Peer to peer mreže

Markovinović, Eugen

Sveučilište J.J. Strossmayera u Osijeku

Odjel za matematiku Sveučilišni preddiplomski studij matematike

Eugen Markovinović

Peer to peer mreže

Završni rad

Osijek, Rujan 2015.

Sveučilište J.J. Strossmayera u Osijeku

Odjel za matematiku Sveučilišni preddiplomski studij matematike

Autor: Eugen Markovinović

Voditelj: prof.dr.sc. Domagoj Matijević

Peer to peer mreže

Završni rad

Osijek, Rujan 2015.

Sažetak:

Ovaj rad obrađuje peer to peer mreže i njihovu upotrebu na konkretnim primjerima. Počevši od samog pojma peer to peer preko povijesti i razvoja do peer to peer-a danas. Vidljivi su mnogi načini provedbe peer to peer-a te njihova učinkovitost. Uspoređujući peer to peer s drugim sustavima, postaju vidljive i prednosti i mane peer to peer-a. Mane peer to peer-a su očite i kod pitanja sigurnosti. Također peer to peer je glavno piratsko sredstvo za raspodjelu ilegalnih kopija proizvoda. Međutim, mnogobrojne aplikacije i protokoli koji koriste peer to peer pokazuju njegovu popularnost, stoga je u ovom uratku obrađen najpoznatiji protokol koji koristi peer to peer, BitTorrent. Objašnjen je njegov način rada te popularne fraze koje su u upotrebi kod korisnika BitTorrenta. Na kraju je obrađena zastupljenost peer to peer-a na internetu u nedavnoj prošlosti gdje je vidljivo koliko je zapravo upotrebljiv u današnja vremena.

Ključne riječi: poslužitelj, klijent, mreža, BitTorrent

Summary:

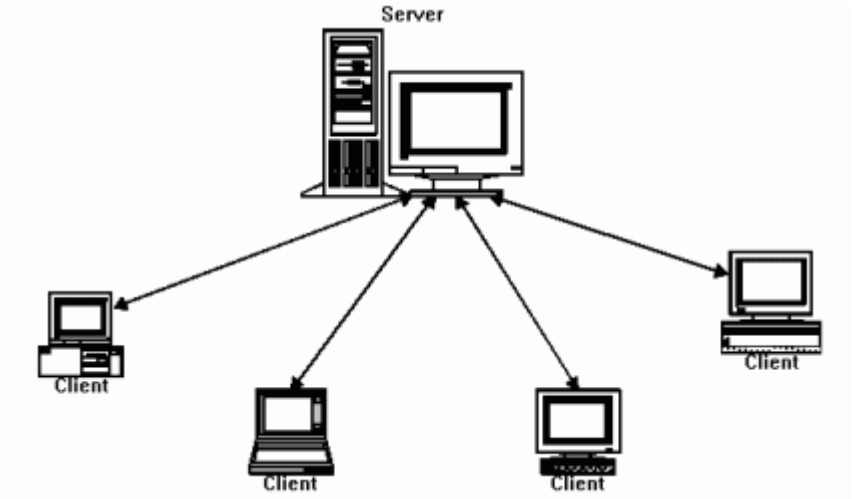
This work examines peer to peer networks and their usage on specific examples. Starting with the concept of peer to peer, it goes on trough history and ends with peer to peer today. You can see many ways of making peer to peer happen and how efficient those ways are. By comparing peer to peer with other systems all the advantages and flaws of peer to peer become clear. Flaws of peer to peer are obvious when it comes to the question of security as well. Moreover, peer to peer is the main tool pirates use for distribution of their illegal copies of products. However, the numerous applications and protocols that use peer to peer show it's popularity, therefore the most famous peer to peer protocol, BitTorrent, is closely examined in this short work. The way how BitTorrent works is explained with addition of some popular phrases that the users of BitTorent use. In the end, the representation of peer to peer on the Internet in recent past is explored, where it becomes clear how usefull peer to peer nowadays actually is.

Key words: server, client, network, BitTorrent

Sadržaj:

1. Uvod

Povećanjem količine podataka koji se razmjenjuju putem interneta javila se potreba za novijom mrežnom strukturom koja će omogućiti bržu razmjenu podataka i rasterećenje same mreže. Zadatak ovog seminara je opisati peer to peer (p2p) mrežnu strukturu i njezinu primjenu te prednosti i mane. Sam pojam peer to peer zapravo označava način komunikacije unutar mreže. Ustaljeni oblik mrežne komunikacije je bio takozvani klijent – poslužitelj, gdje je poslužitelj jedinstven, te klijenti komuniciraju samo s poslužiteljem (Slika 1.1).



Slika 1.1 Klijent-Poslužitelj

Ideja da se omogući međusobna komunikacija klijenata je temelj peer to peer mrežne strukture. Ukratko, peer to peer struktura je struktura u kojoj poslužitelj više nije središte preko kojeg se vrši sva izmjena podataka, već je omogućeno ostalim članovima mreže da međusobno komuniciraju i razmjenjuju podatke.

U drugom poglavlju se govori o nastanku ideje peer to peer mreže i daljnjem razvoju. Poglavlje tri govori o načinu provedbe peer to peer sustava te vrstama. Četvrto Poglavlje uspoređuje klijent-poslužitelj s peer to peer sustavom na primjeru aplikacije za razmjenu datoteka. Peto poglavlje ukratko upozorava na sigurnosne probleme peer to peer sustava te zloupotrebu. Poglavlje šest opisuje način rada BitTorrenta. Sedmo poglavlje nabraja primjere upotrebe peer to peer sustava. Poglavlje osam istražuje blisku prošlost peer to peer sustava i njegovu zastupljenost. Konačno, deveto poglavlje daje zaključak ovog rada te govori nešto o budućnosti peer to peer-a.

2.Povijest

Sama ideja peer to peer načina rada se javlja već u samim počecima interneta. Preteča interneta, Advanced Research Projects Agency Network (ARPANET), koju je razvijalo američko Ministarstvo obrane šezdesetih godina dvadesetog stoljeća, radilo je na peer to peer konceptu jer nije bilo poslužitelja, već se razmjena podataka obavljala direktno između računala priključenih na ARPANET. Trebalo je proći dosta godina dok nije nastao USENET 1979. Koji je zapravo bio već spomenuti klijent – poslužitelj sustav. Međutim kako je bilo više poslužitelja u jednoj mreži i oni su komunicirali međusobno, tako da se može reći da je to donekle bio peer to peer sustav. Kroz daljnje godine, razvitkom interneta, dominantan sustav je bio klijent – poslužitelj. To se mijenja pojavom Napstera 1999. To je bio sustav za razmjenu glazbe. Napster je doduše imao poslužitelja, ali on je imao ulogu spajanja dva klijenta koji bi onda međusobno razmjenjivali podatke. Tako da se može reći da je Napster prvi pravi peer to peer sustav na internetu. Nažalost, pod pritiskom tužbi Napster je ne sretno završio i bankrotirao. Nakon Napstera dolazi Gnutella koja je funkcionirala bez poslužitelja, a za spajanje na nju je bila potrebna adresa bilo kojeg računala unutar mreže. Međutim, pravu revoluciju u peer to peer mrežama je donio BitTorrent protokol za razmjenu podataka kojeg je razvio Bram Cohen. Veliko rasterećenje i brzina mreže se postiže u njemu tako da svi povezani klijenti koji imaju određenu datoteku je šalju u dijelovima onima koji traže tu datoteku. Time je poslužitelj znatno rasterećen te se brzina razmjene povećava ovisno o broju korisnika. Nakon BitTorrenta razvijeni su i brojni slični protokoli, preko kojih se danas odvija veći promet od onoga koji koriste klijent – poslužitelj servisi. Možemo slobodno reći da su peer to peer sustavi zaposjeli internet na desetak godina.

3.Arhitektura

Peer to peer se može provesti na više različitih načina, pa općenito koristimo taj naziv gdje god je omogućena komunikacija između klijenata.

