SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

ANALIZA I IMPLEMENTACIJA ADAPTACIJSKIH ALGORITAMA KORIŠTENIH U ADAPTIVNOM PRIJENOSU VIDEO SIGNALA KOJI U OBZIR UZIMAJU KVALITETU KORISNIČKOG ISKUSTVA

Diplomski rad

Filip Vranješ

Osijek, 2018.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za obranu diplomskog rada

Osijek, 23.09.2018.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za obranu diplomskog rada

Ime i prezime studenta:	Filip Vranješ
Studij, smjer:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 843 R, 27.09.2017.
OIB studenta:	60430794969
Mentor:	Prof.dr.sc. Snježana Rimac-Drlje
Sumentor:	Jelena Vlaović
Sumentor iz tvrtke:	Matija Pul
Predsjednik Povjerenstva:	izv. prof. dr.sc. Josip Job
Član Povjerenstva:	Jelena Vlaović
Naslov diplomskog rada:	Analiza i implementacija adaptacijskih algoritama korištenih u adaptivnom prijenosu video signala koji u obzir uzimaju kvalitetu korisničkog iskustva
Znanstvena grana rada:	**Telekomunikacije i informatika (zn. polje elektrotehnika)**
Zadatak diplomskog rada:	MPEG DASH definira način obavještavanja korisnika o nizu prijenosnih tokova različite kvalitete, zajedno s informacijama potrebnim za odabir odgovarajućeg prijenosnog toka. 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. Utjecaj prijelaznih pojava na kvalitetu posebno je važan za DASH sustave te pogrešne odluke u odabiru razine kvalitete mogu imati veliki utjecaj na opaženu kvalitetu. Dok parametri kvalitete usluge mogu dati dobar uvid u ono što se događa u mreži, kvalitetu korisničkog iskustva opisuju parametri iskustva samog korisnika. Na opaženu kvalitetu
Prijedlog ocjene pismenog dijela ispita (diplomskog rada):	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	23.09.2018.

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, 03.10.2018.

Ime i prezime studenta:	Filip Vranješ
Studij:	Diplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	D 843 R, 27.09.2017.
Ephorus podudaranje [%]:	2

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

izrađen pod vodstvom mentora Prof.dr.sc. Snježana Rimac-Drlje

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

Tema ovog diplomskog rada je analiza i implementacija adaptacijskih algoritama korištenih u adaptivnom prijenosu video signala koji u obzir uzimaju kvalitetu korisničkog iskustva. Adaptivni prijenos video signala realiziran je putem standarda razvijenog od strane ekspertne skupine *Moving Picture Experts Group*, pod nazivom *Dynamic Adaptive Streaming over HTTP* (MPEG-DASH) koji omogućava strujanje (engl. *streaming*) multimedijskih sadržaja promjenjive kvalitete preko postojeće HTTP web infrastrukture.

Osnovna ideja adaptivnog prijenosa je da se sadržaj podijeli u niz manjih segmenata fiksne duljine, tako da svaki segment sadrži određeni interval medijskog sadržaja. Segmenti su dostupni na poslužitelju u više različitih kvaliteta odnosno kodirani u različitim brzinama prijenosa video signala, a klijentski program prilikom gledanja video zapisa automatski odabire sljedeći segment maksimalne kvalitete ovisno o stanju u mreži. Tako se povećava korisničko iskustvo jer nema neželjenog blokiranja reprodukcije (engl. *buffering*) u nestabilnim mrežnim uvjetima.

Budući da će prema određenim istraživanjima, u narednim godinama veliku većinu internetskog prometa činiti video zapisi, pojavila se potreba za standardom koji omogućava kompatibilnost velikog broja uređaja i poslužitelja. Za razliku od do sada korištenih tehnologija poput *HTTP Live Streaming*-a razvijenog od strane Apple-a i *Smooth Streaming* razvijenog od strane Microsoft-a koji su zaštićeni autorskim pravima, MPEG-DASH je međunarodni standard na čijem su razvoju sudjelovale tvrtke poput: Microsoft, Adobe, Apple, Samsung, Akamai, Cisco, Dolby, Ericsson, Harmonic, Qualcomm, Netflix, Intel, Bitmovin itd. Standard je ratificiran 2012. godine i trenutno se primjenjuje na YouTube-u, Netflix-u i sličnim servisima.

U radu su navedene teorijske osnove za razumijevanje MPEG-DASH standarda te njegove mogućnosti, opisan je klijent, poslužitelj i njihova interakcija. Također opisan je i korišteni klijentski program s osnovnim adaptacijskim algoritmom koji predstavlja temelj za nadogradnju implementiranih algoritama. U zasebnom poglavlju opisana su tri implementirana algoritma, njihov teorijski opis, dijagram toka i pseudo kod. U zadnjem poglavlju opisani su načini testiranja implementiranih algoritama. Prikazani su grafovi i usporedbe algoritama na temelju parametara kvalitete korisničkog iskustva (engl. Quality of Experience - QoE). U radu su korišteni C i C++ programski jezici.

2. MPEG-DASH

Hyper Text Transfer Protocol (HTTP) je postao glavna i najčešća metoda prijenosa informacija na World Wide Web-u . Na primjer Adobe-ov *HTTP Dynamic Streaming*, Microsoft-ov *Smooth Streaming* i Apple-ov HTTP *Live Streaming* koriste HTTP kao temeljnu metodu isporuke. Međutim svaka implementacija koristi drugačiji format segmenata, stoga kako bi primio sadržaj s pojedinog poslužitelja, uređaj mora podržavati odgovarajući klijentski protokol koji je kod navedenih zaštićen autorskim pravima. Zbog navedenih problema pojavio se MPEG-DASH kao standard za HTTP strujanje (engl. *streaming*) multimedijskog sadržaja koji omogućava standardiziranom klijentu strujanje sadržaja s bilo kojeg standardiziranog servera čime je omogućena kompatibilnost klijenta i servera različitih proizvođača [3].

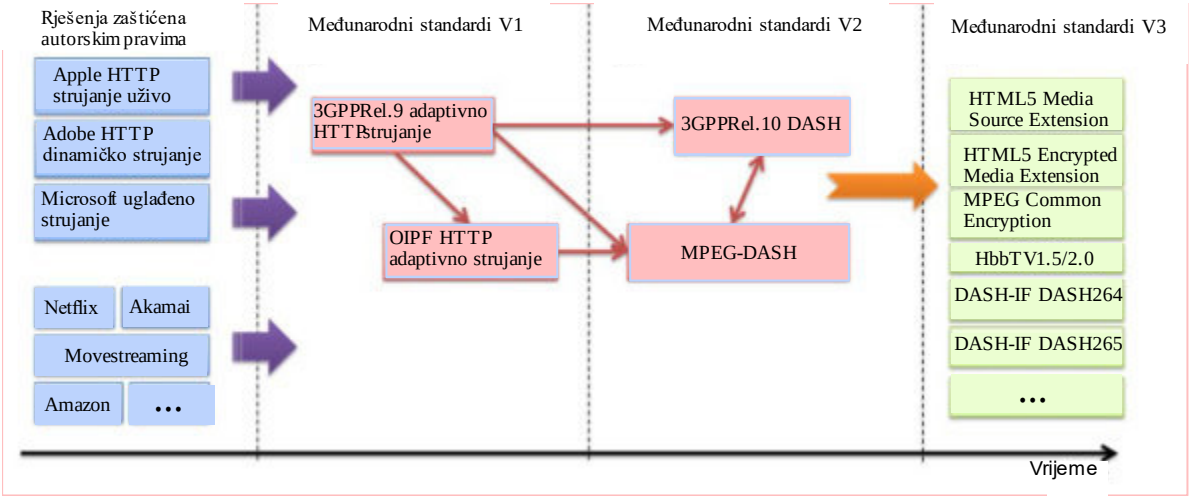
MPEG DASH definira način obavještavanja korisnika o nizu prijenosnih tokova različite kvalitete, zajedno s informacijama potrebnim za odabir odgovarajućeg prijenosnog toka. Postojanje različitih formata datoteka omogućuje učinkovito i neprimjetno prebacivanje između prijenosnih tokova različite kvalitete, što omogućuje prilagodbu promjenjivim uvjetima u mreži bez zaustavljanja prikaza video zapisa. Utjecaj prijelaznih pojava na kvalitetu posebno je važan za DASH sustave te pogrešne odluke u odabiru razine kvalitete mogu imati veliki utjecaj na opaženu kvalitetu. Dok parametri kvalitete usluge mogu dati dobar uvid u ono što se događa u mreži, kvalitetu korisničkog iskustva opisuju parametri iskustva samog korisnika.

Na opaženu kvalitetu različito utječu npr. kvaliteta slike, prostorna ili vremenska adaptacija i slični parametri video zapisa. Ovisno o sadržaju, prijelaz s jedne razine kvalitete na drugu više će ili manje utjecati na opaženu kvalitetu. Dakle, kada se priprema sadržaj za prijenos treba izvršiti i analizu sadržaja kako bi se omogućila poboljšana segmentacija video zapisa i tako bi se izabrao najpovoljniji scenarij adaptacije.

2.1. Standardizacija

Moving Picture Expert Group (MPEG) je radna skupina osnovana 1988. od strane organizacija *International Organization for Standardization* (ISO) i *International Electrotehnical Comission* (IEC) radi donošenja standarda za sažimanje (kompresiju) i prijenos audio i video signala. Do danas je donijela veliki broj različitih standarda koji se koriste u proizvodnji i emitiranju radijskoga i TV signala, kao što su npr. MPEG-1, MPEG-2 i MPEG-4 porodice standarda za sažimanje i

prijenos audio i video signala, a zatim i MPEG-DASH. Promatrajući potrebe tržišta i zahtjeve industrije, MPEG je objavio poziv za prijedloge HTTP standarda za strujanje 2009. godine. Nakon što je do kraja iste godine primio 15 potpunih prijedloga, u dvije naredne godine MPEG je izradio specifikaciju uz sudjelovanje mnogih stručnjaka i uz suradnju s drugim standardizacijskim grupama poput *Third Generation Partnership Project* (3GPP). U siječnju 2011. godine MPEG-DASH je postao nacrt međunarodnog standarda, a u studenom iste godine, međunarodni standard. U travnju 2012. godine objavljen je kao standard ISO/IEC 23009-1. Na slici 2.1. prikazan je proces standardizacije tehnologija za adaptacijsko strujanje medijskih sadržaja.

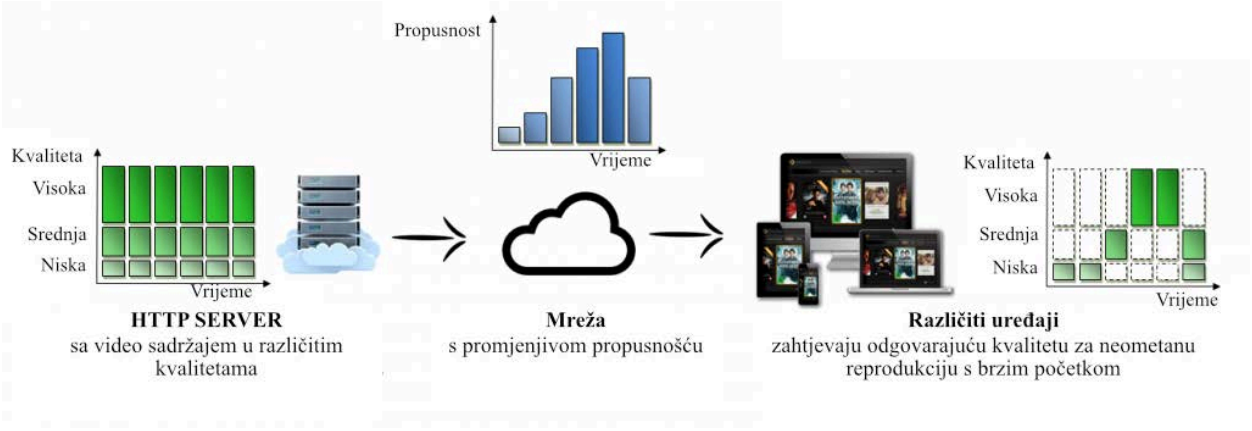


Sl 2.1 Faze standardizacije MPEG-DASH [1]

Posljednjih godina MPEG-DASH je uključen u nove standardizacijske faze kao na primjer HTML5 *Media Source Extension* odnosno podrška za jednostavnu DASH reprodukciju putem HTML5 video i audio tagova, *HTML5 Encrypted Media Extensions* što omogućava reprodukciju zaštićenu DRM-om (engl. *Digital Rights Management*) u Internet pregledniku. Nadalje DRM zaštita je usklađena na različitim sistemima putem *Common Encryption* (MPEG-CENC), a kroz integraciju u HbbTV (engl. *Hybrid broadcast broadband TV*) omogućena je MPEG-DASH reprodukcija na različitim *SmartTV* platformama. Danas MPEG-DASH dobiva sve više i više implementacija poput Netflix-a i Google-a koji su nedavno prebačeni na navedeni standard. S ova dva velika izvora internetskog prometa može se zaključiti da je oko 50 % ukupnog internet prometa već temeljeno na MPEG-DASH standardu.

2.2. **Princip rada**

Osnovni princip rada dinamičkog strujanja putem MPEG-DASH standarda je da se medijska datoteka podjeli u segmente koji se mogu kodirati u različitim prostornim rezolucijama odnosno kvaliteti, prilagođenim odgovarajućim brzinama prijenosa. Segmenti se nalaze na web serveru i pristupa im se putem HTTP standarda tj. putem GET zahtjeva. Na slici 2.2. vidljivo je da su na HTTP serveru dostupne tri različite kvalitete (niska, srednja i visoka) podijeljene u segmente jednake duljine. Prilagodba na odgovarajuću kvalitetu prijenosnog toka odvija se na klijentskoj strani gdje klijent može preuzeti segment više kvalitete ukoliko propusnost to dozvoljava. Način na koji klijent odabire segmente ovisi o adaptacijskom algoritmu koji se primjenjuje. U ovom je radu u zasebnom poglavlju opisano nekoliko algoritama koji u obzir uzimaju korisničko iskustvo. Nakon što klijent preuzme sadržaj prema primijenjenom adaptacijskom algoritmu, sadržaj promjenjive kvalitete se prikazuje na korisničkom uređaju bez prekida uslijed promjena u mreži koje bi inače narušavale percipiranu kvalitetu korisničkog iskustva.



Sl 2.2 Osnovni princip rada dinamičkog strujanja [2]

2.3. **MPD**

Kako bi opisao vremenske i strukturne odnose među segmentima, MPEG-DASH uvodi *Media Presentation Description* (MPD) datoteku. MPD je *Extensible Markup Language* (XML) datoteka koja predstavlja hijerarhijski podatkovni model unutar kojeg su pojedini segmenti različite kvalitete definirani u obliku HTTP poveznica (engl. *Uniform Resource Locator, URL*). Unutar MPD-a nalaze se podaci o brzini prijenosa, rezoluciji, početnom vremenu, trajanju segmenta itd. Budući da se putanje do segmenata i drugi bitni podaci nalaze u MPD datoteci, svaki klijent prvo

dohvaća MPD datoteku, a zatim na osnovu podataka koje sadrži zahtijeva pojedine segmente koji najbolje odgovaraju zahtjevima u smislu brzine i kvalitete.

Na vrhu hijerarhijske strukture MPD datoteke nalazi se *Period* koji opisuje sadržaj atributom trajanja, a unutar svakog *Period* tag-a mogu se nalaziti medijske komponente poput video zapisa za prikaz iz različitih perspektiva ili videa s različitim kodecima, audio komponente za različite jezike, prijevode i sl. MPD može koristiti i više od jednog *Period*-a kada se želi razdvojiti sadržaj za ubacivanje reklama, mijenjanje perspektive prikaza tijekom prijenosa uživo. Ako se želi prikazivati reklame samo u visokoj rezoluciji, moguće je kreirati *Period* s reklamama u visokoj rezoluciji i postaviti ga između dva *Period*-a medijskog sadržaja (film) koji je dostupan za različite brzine i rezolucije od standardne do visoke rezolucije.

Svaki *Period* može sadržavati jedan ili više adaptacijskih setova (engl. *Adaption Set*) unutar kojeg se može nalaziti jedan ili više podatkovnih tokova. Na primjer jedan adaptacijski set može sadržavati više podatkovnih tokova za audio (stereo niže kvalitete i *surround* visoke kvalitete) istog medijskog sadržaja, drugi adaptacijski set može imati više podatkovnih tokova za video zapis itd. Također se unutar adaptacijskog seta mogu nalaziti prijevodi i drugi metapodaci.

Adaptacijski set obično sadrži više reprezentacijskih (engl. *Representation*) elemenata koji predstavljaju alternative medijskog sadržaja, a međusobno se razlikuju po parametrima poput prostorne i vremenske rezolucije, broju kanala, propusnosti mreže i sl. To znači da se u svakom reprezentacijskom elementu nalazi isti sadržaj ali drugačije kodiran, što znači da bi za pružanje reprodukcije bila dovoljna jedna reprezentacija sadržaja, ali je na ovaj način omogućena promjena kvalitete prikaza u nestabilnim uvjetima mreže bez uzrokovanja zastoja (engl. *smooth playback*) jer klijent tijekom reprodukcije može odabirati segmente različitih reprezentacija koji se nalaze unutar *Representation* taga.

Unutar reprezentacijskih elemenata nalaze se elementi veličine segmenta (engl. *Segment Size*) koji imaju atribute *id, size* i *scale. Id* atribut se koristi kao URI (engl. *Uniform Resource Identifier*), odnosno za adresu segmenta na serveru kojemu se može pristupiti putem HTTP GET zahtjeva. *Size* atribut određuje veličinu kvantitativno u obliku znakovnog niza, a *scale* mjernu jedinicu u kojoj je izražena veličina segmenta. Unutar svakog reprezentacijskog elementa nalazi se cjelokupni video zapis podijeljen na segmente fiksne duljine 1, 2, 4 ili više sekundi te je za svaki definirana veličina korištenjem *Segment Size* elementa.

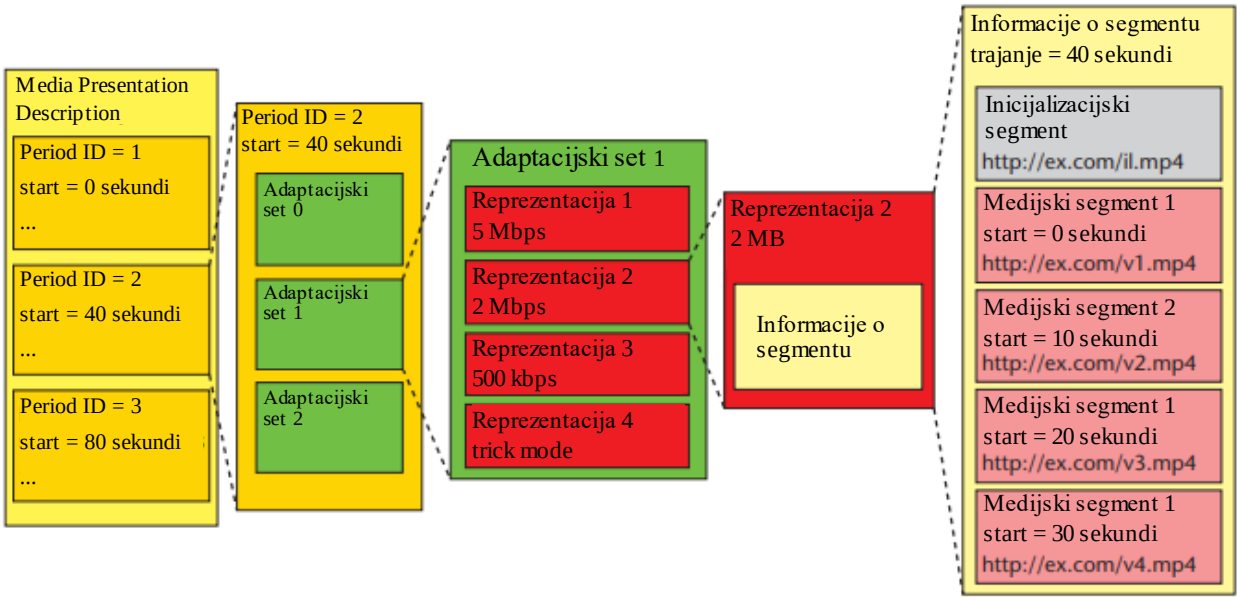
Na slici 2.3. prikazan je primjer formata MPD datoteke (XML format) za animirani video zapis naziva Big Buck Bunny (BBB). Prikazana MPD datoteka sadrži jedan *Period* trajanja 9 minuta i 56 sekundi, jedan adaptacijski set unutar kojeg se nalazi 20 reprezentacijskih elemenata.

```
▼<MPD xmlns="urn:mpeg:dash:schema:mpd:2011" minBufferTime="PT1.500000S" type="static"
 mediaPresentationDuration="PT0H9M56.46S" profiles="urn:mpeg:dash:profile:isoff-live:2011">
  ▼<Period duration="PT0H9M56.46S">
    ▼<AdaptationSet mimeType="video/mp4" segmentAlignment="true" group="1" maxWidth="480"
     maxHeight="360" maxFrameRate="24" par="4:3">
      <SegmentTemplate timescale="96" media="bunny_$Bandwidth$bps/BigBuckBunny_4s$Number$.m4s"
      startNumber="1" duration="384"
      initialization="bunny_$Bandwidth$bps/BigBuckBunny_4s_init.mp4"/>
      ▼<Representation id="320x240 45.0kbps" mimeType="video/mp4" codecs="avc1.42c00d" width="320"
      height="240" frameRate="24" sar="1:1" startWithSAP="1" bandwidth="45226">
        <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"/>
```

Sl 2.3 Primjer formata MPD datoteke

Kako bi započeo proces adaptivnog prijenosa, DASH klijent prvo parsira MPD datoteku. Nakon toga iz skupine dostupnih, klijent odabire reprezentacijski set za korištenje. Odabir je temeljen na korisničkom odabiru i mogućnostima na klijentskoj strani, a najčešće su ključni faktori za to, dostupna brzina prijenosa i popunjenosti međuspremnika (engl. *buffer*).

Na slici 2.4. prikazan je jednostavni primjer hijerarhijske strukture podataka jedne MPD datoteke. U primjeru prikazanom na slici 2.4. MPD sadrži tri *Period*-a koji su trajanja po 40 sekundi. Svaki *Period* sadrži tri adaptacijska seta unutar kojih se nalaze tri reprezentacije video zapisa kodirane različitim brzinama prijenosa, a jedna reprezentacija se koristi za tzv. *trick mode* (premotavanje naprijed-nazad). Unutar pojedinog reprezentacijskog elementa nalaze se informacije o medijskim segmentima koje uključuju inicijalizacijski segment i četiri medijska segmenta te pripadajuće URL putanje.



Sl 2.4 Struktura MPD datoteke [3]

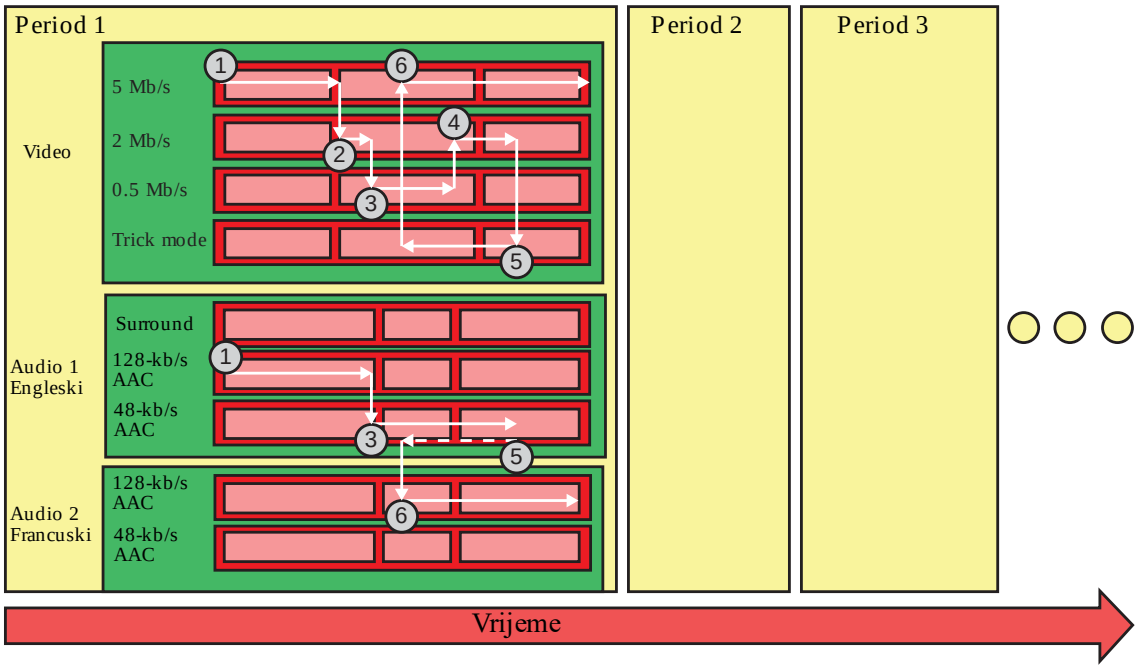
2.4. **Primjer MPEG-DASH dinamičkog strujanja**

Slika 2.5. prikazuje jednostavni primjer dinamičkog, adaptivnog strujanja primjenom MPEG-DASH standarda. U navedenom primjeru medijski sadržaj je podijeljen na video i audio komponente. Video zapis je kodiran u tri različite brzine prijenosa 5, 2 i 0.5 Mb/s, a dodatno je omogućen tzv. *trick mode.* Audio sadržaj dostupan je na engleskom i francuskom jeziku. Engleski jezik kodiran je u *surround sound* te AAC (engl. Advance Video Coding) s brzinama 128 kb/s i 48 kb/s. Francuski jezik kodiran je samo u AAC 128 kb/s i 48 kb/s brzinama. Kružići s brojevima na slici 2.5. prikazuju korake prilagodbe koje se izvode na klijentskoj strani prilikom dinamičkog strujanja.

