SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

ZAVRŠNI RAD BR. 97

Raspodijeljena razmjena datoteka

HRVOJE KOSTIĆ

Zagreb, lipanj 2008.

1. Uvod

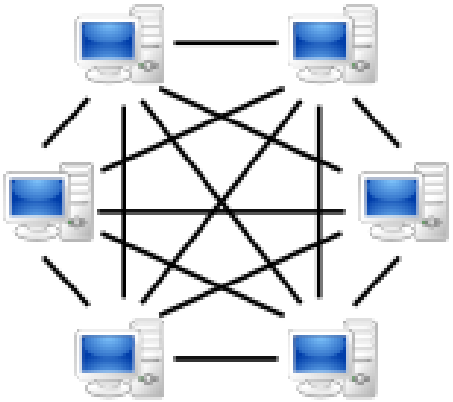
Posljednjih godina povećanjem dostupnosti brzog Interneta, među korisnicima raste zanimanje za dohvaćanje raznovrsnog multimedijalnog sadržaja. Kako dohvaćanje multimedijalnog sadržaja uglavnom troši velike količine poslužiteljskih sredstava, bilo je potrebno rasteretiti poslužitelje te što veći dio posla preusmjeriti na same klijente.

Zadatak ovog završnog rada je proučiti različite modele razmjene datoteka koje se koriste u današnjim P2P (engl. *Peer-to-Peer*) aplikacijama te osmisliti i implementirati jednostavan model razmjene datoteka. Model je napravljen pod operacijskim sustavom Ubuntu 7.10 [4]. Osnovni dijelovi modela su klijenti koji sudjeluju u razmjeni datoteka te poslužitelj koji ima posredničku ulogu.

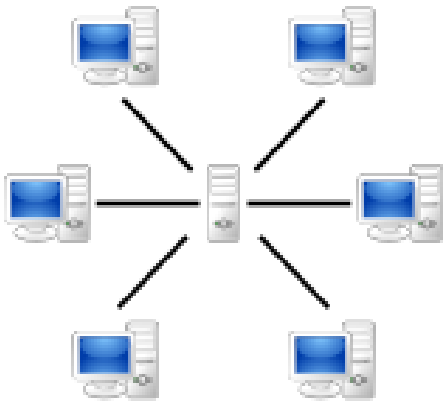
U drugom i trećem poglavlju se detaljnije upoznaje se načelima raspodijeljenog prijenosa datoteka. Navode se prednosti i nedostatci raspodijeljenog prijenosa. U četvrtom poglavlju se razrađuje model jednostavnog prijenosa datoteka između klijentima. Komunikaciju koordinira poslužitelj zadužen da sprema informacije o datotekama te obrađuje upite klijenata.

2. P2P arhitektura

P2P je vrsta arhitekture kojoj je glavna osobina međusobno spajanje i komuniciranje između različitih korisnika (klijenata) u različitim čvorovima mreže (engl. *peer*). Mogućnosti koje pruža su bolje iskorištavanje propusnosti i brzine mreže (engl. *bandwidth*) naspram uobičajenog centraliziranog načina. Kod centraliziranog načina dolazi do zagušenja i nemogućnosti potpunog iskorištavanja mrežnih značajki zbog relativno malog broja poslužitelja koji nisu u stanju posluživati veliki broj klijenata bez gubitaka na brzini prijenosa informacija. Ovakav način, u kojemu nema posebne mrežne infrastrukture, je pogodan za npr. raspodijeljenu razmjenu datoteka ili prijenos informacija u stvarnom vremenu (telefonski razgovori).



Slika 2.1 P2P arhitektura



Slika 2.2 Klijentsko-poslužiteljska arhitektura

Prava P2P mreža nema obilježja klijenata ili poslužitelja, nego svi čvorovi istovremeno djeluju i kao klijenti i kao poslužitelji (za ostale čvorove/klijente). Ta osobina je glavna razlika naspram klijentsko-poslužiteljske arhitekture gdje komunikacija uglavnom djeluje od/prema glavnom, centralnom poslužitelju. Tipičan primjer prijenosa datoteka koji nije P2P je FTP (engl. *File Transfer Protocol*) gdje klijent šalje zahtjev te poslužitelj obrađuje njegov upit i odgovara mu željenom datotekom.

Prva komercijalno korištena P2P mreža je između poslužitelja diskusijskih grupa (engl. *Usenet news server*), gdje su poslužitelji međusobno komunicirali radi razmjene raznih članaka preko cijele Usenet mreže. Ipak, takav poslužitelj je također bio klijentsko-poslužiteljske arhitekture jer su mu korisnici izravno pristupali kako bi pročitali ili napisali članak. Slična struktura se koristi i u SMTP (engl. *Simple Mail Transfer Protocol*) gdje je P2P arhitektura između poslužitelja elektroničke pošte (engl. *Mail Transfer Agenta*) dok je klijentsko-poslužiteljska arhitektura za izravno spajanje korisnika pri dohvatu e-pošte.

Neke mreže i kanali kao npr. *Napster*, *OpenNAP* i *IRC* koriste klijent-poslužitelj arhitekturu za neke aktivnosti (npr. pretraživanje) dok za sami prijenos koriste P2P arhitekturu. Mreže poput *Gnutella* ili *Freenet* koriste samo P2P.

