SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD br. 626.

RAZMJENA DATOTEKA U RASPODIJELJENOM SUSTAVU

Neven Ćubić

Zagreb, lipanj 2009.

Sadržaj

1. Uvod

Današnji Internet je nezamisliv bez razmjene datoteka. Samo prilikom pisanja ovog dokumenta autor je morao sa poslužitelja na Internetu skinuti izvršnu datoteku za instalaciju programskog paketa OpenOffice. Osim izvršnih programa razmjenjuju se audio i video sadržaji, tekstualni dokumenti, prezentacije i razne druge datoteke koje su preplavile Internet. Porastom broja korisnika Interneta raste i broj raspoloživih datoteka, no i potražnja za datotekama. Ako se koristi popularni sustav klijent-poslužitelj za distribuciju datoteka, potrebna je sve veća propusnost prema poslužitelju, veći broj poslužitelja i jači poslužitelj za obradu svih zahtjeva za datotekama. Raspodijeljena arhitektura je jedan od alternativnih pristupa razmjeni datoteka koji se sve češće koristi u rješavanju tog problema i opisan je u ovom radu.

U drugom poglavlju je opisana raspodijeljena arhitektura te su navedene njezine prednosti i nedostaci u odnosu na druge arhitekture razmjene datoteka. U trećem poglavlju su opisani protokoli koji se temelje na raspodijeljenoj arhitekturi BitTorrent i Direct connect i mreža Gnutella. U četvrtom poglavlju je opisana implementacija jednostavnog sustava koji vrši raspodijeljenu razmjenu datoteka. Sustav se sastoji od klijenata koji vrše razmjenu datoteka i od poslužitelja koji ima posredničku ulogu, odnosno služi za pretraživanje dostupnih datoteka i pronalaženje odgovarajućih klijenata koji ih distribuiraju. Dakle ne radi se o potpuno raspodijeljenom sustavu jer se koristi centralni poslužitelj kao posrednik za pronalaženje klijenata s odgovarajućim datotekama, ali se ipak svi sadržaji nalaze raspodijeljeno na raspodijeljenim čvorovima mreže.

2. Raspodijeljena arhitektura

Raspodijeljena arhitektura (engl. Peer to peer, skraćeno P2P) je vrsta mrežne arhitekture u kojoj svaka radna stanica ima jednake mogućnosti i odgovornosti [1]. Kod razmjene datoteka to zapravo znači da svaki od klijenata koji sudjeluju u razmjeni datoteka ujedno služi i kao poslužitelj za druge klijente koji sudjeluju u razmjeni datoteka. Za razliku od arhitekture klijent-poslužitelj gdje poslužitelji sadrže sve datoteke, a klijenti ih samo skidaju, u ovoj arhitekturi klijenti sadrže datoteke i dijele ih sa drugim klijentima. U ovom poglavlju razmatrane su takve arhitekture i njihove prednosti i nedostaci.

2.1 Vrste raspodijeljenih mreža

U ovome radu se opisuju raspodijeljene mreže za razmjenu datoteka. Razmjena datoteka nije jedina upotreba za takve mreže, no kako se često koriste za razmjenu datoteka često se poistovjećuju samo sa razmjenom datoteka. Općenito se može napraviti podjela na tri osnovna zadatka za koja se danas upotrebljavaju raspodijeljene mreže [2]:

– raspodijeljeni računalni sustavi,

– komunikacija u stvarnom vremenu,

– razmjena datoteka.

Kod raspodijeljenih računalnih sustava pokušava se iskoristiti više računalnih resursa za obavljanje istog ili sličnog posla. Najveća primjena ovakvih sustava je u istraživanjima poput projekta *Folding@home* gdje se iskorištavaju neiskorišteni procesorski ciklusi kućnih računala i konzola za medicinska istraživanja. Druga popularna primjena su sustavi za trenutnu komunikaciju porukama koji dopuštaju korisnicima da direktno komuniciraju bez posredovanja poslužitelja kao što je to slučaj kod elektroničke pošte. Ovo obično nisu potpuno raspodijeljene mreže već u određenom trenutku koriste centralni poslužitelj za autorizaciju ili posredovanje pri uspostavljanju veze. Treća upotreba, razmjena datoteka, je predmet razmatranja

ovog rada i iz tog razloga druge primjene se više neće izravno razmatrati.

Već je spomenuto da većina današnjih raspodijeljenih mreža nisu "potpuno" raspodijeljene mreže. Potpuno raspodijeljena mreža sastoji se samo od klijenata koji su jednaki i imaju ulogu i poslužitelja i klijenta. Ne koriste se posrednički poslužitelji već su klijenti sami odgovorni za nalaženje drugih klijenata i za obavještavanje mreže klijenata o dijeljenim datotekama. Primjer ovakve mreže je Gnutella, mreža koja je 2007. bila najpopularnija mreža za razmjenu podataka [3]. Ovaj način razmatranja raspodijeljenih mreža dovodi do druge podijele raspodijeljenih mreža prema stupnju centralizacije. Potpuno raspodijeljena mreža je primjer potpuno decentralizirane mreže, dok većina mreža danas ima bar posrednički poslužitelj koji ažurira listu aktivnih klijenata. Takav poslužitelj obično vodi i listu dijeljenih datoteka sa strane pojedinih klijenata i daje podatke o klijentima da bi se mogla uspostaviti veza između dva ili više klijenata. Klijenti pri spajanju na takav poslužitelj najčešće pošalju listu datoteka koje dijele i zatraže datoteku koju korisnik želi skinuti. Poslužitelj nađe koji klijenti imaju tu datoteku i pošalje taj popis klijentu, koji odabere jednog klijenta, napravi direktnu vezu sa njim i skine traženu datoteku.

Treća podjela raspodijeljenih sustava je prema načinu na koji se klijenti međusobno nalaze, odnosno spajaju u zajedničku mrežu. To je podjela na strukturirane i nestrukturirane mreže. Analizira se način na koji su klijenti povezani i način na koji novi klijent koji pristupa mreži ostvaruje vlastite veze i pretraživanja. Kod strukturirane mreže postoji definirani i dobro poznati protokol koji opisuje način na koji se veze ostvaruju. Cilj je osigurati da zahtjev za datotekom uvijek dođe do klijenta koji sadrži tu datoteku. Jedan od popularnih pristupa izgradnji strukturiranih mreža je upotreba distribuiranih asocijativnih polja (engl. Distributed hash table, skraćeno DHT) [4]. U toj mreži svaki klijent dobije ključ za koji je odgovoran na temelju određenog seta ključeva koji se generira u mreži. Iz nekog podatka (npr. dijeljena datoteka) stvara se sažetak (engl. hash, obično SHA-1) iz kojeg se dobiva ključ tog podatka. Tad se pošalje poruka kroz sustav čvorova do čvora koji je odgovoran za taj ključ i on pohrani podatke asocirane uz taj ključ. Kad neki korisnik zatraži datoteku koja odgovara tom ključu, ostali čvorovi će pomoću svojih tablica usmjeravanja i navedenog ključa poslati taj zahtjev prema odgovarajućem čvoru koji će poslati podatke asocirane uz taj ključ klijentu koji je

započeo pretragu.

Kod nestrukturiranih mreža ne postoji zajednički protokol koji definira način ostvarivanja veza i pretraživanja mreže. Klijent koji se spoji sam odabire način na koji će ostvariti veze sa drugim klijentima i ne postoji poveznica između klijenata i datoteka koje dijele. Prednost ovakve mreže je u jednostavnom načinu spajanja na tu mrežu, no nedostatak je što se svaki upit mora slati prema što više klijenata. S obzirom da nema veze između klijenata i datoteka koje dijele, nema garancije da će upit stići do odgovarajućeg klijenta, što posebno dolazi do izražaja prilikom pretraživanja mreže za datotekama koje su rijetke.