Na početku (oznaka 1) započinje preuzimanje video segmenata najviše kvalitete. Audio se preuzima na engleskom jeziku 128 kb/s AAC jer u prikazanom primjeru uređaj za reprodukciju ne podržava *surround sound*. Nakon preuzimanja prvih audio i video segmenata, DASH klijent mjerenjem mrežne brzine prijenosa utvrdi da je brzina manja od 5 Mb/s pa na idućoj mogućoj točki promjene, spušta kvalitetu videa na 2 Mb/s (oznaka 2). Nakon određenog vremena klijent utvrdi da se mrežna brzina još smanjila pa odabire video zapis još manje kvalitete, ali ovaj put smanjuje i kvalitetu zvuka (oznaka 3). Prilikom sljedećeg povećanja brzine klijent ponovno odabire veću kvalitetu videa (oznaka 4). U određenom trenutku korisnik odluči zaustaviti prijenos i premotati sadržaj unazad. Tada započinje reprodukcija u *trick modu* bez zvuka (oznaka 5), a

korisnik nakon određenog vremena zaustavlja ovaj način reprodukcije i ponovno pokreće video zapis, ali ovaj put sa zvukom na francuskom jeziku (oznaka 6).

Navedeni primjer predstavlja najjednostavniji mogući scenarij dinamičkog strujanja multimedijskog sadržaja. Napredniji slučajevi mogu uključivati prebacivanje između različitih kamera odnosno kuta gledanja, strujanje 3D medijskog sadržaja, prijevode, dinamičko ubacivanje reklama itd.



Sl 2.5 Primjer MPEG-DASH dinamičkog strujanja **[3]**

2.5. **MPEG-DASH klijent**

Na klijentskoj strani korišten je program zaštićen vlasničkom komercijalnom licencom pisan u C i C++ programskim jezicima pokretan na Linux operacijskom sustavu. Unutar klijentskog programa pisani su svi algoritmi ovog diplomskog rada, a osnovna verzija sadrži algoritam temeljen na adaptacijskom algoritmu LIU[11]. Pseudo kod osnovnog adaptacijskog algoritma prikazan je na slici 2.6.

U programu je potrebno postaviti putanju do MPD datoteke na udaljenom poslužitelju. Nakon toga dohvaća se MPD datoteka čiji se sadržaj parsira i na temelju dobivenih podataka odabire prvi segment za preuzimanje. Programa tada kreira video datoteku h.264 formata i u nju sprema svaki preuzeti segment. Korišteni program ne omogućava reprodukciju preuzetog video zapisa nego se

po završetku preuzimanja reprodukcija može pokrenuti u nekom od postojećih programa za reprodukciju poput VLC *player*-a. Budući da reprodukcija preuzetog sadržaja nije omogućena unutar klijentskog programa, korisnik nema kontrolu nad odabirom kvalitete prikaza jer je odabir kvalitete određen isključivo korištenim adaptacijskim algoritmom. Također nije moguće koristiti *trick mode* odnosno zaustavljanja i premotavanje video zapisa.

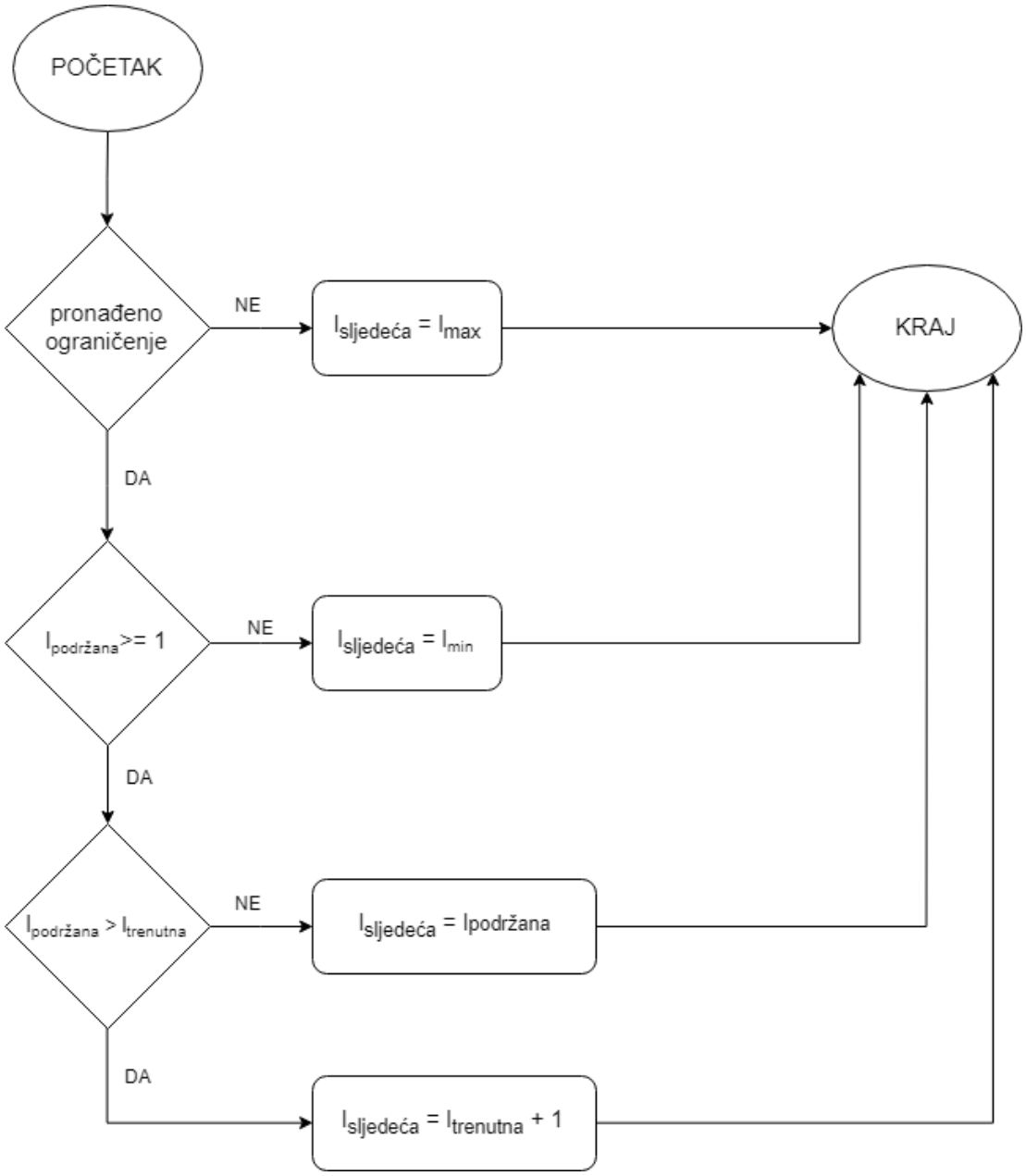
```
ograničenje = 0
za i od 0 do kvaliteta.maksimum činiti
    ako kvaliteta.od(i) ≥ brzina onda
        ograničenje = 1
        prekid
ako ograničenje == 0 onda
    kvaliteta_sljedeća = i
    kraj
ako i ≥ 1 onda
    ako (i − 1) > kvaliteta_trenutna
        kvaliteta_sljedeća = kvaliteta_trenutna + 1
    inače
        kvaliteta_sljedeća = i − 1
inače
    kvaliteta_sljedeća = 0
kraj
```

Sl 2.6 Pseudo kod osnovne verzije adaptacijskog algoritma DASH klijenta

Osnovni algoritam čiji pseudo kod je prikazan na slici 2.6. prvo traži postoji li ograničenje kvalitete s obzirom na trenutnu brzinu preuzimanja odnosno brzinu preuzimanja prethodnog segmenta. Prolazeći kroz petlju za vrijednosti iteratora i od nula, što predstavlja najmanju razinu kvalitete, do maksimalne vrijednosti kvalitete ovisno o dostupnim razinama, provjerava se zahtjeva li neka razina kvalitete brzinu veću od trenutno izmjerene brzine. Ako je takva razina pronađena, zastavica za ograničenje brzine $ograničenje$, se postavlja na vrijednost jedan te se prekida izvođenje petlje. U slučaju da ograničenje s obzirom na izmjerenu brzinu nije pronađeno, kvaliteta se postavlja na vrijednost korištenog iteratora koji je u tom slučaju prošao kroz cijelu

petlju svih dostupnih razina kvalitete pa njegova vrijednost predstavlja maksimalnu razinu kvalitete i nakon toga završava izvođenje algoritma do preuzimanja sljedećeg segmenta. Najniža razina kvalitete se odabire ako je iterator ostao na vrijednosti nula. U svakom drugom scenariju, provjerava se je li ograničenje pronađeno na razini koja je dvije ili više razina iznad trenutne. Ako je, razina kvalitete se povećava za jedan u odnosu na trenutnu, a ako nije postavlja se na jednu razinu ispod vrijednosti iteratora. Drugim riječima povećanje kvalitete izvodi se jednu po jednu razinu, a smanjenje kvalitete u jednom koraku na maksimalnu razinu koju izmjerena brzina podržava.

Na slici 2.7. prikazan je dijagram toka opisanog algoritma osnovne verzije DASH klijenta.



Sl 2.7. Dijagram toka osnovne verzije adaptacijskog algoritma DASH klijenta

3. ADAPTACIJSKI ALGORITMI

Adaptivno kodiranje predstavlja proces obradbe i prijenosa signala pri kojem se linearni i nelinearni sadržaji kodiraju u više profila, optimalne brzine prijenosa pri određenoj rezoluciji sadržaja prilagođenoj za reprodukciju i prikaz na različitim korisničkim uređajima – pametnim telefonima, tabletima, računalima i pametnim televizorima.

Adaptacijski algoritmi korišteni u MPEG-DASH klijentima mogu biti temeljeni na:

- međuspremniku (engl. buffer)
- propusnosti (engl. bandwidth) mreže
- međuspremniku i propusnosti mreže

U ovom diplomskom radu implementirana su sva tri navedena tipa adaptacijskih algoritama te su testiranjem prikazane usporedbe postignutih rezultata. Prvo je implementiran *A QoE-aware DASH system* (QDASH) algoritam [4] koji se temelji na izmjerenoj propusnosti mreže. Nakon toga algoritam zasnovan isključivo na međuspremniku pod nazivom *Buffer-Based Approach to Rate Adaptation* (BBA) [10] te algoritam *Quality of Experience QoE-enhanced adaptation algorithm over DASH* (QAAD) [11] koji promatra propusnost mreže ali i popunjenost međuspremnika.

3.1. QDASH algoritam

Do sada su predloženi različiti adaptacijski algoritmi koji poboljšavaju korisničko iskustvo odabirom odgovarajuće kvalitete video zapisa tijekom reprodukcije. Velik broj tih algoritama temelji se na popunjenosti međuspremnika, međutim iako korištenje međuspremnika može ublažiti fluktuacije mrežne propusnosti, takav pristup ne uzima u obzir utjecaj učestalih promjena kvalitete na percipiranu kvalitetu korisničkog iskustva.

Rezultati subjektivnih eksperimenata provedenih u [4] ukazuju na to da korisnici preferiraju postupne promjene kvalitete između najbolje i najlošije umjesto naglih promjena. S obzirom na spomenuto razvijen je QDASH algoritam, punim nazivom QoE-aware DASH, s ciljem poboljšanja korisničkog iskustva tijekom gledanja video sadržaja na zahtjev.

Unutar DASH sustava klijentski program obično odabire kvalitetu koja ima nižu brzinu prijenosa video zapisa (engl. *bit-rate*) od izmjerene propusnosti tako da je brzina preuzimanja veća

od brzine reprodukcije sadržaja. Tako se izbjegava blokiranje reprodukcije uslijed nedostatka podataka u međuspremniku. Međutim ponašanje različitih klijentskih programa nije jednako pa na primjer klijentski program korišten na servisu Netflix nastoji omogućiti višu razinu kvalitete puno agresivnije od Microsoft *Smooth Streaming* pristupa [5].

Jedan od problema u DASH sustavima koji u obzir uzimaju kvalitetu korisničkog iskustva predstavljaju subjektivne procjene korisnika te načini na koje ocjenjuju percipiranu kvalitetu. Istraživanja [6, 7, 8] ukazuju na to da su negativne reakcije korisnika na degradaciju kvalitete puno veće nego pozitivne reakcije na unaprjeđenje kvalitete.

QDASH algoritam omogućava praktično rješenje za poboljšanje korisničkog iskustva postojećih DASH sustava. Poslužitelji video sadržaja ne moraju ponovno kodirati postojeće video isječke niti instalirati dodatnu programsku podršku (engl. *software*) jer je algoritam potpuno prilagođen MPEG-DASH standardu i implementiran na klijentskoj strani.

Osnovna ideja QDASH algoritma je umetanje među-razina kvalitete prilikom smanjivanja kvalitete umjesto izravne promjene na željenu razinu jer rezultati testiranja provedeni na ispitanicima ukazuju na to da korisnici takvim načinom promjene dobivaju bolje korisničko iskustvo.

QDASH je usmjeren na trenutnu implementaciju u postojeće sustave pa je dizajniran za strujanje H.264 *Advanced Video Coding* (H.264/AVC) video segmenata. Glavni nedostatak korištenja H.264/AVC kodiranja predstavljaju veći troškovi kapaciteta za pohranu na serveru pa se zbog toga počelo koristiti i H.264 *Scalable Video Coding* (H.264/SVC) kodiranje koje kodira video zapis u slojeve kako bi se povećala učinkovitost. Autori u [9] uspoređuju troškove adaptivnog prijenosa u DASH sustavima korištenjem MPEG H.264/AVC i MPEG H.264/SVC načina kodiranja te zaključuju da iako SVC koristi oko 20 % više bitova za postizanje iste kvalitete u odnosu na AVC, njegova učinkovitost rezultira manjim zahtjevima za pohranu na serveru. Troškovi SVC kodiranja su manji sve dok je broj kvaliteta segmentiranog video sadržaja mali, a troškove isporuke video signala u najvećoj mjeri određuje propusnost pa spomenuta prednost nije toliko značajna. Također korištenje SVC-a zahtjeva ponovno kodiranje video sadržaja što dodatno povećava troškove pa je QDASH zbog svega navedenog usmjeren na korištenje H.264/AVC kodiranja.

Mjerenje propusnosti na klijentskoj strani računa se dijeljenjem broja preuzetih bitova s vremenom potrebnim za preuzimanje tog broja bitova

$$bw = \frac{K}{t}, \tag{3-1}$$

Gdje je K količina preuzetih bitova tijekom vremena t.

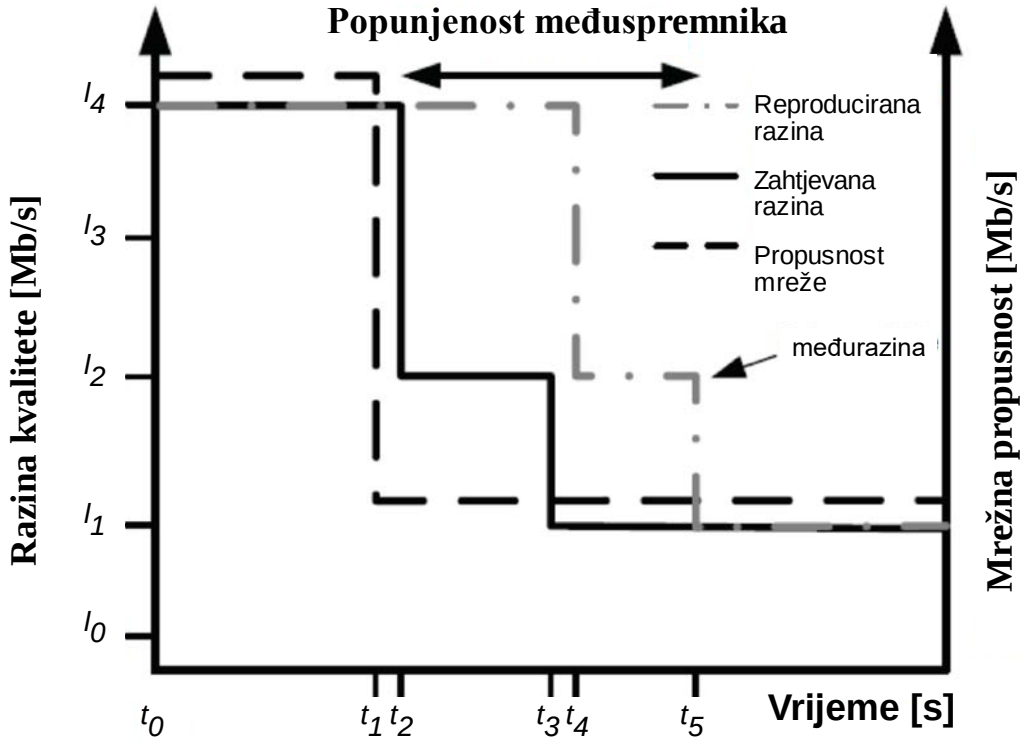
Prilikom mjerenja brzine koriste se sljedeće pretpostavke:

- Dostupne brzine prijenosa video zapisa svih razina kvalitete su poznate;
- Brzina između servera i klijenta je uvijek veća ili jednaka najmanjoj dostupnoj razini kvalitete videa;
- Klijent ima dovoljnu računalnu snagu za renderiranje svih dostupnih kvaliteta video zapisa.

Korištene pretpostavke su realistične s obzirom da DASH sustavi ne mogu poboljšati kvalitetu korisničkog iskustva u slučajevima kada klijent konstantno ima nedovoljno veliku brzinu prijenosa ili nema dovoljno računalne snage za prikaz video zapisa visoke kvalitete.

QDASH uzima u obzir rezultate provedenih testiranja koji pokazuju da se postepenom promjenom kvalitete umetanjem među-razina, postiže bolje korisničko iskustvo. Kao i kod drugih tehnologija poput UDP strujanja ili klasičnog strujanja, koristi se međuspremnik kako bi se smanjila mogućnost prekidanja reprodukcije prilikom promjene odnosno smanjenja brzine prijenosa. Iako je korištenjem međuspremnika moguće strujanje video zapisa veće kvalitete koju trenutna propusnost ne podržava, takav pristup nije preporučljiv jer može doći do zastoja reprodukcije, gubitka paketa te kao posljedica i blokirajućih okvira (engl. *frame*) zbog nedovršenog dekodiranja. Međutim ogovarajućim proračunima moguće je izbjeći neželjene pojave i omogućiti neprekinutu reprodukciju video zapisa veće kvalitete od dostupne brzine prijenosa tijekom kratkog vremenskog perioda. Tijekom spomenutog perioda klijent prilikom smanjenja dostupne brzine prvo preuzima segmente s kvalitetom između prethodne i ciljane sljedeće podržane kvalitete. Tako je omogućeno ublažavanje prijelaza kvalitete između prethodne (više) i sljedeće (niže) odnosno poboljšanje korisničkog iskustva.

Slika 3.1. prikazuje primjer korištenja međuspremnika za ublažavanje prijelaza kvalitete prikaza. Os ordinata na lijevoj strani predstavlja brzine prijenosa video zapisa dostupnih razina kvalitete, os ordinata na desnoj strani izmjerenu mrežnu propusnost, a apscisa predstavlja vrijeme. Ovaj primjer prikazuje video zapis kodiran u pet razina kvalitete $\{l_0,\ldots, l_4\}$. Puna, isprekidana i crta-točka linije predstavljaju razinu kvalitete video zapisa koju zahtijeva klijentski program, mrežnu propusnost i kvalitetu prikazanu korisniku, navedenim redoslijedom.



Sl 3.1. Smanjivanje kvalitete QDASH [4]

U trenutku t_0 popunjenost međuspremnika je maksimalna pa klijentski program dohvaća segmente i reproducira video zapis na maksimalnoj razini kvalitete l_4. Međutim u trenutku t_1 propusnost mreže naglo pada na vrijednost koja je malo iznad kvalitete l_1. U trenutku t_2 klijentski program detektira pad brzine ali ne smanjuje kvalitetu na razinu l_1 jer bi tako došlo do nagle degradacije percipirane kvalitete te kao posljedica narušilo kvalitetu korisničkog iskustva. Kako bi se izbjegli negativni efekti naglog smanjenja kvalitete, QDASH algoritam uvodi među-razinu koja je jednu razinu iznad razine koju podržava trenutna propusnost mreže. U navedenom primjeru sa slike 3.1., u trenutku t_2 podržana razina je l_1 međutim klijentski program preuzima i popunjava međuspremnik segmentima kvalitete l_2. Istovremeno dok preuzima segmente među razine $(t_2 – t_4)$, klijentski program i dalje reproducira video zapis na maksimalnoj razini l_4 sve do trenutka t_4 kada smanjuje kvalitetu na l_2 do trenutka t_5. U trenutku t_5 konačno smanjuje kvalitetu na podržanu razinu l_1. U navedenom scenariju u kojem je brzina reprodukcije veća od brzine preuzimanja segmenata tj. kada se međuspremnik prazni brže nego puni, može doći do neželjenog blokiranja reprodukcije ako se međuspremnik u potpunosti isprazni. Iako umetanje među razina može ublažiti degradaciju kvalitete prikaza, pojave blokiranja se pod svaku cijenu moraju izbjeći jer navedene drastično smanjuju kvalitetu korisničkog iskustva.

3.1.1. Dijagram toka i pseudo kod

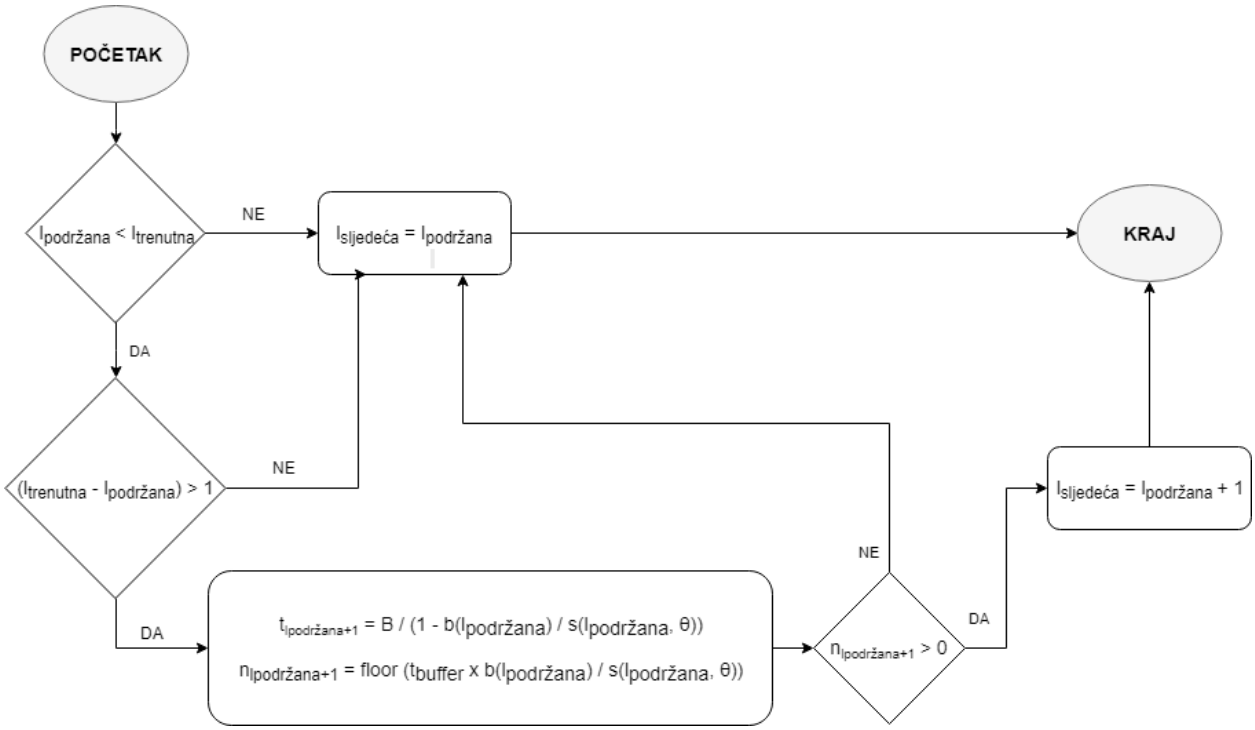
Slika 3.2. prikazuje dijagram toka QDASH algoritma. Na početku se provjerava postoji li ograničenje brzine odnosno je li došlo do smanjenja brzine prijenosa na isti način kao što je opisano kod osnovnog algoritma DASH klijenta. Ako nema ograničenja, sljedeća razina se postavlja na podržanu kvalitetu. Ako je podržana kvaliteta manja od trenutne, provjerava se je li razlika u kvaliteti veća od jedne razine. Ako nije ponovno se sljedeća razina postavlja na podržanu kvalitetu, a ako je izvršava se proračun kojim se računa broj segmenta među-razine za preuzimanje. Budući da je za računanje broja segmenata među-razine potreban podatak o prosječnoj veličini jednog video segmenta na podržanoj razini kvalitete $s(l_{podržana}, \theta)$, MPD datoteka je modificirana tako da sadrži podatke o veličini segmenata (Sl. 2.3.), a u klijentskom programu je omogućeno parsiranje tih podataka i računanje prosječne vrijednosti koja se onda prosljeđuje funkciji za adaptacijsku logiku. Nakon spomenutog proračuna ako je dobiveni broj segmenata među-razine veći od nula, sljedeća kvaliteta postavlja se na jednu razinu iznad trenutnom brzinom podržane kvalitete. Isti postupak opisan je u obliku pseudo koda na slici 3.3. Kod za QDASH algoritam u programskom jeziku C++ dan je u prilogu I.

