



**УНИВЕРЗИТЕТ У НОВОМ САДУ  
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**



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

## **ЗАВРШНИ (BACHELOR) РАД**

**Кандидат:            Ненад Јовановић  
Број индекса:        Е12683**

**Тема рада:            Реализација BitTorrent клијентске апликације за  
репродукцију мултимедијалног садржаја у виду тока  
података**

**Ментор рада:        проф. др Никола Теслић**

**Нови Сад, месец, година**



УНИВЕРЗИТЕТ У НОВОМ САДУ • ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА  
21000 НОВИ САД, Трг Доситеја Обрадовића 6

## КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, <b>РБР:</b>	
Идентификациони број, <b>ИБР:</b>	
Тип документације, <b>ТД:</b>	Монографска документација
Тип записа, <b>ТЗ:</b>	Текстуални штампани материјал
Врста рада, <b>ВР:</b>	Завршни (Bachelor) рад
Аутор, <b>АУ:</b>	Ненад Јовановић
Ментор, <b>МН:</b>	проф. др Никола Теслић
Наслов рада, <b>НР:</b>	Реализација BitTorrent клијентске апликације за репродукцију мултимедијалног садржаја у виду тока података
Језик публикације, <b>ЈП:</b>	Српски / латиница
Језик извода, <b>ЈИ:</b>	Српски
Земља публикавања, <b>ЗП:</b>	Република Србија
Уже географско подручје, <b>УГП:</b>	Војводина
Година, <b>ГО:</b>	2012
Издавач, <b>ИЗ:</b>	Ауторски репринт
Место и адреса, <b>МА:</b>	Нови Сад; трг Доситеја Обрадовића 6
Физички опис рада, <b>ФО:</b> (поглавља/страна/ цитата/табела/слика/графика/прилога)	7/38/0/4/7/0/0
Научна област, <b>НО:</b>	Електротехника и рачунарство
Научна дисциплина, <b>НД:</b>	Рачунарска техника
Предметна одредница/Кључне речи, <b>ПО:</b>	BitTorrent протокол, репродукција тока података
<b>УДК</b>	
Чува се, <b>ЧУ:</b>	У библиотеци Факултета техничких наука, Нови Сад
Важна напомена, <b>ВН:</b>	
Извод, <b>ИЗ:</b>	У овом раду изложене су основе BitTorrent протокола, као и идеје реализоване у клијентској апликацији за преузимање садржаја овим протоколом и његово истовремено репродуковање у виду тока података.
Датум прихватања теме, <b>ДП:</b>	
Датум одбране, <b>ДО:</b>	
Чланови комисије, <b>КО:</b>	Председник:
	Члан:
	Члан, ментор:
	Потпис ментора



## KEY WORDS DOCUMENTATION

Accession number, <b>ANO</b> :	
Identification number, <b>INO</b> :	
Document type, <b>DT</b> :	Monographic publication
Type of record, <b>TR</b> :	Textual printed material
Contents code, <b>CC</b> :	Bachelor Thesis
Author, <b>AU</b> :	Nenad Jovanović
Mentor, <b>MN</b> :	prof. dr Nikola Teslić
Title, <b>TI</b> :	One solution of a BitTorrent stream player client
Language of text, <b>LT</b> :	Serbian
Language of abstract, <b>LA</b> :	Serbian
Country of publication, <b>CP</b> :	Republic of Serbia
Locality of publication, <b>LP</b> :	Vojvodina
Publication year, <b>PY</b> :	2012
Publisher, <b>PB</b> :	Author's reprint
Publication place, <b>PP</b> :	Novi Sad, Dositeja Obradovica sq. 6
Physical description, <b>PD</b> : <small>(chapters/pages/ref./tables/pictures/graphs/appendixes)</small>	7/38/0/4/7/0/0
Scientific field, <b>SF</b> :	Electrical Engineering
Scientific discipline, <b>SD</b> :	Computer Engineering, Engineering of Computer Based Systems
Subject/Key words, <b>S/KW</b> :	BitTorrent protocol, stream playback
<b>UC</b>	
Holding data, <b>HD</b> :	The Library of Faculty of Technical Sciences, Novi Sad, Serbia
Note, <b>N</b> :	
Abstract, <b>AB</b> :	The paper explains the details of the BitTorrent protocol as well as the ideas implemented in the realized client application that simultaneously downloads the content shared using BitTorrent protocol and reproduces the downloaded content as a stream.
Accepted by the Scientific Board on, <b>ASB</b> :	
Defended on, <b>DE</b> :	
Defended Board, <b>DB</b> :	President:
	Member:
	Member, Mentor:
	Mentor's sign

## **Zahvalnost**

Želim da se zahvalim svima koji su na bilo koji način doprineli izradi i kvalitetu ovog rada, a posebno najbližima na podršci, brizi i strpljenju tokom njegove izrade, a i samog školovanja.

Ovim putem bih takođe želeo i da se izvinim BitTorrent zajednici na kršenju njihovih principa i načela, što je nažalost uzrokovano samom temom, ciljem i uslovima izrade ovog rada, naročito zbog toga što te principe i načela lično veoma vrednujem i poštujem.

## SADRŽAJ

1. Uvod.....	1
2. Teorijske osnove .....	3
2.1 P2P mrežni model i protokoli .....	3
2.2 BitTorrent protokol .....	5
2.2.1 Osnove BitTorrent protokola.....	5
2.2.2 Deljenje torrent-a .....	7
2.2.3 Peer wire protokol.....	8
2.2.4 Princip funkcionisanja BitTorrent protokola .....	9
2.2.5 Algoritam gušenja.....	11
2.2.6 Performanse BitTorrent protokola u praksi .....	12
3. Analiza problema.....	14
3.1 Analiza problema vezanih za programsku podršku .....	14
3.2 Analiza problema vezanih za prilagođenje BitTorrent protokola .....	15
4. Koncept rešenja.....	17
4.1 Izbor proširenja protokola za implementaciju.....	17
4.2 Prilagođenje protokola specifičnoj primeni .....	18
4.2.1 Implementirani mehanizmi vezani za delove i zahteve .....	18
4.2.2 Implementirani mehanizmi vezani za učesnike .....	20
4.3 Regulacija reprodukcije sadržaja .....	21
5. Testiranje i rezultati .....	22
5.1 Zavisnost kvaliteta doživljaja od odnosa propusnosti veze i bitske brzine video sadržaja	22
5.2 Ponašanje aplikacije u rojevima sa različitim karakteristikama.....	24

---

5.3	Mogućnost razvijanja dovoljno velike brzine preuzimanja podataka za kontinualnu reprodukciju video sadržaja .....	26
5.4	Količina napravljenog suvišnog saobraćaja tokom preuzimanja sadržaja .....	26
5.5	Ukupni rezultati.....	27
6.	Zaključak .....	28
7.	Literatura.....	29

## SPISAK SLIKA

Slika 2.1 Šematski prikaz dva mrežna modela.....	3
Slika 2.2 Šematski prikaz načina funkcionisanja BitTorrent protokola.....	6
Slika 2.3 Prikaz logičke structure podataka unutar jednog torrent-a .....	7
Slika 2.4 Formati poruka peer wire protokola .....	8
Slika 2.5 Primer jednostavne razmene poruka u peer wire protokolu .....	11
Slika 4.1 Prikaz okvira tokom preuzimanja delova podataka .....	18
Slika 4.2 Prikaz razmene poruka prilikom zahtevanja blokova bez i sa primenom tehinke višestrukih zahteva .....	20

## **SPISAK TABELA**

Tabela 1 Srednje vrednosti posmatranih parametara tokom testiranja zavisnosti kvaliteta doživljaja korisnika od odnosa propusnosti veze i bitske brzine sadržaja.....	23
Tabela 2 Karakteristike video sadržaja torrent-a korišćenih za testiranje ponašanja aplikacije u rojevima sa različitim karakteristikama .....	24
Tabela 3 Karakteristike rojeva testiranih torrent-a i srednje praćenih parametara tokom testova.....	25
Tabela 4 Pregled količine suvišnog saobraćaja napravljenog tokom preuzimanja sadržaja testiranih torrent-a.....	27

**SKRAĆENICE**

<b>P2P</b>	- <i>Peer to Peer</i> , od učesnika do učesnika
<b>URL</b>	<i>Universal Resource Locator</i> , uniformni lokator resursa
<b>IP</b>	<i>Internet Protocol</i>
<b>HTTP</b>	<i>Hypertext Transfer Protocol</i>
<b>SHA</b>	<i>Secure Hash Algorithm</i> , algoritam za sigurnosni heš kod
<b>TCP</b>	<i>Transmission Control Protocol</i>

## 1. Uvod

Za razliku od tradicionalne primene BitTorrent protokola koja se zasniva primarno na deljenju podataka, njihova reprodukcija kao tok podataka zahteva određena prilagođenja u implementaciji protokola i klijentske aplikacije. Kako bi rešenje uopšte funkcionisalo zajedno sa ostalim već postojećim programskim rešenjima, unete izmene ne smeju biti u sukobu sa osnovnom verzijom protokola, tj. moraju biti transparentne za sve ostale učesnike tokom preuzimanja podataka.

Akcent rada je na samom protokolu i problemima koje donosi njegova primena na ovaj način, ali je za potrebe rada realizovana osnovna programska podrška za BitTorrent protokol.

U ovom radu prikazana je jedna realizacija aplikacije za preuzimanje sadržaja putem BitTorrent protokola i njegovu reprodukciju u obliku toka podataka (eng. *stream*). U radu su opisane i osnove samog protokola, izmene unete u sam protokol kao i problemi koji se pri tom javljaju zajedno sa predlozima rešenja.

Rešenje je realizovano za Linux operativni sistem. Aplikacija je testirana na njegovoj verziji za platforme bazirane na x86 arhitekturi, ali kako je razvijana sa prenosivošću na umu, lako se može preneti na druge platforme bazirane na Linux-u.

Rad se sastoji od sedam poglavlja.

U prvom poglavlju je ukratko opisan sadržaj samog rada.

Drugo poglavlje daje uvid u istoriju BitTorrent protokola, njegov razvoj i popularizaciju. Takođe je dat osnovni opis načina funkcionisanja protokola, potreban za dalje razumevanje problematike. Zatim se opisuje njegova namena i primena – deljenje podataka, kao i posledice koje ona nosi sa sobom.

U trećem poglavlju je analizirana mogućnost proširenja primene BitTorrent protokola, dati su predlozi na koji način bi se to moglo postići, kao i prednosti i potencijalne mane rešenja izloženog u ovom radu.

Četvrto poglavlje se bavi samim programskim rešenjem. Kako je ovaj rad više teorijski orijentisan, u ovom poglavlju neće se zalaziti u detalje samog programskog rešenja za podršku BitTorrent protokolu. Biće opisano okruženje u kojem je programska podrška razvijana, zavisnosti, a biće izložen i kratak pregled realizacije programske podrške.

