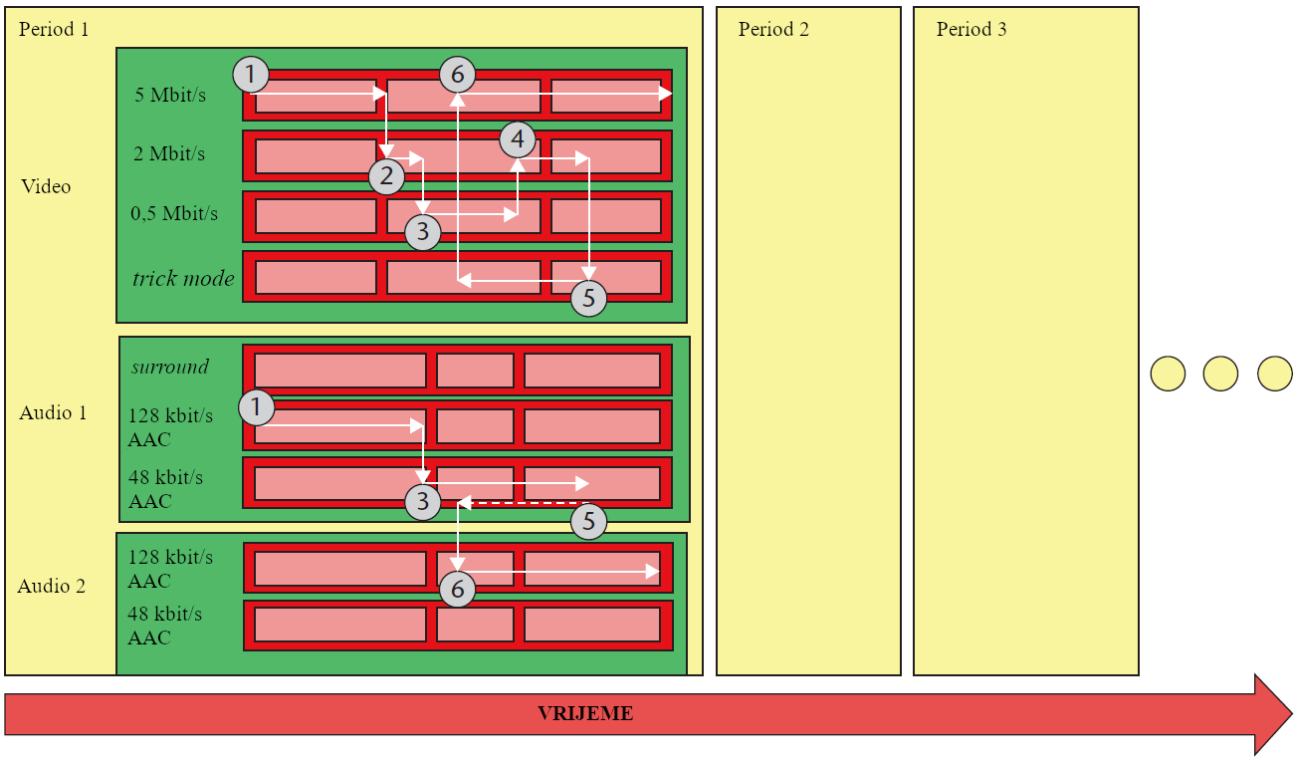
Slika 2.1. Prikaz HTTP-a u TCP/IP modelu [2]

2.2. MPEG-DASH standard

Tijekom 90-ih godina prošlog stoljeća pojavila se potreba za prijenosom video sadržaja preko Interneta. Glavni izazov bio je osigurati prijenos velike količine podataka u odgovarajućem vremenu. U tu svrhu grupa mrežnih inženjera razvila je RTP protokol (*eng. Real-Time Transport Protocol*) čija je namjena bila definirati formate audio i video paketa i omogućiti njihov prijenos uz učinkovito korištenje resursa. Međutim, problem s RTP protokolom bio je taj što paketi često nisu bili propuštani kroz vatrozid (eng. *firewall*). Osim toga, RTP prijenos toka zahtijevao je od poslužitelja pojedinačno upravljanje sjednicom sa svakim korisnikom što je kompliciralo proširenje usluge. Kako se brzina mreže nastavila povećavati više nije bilo potrebe za prijenosom audio i video tokova u manjim paketima. Pojavila se mogućnost prenošenja većih segmenata podataka korištenjem HTTP-a. Takav prijenos podatkovnih tokova ima više prednosti. Budući da se danas velika količina sadržaja distribuira putem mreže za isporuku sadržaja (eng. *CDN - Content Delivery Network*) omogućava se brži prijenos odabirom manje opterećenih i geografski bližih čvorova. Nadalje, mrežna infrastruktura razvijala se u skladu s HTTP protokolom, a mrežni uređaji u većini slučajeva bez dodatne konfiguracije podržavaju uspostavljanje veze s HTTP poslužiteljem. Pružanje usluge velikom broju korisnika je također lako izvedivo jer nema potrebe za pojedinačnim održavanjem sjednice. Zbog navedenih razloga prijenos podataka na ovaj način pokazao se perspektivnim za komercijalne svrhe. Danas postoji više vlasničkih rješenja za prijenos toka podataka korištenjem HTTP protokola od kojih su neka: Appleov HTTP Live Streaming (HLS), Microsoftov Smooth Streaming i Adobeov HTTP Dynamic Streaming. Svako od tih rješenja koristi različite predloške i formate segmenata te zato, kako bi primio sadržaj s poslužitelja, uređaj mora podržavati pripadajući protokol. Promatrajući stanje na tržištu grupa stručnjaka pod nazivom MPEG izdaje 2009. godine poziv za suradnju u stvaranju novog standarda koji bi omogućio usklađenost klijenata i poslužitelja različitih dobavljača. Suradnjom grupe MPEG i mnogih drugih stručnjaka definiran je i konačno 2012. godine odobren standard MPEG-DASH [2,3].

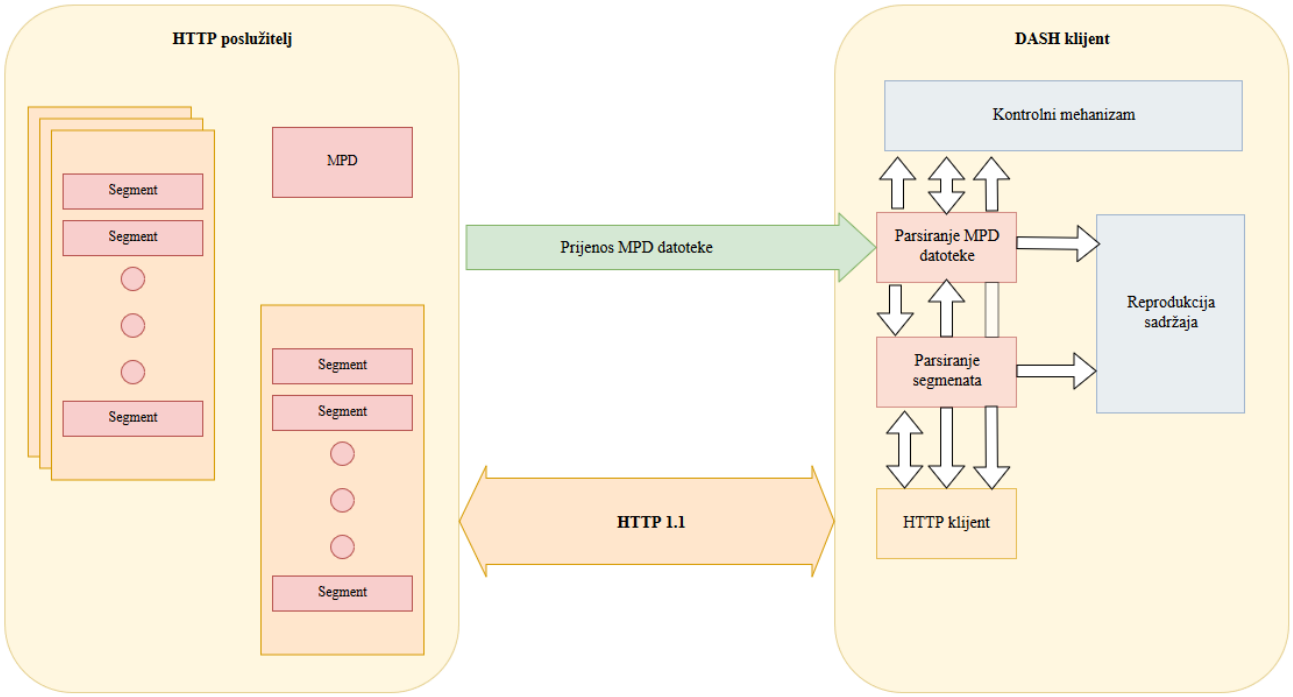
Slika 2.2. predstavlja jedan mogući slučaj korištenja dinamičkog prijenosnog toka audio i video signala. Ulazni video signal ovdje je kodiran u tri različite razine kvalitete uvjetovane brzinama prijenosa od 5, 2 i 0,5 Mbit/s. Posebni način rada kod video toka je tzv. *trick mode* kod kojeg se radi o toku videa s nižim brojem sličica u sekundi sastavljenim isključivo od I-okvira. Audio sadržaj pripadajućeg toka dostupan je na dva jezika, engleskom i francuskom. Sadržaj oba jezika kodiran je

u 48 i 128 kbit/s AAC (eng. *Advanced Audio Coding*) načinu, dok je dodatno za engleski jezik omogućen i *surround* način rada. Na početku (korak 1) uređaj započinje prijenos sadržaja dohvaćanjem video sadržaja najveće kvalitete te odabirom audio sadržaja na engleskom jeziku uz 128 kbit/s AAC kodiranje. Uzimajući u obzir propusnost mreže nakon preuzimanja prvog segmenta, klijent izračuna da je stvarna brzina manja od 5 Mbit/s. U tom slučaju, kako ne bi došlo do prekida u izvođenju sadržaja, uređaj ostaje pri prvotnom odabiru audio toka, ali odluči odabrati video sadržaj kvalitete 2 Mbit/s (korak 2). Ako brzina prijenosa i dalje nastavi padati tada će uređaj odabrati video tok kvalitete 0,5 Mbit/s i audio tok kvalitete 48 kbit/s (korak 3). Uređaj će nastaviti preuzimati sadržaj prema ovim parametrima sve dok ponovno ne dođe do promjene u parametrima mreže (korak 4). Ako korisnik odluči privremeno zaustaviti prijenos, prelazi se u *trick mode* način rada. U tom načinu rada video se reproducira unazad a zvuk je isključen (korak 5). Ponovnim pokretanjem sadržaj se nastavlja reproducirati kako je naznačeno korakom 6. Ovime su obuhvaćene osnovne funkcionalnosti dinamičkog prijenosa multimedijskog sadržaja. Neki napredniji slučajevi korištenja mogu uključivati pregled 3D sadržaja, uključivanje teksta ili dinamičko ubacivanje reklamnog sadržaja [3].



Slika 2.2. Slučaj korištenja dinamičkog prijenosa u MPEG-DASH-u [3]

Multimedijski sadržaj kod MPEG-DASH standarda je pohranjen na poslužitelju i prenosi se korištenjem HTTP-a. Sadržaj je podijeljen na dva dijela. Prvi dio predstavlja MPD datoteku (eng. *Media Presentation Description*) koja sadrži podatke o dostupnom multimedijskom sadržaju kao što su URL adrese i druge karakteristike. Drugi dio sadržaja na poslužitelju su multimedijski tokovi podataka koji se nalaze u jednoj ili više datoteka. Za reprodukciju sadržaja nužno je najprije preuzeti MPD datoteku. Parsiranjem sadržaja te datoteke klijentski program dobiva podatke o dostupnosti medija, razinama kvalitete sadržaja, informacije o zaštiti prava, razlučivostima itd. Nakon parsiranja klijent započinje preuzimanje sadržaja koristeći HTTP GET zahtjeve. MPEG-DASH standard definira jedino sadržaj MPD datoteke i formate segmenata. Način isporuke MPD datoteke ni adaptacijski algoritmi za odabir segmenata nisu zadani standardom. Slika 2.3. predstavlja općenitu strukturu HTTP poslužitelja i DASH klijenta. Komponente obojene crvenom bojom obuhvaćene su standardom [3].



Slika 2.3. Primjer strukture klijenta i poslužitelja u MPEG-DASH standardu [3]

2.2.1. MPD datoteka

MPD datoteka je vrsta XML dokumenta (eng. *Extensible Markup Language*) koja sadrži lokacije segmenata i druge metapodatke potrebne klijentu za dohvaćanje sadržaja. Podaci MPD datoteke postavljeni su hijerarhijski. Na vrhu hijerarhije nalazi se jedan ili više perioda dok su na dnu postavljeni segmenti i podsegmenti. U nastavku su redom opisani svi elementi.

2.2.2. Period

Svaka MPD datoteka se sastoji od jednog ili više perioda. Periodi sadrže podatke za medijske komponente poput kuta snimanja i korištene načine kodiranja, jezike audio komponenti i detalje o dostupnim prijevodima i komentarima. Unutar perioda značajke komponenti se ne mijenjaju. Tijekom trajanja jednog perioda klijent može odabirati bilo koji od dostupnih razlučivosti, brzina ili korištenih kodeka. Nadalje, ovim pristupom moguće je odvojiti reklamni sadržaj od programa te se tako korisniku može prikazati personalizirani sadržaj.

2.2.3. Adaptacijski skup

Unutar jednog perioda postavlja se jedan ili više adaptacijskih skupova (eng. *adaptation set*) koji grupiraju različite multimedijske komponente. Na primjer, komponente koje koriste isti način kodiranja, jezik, razlučivost ili druge karakteristike mogu se postaviti u jedan skup. Korištenjem tog mehanizma klijent može vrlo brzo doći do onih komponenti koje su zanimljive korisniku s obzirom na njegove zahtjeve. Svaki adaptacijski skup obično sadrži više reprezentacija.

2.2.4. Reprezentacija multimedijskog sadržaja

Adaptacijski skup sadrži reprezentacije sadržaja (eng. *representation*). Reprezentacije omogućuju spremanje istog sadržaja kodiranog na različite načine. U većini slučajeva kod video komponenti tako je moguće odabrati željenu razlučivost ili brzinu prijenosa. Nadalje, korisnicima koji sadržaju pristupaju sa starijih uređaja moguće je ponuditi video kodiran starijim kodekom dok će se korisniku s novijim uređajem ponuditi sadržaj kodiran novijim podržanim kodekom. Obično se reprezentacije biraju automatski, a neki klijenti će korisniku dopustiti ručno postavljanje razlučivosti. Ako u jednoj reprezentaciji postoje i video i audio tokovi, a treba postaviti parametre za samo jedan od njih, tada se ti parametri spremaju u podreprezentaciju.