Iz prikazanog dijagrama i pseudo koda vidljivo je da se kod QDASH algoritma kad god je to moguće smanjenje kvalitete izvršava postupno u 2 koraka na način da se kvaliteta prvo smanjuje na među-razinu, a tek onda na trenutno podržanu razinu kvalitete. Međutim povećanje kvalitete za razliku od smanjenja izvršava se agresivno u jednom koraku. Ovakav agresivni pristup povećanja koristi se s ciljem isporuke video zapisa maksimalne kvalitete krajnjem korisniku, ali to može dovesti do učestalih promjena kvalitete u promjenjivim mrežnim uvjetima, a time i smanjenja kvalitete korisničkog iskustva.

Parametri koje koristi QDASH algoritam su sljedeći:

- $l_{podržana}$ – razina kvalitete koju trenutna mrežna propusnost podržava [b/s]
- $l_{trenutna}$ – razina kvalitete posljednjeg preuzetog segmenta [b/s]
- $l_{sljedeća}$ – zahtijevana razina kvalitete video zapisa [b/s]
- $b(l)$ – brzina prijenosa video zapisa [b/s]
- $s(l, \theta)$ – prosječna veličina jednog video segmenta na razini kvalitete l [b]
- B – veličina međuspremnika izražena u sekundama reproduciranog video zapisa [s]
- $t_{l_{podržana}+1}$ – vremenski period za preuzimanje segmenata među-razine [s]

- $n_{l_{podržana}+1}$ – broj cjelovitih segmenata među-razine koji se mogu preuzeti tijekom perioda $t_{l_{podržana}+1}$ [s]



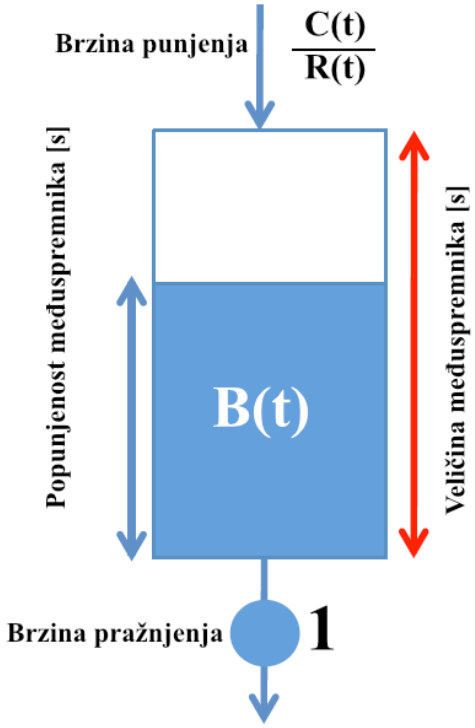
Sl 3.2. Dijagram toka QDASH adaptacijskog algoritma

ako $l_{podržana} < l_{trenutna}$ **onda**
  ako $(l_{trenutna} - l_{podržana}) > 1$ **onda**
    $t_{l_{podržana}+1} = \frac{B}{1-\frac{b(l_{podržana})}{s(l_{podržana},\theta)}}$
    $n_{l_{podržana}+1} = \left\lfloor t_{l_{podržana}+1} \times \frac{b(l_{podržana})}{s(l_{podržana},\theta)} \right\rfloor$
    ako $n_{l_{podržana}+1} > 0$ **onda**
      $l_{sljedeća} = l_{podržana} + 1$
    inače
      $l_{sljedeća} = l_{podržana}$
  inače
    $l_{sljedeća} = l_{podržana}$
inače
  $l_{sljedeća} = l_{podržana}$

Sl 3.3. Pseudo kod QDASH adaptacijskog algoritma

3.2. **BBA algoritam**

Većina postojećih adaptacijskih algoritama kao i prethodno opisani QDASH algoritam temelje se na predviđanju kapaciteta na temelju promatranja mrežne propusnosti. Međutim kod BBA (*Buffer-Based Approach to Rate Adaptation)* algoritma primjenjuje se drugačiji pristup. BBA algoritma, kao što i sami naziv sugerira, temelji se isključivo na promatranju međuspremnika. Razlog korištenja takvog pristupa je to što je predviđanje kapaciteta na temelju prošlih mjerenja pouzdano u uvjetima relativno stabilne propusnosti, međutim u realnim uvjetima moguće su učestale promjene uzrokovane različitim faktorima poput ometanja WiFi interneta, zagušenja mreže, zagušenja na klijentskoj strani (npr. anti-virusni program skenira dolazni HTTP promet) ili zagušenja na preopterećenom serveru.

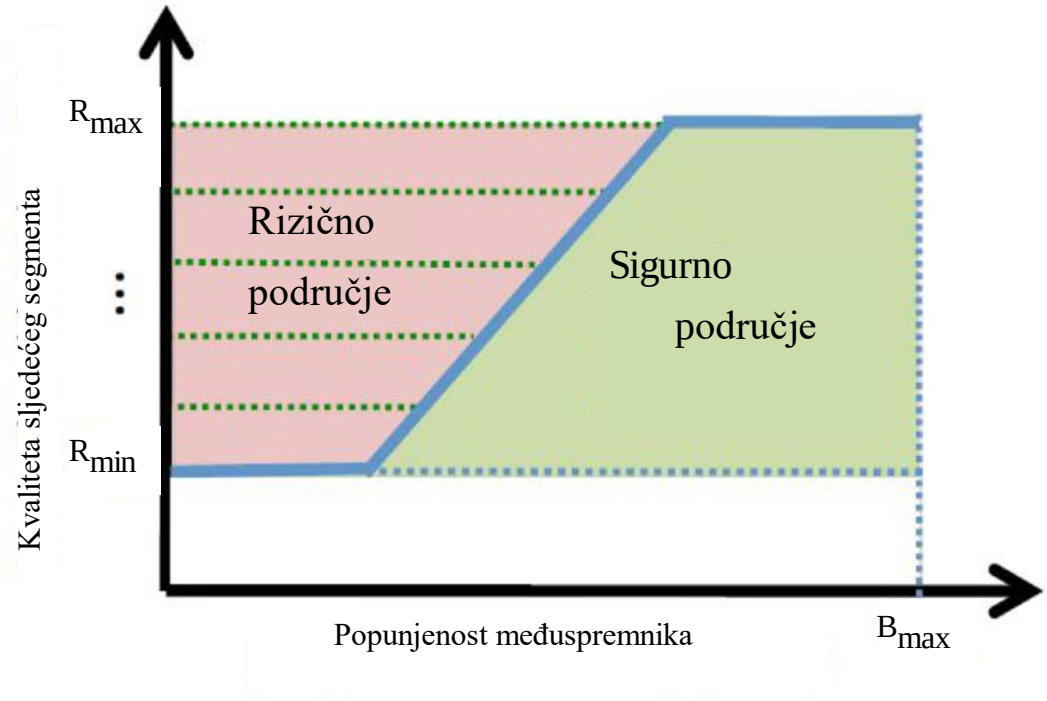


Sl 3.4. Međuspremnik za reprodukciju video zapisa [10]

Slika 3.4. prikazuje dinamiku međuspremnika za reprodukciju na klijentskoj strani. Popunjenost međuspremnika promatra se u sekundama video zapisa. Svake sekunde, jedna sekunda video zapisa uklanja se iz međuspremnika i reproducira korisniku (brzina pražnjenja), dakle međuspremnik se prazni konstantnom brzinom. Klijent zahtjeva segmente video zapisa sa servera koji su fiksne duljine trajanja, ali promjenjive veličine u bajtovima (što je veća kvaliteta segmenta to je veća veličina u bajtovima) te njima popunjava međuspremnik (brzina punjenja).

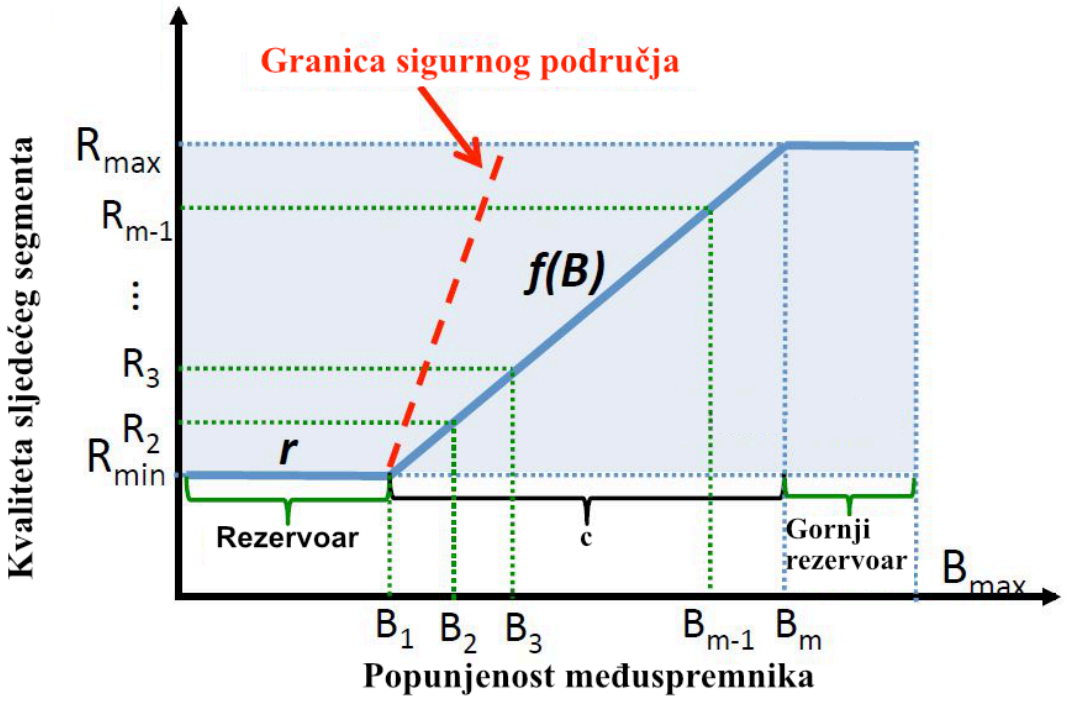
Ako adaptacijski algoritam pogrešno procijeni i dohvaća segmente veće kvalitete *R(t)* od mrežnog kapaciteta *C(t)*, tada se međuspremnik popunjava brzinom $\frac{C(t)}{R(t)} < 1$, tj. zauzeće međuspremnika se smanjuje. Privremeno takva situacija je najčešće održiva jer se međuspremnik koristi upravo radi toga da spriječi prekid reprodukcije u situaciji poput spomenute, međutim ako takav scenarij potraje, međuspremnik se može isprazniti i uzrokovati najgori efekt na korisničko iskustvo, a to je blokiranje reprodukcije.

Budući da je BBA algoritam temeljen na popunjenosti međuspremnika, osnovna ideja je da se prilagodba kvalitete video zapisa obavlja konzervativno kada je međuspremnik u opasnosti od potpunog pražnjenja, a kada je blizu popunjenosti promjenu je moguće obaviti agresivnije. Ako je propusnost mreže odnosno kapacitet veći od brzine prijenosa video zapisa najmanje kvalitete, $C(t) > R_{min}, \forall t > 0$, tada nikad ne mora doći do blokiranja reprodukcije uslijed pražnjenja međuspremnika. Najjednostavniji je to omogućiti odabirom najniže kvalitete R_{min} kada je međuspremnik u rizičnom stanju pražnjenja, a R_{max} kada je blizu maksimalne popunjenosti. Na slici 3.5. prikazan je opisani postupak. Područje između $[0, B_{max}]$ na apscisi i $[R_{min}, R_{max}]$ na ordinati predstavlja sigurno područje i svaka funkcija $f(B)$ unutar tog područja omogućava preslikavanje popunjenosti međuspremnika na kvalitetu sljedećeg segmenta. Međutim prikaz na slici 3.4. ispravan je pod pretpostavkom da su segmenti video zapisa infinitezimalni tako da je moguća kontinuirana promjena kvalitete.



Sl. 3.4. Prilagodba kvalitete u ovisnosti o popunjenosti međuspremnika [10]

U realnim uvjetima veličine segmenata su konačne (npr. 4 sekunde), a segment se smješta u međuspremnik tek nakon završetka preuzimanja. Kako bi se izbjegli prekidi reprodukcije u međuspremniku je potreban minimalno jedan dostupan segment u svakom trenutku. Zbog toga se preslikavanje pomiče u desno čime nastaje dodatni rezervoar označen s r na slici 3.5. Kada se u međuspremniku popunjava rezervoar $0 \leq B \leq r$, kvaliteta sljedećeg segmenta je R_{min}. U trenutku kada se rezervoar popuni kvaliteta se povećava prema funkciji $f(B)$ odnosno prema linearnoj funkciji koja povećava kvalitetu sukladno povećanju popunjenosti međuspremnika. Također je omogućen odabir maksimalne kvalitete R_{max} prije dostizanja maksimalne popunjenosti međuspremnika B_{max} jer zbog konačne veličine segmenata međuspremnik ne ostaje na vrijednosti B_{max} čak ni kada je kapacitet $C(t) \geq R_{max}$. Područje između rezervoara i točke u kojoj se prvi puta dostiže R_{max} je područje preslikavanja i označeno je s c, a područje nakon c naziva se gornji rezervoar.



Sl. 3.5. Preslikavanje popunjenosti međuspremnika na odabir kvalitete sljedećeg segmenta[10]

Budući da odabir kvalitete sljedećeg segmenta nije moguće promijeniti usred preuzimanja segmenta nego tek kada potpuni segment pristigne, ako u procesu preuzimanja brzina preuzimanja naglo padne, međuspremnik se može isprazniti prije nego se kvaliteta smanji. Zbog toga je popunjenost međuspremnika potrebno održavati iznad područja rezervoara kako bi bilo dovoljno

prostora za ublažavanje varijacija uzrokovanih promjenama u mreži i konačnim veličinama segmenata. Stoga, funkcija $f(B)$ se nalazi u sigurnom području ako uvijek odabire segment koji će se preuzeti prije nego popunjenost međuspremnika padne ispod r, kada je $C(t) \geq R_{min}$ za svaki t. Dakle, $\frac{f(B)}{R_{min}} \leq (B - r)$, u suprotnom $f(B)$ je u rizičnom području.

3.2.1. Dijagram toka i pseudo kod

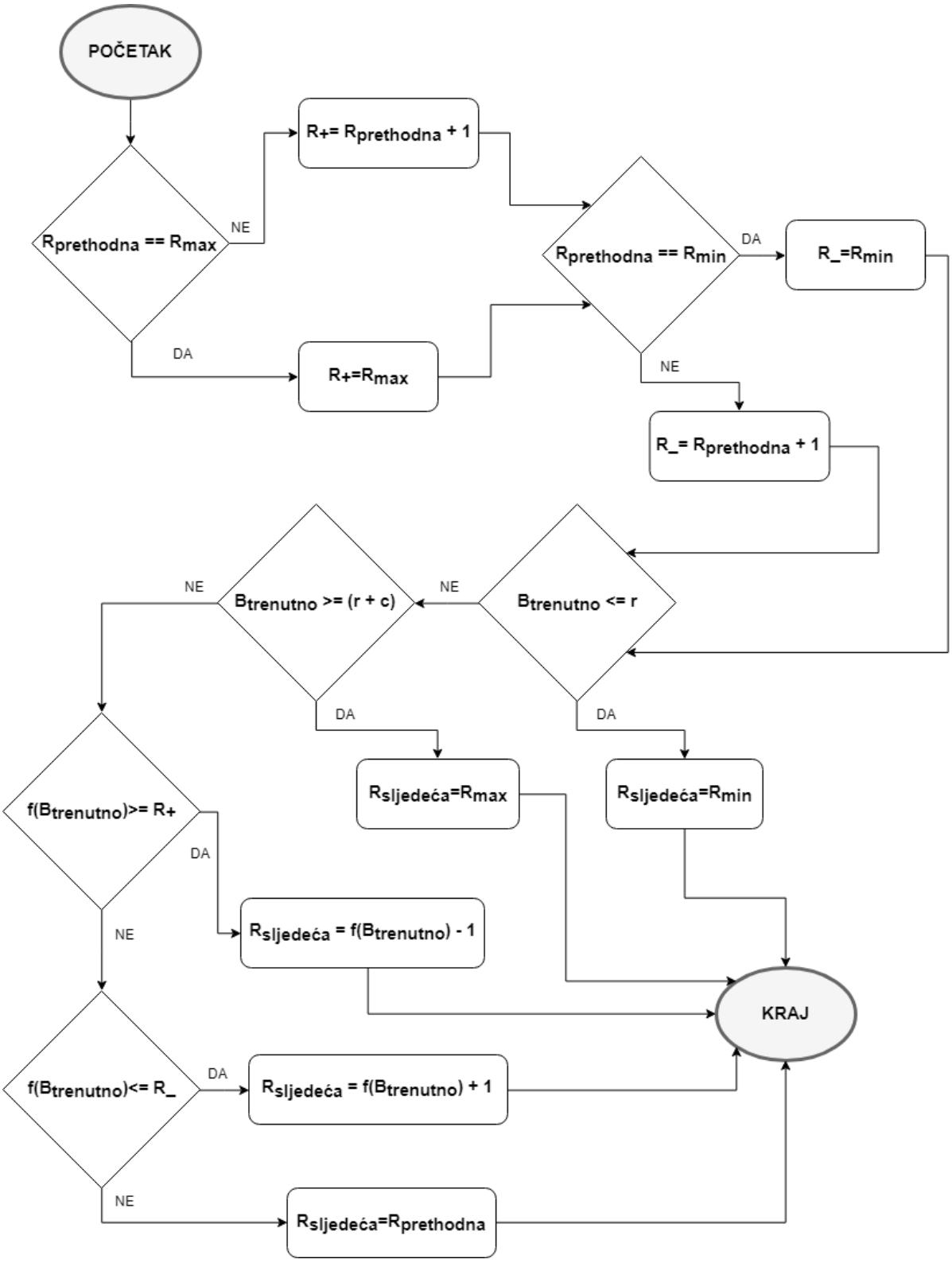
Razine kvaliteta video zapisa su diskretne vrijednosti, a opisana funkcija preslikavanja sa slike 3.5., linearna funkcija pa se zbog toga koristi BBA algoritam opisan u obliku dijagrama toka i pseudo koda na slikama 3.6. i 3.7. Osnovna ideja algoritma je da se ne mijenja trenutna kvaliteta dok funkcija preslikavanja $f(B_{trenutno})$ ne prijeđe sljedeću veću (R_+) ili manju (R_-) diskretnu razinu kvalitete. Kada se dostigne vrijednost barijera R_+ ili R_- kvaliteta se povećava odnosno smanjuje na novu diskretnu vrijednost predloženu funkcijom preslikavanja.

S obzirom na ulazne parametre na početku su postavljene barijere R_+ i R_-. Ako je prethodna kvaliteta maksimalna R_{max} ili minimalna R_{min}, gornja odnosno donja granica zadržavaju maksimalnu odnosno minimalnu vrijednost. Ako gornja granica nije maksimalna, postavlja se na jednu kvalitetu iznad prethodne. Isto tako ako donja granica nije minimalna, postavlja se na jednu kvalitetu ispod prethodne.

Nakon postavljanja barijera, promatra se popunjenost međuspremnika $B_{trenutno}$. Ako je popunjenost međuspremnika manja od vrijednosti rezervoara r preuzima se segment najmanje kvalitete. Ako je popunjenost međuspremnika veća od područja preslikavanja tj. $B_{trenutno} \geq (r + c)$, preuzima se segment najveće kvalitete. Kvaliteta R_{max} prvi puta se postavlja kada međuspremnik dostigne vrijednost B_m, odnosno $(r + c)$.

Ako prolaskom kroz prethodna dva uvjeta ni jedan nije zadovoljen, provjerava se je li razina kvalitete predložena linearnom funkcijom preslikavanja $f(B_{trenutno})$ veća ili jednaka od barijere R_+ odnosno manja ili jednaka od barijere R_-. Ako je, kvaliteta sljedećeg segmenta postavlja se na prvu manju odnosno veću razinu kvalitete predloženu spomenutom funkcijom $f(B_{trenutno})$. Ako i dalje ni jedan uvjet nije zadovoljen, kvaliteta sljedećeg segmenta ne mijenja se u odnosnu na prethodnu.

Kod za BBA algoritam u programskom jeziku C++ dan je u prilogu II.



Sl. 3.6. Dijagram toka BBA adaptacijskog algoritma

ako $R_{prethodna} = R_{max}$ **onda**

$R_+ = R_{max}$

inače

$R_+ = R_{prehodna} + 1$

ako $R_{prethodna} = R_{min}$ **onda**

$R_- = R_{min}$

inače

$R_- = R_{prehodna} - 1$

ako $B_{trenutno} \leq r$ **onda**

$R_{sljedeća} = R_{min}$

inače ako $B_{trenutno} \geq (r + c)$ **onda**

$R_{sljedeća} = R_{max}$

inače ako $f(B_{trenutno}) \geq R_+$ **onda**

$R_{sljedeća} = f(B_{trenutno}) - 1$

inače ako $f(B_{trenutno}) \leq R_-$ **onda**

$R_{sljedeća} = f(B_{trenutno}) + 1$

inače

$R_{sljedeća} = R_{prethodna}$

Sl. 3.6. Pseudo kod BBA adaptacijskog algoritma

Parametri koje koristi BBA algoritam su sljedeći:

- $R_{prethodna}$ – razina kvalitete posljednjeg preuzetog segmenta [b/s]
- $R_{sljedeća}$ – razina kvalitete sljedećeg segmenta za preuzimanje [b/s]
- R_{max} – maksimalna razina kvalitete video zapisa [b/s]
- R_{min} – minimalna razina kvalitete video zapisa [b/s]
- R_+ – gornja barijera [b/s]
- R_- – donja barijera [b/s]
- $B_{trenutno}$ – trenutna popunjenost međuspremnika [s]
- r – rezervoar [s]
- c – područje preslikavanja [s]
- $f(B_{trenutno})$ – funkcija preslikavanja trenutne popunjenosti međuspremnika na odabir diskretne vrijednosti kvalitete sljedećeg segmenta

3.3. **QAAD algoritam**

QAAD (*Quality of experience QoE-enhanced Adaptation Algorithm over DASH*) adaptacijski algoritam omogućava strujanje video sadržaja uz očuvanje minimalne razine međuspremnika čime se postiže poboljšanje korisničkog iskustva minimiziranjem promjena kvalitete prikaza tijekom reprodukcije. Za razliku od prethodno opisanog QDASH algoritma koji smanjuje kvalitetu u dva koraka korištenjem među-razine, QAAD dodatno ublažava smanjenje kvalitete kako bi se broj promjena kvalitete uzastopnih segmenata sveo na minimum. Dakle osnovna ideja QAAD algoritma je pružanje ujednačene kvalitete prikaza video zapisa čak i u uvjetima učestalih fluktuacija mrežne propusnosti.

QAAD koristi mjerenje propusnosti i trenutnu popunjenost međuspremnika kao ulazne parametre. Prvo se mjeri brzina preuzimanja te nakon toga u dijelu za izbor kvalitete u trenutku t, odabire se kvaliteta sljedećeg segmenta za preuzimanje $l_{sljedeća}$, ovisno o izmjerenoj brzini $bw_{procijenjeno}$ i trenutnoj popunjenosti međuspremnika $B(t)$. Najveću kvalitetu čija brzina prijenosa video zapisa ne prelazi izmjerenu brzinu predstavlja $l_{podržana}$, tj. $l_{podržana} = max(b(l) \leq bw_{procijenjeno})$.

U adaptivnom strujanju multimedijskog sadržaja postoji više načina za računanje mrežne propusnosti. Autori u [11] smatraju da periodičko mjerenje propusnosti korištenjem fiksnog intervala vremena omogućava bolje rezultate od mjerenja temeljenih na vremenu obilaska RTT (engl. *round trip time*) i mjerenja temeljenih na brzinama preuzimanja pojedinih segmenata. Međutim zbog izvedbe DASH klijenta u ovom diplomskom radu za sve algoritme propusnost bw_{uzorak}, se promatra za preuzimanje prethodnog segmenta, na način da se količina preuzetih bitova dijeli s vremenom potrebnim za preuzimanje tog broja bitova prema formuli (3-1). Dobivena propusnost ublažava se primjenom težinskog pomičnog prosjeka (engl. weighted moving average, WMA) na sljedeći način

$$bw_{procijenjeno} = \omega * bw_{procijenjeno} + (1 - \omega) * bw_{uzorak} \tag{3-2}$$

Gdje je $bw_{procijenjeno}$ prethodno izračunata propusnost, a ω težinski faktor čija je vrijednost uvijek u rasponu $0 \leq \omega \leq 1$.

3.3.1. Dijagram toka i pseudo kod

Kod izbora kvalitete primjenom QAAD algoritma postoje tri scenarija koja se razmatraju prikazana dijagramom toka i pseudo kodom na slikama 3.7. i 3.8., a to su:

1. Podržana kvaliteta je jednaka trenutnoj kvaliteti $l_{podržana} == l_{trenutna}$
2. Podržana kvaliteta je veća od trenutne kvalitete $l_{podržana} > l_{trenutna}$
3. Podržana kvaliteta je manja od trenutne kvalitete $l_{podržana} < l_{trenutna}$

Za prvi scenarij, promjena kvalitete nije potrebna s obzirom da je podržana kvaliteta jednaka trenutnoj kvaliteti odnosno kvaliteti posljednjeg preuzetog segmenta. Stoga, sljedeća kvaliteta zadržava trenutnu, tj. $l_{sljedeća} = l_{trenutna}$.