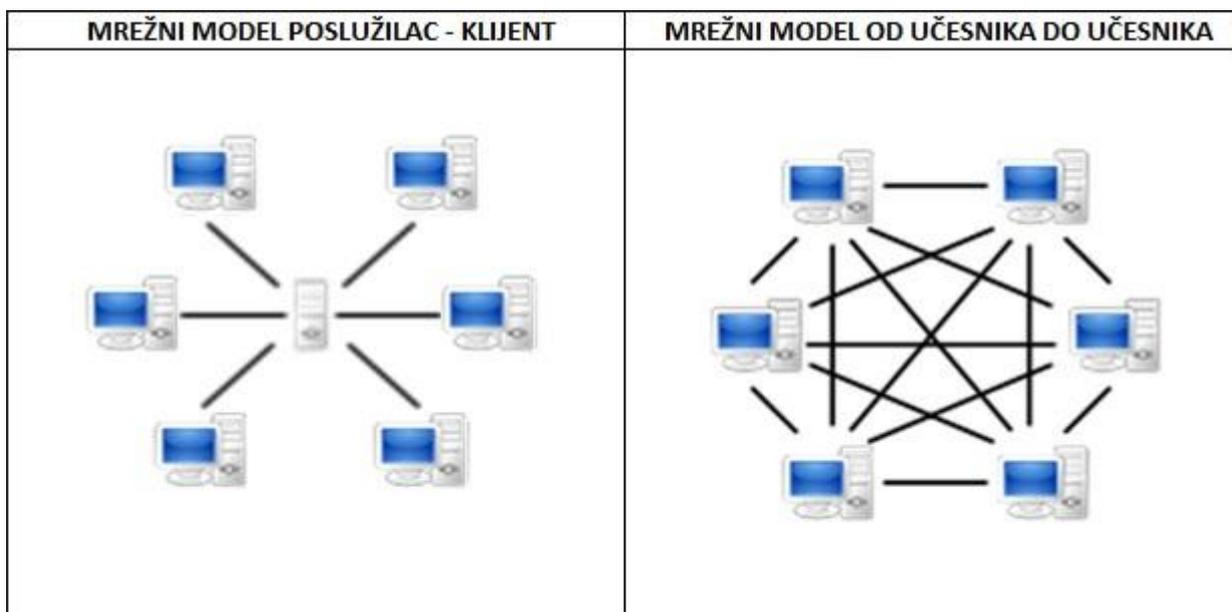
U petom poglavlju je izložen način i okruženje u kojem je rešenje testirano kao i dobijeni rezultati.

Šesto poglavlje predstavlja kratak osvrt na ceo rad, a dati su i dalji mogući pravci razvoja i mogućnosti unapređenja izloženog programskog rešenja.

Na kraju, sedmo poglavlje pruža uvid u literaturu koja je korišćena tokom izrade ovog rada.

## 2. Teorijske osnove

U ovom poglavlju dat je kratak uvid u razvoj protkola po modelu „od učesnika do učesnika“ (eng. *peer to peer*), a potom su detaljno objašnjeni aspekti BitTorrent protokola koji su važni za razumevanje ovog rada, kao i njegove performanse i karakteristike i poređenje sa tradicionalnim rešenjima po modelu „klijent - poslužilac“. Šematski prikazi modela o kojima će se diskutovati se može videti na slici 2.1.



Slika 2.1 Šematski prikaz dva mrežna modela

### 2.1 P2P mrežni model i protokoli

Uobičajeni model po kojem se formiraju mreže i mrežno bazirane usluge u današnjem internetu jeste „klijent - poslužilac“ (eng. *client - server*). On se zasniva na tome da klijenti (najčešće krajnji korisnici) zahtevaju usluge od poslužilaca, što se najčešće svodi na prenos određene količine podataka od poslužioca ka klijentu. Tipičan primer za ovaj model je usluga

videa na zahtev (eng. *video on demand*). Ovaj model omogućava skalabilnost i povećanje brzine na najjednostavniji način – nadograđivanjem poslužioca ili puštanjem u promet novih poslužioca. Ono što je glavna mana ovog modela jeste neravnopravnost učesnika, odnosno asimetričnost interakcije. Zbog toga, poslužiocci su tipično znatno zahtevniji od klijenata po pitanju resursa (procesna moć, operativna i masovna memorija, itd.) i često skupi za iznajmljivanje i održavanje. Internet veza ka i od poslužioca najčešće zahteva drastično veću propusnu moć (eng. *throughput*) u poređenju sa klijentom, kako bi se što više klijenata što brže moglo uslužiti istovremeno. Samim tim lako može doći do preopterećenja veze ka poslužiocu usled prevelikog broja klijenata i, kao posledica, uskraćivanja ili opadanja kvaliteta usluga zbog kojih je poslužilac pušten u rad. Za ovaj model možemo takođe reći i da je centralizovan, jer je za njegovo ispravno funkcionisanje potrebno da je poslužilac u svakom momentu ispravan. Samim tim, poslužilac predstavlja kritičnu tačku otkaza čijim prestankom rada otkazuje ceo sistem. I pored ovih nedostataka, ovaj model je najčešće korišćen model u današnjem internetu, jer poseduje visok stepen determinizma, jednostavnost i, uz odgovarajuće mere, mali stepen otkaza, kao i zbog mogućnosti vrlo raznovrsne primene.

Kao alternativa prethodno opisanom mrežnom modelu, stoji model „od učesnika do učesnika“ (eng. *peer to peer*, P2P). Za razliku od mreža „klijent - poslužilac“, u mrežama tipa P2P svi učesnici su ravnopravni, tj. svi učesnici su istovremeno i poslužiocci i klijenti. Ovakva organizacija ima nekoliko prednosti. Kako su sami klijenti poslužiocci, nema potrebe za zahtevnim i skupim poslužiocima, kao ni za vezama ogromne propusne moći, jer se delimičan ili celokupan teret, u zavisnosti od protokola, raspoređuje po svim učesnicima. Takođe, sa povećanjem broja učesnika, povećava se i kapacitet mreže, pa problem vezan za preopterećenje ne postoji. Ovakve karakteristike dovele su do toga da je najčešća namena P2P protokola deljenje i umnožavanje podataka među korisnicima. Međutim kod P2P mrežnog modela, javljaju se drugi problemi. Kako je P2P model po samoj prirodi decentralizovan, jedan od problema koji se javlja je: Kako uspešno potpuno decentralizovati sistem da bi se uklonilo postojanje kritične tačke otkaza, a zadržati sve pozitivne karakteristike P2P mreža? Prve dve generacije P2P protokola (protokoli korišćeni od strane nekada popularnih servisa za deljenje podataka Napster i FastTrack) nisu uspešno rešavale ovaj problem i ovi protokoli nisu smatrani čistim P2P protokolima (eng. *pure P2P*), jer su za njihovo funkcionisanje bili potrebni ili standardni poslužiocci (sa dodatnim resursima) ili tzv. super čvorovi (eng. *supernodes*), koji bi tu funkciju obavljali. Iako su oni suštinski služili samo za inicijalnu uspostavu veze između učesnika, oni su, zbog dodatne obrade, i dalje morali da imaju značajno veću količinu resursa u odnosu na obične klijente. Drugi problem koji se javlja u P2P mrežama je često pojavljivanje tzv. „slepih putnika“ (eng. *free rider*, *free loader*), odnosno korisnika koji ne izvršavaju dužnost poslužioca tj. ne dele

sadržaj, već ga samo preuzimaju. Kao posledica se javlja smanjena skalabilnost i veća nepouzdanost mreže (nikad se ne zna da li će učesnik odgovoriti na zahtev za podacima ili ne).

Jedan od P2P protokola koji prethodno izložene probleme rešava sa visokom stopom uspešnosti je BitTorrent.

## 2.2 BitTorrent protokol

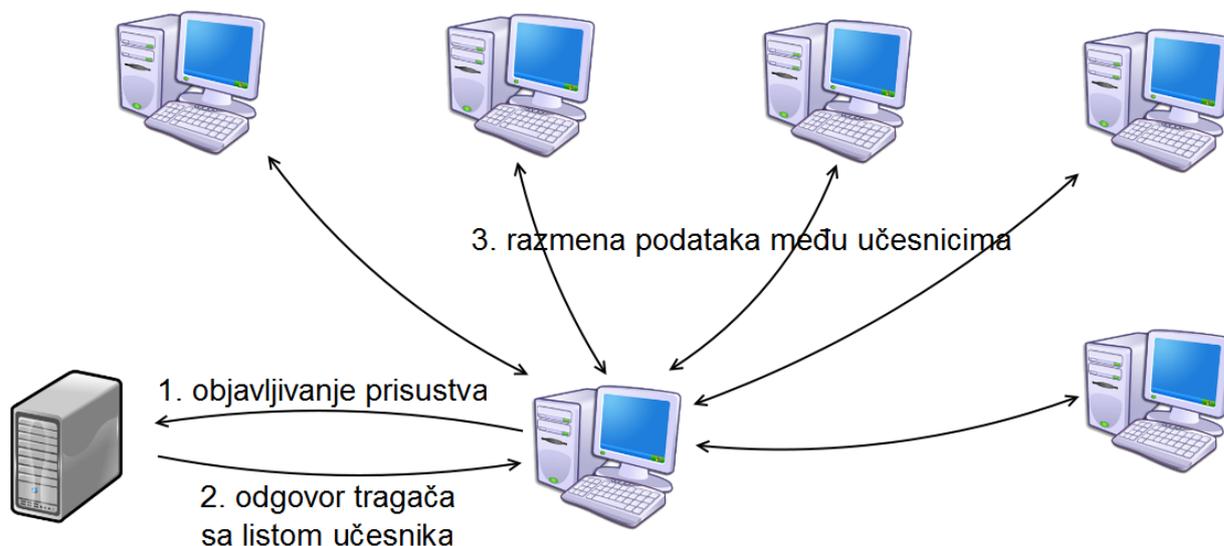
### 2.2.1 Osnove BitTorrent protokola

BitTorrent je P2P protokol koji služi za deljenje i umnožavanje podataka. Najčešće su to veće količine podataka (više stotina, pa čak i hiljada MB), na šta njegovo ime i ukazuje (*bit torrent* – bujica bita). Akcenat protokola je na što bržem umnožavanju i širenju podataka, kao i na što većem prisiljavanju učesnika na poštovanje principa „usluga za uslugu“. To znači da će najviše od P2P mreže profitirati oni korisnici koji najviše budu u nju uložili. BitTorrent protocol je tokom svog kratkog života (prva verzija nastala 2001. godine) dobio veći broj proširenja (unazad kompatibilnih sa osnovnom verzijom protokola) od kojih je, doduše, veoma mali broj zvanično prihvaćen, iako se većina već koristi u velikom broju implementacija. Ovde će biti objašnjena osnovna verzija protokola izložena u [1] i [2], dok će se neka proširenja samo spomenuti.

Pre nego što se pređe na dublje objašnjavanje BitTorrent protokola, treba, u ovom kontekstu, prvo objasniti pojam „torrent“ kao i proces njegovog uspostavljanja. Torrent obuhvata skup datoteka ili podataka koji učesnici međusobom dele BitTorrent protokolom, kao i same učesnika koji ga dele. Kada neki korisnik odluči da želi da podeli neki sadržaj, odnosno skup datoteka, za njega se prvo mora izgenerisati opisna datoteka (eng. *torrent metadata file*) što se može uraditi većinom raspoloživih BitTorrent klijenata. Ova datoteka sadrži neophodne informacije za uspešno deljenje torrent-a, kodovane B kodiranjem (eng. *B encode* ili *bencode*). Podaci koji se nalaze u opisnoj datoteci su npr. veličina podataka koja se deli, informacije o datotekama koje se dele, URL tragača (eng. *tracker*) itd, a može sadržati i neke dodatne informacije. Tragač je poslužilac koji je zadužen za torrent koji treba da se formira. Zadatak tragača je da prati statistiku vezanu za torrent, kao i da prati ko sve učestvuje u deljenju konkretnog torrent-a. On sadrži listu svih učesnika, zajedno sa njihovom IP adresom i prolazom na kojem osluškujaju za nove veze. Kada se izgeneriše opisna datoteka, ona se mora postaviti na neki standardan HTTP poslužilac, kako bi drugi korisnici mogli da ga preuzmu. Kada je ova priprema gotova, deljenje sadržaja može da počne.

