|  |  |  |
| --- | --- | --- |
| |  | | --- | | **УНИВЕРЗИТЕТ У НОВОМ САДУ**  **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**  **НОВИ САД**  **Департман за рачунарство и аутоматику**  **Одсек за рачунарску технику и рачунарске комуникације**  **ИСПИТНИ РАД**  **Кандидат: Алекса Арсић**  **Број индекса: РА119/2015**  **Предмет: Међурачунарске комуникације и рачунарске мреже 1**  **Тема рада: FileMate – једноставни дуал стек програм за брзи пренос датотека**  **Ментор рада: Проф. Илија Башичевић**  **Нови Сад, Децембар 2018.** | |  | |

**Sadržaj**

[1. Zadatak 1](#_Toc533341474)

[2. Koncept rešenja 2](#_Toc533341475)

[3. Opis rešenja 3](#_Toc533341476)

[3.1 IPv4/IPv6 klijentska aplikacija 3](#_Toc533341477)

[3.2 Serverska aplikacija 4](#_Toc533341478)

[4. Testiranje 6](#_Toc533341479)

[5. Zaključak 7](#_Toc533341480)

[6. Literatura 8](#_Toc533341481)

**Spisak slika**

**No table of figures entries found.**

# Zadatak

Po ugledu na program za brzi prenos datoteka (eng. downloaders, GetRight, FlashGet, GoZilla) realizovati aplikacije IPv4 klijenta, IPv6 kliljenta i dual stack servera (prima poruke generisane i preko IPv4 i IPv6 protokola) za prenos segmenta datoteka (od zadate pozicije, u zadatoj dužini) koristeći TCP protokol. Klijent po jednoj vezi prenosi deo datoteke. Formiranjem više istovremenih veza prenosi se cela datoteka, i ubrzava se process prenosa. Server mora da simulira ograničenje brzine po vezi. Klijent po prijemu svih delova datoteke sklapa kompletnu datoteku. Porediti brzinu prenosa pomoću jedne i više veza – grafički prikazati rezultate analize u dokumentaciji. Server istovremeno opslužuje više klijenata.

# Koncept rešenja

Rešenje se zasniva od tri aplikacije – IPv4 klijent, IPv6 klijent i serverska aplikacija; od kojih svaka predstavlja nezavisno implemetiranu celinu. Čitav paket koji se sastoji od dve aplikacije (IPv4/IPv6 klijent i server) nazvali smo *FileMate* *- a simple file sharing application* te u daljem tekstu ako je potrebno sagledati sliku sa nivoa korisnika koristićemo se ovim nazivom.

FileMate osmišljen je tako da korisnik uz pomoć jednog od ponuđenih klijentskih aplikacija uspostavlja vezu sa udaljenim *dual stack* serverom i uz pomoć njega prenosi datoteke koje se nalaze na serverskoj strani. Jedna datoteka prenosi se iz više delova, bilo iterativnim uspostavljanjem veze sa serverom ili uspostavljanjem više istovremenih klijentskih konekcija sa iste IP adrese. Klijent od servera dobija broj delova koji čine jednu datoteku, a korisnik odlučuje koji deo će se u otvorenoj sesiji preneti. Nakon prenošenja svih delova datoteke klijent prelazi u stanje rekonstrukcije originalne datoteke, čime se prenos završava.

Server opslužuje više klijenata istovremeno i na osnovu broja konekcija sa jedne IP adrese reguliše brzinu prenosa. Što je broj konekcija sa iste IP adrese veći to je brzina prenosa veća i obrnuto. Na serverskoj strani datoteka nije rasparčana u delove.

Kroz rešenje FileMate-a implementiran je TCP protokol, a funkcionalno je samo na Win32 sistemima.

Za detaljniji opis gradivnih komponenti FileMate-a pogledati naredno poglavlje.

# Opis rešenja

Kao što je prethodno rečeno FileMate sastoji se od IPv4/IPv6 klijentske aplikacije i serverske aplikacije, u daljem tekstu opisana je svaka aplikacija zasebno.

## IPv4/IPv6 klijentska aplikacija

Suštinska razlika između IPv4 klijentske aplikacije i IPv6 klijentske aplikacije jeste u tome što koriste različite familije IP adresa, dok je implementacija ostatka logike jednaka u oba rešenja.

*IPv4* *adresa* sastoji se od 32 bita i piše se kao četiri decimalna broja razdvojena tačkom. Brojevi između tačaka idu u rasponu od 0 do 255.

Primer: **1.160.10.240**

*IPv6* *adresa* sastoji se od 128 bita heksadecimalnog zapisa razdvojenog dvotačkama.

Primer: **3ffe:1900:4545:3:200:f8ff:fe21:67cf**

Prilikom pokretanja klijentske aplikacije postavljaju se potrebni parametri za uspostavljanje TCP sesije sa serverom. sockaddr\_in(6) strukturom se postavljaju IP adrese servera i klijenta, a postavlja se i socket na kojem klijentska aplikacija osluškuje dolazeće pakete. Parametri se postavljaju atomatski i ne zahtevaju od korisnika njihovo poznavanje i značenje.

Nakon TCP rukovanja klijentska aplikacija i serverska aplikacija su sinhronizovane i spremne za razmenu podataka.