P2P mreže mogu biti razvrstane po:

a) Području primjene:

- Raspodjela datoteka,
- Telefonija,
- Strujanje multimedijskog sadržaja (Multimedia streaming),
- Forumi za raspravljanje (Discussion forums),

b) Stupnju centralizacije:

- klijenti su jednaki, izmjenjujući uloge klijenta i poslužitelja,
- Nepostojanje centralnog poslužitelja koji upravlja mrežom,
- Nepostojanje centralnog usmjerivača,
- Mnogi drugi hibridni P2P sustavi.

2.1 Prednosti P2P arhitekture

Najvažnije prednosti P2P mreža su u tome što svi klijenti pružaju sredstva kao npr. brzinu prijenosa, veličinu diskovnog prostora te računalnu snagu. Kako se broj klijenata povećava, tako se povećava i kapacitet samog sustava. U klijent-poslužitelj arhitekturi se povećavanjem klijenata (sa stalnim brojem poslužitelja) značajke znatno smanjuju.

Robusnost cijelog sustava je također povećana zbog neovisnosti sustava o jednoj točki, tj. ako neki klijent zataji može se nastaviti prijenos sa ostalih klijenata, dok kad bi poslužitelj zatajio komunikacija se ne bi mogla nastaviti do ponovnog podizanja poslužitelja.

2.2 Mreže i protokoli

Tablica 2.1 prikazuje neke trenutno aktualne aplikacije vezane uz određeni P2P protokol te prikazuje njihovu glavnu upotrebu.

Tablica 2.1 Trenutno aktualne aplikacije za P2P protokol

Mreža ili protokol	Upotreba	Aplikacije
BitTorrent	Raspodjela datoteka/Distribucija softvera	ABC, AllPeers, Azureus, BitComet, BitLord, BitTornado, BitTorrent, Burst!, Deluge, FlashGet, G3 Torrent, Halite, KTorrent, LimeWire, MLDonkey, Opera, QTorrent, rTorrent, TorrentFlux, Transmission, Tribler, µTorrent, Thunder, Shareaza
Direct Connect	Raspodjela datoteka	DC++, NeoModus Direct Connect, SababaDC, BCDC++, ApexDC++, StrongDC++
eDonkey	Raspodjela datoteka	aMule, eDonkey2000 (odbačen), eMule, eMule Plus, FlashGet, iMesh, Jubster, lMule, MLDonkey, Morpheus, Pruna, xMule, Shareaza
FastTrack	Raspodjela datoteka	giFT, Grokster, iMesh Kazaa, KCeasy, Mammoth, MLDonkey, Poisoned
Freenet	Distribuirana pohrana podataka	Entropy (njegova mreža), Freenet
Gnutella	Raspodjela datoteka	Acquisition, BearShare, Cabos, FrostWire, Gnucleus, Grokster, gtk-gnutella, iMesh, Kiwi Alpha, LimeWire, MLDonkey, Morpheus, Poisoned, Swapper, XoloX, Shareaza
JXTA	Peer applications	Collanos Workplace (Teamwork software), Sixearch
Napster	Raspodjela datoteka	Napigator, Napster
OpenNap	Raspodjela datoteka	WinMX, Utatane, XNap, Napster
P2PTV	Video strujanje ili Raspodjela datoteka	TVUPlayer, Joost, CoolStreaming, Cybersky-TV, TVants, PPLive, Kontiki, LiveStation
Usenet	Distribuirana diskusija	NewsGreed, Pan i još mnogi drugi.

3. BitTorrent protokol

BitTorrent je P2P protokol za distribuiranu razmjenu datoteka. Omogućava distribuciju veliku količinu podataka bez prisustva originalnog izvora pri tome smanjujući cijenu hardvera, količinu poslužiteljskog prometa. Kada su podatci distribuirani pomoću BitTorrent protokola, svaki klijent pruža dio datoteke novom klijentu, smanjujući tako cijenu i opterećenje jedinstvenog izvora i smanjuje ovisnost o jednom izvoru pritom dodavajući redundanciju koja u ovom načinu komunikacije nije toliko bitan čimbenik.

3.1 Način rada BitTorrent protokola

Prije samog opisa rada BitTorrent protokola potrebno je objasniti osnovne definicije potrebne za razumijevanje. BitTorrent klijent je program koji ima podržano traženje, prijenos i pripremanje svake datoteke za slanje preko mreže koristeći pritom BitTorrent protokol. Klijent (engl. *peer*) je bilo koje računalo koje ima pokrenut instance klijenta. Posrednik (engl. *tracker*) je računalo koje koordinira distribucijom datoteke. Izvor (engl. *seeder*) je računalo koje posjeduje sve dijelove neke datoteke. Inicijalni izvor je računalo koje posjeduje izvornu kopiju datoteke.

Za dijeljenje datoteka potrebno je prvo napraviti malu datoteku, tzv. "*torrent*", koja sadrži opis datoteke koja će se dijeliti te informacije o posredniku. Klijenti koji žele dohvatiti određenu datoteku prvo dohvate torrent datoteku koja sadrži opis te se spoje na definiranog posrednika, koji im odgovara koji klijenti posjeduju dijelove te datoteke.