Da bi se objasnio sam proces razmene podataka u BitTorrent protokolu, mora se prethodno ukazati na dve vrste učesnika koji u tom procesu učestvuju. Učesnici se dele na učesnike sa kompletnim podacima (eng. *seeder*) i učesnike sa delimičnim podacima (eng. *peer*, u literaturi se

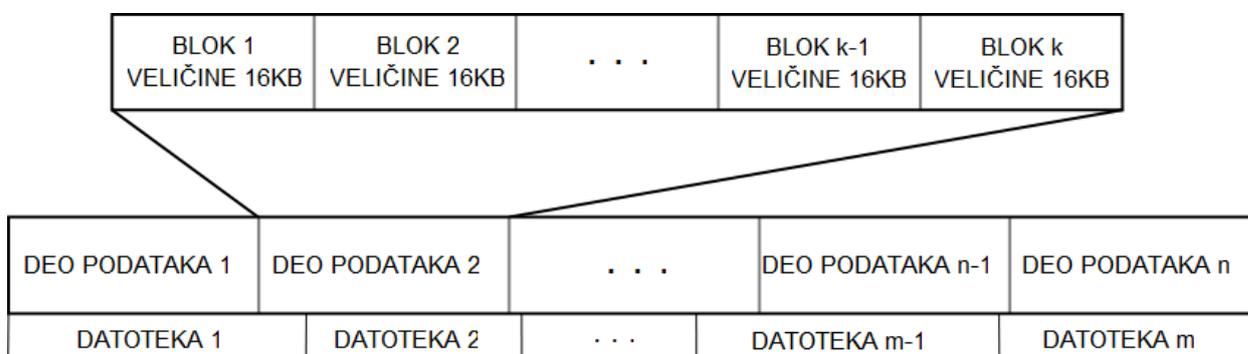
često pojavljuje i *leecher*). Učesnici sa kompletnim podacima su oni koju su ceo deljeni sadržaj već preuzeli i njihova dalja uloga se svodi samo na poslužioca, tj. na dalje deljenje podataka. Učesnici sa delimičnim podacima uzimaju ulogu i klijenta i poslužioca i istovremeno preuzimaju nove podatke i preuzete podatke dele sa drugim učesnicima. Skup svih aktivnih učesnika jednog torrent-a naziva se roj (eng. *swarm*) Kada bilo koja vrsta učesnika želi da se priključi roju torrent-a, učesnik prvo mora da objavi svoje prisustvo (eng. *announce*) tragaču. U osnovnoj verziji protokola, to se vrši HTTP GET zahtevom, sa popunjenim neophodnim atributima od kojih su najbitniji IP adresa učesnika, prolaz na kojem se oslušuju nove dolazne veze, identifikator torrent-a, kao i 20 bajta dugačak identifikator učesnika koji sam učesnik generiše na slučajan način. Identifikator torrent-a je takođe dužine 20 bajta i dobija se generisanjem SHA-1 heš koda određenog dela opisne datoteke torrent-a. Objavljivanje se po pravilu vrši periodično (period specificira tragač i najčešće je par desetina minuta), kako bi tragač znao da je učesnik i dalje aktivan, kao i prilikom napuštanja roja, kako bi tragač znao da je učesnik prestao sa aktivnostima. Po objavljivanju svog prisustva tragaču, učesnik od tragača dobija listu nasumično izabranih učesnika iz roja, zajedno sa njihovim IP adresama i prolazima na kojima oslušuju. Broj učesnika koji tragač šalje zavisi od HTTP GET zahteva, kao i od samog tragača. Po dobijanju liste, učesnik pokušava da uspostavi vezu sa njemu odgovarajućim brojem drugih učesnika. Uspostavljena veza između učesnika je na TCP nivou. Nakon uspostavljenе veze, učesnici dalje komuniciraju peer wire protokolom. Šematski prikaz načina funkcionisanja BitTorrent protokola prikazan je na slici 2.2.



Slika 2.2 Šematski prikaz načina funkcionisanja BitTorrent protokola

## 2.2.2 Deljenje torrent-a

Kako je BitTorrent, pre svega, namenjen deljenju većih količina podataka, efikasnost protokola bi bila znatno manja ako bi svaki učesnik morao prvo da preuzme celokupan sadržaj pre nego što bi bio u mogućnosti da ga dalje deli sa drugim učesnicima. Zbog toga, celokupan sadržaj torrent-a se logički posmatra kao jedna datoteka i logički deli na delove. Svi delovi su iste veličine, koja se tipično kreće između 64 KB i 1 MB, u zavisnosti od količine podataka torrent-a. Veći delovi se obično uzimaju za veće količine podataka, dok kod manjih količina podataka veličina dela smanjuje, jer se dobije izdeljeniji torrent koji je lakše deliti među učesnicima. Poslednji deo je obično manji od ostalih, jer količina podataka najčešće nije jednaka umnošku veličine dela. Kako delovi mogu biti veličine i do 1 MB, preuzimanje celog dela od samo jednog korisnika može potrajati. Ovo se dešava jer često postoji nesimetričnost u propusnosti veze ka i sa internetom. Ovo dalje povećava verovatnoću da će se desiti prekid veze u toku preuzimanja dela i da će sav saobraćaj utrošen na taj deo biti uzaludan. Kako bi se ovi problemi izbegli, delovi se interno, od strane svakog učesnika dele na blokove veličine 16 KB, kako je navedeno u [2]. Ova izdeljena struktura se može videti na slici 2.3, gde  $m$  predstavlja broj datoteka u torrent-u,  $n$  predstavlja broj delova. Umesto razmene celih delova, između učesnika razmenjuju se blokovi. Na taj način gubitak veze neće prouzrokovati gubitak celog dela, a jedan deo može se istovremeno tražiti od više korisnika i time povećati ukupnu brzinu njegovog prenosa. S obzirom na to da se blokovi preuzimaju od raznih drugih učesnika čija se pouzdanost ne zna, nakon preuzimanja celog dela, njegova ispravnost se mora proveriti. Da bi provera bila pouzdana, opisna datoteka torrent-a sadrži SHA-1 heš kodove svih delova torrent-a. Kada učesnik preuzme deo, generiše se njegov SHA-1 heš kod i poredi sa onim u opisnoj datoteci. Ako se kod ne poklapa, deo se odbacuje i mora se preuzeti ponovo, a ako se kod poklapa, deo se smatra ispravnim.



Slika 2.3 Prikaz logičke structure podataka unutar jednog torrent-a

### 2.2.3 Peer wire protokol

Peer wire protokol se nalazi na aplikativnom nivou ISO OSI modela i naleže na TCP protokol. On služi isključivo za komunikaciju između učesnika dok se za komunikaciju između učesnika i tragača koristi, kako je ranije napomenuto, HTTP protokol. Osnovna verzija protokola se sastoji od 9 različitih poruka definisanog formata. Nakon uspostavljanja TCP veze između dva učesnika, razmenjuju se HANDSHAKE poruke. Ova poruka se po formatu jedina razlikuje od ostalih, a ima specifičnu namenu za uspostavljanje veze. Ona sadrži naziv i verziju protokola, heš kod torrent-a (kako bi BitTorrent klijentske aplikacije mogle rukovati sa više torrent-a istovremeno), identifikator učesnika i 8 rezervisanih bajta čija je podrazumevana vrednost nula. Svrha rezervisanih bajta jeste u lakšem proširivanju protokola. Svakom proširenju se dodeljuje jedan jedinstveni bit čija se vrednost postavlja na jedan ako učesnik koji šalje HANDSHAKE poruku to proširenje podržava. Nakon razmenjenih ispravnih (odgovarajući heš kod, ime protokola itd.) HANDSHAKE poruka, smatra se da je veza uspostavljena na nivou peer wire protokola.

Sve ostale poruke su istog formata i počinju sa 4 bajta dugom celobrojnomo vrednošću koja opisuje dužinu poruke (u koju ova 4 bajta nisu uračunata), zatim jedan bajt koji opisuje tip poruke nakon kojeg slede podaci specifični za samu poruku. Sve vrednosti duže od jednog bajta su u formatu big-endian. Formati peer wire poruka se mogu videti na slici 2.4, gde su prikazana polja poruka, kao i njihova veličina u bajtima. Nakon uspostave veze na peer wire nivou, učesnik može poslati BITFIELD poruku koja nosi informaciju o tome koje sve delove torrent-a ima pošiljalac poruke. Svaki deo je označen jednim bitom, redom, gde vrednost 1 obeležava da pošiljalac ima odgovarajući deo, dok vrednost 0 obeležava da odgovarajući deo nedostaje. Ova poruka je validna samo ako se pošalje odmah nakon uspostave veze. Ako učesnik želi naknadno da obavesti ostale učesnike da je preuzeo neki od delova, on to mora učiniti HAVE porukom u kojoj se šalje indeks (početna vrednost je nula) odgovarajućeg dela čija se raspoloživost objavljuje drugim učesnicima.

Format HANDSHAKE poruke

DUŽINA NAZIVA PROTOKOLA	NAZIV PROTOKOLA	REZERVISANI BAJTI	HEŠ KOD TORRENT-A	IDENTIFIKATOR UČESNIKA
1	n	8	20	20

Format ostalih peer wire poruka

DUŽINA PORUKE	TIP PORUKE	PODACI KOJE PORUKA NOSI
4	1	m

Slika 2.4 Formati poruka peer wire protokola

Razmena podataka između učesnika se vrši na zahtev. Učesnik koji želi da preuzme određeni blok od drugog učesnika mora poslati REQUEST poruku u kojoj je specificirano kom delu pripadaju traženi podaci, koliko podataka želi da preuzme (u bajtovima), kao i gde se ti podaci u delu nalaze (odstojanje od nultog bajta, tj. početka dela). Učesnik koji primi zahtev na njega odgovara porukom PIECE koja sadrži traženi blok. U njoj su takođe specificirani indeks dela u kom se poslati blok nalazi i odstojanje bloka unutar tog dela. Ovi podaci su potrebni zato što učesnik može poslati više zahteva za blokove istom ili različitim učesnicima, bez čekanja na odgovor, pa mora znati na koji zahtev mu odgovor stiže. Na ovaj način se brzina prenosa podataka značajno uvećava, jer ne postoji period čekanja iza svakog zahteva, već se to vreme koristi za slanje novih podataka. Ukoliko je učesnik poslao različitim učesnicima zahtev za jedan isti blok, a zatim taj blok od jednog od njih primi, učesnik može da otkáže zahtev preostalim učesnicima kako ne bi bespotrebno trošio saobraćaj svoje internet veze. To se radi porukom CANCEL koja sadrži iste podatke kao REQUEST poruka, da bi se znalo koji se tačno zahtev otkazuje.