2.2.5. Medijski segment

Medijski segmenti su datoteke s medijskim sadržajem koje se u pravilu reproduciraju jedna za drugom kao da su jedna datoteka. Sadržajima segmenata se pristupa preko URL adrese koja je zapisana u ovom elementu. Segmenti unutar reprezentacije imaju jednake duljine i poredani su prema redu izvođenja. MPEG-DASH standard ne propisuje duljinu segmenta niti ne daje preporuku za njihovu duljinu [4]. Kraći segmenti obično traju između jedne i deset sekundi dok duži segmenti mogu trajati i po dva sata. Tablica 2.1. prikazuje prednosti i nedostatke kratkih i dugačkih segmenata.

Tablica 2.1. Značajke segmenata različitih duljina [5]

Trajanje segmenta	Prednosti	Nedostatci
Kratko	Pogodni za prijenos uživo Više mogućnosti za promjenu razine kvalitete	Velik broj datoteka Velik broj URL adresa Velik broj zahtjeva prema poslužitelju
Dugo	Manji broj datoteka Manji broj URL adresa Bolja mogućnost kompresije Bolje korištenje predmemorije	Potreba za indeksiranjem segmenata Različit od prijenosa uživo

Tijekom preuzimanja i reprodukcije sadržaja nije moguće proizvoljno vršiti promjene reprezentacija nego se moraju poštivati ograničenja. Segmenti se ne smiju preklapati niti smiju biti međuovisni. Za rješavanje tog problema MPEG-DASH uvodi točke pristupa toku (eng. *SAP - Stream Access Point*).

Referenciranje segmenata unutar reprezentacije vrši se na jedan od tri načina:

- korištenjem jednog ili više *SegmentList* elementa
- jednim predloškom *SegmentTemplate*
- jednim ili više elementom tipa *BaseURL* s maksimalno jednim *SegmentBase* elementom bez prethodno definiranih *SegmentList* i *SegmentTemplate* elemenata

SegmentBase

Najjednostavniji način referenciranja segmenta je uporabom *SegmentBase* elementa koji se koristi u slučaju gdje unutar jedne reprezentacije postoji samo jedan medijski segment. Tada se taj segment označava URL-om unutar elementa *BaseURL*. Primjer takvog referenciranja prikazan je na slici ispod.

```
<Representation mimeType="video/mp4"
                    frameRate="24"
                    bandwidth="1558322"
                    codecs="avc1.4d401f" width="1277" height="544">
  <BaseURL>http://cdn.bitmovin.net/bbb/video-1500k.mp4</BaseURL>
  <SegmentBase indexRange="0-834"/>
</Representation>
```

Slika 2.4. Primjer uporabe BaseURL elementa [4]

SegmentList

Lista *SegmentURL* elemenata koji se trebaju zadanim redoslijedom reproducirati nalazi se u elementu *SegmentList*. Svaka URL adresa sadrži lokaciju segmenta, a po potrebi i raspon okteta.

```
<Representation mimeType="video/mp4"
                    frameRate="24"
                    bandwidth="1558322"
                    codecs="avc1.4d401f" width="1277" height="544">
  <SegmentList duration="10">
    <Initialization sourceURL="http://cdn.bitmovin.net/bbb/video-1500/init.mp4"/>
    <SegmentURL media="http://cdn.bitmovin.net/bbb/video-1500/segment-0.m4s"/>
    <SegmentURL media="http://cdn.bitmovin.net/bbb/video-1500/segment-1.m4s"/>
    <SegmentURL media="http://cdn.bitmovin.net/bbb/video-1500/segment-2.m4s"/>
    <SegmentURL media="http://cdn.bitmovin.net/bbb/video-1500/segment-3.m4s"/>
    <SegmentURL media="http://cdn.bitmovin.net/bbb/video-1500/segment-4.m4s"/>
  </SegmentList>
</Representation>
```

Slika 2.5. Primjer uporabe SegmentList elementa [4]

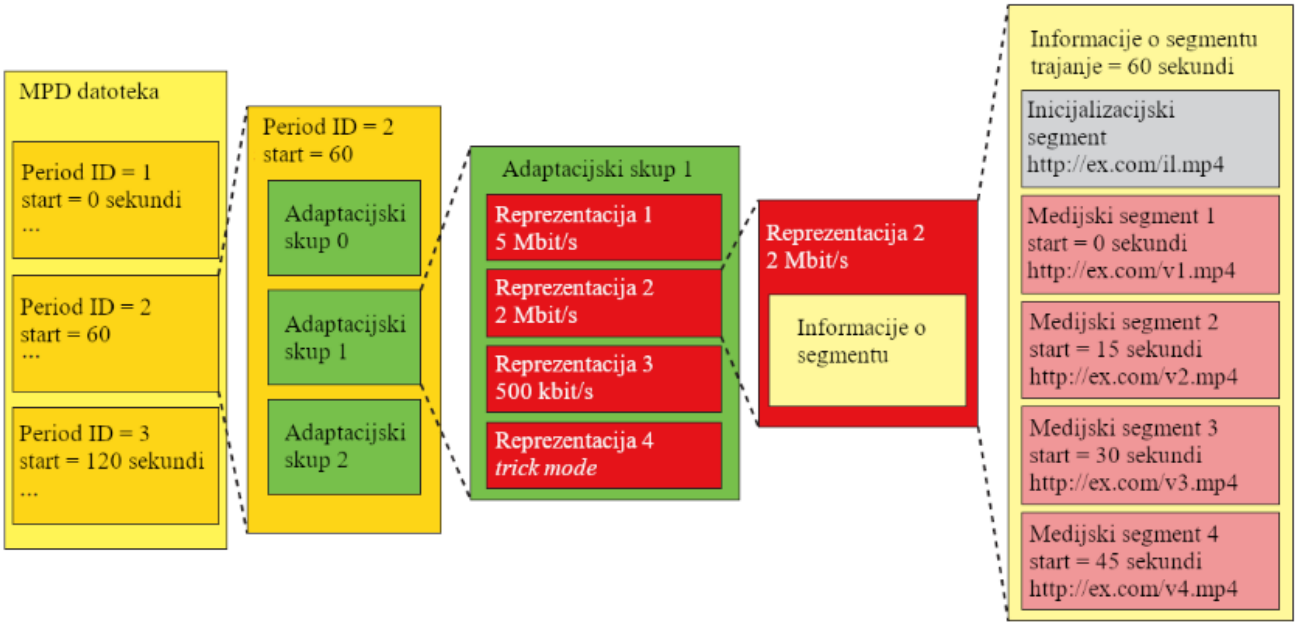
SegmentTemplate

Korištenjem *SegmentTemplate* elementa stvaraju se liste segmenata na temelju zadanog predloška. Identifikatori se tijekom izvršavanja zamjenjuju odgovarajućim vrijednostima. Prednost ovog načina je taj što se veličina MPD datoteke može značajno smanjiti u odnosu na korištenje statičkih listi.

```
<Representation mimeType="video/mp4"
                    frameRate="24"
                    bandwidth="1558322"
                    codecs="avc1.4d401f" width="1277" height="544">
  <SegmentTemplate media="http://cdn.bitmovin.net/bbb/video-1500/segment-$Number$.m4s"
                      initialization="http://cdn.bitmovin.net/bbb/video-1500/init.mp4"
                      startNumber="0"
                      timescale="24"
                      duration="48"/>
</Representation>
```

Slika 2.6. Uporaba elementa SegmentTemplate [4]

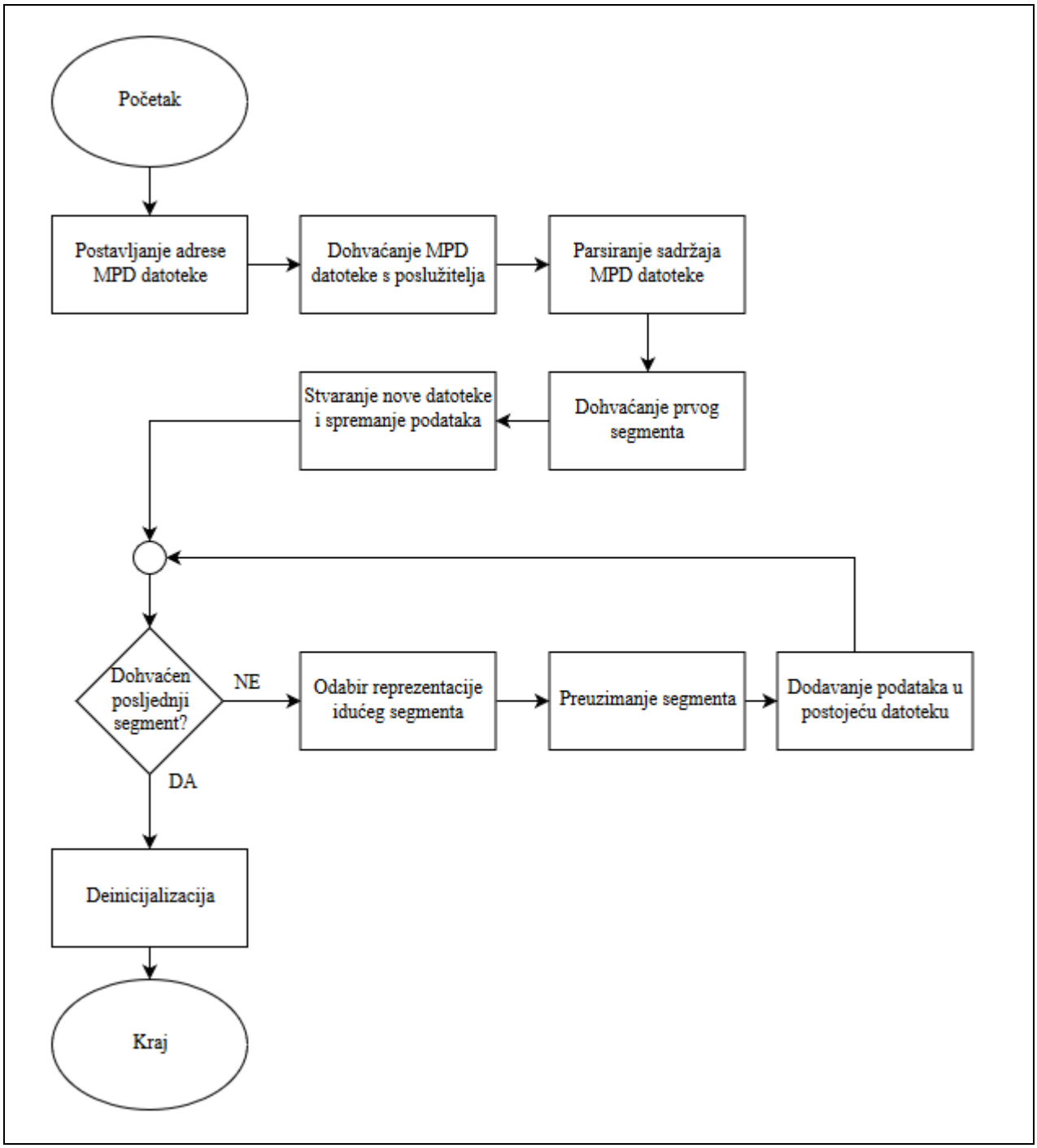
Slika 2.7. grafički prikazuje hijerarhijsku strukturu MPD datoteke s opisanim elementima. Datoteka na vrhu sadrži tri perioda. Period označen na slici sadrži tri adaptacijska skupa koji se dalje dijele na četiri reprezentacije, a svaka sadrži svoje podatke o medijskim segmentima.



Slika 2.7. Struktura MPD datoteke [3]

2.3. DASH klijent

Za implementaciju adaptacijskih algoritama korišten je klijentski program zatvorenog koda napisan programskim jezicima C i C++ koji se pokreće na operacijskim sustavima temeljenim na Linuxu. Pri pokretanju program vrši skupove operacija redoslijedom prikazanim na slici 2.8.



Slika 2.8. Dijagram toka korištenog MPEG-DASH klijenta

Korišteni MPEG-DASH klijent ne sadrži funkcionalnost reprodukcije preuzetog sadržaja. Rezultat izvršavanja programa je multimedijska datoteka koja se po završetku može reproducirati u nekom od proizvoljnih alata koji podržavaju njen format. Budući da se reprodukcija ne izvršava paralelno uz preuzimanje novog sadržaja, korisnik nema mogućnost obavljanja radnji koje utječu na odabir nadolazećih segmenata. Primjer jedne takve radnje je ručno zaustavljanje reprodukcije i odabir *trick mode* načina rada. U osnovnoj inačici klijenta odabir segmenata vrši se automatski prema algoritmu opisanom pseudokodom (slika 2.9.). Jedini ulazni parametar u tom slučaju je trenutna izmjerena propusnost mreže.

```
početak
ulaz odabir
ulaz brzina
ulaz niz kvaliteta
iterator := 0
ograničenje := 0
za iterator od 0 do kvaliteta.maksimum činiti
     ako kvaliteta.od(iterator) >= brzina * 8 onda
     {
          ograničenje := 1
          prekid
     }
ako ograničenje == 0 onda {
     odabir := iterator
     kraj
}
ako iterator >= 1 onda {
     ako iterator - 1 > odabir onda
          odabir := odabir + 1
     inače
          odabir := iterator - 1
}
inače
     odabir := 0
kraj
```

Slika 2.9. Pseudokod ugrađenog adaptacijskog algoritma

Ovaj algoritam izrađen je na temelju LIU adaptacijskog algoritma [6]. Pri pozivu funkcije potrebno je postaviti sljedeće parametre: trenutnu propusnost mreže, odabir kvalitete i niz dostupnih razina kvalitete u trenutno korištenoj reprezentaciji. Prvi korak algoritma je odrediti gdje se brzina

preuzimanja nalazi u odnosu na dostupne razine. Ta provjera se vrši uporabom petlje u kojoj iterator prolazi od minimalne razine kvalitete, označene nulom, do maksimalne podržane razine. Ako se u nekoj iteraciji ispostavi da trenutna razina kvalitete zahtjeva brzinu koja je veća od izmjerene, tada se zastavica ograničenja postavlja u vrijednost jedan a petlja se prekida. Ukoliko se pojavi slučaj u kojem je izmjerena brzina veća od brzine potrebne za prijenos sadržaja najveće kvalitete, ta razina se odabire i algoritam završava s radom. U svim preostalim slučajevima vrši se dodatna provjera. Kako bi se odredila nova vrijednost varijable *odabir*, uspoređuju se iterator i postojeća vrijednost te varijable. Za slučaj u kojem iterator poprima vrijednost koja je najmanje za dvije razine veća od prethodne tada se nova odabrana razina kvalitete povećava za jedan. Kod preostalih slučajeva razina kvalitete postavlja se na jednu razinu niže od vrijednosti iteratora da bi se spriječilo zastajkivanje u prijenosu podataka.