Peer to peer se dijeli na:

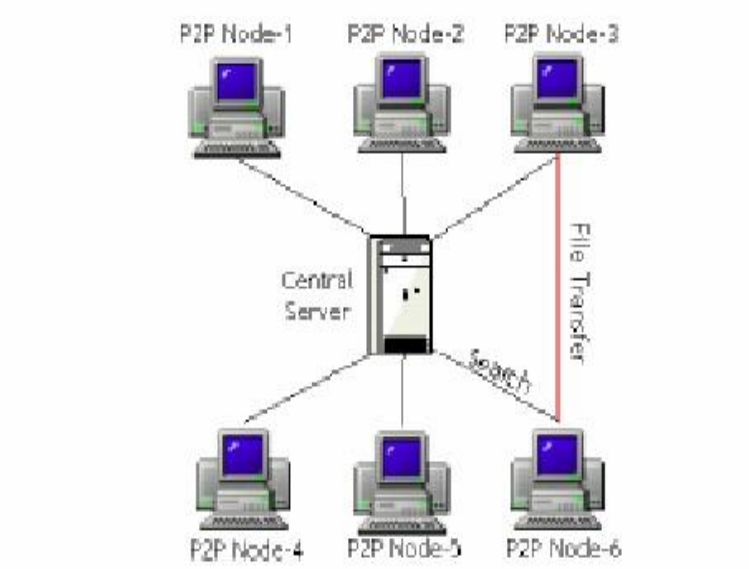
1. Centralizirane P2P sustave
2. Decentralizirane P2P sustave
3. Hibridne P2P sustave

Dalje, decentralizirane P2P sustave se još može podijeliti na:

1. Strukturirane P2P sustave
2. Nestrukturirane P2P sustave.

3.1 Centralizirani P2P sustavi

Centralizirani P2P sustavi su oni kod kojih poslužitelj ipak igra neku ulogu. Poslužitelj je tu da povezuje klijente koji onda međusobno nastavljaju komunikaciju. Primjer takvog sustava je već navedeni Napster. Poslužitelj bi imao bazu podataka s datotekama koje ima određeni klijent, te kada bi se novi klijent povezao na poslužitelja i zatražio određenu datoteku, poslužitelj bi mu poslao adresu klijenta koji posjeduje traženu datoteku te bi se skidanje te datoteke nastavilo direktno između klijenata (Slika 3.1). Velika prednost ovog sustava je brzina kojom se nalazi tražena datoteka, jer ih poslužitelj sve ima u bazi podataka koja se redovito ažurira. Najveća mana ovakvog sustava je da sve ovisi o poslužitelju. U slučaju kvara na poslužitelju, cijeli sustav pada i ne funkcionira.



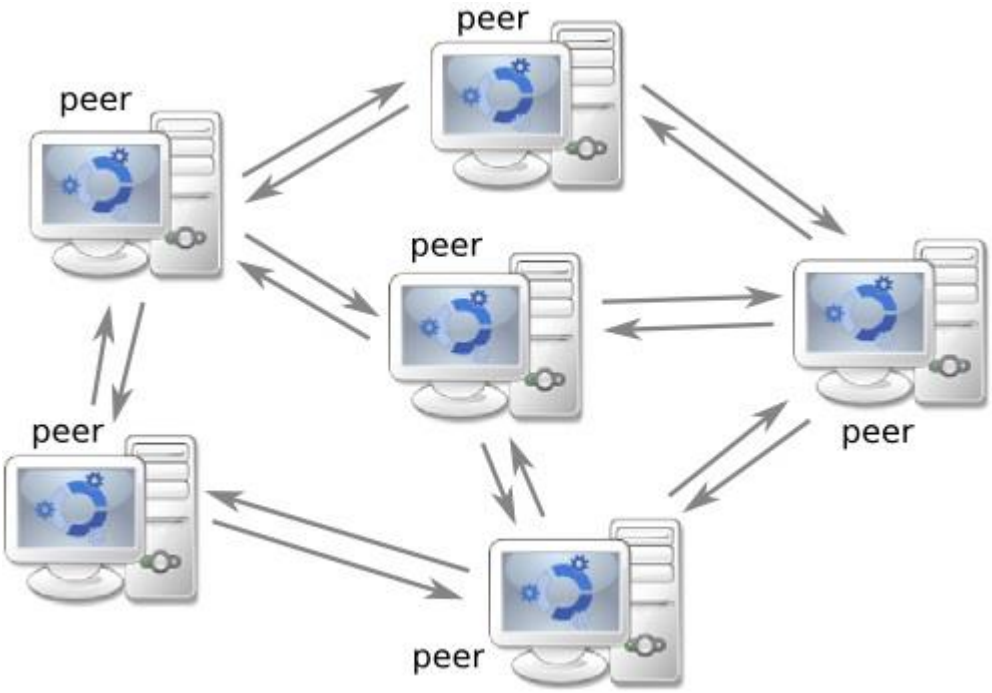
Slika 3.1 Centralizirani P2P

3.2 Decentralizirani P2P sustavi

Glavna karakteristika decentraliziranih peer to peer sustava je da nemaju istaknutog poslužitelja. Njihova podjela na strukturirane i nestrukturirane je rezultat povezanosti same strukture mreže s čvorovima koji se nalaze u njoj.

3.2.1 Nestrukturirani P2P sustavi

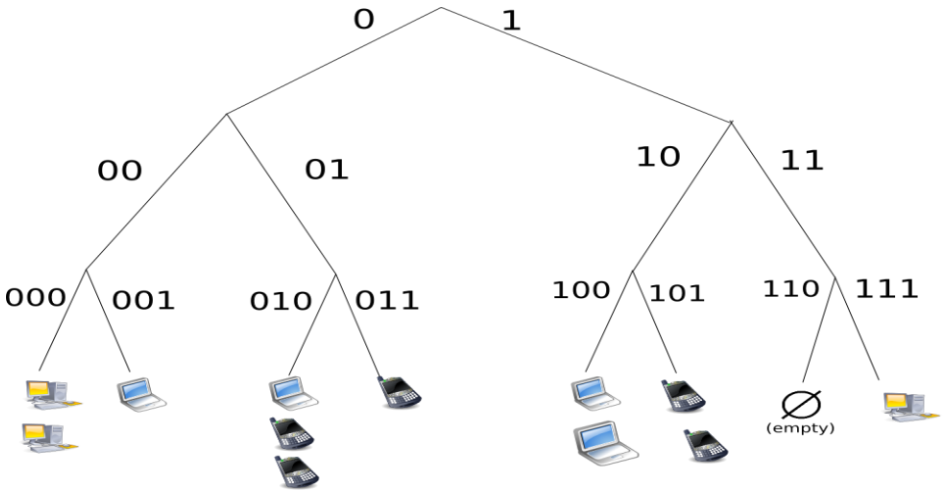
Kod nestrukturiranih P2P sustava su podaci koji se nalaze na pojedinom čvoru proizvoljni. Za traženje određenog podatka se najčešće mora obići veliki broj čvorova jer ne znamo točno gdje se traženi podatak nalazi. Ovakvi sustavi često koriste emitiranje kao način pretrage. Emitiranjem se šalje upit na susjedni čvor (čvor s kojim je povezan) koji, ukoliko nema traženi podatak, šalje upit na svoj susjedni čvor i tako dalje dok se ne prođu svi čvorovi ili dok se ne nađe traženi podatak. Upravo to je i velika zamjerka ovakvih sustava, jer kod velikog broja upita dolazi do preopterećenja mreže. Ipak, prednost je što ovakav sustav neće pasti ako se jedan od čvorova isključi ili pokvari, već će se prilagoditi i nastaviti sa radom za razliku od centraliziranog sustava. Primjer sustava koji koristi ovakvu strukturu je Gnutella.



Slika 3.2 Nestrukturirani P2P

3.2.2 Strukturirani P2P sustavi

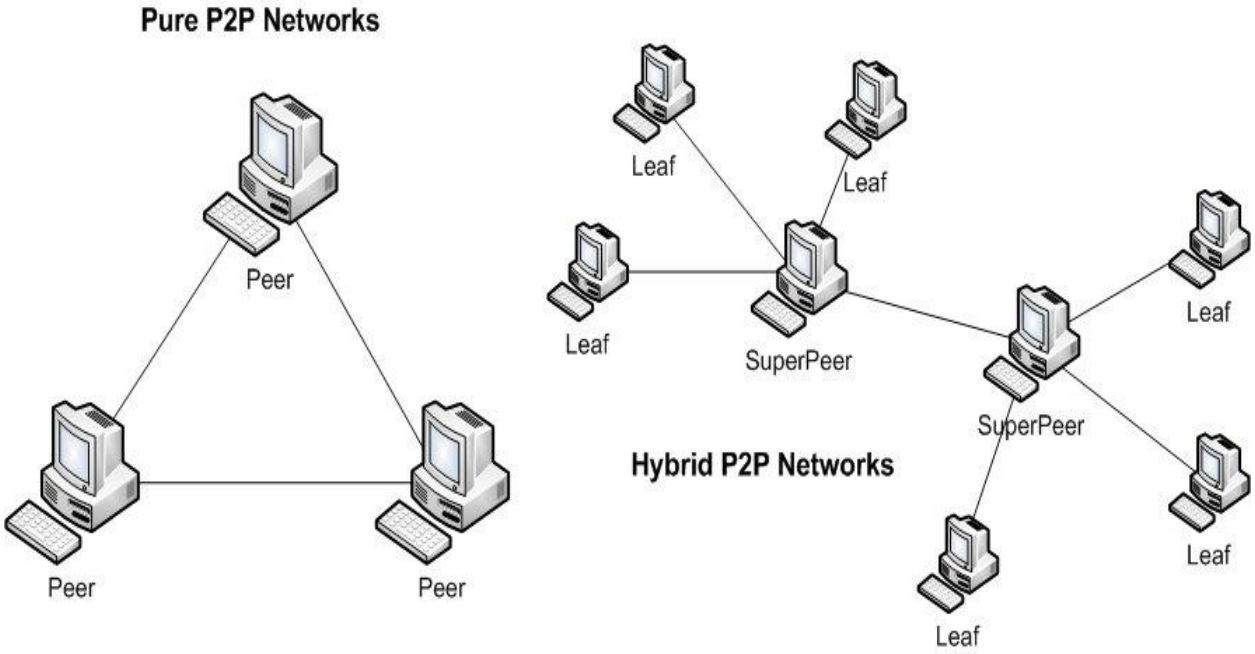
Strukturirani P2P sustavi imaju točno određen mehanizam za traženje određenih podataka. Drugim riječima, podaci su spremljeni na točno određene lokacije. Sustav funkcionira i dolaskom novih podataka i klijenata tako da prilagođava mehanizam te se sama struktura mreže ne mijenja. Jedan od načina na koji se ovo može postići je pomoću hash funkcija na imenima podataka i čvorova. Podatke se stavlja na čvorove koji imaju približno sličnu hash vrijednost kao i dani podaci. Traženje određenog podatka u ovakvoj mreži je brzo i efikasno, no ovakve strukture imaju velike probleme kod prilagođavanja novom klijentu i podatku. Primjer ovakvih struktura je Chord protokol, koji je bio jedan od prvih koji su koristili ovakvu strukturu i hash funkcije.



