

Journal for øvelserne 8 & 9

I samarbejde med grp: 24

Bjørn Nørgaard Sørensen
stud.nr: 201370248
bjornnorgaard@post.au.dk

Joachim Dam Andersen
stud.nr: 201370031
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	4
2.5	Konklusion	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 .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 1 på side 5.

1.4 Resultater

Vi har testet vores system og vedlagt screenshots heraf. På figur 2 ses test af serveren, hvor der sendes to filer. På dette billede ses filer til afsendelse, som ligger i fileservens Debug folder (markeret med lyserød). Billedet illustrerer desuden programflowet med konsoludskrifter.

På figur 3 ses test af klienten fra samme testsekvens som på figur 2. 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 2: 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 3: 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

I denne øvelse i socket programmering, udarbejdes en UDP client og en UDP server. Igen er C# valgt som programmeringssprog. Clienten forbinder til serveren og downloader en fil herfra. Clienten og server køres på hver sin virtuelle linux maskine. I dette dokument beskrives udviklingsforløbet med tilhørende diagrammer og kodeforklaringer.

2.2 Udviklingsforløb

Der oprettes først en forbindelse mellem server og client. Derefter sender clienten en request til serveren, hvorefter denne checker om der findes en matchende kommande til clientens request. Såfremt serveren finder en matchende kommando, aflæses den rette server status, som derefter sendes tilbage til clienten.

Code 3: Hoveddesign for server

```
1 private static void StartListener()  
2 {  
3     //Initiate UDP server  
4  
5     while(true)  
6     {  
7         //Wait for incoming command  
8         //Send matching response  
9     }  
10 }
```

Code 4: Hoveddesign for client

```
1 static void Main(string[] args)  
2 {  
3     //Setup UDP client  
4     //Initialize ip and command from args  
5     //Send command  
6     //Wait for answer and output to console  
7 }
```

2.3 Funktionalitet

Herunder beskrives virkemåde for koden som er skrevet ud fra pseudokoden i vist i 3. Yderligere kan et sekvensdiagram for UDP koden ses på figur 4 på side 6.

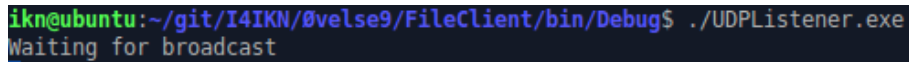
1. Først initialiseres en UDP server og en socket.
2. Serveren venter derefter på at modtage en kommando fra en klient. Der skrives i konsollen hvilken kommando som er modtaget, samt hvem afsenderen var.
3. Der tjekkes på den modtagne kommando, og sendes et matchende svar. Desuden skrives i konsollen hvilken kommando der bliver sendt, samt indholdet af kommandoen.

Her beskrives virkemåden for koden i codeudsnit 4 for klienten.

1. Først initialiseres der en UDP client og en socket.
2. Der indlæses ip adresse og kommando fra brugerens argumenter.
3. Kommandoen sendes til serveren.
4. Der ventes på at serveren svarer, og det modtagne svar udskrives i konsollen.

2.4 Resultater

Vi har testet vores system og vedlagt screenshots heraf. På figur 5 ses test af serveren. Figur 6 viser klientens side af udvekslingen. Billederne illustrerer desuden programflowet med konsoludskrifter.



```
ikn@ubuntu:~/git/I4IKN/Øvelse9/FileClient/bin/Debug$ ./UDPListener.exe
Waiting for broadcast
```

Figure 5: Test af UDP server/client - billede fra server.



```
ikn@ubuntu:~/git/I4IKN/Øvelse9/FileServer/bin/Debug$ ./UDPServer.exe 10.0.0.2 l
Message sent to the broadcast address
Received broadcast from 10.0.0.2:44787 :
l

ikn@ubuntu:~/git/I4IKN/Øvelse9/FileServer/bin/Debug$ ./UDPServer.exe 10.0.0.2 u
Message sent to the broadcast address
Received broadcast from 10.0.0.2:40036 :
u
```

Figure 6: Test af UDP server/client - billede fra client.

2.5 Konklusion

I arbejdet med UDP 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.