2.2 Prednosti i nedostaci raspodijeljenih mreža

Kod razmjene datoteka glavna prednost raspodijeljene mreže je kapacitet mreže (diskovni prostor, propusnost,..). Kod klasične arhitekture klijent-poslužitelj kapacitet mreže ovisi o broju poslužitelja, propusnosti prema tim poslužiteljima i karakteristikama samih poslužitelja (npr. memorijski prostor). Broj poslužitelja je fiksan i povećanjem zahtjeva prema tim poslužiteljima brzine razmjene datoteka padaju. Kod raspodijeljenih sustava svaki klijent je ujedno i poslužitelj pa dolazak novog klijenta povećava kapacitet mreže. To ne znači da će raspodijeljene mreže uvijek dati bolje performanse od mreže poslužitelj-klijent, no mreža će se bolje nositi sa porastom broja korisnika i prikladnija je za razmjenu velike količine multimedijskog sadržaja (npr. besplatna muzika sa servisa Jamendo [5] koji dijeli pjesme uz pomoć BitTorrent protokola). No mali broj korisnika može značiti i dosta loše performanse jer propusnost kao i procesorska moć klijenata obično nije poznata i ne može se kontrolirati u razmjeni datoteka preko Interneta. Raspodijeljene mreže su prikladnije i za razmjenu većih datoteka jer je paralelizam pri skidanju dijelova datoteka od više korisnika lakše izvesti nego kod ograničenog broja poslužitelja. Još jedna dobra značajka raspodijeljenih mreža je otpornost na kvarove. Kvar kod jednog klijenta obično ne utječe puno na cijelu mrežu, a kod potpuno raspodijeljenih sustava klijenti se ne moraju oslanjati na rad posredničkih poslužitelja za pretraživanje.

Najveći nedostatak raspodijeljenih mreža je sigurnost. Datoteke se dijele između klijenata i klijenti ih dijele dalje drugim klijentima. Kako ne postoji samo jedna kopija datoteke na poslužitelju otvaraju se razne mogućnosti manipulacije datotekama. Datoteka koja se čini ispravna prilikom pretrage može biti virus ili neispravna datoteka. Primjer ove opasnosti je napad na mreže koje su koristile protokol FastTrack [6] gdje su u mrežu ubačeni neispravni dijelovi datoteka. Prilikom skidanja datoteke korisnik je skinuo i neispravne dijelove i datoteka je postala neupotrebljiva. Za napad je najvećim dijelom bila zaslužna tvrtka Overpeer [7] koja je iskoristila slabost u algoritmu za stvaranje sažetka (engl. Hash algorithm) UUHash koji su koristile mreže temeljene na protokolu FastTrack. Algoritmi za stvaranje sažetka su jedan od načina na koji se mreže brane od takvih napada. Oni na temelju cijele datoteke ili njezinih dijelova računaju zaštitnu sumu (engl. Checksum) pomoću koje osiguravaju integritet datoteke. Kada korisnik skine datoteku ponovno se računa zaštitna suma i uspoređuje se sa poznatom sumom za tu datoteku da bi se provjerio njezin integritet. Ova tehnika se može primijeniti i na pojedine dijelove datoteke. Problem neispravnih datoteka je na današnjim popularnim P2P mrežama dosta malen zahvaljujući boljim algoritmima i funkcijama za kontrolu integriteta datoteka.

2.3 Zakonski problemi razmjene datoteka

Razvoj raspodijeljenih mreža je omogućio mnogo lakšu i kvalitetniju razmjenu raznog sadržaja preko Interneta. Mp3 format je igrao značajnu ulogu u rastu popularnosti mreža za razmjenu datoteka. Pjesme su postale manje, lakše ih je bilo dijeliti i popularnost njihove razmjene je brzo rasla sa obzirom na ograničenost distribucije pjesama pomoću fizičkih medija. No sa rastom popularnosti razmjene pjesama i sličnih sadržaja rastao je i problem narušavanja autorskih prava. Narušavanje autorskih prava je navuklo na popularne P2P mreže bijes raznih tvrtki koje se bave distribucijom takvih sadržaja, ali i autora. Rizik od tužbi je doveo do rasta klijenata koji pokušavaju osigurati što veću anonimnost korisnika pri korištenju P2P mreže. Poznata tužba zbog kršenja autorskih prava na P2P mreži je sudski postupak protiv osnivača internetske stranice The Pirate Bay [8]. Oni su

držali poslužitelje koji su radili kao BitTorrent trackeri, odnosno pomagali su u komunikaciji između klijenata koji koriste BitTorrent protokol [9]. Tuženi su zbog promicanja kršenja autorskih prava i osuđeni su na godinu dana zatvora i kaznu od 2.7 milijuna eura nakon čega su podnijeli žalbu protiv odluke [10].

3. Protokoli i mreže temeljeni na raspodijeljenoj arhitekturi

Treće poglavlje pokriva neke od popularnih protokola i mreža koje se danas koriste za raspodijeljenu razmjenu datoteka. Prva mreža koja je djelomično podržavala raspodijeljenu arhitekturu i koja se koristila za razmjenu datoteka jest USENET [11]. P2P komunikacija se odvijala između poslužitelja, dok korisnici su sa poslužitelja skidali datoteke po modelu klijent-poslužitelj. Iako je primarna svrha mreže bila razmjena poruka, mreža je dopuštala korisnicima i razmjenu datoteka. Zbog načina rada USENET-a ta mreža se ipak ne smatra prvom P2P mrežom za razmjenu datoteka, već je tu titulu dobila mreža Napster 1999. [12]. Napster također nije bio potpuno raspodijeljena mreža jer je imao centralni poslužitelj koji je pratio dijeljene datoteke. Napster je u početku zamišljen kao mreža za razmjenu pjesama, njegovi autori su željeli bolji način razmjene muzike od pretraživanja drugih mreža poput USENET-a. 2000. izlaze Gnutella, eDonkey2000 i Freenet, 2001. Kazaa i BitTorrent protokol. U 2009.-oj godini najpopularnije P2P mreže za dijeljenje datoteka su Gnutella, eDonkey i mreže temeljene na BitTorrent protokolu [13]. Protokoli koji su navedeni u daljnjem tekstu su BitTorrent i Advanced Direct Connect koji predstavljaju dva različita pristupa raspodijeljenoj razmjeni datoteka, oba sa visokim stupnjem centralizacije. Navedena je i mreža Gnutella koja predstavlja potpuno decentralizirani pristup raspodijeljenoj razmjeni datoteka.

3.1 BitTorrent protokol

BitTorrent je među najpopularnijim protokolima za razmjenu datoteka. 2004. istraživanja su pokazala da je zaslužan za 35% prometa na Internetu. Njegova glavna namjena je razmjena velikih datoteka, no moguće ga je koristiti za sve vrste datoteka. Za protokol je zaslužan Bram Cohen, a prva verzija protokola se pojavila u travnju 2001.. Protokol dijeli korisnike koji sudjeluju u razmjeni na one koji samo distribuiraju cjelokupnu datoteku drugima (izvor, engl. seed) i one koji tu datoteku skidaju jer ju nemaju u potpunosti te skinute dijelove dijele dalje (čvor, engl. peer). Procedura dijeljenja počinje tako da izvori omogućuju drugima skidanje datoteke.

Datoteke se distribuiraju u dijelovima i korisnici koji počnu skidati te dijelove postaju čvorovi. Oni svaki dio koji skinu odmah učine dostupnim drugim korisnicima za skidanje, a kad dovrše skidanje cijele datoteke postaju izvori. Za dobar rad ovog protokola vrlo je važno imati dobar sustav određivanja sa kojim klijentima komunicirati, odnosno kojim klijentima slati podatke, a kojima ne. Problem su klijenti koji daju vrlo mali dio svoje mrežne propusnosti za dijeljenje datoteka, a ostatak za skidanje. Ovom problemu se može pristupiti na više načina, jedan je preferiranje klijenata koji i sami šalju dovoljan broj podataka, dok još jedan način jest čuvati određeni dio mogućeg broja konekcija za slučajan odabir klijenata i testiranje njihove kvalitete.