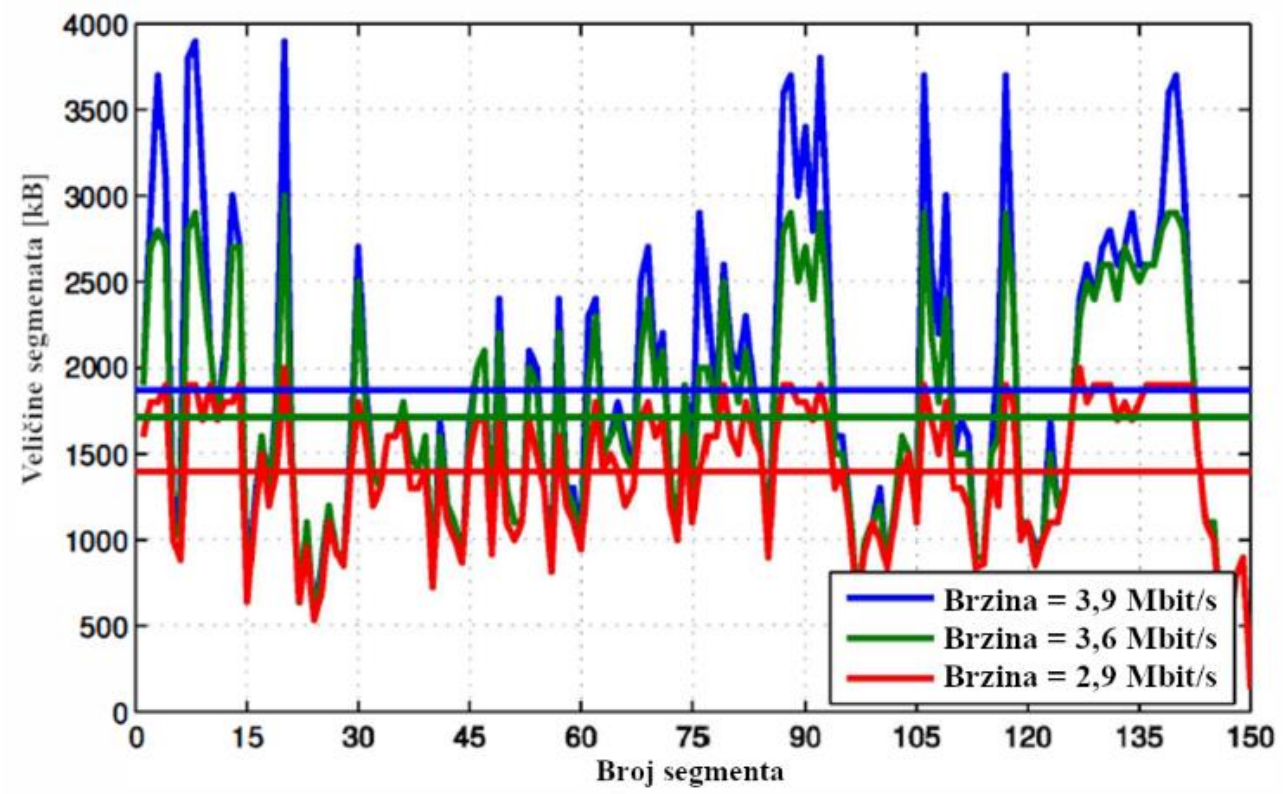
Proširenje programskih modula korištenog klijenta nužno je za ispravnu implementaciju algoritama opisanih u ovom radu. Programski modul za analizu sadržaja MPD datoteke prepoznaje sve oznake i atribute opisane MPEG-DASH standardom. Međutim, kako podatak o veličini pojedinačnog segmenta nije obuhvaćen tim standardom, napisan je programski kod koji će ga prepoznati i smjestiti u za to predviđenu strukturu podataka. Modul za upravljanje segmentima najviše je izmijenjen u odnosu na osnovnu inačicu. U tom dijelu smještena je logika odabira sljedećeg segmenta. Osnovni algoritam je uklonjen te se na njegovom mjestu nalaze novi algoritmi. Svi implementirani adaptacijski algoritmi prilikom odluke koriste podatak o popunjenosti međuspremnika. Stoga je potrebno na neki način simulirati reprodukciju sadržaja tijekom preuzimanja. Nakon preuzimanja početnog segmenta pokreće se nova nit koja se izvršava paralelno uz ostatak programa. Zadatak te niti je periodično mjerenje vremena proteklog od njenog pokretanja. Trenutno stanje međuspremnika se tada može izračunati izrazom $B_n = n \cdot d - t$ gdje je: B_n – stanje međuspremnika nakon n preuzetih segmenata, n – broj preuzetih segmenata, d – trajanje jednog segmenta i t – vrijeme proteklo od pokretanja niti.

3. ADAPTACIJSKI ALGORITMI

3.1. SARA algoritam (eng. *Segment Aware Rate Adaptation)*

3.1.1. Opis algoritma

Jednostavniji adaptacijski algoritmi pri odabiru segmenata često koriste parametre poput prosječne brzine preuzimanja segmenata ili trenutne popunjenosti međuspremnika. Obično klijent započinje preuzimanje odabirom segmenta najniže kvalitete, a odabir nadolazećih segmenata temelji na parametrima koje prikuplja tijekom preuzimanja. Segmenti imaju jednako vremensko trajanje ali se svejedno njihove veličine mogu značajno razlikovati ovisno o sadržaju koji prenose. Slika 3.1. prikazuje odstupanja u veličinama segmenata na primjeru videa Big Buck Bunny. Video je podijeljen na segmente u trajanju od četiri sekunde te su odabrane tri različite razine kvalitete koje iznose 2,9 Mbit/s, 3,6 Mbit/s i 3,9 Mbit/s. Za najveću odabranu razinu kvalitete može se uočiti kako se veličina segmenata kreće u intervalu od 538 kB pa sve do 3,9 MB dok srednja veličina iznosi 1,8 MB. Slično se ponašaju i vrijednosti preostalih razina kvalitete.



Slika 3.1. Veličine pojedinih segmenata pri različitim razinama kvalitete [7]

Adaptacijski algoritam SARA (eng. *Segment Aware Rate Adaptation*) osmišljen je kako bi se poboljšalo predviđanje vremena potrebnog za preuzimanje idućeg segmenta. SARA algoritam pri odabiru optimalne reprezentacije uzima u obzir tri parametra: prosječnu brzinu prijenosa, trenutnu popunjenost međuspremnika i veličinu nadolazećih segmenata. U MPEG-DASH standardu MPD datoteka sadrži informacije poput liste reprezentacija, trajanja segmenata i njihovih lokacija zadanih URL-om. Kako je pokazano da veličina segmenata unutar jedne reprezentacije može značajno varirati, predlaže se proširenje MPD datoteke podacima o veličini segmenata. Dodavanje tog podatka vrši se preinakom datoteke na poslužitelju (slika 3.2.). Takvo rješenje pruža dodatne mogućnosti klijentima koji ga podržavaju dok preostali klijenti mogu zanemariti te podatke bez ikakvih poteškoća.

```
<AdaptationSet mimeType="video/mp4" segmentAlignment="true" group="1" maxWidth="480" maxHeight="360" r
  <SegmentTemplate timescale="96" media="bunny_$Bandwidth$bps/BigBuckBunny_4s$Number$.m4s" startNumbe
 -<Representation id="320x240 45.0kbps" mimeType="video/mp4" codecs="avc1.42c00d" width="320" height="240"
    <SegmentSize id="BigBuckBunny_4s1.m4s" size="168.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s2.m4s" size="184.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s3.m4s" size="200.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s4.m4s" size="168.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s5.m4s" size="176.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s6.m4s" size="168.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s7.m4s" size="176.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s8.m4s" size="176.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s9.m4s" size="176.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s10.m4s" size="192.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s11.m4s" size="168.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s12.m4s" size="192.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s13.m4s" size="192.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s14.m4s" size="176.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s15.m4s" size="176.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s16.m4s" size="184.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s17.m4s" size="176.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s18.m4s" size="168.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s19.m4s" size="160.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s20.m4s" size="184.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s21.m4s" size="192.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s22.m4s" size="168.0" scale="Kbits"/>
    <SegmentSize id="BigBuckBunny_4s23.m4s" size="160.0" scale="Kbits"/>
```

Slika 3.2. Umetanje podataka o veličini segmenata u MPD datoteku

Veza između klijenta i poslužitelja može biti nestabilna a brzina prijenosa varirati. Kako bi se buduća brzina prijenosa mogla bolje procijeniti vrši se mjerenje protoka tijekom cijelog trajanja preuzimanja multimedijskog sadržaja. Korištenjem harmonijske sredine prilikom izračuna brzine

ublažava se utjecaj kratkotrajnih velikih odstupanja u izmjerenoj brzini. Sukladno tome, težinski faktori proporcionalni veličinama segmenata uzimaju se u obzir pri izračunu. Izraz po kojemu se dobiva konačna vrijednost harmonijske sredine iskazan je relacijom (3-1).

$$H_n = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{d_i}} \tag{3-1}$$

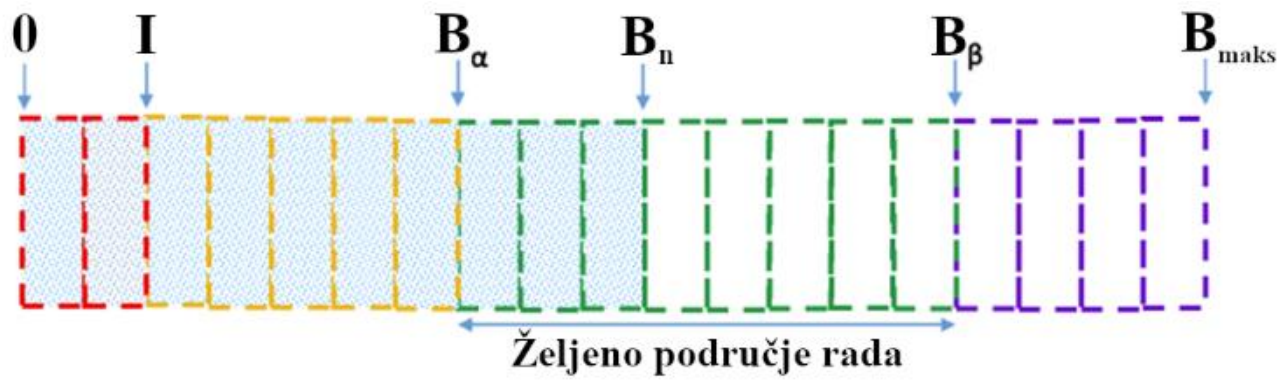
gdje je:

$\boldsymbol{H_n}$ – harmonijska sredina prvih *n* preuzetih segmenata

$\boldsymbol{w_i}$ – veličina *i*-tog preuzetog segmenta

$\boldsymbol{d_i}$ – brzina tijekom preuzimanja *i*-tog segmenta

Podatak o popunjenosti međuspremnika vrlo je bitan pri odabiru odgovarajućih segmenata. Iz tog razloga međuspremnik je podijeljen na četiri dijela. Ti dijelovi su omeđeni vrijednostima kako je prikazano na slici 3.2. i za njih vrijedi relacija: $0 \leq I \leq B_\alpha \leq B_\beta \leq B_{maks}$.



Slika 3.3. Podjela međuspremnika na područja rada [7]

S obzirom na podjelu međuspremnika SARA algoritam razlikuje četiri načina rada [7]:

- **Faza brzog početka**

  Događa se kada je trenutna popunjenost međuspremnika manja ili jednaka razini *I*. Kako bi reprodukcija započela što ranije u ovoj fazi se odabire najniža kvaliteta sadržaja.

- **Faza postepenog povećanja kvalitete**

  Algoritam prelazi u ovu fazu rada kada razina međuspremnika prijeđe vrijednost *I*, ali i dalje ima vrijednost nižu od B_α. Ovdje algoritam postepeno povećava odabranu razinu kvalitete kako bi se spriječio pad razine međuspremnika ispod razine *I*, a time i povratak na fazu brzog početka.

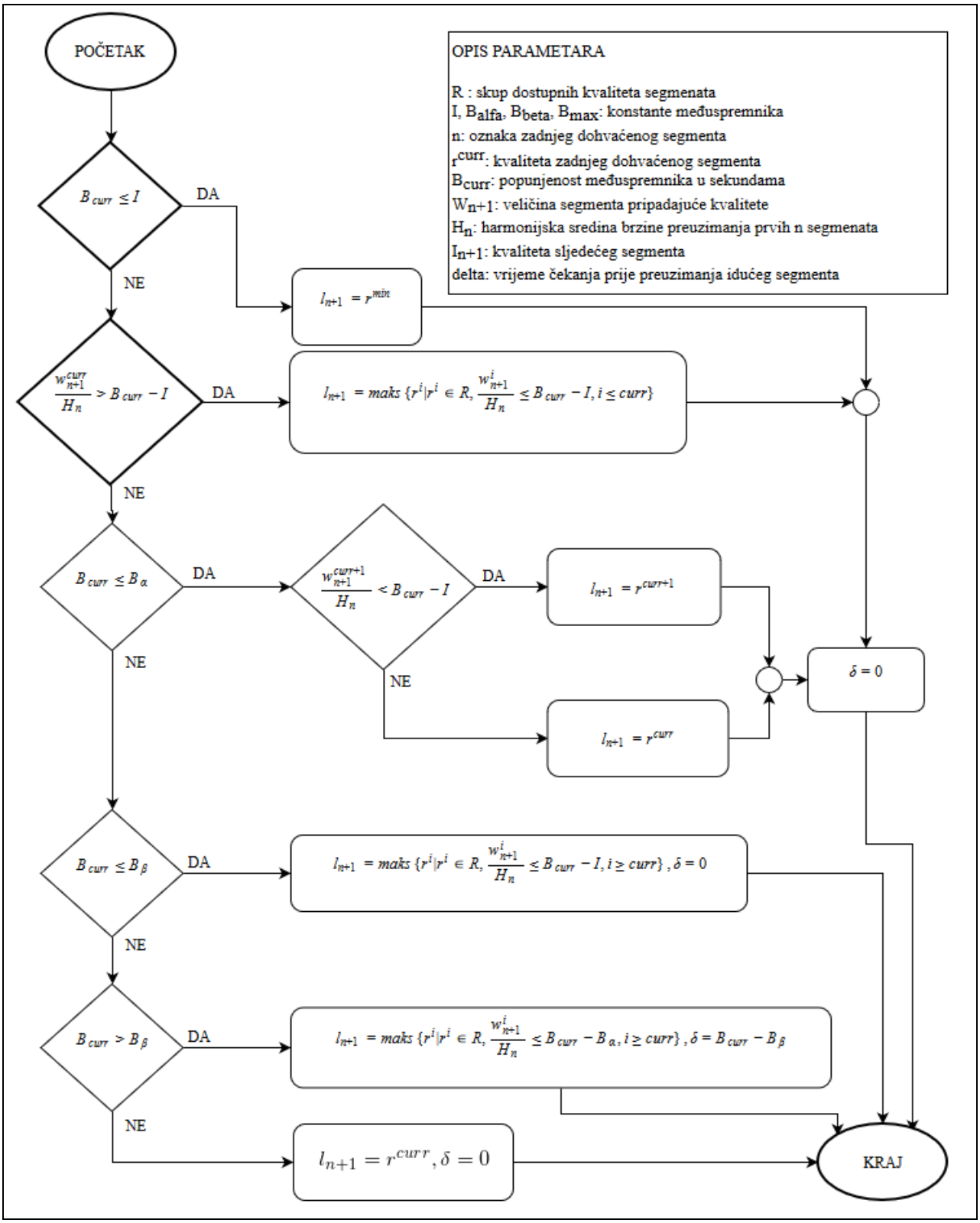
- **Faza agresivnog prebacivanja**

  Pri ovoj fazi međuspremnik se nalazi između razina B_α i B_β. Poželjno je zadržati međuspremnik u tom rasponu. Ovdje se bira maksimalna razina kvalitete koju mreža može podržati uz uvjet da se kvaliteta ne smanji.

