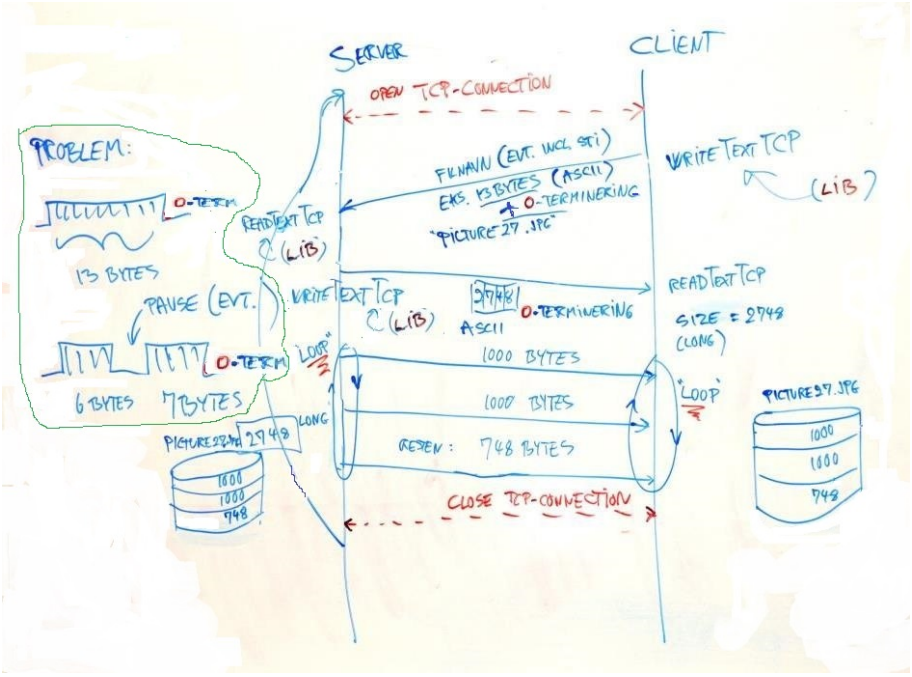
INGENIØRHØJSKOLEN AARHUS

IKN

E, IKT OG EP



Øvelse 8 - Socket Programming

Udarbejdet af:

Simon Thrane Hansen	201500150
Lars Hjerrild	201409555
Kasper Lauge Madsen	201409873

Undervisere:
Torben Gregersen og
Lars Mortensen

28. september 2016

Indhold

1. Indledning

Denne opgave ophandler uarbejdning og test af en TCP-server og TCP-Client.

Serveren skal køre i en virtuel Linux-maskine og kunne supportere en client ad gangen. Serveren skal modtage en tesktstreng fra clienten. Tekststrengen skal indeholde et filnavn, eventuel ledsaget af en stiangivelse. Tilsammen skal informationen i tekststrengen udpege en fil af en vilkårlig type/størrelse i serveren, som en tilsluttet client ønsker at hente fra serveren. Hvis filen ikke findes skal serveren returnere en fejlmelding til client'en. Hvis filen findes skal den overføres fra server til client i segmenter på 1000 bytes ad gangen –indtil filen er overført fuldstændigt. Serverens portnummer skal være 9000. Serveren skal desuden være iterativ og derfor være klar til at supportere en anden client efter en endt filoverførelse.

Clienten skal køre i en anden virtuel Linux-maskine. Denne client skal kunne hente en fil fra den ovenfor beskrevne server. Client'en sender indledningsvis en tekststreng, som er indtastet af operatøren, til serveren. Tekststrengen skal indeholde et filnavn + en eventuel stiangivelse til en fil i serveren. Client'en skal modtage den ønskede fil fejlfrit fra serveren – eller udskrive en fejlmelding hvis filen ikke findes i serveren.

2. Udviklingsforløb

2.1 File-server

I serveren blev, der implementeret en construktor og sendFile-metode.

Construktoren er lavet, så den opretter en ny TCPListener, der lytter på en vilkårlig IP-adresse på en hard-coded Port. Derefter opretter den en . Serveren startes og venter på, at den opretter en TCP-connection med en client. Når forbindelsen med en client er oprettet den en networkstream til clientens socket. Serveren læser derefter, hvilken fil clienten ønsker overført. Derefter tjekker serveren om den ønskede fil findes og sender dette resultat til clienten. Hvis filen findes sendes denne til clienten ved at kalde metoden sendFile.

sendFile (String fileName, long fileSize, NetworkStream io) opretter et array af bytes med en faststørrelse for at overføre filen i stykker af 1000 bytes såfremt dette er muligt. Filen bliver læst ind ved brug af FileStream kommandoer og lagt i en variable, derefter overføres filen over networkStreamen i mindre stykker.

2.2 File-client

I clienten blev, der implementeret en construktor og recieveFile-metode. Constructoren tager to argumenter - et filnavn + stiangivelse og en ip-adresse til serveren. Det første, der sker i constructoren er, at der oprettes en ny TCPClient. Clienten forbinder derefter med en TCP-Server med den angivne IP-adresse og et hard-coded Portnummer. Clienten opretter derefter en networkstream, der skal tales med og sender en forespørgelse over at serveren skal sende den angivne fil til clienten. Clienten modtager derefter svar fra serveren om den angivne fil sendes på serveren. Hvis filens findes på serveren kalder clienten recieveFile og starter modtagelsen af filen. Efter filen er blevet overført lukker clienten forbindelsen til serveren.

```
private file_client (string[] args)
{
long size;
Console.WriteLine ("Client started");
TcpClient clientSocket = new TcpClient ();
```

```
clientSocket.Connect (args[0], PORT);
NetworkStream serverStream = clientSocket.GetStream();
LIB.writeTextTCP (serverStream, args[1]); //Filename skal gives med som argument.
size = LIB.getFileSizeTCP(serverStream);
if(size!= 0)
receiveFile(args[1],serverStream,size);

clientSocket.Close();
}
```