Klijent započinje skidanje datoteke nalaženjem odgovarajuće *.torrent* datoteke. Torrent datoteka sadrži podatke o dijeljenim datotekama i adresu posrednika (engl. tracker). Već je objašnjeno da je posrednik poslužitelj koji pomaže u komunikaciji između klijenata. Za vrijeme skidanja datoteka posrednika je dovoljno kontaktirati u početku, no moguće je i kasnije održavati vezu sa njim zbog dijeljenja statistika o skidanju i dobivanja podataka o novim klijentima u mreži. Posrednik može sadržavati i BitTorrent indeks, odnosno listu .torrent datoteka, no to može biti i na odvojenom poslužitelju. Nakon što se klijent spoji na posrednik naveden u .torrent datoteci, posrednik mu šalje listu drugih klijenata od kojih će skidati dijelove datoteke koju traži. Datoteke su obično podijeljene na dijelove veličine od 64 kB do 4 MB. Svaki dio ima zaštitnu sumu napravljenu pomoću SHA-1 algoritma za računanje sažetka. Zaštitne sume, kao i veličina datoteke i broj dijelova su zapisani u .torrent datoteci. Skidanje datoteke pomoću BitTorrent protokola je specifično iz razloga što se stvara niz konekcija preko različitih TCP priključnica, a skidanje dijelova ide po metodi "*rjeđi prije*" ili po slučajnom odabiru. Najveći problemi BitTorrent protokola su vrijeme potrebno za skidanje dovoljno dijelova od strane pojedinog klijenta da se poveća njegov doprinos mreži i spor rast brzine skidanja. Nekad je potrebno određeno vrijeme da klijent napravi dovoljno konekcija prema dovoljno dobrim klijentima i počne skidati datoteku većom brzinom.

3.2 ADC

Protokol naprednog izravnog povezivanja (engl. Advanced Direct Connect, skraćeno ADC) je P2P protokol koji se temelji na protokolu izravnog povezivanja (engl. Direct Connect, skraćeno DC) [15]. Nastao je kao nadogradnja postojećeg DC protokola sa ciljem unaprjeđenja i ispravljanja starog protokola. Prva verzija protokola se pojavila u 2004. dok je prva službena verzija izašla tek 2007. ADC nije čisti P2P protokol, već kao i većina drugih koristi centralni poslužitelj za neke operacije. Taj poslužitelj se naziva mrežni čvor (engl. hub). Klijenti se spajaju na čvor i tako dobivaju pristup datotekama drugih klijenata. Klijenti mogu pretraživati datoteke samo od korisnika koji se nalaze na čvorovima na koje su i oni trenutno spojeni. Čvorovi služe i kao poslužitelji za razmjenu tekstualnih poruka između korisnika.

Komunikacija preko ADC protokola provodi se sa porukama u običnom tekstu kodiranom prema UTF-8 kodu (kratica za *8-bit Unicode Transformation Format*). Prilikom ostvarivanja konekcije protokol očekuje da klijent prvi pošalje poruku, odnosno ako se jedan klijent spaja na čvor ili drugog klijenta, nakon što se ostvari priključnica klijent koji je zatražio vezu mora prvi poslati poruku. Klijent koji se spaja mora znati vrata transportnog protokola na kojima drugi klijent, odnosno čvor sluša za nove veze jer ne postoje globalno poznata vrata već je izbor proizvoljan za svaki drugi čvor. Komunikacija između klijenata provodi se pomoću TCP protokola za skidanje datoteka, dok se pretrage mogu provoditi pomoću i TCP i UDP protokola. Prilikom prvog spajanja na čvor klijent javi čvoru koje sažetke podržava i na temelju odabranog algoritma za stvaranje sažetka stvaraju se dva krajnja koda, međusobno ovisni jedan o drugome. Klijent će oba koda poslati čvoru koji će, ako ih potvrdi, jedan od njih proslijediti drugim klijentima. Ti kodovi sadrže identifikacijske podatke i ograničeni su na trenutni čvor. Klijent može stvoriti nove kodove za drugi čvor na koji se spoji. Trajanje kodova je ograničeno na trajanje sjednice sa čvorom. Za skidanje datoteka klijenti koriste slotove, odnosno određeni broj dopuštenih konekcija koje drugi klijenti mogu ostvariti da bi skidali podatke od pojedinog klijenta. Broj slotova je često diktiran politikom čvora na koji se klijent spaja i obično se određuje s obzirom na broj drugih čvorova na koje je klijent spojen. U cilju veće kvalitete mreže koju čini pojedini čvor, čvorovi često

osim takvih pravila imaju još niz drugih pravila koje postavljaju novim klijentima. Može se zahtijevati određeni broj dijeljenih datoteka, određeni tip dijeljenih datoteka, minimalan kapacitet veze klijenta i razni drugi zahtjevi. Kad se spoje na čvor klijenti mogu raditi u aktivnom ili pasivnom načinu. Pasivni način je ograničen na skidanje datoteka samo od aktivnih korisnika i ne može dobivati rezultate pretrage direktno od drugih klijenata. Pasivni korisnik prilikom spajanja na čvor ne objavljuje svim korisnicima svoje podatke, već samo pita čvor koji klijent ima datoteku koju želi i kontaktira tog klijenta. Glavna prednost pasivnog načina je manji broj konekcija na koje klijent mora odgovarati, što uključuje zahtjeve za skidanjem i pretrage. Kako korisnici koji su aktivni odmah pošalju svim klijentima obavijest da postoje, klijenti se mogu na njih direktno spajati i od njih pretraživati datoteke. Dodjela slotova korisnicima se ostvaruje pomoću značke (engl. token). Nakon što stigne zahtjev za vezom, pomoću značke se određuje koji klijenti prvi dobivaju slotove.

Prosječno čvorovi imaju nekoliko tisuća korisnika koji međusobno dijele datoteke i razmjenjuju tekstualne poruke. Protokol ADC su razvili autori popularnog klijenta DC++, a jedno od unaprjeđenja u odnosu na originalni DC protokol je slanje svim korisnicima TTH (kratica za Tiger-Tree Hashing) koda svih datoteka koje određeni korisnik dijeli. TTH omogućuje verifikaciju integriteta skinute datoteke i nalaženje datoteke neovisno o njezinom imenu.

3.3 Gnutella

U prvom i drugom dijelu ovog poglavlja obrađena su dva protokola za raspodijeljene mreže. No oni nisu namijenjeni za stvaranje velike mreže koju koriste svi klijenti. Kod BitTorrenta jednu mrežu sačinjavaju klijenti spojeni na isti posrednik koji dijele datoteke navedene u jednoj .torrent datoteci. Ako je u .torrent datoteci navedeno više posrednika onda mrežu sačinjavaju i oni sa svojim klijentima koji dijele iste podatke. .torrent datoteka ne mora biti ista za sve klijente, no podaci koje dijele u toj mreži moraju. Kod ADC i DC protokola mrežu čine klijenti spojeni na isti čvor. Klijenti mogu biti spojeni na više čvorova, no nemaju pristup datotekama koje sadrže klijenti koji nisu spojeni na te čvorove, kao što kod

BitTorrenta klijenti ne mogu pristupiti datotekama koje nisu navedene u .torrent datotekama koje imaju, iako drugi klijenti možda dijele te datoteke. Kod Gnutelle moguće je postići da su svi klijenti međusobno dostupni međusobno jer je Gnutella potpuna raspodijeljena mreža koja ne ovisi o centralnim poslužiteljima. Prvi klijent za Gnutellu su razvili Justin Frankel i Tom Pepper koji su tada radili za Nullsoft. Iako je ubrzo nakon izdavanja klijent povučen sa strane AOL-a koji je kupio Nullsoft, pojavile su se grupe koje su uspjele kopirati protokol i počeli su se razvijati novi klijenti.