Preostale četiri poruke su CHOKE, UNCHOKE, INTERESTED i NOT INTERESTED i njihova namena biće detaljnije objašnjena u sledećem odeljku.

#### **2.2.4 Princip funkcionisanja BitTorrent protokola**

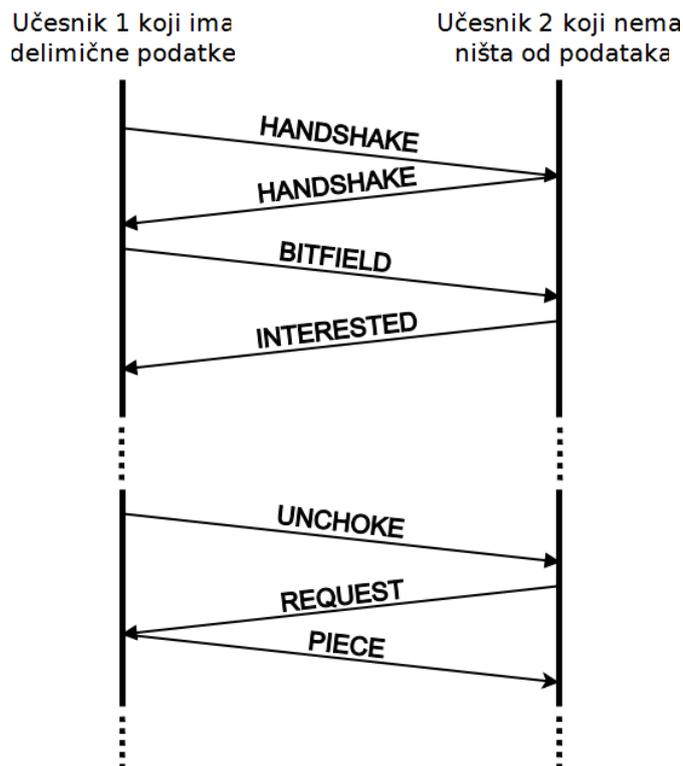
Već je napomenuto da je u BitTorrent protokolu poseban akcenat stavljen na što brže umnožavanje sadržaja i na suzbijanju tzv. „slepih putnika“, odnosno da svaki učesnik ima koristi od mreže proporcionalno tome koliko i sam uloži u nju. Ova dva važna cilja se postižu sa velikom uspešnošću, a za to su odgovorna dva algoritma na kojima se BitTorrent protokol zasniva.

Prvi algoritam se odnosi na odlučivanje koji će se deo sledeći zahtevati od drugih učesnika. Algoritam po kojem se vrši ovo odlučivanje je jednostavan i zove se „prvo najredi“ (eng. *rarest first*). Po njemu, sledeći deo na redu za preuzimanje je uvek onaj koji najmanje na raspolaganju kod učesnika sa kojim je uspostavljena veza. Na ovaj način se osigurava da se sadržaj brzo umnožava i da postoji dovoljno kopija svih delova, odnosno značajno se smanjuje šansa da će u nekom momentu roj ostati bez svih delova torrent-a. Ovaj algoritam i njegove prednosti izložen je u [3].

Drugi algoritam se naziva „algoritam gušenja“ (eng. *choking algorithm*). On je zadužen za eliminisanje „slepih putnika“. Algoritam ima posebne verzije za učesnike sa kompletnim i sa delimičnim sadržajem, a ovde će biti objašnjene obe. Pre nego što se pređe na sam algoritam, treba napomenuti četiri parametra koja svaki učesnik mora održavati za svaku vezu sa drugim učesnicima. Ti parametri su sledeći: „da li gušim“ (eng. *am choking*), „da li me guši“ (eng. *is choling*), „da li sam zainteresovan“ (eng. *am interested*) i „da li je zainteresovan“ (eng. *is*

*interested*). Moguće vrednosti parametara su 0 i 1, gde 0 označava netačan iskaz, a 1 tačan. Parametar zainteresovanosti se može objasniti kroz jednostavan primer: učesnik 1 je zainteresovan za učesnika 2 ako učesnik 2 ima delove koje učesnik 1 nema, tj. želi da preuzme. U ovom primeru, učesnik 1 će za učesnika 2 imati postavljen parametar „da li sam zainteresovan“ na vrednost 1, dok će učesnik 2 za učesnika 1 na vrednost 1 imati postavljen parametar „da li je zainteresovan“. Parametri gušenja se u stvari odnose na dozvolu učesnicima da traže delove od drugih učesnika. I ovaj parametar se može objasniti jednostavnim primerom: učesnik 1 „guši“ učesnika 2 ako učesnik 2 nije dozvolio učesniku 1 da šalje zahteve. Vrednosti parametra kod učesnika 1 i 2 bi bile sledeće: učesnik 1 bi za učesnika 2 na vrednost 1 imao postavljen parametar „da li gušim“, dok bi učesnik 2 za učesnika 1 imao na vrednost 1 postavljen parametar „da li me guši“.

Promene ovih stanja (zainteresovanost i dozvola slanja zahteva) se drugim učesnicima naznačavaju već pomenutim peer wire porukama CHOKe (zabrana slanja zahteva), UNCHOKe (dozvola slanja zahteva), INTERESTED (objavljivanje zainteresovanosti), NOT INTERESTED (objavljivanje nezainteresovanosti). Svaka peer wire veza se inicijalizuje na vrednosti takve da učesnici jedan drugom zabranjuju slanje zahteva i jedan za drugog su nezainteresovani. Preduslov da učesnik počne da dobija podatke od bilo kojeg iz skupa učesnika sa kojima je povezan jeste da je zainteresovan za njega, kao i da je od njega dobio dozvolu za slanje zahteva. Primer jednostavne razmene poruka peer wire protokola predstavljen je na slici 2.5. Ponašanje parametara tokom vremena zavisi od samog algoritma koji je implementiran u klijentskoj aplikaciji, gde će najčešće primenjen algoritam biti objašnjen u sledećem odeljku.



Slika 2.5 Primer jednostavne razmene poruka u peer wire protokolu

### 2.2.5 Algoritam gušenja

Kao što je već napomenuto, algoritam gušenja ima različite verzije za učesnike sa kompletnim podacima i za učesnike sa delimičnim podacima.

Kod učesnika sa delimičnim podacima, učesnik u svakom momentu ima određeni skup učesnika kojima je dozvolio slanje zahteva. Najčešće taj skup čini 4 učesnika. Svakih 10 sekundi, učesnik sortira sve učesnike sa kojima ima uspostavljenu peer wire vezu, gde je kriterijum sortiranja brzina dostavljanja traženih podataka. 4 učesnika koji šalju podatke najvećom brzinom dobijaju dozvolu za slanje zahteva dok je ostalima zabranjeno. Na ovaj način se osigurava da bez slanja podataka, postoji jako mala šansa da će učesnik izvući korist iz mreže. Kako skup od najbolja 4 učesnika ne bi bio konstantan, njemu se svakih 30 sekundi dodaje jedan učesnik kome se daje „optimistična dozvola“ (eng. *optimistic unchoke*). On se bira na nasumičan način. Treba primetiti da je posledica ovog algoritma činjenica da se, u slučaju neslanja podataka, od učesnika sa delimičnim podacima ne mogu dobijati podaci duže od 10 sekundi.

Za učesnike sa kompletnim podacima, postoji dve verzije algoritma koje su najčešće implementirane. U prvoj verziji, učesnik sa kompletnim podacima se ponaša isto kao i učesnik sa delimičnim podacima, ali je kriterijum sortiranja brzina preuzimaja podataka i broj preuzetih delova. To znači da dozvolu dobijaju oni učesnici koji imaju najveću brzinu preuzimanja i najviše preuzetih delova, tj. oni učesnici koji će najpre sami postati učesnici sa kompletnim podacima. Akcenat ove verzije algoritma je na što bržem umnožavanju sadržaja i na uvećanju

broja učesnika koji raspolažu kompletnim podacima, kako bi dalja distribucija bila što brža. U drugoj verziji algoritma, posmatra se period od 30 sekundi, gde je u svakom momentu najviše 4 učesnika data dozvola za slanje zahteva. Svakih 10 sekundi, zainteresovani učesnici koji već imaju dozvolu slanja zahteva se sortiraju po tome ko je kada poslenji put dobio dozvolu za slanje zahteva. Tokom prvih 20 sekundi perioda od 30 sekundi, posle svakog periodičnog sortiranja 3 učesnika koji su poslednji dobili dozvolu dobijaju je ponovo, dok se četvrti učesnik za slanje dozvole bira nasumično iz skupa onih učesnika koji su zainteresovani, ali nemaju dozvolu. Tokom poslednjih 10 sekundi perioda od 30 sekundi, poslednja 4 učesnika koja su imala dozvolu je zadržavaju, tj. nema biranja nasumičnog učesnika. Akcenat ove verzije algoritma je na jednakoj raspodeli vremena svim učesnicima tokom kog oni mogu da zahtevaju i preuzimaju podatke. Efikasnost ovih algoritama je opisana u [4].

Treba napomenuti da, iako su izloženi algoritmi prisutni u velikoj većini implementacija BitTorrent protokola, sama specifikacija protokola ne zahteva njihovu upotrebu.

### **2.2.6 Performanse BitTorrent protokola u praksi**

Iako BitTorrent protokol nema determinističke karakteristike, odnosno učesniku nije garantovana ni brzina preuzimanja, a ni uspešno preuzimanje kompletnog sadržaja torrent-a, ovaj protokol je u praksi ipak pokazao dobre performanse. U radu izloženom u [5], autori daju rezultate analize studije koja je trajala pet meseci tokom koje je posmatran torrent čiji je sadržaj bio veličine 1,77 GB. Studija je pokazala da je BitTorrent visoko efikasan protokol kada se posmatraju brzine prenosa podataka između učesnika. Pokazano je, takođe, da je protokol vrlo otporan na nalete velikog broja učesnika (eng. *flash-crowd*), tj. da tokom naleta performanse ne opadaju. U radu je takođe istaknuto da, iako postoji velik broj učesnika koji nije uspeo da preuzme kompletan sadržaj, razlog neuspelog preuzimanja nije do protokola, već do samih korisnika koji su odlučili da odustanu. Na to ukazuje činjenica da je vreme njihovog zadržavanja u roju značajno manje od prosečnog vremena potrebnog za preuzimanje kompletnih podataka, kao i to da ukupna količina podataka koju su ovi učesnici preuzeli čini samo mali procenat (oko 10%) ukupnog saobraćaja koji je među učesnicima napravljen tokom studije.

Ako se BitTorrent protokol uporedi sa uobičajenim rešenjima po modelu „klijent – poslužilac“, može se uočiti da je kapacitet BitTorrent protokola mnogo veći. Sa porastom broja učesnika kapacitet roja raste, dok kod tradicionalnih rešenja kapacitet zavisi od poslužioca. Trenutno jedino teorijsko usko grlo BitTorrent protokola (koje jos nije dostignuto) predstavljaju tragači donosno njihovo preopterećenje, ali kako je njihov zahtev za resursima veoma mali, a zahvaljujući nekoliko proširenja protokola, ovi zahtevi su dodatno smanjeni, ovo teorijsko

---

ograničenje trenutno ne predstavlja problem. Takođe, zbog malih potreba za resursima, jedan tragač može da opslužuju učesnike velikog broj torrent-a.