BitTorrent protokol se malo razlikuje od samog HTTP protokola u nekoliko bitnih stvari jer on:

- radi puno malih zahtjeva preko različitih TCP utičnica, dok HTTP protokol napravi jedan HTTP GET zahtjev preko jedne TCP utičnice,
- dohvaća nasumično dijelove datoteke ili "najrjeđe-prvo" što povećava dostupnost datoteke, dok HTTP dohvaća slijedno.

Navedene razlike (uz ostale) postižu manju cijenu BitTorrent protokola, veću robusnost naspram običnog HTTP-a uz puno veću redundanciju. Međutim, sve to

ima svoju cijenu. Da bi se postigla maksimalna brzina dohvata potrebno je inicijalno određeno vrijeme radi uspostavljanja što većeg broja konekcija. Također je potrebno određeno vrijeme da bi novo priključeni klijent postao efektivan davatelj.

3.2 Stvaranje i objavljivanje torrenta

Klijent koji distribuira datoteku gleda na nju kao broj podjednako podjeljenjih dijelova veličine od 64kB – 4MB. Svaki dio ima određeni zaštitni dio (engl. *checksum*) koristeći pritom SHA1 [3] algoritam. Zaštitni dio se također zapisuje u torrent datoteku. Kada su veličine dijelova veće (npr. od 512 kB) smanjuje se veličina torrent datoteke, ali se smanjuje učinkovitost samog protokola. Kada klijent primi određeni dio radi svoj zaštitni dio te ga uspoređuje sa dobivenim. Ako su zaštitni dijelovi jednaki dio se prihvaća, inače ga odbacuje i ponovo potražuje taj dio.

Formati torrent datoteke ovise o verziji BitTorrent protokola, ali po dogovoru sufiks torrent datoteke je .torrent. Posjeduje „oglašivački“ (engl. *announce*) dio u kojem se nalazi URL posrednika, „info” dio u kojem se nalaze preporučeni nazivi imena datoteke, njezina duljina, veličina dijela i zaštitu za svaki dio.

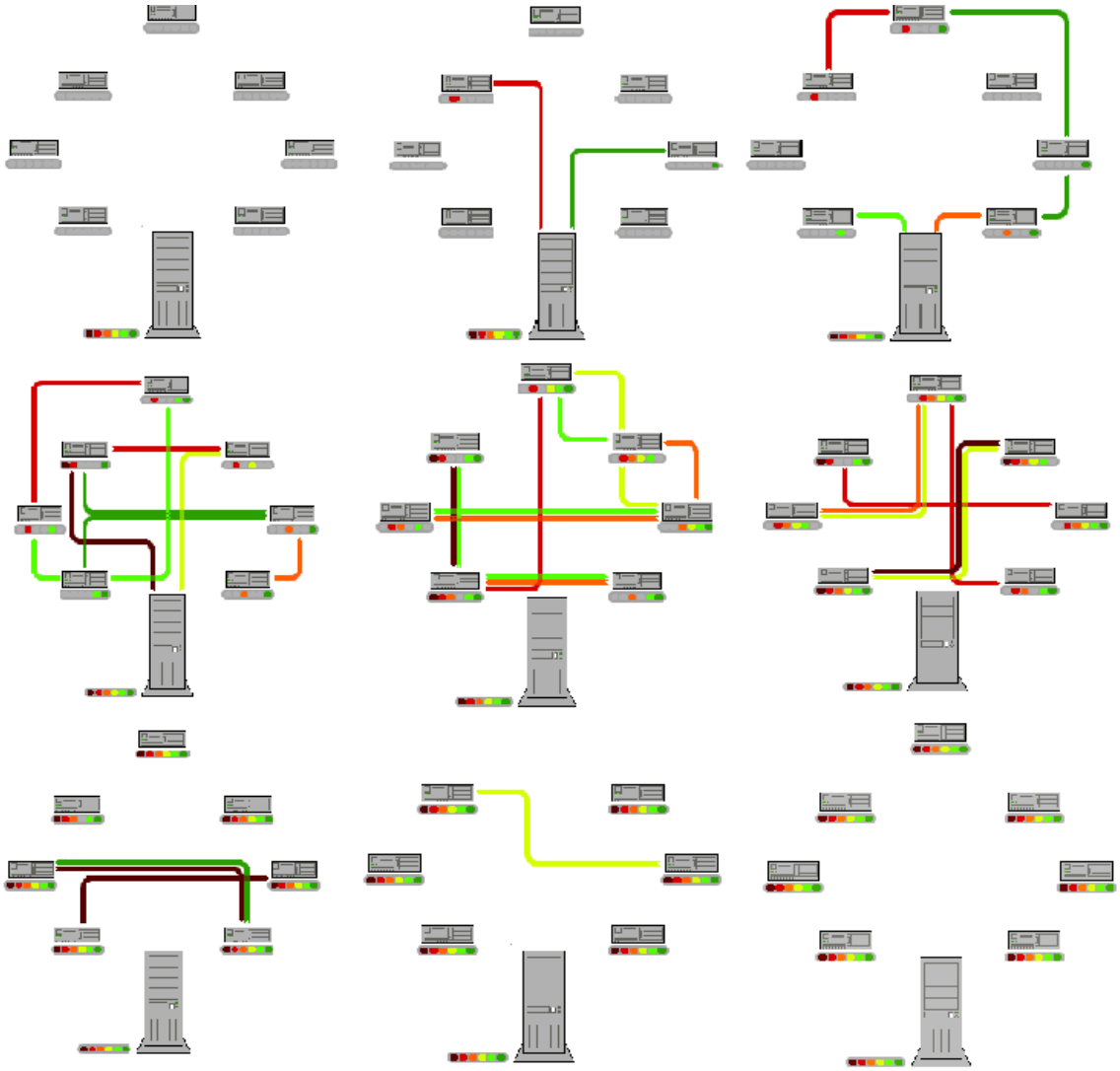
Cjelokupne torrent datoteke su uglavnom objavljene na web stranicama i registrirane na posredniku. Posrednik sadrži listu klijenata koji trenutno sudjeluju u prijenosu datoteke. Alternativno, u sustavima bez posrednika svaki klijent može obavljati funkciju posrednika.