- **Zaustavljanje preuzimanja**

  Ukoliko razina međuspremnika prijeđe razinu B_β odabire se odgovarajući segment ali se zahtjev za preuzimanjem odgađa sve dok se međuspremnik ne isprazni do razine B_β. Tako se sprječava nepotrebno nakupljanje segmenata u međuspremniku i preuzimanje sadržaja ako korisnik ranije odluči napustiti sadržaj.

3.1.2. Dijagram toka



Slika 3.4. Dijagram toka SARA algoritma

3.1.3. Pseudokod

```
početak
ulaz odabir
ulaz H
ulaz međuspremnik
ulaz niz kvaliteta
ulaz niz segment
I := konstanta
Balfa := konstanta
Bbeta := konstanta
ako međuspremnik <= I onda
      odabir := 0
inače
      ako segment.idući(odabir) / H > međuspremnik - I onda
            dok je odabir > 0 činiti
                  ako segment.idući(odabir) / H > međuspremnik - I onda
                        odabir := odabir - 1
                  inače
                        prekid
      inače ako međuspremnik <= Balfa onda
            ako segment.idući(odabir + 1) / H < međuspremnik -I
                  odabir := odabir + 1
      inače ako međuspremnik <= Bbeta onda
            dok je odabir <= kvaliteta.veličina - 1 činiti
                  ako segment.idući(odabir) / H <= međuspremnik - I onda
                        ako odabir == kvaliteta.veličina - 1 onda
                              prekid
                        odabir := odabir + 1
                  inače {
                        odabir := odabir - 1
                        prekid
                  }
      inače
            dok je odabir <= kvaliteta.veličina -1 činiti
                  ako segment.idući(odabir) / H <= međuspremnik – Balfa
                        ako odabir == kvaliteta.veličina - 1 onda
                              prekid
                        odabir := odabir + 1
                  inače {
                        odabir := odabir - 1
                        prekid
                  }
            čekati(međuspremnik - Bbeta)
kraj
```

Slika 3.5. Pseudokod SARA algoritma

Za ispravan rad algoritma potrebno je najprije postaviti odgovarajuće parametre. Varijabla *odabir* početno je postavljena na nulu što označava najnižu kvalitetu prijenosa, a njenu vrijednost je između poziva funkcija potrebno sačuvati. Varijabla *H* prema (3-1) predstavlja harmonijsku vrijednost prethodno preuzetih segmenata. Međuspremnik je prema predloženom modelu (slika 3.3.) opisan s četiri parametra. Varijabla *međuspremnik* sadrži njegovu popunjenost izraženu u sekundama. Vrijednost ove varijable potrebno je računati na početku svakog poziva funkcije. Konstante *I,* B_{alfa} i B_{beta} postavljaju se na samom početku i njihove optimalne vrijednosti se mogu odrediti ovisno o sadržaju koji se očekuje. Podaci o dostupnim razinama kvalitetate kao i podaci o veličinama segmenata spremaju se u nizove prilikom analize MPD datoteke, prije početka preuzimanja. Unutar jednog perioda video sadržaja potrebno je očuvati indeks posljednjeg preuzetog segmenta da bi se iz nizova mogli dohvatiti potrebni podaci. Ostatak pseudokoda podijeljen je u četiri načina rada koji su pobliže opisani u prvom dijelu ovog potpoglavlja.

3.2. CARA algoritam (eng. *Content Aware Rate Adaptation*)

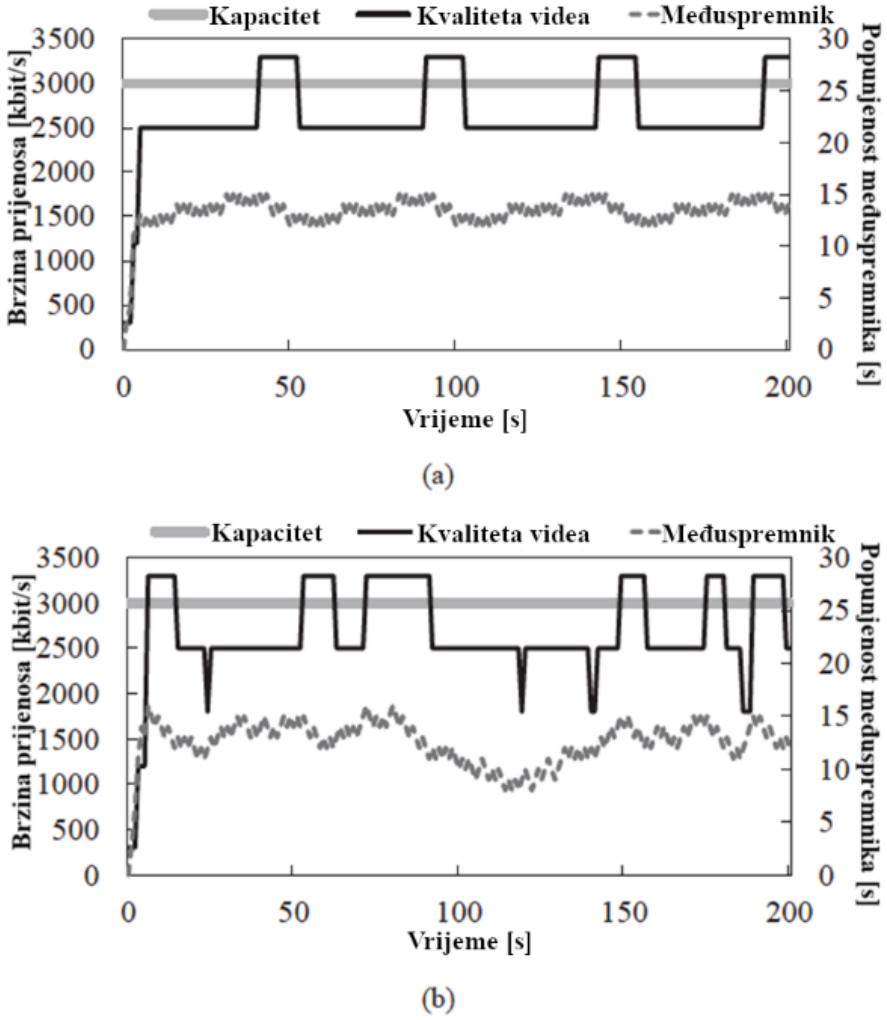
3.2.1. Opis algoritma

Postojeći adaptacijski algoritmi mogu se podijeliti u dvije glavne kategorije. Prva od njih temelji se na mjerenju propusnosti mreže. Izmjene među reprezentacijama ovdje se vrše temeljem podataka o procijenjenoj propusnosti mreže. Većina adaptacijskih algoritama korištenih u komercijalnim rješenjima spadaju u tu kategoriju [8]. Ovaj pristup može uzrokovati smanjenje kvalitete korisničkog iskustva zato što kratke oscilacije u brzini prijenosa dovode do izbora neprikladne reprezentacije.

Druga grupa adaptacijskih algoritama koristi međuspremnik za odluku o preuzimanju segmenata. Cilj ovakvih algoritama je spriječiti pražnjenje međuspremnika ispod željene razine, ali i njegovo prepunjavanje. Neki pružatelji usluga koriste ovaj pristup, a kvalitetu video sadržaja postavljaju na poslužiteljskoj strani [9]. To dodatno opterećuje poslužitelje što ograničava kvalitetu pružanja usluge. Iako se sprječavanjem pražnjenja međuspremnika osigurava neprekidan prijenos sadržaja, zbog čestih promjena odabira kvalitete korisničko iskustvo će biti lošije.

Video preuzet korištenjem MPEG-DASH standarda ne sadrži u svakom segmentu isti broj bitova nego se oni razlikuju ovisno o odabranoj reprezentaciji. Promjenjiva brzina prijenosa utječe na

učinkovitost klijenata koji upotrebljavaju karakteristike međuspremnika pri odabiru segmenata. Ako se usporedi isti video sadržaj kodiran promjenjivom i stalnom brzinom bitova (eng. *bit rate*) može se zaključiti da promjenjiva brzina dovodi do češćih promjena u razini kvalitete nego što je slučaj kod stalne brzine bitova (slika 3.6.). To je uzrokovano manjom mogućnosti procjene vremena za preuzimanje segmenta video sadržaja promjenjive brzine. Za rješavanje tog problema predložen je algoritam koji korištenjem informacije o veličini preuzetih segmenata prilagođava odabir ovisno o stanju mreže.



Slika 3.6. Usporedba kvalitete video pri stalnoj (a) i promjenjivoj (b) brzini bitova [8].

CARA algoritam (eng. *Content Aware Rate Adaptation*) se odvija u tri koraka: procjena brzine, određivanje očekivane popunjenosti međuspremnika i prilagodba kvalitete. Za procjenu

propusnosti mreže koriste se veličine preuzetih segmenata. Nakon završetka preuzimanja segmenta algoritam procjenjuje brzinu prijenosa koristeći relaciju (3-2):

$$x'[n] = \frac{(B_k[n] - B_k[n-1] + 1) \cdot 8}{d[n]} \tag{3-2}$$

gdje je:

$\boldsymbol{B_k[n]}$ – posljednji oktet *n*-tog segmenta pri zadanoj kvaliteti *k*

$\boldsymbol{d[n]}$ – vrijeme potrebno za preuzimanje *n*-tog segmenta

Međuspremnik se modelira korištenjem informacije o trenutnoj i prošloj popunjenosti što je prikazano izrazom (3-3):

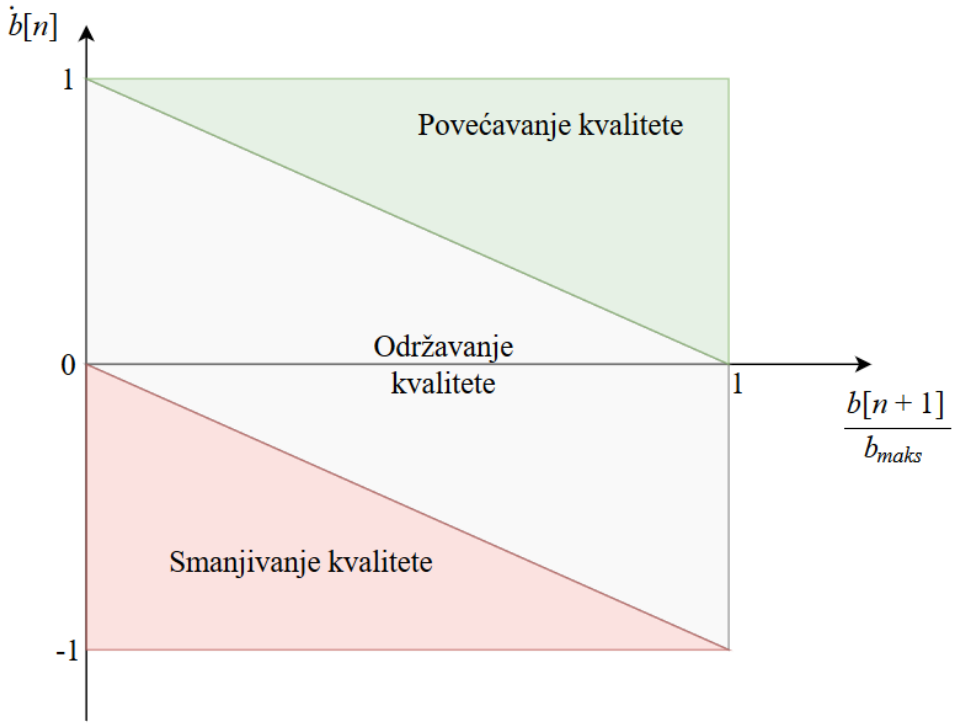
$$\dot{b}[n] \triangleq \frac{b[n] - b[n-1]}{\tau} \tag{3-3}$$

gdje je:

$\boldsymbol{b[n]}$ – razina popunjenosti međuspremnika nakon preuzimanja segmenta *n*

$\boldsymbol{\tau}$ - trajanje jednog segmenta

Kako bi se spriječilo prepunjavanje i pražnjenje međuspremnika, međuspremnik se dijeli na tri dijela (slika 3.7.). Ovisno o tome algoritam prilagođava stupanj sklonosti promjene kvalitete [8].