U drugom scenariju moguće je povećanje kvalitete, ali potrebno je provjeriti stanje međuspremnika kako bi se učestale promjene kvalitete svele na minimum te spriječili nepredviđeni prekidi reprodukcije uslijed pražnjenja međuspremnika. Dakle, kvaliteta se povećava postepeno za jedan, tj. $l_{sljedeća} = l_{trenutna} + 1$, ako je popunjenost međuspreminka veća od unaprijed definirane marginalne veličine međuspremnika μ. U suprotnom, sljedeća kvaliteta se ne mijenja s obzirom na prethodnu tj. $l_{sljedeća} = l_{trenutna}$. Na ovaj način u svakom trenutku je osigurana marginalna vrijednost međuspremnika, što se može iskoristiti u slučaju kada je mrežna propusnost kritično niska.

Kada trenutna mrežna propusnost ne podržava trenutnu kvalitetu, neophodno je smanjenje kvalitete. Međutim, agresivno smanjenje kvalitete za više od dvije razine odjednom rezultiralo bi značajnom degradacijom kvalitete korisničkog iskustva. Zbog toga se konzumacijom segmenata iz međuspremnika pokušava pronaći kvalitetu približno jednaku prethodnoj, a to se radi tako da se računa vrijeme za koje će se potrošiti svi segmenti iz međuspremnika i broj segmenata koji će se preuzeti za isto vrijeme.

Međuspremnik se uvijek prazni konstantno jer svakih τ sekundi troši jedan segment, a popunjavanje ovisi o mrežnoj propusnosti i kvaliteti segmenta koji se preuzima. Za kvalitetu l, međuspremnik se napuni jednim segmentom u $b(l) * \tau / bw_{procijenjeno}$ sekundi. Stoga, derivacija popunjenosti međuspremnika u vremenu t, $dB(t)/dt$ računa se kao:

$$\frac{dB(t)}{dt} = \frac{bw_{procijenjeno}}{b(l)} - 1. za\ b(l) > bw \tag{3-3}$$

Ako su svi segmenti iz međuspremnika potrošeni tijekom t_l, vrijedi:

$$B(t + t_l) = B(t) + \frac{dB(t)}{dt} * t_l = 0 \tag{3-4}$$

S obzirom na (3-3) i (3-4), preostalo vrijeme za potrošiti sve segmente u međuspremniku za razinu kvalitete l, t_l se računa kao:

$$t_l = \frac{B(t)}{1-\frac{bw_{procijenjeno}}{b(l)}}. \tag{3-5}$$

Međutim, trošenjem svih segmenata iz međuspremnika uzrokovao bi se blokiranje reprodukcije i narušila kvaliteta korisničkog iskustva pa je potrebno uvijek očuvati minimalnu popunjenost međuspremnika σ. Ako se u (3-4) nula zamijeni s σ, dobiva se preostalo vrijeme za potrošiti segmente iz međuspremnika uz očuvanje minimalne popunjenosti σ:

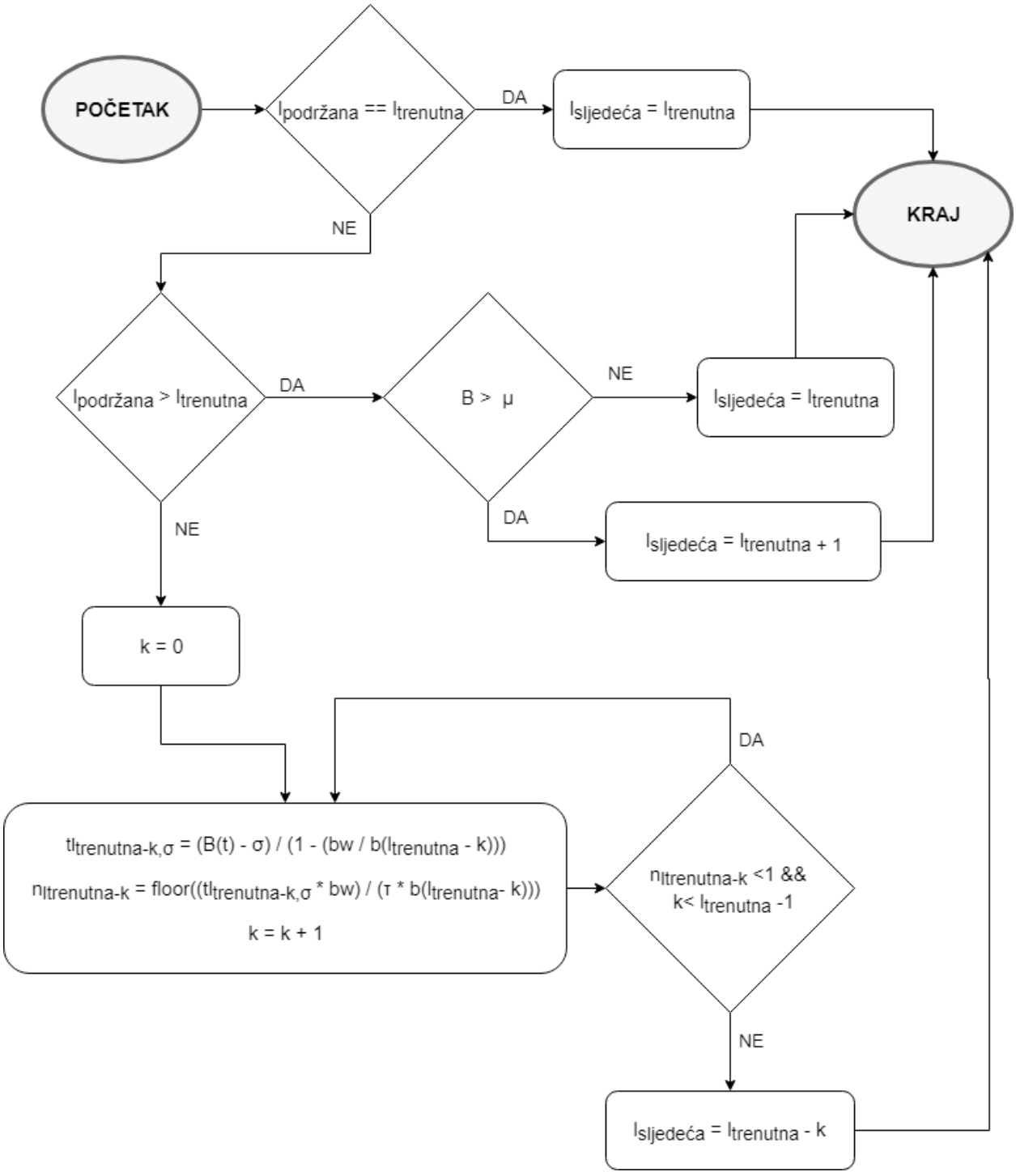
$$t_{l,\sigma} = \frac{B(t)-\sigma}{1-\frac{bw_{procijenjeno}}{b(l)}}. \tag{3-6}$$

Ako je očekivani broj preuzetih segmenata tijekom $t_{l,\sigma}$ manji od 1, odabir kvalitete l ne može očuvati minimalnu popunjenost međuspremnika pa se očekivani broj preuzetih segmenata računa kao:

$$n_l = \left\lfloor \frac{t_{l,\sigma} * bw_{procijenjeno}}{\tau * b(l)} \right\rfloor, \tag{3-7}$$

gdje je $\lfloor x \rfloor$ funkcija koja vraća najveći cijeli broj veći od x (engl. floor function). Ako je $n_l \geq 1$, navedena razina l, definira se kao podržana kvaliteta $l_{podržana}$. Iz dijagrama toka i pseudo koda vidljivo je da se maksimalna podržana kvaliteta manja od $l_{trenutna}$ pretražuje povećanjem iteratora k. Ako je kvaliteta $(l_{trenutna} - k)$ podržana tj. $n_{l_{trenutna}-k} \geq 1$, izvođenje petlje se prekida i smanjenje kvalitete je izvršeno tj. $l_{sljedeća} = l_{trenutna} - k$.

Kod za QAAD algoritam u programskom jeziku C++ dan je u prilogu III.



Sl. 3.7. Dijagram toka QAAD adaptacijskog algoritma

ako $l_{podržana} == l_{trenutna}$ **onda**

$l_{sljedeća} = l_{trenuntna}$

inače ako $l_{podržana} > l_{trenutna}$ **onda**

ako $B > \mu$ **onda**

$l_{sljedeća} = l_{trenutna} + 1$

inače

$l_{sljedeća} = l_{trenutna}$

inače ako $l_{podržana} < l_{trenutna}$ **onda**

$k = 0$

činiti

$t_{l_{trenutna-k},\sigma} = \frac{B(t)-\sigma}{1-\left(\frac{bw_{procijenjeno}}{b(l_{trenutna}-k)}\right)}$

$n_{l_{trenutna-k}} = \left\lfloor \frac{t_{l_{trenutna-k},\sigma} * bw_{procijenjeno}}{\tau * b(l_{trenutna}-k)} \right\rfloor$

$k = k + 1$

dok $n_{l_{trenutna-k}} < 1$ && $k < l_{trenutna} - 1$

$l_{sljedeća} = l_{trenutna} - k$

Sl. 3.8. Pseudo kod QAAD adaptacijskog algoritma

Parametri koje koristi algoritam QAAD su sljedeći:

- $l_{podržana}$ – razina kvalitete koju trenutna mrežna propusnost podržava [b/s]
- $l_{trenutna}$ – razina kvalitete posljednjeg preuzetog segmenta [b/s]
- $l_{sljedeća}$ – zahtijevana razina kvalitete video zapisa [b/s]
- B – popunjenost međuspremnika [s]
- μ – marginalna popunjenost međuspremnika [s]
- σ – minimalna popunjenost međuspremnika [s]
- $t_{l_{trenutna-k},\sigma}$ – preostalo vrijeme za trošenje segmenata iz međuspremnika uz očuvanje minimalne popunjenosti međuspremnika σ [s]
- $n_{l_{trenutna-k}}$ – očekivani broj preuzetih segmenata tijekom perioda $t_{l_{trenutna-k},\sigma}$

4. TESTIRANJE

Osnovna ideja testiranja je usporedba implementiranih adaptacijskih algoritama u promjenjivim mrežnim uvjetima i analiza odnosa između popunjenosti međuspremnika, mrežne propusnosti te kvalitete video zapisa. Za testiranje implementiranih algoritama korištena su tri video zapisa, dostupna na poslužitelju i podijeljena u segmente trajanja šest sekundi. Detaljni podaci o video zapisima korištenim za testiranje prikazani su u tablici 4.1.

Svi algoritmi testirani su pod istim uvjetima i za sve tri video sekvence. Za simulaciju promjenjivih mrežnih uvjeta korištena je skripta *Wondershaper* prikazana na slici 4.1. koja omogućava ograničavanje mrežne propusnosti na proizvoljno definiranu vrijednost. U tablici 4.2. prikazana su postavljena ograničenja za dva korištena testna scenarija. Postavljeno ograničenje se mijenja svakih 10 sekundi tijekom preuzimanja video zapisa i vrti se u petlji prema postavljenim vrijednostima do završetka preuzimanja. Za prvi testni scenarij propusnost se svakih 10 sekundi naizmjenično mijenja između 10 Mb/s i 1 Mb/s. Kod drugog scenarija propusnost je na početku postavljena na 1 Mb/s pa se svakih 10 sekundi mijenja redom na 3 Mb/s, 5 Mb/s, 7 Mb/s, 3 Mb/s te nakon toga ponovno na iste vrijednosti počevši s 1 Mb/s.

Tab. 4.2. Značajke testnih video zapisa

	BBB	ED	OFAM
Tip sadržaja	Animirani film	Animirani film	Dokumentarni film
Trajanje video zapisa [min]	9:56	10:54	7:33
Trajanje segmenta [s]	6	6	6
Dostupne rezolucije	320x240 480x360 854x480 1280x720 1920x1080	320x240 480x360 854x480 1280x720 1920x1080	320x240 480x360 854x480 1024x576
Brzina izmjene okvira [slika / s]	24	24	24
Koder	H.264	H.264	H.264
Razine kvalitete	20	20	19

Tab. 4.2. Testni scenariji za simulaciju promjenjivih mrežnih uvjeta

Propusnost [Mb/s]	1. scenarij	2. scenarij
	10	1
	1	3
		5
		7
		3

```
#!/bin/bash
cd ~/wondershaper
while :
do
    sudo ./wondershaper -c -a enp5s0
    sudo ./wondershaper -a enp5s0 -d 1000
    echo 1000
    sleep 10;
    sudo ./wondershaper -c -a enp5s0
    sudo ./wondershaper -a enp5s0 -d 2000
    echo 2000
    sleep 10;
    sudo ./wondershaper -c -a enp5s0
    sudo ./wondershaper -a enp5s0 -d 3000
    echo 3000
    sleep 10;
    sudo ./wondershaper -c -a enp5s0
    sudo ./wondershaper -a enp5s0 -d 2000
    echo 2000
    sleep 10;
done
```

Sl. 4.1. Skripta za ograničavanje brzine prijenosa video zapisa

Radi jednostavnije usporedbe rezultata za svaki implementirani algoritam gornja granica međuspreminka postavljena je na 48 sekundi odnosno osam segmenata trajanja šest sekundi. S ciljem izbjegavanja prelijevanja (engl. *overflow*), kada popunjenost međuspremnika prelazi 48 sekundi, u klijentskom programu omogućeno je pauziranje preuzimanja dok se međuspremnik ne isprazni do postavljene gornje granice.

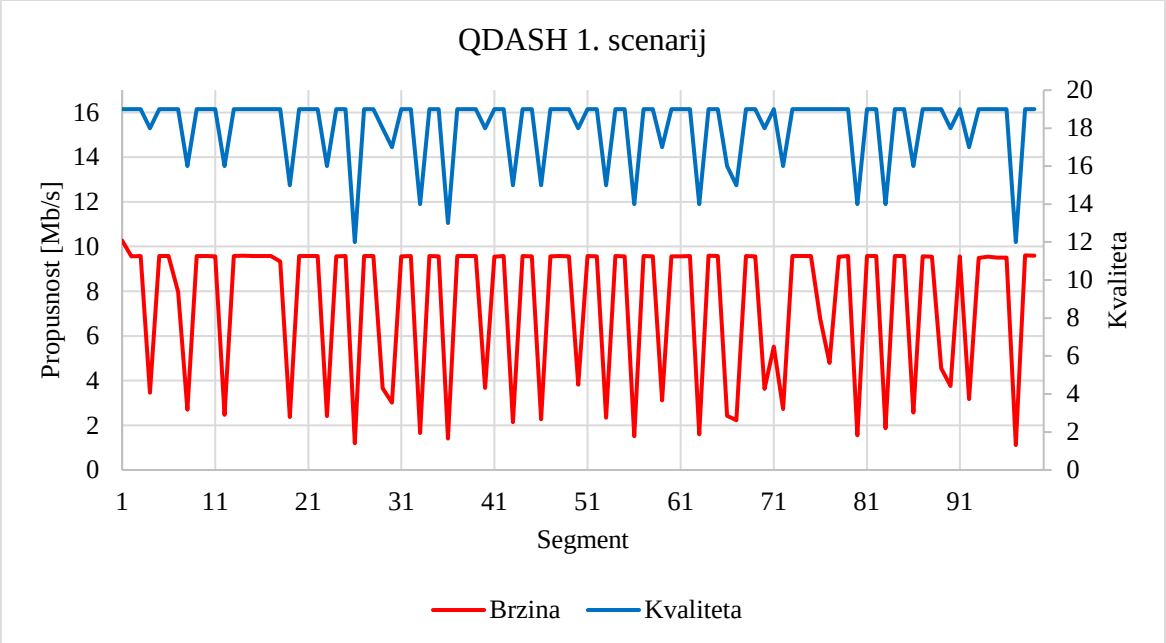
Za BBA algoritam postavljen je rezervoar veličine 12 sekundi odnosno dva segmenta, što se prilikom provedenog testiranja pokazalo dovoljnim za izbjegavanje prekida reprodukcije. Za razliku od autora [10] koji koriste rezervoar veličine 90 sekundi jer je algoritam testiran u stvarnim uvjetima na platformi Netflix, postavljeni rezervoar je značajno manji jer je testiran na relativno kratkim video zapisima maksimalnog trajanja do 11 minuta. Dakle, primjenom BBA algoritma uvijek se na početku preuzima dva segmenta najniže kvalitete čime je osigurana minimalna popunjenost međuspremnika. Gornji rezervoar postavljen je na 36 sekundi odnosno 75 % popunjenosti međuspremnika i zauzima također dva segmenta odnosno 12 sekundi.

Minimalna razina međuspremnika za QAAD algoritam postavljena je na devet sekundi, a marginalna razina na 30 sekundi. Za QDASH algoritam nema unaprijed postavljenih parametara.

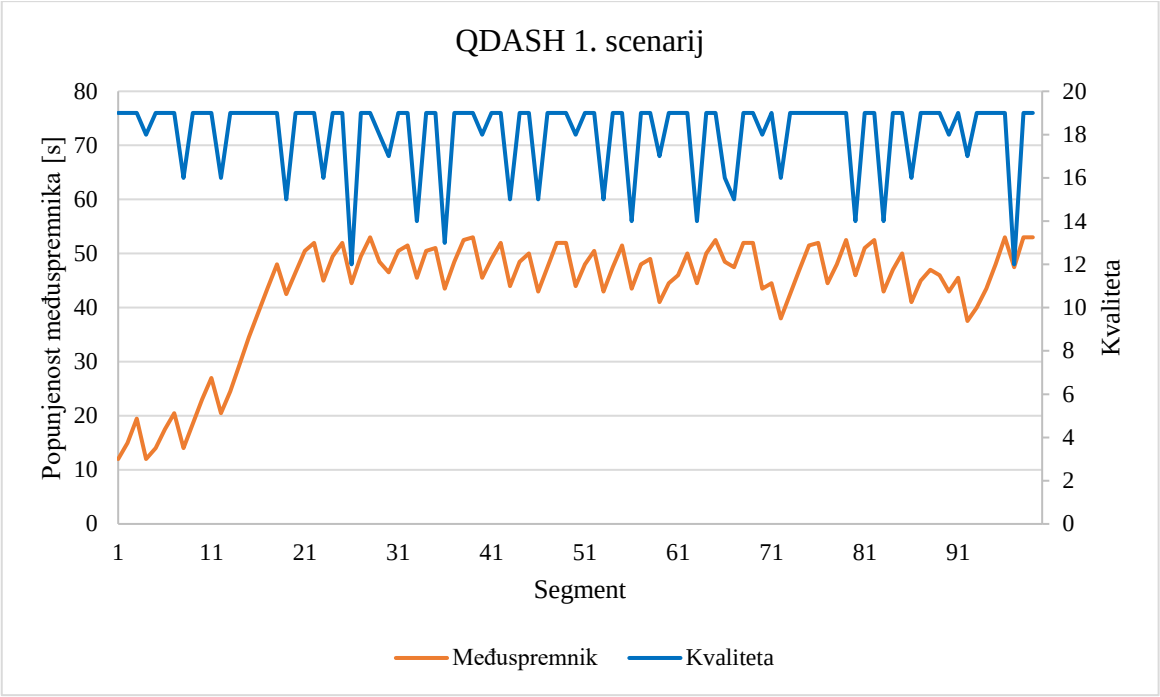
4.1. **Analiza rezultata**

Na slikama 4.3. – 4.. prikazani su rezultati testiranja za prvi testni scenarij na BBB video zapisu pomoću grafova koji prikazuju odnose između kvalitete i propusnosti te kvalitete i međuspremnika. Iako je propusnost za prvi testni scenarij naizmjenično ograničena na 1 Mb/s i 10 Mb/s vidljivo je da propusnost (crvena linija) ipak oscilira pa u nekim trenucima prelazi granicu od 10 Mb/s, ali ponekad se i ne smanjuje skroz do 1 Mb/s. Iako prvi testni scenarij ne prikazuje stanje mreže u realnim uvjetima, cilj ovakvog testa je prikaz ponašanja algoritama prilikom velikih oscilacija propusnosti i njihovu sposobnost adaptacije u ekstremnim uvjetima.

QDASH algoritam izravno prati propusnost budući da je temeljen isključivo na promatranju propusnosti pa zato unatoč korištenju među-razina dolazi do brojnih oscilacija kvalitete. Također je vidljivo da se ne koristi tzv. početna faza kojom bi se osigurala minimalna popunjenost međuspremnika nego je kvaliteta prvog segmenta izravno povezana s propusnošću u tom trenutku. Posljedica toga je da se međuspremnik puni duže, ali je krajnjim korisnicima omogućena bolja kvaliteta korisničkog iskustva budući da se i na početku reprodukcije isporučuje visoka kvaliteta.

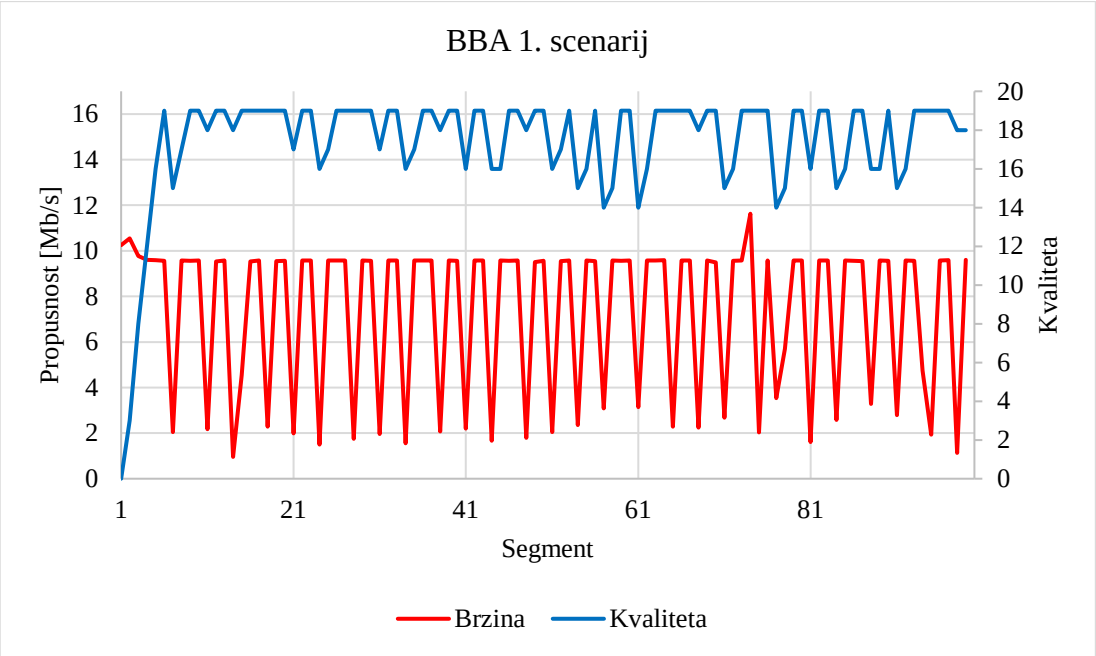


Sl. 4.3. Odnos kvalitete i brzine za 1. scenarij primjenom QDASH algoritma za video zapis BBB

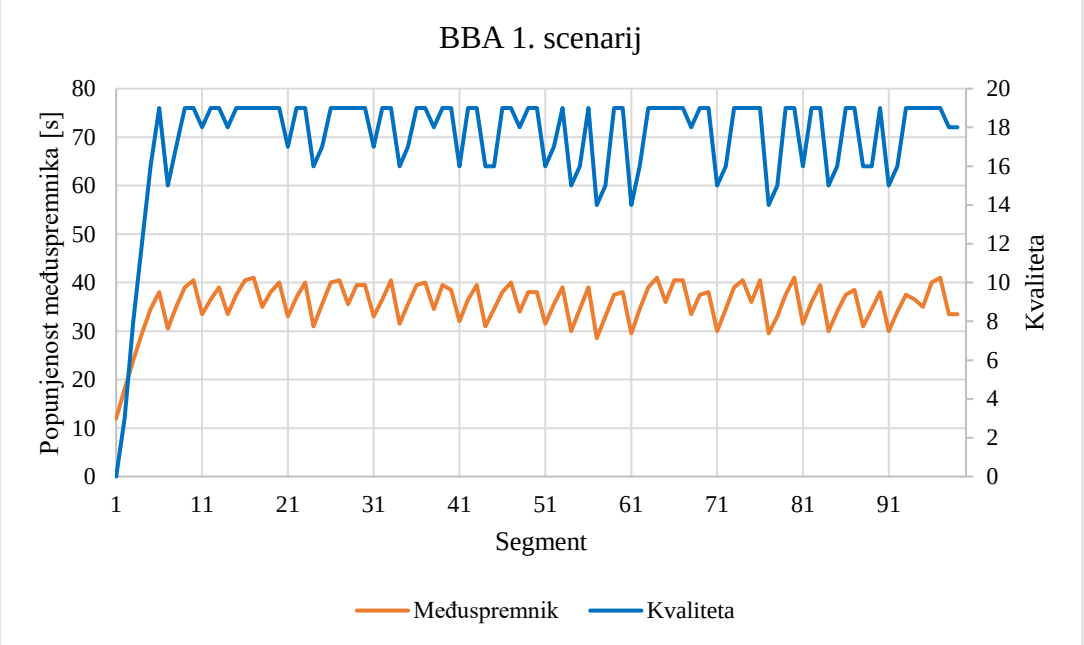


Sl. 4.4. Odnos kvalitete i međuspremnika za 1. scenarij primjenom QDASH algoritma za video zapis BBB

BBA algoritam za razliku od QDASH-a, temelji se na popunjenosti međuspremnika te koristi početnu fazu koja traje 12 sekundi. Dakle, prva dva segmenta najniže su kvalitete čime je kratkotrajno narušena kvaliteta korisničkog iskustva, međutim u tom kratkom periodu omogućeno je popunjavanje međuspremnika značajno većom brzinom od QDASH algoritma. Za QDASH algoritam međuspremnik se popunio tek preuzimanjem 18. segmenta dok se za BBA algoritam popunio preuzimanjem sedmog segmenta (slika 4.6.). Također vidljive su nešto manje oscilacije kvalitete (slika 4.5.), što doprinosi kvaliteti korisničkog iskustva i prosječno veća kvaliteta prikaza budući da osim u početnoj fazi, kvaliteta ne pada ispod 14. razine dok kod QDASH-a unatoč dovoljnoj popunjenosti međuspremnika, pada i do 10. razine.