3.3 Dohvaćanje torrenta i dijeljenje datoteka

Korisnici pretražuju internet za željenim torrentom, dohvate ga te pokrenu u lokalnom BitTorrent klijentu. Klijent se spaja na posrednika koji je zapisan u torrent datoteci, od kojeg prima listu klijenata koji trenutno dohvaćaju dijelove željene datoteke. Zatim se klijent spaja na te klijente te potražuje željene dijelove. Grupe međusobno spojenih klijenata koji razmjenjuju datoteku se nazivaju roj (engl. *swarm*). Ako se roj sastoji samo od inicijalnog izvora, klijent se direktno spaja na njega i potražuje od njega dijelove. Kad se novi klijenti spoje u roj, počinju izmjenjivati međusobno dijelove, umjesto direktnog sanjanja na inicijalni izvor.

Klijenti posjeduju mehanizme za optimiranje odnosa dohvaćanja i pružanja datoteke. Npr. dohvaćaju nasumično dijelove datoteke kako bi se povećala mogućnost razmijene dijelova, što je moguće jedino ako klijenti posjeduju različite dijelove datoteke.

Učinkovitost raspodjele datoteka bitno ovisi o načinu na koji klijent odlučuje kome će slati dijelove, a kome ne. Klijenti preferiraju slati dijelove datoteke drugim klijentima koji im vraćaju druge dijelove što im osigurava razmjenu. Ali čvrsto držanje takvom načinu smanjuje učinkovitost raspodjele jer npr. novo spojeni klijent ne može primati podatke jer ne posjeduje nikakve dijelove za razmjenu, Drugi primjer je kada dva klijenta sa dobrom vezom ne razmjenjuju dijelove zbog toga što niti jedna strana neće započeti komunikaciju. Da bi se takve situacije spriječile BitTorrent klijenti koriste mehanizam „optimistično smanjivanje zagušenosti" (engl „*optimistic unchoking*") koji koristi dio svoje širine pojasa (engl. *bandwidth*) za slanje dijelova nasumičnim klijentima u nadi da će otkriti nove i bolje partnere i pritom omogućiti novo pridošlim klijentima mogućnost priključenja u roj.



3.1 Grafički prikaz simulacije rada BitTorrent protokola

Slika 3.1 grafički prikazuje kako BitTorrent djeluje. Na slikama obojene točkice prikazuju dijelove koje pojedini klijent posjeduje. Nakon što inicijalni izvor pošalje dijelove datoteke, dijelovi se međusobno prosljeđuju među klijentima.

3.4 Pravni detalji

Postoje velike nesuglasice u vezi legalnosti prijenosa datoteka i samih poslužitelja koji posjeduju torrente. Torrenti sami po sebi ne sadrže zaštićene podatke, tako da BitTorrent portokol sam po sebi nije ilegalan. Trenutno se vodi mnogo parnica protiv stranica koje udomljuju BitTorrent posrednike. Osim sudskim postupcima, borbu protiv torrenta vode i neki davatelji Internet usluga (*ISP*).

Iako BitTorrent posjeduje dvije bitne nepogodnosti za dijeljenje zaštićenih datoteka, svejedno se njime prenose velike količine ilegalnog sadržaja. Prva nepogodnost je teško pronalaženje torrent datoteka jer ne postoji implementirana tražilica, nego samo što se pronađe na web tražilicama. Druga stvar je ta što BitTorrent ne pokušava sakriti centralno računalo odgovorno za dijeljenje datoteke. Kada osoba želi napraviti torrent datoteku mora u nju upisati adresu centralnog računala što omogućuje pravne aktivnosti da se ta datoteka i ukloni sa poslužitelja.