Porastom popularnosti način rada Gnutelle je prilagođen za veći broj korisnika. Originalno klijenti su prilikom početka rada koristili neku metodu za pronalazak bar jednog drugog klijenta, npr. imali su već poznatu adresu klijenta. Taj jedan pronađeni klijent bi drugom klijentu koji ga je kontaktirao vratio listu klijenata za koje on zna i drugi klijent bi pokušao kontaktirati te klijente. Postupak bi se ponavljao dok se ne bi ostvarila određena kvota broja klijenata sa kojim se može povezati. Klijenti se u Gnutelli nazivaju čvorovima (engl. node). Za skidanje datoteke klijent je prvo slao upit prema svim klijentima na koje je spojen. Oni su taj upit proslijedili prema klijentima na koje su oni spojeni i postupak se ponavljao dok upit nije došao do klijenta koji je imao traženu datoteku ili se nije ostvario predodređeni broj skokova od klijenta. Klijent sa rezultatima pretrage je kontaktirao klijenta koji je započeo pretragu šaljući mu poruku putem kojim je originalni zahtjev za pretragu došao. Kasnije je nadogradnja protokola omogućila direktno vraćanje poruke klijentu koji je započeo pretragu pomoću UDP protokola. Verzija 0.6 je bitno promijenila način na koji mreža funkcionira. Klijenti od 0.6 više nisu bili jednaki u mreži, već su se podijelili na listove (engl. leaf node) i ultračvorove (engl. ultra node). Listovi su klijenti koji nemaju odgovornost usmjeravanja poruka i pretraga kroz mrežu. Pri spajanju na mrežu svi klijenti se tretiraju kao listovi i vode se kao krajnji klijenti. Klijenti koji imaju mogućnost usmjeravanja se unaprijeđuju u ultračvorove, klijente koji su zaduženi za usmjeravanje mrežnih poruka i pretraga kroz mrežu. Time je riješen problem starog načina u kojem su klijenti koji su bili slabijih mogućnosti usporavali rad mreže prilikom prosljeđivanja i usmjeravanja poruka. Listovi su obično spojeni na malen broj ultračvorova, dok ultračvor je spojen na najmanje 32 druga ultračvora. Još jedan problem mreže je način pretraživanja. Pretraga se obavlja po ključnim riječima koje se šalju klijentima koji

se stalno spajaju i odspajaju sa mreže. Zato nije osigurano da će upit uvijek doći odgovarajućeg klijenta. Verzija 0.6 je sa podijelom klijenata pokušala riješiti i taj problem. Za povećanje performansi su uvedeni i protokol usmjeravanja upita (engl. Query Routing Protocol, skraćeno QRP) i protokol dinamičkog slanja upita (engl. Dynamic Querying, skraćeno DQ). DQ zaustavlja pretragu čim klijent dohvati dovoljno potrebnih podataka, a QRP omogućuje da upit za datotekom dođe samo do odgovarajućih korisnika. To je moguće razmjenjivanjem tablice usmjeravanja upita (engl. Query Routing Table, skraćeno QRT) koja sadrži sažetke ključnih riječi. Listovi stvaraju vlastiti QRT i predaju ga ultračvoru na koji su spojeni. On spaja sve QRT-ove listova koji su spojeni na njega i vlastiti QRT ako dijeli podatke, te novonastali QRT šalje svim ultračvorovima na koje je spojen. Kad neki ultračvor primi upit prvo će napraviti raspršni kod iz riječi sadržanih u upitu i provjeriti da li odgovaraju ikojem unosu u QRT-u, te tek ako odgovaraju proslijediti upit listu. Za razmjenu datoteke klijenti ostvaruju izravnu vezu, no moguće je iskoristiti i posrednike koji se tada nazivaju *push proxies*. To je najčešće ultračvor na kojem je spojen list koji sadrži datoteku, a obično se koriste ako je pristup tom listu ograničen i nije moguće direktno uspostaviti vezu.

4. Simulacija raspodijeljene razmjene datoteka

U četvrtom poglavlju je opisan jednostavan model raspodijeljene razmjene datoteka koji je u usporedbi sa protokolima opisanim u trećem poglavlju najsličniji Direct Connect protokolu. Model se sastoji od dva osnovna dijela, klijenata koji međusobno razmjenjuju datoteke i centralnog poslužitelja koji prati datoteke koje klijenti dijele. Zbog uporabe poslužitelja navedena mreža nije potpuna P2P mreža no klijenti odgovaraju temeljnoj definiciji P2P mreže: imaju jednake mogućnosti i odgovornosti. Svaki klijent distribuira datoteke drugim klijentima i ujedno ih i skida od drugih klijenata po želji korisnika. Sva komunikacija je tekstualna, odnosno provodi se pomoću poruka kodiranih u ANSI zapisu. Klijenti pretražuju dostupne datoteke šaljući poslužitelju upit sa imenom datoteke, a poslužitelj prima od svih klijenata listu datoteka koje dijele. Razmjena datoteka se odvija u dijelovima veličine 2 MiB. Klijenti pokušavaju paralelno skidati dijelove od više klijenata, a klijentima šalju samo zatraženi dio datoteke pri posluživanju. Ovo poglavlje je podijeljeno na tri dijela, prvi dio opisuje rad centralnog poslužitelja, drugi dio klijente koji dijele datoteke, a treći dio daje upute za korištenje ostvarene implementacije i dodatan opis interakcija komponenti. Programska implementacije je napisana u jeziku C# u programskom okruženju Visual Studio 2008 sa instaliranim .NET paketom 3.5. Testiranja su provedena na računalu sa instaliranim operacijskim sustavom Windows XP Professional i instaliranim paketom zakrpa Service Pack 3.

4.1 Centralni poslužitelj

Kao što je već navedeno, poslužitelj primarno prati koje datoteke klijenti dijele i pamti IP adrese i transportna vrata tih klijenata. Prilikom pokretanja poslužitelj stvara novu dretvu koja započinje slušati zahtjeve za konekcijom na zadanim vratima i IP adresi. To pokretanje dretve je nužno jer konekcije u programu se ostvaruju na sinkroni način i dretva koja sluša je blokirana dok ne primi konekciju.

Zato je nužno slušanje obavljati u novoj dretvi i omogućiti iscrtavanje grafičkog sučelja i ispisa tijeka izvođenja. Glavna klasa koja predstavlja rad poslužitelja je klasa *Server*. Njezine metode su opisane kroz opis rada poslužitelja.

4.1.1. Opis elemenata poslužitelja

Glavna klasa poslužitelja i metode važne za rad poslužitelja:

```
class Server{
        public Server( TextBox textBox );
        public Server( TextBox textBox, int port, IPAddress ip );
        public void startServer();
        private void buildFileList( String inReceivedString );
        private String retrieveClients( string inReqFile );
        public FileList ServerFileList;
}
```

Metoda koja se poziva pri početku rada je:

public void startServer() - metoda koja započinje slušati zahtjeve za konekcijama i poziva odgovarajuće metode kad primi zahtjev. Sama klasa se stvara pomoću konstruktora

public Server(TextBox textBox, int port, IPAddress ip).

Konstruktor prima vrata i IP adresu na kojima će poslužitelj slušati dolazeće upite i objekt *textBox* pomoću kojeg se u grafičko sučelje ispisuje tijek rada poslužitelja. Kad poslužitelj primi zahtjev za konekcijom on taj zahtjev obradi u istoj dretvi jer nije potrebno paralelizirati kratke zadatke koje poslužitelj obavlja u sklopu ove implementacije. To ne bi bio slučaj sa složenim poslužiteljima koji sadržavaju velike baze podataka o datotekama i klijentima u sustavu. Prilikom ostvarivanja

konekcije poslužitelj očekuje tri moguća zahtjeva od klijenta: slanje liste datoteka koje klijent dijeli, zahtjev za nalaženjem klijenata koji sadrže određenu datoteku ili zahtjev za brisanjem iz liste aktivnih klijenata. U prvom slučaju klijent mora poslužitelju poslati tekstualnu poruku koja započinje sa "*LIST:"*. Kada primi takvu poruku klijent poziva funkciju sa prototipom:

private void buildFileList(String inReceivedString)

Ta funkcija iz primljene poruke vadi listu imena datoteka koje klijent ima, njegova transportna vrata i IP adresu na kojoj sluša zahtjeve za konekcijama. Očekuje se sljedeći format poruke:

IP:*ip_adresa* PO:*transportna_vrata* FI:*lista_datoteka* PRT:*broj dijelova*

Na temelju dobivene poruke gradi se baza podataka o klijentima. Bazu podataka predstavlja klasa *FileList* u kojoj je sadržana lista objekata tipa *FileInfo* i sljedeće metode:

public void AddFileInfo(FileInfo newFileInfo) - dodaje informacije o novom klijentu u bazu podataka i

public List<FileInfo> ReturnFileInfo(string fileName) - vraća listu klijenata koji sadrže zadano ime datoteke u varijabli *fileName.*

public int RemoveFiles(IPAddress ip, int port) - briše sve zapise u listi sa navedenom ip adresom i transportnim vratima.