Slika 3.3 Strukturirani P2P

3.3. Hibridni P2P sustavi

Hibridni P2P sustavi su mješavina centraliziranih i decentraliziranih sustava. Koriste vrlo zanimljiv koncept tzv. „Superčvorova“ koji imaju sve normalno kao i „obični“ čvorovi, međutim rade značajan posao kod pretrage za određenim podatkom. Slično kao što kod centraliziranih P2P sustava poslužitelj ima bazu podataka, tako i superčvorovi uzmu dio „običnih“ čvorova iz mreže i naprave listu svih datoteka tih čvorova. Kada neki čvor traži određeni podatak, prvo šalje upit svom nadređenom superčvoru. Taj superčvor zatim gleda u svoju bazu podataka i ukoliko ne posjeduje traženi podatak vrši emitiranje među ostalim superčvorovima. Ovim načinom se koriste sve prednosti centraliziranih i decentraliziranih sustava.



Slika 3.4 Obični i hibridni P2P

Jedan primjer hibridnog P2P sustava je spotify, švedska kompanija za prikazivanje glazbe i videa.

4. Usporedba P2P i K-P sustava

Usporedit ćemo peer to peer sustav s klijent – poslužitelj sustavom na temelju razmjene datoteka. Razmotrimo model u kojem poslužitelj ima jednu datoteku i želi ju podijeliti s n klijenata. Označimo brzinu uploada poslužitelja s u_s (u bitovima po sekundi), a brzinu uploada i-tog klijenta s u_i, $i\epsilon\{1,2,\dots,n\}$. Isto tako označimo brzinu downloada pojedinog klijenta s d_i, $i\epsilon\{1,2,\dots,n\}$. Još označimo veličinu datoteke s F, u bitovima. Vrijeme raspodjele datoteke je vrijeme koje je potrebno da poslužitelj pošalje svim klijentima datoteku. Sada promotrimo koliko je vrijeme raspodjele u slučaju kada se radi o klijent – poslužitelj sustavu, označit ćemo to vrijeme s D_{kp}. Primijetimo da poslužitelj mora poslati kopiju datoteke svim klijentima, to znači da mora poslati nF bitova. Kako je brzina poslužiteljevog uploada u_s, vrijeme da pošalje sve kopije je barem $^{nF}/_{u_s}$ sekundi. Označimo s d_{min} najsporiju brzinu downloada, $d_{min} = min\{d_i\}, i\epsilon\{1,2,\dots,n\}$. Klijent s najsporijom download brzinom ne može dobiti cijelu datoteku u manje od $^{F}/_{d_{min}}$ sekundi. Kada povežemo ovo dvoje dobijemo:

$$D_{kp} \geq \max\{{}^{nF}/_{u_s}, {}^{F}/_{d_{min}}\}.$$

Uzmimo donju granicu gornje nejednadžbe kao najmanje moguće vrijeme raspodjele za klijent – poslužitelj:

$$D_{kp} = \max\{{}^{nF}/_{u_s}, {}^{F}/_{d_{min}}\} \qquad (4.1)$$

Vidimo da za dovoljno velik n vrijeme raspodjele klijent – poslužitelj sustava raste linearno, i dano je s $^{nF}/_{u_s}$. Znači da ukoliko broj klijenata naraste tisuću puta, vrijeme raspodjele će biti tisuću puta veće.

Napravimo sada sličnu analizu za peer to peer sustav, ali ovoga puta ćemo koristiti sličnu stvar kao i BitTorrent, klijenti će pomagati poslužitelju u raspodjeli datoteke. Drugim riječima, kada

klijent primi podatke od servera, on će primljene podatke slati dalje, drugim klijentima koji ih nemaju. Primijetimo da je vrijeme raspodjele datoteke kod peer to peer sustava malo teže za izračunati zbog dodatnih parametara.

Na početku samo poslužitelj ima datoteku, te ju on mora poslati barem jedanput. Za razliku od klijent – poslužitelj sustava gdje poslužitelj mora više puta slati datoteku, ovdje je dovoljno jedno slanje jer klijenti dalje među sobom preraspoređuju i šalju podatke. Koristeći oznake od prije, vrijeme raspodjele datoteke će po tome biti barem $^{F}/_{u_s}$.

Kada gledamo sa strane klijenta koji ima najsporiju download brzinu, on ne može primiti cijelu datoteku u manje od $^{F}/_{d_{min}}$ sekundi.

Napokon, promotrimo ukupnu upload brzinu sustava. Ona je jednaka zbroju brzine servera i svih klijenata, $u_{ukupno} = u_s + u_1 + \cdots + u_n$. Znamo da sustav mora dostaviti F bitova svakom od n klijenata, sveukupno nF bitova. Raspodjela tih bitova se ne može obaviti brže od u_{ukupno} što znači da je vrijeme raspodjele barem $^{nF}/_{u_{ukupno}}$.

Složimo li ove tri točke zajedno dobivamo minimalno vrijeme raspodjele peer to peer sustava, označimo ga s D_{P2P}.

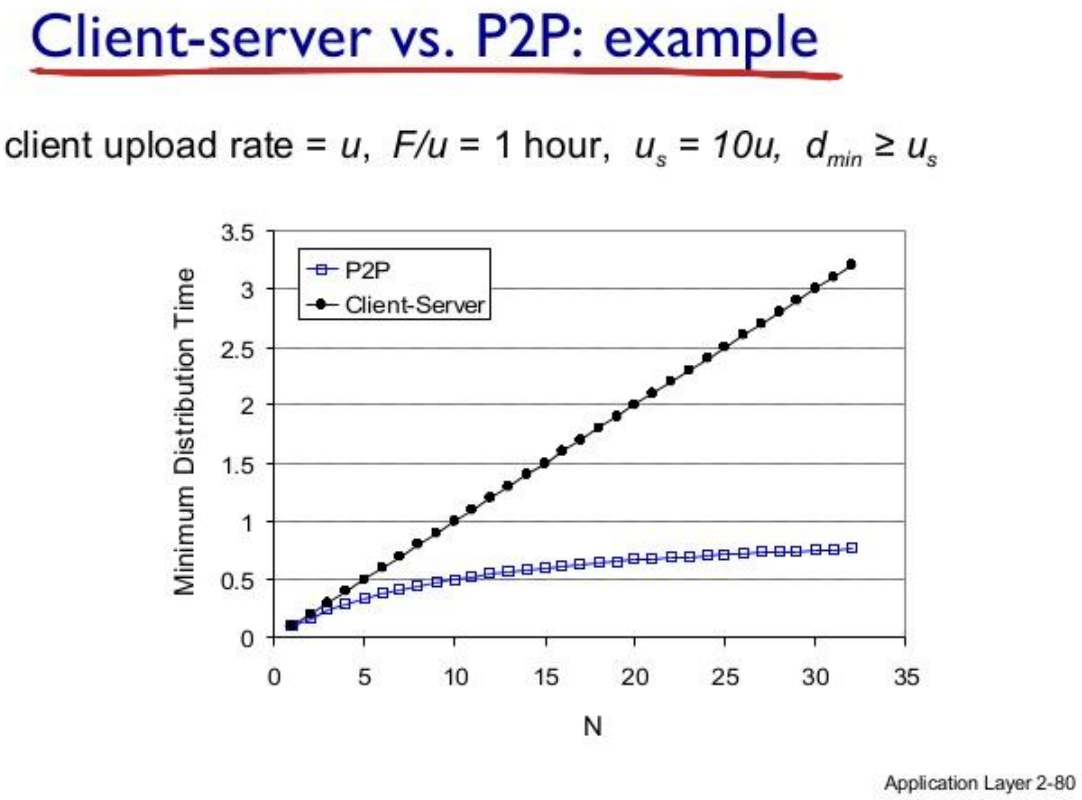
$$D_{P2P} \geq \max\{^{F}/_{u_s}, ^{F}/_{d_{min}}, ^{nF}/_{u_{ukupno}}\} \tag{4.2}$$

Uzmimo najmanje moguće vrijeme od nejednadžbe (4.2).

$$D_{P2P} = \max\{^{F}/_{u_s}, ^{F}/_{d_{min}}, ^{nF}/_{u_{ukupno}}\} \tag{4.3}$$

Ako uzmemo da svi klijenti imaju jednaku brzinu uploada, označimo s u te server ima deset puta brži upload, $u_s = 10u$, te neka je brzina downloada najsporijeg klijenta veća ili jednaka od

brzine servera $d_{min} \geq u_s$. Također neka je vrijeme za upload datoteke jednog klijenta jedan sat, $F/u = 1\ sat$. Nacrtajmo minimalna vremena raspodjele za klijent – poslužitelj (4.1) i peer to peer (4.2) sustave s danim parametrima u ovisnosti o broju klijenata u sustavu.(Slika 4.1)



Slika 4.1 Usporedba

Jasno se vidi da minimalno vrijeme raspodjele za klijent – poslužitelj linearno raste s povećanjem klijenata, dok kod peer to peer sustava vrijeme raspodjele ne samo da je manje od klijent – poslužiteljevog nego je i uvijek manje od jednog sata (horizontalna asimptota je 1). Detalje ove analize možete pogledati u [1, str 145-148].