Klijentska aplikacija je od serverske strane dobija broj delova iz kojih se sastoji originalna datoteka koja se prenosi. Ovim saznanjem klijentska aplikacija prepoznaje da je konektovana na server te obaveštava korisnika da je sve spremno za prenos delova datoteke. Ona takođe korisniku daje informaciju o tome iz koliko se delova sastoji datoteka koja se prenosi.

Odabirom rednog broja dela datoteke koja se prenosi, klijentska aplikacija obaveštava server o tome i time započinje prenos dela datoteke. Prenos se završava kada klijentska aplikacija od servera dobije kontrolnu poruku „EOF\0“.

Nakon što je prenesen deo datoteke koja se prenosi, klijentska aplikacija proverava da li je broj fajlova u direktorijumu u kojem su primljeni delovi jednak broju koji je dobila na od serverske aplikacije na početku komunikacije.

U slučaju u kojem su ti brojevi jednaki, klijentska aplikcija poziva funkciju koja na osnovu zaprimljenih delova rekonstruiše originalnu datoteku i ukljanja prethodno zaprimljene delove. Time se klijentska aplikacija uspešno završava te zatvara komunikaciju sa serverom.

U slučaju da broj zaprimljenih fajlova i broj koji je klijentska aplikacija dobila od servera na početku sesije, po zaprimanju željenog dela datoteke ne dešava se ništa. Ovim je takođe uspešno završena i zatvara se komunikacija sa serverom.

## Serverska aplikacija

Prilikom pokretanja serverske aplikacije, kao i kod klijentske aplikacije, postavljaju se potrebni parametri kako bi se ostvarila komunikacija sa klijentskom stranom. sockaddr\_in(6) strukturom postavljaju se IP adresa servera i klijenta, a postavlja se i socket pomoću kojeg server osluškuje poruke sa klijentske strane.

Server može da opslužuje više klijentskih aplikacija istovremeno, a to je implementirano pomoću Thread-ova koji se prave sa svakom novom konekcijom po jedan.

Kada je konekcija uspešno uspostavljena serverska aplikacija upisuje novu konekciju u listu aktivnih konekcija (*Double ended queue*). Lista konekcija sadrži strukture koje čuvaju IP adresu aktivne sesije, kao i broj otvarenih konekcija sa iste IP adrese. Nakon toga server klijentskoj strani šalje broj iz kojih se sastoji datoteka koja se prenosi. (broj iz kojih se sastoji datoteka koja se prenosi je proizvoljno podesiv pre pokretanja serverske aplikacije). Time označava da je spreman za komunikaciju sa klijentskom stranom.

Datoteka se na serverskoj strani čuva u celini, a ne u delovima te server od klijentske strane dobija redni broj dela datoteke koje se prenosi i pomoću njega izračunava odakle i dokle je potrebno izdvojiti deo koji klijent želi. Nakon slanja željenog dela, server klijentskoj strani šalje kontrolnu poruku “EOF\0” nakon određenog vremenskog intervala. Ovom kontrolnom porukom server je označio kraj prenosa željenog dela datoteke.

Pošto se čitav process obrade i slanja odvija u Thread-ovima potrebno je paziti na kritične sekcije i zaključati ih po potrebi upotrebom *mutex*-a kako ne bi došlo do pojave *deadlock*-a. Kritične sekcije na koje nailazimo su:

* Dodavanje aktivne sesije u listu aktivnih konekcija
* Pristupanje članovima liste aktivnih konekcija kod preračunavanja brzine prenosa
* Pristupanje članovima liste aktivnih konekcija prilikom uklanjanja završene sesije

Nakon uspešnog prenosa server zatvara sesiju sa trenutnim klijentom i ukljanja ga iz liste aktivnih konekcija ili dekremetira vrednost strukture člana liste koji označava koliko je aktivnih konekcija sa iste IP adrese.

Server takođe na standardni izlaz daje informaciju o IP adresi na kojoj je obavljen prenos, port i veličinu prenetih podataka u kB.

Brzina prenosa simulirana je na način da se pred svako slanje podataka klijentskoj aplikaciji, Thread trenutne sesije uspava na određeni vremenski period, po formuli:

Gde su:

* sleepInterval – interval uspavljivanja Thread-a
* THREAD\_SLEEP – makro koji označava osnovni interval uspavljivanja Threada
* aConnections – broj trenutno aktivnih konekcija sa iste IP adrese

# Testiranje

# Zaključak

U ovom tekstu dat je opis IPv4/IPv6 klijentske i *dual stack* serverske aplikacije pomoću kojih je omogućen prenos podataka od udaljenog servera ka korisniku korištenjem TCP protokola.

Potencijalni problem koji se može javiti kod pristupa realizacije slanja podataka u ovom rešenju jeste što nemamo konstantno opterećenje mreže već se paketi šalju u određenim intervalima između kojih, u izolovanom okruženju, na mrežni nema saobraćaja. Međutim, za potrebe ovog rešenja ovakva metoda simulacija brzine prenosa je sasvim prihvatljiva.

# Literatura

1. *Priručnik radnog okruženja za pisanje protokola, Verzija 0.2*, Univerzitet u Novom Sadu, Fakultet Tehničkih Nauka, 2007
2. https://www.webopedia.com/DidYouKnow/Internet/ipv6\_ipv4\_difference.html