Iako BitTorrent protokol nije u potpunosti eliminisao poslužioce i njihov problem jedne tačke otkaza, ovaj problem je ipak sveden na minimum. Jedini poslužioc u sistemu je tragač, a njegova uloga je važna samo pri inicijalnom uključenju učesnika u roj. Ako bi u bilo kom momentu došlo do otkaza tragača, svi učesnici koji su već u roju bi bez problema nastavili sa deljenjem sadržaja, dok bi jedino novi učesnici bili onemogućeni da se priključe. Ovaj problem je dodatno sveden na minimum kroz nekoliko proširenja protokola. Jedno od njih je proširenje koja dozvoljava zaduženje više tragača za jedan isti torrent, dok nekoliko drugih proširenja dozvoljavaju razmenu informacija o učesnicima (IP adresa i prolaz) među samim učesnicima, na više načina. Na ovaj način i u slučaju otkaza jednog tragača, učesnici imaju na raspolaganju druge.

Još jedan od dokaza rasprostranjenosti i efikasnosti BitTorrent protokola je podatak izvučen iz studije kompanije CacheLogic koji govori da je BitTorrent protokol u proseku odgovoran za negde oko 35% celokupnog internet saobraćaja.

## 3. Analiza problema

Problemi koji su se pojavili tokom izrade ovog rada mogu se podeliti u dve grupe: jedna grupa su problemi vezani za realizaciju programske podrške za peer wire protokol, dok drugu grupu čine problemi vezani za prilagođenje BitTorrent protokola specifičnoj nameni uz očuvanje kompatibilnosti.

### 3.1 Analiza problema vezanih za programsku podršku

Jedan od najvećih problema prilikom realizacije programske podrške jeste velik nedostatak informacija vezanih za ovaj protokol. Specifikaciju protokola održava kompanija BitTorrent Inc. čiji je suosnivač njegov tvorac Bram Cohen. Međutim, specifikacija je već odavno zastarela. Jedno vreme je specifikacija protokola bila održavana kroz otvoren kod BitTorrent klijenta razvijenog od strane kompanije, ali od verzije 6, kod klijenta više nije otvoren, pa je jedini preostao zvaničan izvor zastarela specifikacija. Dodatni problem je činjenica da je najveći deo sadržaja koji je deljen BitTorrent protokolom u stvari sadržaj zaštićen autorskim pravima (muzika, filmovi, aplikacije, video igre, itd.) pa je njegova distribucija tog sadržaja zabranjena. Posledica ovoga je da velik broj strana koje pripadaju komercijalnom svetu ulaže velike napore u umanjivanje popularnosti protokola i njegovo onemogućavanje. Kao posledica ovoga javlja se njegova jako slaba komercijalna upotreba, a samim tim i manjak motivacije za pružanje veće količine detaljnijih informacija kao i manjak motivacije za pisanje naučnih radova na ovu temu kojih ima relativno malo.

Drugi problem čine već pomenuta proširenja protokola. Uprkos tome što je mali broj proširenja prihvaćen zvaničnom specifikacijom, velik broj se koristi kod većine klijentskih aplikacija, pa su na neki način nezvanično postala standard. Trebalo je odrediti minimalan skup proširenja (kako bi se što manje vremena utrošilo na realizaciju programske podrške) koji će ipak

omogućiti nesmetan rad i kompatibilnost razvijene klijentske aplikacije sa ostalim klijentskim aplikacijama i tragačima.

Do sada navedeni problemi su uzrokovali konstatno dopunjavanje i unapređivanje realizovane klijentske aplikacije dok se nije došlo do zadovoljavajućeg i prihvatljivog rezultata.

Na kraju, testiranje samog rešenja je bilo problem za sebe. Kako bi izrada pogodnog okruženja za testiranje zahtevala dodatne napore i vreme (izrada tragača, kao i klijentskih aplikacija drugih učesnika), rešenje je od samog početka testirano u realnom okruženju. Dodatno otežanje predstavlja odlika pojedinih tragača da odbijaju HTTP GET zahteve učesnika ako su se učesnici javljali tokom određenog prethodnog perioda čiju dužinu sami tragači određuju. Usled ovog učestanost izvršavanja testova bila je znatno manja.

## **3.2 Analiza problema vezanih za prilagođenje BitTorrent protokola**

Cilj ovog rada bio je razvoj klijentske BitTorrent aplikacije koja će deljeni sadržaj apstrahovati kao tok podataka i na taj način pokušati da ga reprodukuje korisniku. To znači da reprodukcija sadržaja mora početi pre njegovog kompletnog preuzimanja, tj. u BitTorrent protocol, koji je sam po sebi dosta stohastičan, treba uneti što veću dozu determinizma. Važno je napomenuti i jedan dodatni, vrlo važan kriterijum ovog rada, a to je izrada aplikacije za potencijalnu komercijalnu upotrebu. Kako je velik deo sadržaja ilegalan za distribuciju, kao posledica kriterijuma nameće se isključivanje deljenja podataka iz same podrške protokolu, tj. realizovana programska podrška mora biti u stanju da efikasno preuzima podatke, ali ih ne sme dalje deliti sa drugim korisnicima. Realizovana klijantska aplikacija se mora ponašati kao „slepi putnik“. Ovaj zahtev dalje nameće nekoliko ograničenja vezanih, kako za realizaciju, tako i za primenu. U prethodnom poglavlju izložen je algoritam gušenja i pokazano je da je vrlo efikasan u suzbijanju „slepih putnika“. Ako se posmatra veza između jednog takvog učesnika i jednog fer učesnika sa delimičnim sadržajem, fer učesnik neće slati podatke „slepom putniku“, jer nikakve podatke ne dobija. U najboljem slučaju dozvoliće slanje zahteva optimističnom dozvolom, ali to je period od samo 10 sekundi posle kojih podaci fer učesnika više neće biti dostupni. Očigledno je da se na učesnike sa delimičnim sadržajem ne može računati kao dugoročan izvor podataka. Preostaju još učesnici sa kompletnim podacima i kod njih je situacija drugačija. Kako oni ne preuzimaju podatke ni od koga, njihov kriterijum biranja učesnika za slanje podataka je drugačiji. U zavisnosti od primenjenog algoritma, učesnici sa kompletnim podacima će ili slati svima podjednako ili će im prioritet biti oni učesnici koji će najpre uspeti da preuzmu kompletan sadržaj, tj. oni koji preuzimaju podatke najvećom brzinom i koji najviše delova imaju u svom posedu. U rojevima gde većinu učesnika čine oni sa delimičnim sadržajem ovo će predstavljati

problem jer će i većinu učesnika sa liste dobijene od tragača činiti učesnici sa delimičnim sadržajem koji, u našem slučaju, nisu pogodan izvor podataka. Na osnovu ovoga može se zaključiti da je najpogodnije okruženje „zasićeni“ roj ili roj gde većinu čine učesnici sa kompletnim sadržajem. U ovakvom okruženju algoritam gušenja je najmanje efikasan, dok će većinu učesnika sa liste dobijene od tragača činiti učesnici sa kompletnim sadržajem. Kako bi se ovo okruženje potpuno iskoristilo, razvijena programska podrška mora postizati maksimalnu moguću brzinu prenosa prilikom preuzimanja podataka, kako bi klijentska aplikacija bila favorizovana od strane drugih učesnika (koji će pretežno biti oni sa kompletnim sadržajem). To znači da prioritet u razvoju programske podrške mora biti razvoj što veće brzine prilikom prenosa. Takođe važna odlika razvijene programske podrške mora biti transparentnost ka drugim učesnicima. Nijedan učesnik ni u kom momentu ne sme posumnjati da razvijena klijentska aplikacija predstavlja učesnika koji je „slepi putnik“.

Pošto sadržaj koji se preuzima mora što pre početi sa reprodukcijom, klijentska aplikacija ne može priuštiti da delove preuzima nasumično niti po algoritmu „prvo najređi“, već ih mora preuzimati redom, počevši sa prvim. Ovo predstavlja značajan problem u poređenju sa standardnim klijentskim aplikacijama. Standardnim klijentskim aplikacijama nije bitno koje blokove dobijaju dokle god priliv podataka postoji, pa privremena stagnacija priliva podataka od strane nekog učesnika ne predstavlja problem, jer nema vremenskih ograničenja, pa će blok kad tad stići. Za to vreme klijentska aplikacija može preuzimati blokove od drugih učesnika i time probati da nadoknadi manjak priliva podataka. U slučaju ciljne klijentske aplikacije, delovi koji su prvi po redosledu imaju prioritet i na te delove se ne može čekati jer su potrebni za uspešnu reprodukciju sadržaja. Jedno od rešenja koje se prvo nameće je slanje zahteva za prioriternim delovima svim učesnicima koji ga imaju. Iako bi to pomoglo u dobavljanju zahteva na vreme, izazvalo bi određenu količinu viška saobraćaja, a kako je saobraćaj nedeterminističan, treba iz svakog učesnika izvući maksimum, bez nepotrebnih zahteva, jer se nikad ne zna da li će taj učesnik u nekom momentu odlučiti da želi nekom drugom da šalje podatke. Krajnje rešenje mora biti u stanju da dobavi odgovarajuće delove na vreme, uz minimalan višak zahteva i saobraćaja.

## 4. Koncept rešenja

U prethodnom poglavlju izloženi su problemi koji su se javili prilikom izrade ovog rada, a u ovom poglavlju će se izneti implementirani predlozi rešenja za navedene probleme.

Treba napomenuti da je, zbog oskudnosti informacija o protokolu, bilo neophodno naći dodatne informacije. Ovo je delimično postignuto i posmatranjem postojećih klijentskih aplikacija (najviše uTorrent aplikacije), kako statistike koju pruža sama aplikacija, tako i saobraćaja koji se tokom njenog rada stvara. Za posmatranje saobraćaja korišćena je aplikacija Wireshark.

### 4.1 Izbor proširenja protokola za implementaciju

Prvi problem koji je trebalo rešiti pre nego što se uopšte započelo sa razvojem programske podrške protokolu bio je problem određivanja minimalnog skupa proširenja protokola koja će se implementirati. U ovaj skup ušlo je tri proširenja. Prvo se odnosi na novi tip tragača koji, umesto HTTP-a, za komunikaciju sa učesnicima koristi UDP protokol. Cilj ovog proširenja je rasterećenje tragača, jer je saobraćaj ka njemu i od njega u ovom slučaju mnogo manji (oko 50%), a uvedena je iz razloga što su ovi tipovi tragača vremenom postali dosta češći i popularniji u odnosu na klasične HTTP tragače. Njihova zastupljenost je porasla do te mere da pojedini torrent-i nisu uopšte vezani za HTTP tragače, već samo za tragače novog tipa, pa je proširenje bilo neophodno implementirati zbog kompatibilnosti. Drugo proširenje je vezano za format liste učesnika koja se dobija od tragača pri objavljivanju prisustva i takođe je namenjeno rasterećenju tragača odnosno smanjenju saobraćaja od tragača ka učesnicima. I ovo proširenje implementirano je radi bolje kompatibilnosti, jer pojedini tragač samo novi format liste podržavaju. Treće i poslednje proširenje je vezano za opisnu torrent datoteku. U inicijalnoj verziji protokola, torrent je mogao biti vezan samo za jednog tragača koji je bio naveden u opisnoj datoteci. Sa ovim proširenjem tvorca torrent-a ima izbor da u opisnu datoteku stavi listu

od više tragača kod kojih je torrent registrovan. Na ovaj način, u slučaju otkaza jednog tragača, postoje alternative i, što je u našem slučaju veoma važno, učesnik može lakše doći do veće liste učesnika sa kojima može da uspostavi vezu. Važnost ove osobine će biti objašnjena u narednim odeljcima.