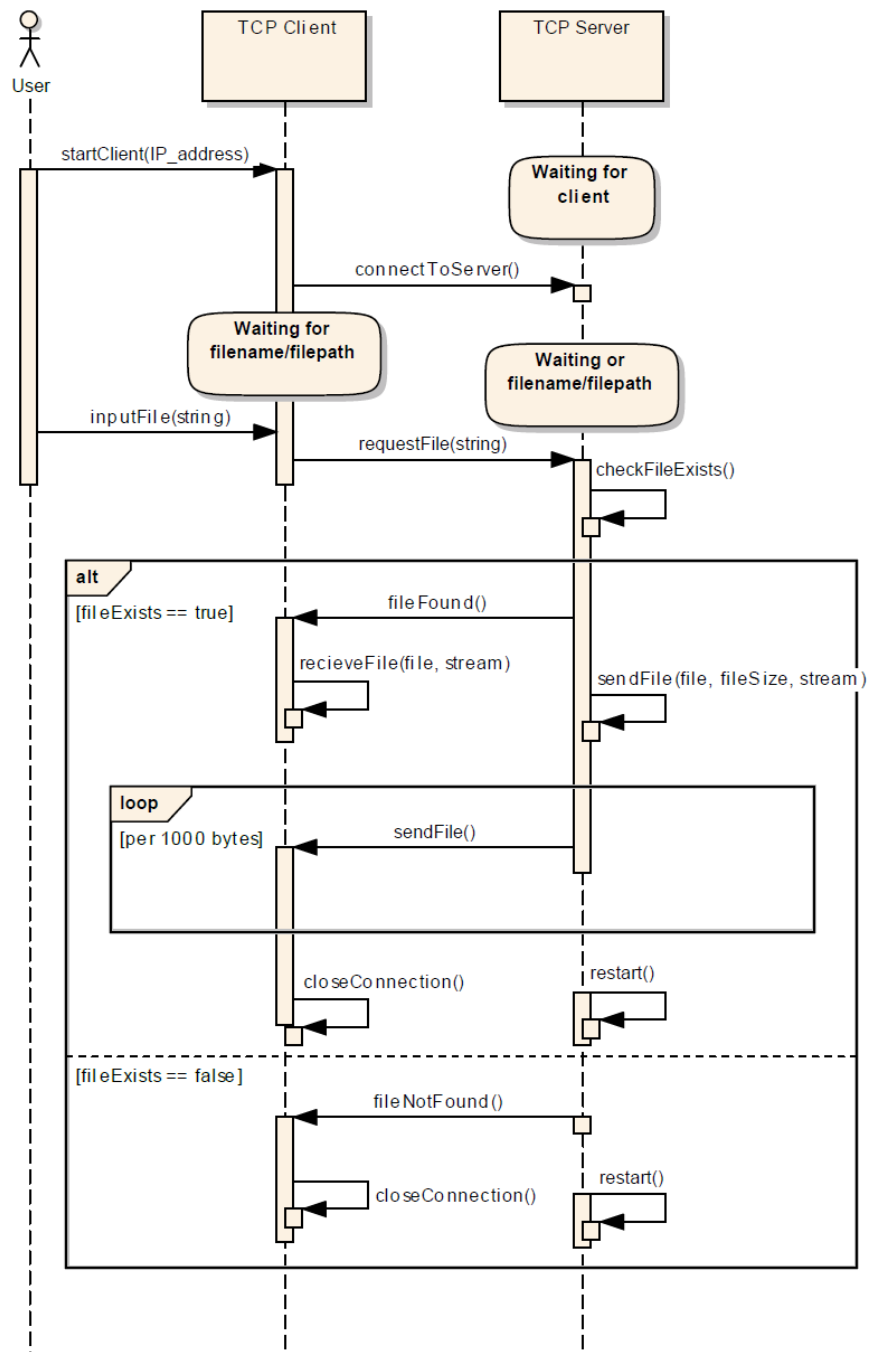


Figure 1: Sekvensdiagram for TCP server/client - uden Send og Receive metoder

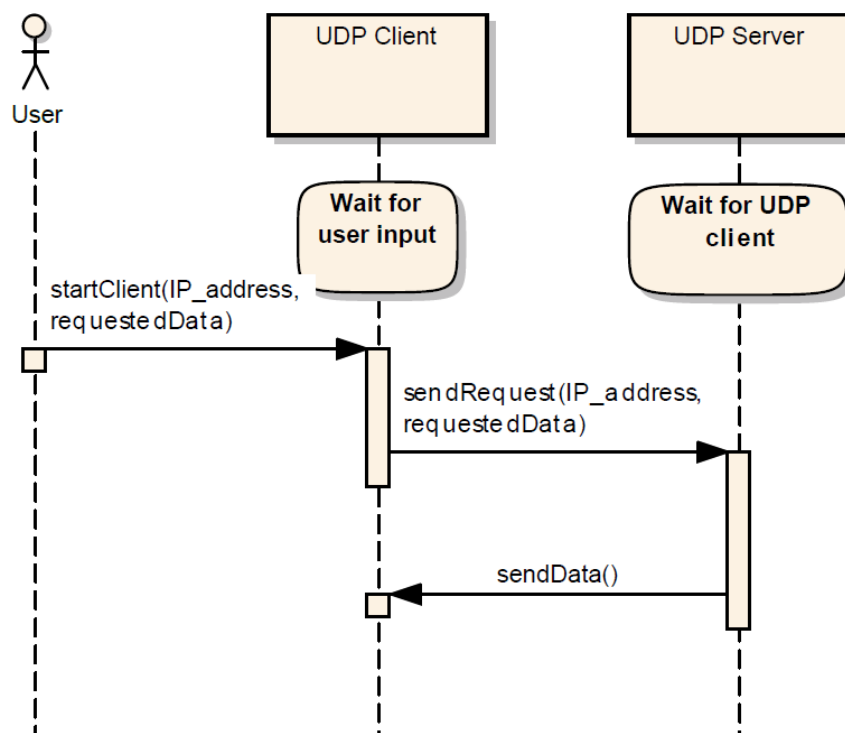


Figure 4: Sekvensdiagram for UDP server/client.