Slika 3.7. Prikaz ovisnosti popunjenosti međuspremnika o načinu rada[8]

U koraku prilagodbe kvalitete algoritam odabire segment na sljedeći način (3-4):

$$r[n+1] = \begin{cases} argmin\ R_k, R_k > x'[n]\ za\ \dot{b} \geq -\beta + 1 \\ argmaks\ R_{k,} R_k < x'[n]\ za\ \dot{b} < -\beta \\ r[n], inače \end{cases} \quad (3\text{-}4)$$

gdje je:

$\boldsymbol{\beta}$ – omjer vrijednosti $b[n+1]$ i b_{maks}

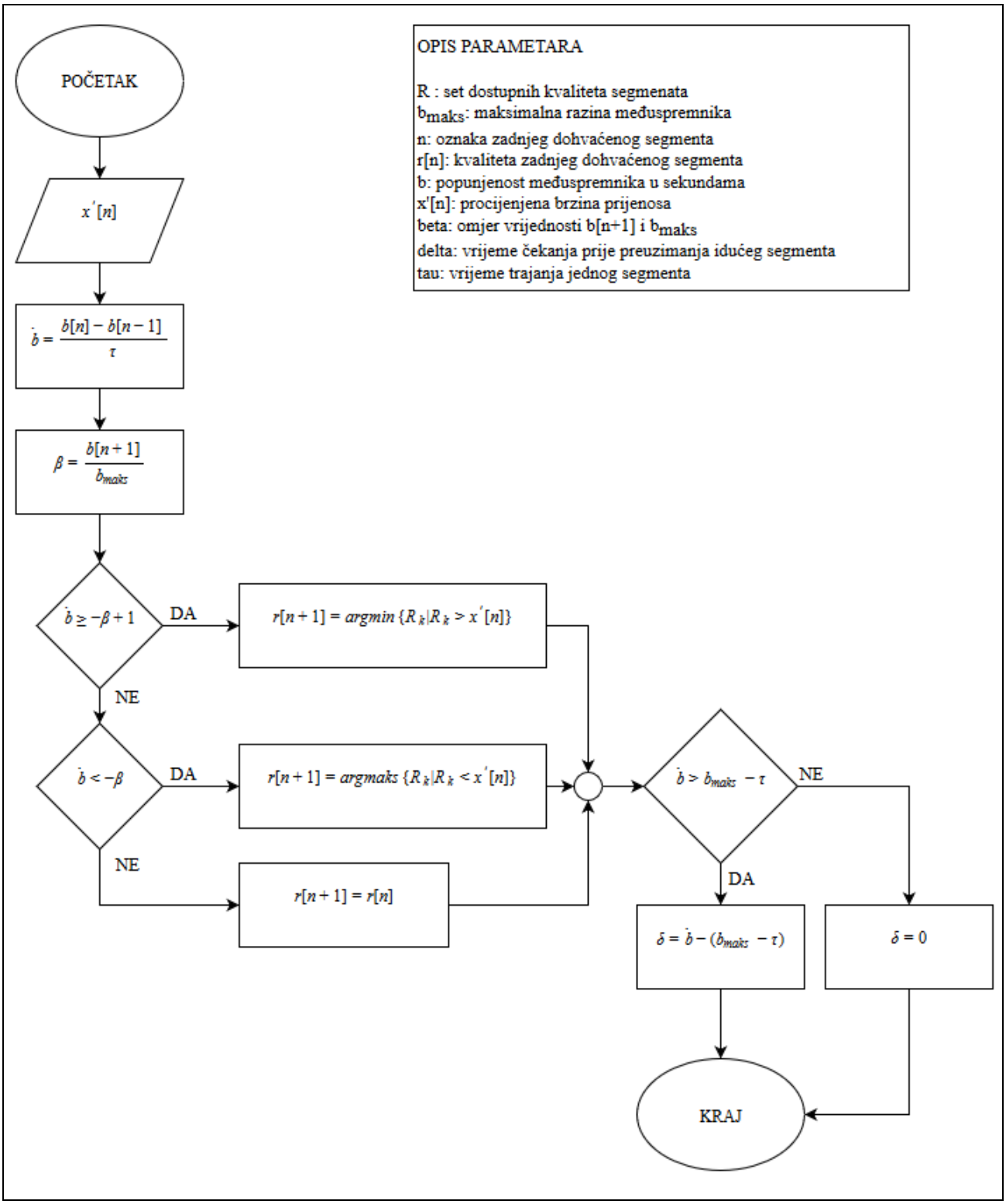
R – skup dostupnih razina kvalitete

$\boldsymbol{x'[n]}$ –procijenjena brzina prijenosa nakon preuzimanja segmenta *n*

$\boldsymbol{\dot{b}}$ –popunjenost međuspremnika

$\boldsymbol{r[n]}$ – odabrana razina kvalitete *n*-tog segmenta

3.2.2. Dijagram toka



Slika 3.8. Dijagram toka CARA algoritma

3.2.3. Pseudokod

```
početak
ulaz odabir
ulaz brzina
ulaz niz kvaliteta
ulaz međuspremnik
statička varijabla međuspremnik_prošli := 0
Bmax := konstanta
T := konstanta
beta := (međuspremnik + T) / Bmax
b := međuspremnik - međuspremnik_prošli
međuspremnik_prošli := međuspremnik
ako b >= 1 - beta onda {
      odabir := kvaliteta.veličina - 1
      dok je odabir > 0 činiti {
            ako kvaliteta.od(odabir) > brzina * 8 onda
                  odabir := odabir - 1
            inače {
                  ako odabir < kvaliteta.veličina - 1 onda
                        odabir := odabir + 1
                  prekid
            }
      }
}
inače ako b < beta * (-1) onda {
      odabir := 0
      dok je odabir < kvaliteta.veličina činiti {
            ako kvaliteta.od(odabir) < brzina * 8 onda {
                  ako odabir == kvaliteta.veličina - 1 onda prekid
                  odabir := odabir + 1
            }
            Inače {
                  ako odabir > 0 onda
                        odabir := odabir - 1
                  prekid
            }
      }
}
inače
      odabir := odabir
ako međuspremnik > Bmax - T onda
      čekati(međuspremnik - Bmax + T)
kraj
```

Slika 3.9. Pseudokod CARA algoritma

3.3. SIA algoritam (eng. *Segment Importance Based Rate Adaptation*)

3.3.1. Opis algoritma

U prijenosu sadržaja medijski segmenti nemaju jednaku važnost za korisnike. Prilikom prijenosa nogometne utakmice segment koji prikazuje zgoditak ima veću važnost od segmenta za vrijeme kojeg nema posebnih događanja. Kvaliteta važnijih segmenata značajno utječe na kvalitetu korisničkog iskustva. Kako bi se poboljšala kvaliteta korisničkog iskustva predložen je adaptacijski algoritam naziva SIA (eng. *Segment Importance Based Rate Adaptation*). Taj algoritam nastoji odabrati veću razinu kvalitete kada su u pitanju segmenti veće važnosti uz što manji utjecaj na kvalitetu segmenata niže važnosti. SIA algoritam se sastoji od modela važnosti (eng. *SIM - Segment Importance Model*) i strategije zahtijevanja segmenata (eng. *SRS - Segment Request Strategy*) koji su opisani u nastavku.

SIM (eng. *Segment Importance Model*)

Podatak o važnosti segmenata potrebno je prije korištenja upisati u MPD datoteku na poslužitelju. Radi pojednostavljenja algoritma važnost segmenta može poprimiti vrijednosti nula ili jedan. Vrijednost nula označuje segment manje važnosti, a vrijednost jedan označuje segment veće važnosti za korisnika. Procjena budućeg stanja međuspremnika u SIM modelu računa se na temelju podataka o trenutnom stanju međuspremnika i brzini prijenosa. Veličine nadolazećih segmenata koriste se da bi procjena bila točnija što je prikazano izrazom (3-5).

$$B_{n+1}(r^i) = B_n + D - \frac{s_{n+1}(r^i)}{H_n} \tag{3-5}$$

gdje je:

$\boldsymbol{B_n}$ – popunjenost međuspremnika nakon preuzimanja *n*-tog segmenta

$\boldsymbol{r^i}$ – odabrana razina kvalitete

$\boldsymbol{D}$ – trajanje jednog segmenta

$\boldsymbol{s_n}$ – veličina *n*-tog segmenta

$\boldsymbol{H_n}$ – harmonijska sredina brzine preuzimanja za prvih *n* segmenata prema (3-1)

Za potrebe ovog rada algoritam uzima u obzir važnost dva iduća segmenta. Uzimanjem većeg broja nadolazećih segmenata u obzir značajno se povećava složenost algoritma, a zbog dužeg trajanja prijenosa teže je procijeniti stanje međuspremnika. Tijekom preuzimanja segmenata manje važnosti nastoji se osigurati veća popunjenost međuspremnika kako bi se za segmente veće važnosti mogla odabrati veća razina kvalitete. SIM razlikuje četiri načina rada ovisno o važnostima nadolazeća dva segmenta, a to su: 11, 10, 01 i 00.

- **Način rada 00**

  U ovom načinu rada glavni cilj je spriječiti pražnjenje međuspremnika ispod donje granice, I, te istovremeno povećati razinu kvalitete ako je to moguće. Povećanje kvalitete vrši se ovisno o razini međuspremnika. Zadana je vrijednost α koja dijeli međuspremnik na dva dijela uz uvjet $\alpha > I$. Ako razina međuspremnika ima vrijednost manju od α tada se odabir razine kvalitete mijenja postepeno. U suprotnom se odabire najveća podržana razina kvalitete.

- **Način rada 10**

  Prvi sljedeći segment ima veću važnost nego njegov sljedbenik. Za važniji segment odabire se najveća moguća kvaliteta pri kojoj neće doći do zastoja u reprodukciji. Kvaliteta segmenta manje važnosti u tom slučaju se ne uzima u obzir.

- **Način rada 01**

  Kako bi se za segment veće važnosti mogla odabrati veća razina kvalitete potrebno je smanjiti vrijeme preuzimanja segmenta manje važnosti. Zbog toga se odabir kvaliteta nadolazećih segmenata odvija u dva koraka. Najprije se određuje najveća podržana razina kvalitete važnijeg segmenta. U drugom koraku računa se maksimalna razina kvalitete segmenta manje važnosti koja neće uzrokovati smanjenje kvalitete važnijeg segmenta.

- **Način rada 11**

  Učestala promjena razine kvalitete negativno utječe na kvalitetu korisničkog iskustva. Pošto oba nadolazeća segmenta imaju jednaku važnost odabire se jedinstvena razina kvalitete koja neće uzrokovati pražnjenje međuspremnika ispod donje granice.

SRS (eng. *Segment Request Strategy*)

Ovim mehanizmom osigurava se optimalan izbor kvalitete sljedećeg segmenta. SRS je podijeljen na tri dijela ovisno o trenutnoj popunjenosti međuspremnika. Pri tome se nastoji održati razinu međuspremnika unutar zadanih granica.

1. Mehanizam brzog početka reprodukcije

Kada se popunjenost međuspremnika nalazi ispod početne razine algoritam odabire najmanju razinu kvalitete, r^{min}. Time se smanjuje vrijeme potrebno za početak reprodukcije i sprječava zastoj uslijed pražnjenja međuspremnika.

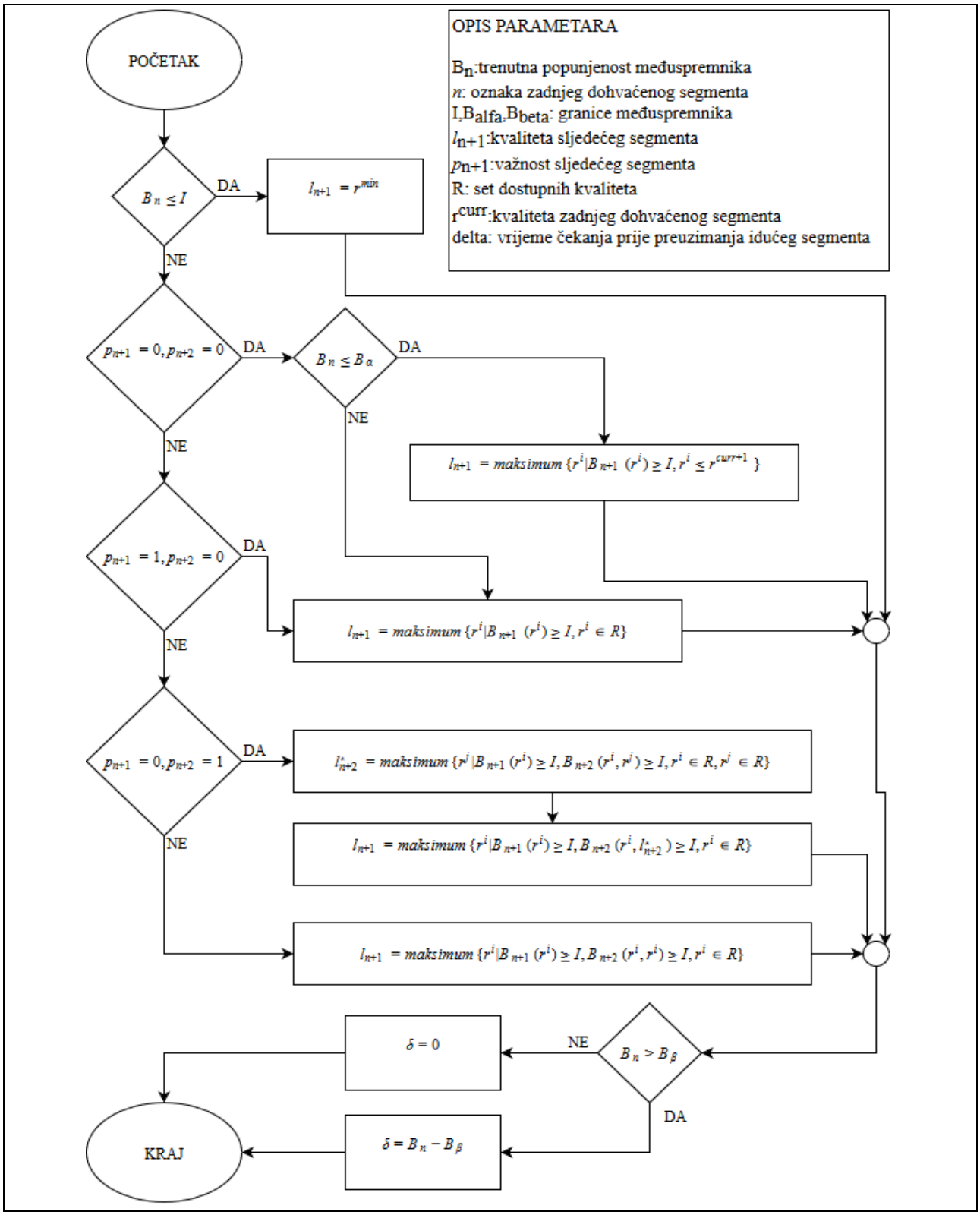
2. Privremeno zaustavljanje preuzimanja

Ako je popunjenost međuspremnika visoka, a brzina preuzimanja veća od brzine potrebne za odabranu razinu kvalitete doći će do prelijevanja. Zbog toga se definira razina β za koju vrijedi $\beta \geq \alpha$. Ukoliko popunjenost međuspremnika, B, prijeđe razinu β privremeno se zaustavlja preuzimanje segmenata. Vrijeme odgode δ računa se prema relaciji $\delta = B - \beta$.