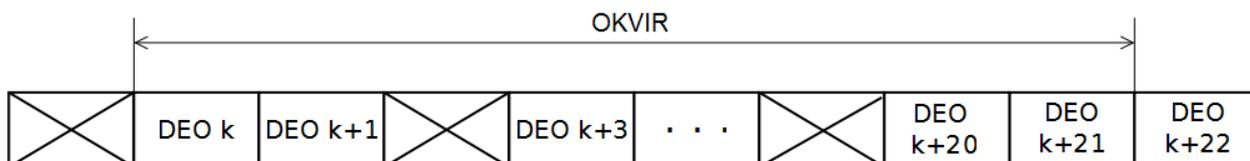
Ostala proširenja su bila ili komplikovana za realizaciju ili nisu donosila prednosti dovoljno bitne za specifičnu primenu aplikacije da bi se na njih utrošilo dodatno vreme.

## 4.2 Prilagođenje protokola specifičnoj primeni

Kako bi se podaci deljeni BitTorrent protokolom uspešno reprodukovali kao tok podataka, implementirano je nekoliko mehanizama osmišljenih za tu primenu. Možemo ih podeliti na mehanizme vezane za delove i mehanizme vezane za učesnike, gde je svrha svih ili povećanje brzine ili dobavljanje pojedinih delova na vreme za njihovu reprodukciju.

### 4.2.1 Implementirani mehanizmi vezani za delove i zahteve

Preuzimanje sadržaja počinje sa određenim maksimalnim brojem delova čiji blokovi mogu biti u potražnji. Iskustvo je pokazalo da je za većinu primena dovoljno da se istovremeno zahtevaju blokovi 20 različitih delova. Tih 20 delova čini „okvir“ koji se tokom preuzimanja kreće po sadržaju torrent-a pri čemu se traže samo blokovi koji pripadaju delovima u „okviru“. Međutim, to nije klasičan „okvir“. Kako se redosled pristizanja blokova ne može predvideti jer se oni istovremeno traže od strane više učesnika, česte su situacije kada prvi završeni deo „okvira“ ne bude i prvi deo po redosledu, tako da „okvir“ u stvari čine 20 delova čiji se blokovi trenutno zahtevaju, koji ne moraju po redosledu biti jedan iza drugog, ali koji teže da taj uslov ispune, tj. da razmak u redosledu između njih bude minimalan. Prikaz okvira dat je na slici 4.1.



☒ - delovi koji su preuzeti

☐ - delovi koji su u toku preuzimanja

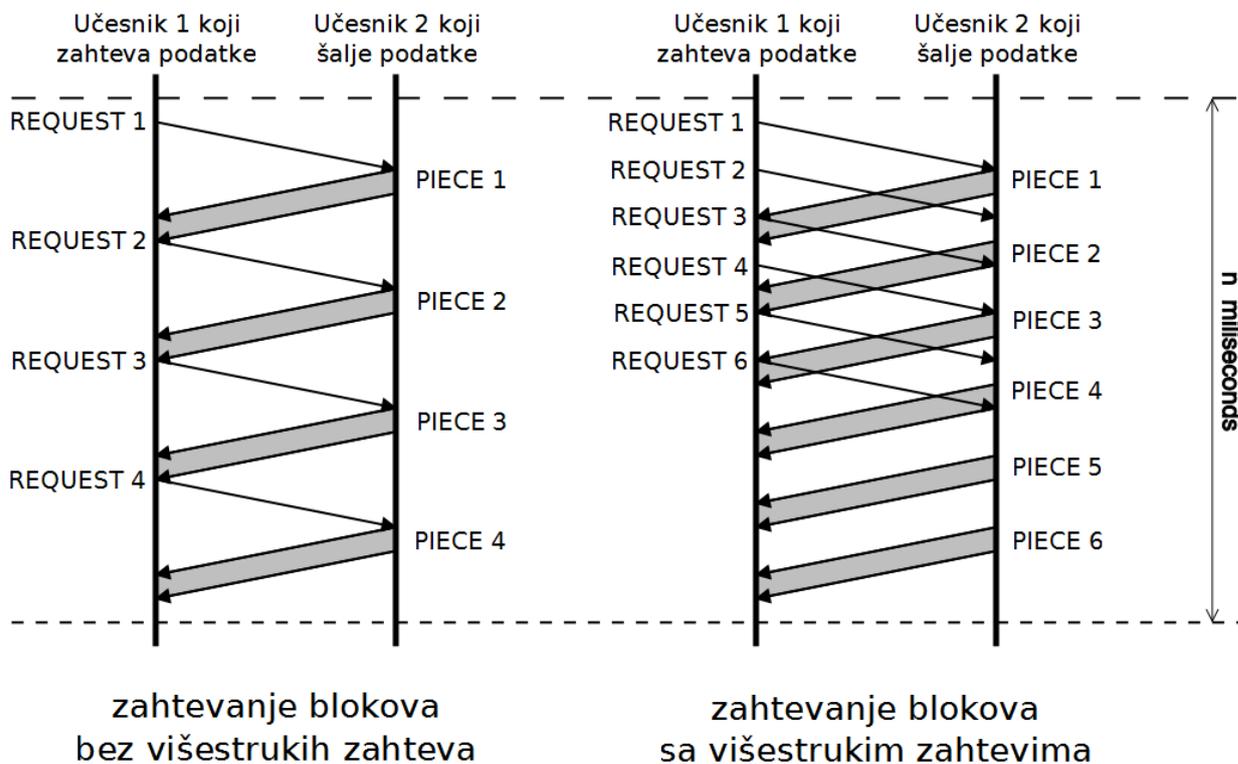
Slika 4.1 Prikaz okvira tokom preuzimanja delova podataka

Pošto ograničenje broja delova ograničava i broj blokova koji se mogu zahtevati, u slučaju da dođe do situacije da su svi blokovi svih delova „okvira“ zahtevani od učesnika, maksimalan broj delova u „okviru“ se povećava za jedan, kako bi se iskoristili moguće kapaciteti zahteva, tj. „okvir“ je po potrebi fleksibilan. Sa druge strane, povećavanje okvira povećava i potrošnju operativne memorije, pa veličina okvira ima i gornju granicu koja je 50 MB utrošene memorije.

Istovremenim zahtevanjem što većeg broja različitih blokova postiže se veća brzina. Kako je u prethodnom poglavlju napomenuto, istovremeno zahtevanje istog bloka od više različitih učesnika može izazvati višak saobraćaja, pa je ovakve slučajeve potrebno svesti na minimum. Ovo se postiže uvođenjem prioriteta za delove koji pripadaju „okviru“. Svaki deo za čije je blokove poslat zahtev, počinje sa nultim prioritetom. Prioritet, u stvari, označava od koliko se dodatnih učesnika traže blokovi koji pripadaju odgovarajućem delu. Tako, na primer, ako deo ima prioritet 2, njegovi blokovi se traže od 3 različita učesnika odnosno od jednog prvobitnog učesnika i dva dodatna. Da bi se što više smanjio višak saobraćaja, prioritet veći od nule može imati samo deo koji je po redosledu prvi u „okviru“, odnosno onaj deo koji je za reprodukciju najhitniji. Prioritet odgovarajućeg dela se podiže samo u slučaju da njegovo preuzimanje traje dva puta duže od vremena potrebnog da se isti deo reprodukuje. Nakon ovog perioda, smatra se da deo dovoljno kasni, i njegov prioritet se podiže. Prilikom podizanja prioriteta, traže se samo blokovi koji nedostaju, već preuzeti blokovi se, naravno, ne zahtevaju opet. Kada se blok prioritelnog dela konačno preuzme od nekog učesnika, ostalim učesnicima od kojih je blok tražen se šalje poruka CANCEL da bi se zahtev poništio i sprečio nepotreban saobraćaj. Na ovaj način, hitni delovi se dobavljaju na vreme, dok se višak saobraćaja, koji se time može napraviti, smanjuje na minimum. Takođe se postiže da se svi blokovi, osim prioritelnih, traže od samo jednog učesnika.

Radi postizanja veće brzine prilikom preuzimanja podataka, korišćena je tehnika „višestrukih zahteva“ (eng. *queuing*). Ona se svodi na slanje višestrukih zahteva za različite blokove jednom istom učesniku. Ovim se postiže da učesnik koji šalje blokove ne mora posle svakog bloka da čeka na novi zahtev pri čemu se vreme koje može da se troši na slanje podataka delimično troši na čekanje novog zahteva. Umesto toga, razvijena klijentska aplikacija se trudi da svaki povezani učesnik u svakom trenutku ima nekoliko zahteva više od onoga što može da ispuni. Na taj način se eliminiše čekanje na učesnika na zahtev i iskorištava se kapacitet veze u potpunosti. Primer razmene peer wire poruka prilikom zahtevanja blokova sa i bez upotrebe ove tehnike prikazan je na slici 4.2. Na njoj se jasno vidi da je količina preuzetih podataka za isti vremenski period znatno veća prilikom upotrebe tehnik višestrukih zahteva u odnosu na količinu podataka dobijenu bez njene upotrebe. Sa druge strane, kapacitet veze se razlikuje od učesnika do učesnika. Shodno tome i broj zahteva koji svaki učesnik dobije ne treba da bude isti. Ako bi se svima slalo malo zahteva, kapaciteti pojedinih veza ne bi bili iskorišćeni, a ako bi se svima slalo mnogo zahteva neki blokovi bi nepotrebno bili zahtevani od učesnika koji taj zahtev neće nikad ili duže vreme ispuniti. Zbog toga se svakom učesniku inicijalno šalje 20 zahteva. Performanse učesnika se mere svake sekunde. Ako učesnik uspe da u protekloj sekundi bude dovoljno efikasan da mu ostane manje od 10 neodgovorenih zahteva, maksimalan broj zahteva

koji mu se šalje se uvećava za 10. Sa druge strane, ako je učesnik neefikasan i ako mu posle protekle sekunde ostane više od 10 neodgovorenih zahteva, maksimalan broj zahteva koji mu se šalje se umanjuje za 1. Na ovaj način, ako se pojavi učesnik sa vezom velikog kapaciteta, njegov kapacitet se relativno brzo ispuni, kratke oscilacije u vezi se ne kažnjavaju velikim umanjivanjem maksimalnog broja zahteva i zahtevi se na troše na učesnike sa vezom slabog kapaciteta.