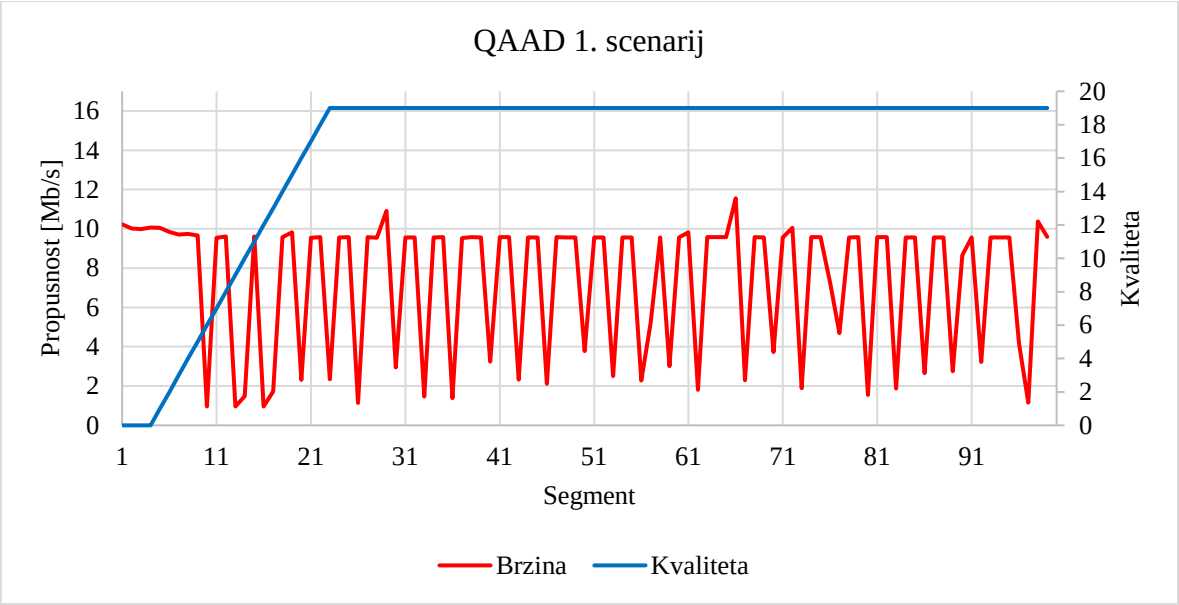


Sl. 4.5. Odnos kvalitete i brzine za 1. scenarij primjenom BBA algoritma za video zapis BBB

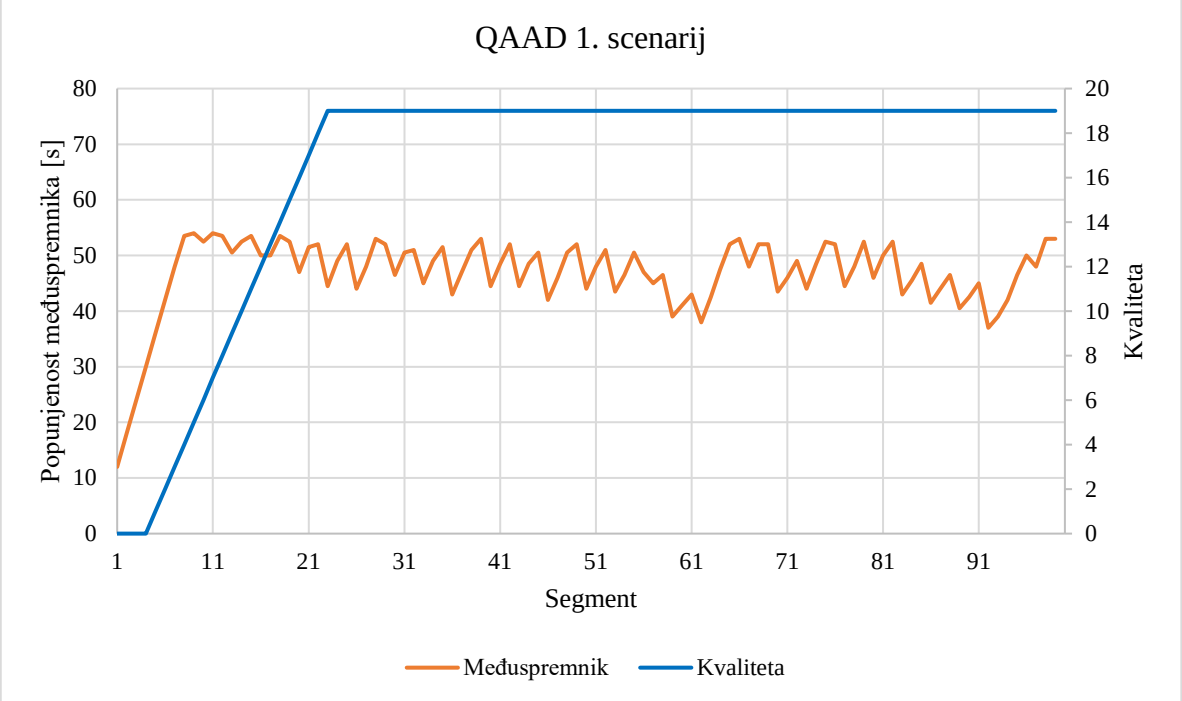


Sl. 4.6. Odnos kvalitete i međuspremnika za 1. scenarij primjenom BBA algoritma

QAAD algoritam za razliku od prethodnih QDASH i BBA obavlja povećanje kvalitete postupno korak po korak što rezultira značajno sporijim povećanjem kvalitete pa tako unatoč maksimalnoj popunjenosti međuspremnika kod preuzimanja osmog segmenta, kvaliteta dostiže maksimalnu razinu tek u trenutku preuzimanja 23. segmenta (slike 4.7. i 4.8.). Kompenzacija narušene kvalitete korisničkog iskustva u nešto dužoj početnoj fazi postignuta je zadržavanjem maksimalne kvalitete nakon početne faze, sve do kraja reprodukcije.

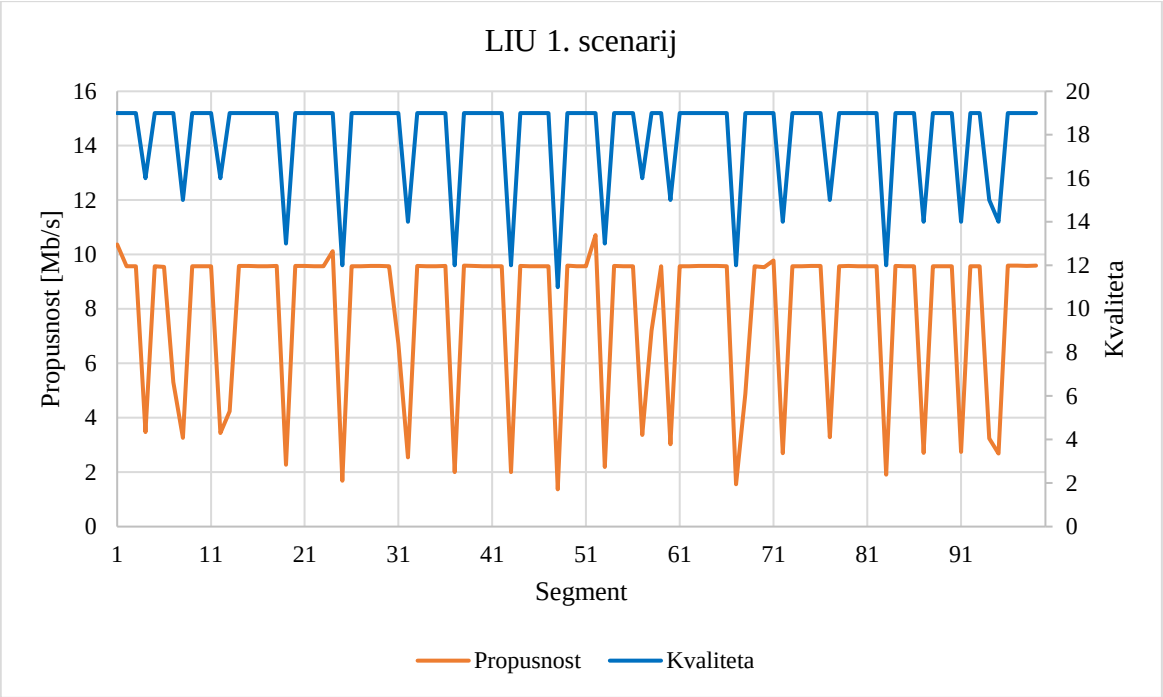


Sl. 4.7. Odnos kvalitete i brzine za 1. scenarij primjenom QAAD algoritma za video zapis BBB



Sl. 4.8. Odnos kvalitete i međuspremnika za 1. scenarij primjenom QAAD algoritma za video zapis BBB

Slika 4.9. prikazuje odnos kvalitete i propusnosti za LIU algoritam, a budući da LIU algoritam ne koristi međuspremnik nije prikazan graf ovisnosti međuspremnika o propusnosti.



Sl. 4.9. Odnos kvalitete i propusnosti za 1. scenarij primjenom LIU algoritma za video zapis BBB

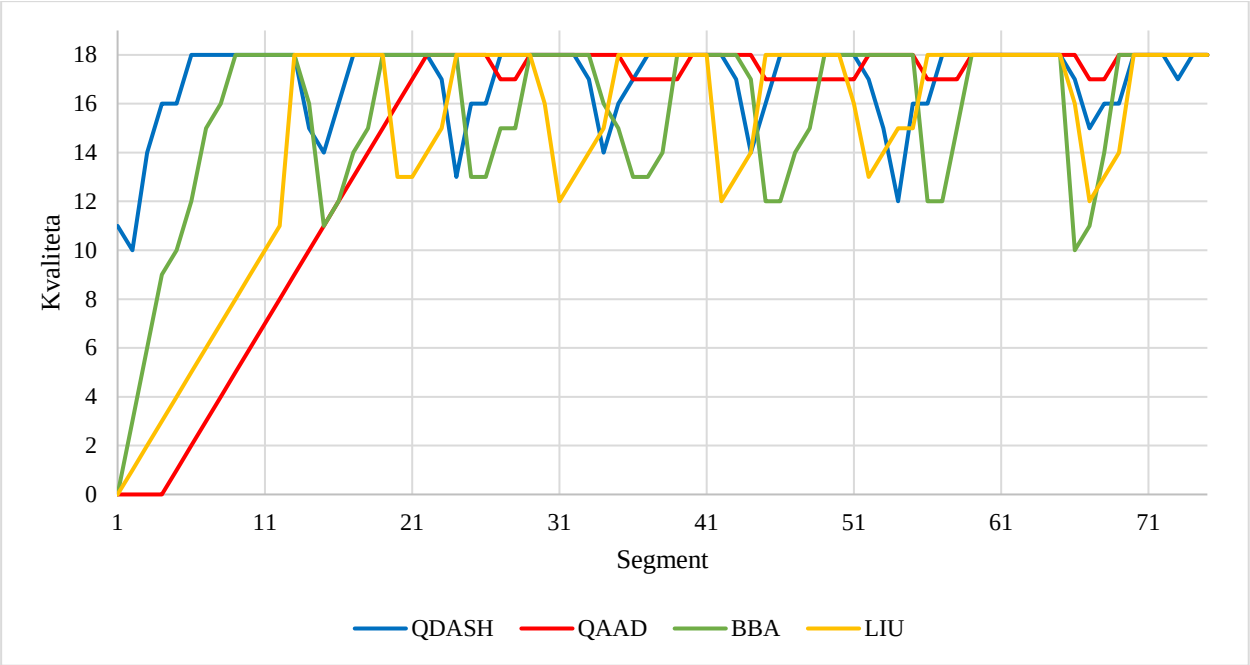
LIU algoritam najsličniji je QDASH algoritmu za navedeni testni scenarij jer ne koristi početnu fazu ako propusnost podržava maksimalnu kvalitetu. Dakle za ovaj testni scenarij od početka preuzima segmente maksimalno podržane kvalitete. Kao i QDASH temelji se isključivo na brzini pa kvaliteta izravno prati promjene propusnosti. Razlika u odnosu na QDASH je u među-razinama koje LIU ne koristi pa je za isti testni scenarij vidljiv pad kvalitete do 11. razine kvalitete što kod QDASH-a nije zabilježeno, nego je kao što je već spomenuto najveći pad do 12. razine kvalitete. Također, za razliku od svih prethodnih algoritama, za LIU graf prikazuje manje promjena kvalitete međutim to se događa zato što u osnovnoj verziji klijentskog programa nije omogućeno pauziranje preuzimanja kada se međuspremnik napuni pa zato preuzimanje video zapisa primjenom LIU algoritma traje kraće i rezultira grafom prikazanim na slici 4.9.

Slike 4.10. i 4.11. prikazuju skupne rezultate testiranja za drugi testni scenarij na dokumentarnom video zapisu OFAM, a ostali pojedinačni rezultati za sve video zapise u oba testna scenarija prikazani su u prilozima IV, V i VI. Na slici 4.10. prikazane su promjene kvalitete tijekom preuzimanja video zapisa za sva četiri testirana algoritma, dok je na slici 4.11. koja prikazuje popunjenost međuspremnika tijekom preuzimanja, izostao prikaz za LIU algoritam jer ne koristi međuspremnik. U tablici 4.1. prikazane su promjene propusnosti za navedeni testni scenarij pa iste nisu prikazane na slici 4.10. zbog preglednosti. Za razliku od prethodnog testiranja,

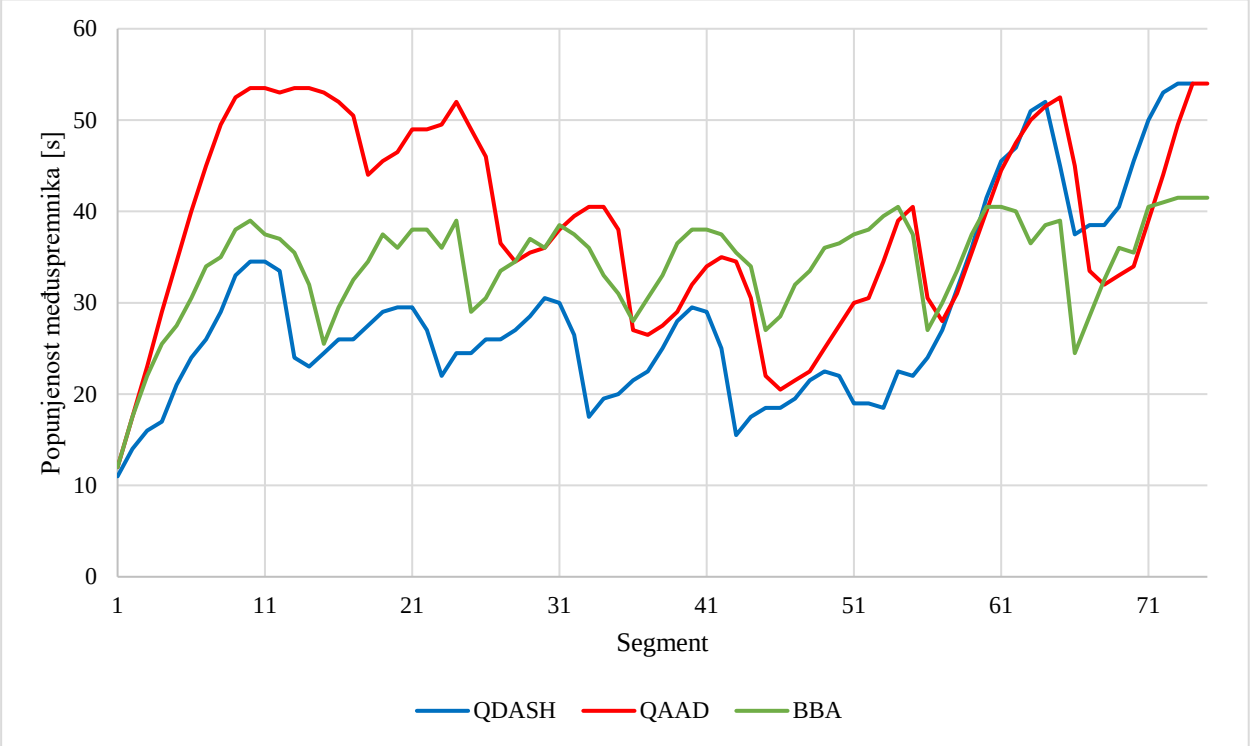
ovakav test realnije simulira stvarne uvjete jer se ovakve oscilacije mogu pojaviti u prosječnom kućanstvu u kojem više osoba koristi internet istovremeno.

Za razliku od prethodnog testiranja, jedino QDASH ne koristi početnu fazu te od početka preuzima segmente značajno više kvalitete od ostalih algoritama. LIU prilikom početka ne detektira maksimalnu kvalitetu pa se povećanje kvalitete obavlja postupno te je za ovaj testni scenarij sličniji QAAD algoritmu u početnoj fazi, međutim nakon početne faze ima najveće oscilacije kvalitete. BBA algoritam najbrže završava početnu fazu pa unatoč sličnim promjenama kvalitete nakon početne faze poput LIU algoritma, omogućava prosječno bolje korisničko iskustvo. Početna faza QAAD algoritma traje predugo u odnosu na ostale algoritme (približno dvije minute) pa iako nakon početne faze ima minimalne promjene kvalitete, ovakav konzervativni pristup u početnoj fazi definitivno narušava kvalitetu korisničkog iskustva cijelog video zapisa. Na slici 4.10. kod segmenata 30, 40 i 65 (približno jer promjene propusnosti nisu identične za svaki algoritam što je vidljivo na slikama 4.3.-4.9.) vidljiva je razlika između QDASH-a koji koristi među-razine kvalitete i LIU koji agresivno smanjuje kvalitetu. QDASH u tim trenucima unatoč sličnoj promjeni propusnosti zadržava kvalitetu dvije i više razina iznad LIU algoritma.

Na slici 4.11. vidljivo je da QAAD za vrijeme početne faze puno više popunjava međuspremnik od QDASH i BBA, međutim kasnije se popunjenost međuspremnika smanji zbog preuzimanja segmenata visoke kvalitete. BBA zadržava stabilnu razinu popunjenosti tijekom cijelog preuzimanja, a QDASH iako ima najmanju popunjenost, zadržava visoku razinu kvalitete korisničkog iskustva te budući da ni u jednom trenutku nije došlo do pražnjenja međuspremnika, ima najbolje rezultate u ovom testnom scenariju.



Sl. 4.10. Promjene kvalitete u 2. testnom scenariju za video zapis OFAM



Sl. 4.11. Popunjenost međuspremnika u 2. testnom scenariju za video zapis OFAM

5. ZAKLJUČAK

MPEG-DASH standard omogućava strujanje video sadržaja promjenjive kvalitete preko postojeće HTTP web infrastrukture. Trenutno se primjenjuje na servisima YouTube i Netflix što dovoljno govori o popularnosti i potencijalu MPEG-DASH-a, a osnovni cilj ovakvog standarda je omogućavanje visoke kvalitete korisničkog iskustva korisnicima u vremenu konstantnog povećanja multimedijskih sadržaja na internetu i želje za konzumacijom istih u realnom vremenu.

U okviru ovog diplomskog rada u programskim jezicima C i C++ implementirana su tri adaptacijska algoritma, QDASH, BBA i QAAD, kao moduli MPEG DASH klijentskog programa za adaptivno strujanje video sadržaja. U radu su pojedinačno opisani algoritmi i njihov način rada, prikazan je dijagram toka i pseudo kod za svaki algoritam ali ne i cijeli programski kod u C i C++ programskim jezicima jer je klijentski program zaštićen vlasničkom komercijalnom licencom.

Implementirani algoritmi QDASH, BBA i QAAD, kao i osnovni LIU algoritam koji je već bio implementiran unutar korištenog klijentskog programa, testirani su na dva testna scenarija promjenjive propusnosti mreže. Analizirani su odnosi između kvalitete, propusnosti i popunjenosti međuspremnika za svaki pojedini algoritam te je napravljena usporedba algoritama.

Budući da ne koristi početnu fazu, QDASH od početka omogućava strujanje sadržaja visoke kvalitete, a tijekom cijelog prijenosa oscilira manje od LIU algoritma. Iako omogućava bolje korisničko iskustvo, QDASH ne kontrolira popunjenost međuspremnika, što bi moglo u implementacijama s manjim međuspremnikom i uvjetima većih oscilacija propusnosti dovesti do potpunog pražnjenja međuspremnika čime bi se značajno smanjilo korisničko iskustvo.

BBA zadržava optimalnu razinu popunjenosti međuspremnika tijekom cijelog prijenosa, ali u početnoj fazi kratkotrajno narušava kvalitetu korisničkog iskustva jer preuzima segmente niže kvalitete.

QAAD algoritam pruža najbolje korisničko iskustvo s najmanjim oscilacija nakon što prođe početnu fazu, ali početna faza traje predugo za kraće video zapise poput video zapisa korištenih u ovom radu.

Primijenjeni adaptacijski algoritmi imaju svoje prednosti i nedostatke, a za bolju procjenu kvalitete korisničkog iskustva, uz navedenu analizu bilo bi potrebno provesti i subjektivnu ocjenu kvalitete, što zbog složenosti i dugotrajnosti ovakvih ispitivanja nije u okviru postavljenog zadatka ovog diplomskog rada.

LITERATURA

[1] https://ox4zindgwb3p1qdp2lznn7zb-wpengine.netdna-ssl.com/wp-content/uploads/2016/04/standards-1024x413.png [pristup ostvaren: 01.06.2018]

[2] https://ox4zindgwb3p1qdp2lznn7zb-wpengine.netdna-ssl.com/wp-content/uploads/2016/04/adaptive-streaming-1024x350.png [pristup ostvaren: 01.06.2018]

[3] Anthony Vetro. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. IEEE Computer Society, 2011.

[4] Ricky K. P. Mok, Xiapu Luo, Edmond W. W. Chan and Rocky K. C. Chang. QDASH: A QoE-aware DASH system. 2012.

[5] S. Akhshabi, A. Begen, and C. Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over HTTP. In Proc. ACM MMSys, 2011.

[6] N. Cranley, P. Perry, and L. Murphy. User perception of adapting video quality. Int. Journal of human-computer studies, 64(8):637 – 647, 2006.

[7] M. Pinson and S. Wolf. Comparing subjective video quality testing methodologies. Visual Communications and Image Processing, 5150(1):573 – 582, 2003.

[8] M. Zink, O. K¨unzel, J. Schmitt, and R. Steinmetz. Subjective impression of variations in layer encoded videos. In Proc. IWQoS, 2003.

[9] Hari KALVA, Velibor ADZIC, and Borko FURHT. Comparing MPEG AVC and SVC for Adaptive HTTP Streaming. IEEE 2012.

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.713.2845&rep=rep1&type=pdf

[10] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, Mark Watson Stanford University, Netflix. A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. Stanford University, Netflix 2014.

[11] Dongeun Suh, Insun Jang, and Sangheon Pack, School of Electrical and Engineering

Korea University. QoE-enhanced Adaptation Algorithm over DASH for Multimedia Streaming. IEEE 2014.

SAŽETAK

U ovom diplomskom radu opisane su teorijske osnove za razumijevanje adaptivnog strujanja video zapisa korištenjem MPEG-DASH standarda. Opisan je postupak standardizacije MPEG-DASH-a, princip rada i interakcija između klijenta i poslužitelja. Detaljno je objašnjena uloga i struktura MPD datoteke s opisima pojedinog elementa unutar MPD-a. Unutar klijentskog programa uz prethodno implementirani LIU adaptacijski algoritam, implementirani su QDASH, BBA i QAAD adaptacijski algoritmi koji u obzir uzimaju kvalitetu korisničkog iskustva. Prikazani su dijagram toka, pseudo kod i način rada pojedinog algoritma. U svrhu usporedbe i provjere uspješnosti implementacije algoritama provedena su testiranja na opisanim testnim scenarijima te su analizirani rezultati testiranja prikazani linijskim grafovima.

Ključne riječi: MPEG-DASH, MPD, adaptacijski algoritam, strujanje video zapisa, segment, klijentski program, testni scenarij

ABSTRACT

In this master thesis theoretical basis are given for understanding adaptive video streaming using the MPEG-DASH standard. MPEG-DASH standardization process, how it works and the interaction between client and server are described. Detailed description of the role and structure of the MPD file is given with descriptions of each element within the MPD. Within the client program with the previously implemented LIU adaptation algorithm, QDASH, BBA and QAAD adaptive algorithms were implemented with a goal of improving perceived quality of experience for users. The flow chart and pseudo code are shown and how a particular algorithm works is described. For the purposes of comparison and success verification of the algorithm implementations, testing was carried out on the described test cases and the test results were analyzed using line graphs.