3. Odabir kvalitete prema važnosti

Algoritam se nalazi u ovom načinu rada kada je popunjenost međuspremnika veća od početne razine. Tada se odabir sljedećeg segmenta vrši prema SIM modelu. Kvaliteta važnih segmenata u ovom slučaju ne smije biti manja od kvalitete prethodnog segmenta.

3.3.2. Dijagram toka



Slika 3.10. Dijagram toka SIA algoritma

3.3.3. Pseudokod

```
početak
ulaz niz kvaliteta
ulaz niz segment
ulaz niz važnost
ulaz odabir, H, međuspremnik, trajanje
varijabla granica, i, j, B1, B2
I := konstanta
Balfa := konstanta
Bbeta := konstanta
ako međuspremnik <= I onda
  i := 0
inače {
  ako važnost.od(trenutni + 1) == 0 I
       važnost.od(trenutni + 2) == 0 onda {
    ako međuspremnik < Balfa onda
      granica := odabir + 1
    inače
      granica := kvaliteta.veličina - 1
    i := 0
    dok je i < granica činiti {
      B1 := međuspremnik + trajanje - segment.idući(i) / H
      ako B1 >= I onda
        i := i + 1
      inače {
        i:= i - 1
        prekid
      }
    }
  }
  inače ako važnost.od(trenutni + 1) == 0 I
            važnost.od(trenutni + 2) == 1 onda {
    i := 0
    j := 0
    dok je j < kvaliteta.veličina - 1 činiti {
      B2 := međuspremnik + 2 * trajanje - (segment.od(trenutni + 1).(i)
                                    + segment.od(trenutni + 2).(j)) / H
      ako B2 >= I onda
        j := j + 1
      inače{
        j := j - 1
        prekid
      }
    }
```

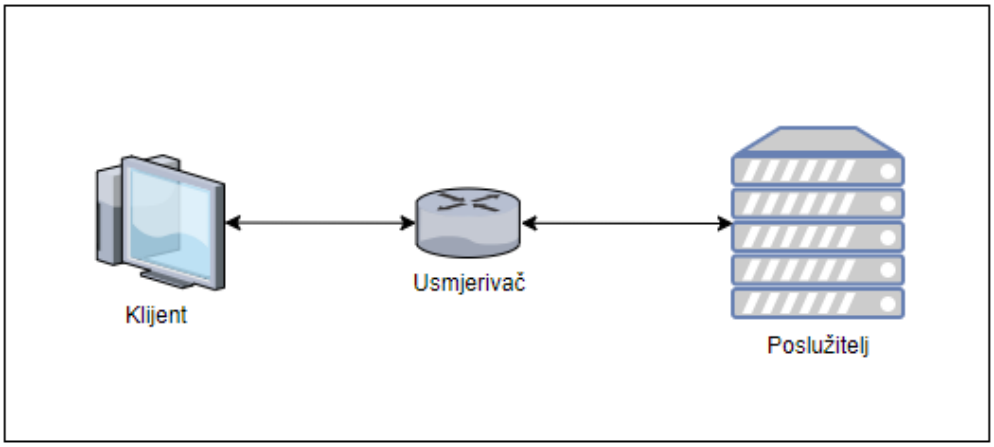
```
    dok je i < kvaliteta.veličina - 1 činiti {
      B1 := međuspremnik + trajanje - segment.idući(i) / H
      B2 := međuspremnik + 2 * trajanje - (segment.od(trenutni + 1).(i)
                                    + segment.od(trenutni + 2).(j)) / H
      ako B1 < I ILI B2 < I onda {
        i := i - 1
        prekid
      }
      inače
        i := i + 1
    }
  }
  inače ako važnost.od(trenutni + 1) == 1 I
            važnost.od(trenutni + 2) == 0 onda {
    i := 0
    dok je i < kvaliteta.veličina - 1 činiti {
      B1 := međuspremnik + trajanje - segment.idući(i) / H
      ako B1 >= I onda
        i := i + 1
      inače {
        i:= i - 1
        prekid
      }
    }
  }
  inače {
    i :=  0
    dok je i < kvaliteta.veličina - 1 činiti {
      B1 := međuspremnik + trajanje - segment.idući(i) / H
      B2 := međuspremnik + 2 * trajanje - (segment.od(trenutni + 1).(i)
                                    + segment.od(trenutni + 2).(i)) / H
      ako B1 < I ILI B2 < I onda {
        i := i - 1
        prekid
      }
      inače
        i := i + 1
    }
  }
}
odabir := i
ako međuspremnik > Bbeta onda
  čekati(međuspremnik - Bbeta)
kraj
```

Slika 3.11. Pseudokod SIA algoritma

4. TESTIRANJE ALGORITAMA

4.1. Metoda mjerenja učinkovitosti algoritama

Za potrebe mjerenja učinkovitosti korištena su dva računala spojena na lokalnu mrežu prema slici 4.1. Jedno računalo ima ulogu poslužitelja na kojemu su smješteni video segmenti i pripadajuće MPD datoteke. Poslužiteljsko računalo koristi operacijski sustav Linux Ubuntu inačice 16.04.5 i na njemu je pokrenut poslužiteljski program Apache HTTP Server inačice 2.4.18. Klijentsko računalo sadržava operacijski sustav Linux Mint, inačice 18.2., i MPEG-DASH klijentski program u kojemu su implementirani adaptacijski algoritmi.



Slika 4.1. Mrežna struktura korištena pri testiranju algoritama

Ograničenje brzine se vrši na klijentskom računalu uporabom alata Wondershaper inačice 1.4. Wondershaper je skripta koja pojednostavljuje postavljanje ograničenja propusnosti mrežnih sučelja, a za tu svrhu koristi program traffic control (tc), dio paketa iproute napisanih za operacijske sustave temeljene na Linuxu [11]. Video sadržaj korišten za testiranje sastoji se od tri sekvence: Big Buck Bunny, Elephants Dream i Of Forest and Men [12]. Njihove značajke prikazane su tablicom 4.1.

Tablica 4.1. Značajke korištenih video sekvenci

Naziv	Big Buck Bunny	Elephants Dream	Of Forest and Men
Tip sadržaja	Animirani film	Animirani film	Dokumentarni film
Trajanje segmenta [s]	6	6	6
Broj segmenata	100	109	76
Broj sličica u sekundi	24	24	25
Broj perioda	1	1	1
Broj reprezentacija	20	20	19
Minimalna razlučivost	320x240	320x240	320x240
Maksimalna razlučivost	1920x1080	1920x1080	1024x576
Minimalna brzina prijenosa [bit/s]	45541	45929	46881
Maksimalna brzina prijenosa [bit/s]	3858484	4019433	3732333
Video format	H.264	H.264	H.264

Prilikom preuzimanja korišteni su sljedeći parametri međuspremnika: $I = 8\ s$, $B_\alpha = 16\ s$, $B_\beta = 32\ s$ i $B_{maks} = 40\ s$. Za testiranje SIA algoritma 6% od ukupnog broja segmenata predstavlja segmente veće važnosti u odnosu na preostale. Kod osnovnog algoritma nije potrebno postaviti dodatne parametre pošto ne uzima u obzir ni razinu međuspremnika niti važnost sadržaja pojedinih segmenata. Radi usporedbe rezultata zabilježene su vrijednosti sljedećih parametara: trenutna brzina prijenosa, odabrana razina kvalitete i trenutna popunjenost međuspremnika izražena u sekundama. Testiranje svakog od tri videa provedeno je u pet testnih slučajeva. Interval između svake promjene brzine iznosi deset sekundi, a testni slučajevi periodički se ponavljaju tijekom cijelog trajanja preuzimanja. Testni slučajevi prikazani su tablicom 4.2.

Tablica 4.2. Promjena brzine preuzimanja pri testnim slučajevima

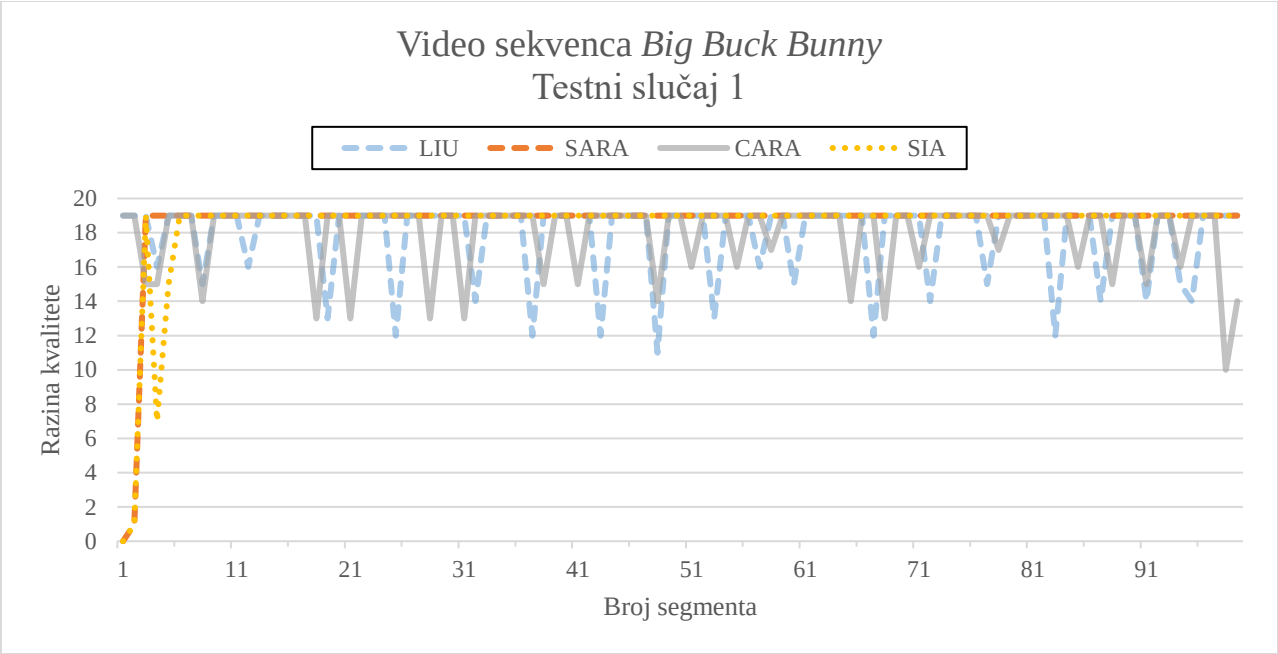
	1. slučaj	2. slučaj	3. slučaj	4. slučaj	5. slučaj
Brzina [Mbit/s]	10	1	1	1	1
	1	2	3	3	3
		3	5	5	1
		2	7	7	3
			5		5
			3		8
					8
					8
					8

4.2. Rezultati mjerenja

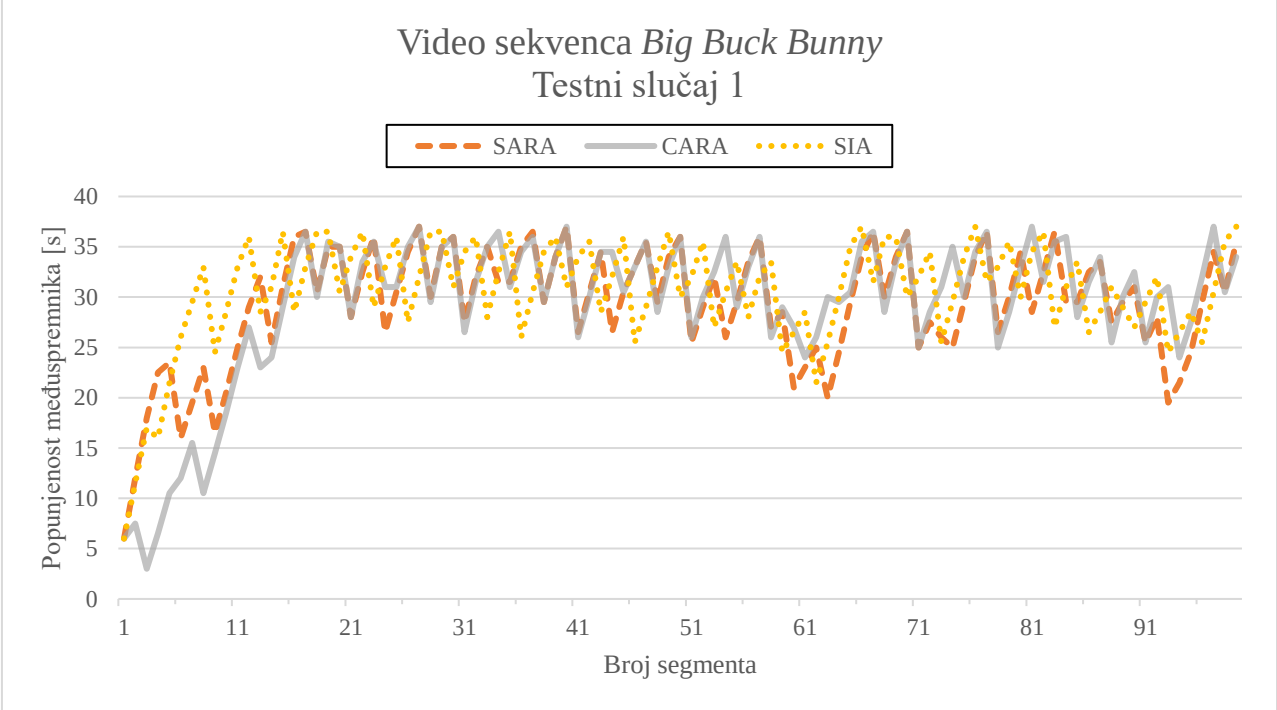
U ovom dijelu su prikazani grafovi dobiveni preuzimanjem videa za tri implementirana algoritma kao i za osnovni (LIU) algoritam. Za sve algoritme izrađen je graf koji prikazuje odabranu razinu kvalitete pojedinog segmenta po testnim slučajevima. Razina kvalitete videa poprima diskretne vrijednosti definirane brojem reprezentacija pripadajuće MPD datoteke. Kvaliteta sadržaja medijskog segmenta uvjetovana je prostornom razlučivosti pojedine reprezentacije i stupnjem sažimanja pri kodiranju. Unutar MPD datoteke nalaze se vrijednosti minimalne potrebne propusnosti mreže za prijenos segmenata određene reprezentacije. Dodatno su za implementirane algoritme izrađeni grafovi koji prikazuju popunjenost međuspremnika tijekom preuzimanja izraženu u sekundama. Budući da osnovni algoritam ne koristi međuspremnik za njega nije moguće prikazati tu ovisnost. Mjerenja su provedena za sve tri video sekvence, a u nastavku su prikazani neki od dobivenih rezultata.