3.5 Ograničenja i sigurnost

Razmatranje ograničenja i sigurnosti možemo gledati sa nekoliko aspekata.

a) Anonimnost

- BitTorrent ne osigurava anonimnost korisnika te je moguće saznati IP adresu svih članova koji sudjeluju u dijeljenju. Saznavanjem IP adrese se korisnici izlažu mogućem napadu na njihovo računalo.

b) Ograničenje sporih veza

- BitTorrent je osmišljen za stalno spojene veze i nepogodan je za korisnike s analognim modemima zbog čestog odspajanja i malih brzina.

c) Problem dijeljenja nakon dovršetka dohvata (engl. *The Leech Problem*)

- Nakon što korisnici dohvate datoteku više nemaju želje za dijeljenjem, tj. odspajaju se iz mreže. Posljedica toga je smanjivanje grupe ljudi koja ima tu datoteku. Neke web stranice pokušavaju to kompenzirati tako da vode evidenciju o skidanju i slanju podataka te nove i popularnije datoteke daju samo korisnicima sa boljim omjerim. Također korisnicima s manjim odnosom se blokira brzina dok ne pošalju više podataka. Postoji još jedan način zadržavanja korisnika. Neki klijenti promišljeno zadržavaju određeni dio datoteke za sebe, a sve ostale dijelove normalno daju na raspolaganje. Nakon nekog vremena, kada se poveća broj potencijalnih izvora puste u opticaj i taj zadnji dio.

d) Problem nepoštenih klijenata (engl. *The Cheater Problem*)

- Postoje klijenti poput *BitThief* koji omogućavaju dohvaćanje datoteke bez slanja svojih dijelova. Razlog postojanja takvog klijenta je što se ponekad ostvari veća brzina dohvaćanja, ali to izrazito negativno utječe na prirodu BitTorrent protokola.

4. Simulacija jednostavnog BitTorrent protokola

Implementacija jednostavnog BitTorrent protokola se sastoji iz dva dijela.

U prvom dijelu je potrebno osmisliti jednostavni model poslužitelja koji će voditi evidenciju o detaljima datoteka koje određeni aktivni klijent posjeduje. Osim detalja o datotekama poslužitelj također mora imati pohranjene adrese i vrata klijenata. Druga bitna mogućnost poslužitelja je odgovaranje klijentima na njihove zahtjeve. Klijent šalje u strukturi torrent_info ime datoteke, odnosno detalje o datoteci koju želi dohvatiti. Kada poslužitelj prihvati upit ispituje o kakvom se zahtjevu radi. Ako je upit bio samo ime datoteke, pretražuje listu svih datoteka te sve podudarnosti sprema u listu *peer_list* koju će kasnije kompatno prepisati u strukturu *peers* koju je moguće prenjeti preko mreže (Lista je razbacana po memoriji te ju nije moguće samo proslijediti klijentu). Ako je upit bio samo o detaljima, tada se stvar malo komplicira. Problem nastaje jer određene ključne riječi u opisu mogu vratiti nekoliko različitih datoteka. Tada poslužitelj odgovara klijentu sa svim nađenim detaljima te korisnik mora odabrati koju datoteku želi. Nakon primljenog odgovora pokreće se sličan postupak kako i kod pretrage po imenu, samo se u ovom slučaju pretražuje po potpuno istim detaljima. Nakon obavljene pretrage poslužitelj šalje odgovor klijentu. Odgovor se sastoji od adresa i portova odgovarajućih klijenata.

Drugi dio se sastoji od implementacije modela klijenta. Klijent treba imati mogućnost lokalnog pretraživanja torrent datoteka kako bi mogao poslati svoje informacije poslužitelju. Klijent mora imati barem jednu datoteku za dijeljenje kako bi poslužitelj mogao obraditi njegov upit. Nakon što pronađe dijeljene datoteke i pošalje ih poslužitelju, može zatražiti datoteku. Nakon što dobije odgovor od poslužitelja, prividno se dijeli na sva dijela. Dio koji će služiti za obradu klijentskih zahtjeva i dio koji će služiti da dohvaćanje željene datoteke. Dio koji služi za dogvaćanje se slijedno spaja klijente koje je primio u odgovoru i traži dijelove datoteka koje mu nedostaju. Nakon što prođe sve klijente, rezultat može biti da je datoteka potpuno prenesena ili da nedostaju pojedini dijelovi. Ako nedostaju dijelovi klijent se može kasnije spojiti na poslužitelja i ponovo zatražiti istu datoteku te će skinuti samo dijelove koji mu nedostalu. Nakon prijenosa klijentski dio završava, ali poslužiteljski dio nastavlja obradu sve do izlaska iz programa.

4.1 Klijentske strukture podataka

Struct torrent je struktura koja sadrži osnovne podatke o torrentima koji će se dijeliti. To su ime, opis torrenta, ime posrednika (ostavljeno za buduće verzije), veličinu datoteke, zaštitu cijele datoteke, broj dijelova te veličinu samog dijela.

```
struct torrent {
    char fname[MAX_SIZE];
    char fileinfo[MAX_SIZE];
    char server[MAX_SIZE];
    long filesize;
    unsigned char hash[MD5_DIGEST_LENGTH];
    int parts;
    int part_size;
};
```

Struktura chunk sadrži zaštitu pojedinog dijela datoteke. MD5_DIGEST_LENGTH je duljina hash koja je u ovom slučaju 16.

```
struct chunk {
    unsigned char hash[MD5_DIGEST_LENGTH];
};
```

Struktura torrent_info sadrži i ime i opis datoteke. Služi pri komunikaciji poslužitelja i klijenta kada klijenta šalje zahtjev poslužitelju za željenom datotekom. Struktura ne može biti prazna. Može se pretraživati po imenu datoteke, detaljima ili imenu i detaljima.

```
struct torrent_info {
    char fname[MAX_SIZE];
    char fileinfo[MAX_SIZE];
};
```