Tip *FileInfo* je klasa koja sadrži podatke o pojedinoj dijeljenoj datoteci. Pamti ime datoteke u varijabli instance *name*, IP adresu u varijabli *ip,* transportna vrata u varijabli *port* i broj dijelova datoteke u varijabli *numParts.* Za jedno ime datoteke može postojati više objekata tipa *FileInfo* ako više različitih klijenata sadrži tu datoteku. U tom slučaju funkcija *ReturnFileInfo* vraća sve navedene klijente. Klasa *FileInfo* sadrži tri metode:

public FileInfo(string name, IPAddress ip, int port, int numParts) - kontruktor klase koji prima ime datoteke, IP adresu i vrata klijenta koji ju sadrži ,

public override string ToString() - metoda koja vraća IP adresu i vrata klijenta u tekstualnom zapisu i

public String ReturnInfo() - vraća IP adresu, vrata klijenta i ime datoteke u

formatu za ispis koji se vrši kad korisnik odabere opciju ispisa svih zapisa koje poslužitelj trenutno sadrži.

U drugom slučaju poruka ne mora početi sa određenim znakovima već se cijela poruka tretira kao ime datoteke koju klijent traži. Poslužitelj tada poziva metodu

private String retrieveClients(string inReqFile)

koja na temelju zadanog imena datoteke iz baze podataka dobiva listu klijenata koji imaju tu datoteku i stvara poruku sljedećeg formata:

NM:*broj dijelova* CL:*ip adresa klijenta* PO:*vrata klijenta.....*

Ako u bazi podataka nema ni jedan klijent sa tom datotekom stvara se poruka *NULL.* U oba slučaja poruka se šalje natrag klijentu koji je zatražio datoteku.

Treći slučaj, odnosno zahtjev za brisanjem, se ostvaruje kad poslužitelj primi poruku koja počinje sa "QT:". Poslužitelj tada poziva metodu RemoveFiles iz klase *FileList* koja briše sve unose u listi klijenata vezane uz ip adresu i vrata koji su navedeni u dobivenoj poruci. Lista klijenata i datoteka se nalazi u varijabli klase *fileList* klase *Server.*

Navedena tri slučaja, odnosno tipa zahtjeva koje poslužitelj može primiti predstavljaju glavnu funkcionalnost poslužitelja. Poslužitelj ne može paralelno obraditi više različiti zahtjeva od istog klijenta, već se za svaki zahtjev mora stvoriti nova konekcija i poslati novi zahtjev.

4.2 Klijent

Rad klijenta se može podijeliti na tri glavna zadatka. Prvi zadatak je spajanje na centralni poslužitelj i predaja liste datoteka koje klijent distribuira, drugi zadatak je slušanje za upitima od strane drugih klijenata koji žele skinuti dio datoteke koji taj klijent dijeli i treći zadatak je skidanje datoteke koju korisnik zatraži. Osnovna klasa koja sadrži glavnu funkcionalnost klijenta je klasa *Client*.

4.2.1 Opis elemenata klijenta

Glavna klasa odgovorna za rad poslužitelja i najvažnije metode te klase:

```
class Client{
        public Client( TextBox textBox );
        public Client( TextBox textBox, int clientPort, IPAddress clientIP, int
                serverPort,  IPAddress serverIp );
        public void StartClient() ;
        private void sendFileList();
        private void startListening();
        public void RequestDownloadFile( String reqFile);
        private List<IPEndPoint> requestFile( String reqFile, out int
                numberOfParts )
        private void downloadFile( List<IPEndPoint> availableClients, String
                reqFile, int numParts )
        private void sendNewFileToServer( String name, int numParts );
}
```

Konstruktor klase koji se poziva na početku stvaranja klijenta je:

public Client(TextBox textBox, int clientPort, IPAddress clientIP, int serverPort, IPAddress serverIp)

koja prima vrata i IP adresu na kojima klijent sluša za upite i vrata i IP adresu na kojima poslužitelj sluša za upite. Pomoću objekta *textBox* vrši se ispis stanja programa u grafičko sučelje programa.

Klijent započinje rad s pozivom metode klase *Client:*

public void StartClient()

koja poziva dvije privatne metode iste klase:

private void sendFileList() - metoda koja šalje poslužitelju listu datoteka koje

klijent dijeli i

private void startListening() - metoda koja započne slušanje za upite od strane drugih klijenata i brine se o slanju datoteka i odgovora tim klijentima.

Prvo se poziva metoda *sendFileList()*. Metoda prvo stvara listu datoteka koje klijent dijeli tako da napravi instancu klase *ClientFileList*. Klasa *ClientFileList* sadrži sljedeće metode:

public ClientFileList(string shareFileName) - konstruktor klase koji prima kao argument ime datoteke u kojoj su zapisane datoteke koje klijent dijeli,

public Boolean FileExists(string fileName) - metoda koja provjerava da li klijent ima datoteku koju drugi klijent od njega zatraži,

private void updateList() - metoda koja započinje izgradnju liste datoteka,

public override string ToString() - metoda koja vraća listu datoteka u formatu za slanje poslužitelju i

public void AddFileToList(String name, int numParts) - metoda koja ažurira listu sa novom datotekom, obično se koristi kad klijent skine novu datoteku. Osim tih metoda *ClientFileList* sadrži i listu imena datoteka koje klijent dijeli te broj dijelova pojedine datoteke. Metoda *sendFileList* nakon stvaranja liste tu listu postavlja u odgovarajući format (opisan u poglavlju 4.1) i šalje poslužitelju kao tekstualnu poruku. Sad poslužitelj sadrži referencu na trenutnog klijenta i mogući su zahtjevi za skidanjem dijelova datoteka pa se poziva metoda *startListening* koja započne proces slušanja. Klijent za svaki primljeni upit stvara novi objekt tipa *ClientRequest* i predaje mu ostvarenu priključnicu i listu datoteka koje dijeli pozivajući konstruktor:

public ClientRequest(ClientFileList files, Socket clientSock, TextBox textBox).

Zadaća tog objekta je poslužiti klijenta koji je poslao upit i za svaki objekt stvara se nova dretva koja izvodi metodu objekta:

public void HandleClientRequest() - metoda koja šalje zatraženi dio datoteke klijentu.

Metoda prvo provjerava postoji li uopće zatražena datoteka na aktualnom klijentu. Ako postoji, metoda otvara tu datoteku i učitava traženi dio datoteke u međuspremnik veličine 2 MiB i šalje ga klijentu. Ako traženi dio nije dostupan biti će poslan blok podataka veličine 0 bajtova što će sa strane klijenta koji očekuje blok biti interpretirano kao pogreška.

Zadnji zadatak klijenta je skidanje datoteke koju korisnik zatraži preko grafičkog sučelja. Klijent pročita ime datoteke koje je korisnik zadao i na temelju njega izvede niz metoda kojima prvo od poslužitelja dobije listu drugih klijenata koji sadrže tu datoteku pa od navedenih klijenata skine tu datoteku. Kada korisnik zatraži datoteku poziva se metoda:

public void RequestDownloadFile(String reqFile) - metoda zapravo predstavlja sučelje prema privatnim metodama klase *Client:*

private List<IPEndPoint> requestFile(String reqFile, out int numberOfParts) i

private void downloadFile(List<IPEndPoint> availableClients, String reqFile, int numParts).

Metoda *requestFile* prima ime tražene datoteke i spaja se na poslužitelj te šalje ime datoteke koju želi skinuti. Već je opisano u poglavlju 4.1 kako poslužitelj odgovara na takav upit, on sastavlja listu klijentata koji sadrže tu datoteku i šalje poruku u odgovarajućem formatu klijentu. Klijent iz te poruke vadi podatke o klijentima koji sadrže traženu datoteku i broj dijelova te datoteke. Te podatke vraća u pozivajuću metodu *RequestDownloadFile* koja sa dobivenim podacima poziva metodu *downloadFile*. Metoda *downloadFile* će stvoriti onoliko objekata tipa *Download* koliko je dijelova datoteke koju korisnik skida. Zadatak svakog objekta je spojiti se na klijent koji mu je zadan i sa njega skinuti dio datoteke za koji je odgovoran. Stvara se niz dretvi koje izvode metodu tih objekata

public void DownloadPartCallback(Object threadNumber) - metoda koja skida traženi dio datoteke i zapisuje ga u trajnu memoriju računala.