5. Sigurnost i zloupotreba P2P sustava

5.1 Sigurnost P2P sustava

Peer to peer sustavi uz svoje velike prednosti u brzini, predstavljaju poseban problem kod sigurnosti samih sustava. Kako svi klijenti sudjeluju u komunikaciji i međusobno komuniciraju, ovakav sustav je posebno osjetljiv na napadače koji vrlo dobro poznaju strukturu sustava. Primjerice, ukoliko se radi o decentraliziranom peer to peer sustavu na koji se priključio zlonamjeran korisnik, on može prilikom emitiranja odgovarati krivo na upite koji mu dolaze ili slati krive informacije. Time se u sustavu događa pogreška i sustav, ukoliko se ne može izregulirati automatski, je prisiljen na rušenje ili ponovno pokretanje. Također sustav je osjetljiv na „Denial of service“ napade, gdje zlonamjerni korisnici mogu preplaviti upitima druge korisnike ili poslužitelja te tako onemogućiti njihovo normalno funkcioniranje. Spajanje na peer to peer mrežu je stoga uvijek rizično, ali s obzirom da je spajanje na sam internet često gotovo jednako rizično, to možemo uzeti s rezervom.

5.2 Zloupotreba P2P sustava

Zbog svojih svojstava peer to peer je savršen sustav za dijeljenje ilegalnog sadržaja. Protokoli kao BitTorrent omogućuju anonimnost ljudima koji su sposobni probiti lozinke i sigurnosne zapreke aplikacija te zatim te aplikacije besplatno dijeliti na internetu. Takvi se ljudi, uključujući i one koji skidaju ilegalne programe i aplikacije, na internetu popularno nazivaju pirati. Zbog njihovih ilegalnih radnji originalni proizvođači su često oštećeni za milijune dolara. Ozbiljnosti ove situacije dovoljno svjedoče brojne tužbe originalnih proizvođača usmjerene prema ljudima koji drže poslužitelje koji raspodjeljuju ovakve sadržaje. Međutim, gotovo je ne moguće iskorijeniti ovakve ilegalne aktivnosti na internetu, te će se zloupotreba peer to peer-a zasigurno nastaviti i u budućnosti. Vjerojatno najpoznatiji protokol koji se zloupotrebljava je BitTorrent, o čemu više u idućem poglavlju.

6. BitTorrent

Vjerojatno najuspješniji protokol koji koristi peer to peer tehnologiju je BitTorrent. Namjenjen je razmjeni datoteka te funkcionira najbolje s mnogo korisnika. Njegov način rada je poprilično složen, no pokušajmo opisati osnovne principe.

6.1 Rad BitTorrenta

Prva stvar koja upada u oko kod BitTorrenta je način slanja datoteka. Datoteke bivaju razbijene na dijelove, tzv. chunkove (eng. komad), najčešće veličine 256 KB. Potom svaki chunk biva indeksiran. Ovim postupkom je omogućeno slanje datoteke u dijelovima te optimiziran način traženja pojedinih chunkova.

Naziv za sve klijente koji sudjeluju u razmjeni pojedine datoteke je torrent. Kada se klijent spoji i zatraži određenu datoteku, on se svrstava u torrent za tu datoteku. Klijent nema nijedan chunk na samome početku te ih on dobiva tokom vremena. Kada prikupi sve chunkove datoteke, on može napustiti torrent, ili ostati i dijeliti drugima chunkove koje nemaju. Svaki torent ima metodu (tzv. tracker) koja prati klijente te tako provjerava jesu li napustili torrent ili su ostali. Torrent može imati manje od 5 klijenata, a također može imati tisuće. Kada se klijent prvi puta spoji, tracker mu šalje 50 IP adresa od ostalih u torrentu, te klijent pokušava napraviti TCP konekcije s njima. Sve klijente s kojima je napravljena uspješna konekcija nazivamo „susjedni klijenti“. Lista susjednih klijenata je dinamična i stalno se mijenja kako klijenti odlaze i novi se spajaju. Svaki od klijenata ima svoje chunkove koje je prikupio, te se redovito traži lista chunkova od svakog susjednog klijenta (preko TCP konekcije). Zatim se iz liste isčitaju chunkovi koji nedostaju klijentu i pokušavaju se skinuti. Sada možemo ustanoviti da u svakom trenutku će svaki klijent imati neki podskup skupa svih chunkova i informaciju koje chunkove posjeduju susjedni klijenti. Dolazimo do bitna dva pitanja za nastavak algoritma bez kojih bi efikasnost BitTorrenta bila vrlo loša.

1. Koje chunkove tražiti od svojih susjeda?
2. Kojim susjedima poslati chunkove koje traže?

Odgovor na prvo pitanje je tehnika „najrjeđi prvi“. Ideja je da odredi chunkove među svojim susjedima kojih ima najmanje i njih preuzima prvo. Na taj način se najrjeđi chunkovi raspodjeljuju brže od ostalih te se nastoji otprilike izjednačiti broj kopija svakog chunka u torrentu jer se preuzimanjem chunkova kojih ima najmanje uvećava njihov broj i omogućuje daljnje slanje istih.

Drugo pitanje je riješeno zanimljivim algoritmom razmjene. Osnovna ideja je slati chunkove onim klijentima od kojih su dosadašnji chunkovi primljeni po najvećoj brzini. Da bi lakše pokazali kako funkcionira sustav, nazovimo jednog klijenta Dino. Dino mjeri svojim susjedima brzinu kojom mu šalju podatke, te radi listu od četiri klijenta koja mu šalju podatke po najvećoj stopi. Dino zatim šalje chunkove samo toj četvorici klijenata s liste. Svakih deset sekundi se lista obnavlja te je onda dinamična. Bitno je dodati da svakih trideset sekundi dino bira nasumičnog susjeda i njemu šalje chunkove. Kako Dino šalje nasumičnom klijentu chunkove, Dino možda može dospjeti na listu od nasumičnog klijenta, te će Dino primati od njega podatke pa će možda i on dospjeti na Dininu listu. Ova metoda se može u slobodnom prijevodu nazvati „optimistično slanje“ ili „optimistični odabir“. Važnost ove metode leži u činjenici da se s vremenom sparuju klijenti sličnih brzina. Osim toga, klijenti koji su se tek spojili nemaju nikakve chunkove, i optimistično slanje je jedini način na koji mogu dobiti početne chunkove.

Primijetite da BitTorrent ne bi bio ni približno ovako uspješan bez algoritma razmjene koji je odgovor na drugo pitanje, često nazivan „tit-for-tat“ Jer bi većina korisnika samo pokušavala skinuti podatke od drugih bez ikakvog dijeljenja. Na ovaj način svi sudjeluju u raspodjeli podataka te uvijek postoji netko tko dijeli podatke. Više informacija o ovoj temi možete potražiti u [1, str 149-151].

6.2 Terminologija korištena u torrentima

Brzim razvojem ovog načina prijenosa podataka, razvila se i popularna terminologija vezana isključivo za torrente.

-Leeches- korisnici koji skidaju (downloadaju) datoteku ali ju ne dijele s ostalim korisnicima