ReceiveFile (String fileName, NetworkStream io, long FileSize) RecieveFile laver en fil med det pågældende filenavn, den har fået med som parameter. Den modtager også en networkstream, hvor dataene skal modtages fra, og så modtager den også størrelsen på filen som parameter. Metoden læser på networkstreamen og overfører dette til en buffer (Der bliver maksimalt aflæst 1000 bytes ad gangen). Når data er kommet ind i bufferen bliver de skrevet til den oprette fil. Denne proces fortsættes ind til, at der totalt er læst ligeså mange bytes som filesize. Man kan se koden for metoden nedenunder:

```
private void receiveFile (String fileName, NetworkStream io, long fileSize)
{
var file = File.Create (fileName);
var buffer = new byte[BUFSIZE];
int bytesRead = 0;
long accumulatedBytes = 0;

while (accumulatedBytes < fileSize)
{
bytesRead = io.Read(buffer,0,BUFSIZE);
accumulatedBytes += bytesRead;

file.Write (buffer, 0, bytesRead);
}
}
```

3. Test af TCP-server og client

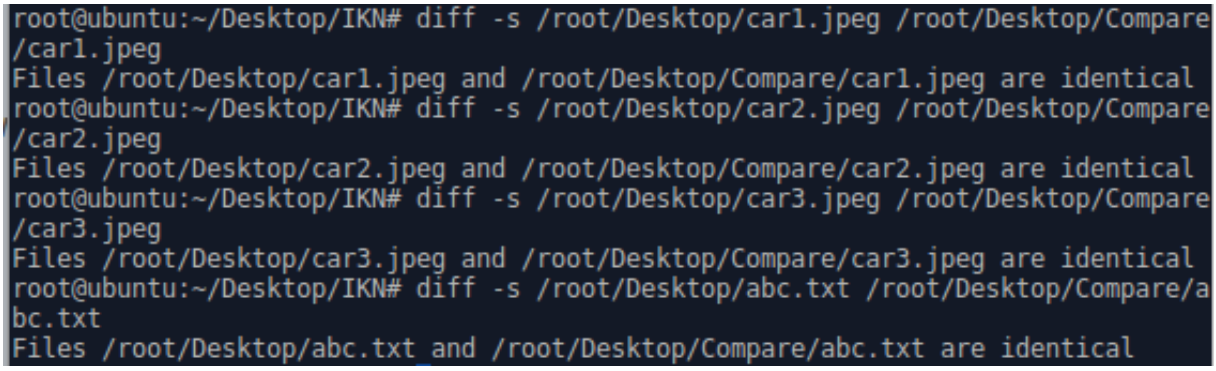
TCP-clienten og serveren blev testet ved at overføre diverse billeder og tekstfiler fra serveren på den ene virtuelle maskine til clienten den anden virtuelle maskine.
Nedenfor ses nogle eksempler på overførelser:

```
root@ubuntu:~/Desktop/IKN# ./Exercise8/file_client/bin/Debug/file_client.exe 10.
0.0.1 /root/Desktop/car2.jpeg
Client starts...
Client started
root@ubuntu:~/Desktop/IKN# ./Exercise8/file_client/bin/Debug/file_client.exe 10.
0.0.1 /root/Desktop/car3.jpeg
Client starts...
Client started
root@ubuntu:~/Desktop/IKN# ./Exercise8/file_client/bin/Debug/file_client.exe 10.
0.0.1 /root/Desktop/abc.txt
Client starts...
Client started
root@ubuntu:~/Desktop/IKN# ./Exercise8/file_client/bin/Debug/file_client.exe 10.
0.0.1 /root/Desktop/abc.txt
Client starts...
Client started
```

Figur 3.1: Eksempler på clientkommandoer

```
root@ubuntu:~/Desktop/IKN# ./Exercise8/file_server/bin/Debug/file_server.exe
Server starts...
 >> Server Started
 >> Data from client - /root/Desktop/car2.jpeg
 >> Data from client - /root/Desktop/car3.jpeg
 >> Data from client - /root/Desktop/abc.txt
 >> Data from client - /root/Desktop/abc.txt
```

Figur 3.2: Eksempler på serversvar på de pågældende clientkommandoer



Figur 3.3: Test af filernes lighed

Som det kan ses på ovenstående figurer kører serveren iterativt. Efter serveren er blevet startet bliver den ved med at køre og lytte efter clients. Clients kan så kalde applikationen med parameterene <ip-adresse> <filnavn>, hvorefter serveren sender denne fil til clienten hvis den findes. Ellers reutneres en fejlbesked. Efter filen er blevet modtaget lukker clienten forbindelsen, og en ny client kan komme til.

4. Konklusion

I denne opgave er der blevet implementeret en TCP-Server og TCP-client i C#. Clienten er blevet lavet, så den kan modtage to argument på en IP-adresse til serveren og et filnavn, der angiver filen, den skal hente os serveren. Serveren er blevet udviklet som en iterativ server, der kan snakke med en vilkårlig IP-adresse på en fastdefineret port (9000). Den udarbejde server og client blev testet ved brug af diff -s kommandoen i Linux for at teste, at de hentede filer og de oprindelige filer var identiske. Underarbejdelse er der opnået erfaringer med følgende fag områder: TCP, Socket Programmering, Client- og Serverbegrebet.