Objekt se stvara pomoću konstruktora:

public Download(int num, String reqFile, IPEndPoint curIp, TextBox textBox, ManualResetEvent doneEvent) - konstruktor prima broj dijela datoteke koji mora skinuti, ime te datoteke i ip adresu i transportna vrata klijenta od kojeg će zatražiti tu datoteku.

Metoda *downloadFile* će pokušati svakom objektu dati drugi klijent na koji se mora spojiti, no kako je broj klijenata možda manji od broja dijelova neki objekti će dobiti istog klijenta. Isto tako ako je broj klijenata veći od broja dijelova neki klijenti neće biti dodijeljeni niti jednom objektu. Nakon stvaranja potrebnog broja objekata i dretvi metoda će čekati da sve završe sa izvođenjem, odnosno sve pokušaju skinuti zadani dio datoteke.

Metoda *DownloadPartCallback* prvo pokušava uspostaviti vezu sa drugim

klijentom koji joj je zadan i ako uspije pita ga ima li traženu datoteku. Ako taj klijent odgovori sa porukom "NO" onda se prekida rad jer datoteka nije dostupna i članska varijabla *downloadSuccess* se postavlja na *false*. Ta varijabla se koristi za provjeru je li uspjelo skidanje traženog dijela i postavlja se na *true* samo ako je primljen dio koji sadrži više od nula bajtova. Ako je odgovor od drugog klijenta "OK" tada se započinje sa skidanjem traženog dijela. Kad se završi sa skidanjem trenutna dretva zaključava pristup novostvorenoj datoteci i u nju upisuje dio koji je upravo skinula. Kad sve dretve završe sa svojim poslom nastavlja se izvođenje glavne dretve. Ona prvo provjeri jesu li svi dijelovi ispravno skinuti, ako nisu onda sekvencijalno zadaje objektima koji nisu skinuli svoj dio druge klijente od kojih mogu skinuti taj dio. Ovaj dio rada nije paraleliziran jer se očekuje maleni broj dijelova koje se treba ponovno pokušati skinuti. Nakon te provjere poziva se metoda *AddFileToList* klase *ClientFileList* koja u trenutnu listu datoteka dodaje ime upravo skinute datoteke i njezin broj dijelova. Onda se poziva metoda:

private void sendNewFileToServer(String name, int numParts) - ona kontaktira poslužitelj i javlja mu da je ta datoteka sad dostupna i kod trenutnog klijenta. Poruka se šalje u istom formatu u kojem se šalje cijela lista dijeljenih datoteka, samo je navedena jedna datoteka.

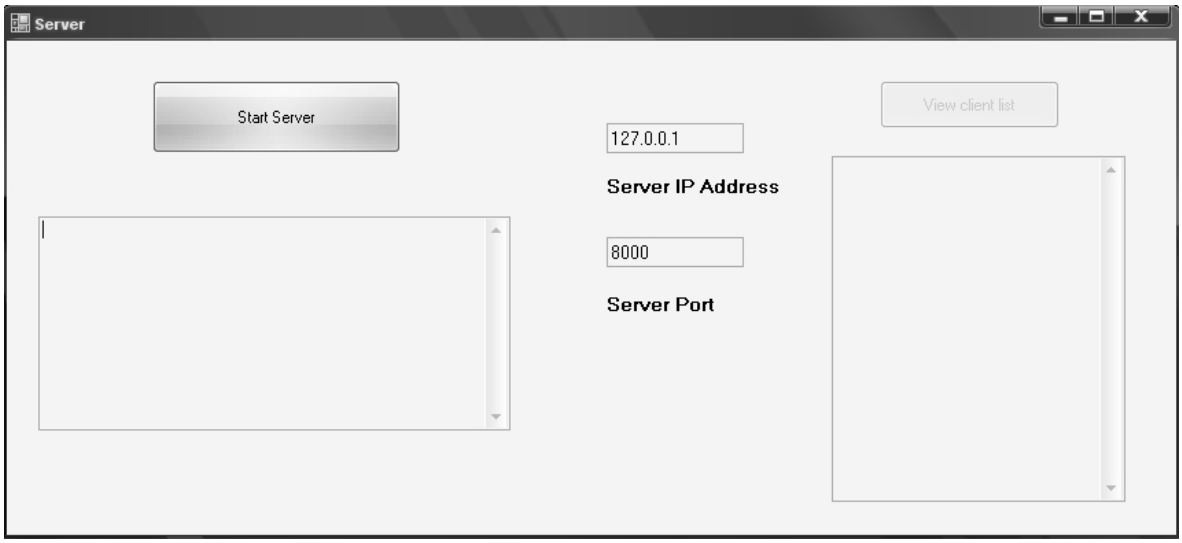
Klijent može skinuti proizvoljan broj datoteka za vrijeme rada, nema nikakvih ograničenja kao što nema ni ograničenja na broj dijelova koje klijent može poslati. Kada korisnik odluči zaustaviti rad klijenta, klijent će pri zatvaranju poslati poruku poslužitelju da izbriše sve unose vezane uz njega. Taj mehanizam je uveden da bi se spriječilo slanje nepostojećih klijenata drugim klijentima što usporava rad sustava. Poslužitelj može imati podatke o nepostojećim klijentima samo ako se izvođenje klijenta završi na nepredviđen način i ne stigne se poslati poruka za brisanje poslužitelju.

4.3 Upute za korištenje programskog paketa i primjer izvođenja

Upute za korištenje su zadane kroz primjer korištenja sustava. Taj primjer demonstrira kako podignuti poslužitelj, tri klijenta koji sadrže datoteku n.mp3 i kako sa četvrtim klijentom tu datoteku skinuti. Pritom su objašnjenje sve pripreme potrebne da bi sustav radio i opcije koje se mogu koristiti. Primjer dodatno

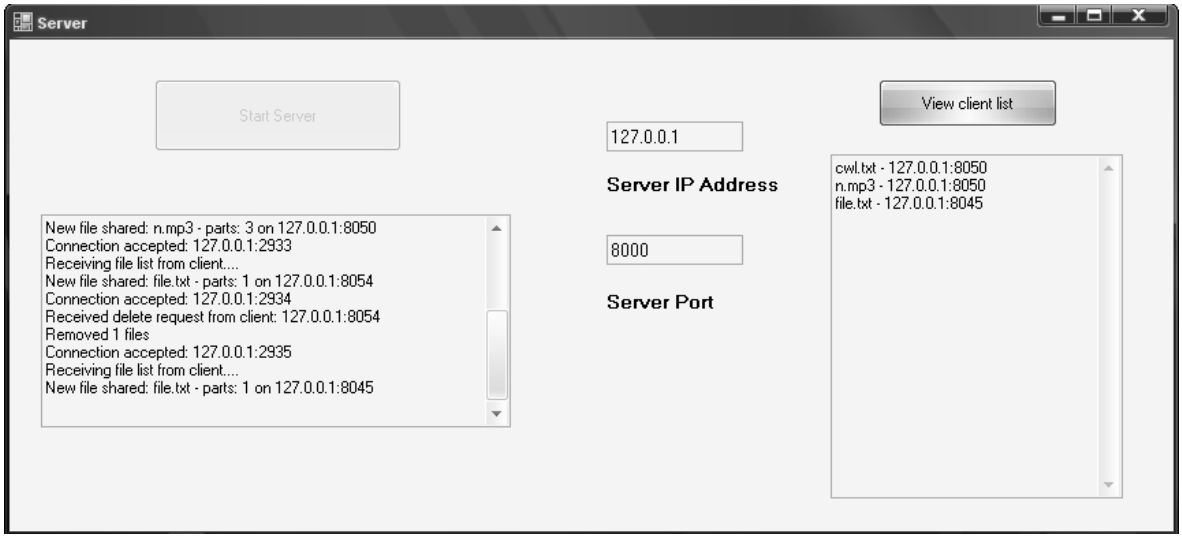
pojašnjava princip rada sustava iz korisničke perspektive.