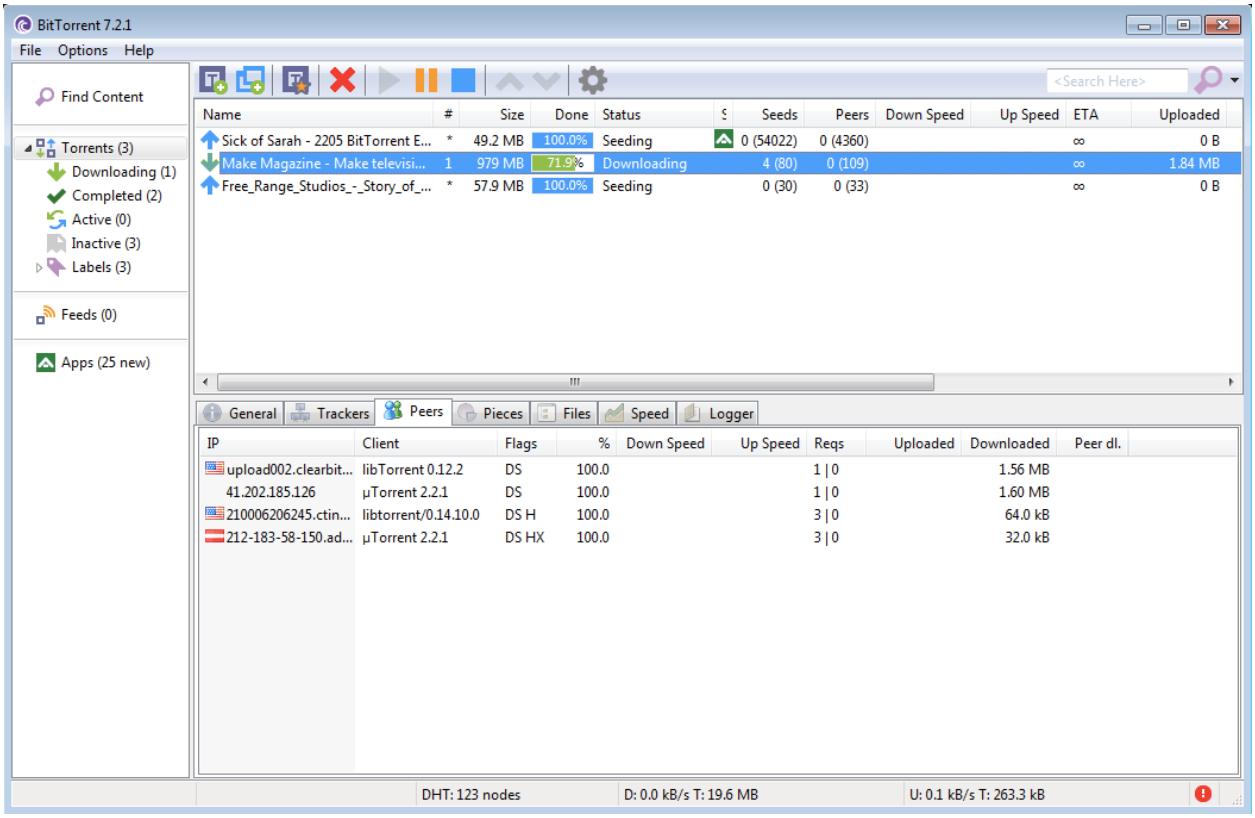
-Seed - potpuna kopija datoteke koja je dostupna drugim klijentskim računalima

-Seeder - klijent koji je završio sa skidanjem datoteke, ima cijelu kopiju datoteke i može ju dalje aktivno dijeliti

-Swarm - Grupa korisnika koja istovremeno šalje (uploada) odnosno prima (downloada) istu datoteku

-Share ratio - broj koji se dobiva dijeljenjem količine podataka koju pošalje klijent i koji skine

-Availability - broj potpunih kopija koji je na raspolaganju klijentu



Slika 6.1 Torrent

Slika 6.1 prikazuje što korisnici vide u jednoj od torrent aplikacija.

7.Primjeri P2P sustava

Osim BitTorrenta opisanog u prethodnom poglavlju, postoje i brojni drugi protokoli i aplikacije koje koriste P2P mreže. Navedimo neke od njih:

-Tradepal i M-commerce- aplikacije za online trgovinu

-Bitcoin i alternative kao Peercoin i Nxt- peer to peer bazirane digitalne valute

-I2P- mreža kreirana za surfanje internetom anonimno

-Infinit- neograničena i enkriptirana peer to peer aplikacija za razmjenu podataka digitalnih umjetnika, napisana u C++

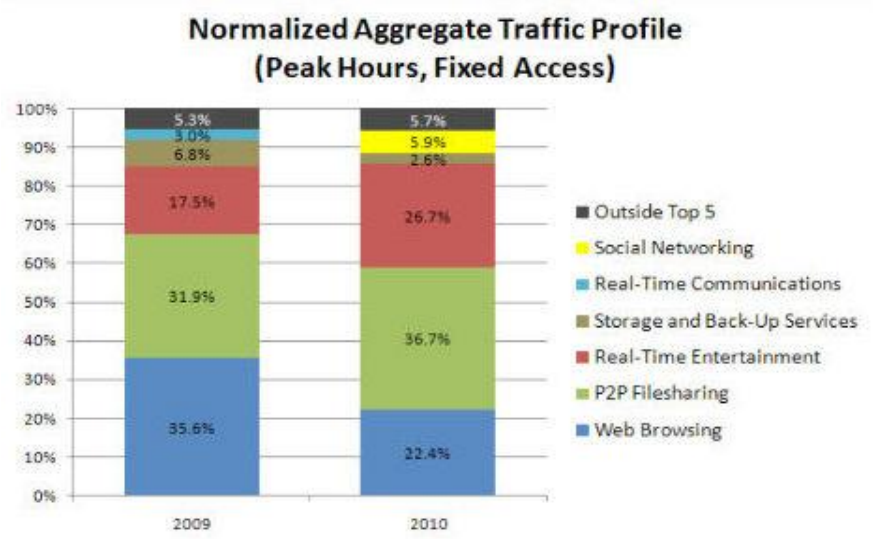
-Netsukuku- bežična mrežna zajednica, dizajnirana da bude odvojena od interneta

Brojni drugi, od kojih ćemo spomenuti samo Dalesa, Open Garden, Chord project, JXTA, Midpoint, CurrencyFair…

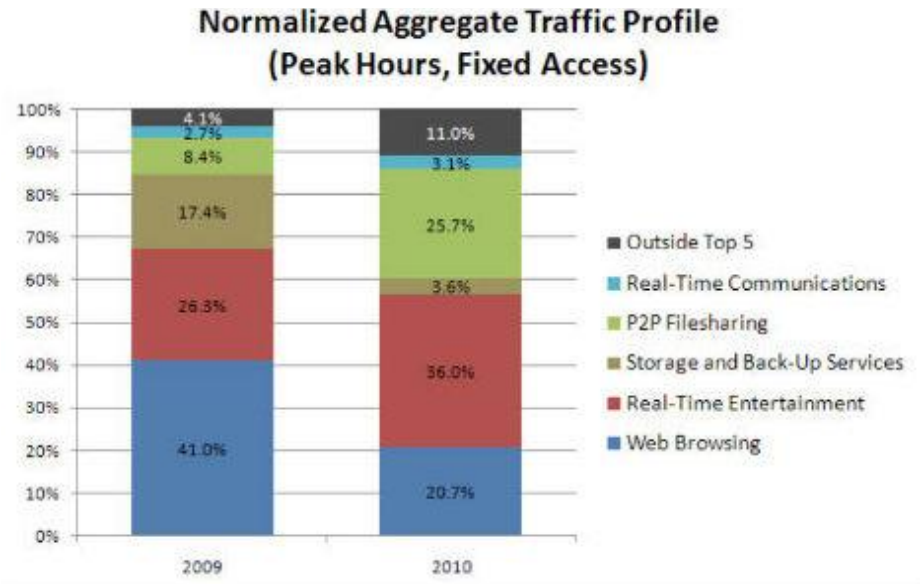
Postoji nekoliko značajnih protokola koji koriste brojni programi, neki od njih su: Ares, BitTorrent, Direct Connect, Fast Track, eDonkey, Gnutella, Bitcoin, OpenNap, RShare…

8. P2P danas

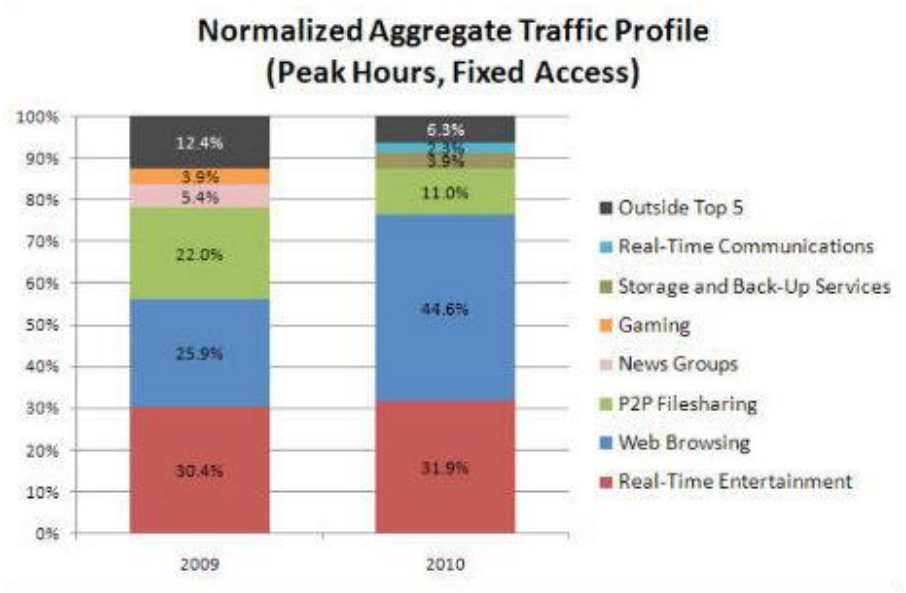
Vrlo je zanimljivo pratiti razvoj peer to peer sustava od svojih samih početaka. Sam koncept peer to peer-a je vrlo prirodan i početne verzije mreža su ga podrazumijevale. Zapravo je vrlo čudno da je klijent – poslužitelj bio toliko dominantan sve te godine i da nitko nije previše pokušavao ostvariti komunikaciju između dva klijenta. Međutim, kada je P2P prepoznat kao vrijedan alat, vrlo brzo je preuzeo internet. Podaci 2010. istraživanja kanadske kompanije Sandvine govore da je oko polovina sveukupnog upstream prometa na internetu napravljena preko peer to peer aplikacija, od čega više od pola koristeći BitTorrent protokol. Podaci za downstream promet su dosta skromniji sa oko desetinom prometa preko peer to peer aplikacija. Podaci se razlikuju po kontinentima što se može vidjeti iz Slika 8.1(Latinska Amerika), Slika 8.2(Azija i Pacifik), Slika 8.3(Europa), Slika 8.4(Amerika).



