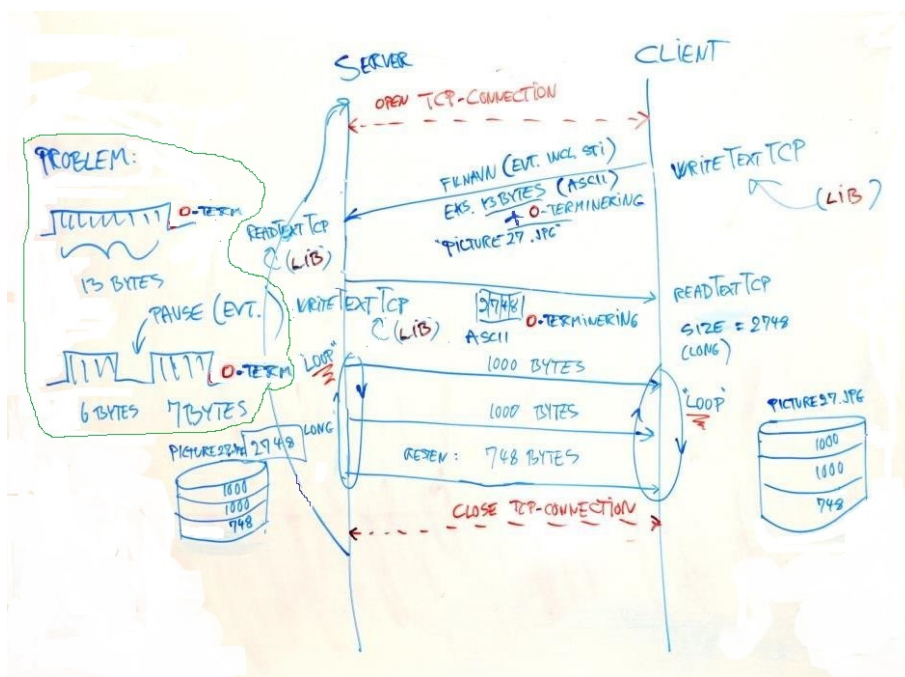


# INGENIØRHØJSKOLEN AARHUS

IKN

E, IKT og EP



## Øvelse 8 - Socket Programming

Udarbejdet af:

Simon Thrane Hansen

Lars Hjerrild

Kasper Lauge Madsen

201500150

201409555

201409873

Undervisere:

Torben Gregersen og

Lars Mortensen

28. september 2016

# Indhold

<b>1</b>	<b>Indledning</b>	<b>2</b>
<b>2</b>	<b>Udviklingsforløb</b>	<b>3</b>
2.1	File-server . . . . .	3
2.2	File-client . . . . .	3
<b>3</b>	<b>Test af TCP-server og client</b>	<b>5</b>
<b>4</b>	<b>Konklusion</b>	<b>6</b>

# 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 tekststreng 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 konstruktor og sendFile-metode.

Konstrukturen 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 klienten ønsker overført. Derefter tjekker serveren om den ønskede fil findes og sender dette resultat til klienten. Hvis filen findes sendes denne til klienten 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 klienten blev, der implementeret en konstruktor og recieveFile-metode. Konstrukturen tager to argumenter - et filnavn + stiangivelse og en ip-adresse til serveren. Det første, der sker i konstruktoren er, at der oprettes en ny TCPClient. Klienten forbinder derefter med en TCP-Server med den angivne IP-adresse og et hard-coded Portnummer. Klienten opretter derefter en networkstream, der skal tales med og sender en forespørgelse over at serveren skal sende den angivne fil til klienten. Klienten modtager derefter svar fra serveren om den angivne fil sendes på serveren. Hvis filen findes på serveren kalder klienten recieveFile og starter modtagelsen af filen. Efter filen er blevet overført lukker klienten 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)

```
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 belv test ved at overføre billedet med stien: /root/desktop/Car1.jpeg fra clienten til serveren ved at kalde følgende kommandoer:

TCP-Serveren blev startet ved at kalde:

TCP-Clienten blev efter følgende startet og sat til at hente filen:

Da filen var blevet overført blev den samme fil overført ved hjælp af mail. Efterfølgende blev de to filer sammenlignet ved brug af den indbyggede compare kommando i linux-terminalen. Resultatet af testen kan ses nedenunder:

## 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.