Da bi se započelo sa radom prvo se mora podignuti poslužitelj. Slika 4.1 prikazuje izgled grafičkog sučelja poslužitelja prije početka rada (izgledi sučelja navedenih u ovom dokumentu mogu varirati ovisno o postavkama operacijskog sustava na kojem se program izvršava).



Slika 4.1 – izgled poslužitelja prije početka rada.

Poslužitelj se pokreće klikom na tipku **Start Server**. Poslužitelj će početi slušati na ip adresi navedenoj u polju iznad labele **Server IP Address**. Ako se ne želi započeti poslužitelj sa adresom 127.0.0.1 onda se adresa mora promijeniti prije nego što se pokrene rad poslužitelja. Transportna vrata se unose u polje iznad labele **Server Port** i isto se moraju promijeniti prije početka rada ili će poslužitelj početi slušati na vratima 8000. Nakon što se pokrene poslužitelj, pomoću tipke **View Client List** se može u prozoru sa tekstom ispod te tipke vidjeti ispis svih datoteka i klijenata koji ih sadrže i registrirali su se u poslužitelju. Slika 4.2 prikazuje izgled poslužitelja za vrijeme aktivnog rada. Treba primijetiti da se u prozor ispod tipke Start Server ispisuje tijek izvođenja poslužitelja.



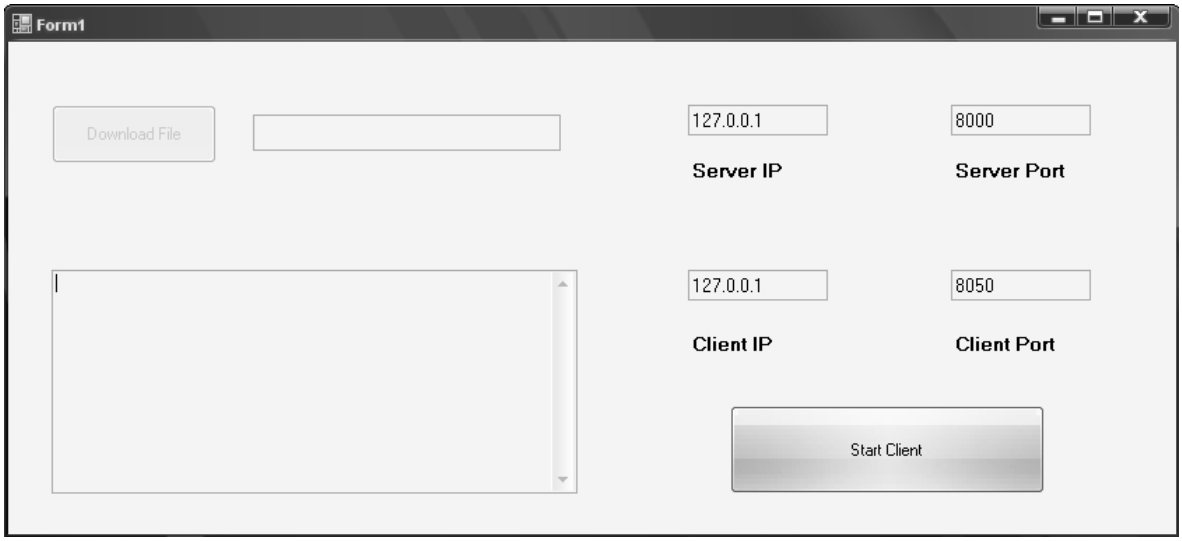
Slika 4.2 – izgled poslužitelja za vrijeme rada sa ispisanom listom datoteka koju trenutno sadrži u desnom prozoru sa tekstom i ispisanim tijekom izvođenja u lijevom prozoru sa tekstom.

Sad je potrebno podignuti klijente koji će simulirati udaljene korisnike od kojih želimo skinuti datoteku. Prije podizanja klijenta potrebno je sve datoteke koje dijeli staviti u isti direktorij u kojem se nalazi klijent te je potrebno stvoriti datoteku sa imenom **share.txt** u kojoj se zapišu imena svih datoteka koje se želi da klijent dijeli i broj njihovih dijelova. Zapis mora biti u sljedećem formatu:

ime_datoteke broj_dijelova

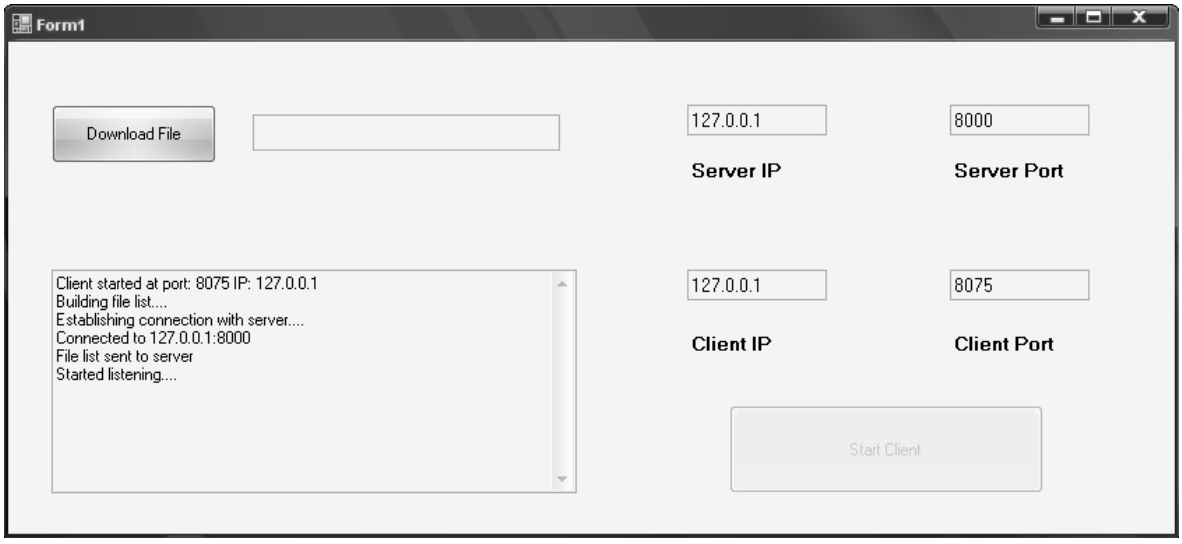
ime_druge_datoteke broj_dijelova_druge_datoteke

Broj dijelova je broj blokova od 2 MiB koji čine datoteku. Obavezan je razmak između imena i broja, te svaka datoteka mora biti u svojem redu. Slika 4.3 prikazuje izgled klijenta prije pokretanja.



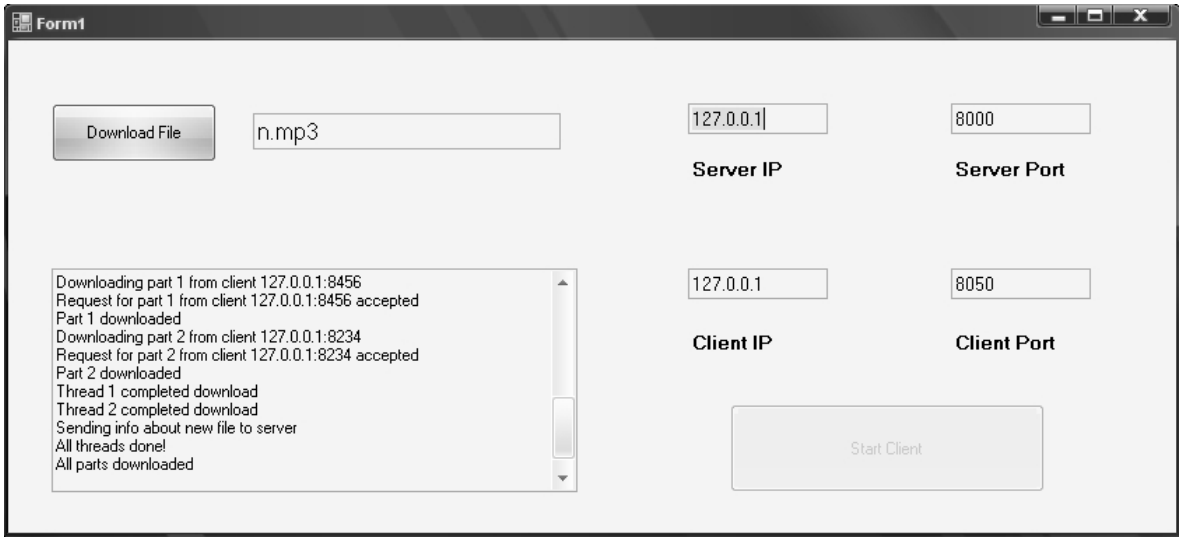
Slika 4.3 – izgled klijenta prije pokretanja.