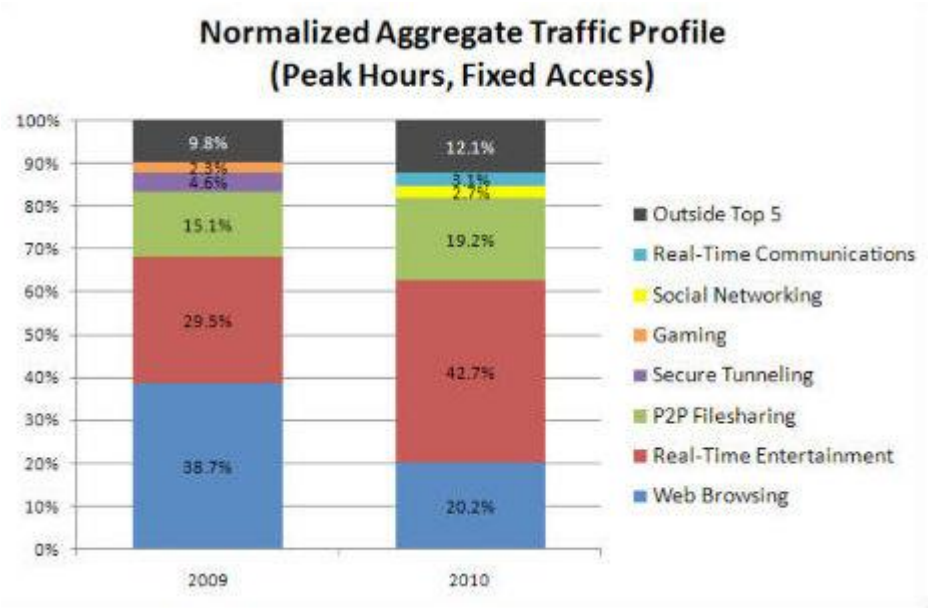
Slika 8.1 Latinska Amerika



Slika 8.2 Azija i Pacifik



Slika 8.3 Europa



Slika 8.4 Amerika

Treba imati na umu da su ovo već stari podaci i da je do 2015. situacija mnogo drugačija. Pojavom iTunes-a, Netflix-a i drugih poslužitelja koji su davali legalne besplatne usluge, promet peer to peer sustava je krenuo opadati. Činjenica je da je velik udio peer to peer prometa napravljen dijeljenjem ilegalnog sadržaja preko torrenata te pojava legalnog besplatnog sadržaja znatno utječe na taj promet. Također, veliki pritisak tužbi je bio na leđima osnivača raznih torrent aplikacija, te su se njihove aktivnosti pomno pratile. Zbog navedenih razloga dogodio se znatan pad u peer to peer prometu. No pad je zaustavljen oko 2012. godine, te je od tada P2P promet uglavnom konstantan.

9. Zaključak

Premda se P2P mreže zloupotrebljavaju i imaju razne mane, njihov je razvitak donio mnogo više dobrih stvari nego loših. Dijeljenje datoteka nikada nije bilo jednostavnije ni brže zahvaljujući protokolima kao BitTorrent. Klijent-poslužitelj je napokon dobio alternativu, te se kombinacijom ova dva sustava prijenos podataka znatno ubrzao. Pokušavajući kontrirati ilegalni trend razmjene datoteka pojavljuju se poslužitelji koji nude besplatni sadržaj kao Netflix za filmove. Također aplikacija za povoljnu kupovinu računalnih igara, Steam, doživljava svoj procvat zbog procjene ljudi da je bolje uzet povoljni originalni proizvod nego riskirati sa ilegalnim sadržajima koji su besplatni ali često puni grešaka te ne potpuno funkcionalni. Konačno, gledamo li u budućnost peer to peer-a, za njega će definitivno biti mjesta u daljnjem razvoju interneta. Teško je predvidjeti kakve će još primjene doživjeti, no postoje već organizacije koje se zalažu za novi društveni poredak zasnovan na P2P principima, kao P2P Fundation. Tko zna, možda peer to peer koncept neće promjeniti samo Internet već i samo društvo.

10. Primjer P2P u Pythonu

Evo primjera jedne P2P aplikcije napisane u programskom jeziku Python. Ideja aplikacije je ista kao kod Napstera, postoji poslužitelj koji povezuje klijente, koji zatim međusobno razmjenjuju podatke.
Primjedba 1: korištene su python biblioteke time, socket, threading, OS
Primjedba 2: Izbačeni su neki tehnički detalji radi bolje preglednosti.
Poslužitelj:

```
1.  class server:
2.      def __init__(self):
3.          self.Zavrsna=[]
4.          self.Adrese={}
5.          self.Updates={}
6.          self.path=os.path.dirname(sys.argv[0])
7.          f=open(self.path+'\\log.txt','a')
8.          f.write('\n')
9.          f.write('-----------------------------------------------------\n')
10.         f.write('Starting log on: '+time.strftime("%d/%m/%Y")+' '+time.strftime('%H:%M:%S')+'.\n')

11.         f.write('-----------------------------------------------------\n')
12.         f.close()
13.         self.lock=threading.Lock()
14.         self.running()
15.     def sendm(self,sl,data):
16.         data=bytes(data,'UTF-8')
17.         multi=256-len(data)
18.         data=data+multi*bytes(' ','UTF-8')
19.         sl.send(data)
20.     def recvm(self,sl):
21.         data=bytes('','UTF-8')
22.         while True:
23.             data=data+sl.recv(256-len(data))
24.             if len(data)==256:
25.                 break
26.         data=str(data,'UTF-8')
27.         i=len(data)-1
28.         while data[i]==' ':
29.             i=i-1
30.         data=data[:i+1]
31.         return(data)
32.     def delete(self,pomocna, add):
33.         for x in pomocna:
34.             self.Zavrsna.remove(x)
35.         del self.Adrese[add]
36.         del self.Updates[add]
37.         for i in self.Updates.keys():
38.             self.Updates[i].append(add)
39.         writing='['+time.strftime('%H:%M:%S')+'] Client '+add[0]+':'+str(add[1])+' has disconnecte
    d.'
40.         print(writing)
41.         f=open(self.path+'\\log.txt','a')
42.         f.write(writing+'\n')
43.         f.close()
44.     def backupthread(self, c, add):
45.         pomocna=[]
46.         dulj=len(self.Zavrsna)
47.         for i in range(dulj):
48.             if self.Zavrsna[i][0]==add:
49.                 for j in range(i,dulj):
50.                     if self.Zavrsna[j][0]!=add:
```

```
                        Zavrsnap=self.Zavrsna[j:]
                        self.Zavrsna=self.Zavrsna[:i]
                        self.Zavrsna.extend(Zavrsnap)
                        break
                if len(self.Zavrsna)==dulj:
                    self.Zavrsna=self.Zavrsna[:i]
                break
        for i in range(0,int(self.recvm(c))):
            ime=self.recvm(c)
            pravavel=self.recvm(c)
            pravavel=pravavel.split(' ')
            pomocna.append([add,ime,pravavel[0],pravavel[1],pravavel[2],pravavel[3]])
        self.Zavrsna.extend(pomocna)
        return(pomocna)
    def handleclient(self, c,add):
        self.lock.acquire()
        writing='['+time.strftime('%H:%M:%S')+'] Client '+add[0]+':'+str(add[1])+' has connected.'

        print(writing)
        f=open(self.path+'\\log.txt','a')
        f.write(writing+'\n')
        f.close()
        try:
            self.Adrese[add]=int(self.recvm(c))
        except ConnectionResetError:
            writing='['+time.strftime('%H:%M:%S')+'] Client '+add[0]+':'+str(add[1])+' has disconn
ected.'
            print(writing)
            f=open(self.path+'\\log.txt','a')
            f.write(writing+'\n')
            f.close()
            self.lock.release()
            return()
        if len(self.Zavrsna)==0:
            self.Updates[add]=[]
        else:
            self.Updates[add]=[]
            for i in self.Updates.keys():
                if i!=add:
                    self.Updates[i].append(add)
                    self.Updates[add].append(i)
        pomocna=[]
        self.lock.release()
        while True:
            try:
                info=self.recvm(c)
            except ConnectionResetError:
                info=''
            self.lock.acquire()
            if info=='1':
                try:
                    pomocna=self.backupthread(c, add)
                except ConnectionResetError:
                    self.delete(pomocna,add)
                    self.lock.release()
                    break
                for i in self.Updates.keys():
                    if i!=add:
                        self.Updates[i].append(add)
                writing='['+time.strftime('%H:%M:%S')+'] Client '+add[0]+':'+str(add[1])+' has cha
nged his upload data.'
                print(writing)
                f=open(self.path+'\\log.txt','a')
                f.write(writing+'\n')
                f.close()
            elif info=='2':
```

```
                pass
            elif info=='down':
                try:
                    self.down(c)
                except ConnectionResetError:
                    self.delete(pomocna,add)
                    self.lock.release()
                    break
            else:
                self.delete(pomocna,add)
                self.lock.release()
                break
            if info!='down':
                if len(self.Updates[add])>0:
                    self.sendm(c,'1')
                    dulj=len(self.Zavrsna)
                    Sendlist=[]
                    for i in range(dulj):
                        if self.Zavrsna[i][0]in self.Updates[add]:
                            Sendlist.append(self.Zavrsna[i])
                    self.sendm(c,str(len(self.Updates[add])))
                    for i in self.Updates[add]:
                        self.sendm(c,i[0]+':'+str(i[1]))
                    dulj=len(Sendlist)
                    self.sendm(c,str(dulj))
                    for i in Sendlist:
                        self.sendm(c,i[1])
                        self.sendm(c,i[0][0]+' '+str(i[0][1])+' '+i[2]+' '+i[3]+' '+i[4]+' '+i[5])

                    self.Updates[add]=[]
                else:
                    self.sendm(c,'2')
            self.lock.release()
    def down(self, c):
        adress=self.recvm(c)
        adress=adress.split(':')
        if (adress[0],int(adress[1])) in self.Adrese:
            self.sendm(c,str(self.Adrese[(adress[0],int(adress[1]))]))
        else:
            self.sendm(c,'error')
    def running(self):
        self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.s.bind(('',3000))
        self.s.listen(1)
        while True:
            (c,add)=self.s.accept()
            t = threading.Thread(target = self.handleclient, args=[c,add])
            t.start()
```