Slika 4.2 Prikaz razmene poruka prilikom zahtevanja blokova bez i sa primenom tehnik višestrukih zahteva

#### 4.2.2 Implementirani mehanizmi vezani za učesnike

Jedna od prvih ideja koje su implementirane jeste ideja o proceni „kvaliteta“ učesnika sa kojima je klijentska aplikacija u vezi. Pri tome „najkvalitetniji“ su oni učesnici koji su najskorije bili aktivni i od kojih je preuzeto najviše podataka. Prilikom odabira učesnika kome će se uputiti zahtev za sledeći blok, uvek prednost imaju „najkvalitetniji“, dok se ne dostigne njihov maksimalan broj zahteva, kada se prelazi na manje kvalitetne učesnike. Takođe, pri zahtevanju blokova koji pripadaju delu čiji je prioritet povišen, dodatni učesnici od kojih se ti blokovi traže uvek su učesnici najboljeg „kvaliteta“.

Maksimalan broj učesnika sa kojim se klijentska aplikacija može povezati je 20. Testiranje je pokazalo da je ovaj broj za većinu primena dovoljan i optimalan. Da se zahtevi ne bi konstantno „trošili“ na neaktivne učesnike (učesnike koji ne šalju podatke) i da bi se umesto njih dala šansa novim učesnicima, skup učesnika sa kojim je klijentska aplikacija povezana se

konstantno menja. Svakom povezanom učesniku se meri vreme neaktivnosti, tj. vreme od kada je poslao poslednju peer wire poruku. Ako je učesnik neaktivan predugo, veza sa njim se prekida i pokušava se uspostavljanje veze sa novim učesnikom. Kako bi ovo bilo moguće, pri samoj inicijalizaciji torrent-a, formira se „rezerva učesnika“. Ona se formira tako što se prilikom objavljivanja prisustva tragačima traži lista sa što više učesnika. Jedan deo informacija se iskoristi za formiranje inicijalnog skupa učesnika, dok se ostatak informacija čuva za dalju upotrebu, ako dođe do potrebe da se učesnici menjaju. Ovim se postiže da se nakon nekog vremena formira tim učesnika koji većinom šalju podatke, pa brzina preuzimanja podataka postaje relativno stabilna.

Ovde treba napomenuti i to da, iako je uloženi napor da realizovana klijentska aplikacija što više poštuje BitTorrent protokol, ona ga ipak, iz već objašnjenih razloga, blago narušava. Naime, peer wire protokol zahteva da svaki učesnik obaveštava učesnika sa kojima je povezan o delovima koje ima u posedu. Za to služe HAVE i BITFIELD poruke. Realizovana aplikacija ispunjava samo deo ove dužnosti, tj. o ovome obaveštava samo učesnika sa kompletnim podacima. Učesnici sa delimičnim delovima se ne obaveštavaju, kako ne bi nepotrebno slali zahteve, dok se obaveštenja šalju učesnicima sa kompletnim zahtevima kako bi se, po mogućstvu, podigla „popularnost“ (po jednoj varijanti izloženog algoritma gušenja) klijentske aplikacije kod njih.

### **4.3 Regulacija reprodukcije sadržaja**

Jedan od zadataka aplikacije bila je i regulacija reprodukcije preuzetog sadržaja. Algoritam primenjen je krajnje jednostavan. Kada se preuzme dovoljno sadržaja, tako da delovi koji su po redosledu jedan za drugim (počevši od prvog) – kontinualni preuzeti podaci, u zbiru daju bar 10% veličine ukupnog sadržaja, pokreće se reprodukcija. Ako u bilo kom trenutku nakon toga razlika između količine podataka iskorištenih za reprodukciju i kontinualnih preuzetih podataka (razlika na početku reprodukcije je 10% ukupne veličine podataka) padne na manje od 1% ukupne veličine sadržaja, reprodukcija se privremeno zaustavlja. Kada se ista razlika ponovo popne na više od 2% ukupne veličine podataka, reprodukcija se nastavlja.

U ovaj deo aplikacije je, nažalost, uloženo malo vremena, pa se ,shodno tome, izloženi algoritam nije pokazao kao najbolje rešenje. Njegov značaj se uvideo tek nakon testiranja, kada je bilo prekasno za velike izmene. Unapređenjem ili promenom ovog algoritma dobilo bi se dosta na ugodaju korisnika prilikom korišćenja realizovane klijentske aplikacije.

## 5. Testiranje i rezultati

Realizovana klijentska aplikacija je testirana u realtivno kontrolisanom okruženju prilikom čega je analizirano preuzimanje podataka sedam različitih torrent-a, čije su se odlike razlikovale ili u bitskoj brzini sadržaja (eng. *bitrate*) ili u karakteristikama roja. Operativni sistem na kojem je vršeno testiranje je PC linux (distribucija Ubuntu 11.04). Za komunikaciju sa internetom korištena je veza maksimalne brzine 10 Mbit/s za preuzimanje podataka i 1 Mbit/s za slanje podataka, gde je testni računar sa internetom bio povezan bežično. Za svrhe ograničavanja protoka korištena je linux aplikacija Wondershaper.

U ovom radu smatra se da je doživljaj korisnika kvalitetniji ako je vreme čekanja na početak reprodukcije manje, kao i ako su pauze tokom reprodukcije dešavaju što kraće, a njihova učestanost što manja. Najboljim doživljajem se smatra reprodukcija bez prekida, dok čekanje na početak reprodukcije može maksimalno iznositi 180 sekundi (3 minuta). U cilju sticanja što boljeg utiska upotrebljivosti aplikacije uz što veći kvalitet doživljaja korisnika, analizirana su četiri aspekata njenih performansi:

- zavisnost kvaliteta doživljaja od odnosa propusnosti veze i bitske brzine video sadržaja
- poanšanje aplikacije u rojevima sa različitim karakteristikama
- mogućnost razvijanja dovoljno velike brzine preuzimanja podataka za kontinualnu reprodukciju video sadržaja
- količina napravljenog suvišnog saobraćaja tokom preuzimanja sadržaja

### 5.1 Zavisnost kvaliteta doživljaja od odnosa propusnosti veze i bitske brzine video sadržaja

Kako je aplikacija namenjena prvenstveno preuzimanju i reprodukciji video sadržaja, prvi aspekt koji je analiziran jeste mogućnost razvijanja dovoljno velike brzine preuzimanja podataka

,pri ograničenoj propusnosti veze, da bi se mogao kontinualno reprodukovati video sadržaj. Testiranje je vršeno tako što se sadržaj istog torrent-a preuzimao više puta ali sa različitim ograničenjem propusnosti veze ka internetu. Kako su potrebe aplikacije za slanjem podataka veoma male, ograničenje se postavljalo samo na maksimalnu brzinu preuzimanja podataka. Torrent čiji se sadržaj preuzimao je imao sledeće karakteristike:

- veličina sadržaja koji se preuzimao je 340 MB, trajanje reprodukcije 2597 sekundi (43 minuta i 17 sekundi), a prosečna bitska brzina 1072 kbit/s (ili 134 KB/s)
- roj je bio dobro zasićen sa 1246 učesnika sa kompletnim sadržajem i 25 učesnika sa delimičnim sadržajem (ovi podaci su samo okvirni, jer se karakteristike roja konstantno menjaju)

Testiranjem je pokušano da se utvrdi koja je propusnost veze neophodna da bi se sadržaj testnog torrent-a reprodukovao kontinualno. Rezultati testiranja dati su u tabeli 1. Za svako izloženo ograničenje izvršeno je 3 testa, dok su u tabeli izložene srednje vrednosti posmatranih parametara.

<b>POSTAVLJENO OGRAIČENJE BRZINE</b>	<b>VREME DO POČETKA REPRODUKCIJE</b>	<b>ZBIR SVIH PAUZA TOKOM REPRODUKCIJE</b>	<b>UKUPAN BROJ PAUZA TOKOM REPRODUKCIJE</b>
1300 kbit/s	290,33 s	297,33 s	8
1500 kbit/s	222,66 s	147,33 s	2,66
1800 kbit/s	154,33 s	21,33 s	0.66
2000 kbit/s	120 s	0 s	0

Tabela 1 Srednje vrednosti posmatranih parametara tokom testiranja zavisnosti kvaliteta doživljaja korisnika od odnosa propusnosti veze i bitske brzine sadržaja

Dobijeni rezultati pokazuju da, iako je bitska brzina sadržaja 1072 kbit/s, za kvalitetniji ugođaj potrebna znatno veća propusnost veze ka internetu, jer se najbolji doživljaj (bez prekida reprodukcije) dobija tek sa propusnošću od 2000 kbit/s. Iz rezultata se takođe može zaključiti da je kvalitet učesnika u roju relativno loš. Naime, iako je prosečna brzina preuzimanja podataka u poslednja dva slučaja morala biti negde oko 1072 kbit/s, tokom testiranja primećeno je da trenutna brzina značajno osciluje. To znači da je ona deo vremena bila ispod minimalne neophodne brzine, dok se u nekim periodima taj manjak nadonadio brzinom znatno većom od minimalne neophodne. Iz toga se može zaključiti da, iako je velika većina učesnika u roju bila ona sa kompletnim sadržajem, klijentska aplikacija nije uspela da nađe učesnike koji su bili dugotrajni pouzdani izvori podataka sa stabilnom brzinom slanja. Na loš kvalitet učesnika u roju

ukazuje i činjenica da je prilikom preuzimanja sadržaja ovog torrent-a sa vezom ka internetu bez ograničenja (10 Mbit/s) maksimalna postignuta brzina preuzimanja podataka iznosila oko 2800 kbit/s. Čak i pri uslovima bez ograničenja, brzina nije bila stabilna. Zbog čestih iscilacije brzine tokom preuzimanja podataka, idealni uslov za rad aplikacije jeste propusnost veze ka internetu značajno veća od minimalne neophodne brzine preuzimanja podataka, kako bi se, kada to uslovi dozvole, pad brzine preuzimanja podataka mogao nadokanditi.

Važno je napomenuti i to da nemaju svi rojevi učesnike sa ovakvim karakteristikama. Iako su oscilacije u brzini skoro uvek prisutne, one u zavisnostiod učesnika mogu biti manje ili veća. Da je kvalitet učesnika u roju bio bolji, vrlo je verovatno da bi se najbolji doživljaj mogao postići i sa sporijom vezom ka internetu.

## 5.2 Ponašanje aplikacije u rojevima sa različitim karakteristikama

Pod karakteristikama roja se prvenstveno misli na odnos učesnika sa kompletnim sadržajem i učesnika sa delimičnim sadržajem, kao i njihov zbir, tj. ukupan broj učesnika u roju. Da bi se analizirao ovaj aspekt performansi aplikacije, posmatrano je njeno ponašanje tokom preuzimanja sadržaja 4 različita torrent-a, gde je svaki imao različite karakteristike roja. Ovim testovima pokušano je da se utvrdi da li je aplikacija upotrebljiva i za torrent-e sa rojevima slabijih karakteristika i ako jeste, kakav je doživljaj korisnika. Prilikom ovih testova, ograničenje na brzinu preuzimanja podataka sa interneta nije postavljano, tj. bilo je 10 Mbit/s. Karakteristike video sadržaja testiranih torrent-a predstavljene su u tabeli 2:

<b>UKUPNA VELIČINA SADRŽAJA</b>	<b>TRAJANJE REPRODUKCIJE SADRŽAJA</b>	<b>PROSEČNA BITSKA BRZINA SADRŽAJA</b>
529 MB	1389 s (23 min i 9 s)	3120 kbit/s
354 MB	2554 s (42 min i 34 s)	1135 kbit/s
421 MB	2613 s (43 min i 33 s)	1320 kbit/s
148 MB	1431 s (23 min i 51 s)	847 kbit/s

Tabela 2 Karakteristike video sadržaja torrent-a korišćenih za testiranje ponašanja aplikacije u rojevima sa različitim karakteristikama

Karakteristike rojeva testiranih torrent-a, kao i rezultati testova mogu se videti u tabeli 3. Za svaki torrent vršena su tri testa, radi verodostojnijih rezultata, dok su u tabeli izložene srednje

vrednosti praćenih parametara tokom tri testa. Redosled rezultata u tabeli tri odgovara redosledu torrent-a u tabeli 2.