Tipka **Start Client** započne rad klijenta sa parametrima koji su u tom trenutku uneseni u polja iznad labela **Server IP, Server Port, Client IP** i **Client Port**. Značenje parametara je istovjetno nazivu labele ispod polja u koje je unesen određeni parametar, odnosno u prvom redu unosi se ip adresa poslužitelja pa vrata poslužitelja, a u drugom redu ip adresa klijenta pa vrata klijenta. Ako se vrijednosti ne promjene program će tražiti poslužitelj na adresi 127.0.0.1 i vratima 8000, a početi će slušati na adresi 127.0.0.1 i vratima 8050. U ovom primjeru klijenti rade na istom računalu pa će svi biti pokrenuti sa istim adresama no moraju se promijeniti vrata na kojima slušaju. Slika 4.4 prikazuje izgled pokrenutog klijenta koji je spreman za posluživanje i skidanje datoteka.



Slika 4.4 – izgled klijenta nakon pokretanja, u ispisu se vidi da se spojio na poslužitelj i počeo je slušati za dolazećim konekcijama.

Nakon pokretanja postala je dostupna tipka **Download File**. Ako se unese ime datoteke u polje desno od te tipke i klikne na nju, klijent će početi proces skidanja datoteke. Tipka će postati nedostupna dok se ne završi skidanje trenutne datoteke ili dok ne stigne dojava da datoteka nije pronađena. U svrhu ovog primjera podignuta sa tri klijenta, sva tri dijele istu datoteku n.mp3 i nalaze se na istom računalu. U polje za pretragu unese se ime n.mp3 i klikom na tipku Download File klijent će se spojiti na poslužitelj i dohvatiti listu klijenata koji imaju tu datoteku. U ovom slučaju raditi će se o tri klijenta, a kako datoteka ima tri dijela stvoriti će se tri dretve, svaka će pokušati skinuti jedan dio od jednog od klijenata. Nakon završetka skidanja datoteka n.mp3 će se pojaviti u direktoriju u kojem je smješten četvrti klijent koji je započeo zahtjev za tom datotekom. Slika 4.5 prikazuje izgled klijenta nakon uspješnog dobavljanja datoteke n.mp3 od druga tri klijenta. U ovom primjeru svaka dretva je uspješno skinula svoj dio i nije bilo potrebe za dodatnim pokušajima skidanja nekog dijela. Nakon što korisnik odluči završiti rad klijenta potrebno je zatvoriti prozor klijenta pri čemu će se poslužitelju poslati poruka da se izbriše iz liste. Prilikom zatvaranja prvo će se pojaviti dijalog sa pitanjem želi li korisnik sigurno zatvoriti program i ako se odabere opcija “No” klijent će nastaviti sa radom i poruka za brisanjem neće biti poslana.



Slika 4.5 – izgled klijenta nakon uspješnog skidanja datoteke sa ispisom uspješnog skidanja pojedinih dijelova datoteke

5. Zaključak

Raspodijeljena razmjena datoteka je vrlo učinkovit model razmjene datoteka kojemu popularnost raste već godinama. Veliki dio prometa ostvarenog na Internetu pripada raznim mrežama koje dijele datoteke prema raspodijeljenom modelu. Današnje P2P mreže su vrlo pogodne za razmjenu multimedijskog sadržaja, a njihova otpornost na kvarove i veliki stupanj paralelizma pri skidanju su odličan izbor za skidanje velikih datoteka. Mali zahtjevi potrebni za dijeljenje datoteka omogućuju lagano distribuiranje sadržaja bez potrebe za skupim poslužiteljima i brzim vezama. Iako imaju svojih nedostataka, ove ali i mnoge druge prednosti omogućuju daljnji rast u popularnosti i razvoju takvih mreža i protokola. Model razvijen u sklopu ovog rada demonstrira jednostavnost i prednosti korištenja P2P mreže za dijeljenje datoteka svih veličina.

Unaprijeđenjem veza korisnika prema Internetu danas se sve više koriste usluge slanja i gledanja multimedijskog sadržaja u realnom vremenu. Trenutno se ta usluga većinom gradi pomoću arhitektura klijent-poslužitelj koje zbog svojih karakteristika postaju sve manje pogodne za taj posao. Rastom korisnika poslužitelji su sve opterećeniji. P2P mreže osim tradicionalnog dijeljenja datoteka počinju se predstavljati kao dobro rješenje i za taj sve popularniji način razmjene sadržaja. Jedan od problema koji se javlja pri korištenju P2P-a za takve usluge je nepouzdanost čvorova. Svaki čvor koji sudjeluje u prijenosu sadržaja se može svakog trena odspojiti. Zato je potrebno usavršiti uslugu da postane otporna na odspajanje čvorova, ali i na neke druge probleme.

6. Literatura

[1] http://webopedia.internet.com/TERM/p/peer_to_peer_architecture.html 10.6.2009.

[2] http://www.webopedia.com/DidYouKnow/Internet/2005/peer_to_peer.asp 10.6.2009.

[3] http://en.wikipedia.org/wiki/Gnutella 10.6.2009.

[4] http://en.wikipedia.org/wiki/Distributed_hash_table 10.6.2009.

[5] http://en.wikipedia.org/wiki/Jamendo 10.6.2009.

[6] http://en.wikipedia.org/wiki/FastTrack 10.6.2009.

[7] http://www.slyck.com/story1019.html 10.6.2009.

[8] http://thepiratebay.org/ 10.6.2009.

[9] http://en.wikipedia.org/wiki/Torrent_tracker 11.6.2009.

[10] http://en.wikipedia.org/wiki/The_Pirate_Bay_trial 11.6.2009.

[11] http://en.wikipedia.org/wiki/Usenet 11.6.2009.

[12] http://en.wikipedia.org/wiki/Napster 11.6.2009.

[13] http://en.wikipedia.org/wiki/File_sharing 11.6.2009.

[14] http://en.wikipedia.org/wiki/BitTorrent_index 12.6.2009.

[15] http://en.wikipedia.org/wiki/Advanced_Direct_Connect 12.6.2009.

Sažetak

Naslov: Razmjena datoteka u raspodijeljenoj mreži

U ovom radu analizirani su moderni pristupi razmjeni datoteka pomoću raspodijeljenih protokola. Navedene su prednosti i nedostaci raspodijeljene razmjene datoteka i vrste mreža koje se danas koriste za taj zadatak. Za dodatnu analizu takvih mreža u sklopu ovog rada razvijen je jednostavan model raspodijeljene razmjene datoteka koji se sastoji od centralnog poslužitelja koji vodi listu datoteka i klijenata koji obavljaju raspodijeljenu razmjenu. Protokol za komunikaciju između komponenti koristi tekstualne poruke i klijenti razmjenjuju datoteke u dijelovima pomoću dretvi koje paralelno skidaju pojedini dio.

Ključne riječi: raspodijeljena mreža, raspodijeljena arhitektura, razmjena datoteka, BitTorrent, Advanced Direct Connect, Gnutella, decentralizirana mreža.

Title: File sharing in peer to peer network

This document discusses modern peer to peer file sharing protocols. It looks at advantages and disadvantages of peer to peer networks and types of peer to peer networks that are in widespread use on the Internet today. To better analyse peer to peer network a simple simulation has been developed. It contains central server which keeps track of files shared by clients and clients that are exchanging files over peer to peer protocol. All communication with that protocol is text based and files are exchanged in parts. Each part is downloaded by a different thread and threads are working in parallel to download all parts.

Keywords: peer to peer network, peer to peer architecture, file sharing, BitTorrent, Advanced Direct Connect, Gnutella, decentralized network.