Struktura peer_torrent služi pri kreiranju torrent datoteke i pri komunikaciji klijenata pri razmjeni datoteka. Sastoji se od strukture torrent, chunk i polja charova koji opisuju da li klijent posjeduje određeni dio datoteke ili ne. Ako je neki element vrijednosti 1 tada klijent posjeduje taj dio.

```
struct peer_torrent {
    struct torrent *torrent;
    struct chunk *part_hash;
    unsigned char *parts_have;
};
```

Struktura torrent je jednostruko povezana lista koja sadrži sve torrente koje klijent sadrži.

```
struct torrent_list {
    struct torrent *torrent;
    struct torrent_list *next;
};
```

Struktura t_list je pokazivač na početak liste torrent.

```
struct t_list {
    struct torrent_list *head;
};
```

Struktura client_torrents sadrži kompaktno zapisane informacije o torrentima koje klijent posjeduje (samo ime i detalje), broj torrenta koji klijent posjeduje, te informacije o klijentovoj adresi i portu na kojemu prima zahtjeve od ostalih klijenata. Struktura služi za slanje poslužitelju kako bi imao u evidenciji što koji klijent posjeduje. Broj torrenta služi radi mogućnosti alokacije dovoljne količine memorije za sve strukture torrent_info koja se prenosi kao polje sa 2 elementa.

```
struct client_torrents {
    struct sockaddr_in my_addr;
    int torr_num;
    struct torrent_info torrent[1];
};
```

Struktura peers je odgovor poslužitelja na zahtjev klijenta o klijentima koji posjeduju željenu datoteku. Slično se alocira kao i struktura client_torrents.

```
struct peers {
    int peer_num;
    struct sockaddr_in host_addr[1];
};
```

4.2 Poslužiteljske strukture podataka

Struktura torrent_info sadrži ime ondosno detalje datoteke koje klijent potražuje.

```
struct torrent_info {
    char fname[MAX_SIZE];
    char fileinfo[MAX_SIZE];
```

```
};
```

Struktura client_torrents sadrži kompaktno spremljenu listu u polje. Poslužitelj prima tu strukturu od klijenta i tako saznaje koje datoteke on posjeduje.

```
struct client_torrents {
    struct sockaddr_in my_addr;
    int torr_num;
    struct torrent_info torrent[1];
};
```

Struktura clients je lista svih klijenata i svih datoteka o kojima poslužitelj vodi evidenciju.

```
struct clients {
    struct client_torrents *client;
    struct clients *next;
};
```

Struktura clients_head sadrži pokazivač na glavu prethodno navedene liste svih datoteka.

```
struct clients_head {
    struct clients *head;
};
```

Struktura peer_list je lista koja sadrži klijente koji posjeduju prethodno traženu datoteku. Lista se stvara nakon primanja upita.

```
struct peer_list {
    struct sockaddr_in host_addr;
    struct peer_list *next;
};
```

Struktura peers je kompaktno spremljena lista klijenata koja se šalje kao odgovor klijentu na traženi upit

```
struct peers {
    int peer_num;
    struct sockaddr_in host_addr[1];
};
```

4.3 Detaljan opis rada poslužitelja

```
Početak
   učitaj preko komandne linije broj vrata;
```

```
    stvori utičnicu;
    poveži utičnicu na adresu;
    slušaj konekcije klijenata;
    inicijaliziraj listu_torrenta;
    radi zauvijek {
        čekaj konekciju;
        napravi novu klijentsku utičnicu;
        ako (konekcija došla){
            obradi zahtjev(nova_utičnica, lista_torrenta);
        }
    }
Kraj

Funkcija obradi zahtjev (utičnica, lista_torrenta){
    Primi listu klijentovih torrenta;
    Spremi ju u lista_torrenta;
    Primi strukturu torrent_info sa podatcima o datoteci;
    Pročitaj što klijent traži;
    Stvori peer_list sa detaljima klijenata koji
    posjeduju traženu datoteku;
    Peer_list = find peers(lista_torrenta, ime_datoteke);
    Kompaktno spremi peer_list u strukturu peers;
    Pošalji klijentu strukturu peers kao odgovor;
    Kraj;
}
Funckija find peers(lista_torrenta, detalji_datoteke) {
    Provjeri da li se pretražuje po imenu ili detaljima;
    Ako se pretražuje po imenu {
        Dok ima klijenata{
            Dok ima torrenta u klijentu {
                Ako se imena podudaraju {
                  spremi u listu podatke o klijentu;
                }
            }
        }
    Inače {
        Pronađi sve slične detalje;
        Spremi ih u polje;
        Pošalji upit klijentu koje detalje želi;
        Primi detalje;
        Dok ima klijenata{
            Dok ima torrenta u klijentu {
                Ako se detalji podudaraju {
                  spremi u listu podatke o klijentu;
                }
            }
        }
    Vrati peer_list;
}
```