Klijent:

```
class client:
    def __init__(self):
        self.online=True
        self.optioname=''
        self.serveraddress=''
        self.downloadfolder=''
        self.uploadfolder=''
        self.s=''
        self.start()
        self.clearing()
        self.A=[]
        self.spremlist=[]
        self.status={}
        self.Adata=[]
```

```
        self.iterator=-1
        self.lock = threading.Lock()
        self.inithelp()
    def clearing(self):
        if sys.platform == "linux" or sys.platform == "linux2":
            os.system('clear')
        elif sys.platform == "win32":
            os.system('cls')
    def start(self):
'''Technical part where IP address of server and other options are configured '''
    def sendm(self,sl,data):
        data=bytes(data,'UTF-8')
        multi=256-len(data)
        data=data+multi*bytes(' ','UTF-8')
        sl.send(data)
    def recvm(self,sl):
        data=bytes('','UTF-8')
        while True:
            data=data+sl.recv(256-len(data))
            if len(data)==256:
                break
        data=str(data,'UTF-8')
        i=len(data)-1
        while data[i]==' ':
            i=i-1
        data=data[:i+1]
        return(data)
    def upfind(self,B, Bdata,poz, downlist):
        downlist.append(poz)
        for i in range(poz+1, len(B)):
            if Bdata[i][1]==poz:
                if Bdata[i][0]=='D':
                    self.upfind(B,Bdata,i, downlist)
                else:
                    downlist.append(i)
        return(downlist)
    def uploadprocess(self,B,Bdata,poz, sokd):
        if Bdata[poz][0]=='F':
            g=open(Bdata[poz][3],'rb')
            rsize=os.path.getsize(Bdata[poz][3])
            self.sendm(sokd, str(rsize))
            if rsize<1048576:
                sokd.send(g.read(rsize))
            else:
                tsize=bytes('','UTF-8')
                while True:
                    rd=g.read(1048576)
                    sokd.send(rd)
                    tsize=tsize+rd
                    self.sendm(sokd,str(len(tsize)))
                    if rsize-len(tsize)<1048576:
                        sokd.send(g.read(rsize-len(tsize)))
                        break
            g.close()
    def handleclient(self,d):
        try:
            vel=int(self.recvm(d))
            self.lock.acquire()
            B=[]
            Bdata=[]
            for i in self.A:
                B.append(i)
            for i in self.Adata:
                Bdata.append(i)
            self.lock.release()
            proc=False
```

```
            if vel== len(B):
                proc=True
                self.sendm(d,'OK')
                for i in range(vel):
                    tmp=self.recvm(d).split(' ')
                    if tmp[-1]==str(Bdata[i][1]) and tmp[-2]==Bdata[i][0] and tmp[:-
2]==B[i].split(' '):
                        self.sendm(d,'OK')
                    else:
                        self.sendm(d,('no'))
                        proc=False
                        break
            if proc==True:
                poz=int(self.recvm(d))
                self.sendm(d, str(Bdata[poz][2]))
                downlist=self.upfind(B,Bdata,poz, [])
                for i in downlist:
                    self.uploadprocess(B,Bdata,i,d)
        except ConnectionResetError:
            pass

    def background(self,sok):
        sok.listen(1)
        while True:
            (d,pomadd)=sok.accept()
            b = threading.Thread(target = self.handleclient, args=[d])
            b.start()
    def openfold(self,veznalist,poz, stop):
        if veznalist[poz][4]==stop:
            return(self.downloadfolder+'\\'+veznalist[poz][0])
        else:
            return(self.openfold(veznalist,int(veznalist[poz][4]), stop)+'\\'+veznalist[poz][0])
    def downloadprocess(self,veznalist, poz, stop, sokd):
        if veznalist[poz][3]=='F':
            g=open(self.openfold(veznalist, poz, veznalist[stop][4]),'wb')
            rsize=int(self.recvm(sokd))
            if rsize<1048576:
                bdata=bytes('','UTF-8')
                while len(bdata)<rsize:
                    bdata=bdata+sokd.recv(rsize-len(bdata))
                g.write(bdata)
            else:
                statpom=self.status[veznalist[stop][0]][0]
                while True:
                    bdata=bytes('','UTF-8')
                    while len(bdata)<1048576:
                        bdata=bdata+sokd.recv(1048576-len(bdata))
                    g.write(bdata)
                    tsize=int(self.recvm(sokd))
                    self.status[veznalist[stop][0]][0]=statpom+tsize
                    if (rsize-tsize) <1048576:
                        bdata=bytes('','UTF-8')
                        while len(bdata)<rsize-tsize:
                            bdata=bdata+sokd.recv(rsize-tsize-len(bdata))
                        g.write(bdata)
                        break
            g.close()
        else:
            path=self.openfold(veznalist, poz, veznalist[stop][4])
            if os.path.exists(path)==False:
                os.makedirs(path)
    def downfind(self,veznalist,poz,downlist):
        downlist.append(poz)
        for i in range(poz+1, len(veznalist)):
            if veznalist[i][4]==str(poz):
                if veznalist[i][3]=='D':
```

```
                    self.downfind(veznalist,i, downlist)
                else:
                    downlist.append(i)
        return(downlist)

    def downpomoc(self,sokd,adress):
        self.lock.acquire()
        self.status[adress[0]]=[0,1]
        veznalist=[]
        for i in self.spremlist:
            if i[1]==adress[1] and i[2]==adress[2]:
                veznalist.append(i)
        self.lock.release()
        proc=True
        try:
            self.sendm(sokd,str(len(veznalist)))
            if self.recvm(sokd)=='OK':
                for i in veznalist:
                    self.sendm(sokd,i[0]+' '+i[3]+' '+i[4])
                    if self.recvm(sokd)!='OK':
                        proc=False
                        break
            else:
                proc=False
        except ConnectionResetError:
            proc=False
        poz=veznalist.index(adress)
        if proc:
            try:
                self.sendm(sokd,str(poz))
                self.status[adress[0]][1]=int(self.recvm(sokd))
                downlist=self.downfind(veznalist,poz, [])
                for i in downlist:
                    self.downloadprocess(veznalist, i, poz, sokd)
                self.status[adress[0]]='Download finished'
                sokd.shutdown(socket.SHUT_RDWR)
                sokd.close()
            except ConnectionResetError:
                self.status[adress[0]]='An error occured, try downloading again.'
        else:
            sokd.shutdown(socket.SHUT_RDWR)
            sokd.close()
            self.status[adress[0]]='An error occured, try downloading again.'
    def down(self,n):
        r=n[5:]
        try:
            if int(r)%1!=0:
                raise SyntaxError
        except:
            return('Your entry is wrong.')
        r=int(r)
        if r<0:
            return('You cant enter negative numbers.')
        self.lock.acquire()
        if r>len(self.spremlist):
            return('Your number is not in the list.')
            self.lock.release()
        try:
            adress=self.spremlist[r]
            self.sendm(self.s,'down')
            self.sendm(self.s,adress[1]+':'+adress[2])
            sokd = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
            asd=int(self.recvm(self.s))
        except ValueError:
            self.lock.release()
            return('User '+adress[1]+':'+adress[2]+' has disconnected.')
```

```
212.         except ConnectionResetError:
213.             self.clearing()
214.             self.online=False
215.             self.lock.release()
216.             return('Connection to server lost.')
217.         try:
218.             sokd.connect((adress[1],asd))
219.             e = threading.Thread(target = self.downpomoc, args=[sokd,adress])
220.             e.start()
221.         except:
222.             self.lock.release()
223.             return('There has been a problem with the connection, try again later.')
224.         self.lock.release()
225.         return('Your download has started.')
226.     def statusi(self):
227.         if len(self.status.keys())==0:
228.             print('No downloads currently in progress.')
229.             print('\n')
230.         else:
231.             todel=[]
232.             for i in self.status.keys():
233.                 try:
234.                     print(i+'      '+str(100*self.status[i][0]/self.status[i][1])[:5]+'%')
235.                 except:
236.                     print(i+'      '+self.status[i])
237.                     todel.append(i)
238.             print('\n')
239.             if todel!=[]:
240.                 for i in todel:
241.                     del self.status[i]
242.     def help(self):
243. '''Prints helpful information'''
244.         self.routine(-1)
245.     def cmds(self):
246.         print('The following commands are avalivable')
247.         print('!help, !down number,!status,!open number,')
248.         print('!back, !refresh, !options, !cmds, !exit')
