

Journal for øvelserne 8 & 9

soæidjgæo isdjoī jdsfægij fdoi

Bjørn Nørgaard Sørensen	Joachim Dam Andersen
stud.nr: 201370248	stud.nr: 201370031
bjornnorgaard@post.au.dk	mr.anderson@post.au.dk

Arni Thor Thorsteinsson
stud.nr: 201370318
201370318@post.au.dk

October 26, 2015

Indholdsfortegnelse

1. Øvelse 8 - TCP/IP socket programming	1
1.1. Introduktion	1
1.2. Udviklingsforløb	1
1.3. Funktionalitet	1
1.4. Resultater	2
1.5. Konklusion	2
2. Øvelse 9 - UDP/IP socket programming	3
2.1. Introduktion	3
2.2. Udviklingsforløb	3
2.3. Funktionalitet	3
2.4. Resultater	3
2.5. Konklusion	3
A. Appendix: Sekvens diagrammer	4

1. Øvelse 8 - TCP/IP socket programming

1.1. Introduktion

I denne øvelse i socket programmering, laves en TCP client og TCP server. Vi har valgt at programmere i det objektorienterede sprog C#. Clienten skal kunne forbinde til serveren, og downloade en fil herfra. Clienten og serveren køres på hver sin virtuelle linux maskine. I dette dokument beskrives udviklingsforløbet med tilhørende digrammer og kodeforklaringer.

1.2. Udviklingsforløb

Vi har designet koden således at server og client er opdelt i to hoveddele, hhv. constructor og overføringsfunktionalitet. For serveren betyder dette afsendingsfunktionalitet, og for clienten modtagelsesfunktionalitet. På kodeudsnit: 1 Hoveddesign for server og 2 Hoveddesign for client, kan pseudokoden for metoderne ses. Ønsker nærmere gennemgang af kode, kan source koden findes i medfølgende src.zip fil.

Code 1: Hoveddesign for server

```
1 public FileServer ()
2 {
3     //Setting up server and connecting client
4     //Getting filename an calculating lenght
5     //Sending file
6     //Closing connection
7 }
8 public void SendFile(string filename, long filesize, NetworkStream stream)
9 {
10    //Local variables
11    //Assigning variables
12    //Sending file
13    //Closing connection
14 }
```

Code 2: Hoveddesign for client

```
1 public FileClient ()
2 {
3     //Declaring variables
4     //Assigning variables
5     //Requesting and receiving file
6     //Closing connection
7 }
8 public void ReceiveFile(string filename, NetworkStream stream)
9 {
10    //Declaring variables
11    //Assigning variables and setting up fileStream
12    //Receiving data
13    //Closing connection
14 }
```

1.3. Funktionalitet

Vores design gør det muligt for brugeren at selv indtaste en ønsket TCP servers IP-adresse. Dette giver mulighed for yderligere udvidelse af programmets funktionalitet. Der oprettes nu forbindelse til serveren med den indtastede adresse. Herefter venter clienten på at brugeren indtaster et navn på den fil der ønskes downloadet fra serveren. Clienten anmoder da serveren om den specifikke fil, hvorefter serveren melder tilbage. Hvis filen eksisterer påbegyndes overførslen. Der er udarbejdet en sekvensdiagram der illustrer det overordnede system. Se dette på figur 3 på side 4.

1.4. Resultater

Vi har testet vores system og vedlagt screenshots heraf. På figur 1 ses test af serveren, hvor der sendes to filer. På billedet ses filer til afsendelse, som ligger i fileserverens Debug folder (markeret med lyserød). Billedet illustrerer desuden programflowet med konsoludskrifter.

På figur 2 ses test af clienten fra samme testsekvens som på figur 1. Her modtages to filer, hhv. Herp.jpg og Derp.jpg. Igen illustreres programflowet med konsoludskrifter.

```
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileServer/bin/Debug$ ls
Derp.jpg      FileServer.exe.config  Herp.jpg      LIB.dll.mdb
FileServer.exe  FileServer.exe.mdb    LIB.dll
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileServer/bin/Debug$ ./FileServer.exe
Starting server...
Waiting for client...
Client connected - waiting for filename.
Requested file: Herp.jpg of length: 9099112
Sending file
Sent 9100 of 9100 packets to client
Restarting server...

Starting server...
Waiting for client...
Client connected - waiting for filename.
Requested file: Derp.jpg of length: 20464
Sending file
Sent 21 of 21 packets to client
Restarting server...

Starting server...
Waiting for client...
^C
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileServer/bin/Debug$
```

Figure 1: Test af TCP server/client - billede fra server.

```
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileClient/bin/Debug$ ls
FileClient.exe  FileClient.exe.config  FileClient.exe.mdb  LIB.dll  LIB.dll.mdb
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileClient/bin/Debug$ ./FileClient.exe 10.0.0.1 Herp.jpg
Starting client...
Size of file: 9099112
File received.
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileClient/bin/Debug$ ls
FileClient.exe  FileClient.exe.config  FileClient.exe.mdb  Herp.jpg  LIB.dll  LIB.dll.mdb
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileClient/bin/Debug$ ./FileClient.exe 10.0.0.1 Derp.jpg
Starting client...
Size of file: 20464
File received.
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileClient/bin/Debug$ ls
Derp.jpg      FileClient.exe.config  Herp.jpg      LIB.dll.mdb
FileClient.exe  FileClient.exe.mdb    LIB.dll
ikn@ubuntu:~/git/I4IKN/Øvelse8/FileClient/bin/Debug$
```

Figure 2: Test af TCP server/client - billede fra client.

1.5. Konklusion

I arbejdet med TCP socket programmering er vi kommet frem til en løsning der opfylder kravene givet i opgaven. Det kan derfor konstateres at teorien stemmer overens med praksis.

2. Øvelse 9 - UDP/IP socket programming

2.1. Introduktion

2.2. Udviklingsforløb

2.3. Funktionalitet

2.4. Resultater

2.5. Konklusion

A. Appendix: Sekvens diagrammer

Figure 3: Sekvensdiagram for TCP server/client - uden Send og Receive metoder