Keywords: MPEG-DASH, MPD, adaptation algorithm, video streaming, segment, client program, test case

ŽIVOTOPIS

Filip Vranješ rođen je 29.08.1994.g. u Osijeku. Osnovnu školu završio je u Višnjevcu nakon koje upisuje Elektrotehničku i prometnu školu u Osijeku, smjer tehničar za računalstvo. Završava srednju školu 2013.g. te iste godine upisuje sveučilišni preddiplomski studij računarstva na Elektrotehničkom fakultetu u Osijeku. Do 2016. godine aktivno nastupao u prvoj hrvatskoj odbojkaškoj ligi za klubove „Mok Mursa Osijek“ i „Hok Borovo Vukovar“ s kojima osvaja kup hrvatske i drugo mjesto prvenstva hrvatske te sudjeluje u europskim natjecanjima Confédération Européenne de Volleyball (CEV) kupu i CEV Volleyball Challenge Cupu te je nakon toga kategoriziran s III. razinom kategorizacije sportaša. Na sveučilišnom prvenstvu osvaja dvije zlatne i jednu srebrnu medalju te dva puta dobiva nagradu za najboljeg igrača sveučilišta. Također nastupa za odbojkašku reprezentaciju sveučilišta na državnom prvenstvu gdje osvaja drugo mjesto. Godine 2016. upisuje diplomski studij smjer informacijske i podatkovne znanosti te iste godine postaje stipendist tvrtke RT-RK. Poznaje engleski jezik u govoru i pismu.

PRILOG I – Programski kod QDASH algoritma

```
1. void SegmentManager::adapt(unsigned int* n, int speed, int stream_type, DASH_C
   ontext* dash_context) {
2.     unsigned int i = 0;
3.     int supp, B;
4.     int band = 0;
5.     bool limitedBandwitdh = false;
6.     if (speed * 8 < 0) {
7.         *n = dash_context->bandsVideo.size() - 1;
8.         goto bez_ogranicenja;
9.     }
10.    for (i = 0; i < dash_context->bandsVideo.size(); i++) {
11.        if (speed * 8 <= dash_context-
   >bandsVideo.at(i)) //download_speed <= bandwidth[i]
12.        {
13.            limitedBandwitdh = true; //bandwidth limited
14.            break;
15.        }
16.    }
17.    if (!limitedBandwitdh) //no limits found
18.    {
19.        *n = i; //use bandwidth[max]
20.        goto bez_ogranicenja;
21.    }
22.    supp = i - 1;
23.    B = dash_context->videoBufferFullness;
24.    if (i >= 1) //limit not on bandwidth[0]
25.    {
26.        if ((int) (*n - supp) > 1) // limit on bandwidth[i] is 2 or more level
   s greater than last bandwidth
27.        {
28.            float nfrag = floor(tbuff * (dash_context-
   >bandsVideo.at(supp) / dash_context->segmentSizeAverage.at(supp)));
29.            if (nfrag > 0) {
30.                *n = supp + 1;
31.            } else {
32.                *n = supp;
33.            }
34.        } else
35.        {
36.            *n = supp;
37.        }
38.    } else // limit on bandwidth[0]
39.    {
40.        *n = 0;    // use bandwidth[min]
41.    }
42.    bez_ogranicenja:
43.    if (*n >= dash_context->bandsVideo.size()) {
44.        *n = dash_context->bandsVideo.size() - 1;
45.    }
46.    DBG_VERBOSE("BUFFER = %lf\n", dash_context->videoBufferFullness);
47.    DBG_VERBOSE("SPEED = %d\n", speed * 8);
48.    DBG_VERBOSE("QUALITY = %d\n", *n);
49.    if (dash_context->videoBufferFullness >= 48) {
50.        usleep(1000 * 1000 * (dash_context->videoBufferFullness - 48));
51.    }
52. }
```

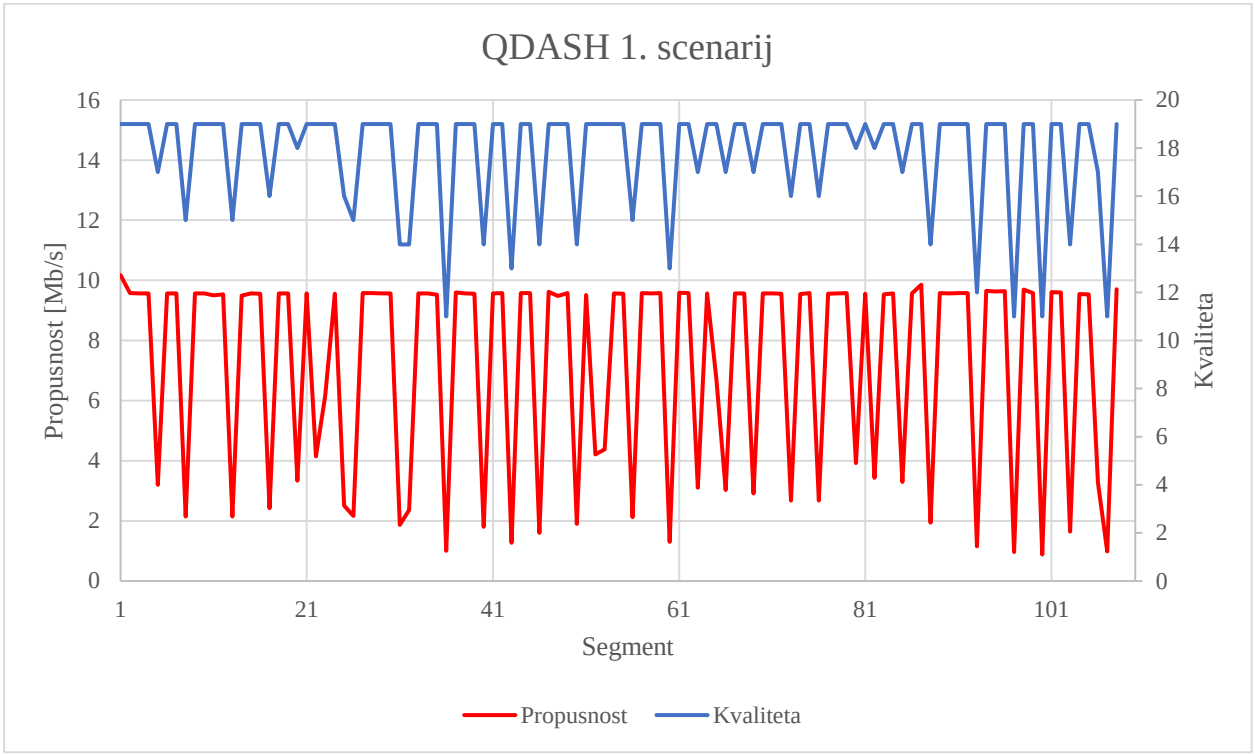
PRILOG II - Programski kod BBA algoritma

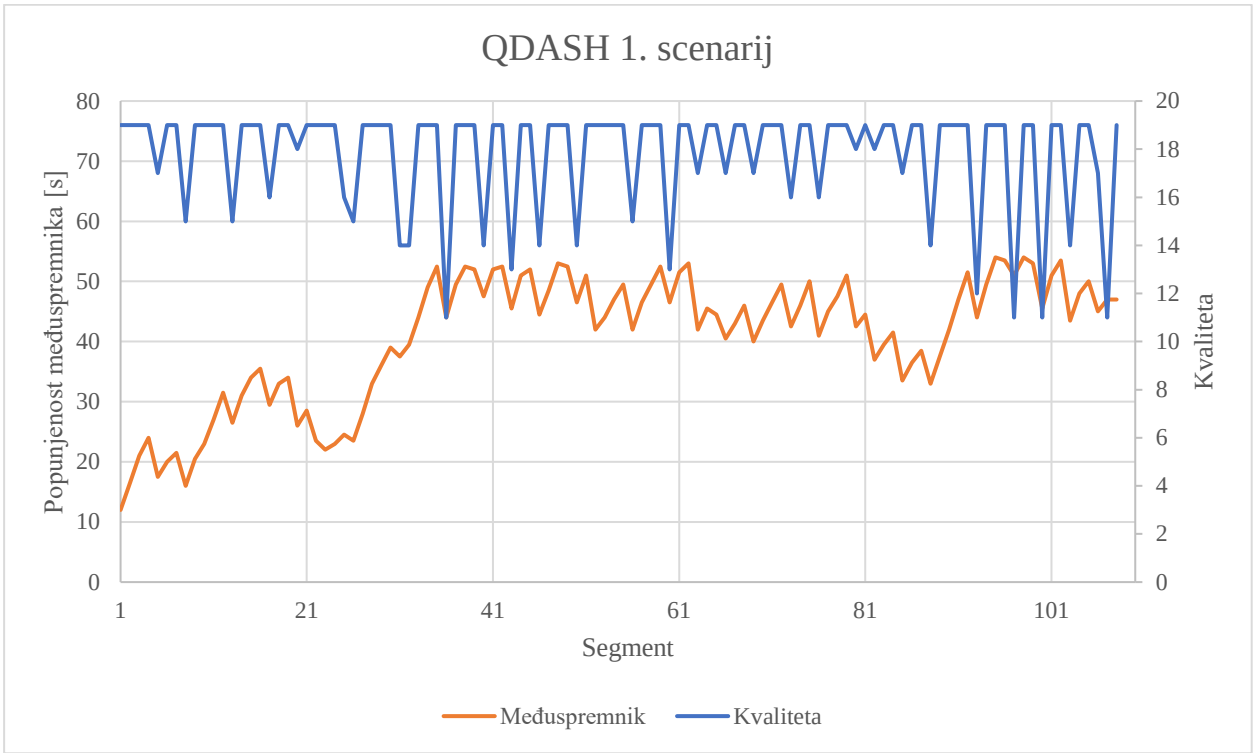
```
1. void SegmentManager::adapt(unsigned int* n, int speed, int stream_type, DASH_C
   ontext* dash_context)
2. {
3.     unsigned int i = 0;
4.     unsigned int reservoir = 12;
5.     unsigned int cushion = 24;
6.     unsigned int upper_reservoir = 12;
7.     unsigned int up_limit;
8.     unsigned int down_limit;
9.     if(*n >= (dash_context->bandsVideo.size()- 1))
10.    {
11.        up_limit = dash_context->bandsVideo.size()- 1;
12.    }
13.    else
14.    {
15.        up_limit = *n + 1;
16.    }
17.    if(*n == 0)
18.    {
19.        down_limit = 0;
20.    }
21.    else
22.    {
23.        down_limit = *n-1;
24.    }
25.    int map_buffer_to_rate = ((dash_context-
   >videoBufferFullness - reservoir) * ((double)(dash_context-
   >bandsVideo.size() - 1)/cushion));
26.    if(dash_context->videoBufferFullness <= reservoir)
27.    {
28.        *n = 0;
29.    }
30.    else if(dash_context->videoBufferFullness >= (reservoir + cushion))
31.    {
32.        *n = dash_context->bandsVideo.size()-1;
33.    }
34.    else if(map_buffer_to_rate >= up_limit)
35.    {
36.        *n = map_buffer_to_rate - 1;
37.    }
38.    else if(map_buffer_to_rate <= down_limit)
39.    {
40.        *n = map_buffer_to_rate + 1;
41.    }
42.    else
43.    {
44.        DBG_VERBOSE("quality same as previous\n");
45.    }
46.    DBG_VERBOSE("BUFFER = %lf\n", dash_context->videoBufferFullness);
47.    DBG_VERBOSE("SPEED = %d\n", speed * 8);
48.    DBG_VERBOSE("QUALITY = %d\n", *n);
49.    if(dash_context->videoBufferFullness >= (reservoir + cushion))
50.    {
51.        usleep(1000 * 1000 * (dash_context-
   >videoBufferFullness - (reservoir + cushion)));
52.    } }
```

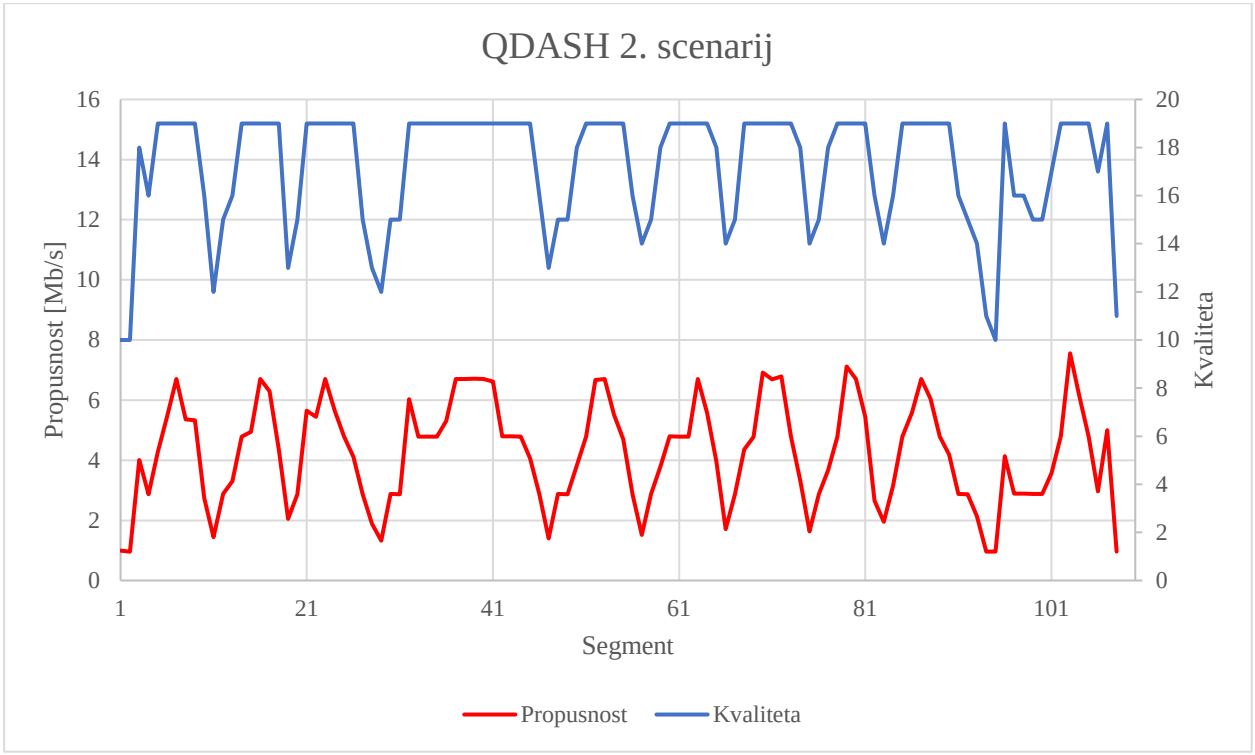
PRILOG III - Programski kod QAAD algoritma

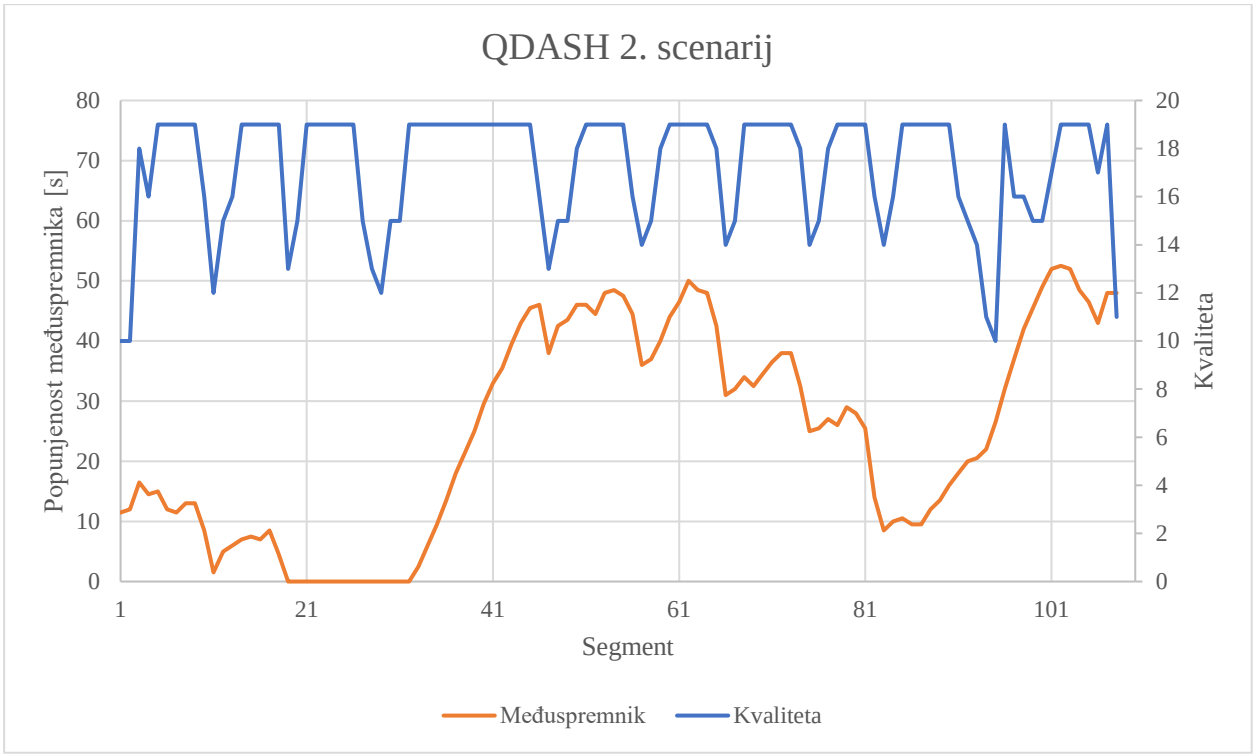
```
1. void SegmentManager::adapt(unsigned int* n, int speed, int stream_type, DASH_C
   ontext* dash_context)
2. {
3.     double current_buffer = dash_context->videoBufferFullness;
4.     if(bw_estimated_set == 0){
5.         bw_estimated = dash_context->bandsVideo.at(0);
6.         bw_estimated_set = 1;}
7.     unsigned int i = 0;
8.     unsigned int k;
9.     double time_to_deplete_buffer;
10.    int expected_num_segments_downloaded;
11.    int l_prev = *n;
12.    double w = 0.6;
13.    int marginal_buffer = 30;
14.    int minimum_buffer = 9;
15.    double bw_sample = speed * 8.0;
16.    bw_estimated = w * bw_estimated + (1-w) * bw_sample;
17.    for(i = 0; i < dash_context->bandsVideo.size(); i++)
18.    {
19.        if(bw_estimated <= dash_context-
   >bandsVideo.at(i)) //download_speed <= bandwidth[i]
20.        {
21.            break;
22.        }
23.    }
24.    int l_best = i - 1;
25.    if(l_best > l_prev)
26.    {
27.        if(current_buffer > marginal_buffer)
28.        {
29.            *n = l_prev + 1;
30.        }
31.    }
32.    else if(l_best < l_prev)
33.    {
34.        k = 0;
35.        do
36.        {
37.            time_to_deplete_buffer = (current_buffer - minimum_buffer) /
38.            (1 - (bw_estimated / dash_context->bandsVideo.at(l_prev - k)));
39.            expected_num_segments_downloaded = floor((time_to_deplete_buffer
40.            * bw_estimated) / (SEGMENT_DURATION * dash_context-
   >bandsVideo.at(l_prev - k)));
41.            k++;
42.        }
43.        while((expected_num_segments_downloaded < 1) && (k < (l_prev - 1)));
44.        *n = l_prev - k;
45.    }
46.    DBG_VERBOSE("QUALITY = %d\n", *n);
47.    DBG_VERBOSE("BUFFER = %lf\n", dash_context->videoBufferFullness);
48.    DBG_VERBOSE("SPEED = %d\n", speed*8);
49.
50.    if(dash_context->videoBufferFullness >= 48)
51.    {
52.        usleep(1000 * 1000 * (dash_context->videoBufferFullness - 48));
53.    }}
```

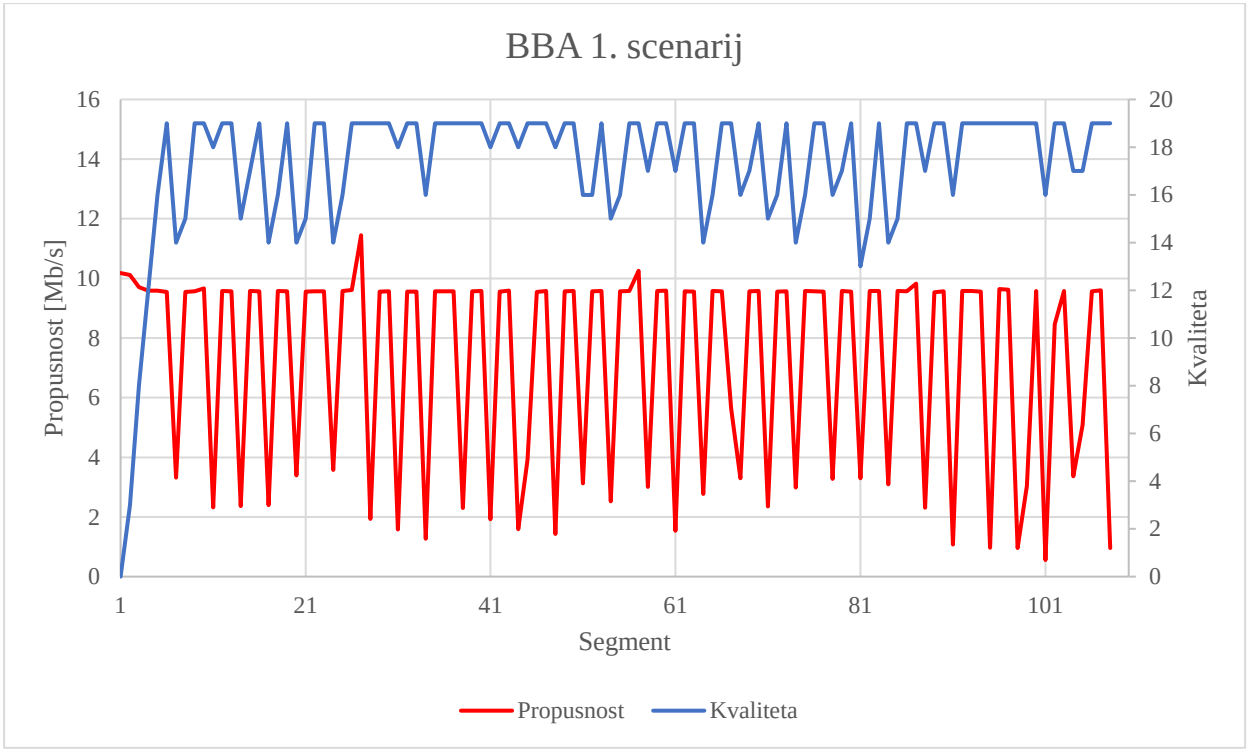
PRILOG IV – Rezultati testiranja za video zapis ED