Kod prvog testnog slučaja propusnost mreže poprima vrijednosti od približno 10 Mbit/s i 1 Mbit/s. Na samom početku videa može se uočiti razlika između algoritama (slika 4.2). Ovdje SARA i SIA algoritmi najprije odabiru segmente najniže kvalitete, dok algoritmi LIU i CARA odabiru segmente najveće kvalitete. Razlog tomu leži u primjeni metode brzog početka (eng. *fast start*). Kako bi se vrijeme čekanja do početka reprodukcije svelo na minimum, algoritmi SARA i SIA odabiru segment najniže kvalitete jer je njegova veličina manja, a time i vrijeme preuzimanja kraće. Kada popunjenost međuspremnika prijeđe donju granicu ta dva algoritma prelaze u stanje aditivnog povećanja kvalitete. U tom načinu rada postepeno se povećava odabir kvalitete segmenta s ciljem povećanja razine međuspremnika iznad granice B_α. Prelaskom te granice popunjenost međuspremnika se nalazi u optimalnom rasponu i odabire se najveća razina kvalitete koja neće dovesti do pražnjenja međuspremnika ispod donje granice. Za razliku od spomenutih algoritama, LIU i CARA algoritmi na samom početku odabiru segment najveće kvalitete jer je u tom trenutku brzina preuzimanja veća od brzine potrebne za preuzimanje segmenta. Međutim, posljedica toga vidljiva je već na trećem segmentu kod CARA algoritma odnosno četvrtom segmentu LIU algoritma. Na tom mjestu dolazi do prvog pada u razini kvalitete uzrokovane nedovoljnom popunjenosti međuspremnika popraćenu skokovitom promjenom brzine mreže na manju vrijednost. SARA i SIA algoritmi uspješno održavaju najveću razinu kvalitete tijekom cijelog trajanja preuzimanja. Razlog tomu je to što se za vrijeme visoke brzine prijenosa međuspremnik napuni kako bi tijekom razdoblja niže brzine prijenosa mogao koristiti nakupljenu rezervu za održavanje postavljene kvalitete videa. LIU i CARA algoritmi

su osjetljiviji na promjene u brzini prijenosa što rezultira čestim promjenama u razini kvalitete. Većinu vremena međuspremnik ima visoku razinu popunjenosti, a mehanizmom privremenog zaustavljanja preuzimanja sprječava se prelazak preko gornje granice od 40 sekundi (slika 4.3.).



Slika 4.2. Odabir razine kvalitete pojedinačnih segmenata kod prvog testnog slučaja



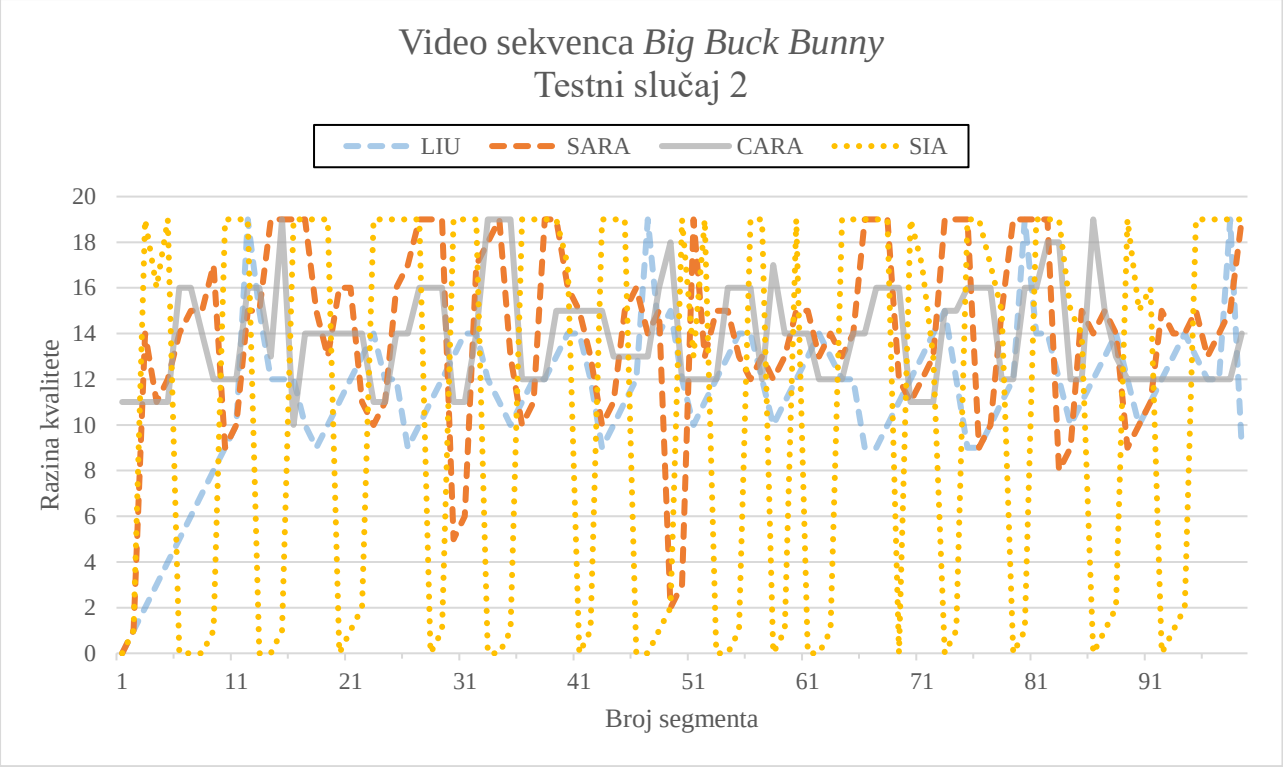
Slika 4.3. Popunjenost međuspremnika tijekom preuzimanja kod prvog testnog slučaja

Tablicom 4.3. prikazana je prosječna razina kvalitete za pojedine algoritme te prosječna popunjenost međuspremnika implementiranih algoritama. Prosječna popunjenost međuspremnika kod svih algoritama premašuje granicu α što znači da se spremnik nalazi u optimalnom području rada. Prosječna odabrana razina kvalitete u svim slučajevima dostiže vrlo visok stupanj. Budući da osnovni algoritam ne koristi međuspremnik pri radu tu vrijednost nije moguće upisati u tablicu.

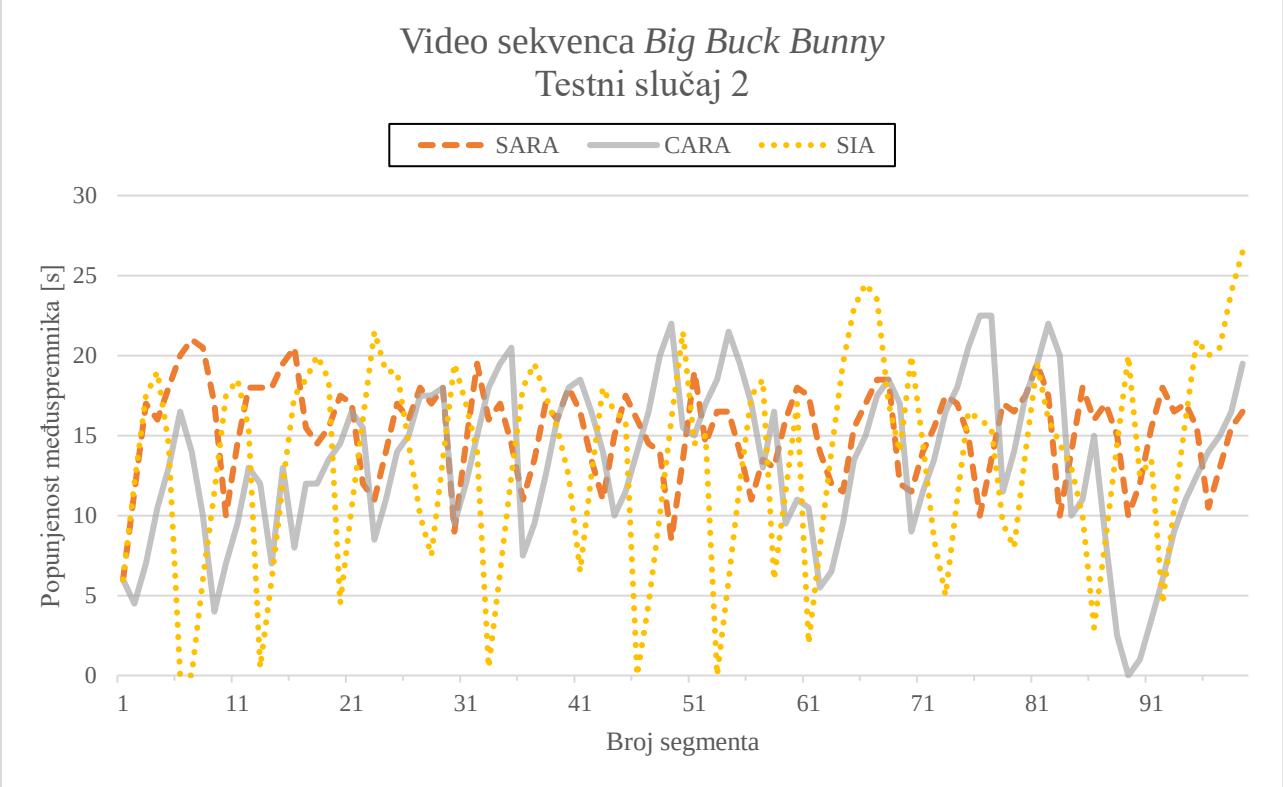
Tablica 4.3. Prikaz rezultata dobivenih za prvi testni slučaj

	LIU	SARA	CARA	SIA
Prosječna razina kvalitete	17	19	19	18
Prosječna popunjenost međuspremnika [s]	-	29,3	29,3	30,6

U drugom testnom slučaju prosječna brzina preuzimanja je znatno niža nego u prvom slučaju. Sukladno tome i prosječna razina kvalitete segmenta je manja kao i popunjenost međuspremnika (slika 4.4-4.5.). Početni porast razine kvalitete odvija se sporije, a promjene kvalitete su češće. Oscilacija u kvaliteti videa posebice je izražena kod SIA algoritma. Zbog male razlike između granica I (12 sekundi) i B_α (16 sekundi) algoritam uglavnom radi u načinu brzog početka gdje se automatski odabire najniža razina te u načinu odabira najveće podržane razine. Posljedice toga su česte visoke oscilacije u kvaliteti i pretjerano pražnjenje međuspremnika. SARA algoritam koji svojim načinom rada sliči SIA algoritmu također sadrži oscilacije u kvaliteti, ali su one rjeđe i manjeg intenziteta. Razina međuspremnika SARA algoritma kreće se u rasponu od 10 do 20 sekundi, a pražnjenje ispod početne razine nije učestalo kao što je slučaj kod CARA i SIA algoritama. Kada je riječ o odabiru kvalitete pri ovom testnom slučaju, algoritmi LIU i CARA rijetko odabiru segment najveće kvalitete te se uglavnom kreću oko razina 12 i 14 što odgovara vrijednostima 1,2 i 2,1 Mbit/s.



Slika 4.4. Odabir kvalitete segmenta pri drugom testnom slučaju



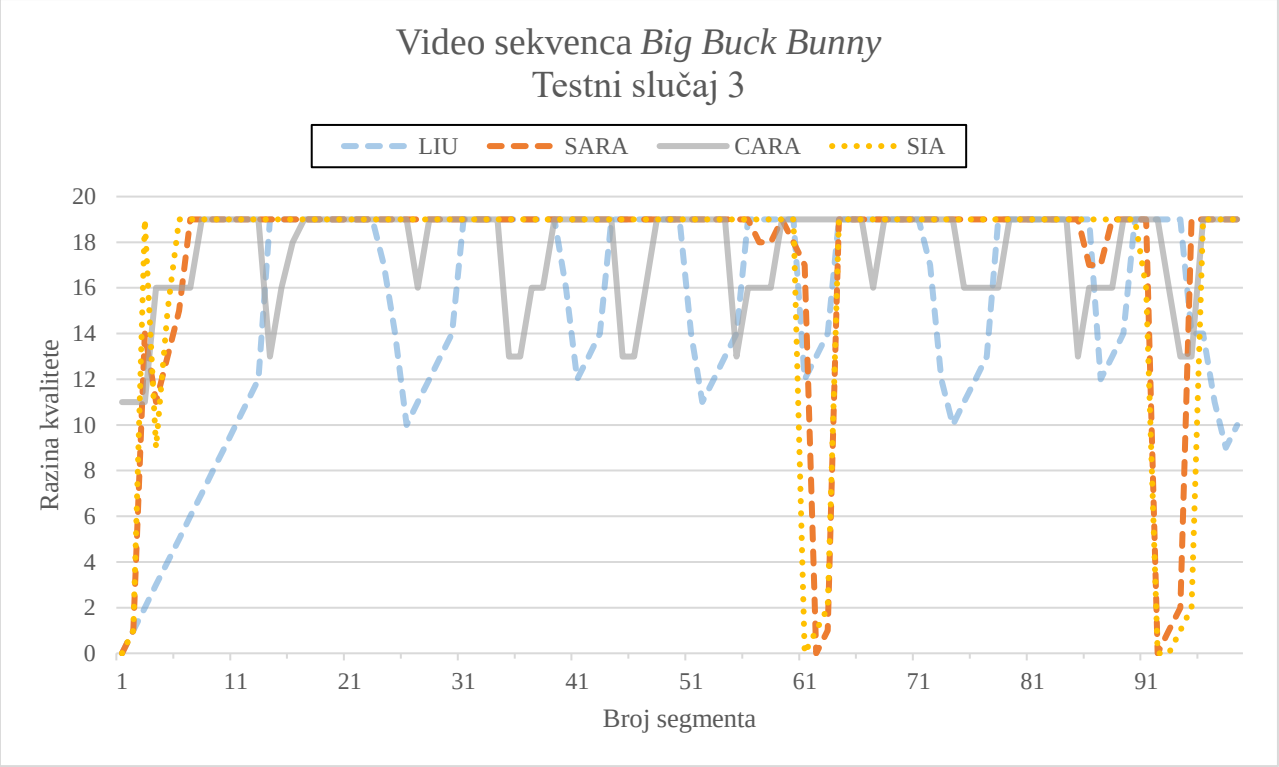
Slika 4.5. Kretanje razine popunjenosti međuspremnika za drugi testni slučaj

Srednja vrijednost odabrane razine kvalitete i prosječna popunjenost međuspremnika prikazana je tablicom 4.4. u nastavku. Prosječna popunjenost međuspremnika nalazi se između razina I i α što dovodi do česte promjene u načinima rada algoritama. Odabrana razina kvalitete niža je kod LIU i SIA algoritama nego kod SARA i CARA algoritama.