4.4 Detaljan opis rada klijenta

```
Početak
  učitaj preko komandne linije adresu poslužitelja, broj
  vrata, detalje o željenoj datoteci;
  kreiraj utičnicu za spajanje ostalih klijenata;
  kreiraj utičnicu za spajanje na poslužitelja;
  omogući ponovno korištenje utičnica;
  poveži utičnicu za klijente na bilo koja slobodna vrata;
  poveži utičnicu za poslužitelja na određena vrata;
  ubaci lokalne torrente u listu;
  listu kompaktno spremi u polje;
  pošalji polje torrenta poslužitelju;
  pošalji zahtjev za datotekom;
  ako je zahtjev samo sa detaljima{
      odaberi odgovarajuću datoteku prema detaljima;
      pošalji odgovor poslužitelju;
  }
  primi listu klijenata;
  kreiraj novi proces za slušanje klijentskih zahtjeva {
      radi zauvijek {
          primi upit za datoteku;
          otvori opis datoteke;
          pošalji opis datoteke klijentu;
          pošalji dijelove datoteke koje posjedujem;
          pošalji hash dijelova;
          radi zauvijek{
              primi zahtjev za dijelom;
                  ako je broj dijela terminirajući broj{
                    izađi iz petlje;
              }
              pročitaj iz datoteke;
              pošalji dio;
          }
      }

  Ako je broj klijenata > 0{
      Provjeri da li je zadatak već postojao;
      Ako je postojao {
          Dohvati brojeve dijelova koje posjedujem;
          Flag_postojao = 1;
      }
      Dok ima klijenata {
          Napravi utičnicu za spajanje
          Spoji se na klijenta;
          Primi detalje o datoteci;
          Primi dijelove datoteke koje posjeduje;
          Primi hash dijelova;
          Ako (Flag_postojao == 0)
              Stvori praznu datoteku veličine originalne;
              Postavi sve zastavice dijelova u 0;
          Dok nisu svi partovi pokušani dohvatiti{
```

```
            Ako posjedujem dio{
                Povećaj brojač i skoči na početak petlje;
            }
            Ako je zadnji dio{
                Last_chunk = 1;
            }
            Ako (klijent posjeduje dio&&ako je greška < 2){
                Pošalji zahtjev za dijelom;
                Dohvati dio;
                Izračunaj hash;
                Ako se hash ne podudara sa dobivenim{
                  greška++;
                  postavi da se ponovo traži taj dio;
                  skoči na početak petlje;
                }
                zapiši dio u datoteku;
                zapiši da posjedujem taj dio;
            inače {
                ako je greška == 2 {
                  greška = 0;
                  izađi iz petlje;
                }
            }
        }
        pošalji da je to zadnji paket;

        Provjeri da li je hash cijele datoteke jednak;
        Ako je jednak {
            završi prijenos;
            iskoči iz petlje;
        }
        inače
            zatvori utičnicu;
    }
    Zapiši sve nove podatke u torrent datoteku;
    Obavijesti korisnika da li je cijela datoteka prenesena
    ili samo dijelovi;
  }
Kraj;}
```

4.5 Korištene funkcije i njihovi opisi

U ovom poglavlju će biti detaljnije opisane korištene funkcije radi lakšeg snalaženja u kodu.

4.5.1 Klijentske funkcije

`void` ***DieWithError***`(char *errorMessage)` - funkcija koja ispisuje željenu *errorMesasge* grešku te potom izlazi iz programa.

`void HandleTCPClient(int clntSocket,int processID)` – funkcija koja se pokreće nakon spajanja klijenta na poslužitelja. Ova funkcija je glavna za obradu klijenta. Za pozivanje su joj protrebni prethodno stvorena utičnica i ID procesa koji pokreće tog klijenta.

`int CreateTCPServerSocket(unsigned short port)` – funkcija koja stvara utičnicu, veže ju za određenu adresu i vrata te zapičinje slušanje.

`int AcceptTCPConnection(int servSock)` – funkcija koja čeka zahtjeve za spajanja od strane klijenata.

`ssize_t writen(int fd, const void *vptr, size_t n)` i `ssize_t readn(int fd, void *vptr, size_t n)` su funkcije za zapisivanje u utičnicu, odnosno čitanje iz utičnice. Posebno su napravljene jer ponekad u utičnicu se može zapisati/pročitati manje nego što je zahtijevano, a ova funkcija zapisuje točno onoliku veličinu koju joj zadamo.

`void initlist(struct t_list *list)` – funkcija koja inicijalizira listu tj. postavlja vrijednost glave na NULL.

`void insertfront(struct t_list *list, struct torrent *torrent_add)` - funkcija koja dodaje zadanu strukturu torrent pa početak liste.

`int create torrent file list(char *folder, char *torrentExt, struct t_list *list)` - funkcija koja pretražuje trenunti direktorij za datotekama ekstenzije torrentExt te ih dodaje u listu t_list. Vraća broj pronađenih torrenta.

`void *read torrent(char *filename, unsigned char TORR)` – funkcija koja čita podatke o datoteci iz torrent datoteke u ovisnosti varijable TORR. Ako je TORR==1 tada čita samo osnovne podate npr. veličina, broj dijelova i sl. Ako je TORR==0 tada čita sve podatke uključujući i zaštite pojedinih dijelova te dijelove koje posjeduje.

`char *hash part(char *, int, char *)` – funkcija koja stvara zaštitu za određeni dio datoteke zadane veličine.

`char *hash file(FILE *, char *)` – funkcija koja stavara zaštitu cijele datoteke.

`struct peer_torrent *get_parts(char *filename)` – funkcija koja vraća NULL ako datoteka ne postoji, a ako postoji vraća torrent datoteku s dijelovima koje posjeduje te pripadajućim zaštitama.