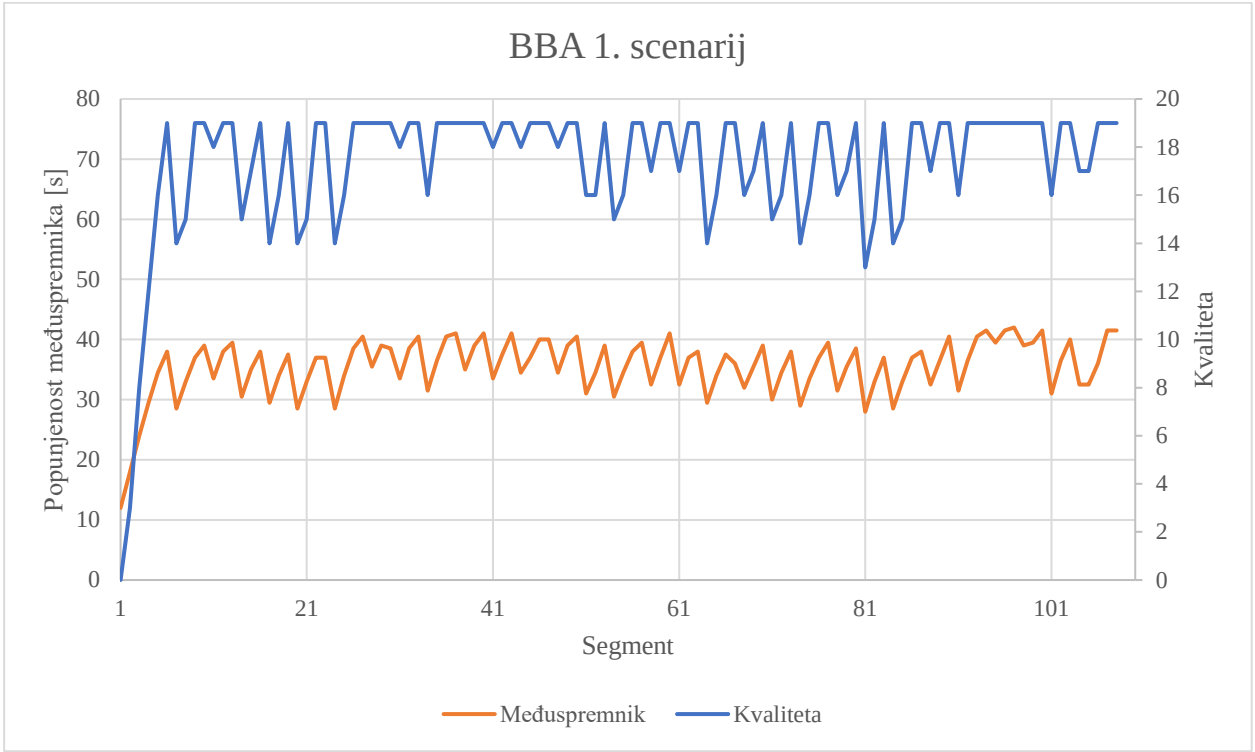


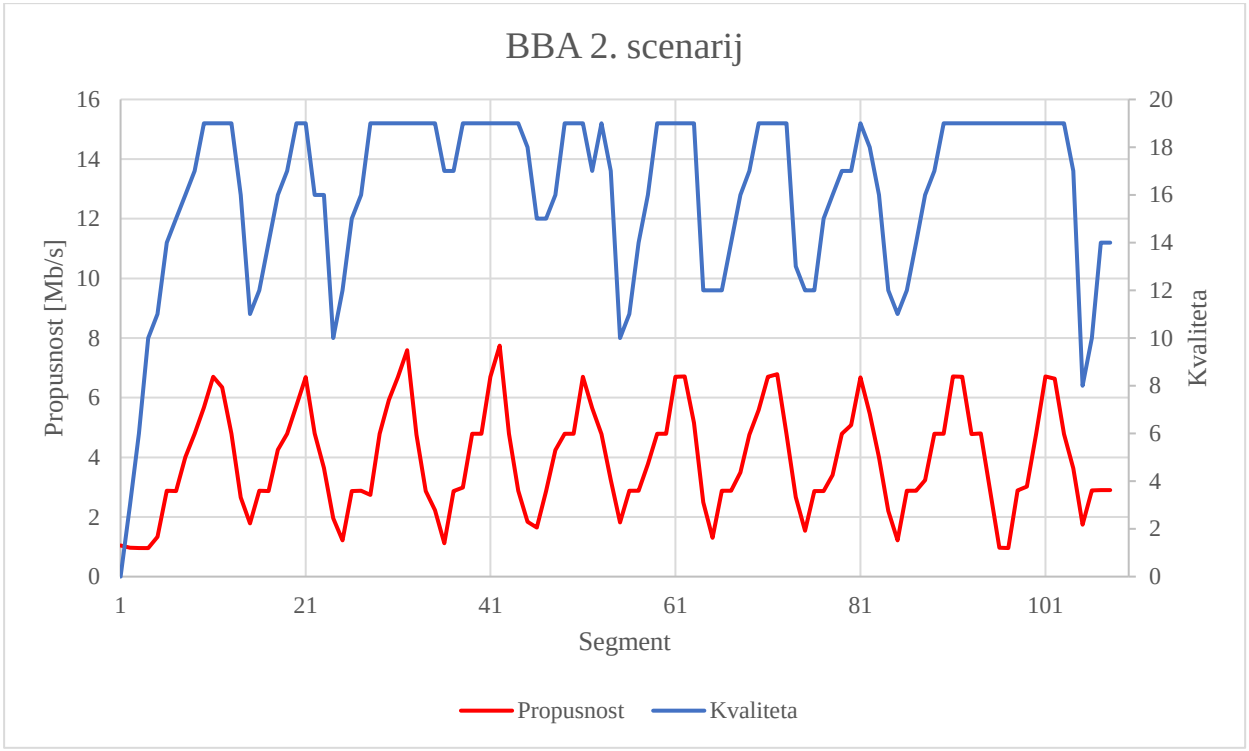


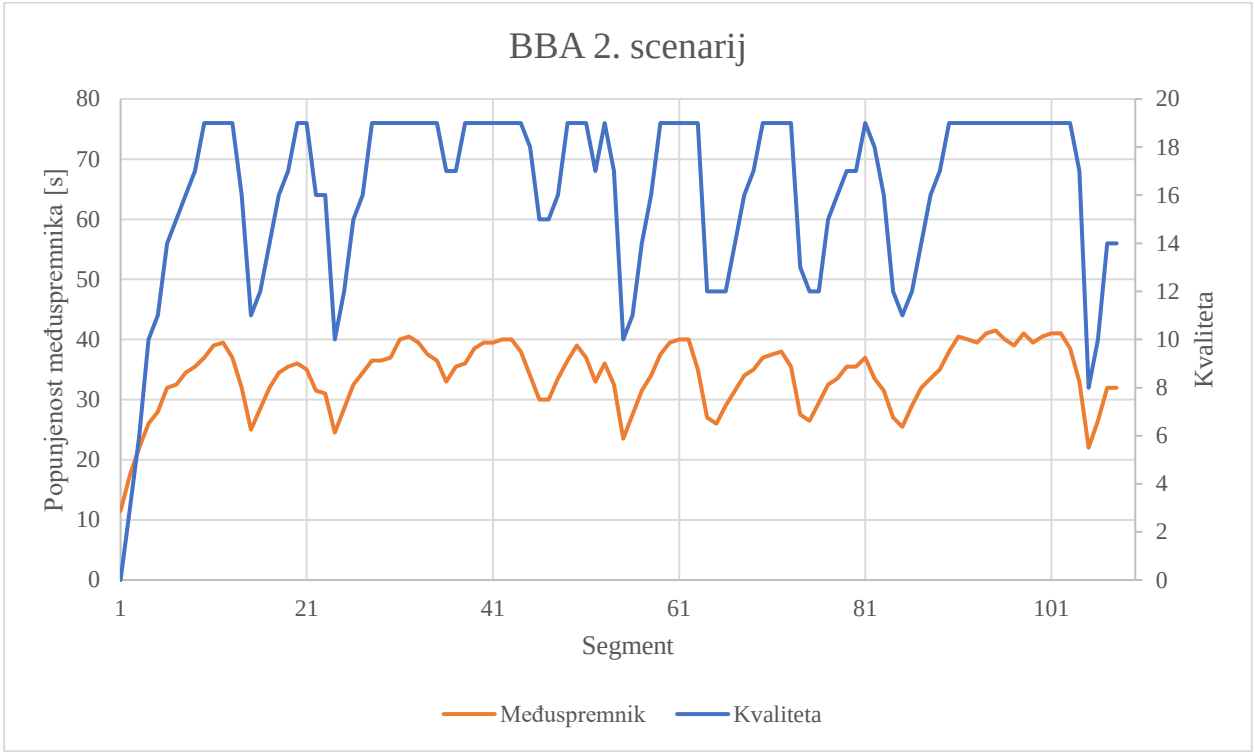

QDASH 2. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta

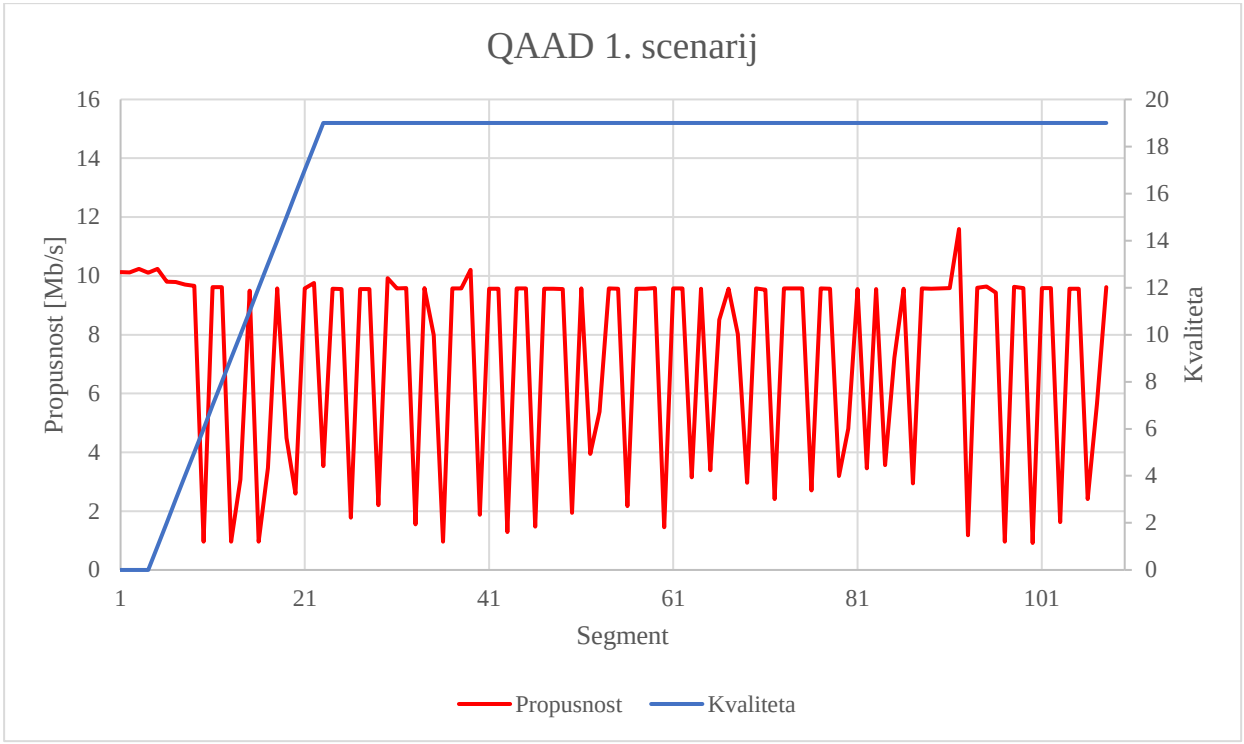

QDASH 2. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta

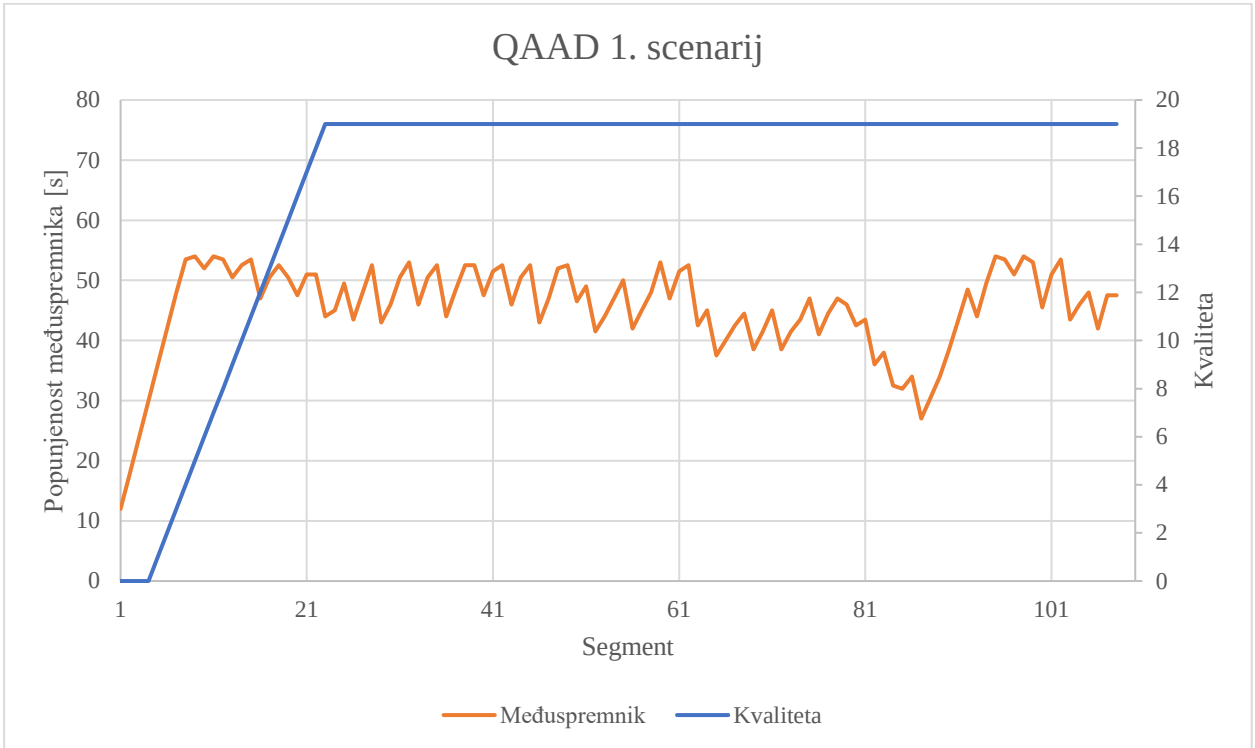

BBA 1. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta

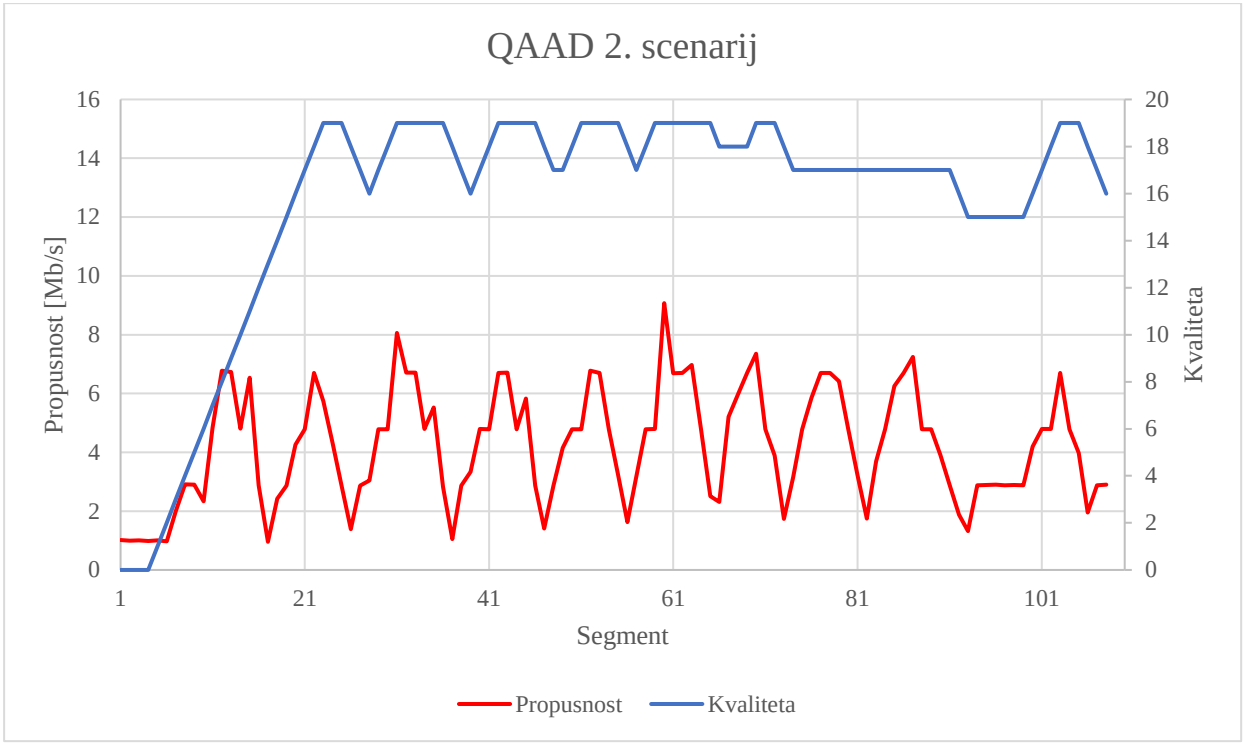

BBA 1. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta

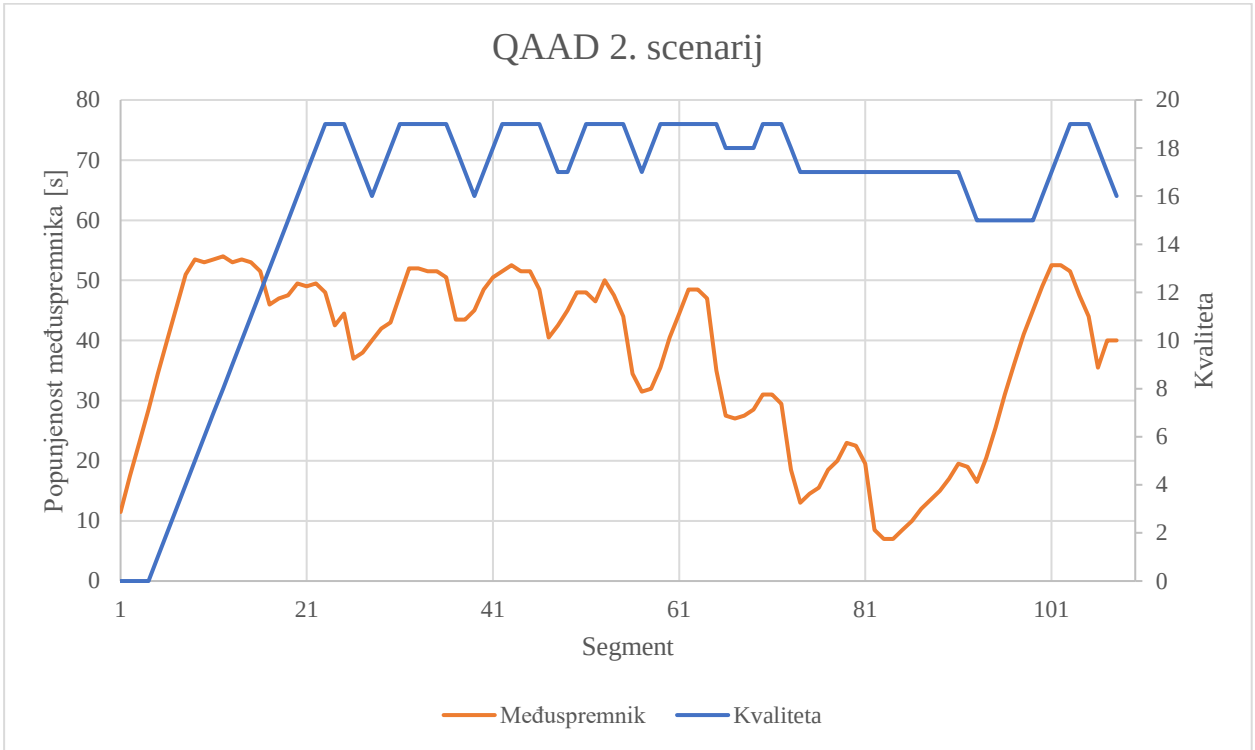

BBA 2. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta


BBA 2. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta

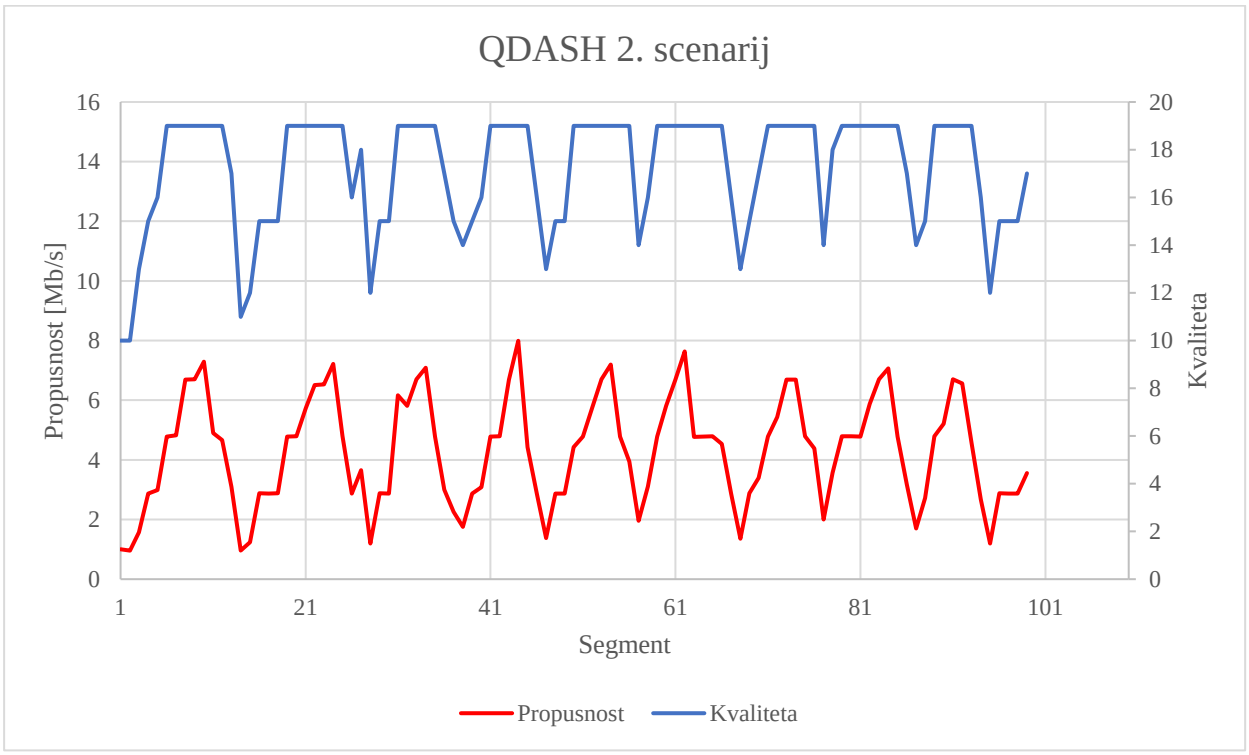

QAAD 1. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta

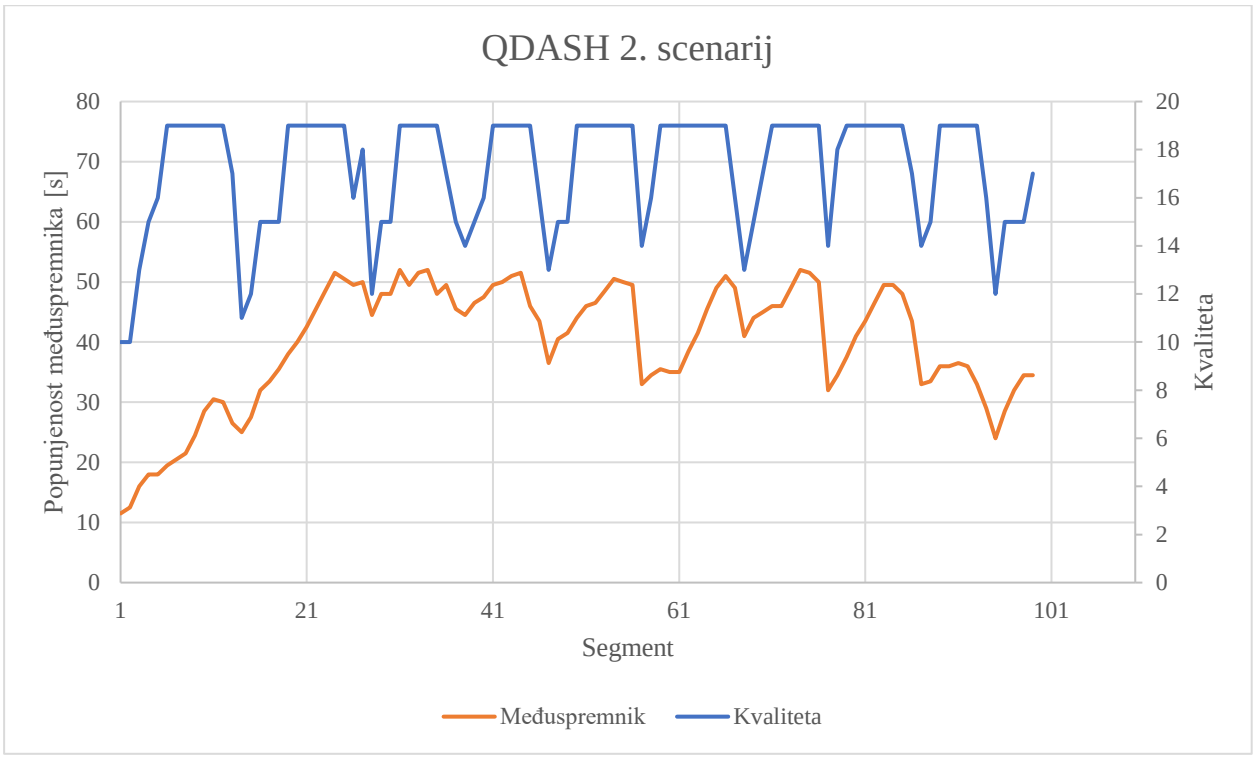

QAAD 1. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta

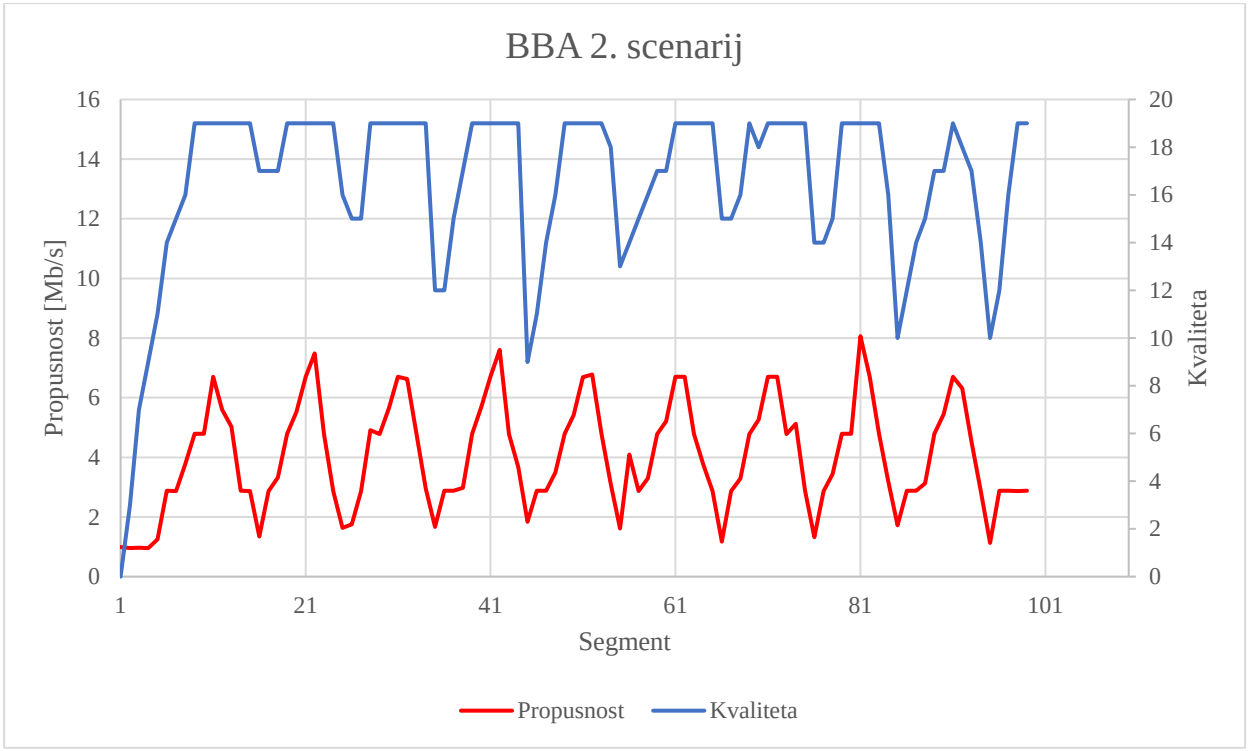

QAAD 2. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta

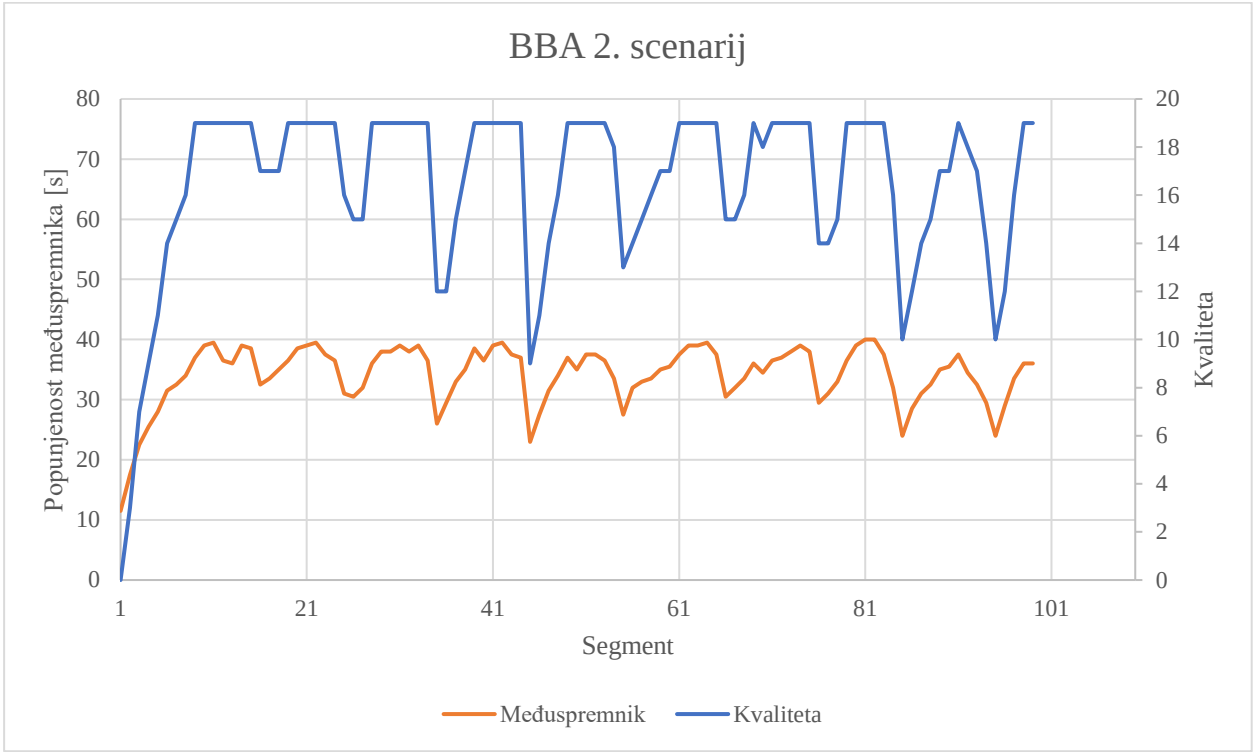