<b>BROJ KORISNIKA SA KOMPLETNIM / DELIMIČNIM SADRŽAJEM</b>	<b>VREME DO POČETKA REPRODUKCIJE</b>	<b>ZBIR SVIH PAUZA TOKOM REPRODUKCIJE</b>	<b>UKUPAN BROJ PAUZA TOKOM REPRODUKCIJE</b>
730/10	72,66 s	0	0
320/407	67,33 s	0	0
55/114	160	0	0
12/1	231,33 s	>1431	38,33

Tabela 3 Karakteristike rojeva testiranih torrent-a i srednje praćenih parametara tokom testova

Rezultati testiranja prvog torrent-a su očekivani i aplikacija se dobro ponašala u roju u kome je velika većina učesnika imala kompletan sadržaj i služila isključivo kao izvor podataka. Može se reći i da su učesnici bili kvalitetni, s obzirom na to da brzina preuzimanja podataka u pojedinim momentima dostizala i maksimum korištene veze ka internetu, te je sadržaj preuzet sva tri puta bez ikakvih prekidanja reprodukcije.

Rezultati testiranja drugog i trećeg torrent-a nisu toliko očekivani, ali su dobri i ukazuju na to da se realizovana klijentska aplikacija dobro ponaša i u rojevima u kojima je broj učesnika oba tipa podjednak, kao i u rojevima u kojima broj učesnika sa delimičnim sadržajem preovladava. Iz ovog se može izvesti zaključak da je aplikaciji sasvim dovoljno da roj sadrži dovoljan broj (iz rezultata testiranja se vidi da to može biti samo nekoliko desetina) kvalitetnih učesnika sa kompletnim sadržajem kako bi reprodukcija prošla bez prekida a kvalitet doživljaja korisnika bio najbolji. Razlika proteklog vremena do početka reprodukcije između ova dva torrent-a se može objasniti malom količinom učesnika sa kompletnim sadržajem (koji su aplikaciji glavni izvor podataka) u trećem torrent-u, pa je aplikaciji trebalo više vremena da nađe dovoljnu količinu učesnika sa kojom može da razvije zadovoljavajuću brzinu preuzimanja podataka.

Prilikom testiranja četvrtog torrent-a, čiji roj ima jako mali broj učesnika, aplikacija od tri testa samo u prvom testu uspela da preuzme sadržaj dovoljno brzo da zbir pauza bude manji od trajanja reprodukcije sadržaja. U naredna dva testa, učesnika koji je bio primarni izvor podataka najverovatnije je napustio roj, jer su rezultati drastično lošiji. Iako je aplikacija u sva tri testa uspešno preuzela kompletan sadržaj, ugođaj korisnika je veoma loš, pa se može smatrati da realizovana aplikacija nije uspela da ispuni svoju namenu i da je za ovakve slučajeve slabo

primenljiva. Ipak, ovakvi slučajevi su retki i ne predstavljaju osnovne slučajeve kojima je aplikacija namenjena.

### **5.3 Mogućnost razvijanja dovoljno velike brzine preuzimanja podataka za kontinualnu reprodukciju video sadržaja**

U ovom odeljku razmatra se sposobnost aplikacije da razvije velike brzine preuzimanja podataka dovoljne i za kontinualnu reprodukciju video sadržaja velikih bitskih brzina, tj. sadržaj rezolucije 1080p (full HD). Za ovu analizu mogu se iskoristiti testovi izloženi u prethodnom odeljku

Ako se pogledaju testovi prva tri torrent-a iz prethodnog odeljka, može se zaključiti da realizovana aplikacija bez problema razvija minimalnu prosečnu brzinu preuzimanja podataka potrebnu za kontinualnu reprodukciju, kako video sadržaja manjih bitskih brzina (oko 1100 kbit/s) tako i za video sadržaj većih bitskih brzina (oko 3100 kbit/s). Tako je sadržaj prvog testiranog torrent-a iz prethodnog odeljka reprodukovano bez prekidanja reprodukcije, sa relativno malim čekanjem na početak reprodukcije, pri čemu je video sadržaj torrent-a rezolucije 1080p sa frekvencijom od 23,8 slika u sekundi kodovan H264 koderom. Važno je napomenuti da je aplikacija tokom preuzimanja sadržaja ovog torrent-a uspela da postigne brzinu preuzimanja podataka od preko 9500 kbit/s, tj. da iskoristi preko 95% propusnosti veze ka internetu, pa se može reći da su uslovi u kojima je aplikacija radila bili vrlo pogodni. Iz toga se može zaključiti da je aplikacija u stanju da sa visokim kvalitetom doživljava korisnika preuzima i reprodukuje video sadržaj visokog kvaliteta, ako uslovi to dozvoljavaju, gde su to pre svega karakteristike roja i kvalitet učesnika u roju.

### **5.4 Količina napravljenog suvišnog saobraćaja tokom preuzimanja sadržaja**

Kao poslenji važan aspekt procene performansi aplikacije daje se uvid u količinu suvišnog saobraćaja koji aplikacija napravi prilikom preuzimanja sadržaja nekog torrenta. Radi lakšeg izlaganja, korišćiće se podaci dobijeni u testovima koji su analizirani u prethodna dva odeljka za torrent-e čiji je sadržaj opisan u tabeli 2.

U tabeli 4 izložene su srednje vrednosti količine suvišnih bajtova koji su preuzeti tokom izvršavanja svakog od tri testa po torrent-u.

UKUPNA VELIČINA SADRŽAJA	KOLIČINA SUVIŠNOG SAOBRAĆAJA	ODNOS VIŠKA I UKUPNE VELIČINE SADRŽAJA
529 MB	7,59 MB	1,43%
354 MB	3,81 MB	1,08%
421 MB	10,06 MB	2,39%

Tabela 4 Pregled količine suvišnog saobraćaja napravljenog tokom preuzimanja sadržaja testiranih torrent-a

Iz tabele se vidi da se količina viška saobraćaja kreće od 1-3% ukupne veličine sadržaja torrent-a. Kako se višak sadržaja generiše samo prilikom slanja istih zahteva različitim učesnicima prilikom dobavljanja prioriternih delova, može se zaključiti da cena dobavljanja delova na vreme nije velika i da se može tolerisati.

## 5.5 Ukupni rezultati

Na osnovu priloženih rezultata testova može se zaključiti da se aplikacija dobro ponaša u različitim uslovima, čak i u onim za koje se to nije očekivalo. Kvalite doživljaja korisnika najviše zavisi od kvaliteta učesnika u roju torrent-a, kao i od samih karaktersitika roja, jer je pokazano da je razvijena klijentska aplikacija u stanju da se nosi i video sadržajem većih bitskih brzina. Za negativnu osobinu aplikacije može se smatrati njena potreba za većom propusnošću veze ka internetu od same bitske brzine sadržaja koji se preuzima. U situacijama kada su ove dve veličine bliske po vrednosti, kvalitet doživljaja korisnika je znatno lošiji. Shodno ovome, predlog glavnog smera daljeg napretka aplikacije bio bi razvoj boljeg algoritma za kontrolu reprodukcije, koji bi uzeo u obzir i brzinu preuzimanja podataka. Kako su u spornim situacijama pauze učestanije i duže, smatra se da bi se kvalitet doživljaja korisnika poboljšao ako bi inicijalno čekanje na početak reprodukcije bilo produženo, kako pauze u reprodukciji nakon toga u potpunosti sprečile. Novi algoritam regulisanja reprodukcije sadržaja morao bi da teži ovom rešenju.

## 6. Zaključak

U ovom radu prikazan je drugačiji pristup BitTorrent protokolu u cilju boljeg doživljaja korisnika prilikom njegovog korišćenja. Prikazane su upotrebljene tehnike koje je autor rada implementirao kako bi što bolje unapredio performanse realizovane klijentske aplikacije. S obzirom na to da je programska podrška razvijana od početka, stečeno je veliko iskustvo i znanje vezano, kako za BitTorrent protokol, tako i za mreže i mrežno programiranje uopšte.

Iako BitTorrent dosad nije imao komercijalnu upotrebu, rezultati testiranja ove implementacije pokazuju da se BitTorrent i tim uslovima može prilagoditi, iako sa nešto smanjenom upotrebnom vrednošću. Autor rada veruje da bi se dodatnim testiranjem i implementacijom nešto naprednijih algoritama i problem upotrebne vrednosti sveo na minimum.

Dalji smerovi razvoja bi mogli biti razvoj boljeg algoritma za regulisanje reprodukcije sadržaja kao razvoj okruženja za testiranje (kontrolisani učesnici, tragači itd.) čime bi se omogućila bolja analiza problema koji se javljaju. Ovim bi dobijeni rezultati bili verodostojniji i otvorila bi se mogućnost za razvijanje naprednijih i pouzadnijih algoritama za regulisanje parametara od kojih brzina prenosa podataka u BitTorrent protokolu zavisi. Takođe, određeno vreme moglo bi se uložiti na realizaciju dodatnih proširenja protokola, koja bi dalje poboljšali komptibilnost i performanse aplikacije.

Aplikacija je realizovana sa prenosivošću u vidu, tako da dodatne biblioteke nisu korišćene i aplikacija se oslanja samo na linux systemske pozive koji se već duže vreme podržani u izdanjima linux kernela.

Na kraju, treba napomenuti da zvaničnih projekata ove vrste ima vrlo malo (u momentu izrade rada), pa mesta za napredak ima mnogo. Sa realizovanom osnovnom programskom podrškom za sam protokol (u kojoj takođe ima prostora za razvoj), veća pažnja se može posvetiti daljem prilagođavanju BitTorrent protokola ovoj specifičnoj nameni.

## 7. Literatura

- [1] Zvanična specifikacija BitTorrent 1.0 protokola i njegovih proširenja, <http://www.bittorrent.org/>, učitano 18.07.2012.
- [2] Nezvanična, ali detaljnija specifikacija BitTorrent 1.0 protokola, <http://wiki.theory.org/BitTorrentSpecification>, učitano 18.07.2012.
- [3] Bram Cohen: *Incentives Build Robustness in BitTorrent*, Berkeley, USA, Jun 2003.
- [4] Arnaud Legout, G. UrvoyKeller and P. Michiardi: *Rarest First and Choke Algorithms Are Enough*, INRIA, Sophia Antipolis, Septembar 2006.
- [5] M. Izal, G. Urvoy-Keller, E.W. Biersack, P.A. Felber, A. Al Hamra, and L. Garcés-Erice: *Dissecting BitTorrent: Five Months in a Torrent's Lifetime*, PAM '04, Antibes Juan-les-Pins, France, April 2004.