4.5.2 Poslužiteljske funkcije

Poslužitelj također posjeduje prethodno navedene funkcije:

- `void DieWithError(char *errorMessage),`
- `void HandleTCPClient(int clntSocket,int processID),`
- `int CreateTCPServerSocket(unsigned short port).,`
- `int AcceptTCPConnection(int servSock) ,`
- `ssize_t writen(int fd, const void *vptr, size_t n),`
- `ssize_t readn(int fd, void *vptr, size_t n) .`

`struct peer_list *find_peers (struct clients_head *clients, struct torrent_info *req_file, char **params, int n_params, int clntSocket)` – funkcija koja pretražuje sve torrente koje posjeduje (struktura clients_head) tražeci prema podatcima iz strukture torrent_info, param su već prirađeni parametri po kojima da traži, broj parametara te utičnica preko koje da komunicira sa klijentom.

`void insert_client(struct clients_head *clients, struct client_torrents *client_add)` – funkcija koja dodaje u svoju evidenciju podatke o novo spojenom klijentu.

4.6 Upute za rad s programom

Za korištenje programa potrebno je prethodno napraviti torrent datoteku. Torrent datoteka se stvara pozivanjem iz komandne linije naredbe:

- ./*createt* ime_ulazne_datoteke kratki_opis_datoteke.

Opis datoteke se sastoji od nekih ključnih riječi. Riječi u opisu moraju biti odvojeni donjom crticom (_). Nakon stvaranja torrent datoteke potrebno je pokrenuti poslužitelja naredbom:

- ./*server* broj_vrata.

Kada je poslužitelj pokrenut možemo pristupiti pokretanju klijenata. Klijenti se pokreću naredbom:

- ./*client* adresa_servera broj_vrata –f ime_tražene_dat –d opis_datoteke

Barem jedna opcija (-f ili –d) mora biti ispunjena, u protivnom će program javiti grešku pri pozivanju.

5. Zaključak

BitTorrent protokol omogućuje brz, učinkovit i pouzdan način razmjene velikih datoteka putem Interneta. U ovom radu se opisuju metode kako ostvariti simulaciju tog protokola.

Radi ostvarenja simulacije bilo je potrebno osmisliti načine prezentacije strukture podataka, načine prijenosa podataka, zaštite od grešaka pri prijenosu, pretragu za željenim datotekama te dohvaćanje pojedinih dijelova datoteke. Iako je ova simulacija jednostavna, zbog složenosti samog protokola, pojavljuje se nekoliko problema kao što su sigurnost klijenata, odabir najboljeg partnera za dohvaćanje te nepoštenih klijenata. Rezultat ovog završnog rada je funkcionalna simulacija otvorena za prilagodbu, nadogradnju i daljnji razvoj.

BitTorrent je pogodan za prijenos velikih datoteka, kakvih je na Internetu sve više zbog raznih multimedijalnih sadržaja, te se može očekivati stalni porast korištenja takvog načina prijenosa podataka.

6. Literatura

[1] Peer-to-Peer, http://en.wikipedia.org/wiki/Peer-to-peer, 4.6.2008

[2] BitTorrent (protocol) http://en.wikipedia.org/wiki/BitTorrent_%28protocol%29 , 5.6.2008

[3] SHA-1, *SHA hash functions* http://en.wikipedia.org/wiki/Sha-1 , 10.6.2008

[4] Ubuntu http://en.wikipedia.org/wiki/Ubuntu_(Linux_distribution) , 11.6.2008

7. Sažetak

7.1 Raspodijeljena razmjena datoteka

BitTorrent je protokol, izgrađen na P2P arhitekturi, koji se sve češće izabire za objavu i dijeljenje velikih datoteka putem Interneta. Razlikuje se od ostalih P2P programa zbog svoje mogućnosti podjele datoteka u manje dijelove.

Cilj ovog rada je bio napraviti model BitTorrent protokola. Model se sastoji od klijenata i poslužitelja. Klijenti mogu međusobno, ali i sa poslužiteljem komunicirati radi dohvata podataka. Datoteka se dohvaća po dijelovima koje određeni klijent posjeduje te se vrši provjera ispravnosti prijenosa. Ukoliko datoteka nije potpuno dohvaćena, prijenos se može nastaviti ponovnim spajanjem na poslužitelja radi dohvaćanja novih informacija ako je pristigao novi klijent sa željenom datotekom.

Ključne riječi: Klijent, poslužitelj, izvor, roj, P2P, posrednik, BitTorrent model, simulacija

7.2 Distributed file sharing

BitTorrent is a P2P file sharing system that is quickly becoming the method of choice for publishing and sharing large files across the Internet. The difference between BitTorrent and other P2P programs is that it breaks down files into smaller packets of information.

The goal of this thesis was to create a model of the BitTorrent protocol. The model consists of clients and a server. Clients can interact amongst themselves or communicate with the server to acquire data. The file is retrieved in parts from certain clients who own that specific part. After the download, the part is validated. If the file is not completely downloaded, the transfer can be continued when a new client who owns the necessary parts of the file connects to the server.

Keywords: Peer, tracker, seed, swarm, P2P, BitTorrent model, simulation