249.     def printing(self,numb,name,fi,ad,port,size,by):
250.         print(numb+' '*(5-len(numb))+' '+name[:10]+' '*(10-len(name))+'   '+fi+' '*(9-
    len(fi))+'   '+ad+':'+port+' '*(15-len(ad))+' '+size+' '+by)
251.     def showlist(self,num):
252.         '''Prints the list of downloads from spremlist.'''
253.     def Folderhelp(self,path, num):
254.         self.iterator=self.iterator+1
255.         folsize=0
256.         for file in os.listdir(path):
257.             if os.path.isdir(path+'\\'+file)== True:
258.                 self.A.append(file)
259.                 self.Adata.append(['D',num, 0,path+'\\'+file])
260.                 helpit=self.iterator
261.                 fol=self.Folderhelp(path+'\\'+file, self.iterator)
262.                 folsize=folsize+fol
263.                 self.Adata[helpit][2]=fol
264.             else:
265.                 filesize=os.path.getsize(path+'\\'+file)
266.                 self.Adata.append(['F',num, filesize, path+'\\'+file])
267.                 self.A.append(file)
268.                 folsize=folsize+filesize
269.                 self.iterator=self.iterator+1
270.         return(folsize)
271.     def backupthread(self):
272.         while True:
273.             B=self.A
274.             self.lock.acquire()
275.             self.A=[]
276.             if self.A!=[]:
```

```
277.                 print(self.A[-1])
278.             self.iterator=-1
279.             try:
280.                 self.Folderhelp(self.uploadfolder,-1)
281.                 if B==self.A:
282.                     self.sendm(self.s,'2')
283.                 else:
284.                     self.sendm(self.s,'1')
285.                     self.sending()
286.                 msg=int(self.recvm(self.s))
287.                 if msg==1:
288.                     msg=int(self.recvm(self.s))
289.                     todel=[]
290.                     for i in range(msg):
291.                         dat=self.recvm(self.s)
292.                         dat=dat.split(':')
293.                         todel.append(dat)
294.                     todel2=[]
295.                     for i in self.spremlist:
296.                         if [i[1],i[2]]in todel:
297.                             todel2.append(i)
298.                     for i in todel2:
299.                         self.spremlist.remove(i)
300.                     msg=int(self.recvm(self.s))
301.                     for i in range(msg):
302.                         name=self.recvm(self.s)
303.                         datas=self.recvm(self.s)
304.                         datas=datas.split(' ')
305.                         self.spremlist.append([name,datas[0],datas[1],datas[2],datas[3],datas[4],d
    atas[5]])
306.                 else:
307.                     pass
308.             except ConnectionResetError:
309.                 self.clearing()
310.                 print('Lost connection to server.')
311.                 self.online=False
312.                 self.lock.release()
313.                 break
314.             self.lock.release()
315.             time.sleep(5)
316.     def sending(self):
317.         length=len(self.A)
318.         self.sendm(self.s,str(length))
319.         for i in range(length):
320.             self.sendm(self.s,self.A[i])
321.             b=self.Adata[i][2]
322.             it=0
323.             while b>1024:
324.                 it=it+1
325.                 b=b//1024
326.             if it==0:
327.                 self.sendm(self.s,self.Adata[i][0]+' '+str(self.Adata[i][1])+' '+str(self.Adata[i]
    [2])[:5]+' B')
328.             elif it==1:
329.                 self.sendm(self.s,self.Adata[i][0]+' '+str(self.Adata[i][1])+' '+str(self.Adata[i]
    [2]/1024)[:5]+' KB')
330.             elif it==2:
331.                 self.sendm(self.s,self.Adata[i][0]+' '+str(self.Adata[i][1])+' '+str(self.Adata[i]
    [2]/(1024*1024))[:5]+' MB')
332.             elif it==3:
333.                 self.sendm(self.s,self.Adata[i][0]+' '+str(self.Adata[i][1])+' '+str(self.Adata[i]
    [2]/(1024**3))[:5]+' GB')
334.             elif it==4:
335.                 self.sendm(self.s,self.Adata[i][0]+' '+str(self.Adata[i][1])+' '+str(self.Adata[i]
    [2]/(1024**4))[:5]+' TB')
336.     def routine(self,num):
```

```
337.         self.lock.acquire()
338.         if self.online==False:
339.             print('Start the program again to reconnect.')
340.             input('Press enter to exit.')
341.             return()
342.         if self.spremlist==[]:
343.             print('\n \n')
344.             print('There is nothing to download at the moment.')
345.             print('\n \n')
346.         else:
347.             try:
348.                 self.showlist(num)
349.             except:
350.                 self.lock.release()
351.                 return(self.routine(-1))
352.         spremlist2=[]
353.         for i in self.spremlist:
354.             spremlist2.append(i)
355.         self.lock.release()
356.         n=input('Expecting further commands:')
357.         if n=='!cmds':
358.             self.clearing()
359.             self.cmds()
360.             input('Press enter to continue.')
361.             self.clearing()
362.             return(self.routine(num))
363.         elif n=='!exit':
364.             return()
365.         elif n=='!help':
366.             self.clearing()
367.             self.help()
368.         elif n=='!status':
369.             self.clearing()
370.             self.statusi()
371.             input('Press enter to continue.')
372.             self.clearing()
373.             return(self.routine(num))
374.         elif n[:5]=='!down':
375.             pom=0
376.             for i in range(len(n)):
377.                 uvrst=i-pom
378.                 if n[uvrst]==' ' or n[uvrst]=='_':
379.                     n=n[:uvrst-1]+n[uvrst:]
380.                     pom=pom+1
381.             self.clearing()
382.             print(self.down(n))
383.             input('Press enter to continue.')
384.             self.clearing()
385.             return(self.routine(-1))
386.         elif n[:5]=='!open':
387.             pom=0
388.             for i in range(len(n)):
389.                 uvrst=i-pom
390.                 if n[uvrst]==' ' or n[uvrst]=='_':
391.                     n=n[:uvrst-1]+n[uvrst:]
392.                     pom=pom+1
393.             try:
394.                 n=int(n[5:])
395.             except TypeError:
396.                 self.clearing()
397.                 print('Your input must be integer.')
398.                 return(self.routine(num))
399.             try:
400.                 if spremlist2[n][3]=='D':
401.                     self.clearing()
402.                     return(self.routine(n))
```

```
                else:
                    self.clearing()
                    print('You can open only directories.')
                    input('Press enter to continue.')
                    self.clearing()
                    return(self.routine(num))
            except IndexError:
                self.clearing()
                print('Your number must be larger than 0 and in list.')
                input('Press enter to continue.')
                self.clearing()
                return(self.routine(num))
        elif n=='!back' or n=='!b':
            self.clearing()
            if num==-1 or int(spremlist2[num][4])==-1:
                return(self.routine(-1))
            else:
                for i in range(len(spremlist2)):
                    if spremlist2[i][1]==spremlist2[num][1] and spremlist2[i][2]==spremlist2[num][
    2]:
                        tpos=i
                        break
                return(self.routine(tpos+int(spremlist2[num][4])))
        elif n=='!refresh' or n=='!r' or n=='!ref':
            self.clearing()
            return(self.routine(-1))
        elif n=='!options':
'''Prints options'''
            return(self.routine(num))
        else:
            self.clearing()
            print('The command you entered is wrong, avalivable commands are: !cmds,')
            print('!down number,!status,!open number,')
            print('!back, !refresh, !cmds, !help, !options, !exit')
            input('Press enter to continue.')
            self.clearing()
            return(self.routine(num))
    def inithelp(self):
        sp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        i=0
        while True:
            try:
                sp.bind(('',3001+i))
                break
            except:
                i=i+1
        t = threading.Thread(target = self.background, args=[sp])
        t.setDaemon(True)
        t.start()
        self.lock.acquire()
        self.sendm(self.s,str(sp.getsockname()[1]))
        self.lock.release()
        thr=threading.Thread(target = self.backupthread)
        thr.setDaemon(True)
        thr.start()
        self.help()
```

Literatura:

[1] James F. Kurose, Keith W. Ross, Computer networking: A top down approach sixth edition, Pearson, SAD, 2012.

[2] torrentkb.weebly.com

[3] en.wikipedia.org/wiki/Peer-to-peer

[4] torrentfreak.com/bittorrent-still-dominates-global-internet-traffic-101026/

[5] www.deepfield.com/2012/04/the-rise-and-fall-and-rise-of-p2p/

[6] p2pfoundation.net/Main_Page