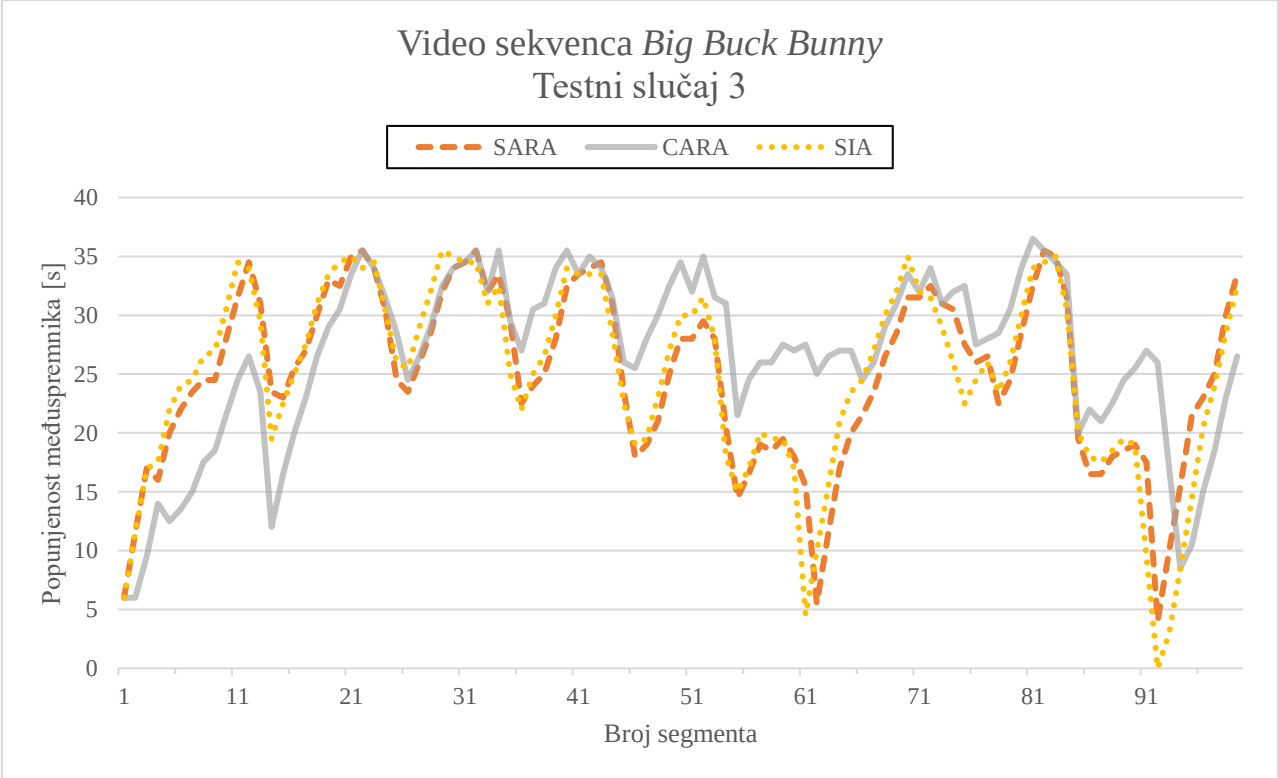
Tablica 4.4. Rezultati dobiveni u drugom testnom slučaju

	LIU	SARA	CARA	SIA
Prosječna razina kvalitete	11	14	14	11
Prosječna popunjenost međuspremnika [s]	-	15,3	13,4	13,4

Slike 4.6. i 4.7. predstavljaju rezultate mjerenja dobivene u trećem slučaju. Odabir razine kvalitete vrlo je sličan rezultatima dobivenim za prvi slučaj. Kod LIU algoritma može se uočiti da je početni rast kvalitete sporiji, a broj promjena razina kvalitete manji nego u prvom slučaju. Isto razmatranje se može primijeniti i na CARA algoritam. Kod SIA i SARA algoritama nakon izlaska iz početnog stanja razina međuspremnika pada ispod početne razine na dva mjesta. To se očituje naglim padom kvalitete segmenata na mjestima 60-62 i 92-95. Izuzev ta dva slučaja, SIA i SARA algoritmi održavaju najvišu razinu kvalitete kao i u prvom testnom slučaju, ali je za razliku ovdje prosječna razina međuspremnika niža. Iz tog razloga preuzimanje segmenata se ne zaustavlja toliko često.



Slika 4.6. Odabir razine kvalitete u trećem testnom slučaju



Slika 4.7. Promjena razine međuspremnika za treći testni slučaj

U tablici 4.5. se nalaze rezultati trećeg testnog slučaja. Najmanja prosječna razina kvalitete ostvarena je kod LIU algoritma. Za preostale algoritme prosječna popunjenost međuspremnika nalazi se u optimalnom području, kao i u prvom testnom slučaju.

Tablica 4.5. Rezultati testiranja za treći testni slučaj

	LIU	SARA	CARA	SIA
Prosječna razina kvalitete	15	17	18	17
Prosječna popunjenost međuspremnika [s]	-	24,9	26,5	25

Četvrti i peti testni slučajevi postižu rezultate vrlo slične trećem testnom slučaju te iz tog razloga nisu prikazani u ovom poglavlju. Cjelokupne tablice s mjerenjima i grafovima za svaku video sekvencu nalaze se u pripadajućim datotekama na priloženom elektroničkom mediju.

5. ZAKLJUČAK

Implementirani algoritmi u gotovo svim prikazanim slučajevima predstavljaju poboljšanje u odnosu na osnovni LIU algoritam. Iznimka je SIA algoritam u drugom testnom slučaju gdje dolazi do učestalih naglih promjena u odabiru razina kvalitete. Razlog tomu je pogreška pri procjeni vremena potrebnog za preuzimanje nadolazećeg segmenta uslijed česte promjene propusnosti mreže i mala razlika između granice B_α i donje granice (I). Svaki pad razine međuspremnika ispod donje granice automatski uzrokuje odabir segmenta najniže kvalitete. Povećanjem vrijednosti parametra B_α povećala bi se popunjenost međuspremnika te bi se održala veća razina kvalitete tijekom dužeg perioda što bi na kraju dovelo do boljeg korisničkog iskustva. U slučajevima kod kojih propusnost mreže poprima vrijednosti veće od maksimalne potrebne propusnosti, SARA i SIA algoritmi uspijevaju održati najveću dostupnu razinu kvalitete gotovo tijekom cijelog trajanja preuzimanja videa. LIU i CARA algoritmi su se u istim slučajevima pokazali osjetljivijim na promjenu propusnosti mreže što je rezultiralo čestim promjenama u odabiru kvalitete. Međutim, u slučajevima gdje je propusnost mreže bila niska, u rasponu od 1 do 3 Mbit/s, LIU i CARA algoritmi su ostvarili bolje rezultate. Kod tih algoritama odabrana razina kvalitete se često mijenjala, ali ta promjena nije izražena kao kod SARA i SIA algoritama. Rezultati mjerenja za preostale dvije video sekvence, Elephants Dream i Of Forest and Men, vrlo su slični rezultatima za video sekvencu Big Buck Bunny što znači da u ovom slučaju učinkovitost algoritama ne ovisi o sadržaju. Da bi se sa sigurnošću moglo utvrditi utječe li vrsta video sadržaja na učinkovitost algoritama potrebno je obaviti dodatna mjerenja nad većim brojem različitih video sekvenci. Pri mjerenju učinkovitosti u ovom radu postavljeni parametri algoritama su konstantni, a isto vrijedi i za trajanje segmenata. Ovaj rad bi stoga bilo moguće proširiti vršenjem mjerenja nad dodatnim testnim slučajevima u svrhu pronalaska optimalnih vrijednosti parametara. Drugo proširenje evaluacije učinkovitosti algoritama može se provesti izvođenjem testova nad video sekvencama s različitim trajanjima pojedinačnih segmenata i pronalaskom veze između vrijednosti postavljenih parametara i duljine trajanja segmenata.

LITERATURA

[1] https://www.jmarshall.com/easy/http/ [pristup ostvaren 12.7.2018.]

[2] https://docs.oracle.com/cd/E19455-01/806-0916/ipov-10/index.html

[pristup ostvaren 12.7.2018.]

[3] I. Sodagar, „The MPEG-DASH standard for multimedia streaming over the Internet“, IEEE MultiMedia, sv. 18, br. 4, str. 62-67, travanj 2011.

[4] https://bitmovin.com/dynamic-adaptive-streaming-http-mpeg-dash/

[pristup ostvaren 13.7.2018.]

[5] https://www.w3.org/2011/09/webtv/slides/W3C-Workshop.pdf [pristup ostvaren 18.7.2018.]

[6] C. Liu, I. Bouazizi, M. Gabbouj, „Rate adaptation for adaptive HTTP streaming“, ACM MMSys, str. 169-174, veljača 2011.

[7] P. Juluri, V. Tamarapalli, and D. Medhi, „SARA: Segment aware rate adaptation algorithm for dynamic adaptive streaming over HTTP“, IEEE ICCW, str. 1765-1770, lipanj 2015.

[8] M. Kim, J. Park, K. Chung, „Content-aware rate adaptation scheme to improve stability in HTTP adaptive streaming“, IEEE ICOIN, str. 401-405, travanj 2017.

[9] L. D. Cicco, S. Mascolo, and V. Palmisano, „Feedback control for adaptive live video streaming”, ACM MMSys, str. 145–156, veljača 2011.

[10] Y. Zheng, Y. Wang, Y. Liu, „SIA: Segment importance based rate adaptation for Dynamic Adaptive Streaming over HTTP”, IEEE ICNIDC, str. 442-446, rujan 2016.

[11] https://github.com/magnific0/wondershaper [pristup ostvaren 16.9.2018.]

[12] Stefan Lederer, Christopher Müller and Christian Timmerer, “Dynamic Adaptive Streaming over HTTP Dataset”, ACM MMSys, veljača 2012.

POPIS KRATICA

HTTP – *Hypertext Transfer Protocol*

MPEG-DASH – *Moving Picture Experts Group Dynamic Adaptive Streaming over HTTP*

TCP – *Transmission Control Protocol*

IP – *Internet Protocol*

WWW – *World Wide Web*

URL – *Uniform Resource Locator*

RTP – *Real-Time Transfer Protocol*

CDN – *Content Delivery Network*

AAC – *Advanced Audio Coding*

MPD – *Media Presentation Description*

XML – *Extensible Markup Language*

SAP – *Stream Access Point*

SARA – *Segment Aware Rate Adaptation*

CARA – *Content Aware Rate Adaptation*

SIA – *Segment Importance Based Rate Adaptation*

SRS – *Segment Request Strategy*

SIM – *Segment Importance Model*

TC – *Traffic control*

SAŽETAK

U današnjem svijetu velik dio internetskog prometa odlazi na prijenos video signala. Da bi se krajnjem korisniku mogla pružiti što bolja usluga, tijekom godina su razvijena razna rješenja. Jedno od popularnijih rješenja pod nazivom MPEG-DASH korišteno je u okviru ovog rada. MPEG-DASH se temelji na HTTP protokolu aplikacijskog sloja TCP/IP mrežnog modela i omogućava adaptivan prijenos video signala promjenom kvalitete video sadržaja ovisno o parametrima mreže. Kod MPEG-DASH standarda video sadržaj je podijeljen na segmente jednakog trajanja koji su kodirani u više razina kvalitete i postavljeni na poslužitelju. Za odabir segmenata odgovarajuće razine kvalitete koriste se adaptacijski algoritmi smješteni na klijentskoj strani. Iako svi segmenti imaju jednako vremensko trajanje njihova veličina se razlikuje ovisno o sadržaju kojeg prenose. Radi bolje procjene vremena potrebnog za preuzimanje segmenta predlaže se korištenje podatka o veličini pojedinačnog segmenta pri odabiru. U ovom radu analizirana su i implementirana tri odabrana adaptacijska algoritma koji u obzir uzimaju veličinu segmenta. Nakon implementacije izvršena su mjerenja nad različitim video sekvencama, a rezultati su uspoređeni s osnovnim adaptacijskim algoritmom.

Ključne riječi: prijenosni tok, mreža, HTTP, MPEG-DASH, adaptacija, segment, veličina

ABSTRACT

Analysis and implementation of adaptation algorithms used in adaptive video streaming which take segment size into account

Nowadays video streaming services make up majority of global Internet traffic. In order to provide best possible quality of service for users, various solutions were developed. One of the most widespread standards, named MPEG-DASH, is used in this paper. MPEG-DASH is based on application layer HTTP protocol of TCP/IP network model and enables adaptive video streaming considering current network parameters. Video content is divided into smaller segments of same duration encoded in different bit rates and stored on server. During media download client side software chooses segments of appropriate bit rate by using adaptation algorithms. Although each segment has the same duration, their sizes vary depending on content. To provide better estimation of time required to download next segment, it was suggested to utilize information about size of each segment. In this paper three segment aware adaptation algorithms were analyzed and implemented. Afterwards their performance was tested and results were compared with results achieved by basic adaptation algorithm.

Keywords: streaming, network, HTTP, MPEG-DASH, adaptation, segment, size

ŽIVOTOPIS

Dominik Grabić rođen je 7.3.1994. u Osijeku. U rujnu 2008. godine upisuje III. gimnaziju Osijek koju završava 2012. godine. Iste godine upisuje sveučilišni preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku. Završetkom preddiplomskog studija 2016. godine stječe zvanje: sveučilišni prvostupnik (baccalaureus) inženjer računarstva. Obrazovanje nastavlja iste godine upisom sveučilišnog diplomskog studija računarstva, smjer Programsko inženjerstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. U listopadu 2016. godine postaje stipendist Instituta RT-RK Osijek.

_____

Dominik Grabić