QAAD 2. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta

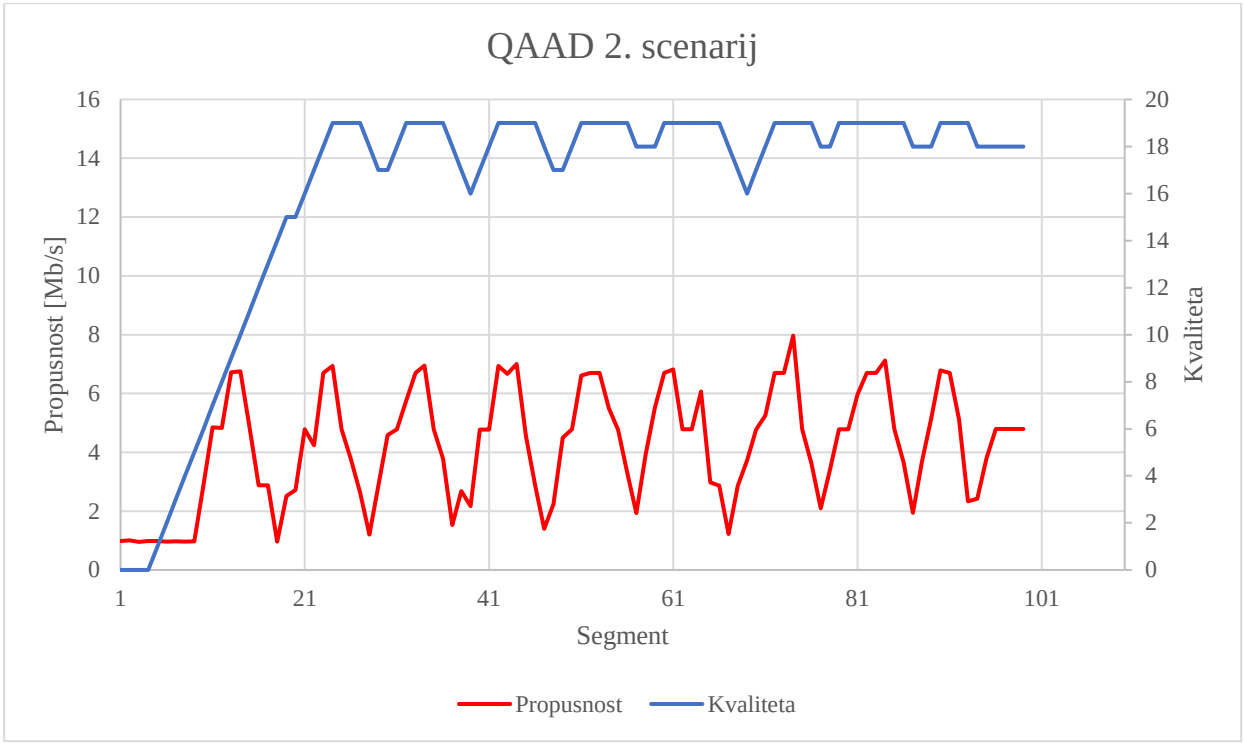
PRILOG V – Rezultati testiranja za video zapis BBB





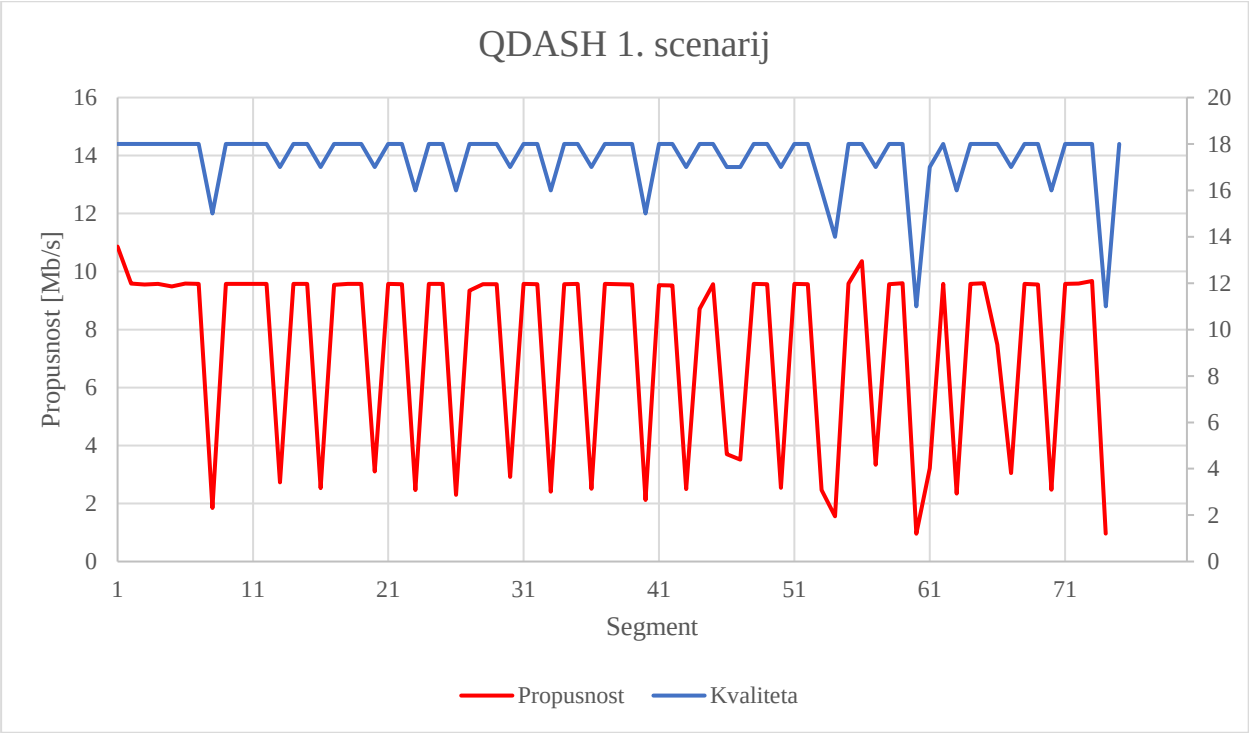

BBA 2. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta

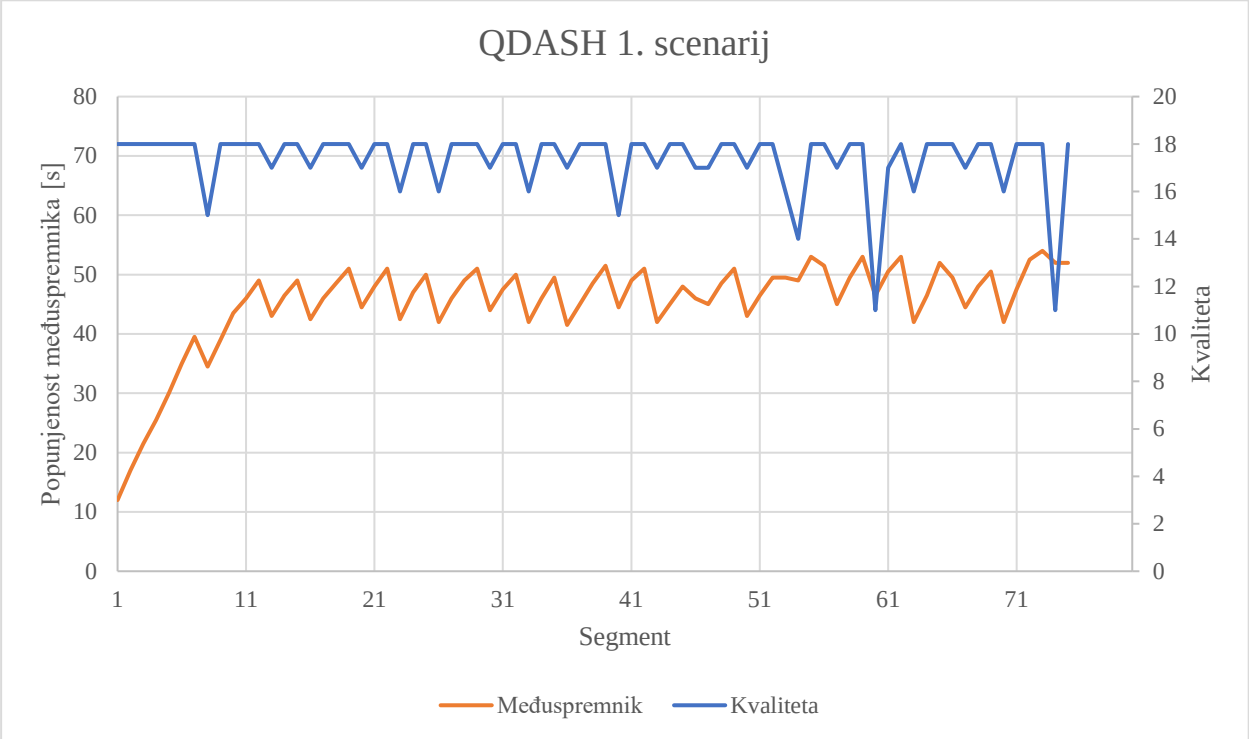

BBA 2. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta

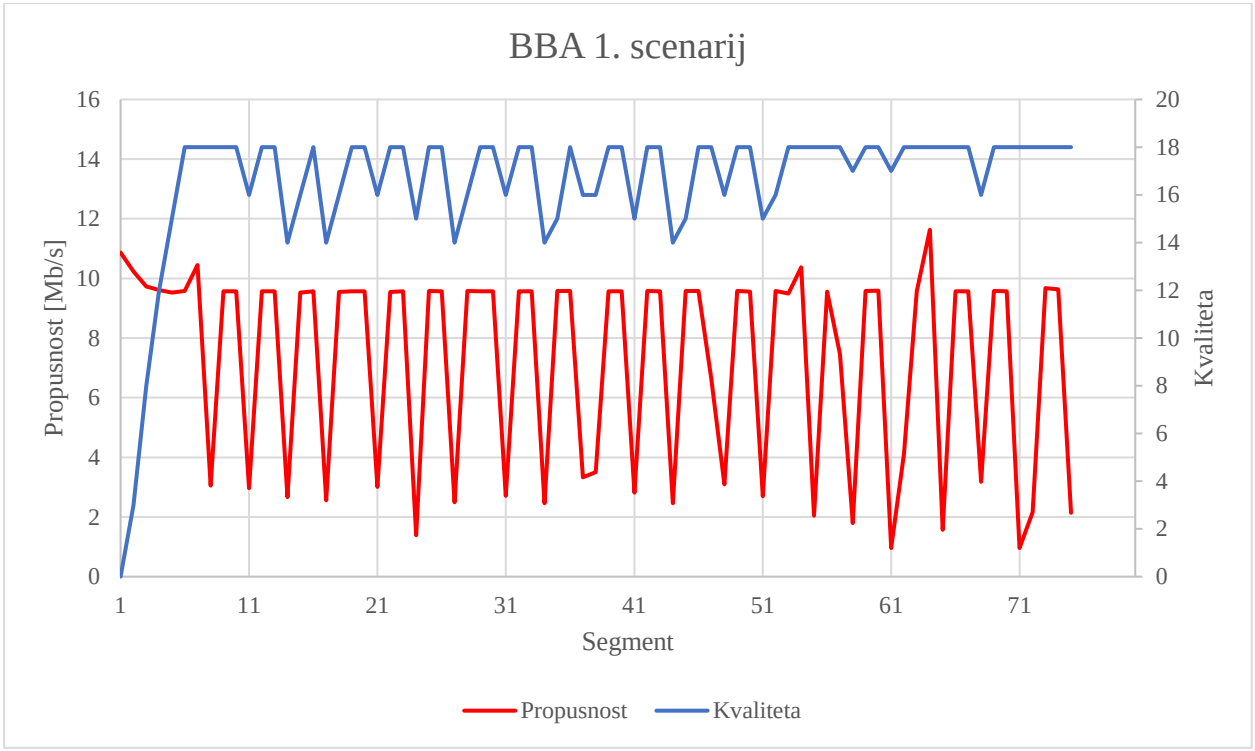

QAAD 2. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta

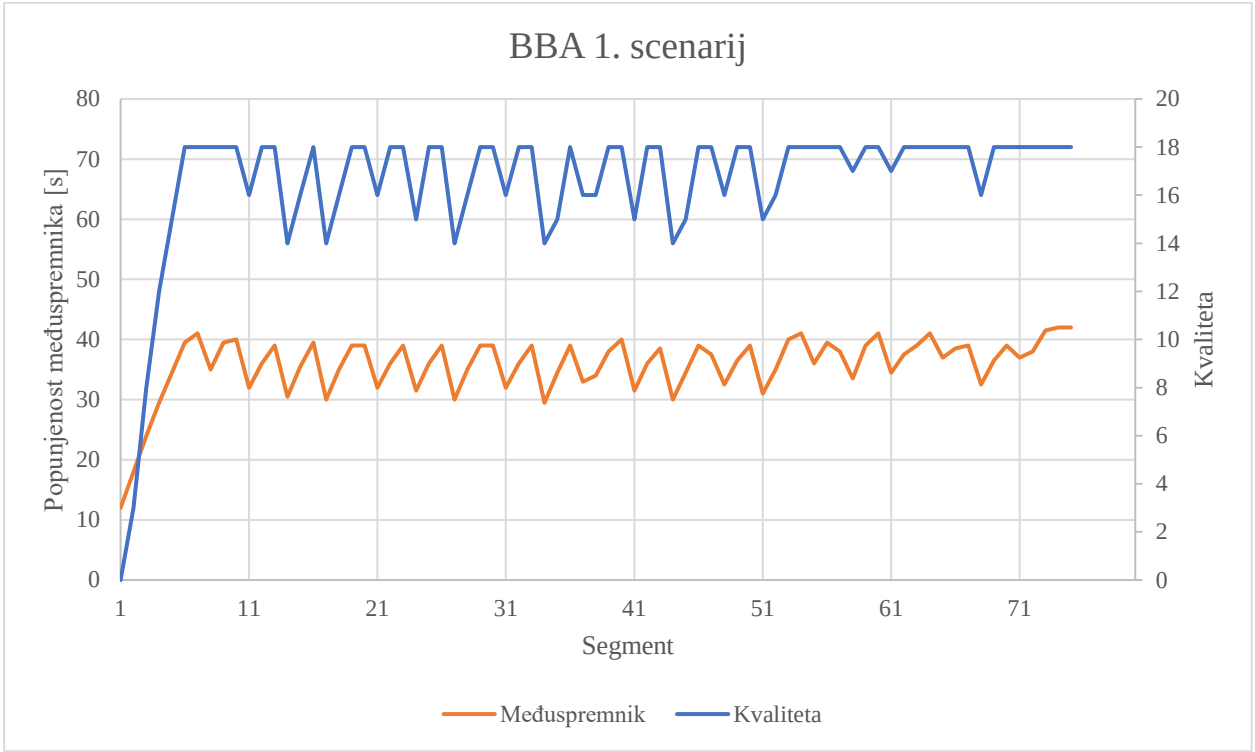

QAAD 2. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta

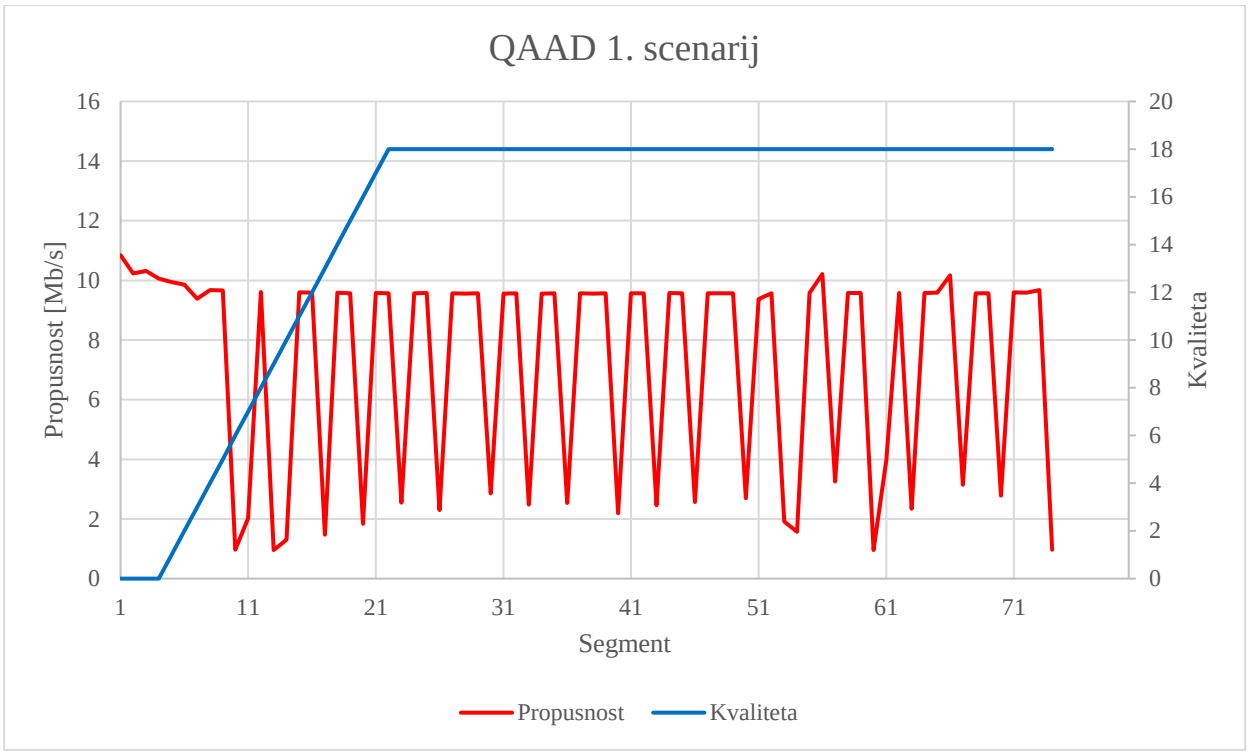
PRILOG VI – Rezultati testiranja za video zapis OFAM

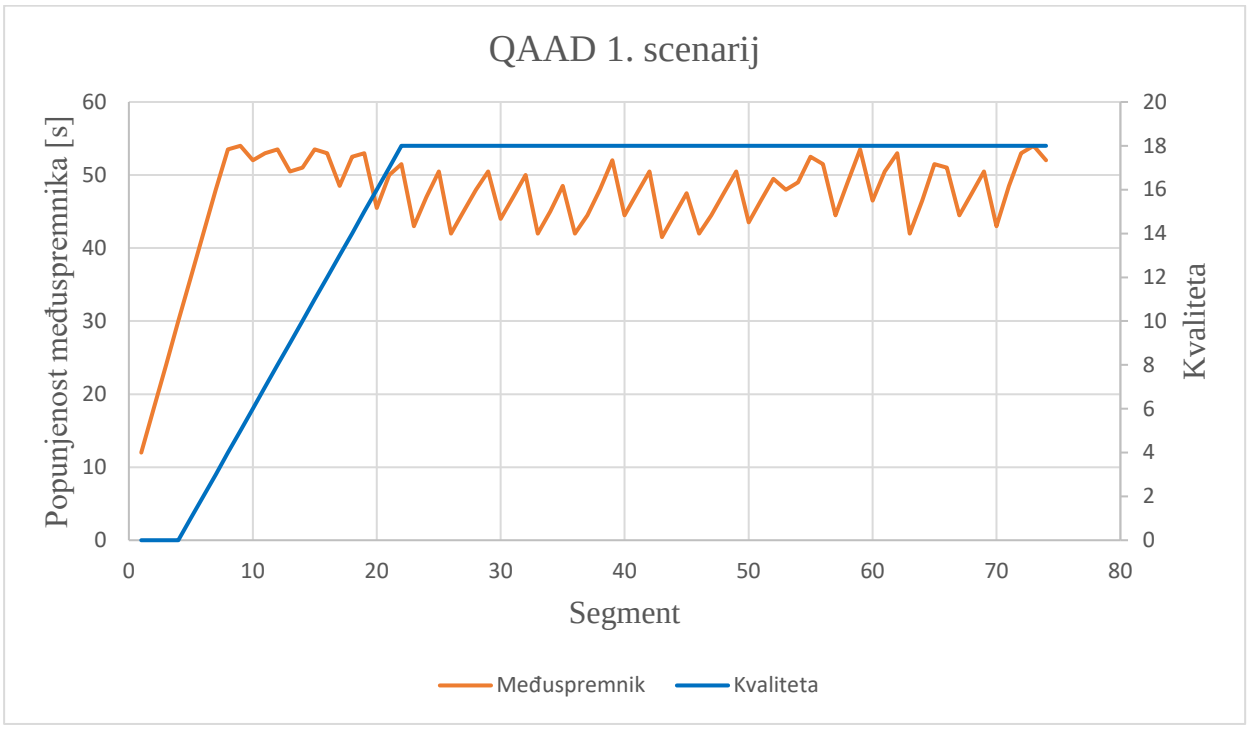





BBA 1. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta


BBA 1. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta


QAAD 1. scenarij
Propusnost [Mb/s]
Kvaliteta
Segment
Propusnost
Kvaliteta


QAAD 1. scenarij
Popunjenost međuspremnika [s]
Kvaliteta
Segment
Međuspremnik
Kvaliteta