I4IKN - Øvelse 7

Application Layer, Transport Layer, TCP/IP Socket Programming

|  |  |
| --- | --- |
| Jonas Lind | 201507296 |
| Tais Hjortshøj | 201509128 |

# Problemformulering

I denne øvelse skal der udvikles en iterativ server og en client som skal køre i hver deres virtuel Linux-maskine. Serveren skal kunne modtage en tekststreng fra én client ad gangen. Tekststrengen som clienten sender skal indeholde filnavn og eventuelt stiangivelse og herved udpege en fil af en vilkårlig type/størrelse i serveren, som den tilsluttede client ønsker at hente fra serveren. Filen skal overføres fra server til client i segmenter på 1000 bytes ad gangen indtil filen er overført fuldstændigt. Der returneres en fejlmeddeles fra serveren til clienten hvis den ønskede fil ikke findes i serveren. Efter filoverførslen er færdig skal serveren kunne håndtere en ny forespørgsel fra en client.

Som kvalitetskontrol for client/server systemet skal den overførte fil kunne sammenlignes med den oprindelige fil. Der må ikke være nogle forskelle i filerne mht. til størrelse eller mht. indhold.

# Server

|  |
| --- |
| PORT **=** 9000  HOST **=** ''  BUFSIZE **=** 1000  **def** main**(**argv**):**  **try:**  #Initiate server  serversocket **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_STREAM**)**  #Bind socket to PORT and localhost  serversocket**.**bind**((**HOST**,**PORT**))**  serversocket**.**listen**(**1**)** #We want to queue up to just one client  **print** 'Socket connection on: '**,** serversocket**.**getsockname**()**  **except** socket**.**error **as** msg**:**  **print** 'Error connecting with serversocket: %s\n Terminating program.'  **%**msg  serversocket**.**close**()**  sys**.**exit**()** |

Snippet – Oprettelse af serversocket.

Snippet 1 viser koden for initiering af vores socket på serveren. Vi udskriver til terminalen hvilken adresse serveren er forbundet til. SOCK\_STREAM gør at det er en TCP forbindelse. Vi håndterer fejl ved at udskrive en fejlmeddelse, lukke socketten og forlade programmet.

|  |
| --- |
| **while(**1**):**  #Wait for connections  **(**clientsocket**,** address**)** **=** serversocket**.**accept**()**  **print** 'Incomming connection from'**,** address  filename **=** Lib**.**readTextTCP**(**clientsocket**)** #connect with client  **print** '1 '**,** filename    filesize **=** Lib**.**check\_File\_Exists**(**filename**)** #returns size  **print** '2 filesize '**,** filesize    **if** filesize **!=** 0**:**  sendFile**(**filename**,** filesize**,** clientsocket**)**  **else:**  Lib**.**writeTextTCP**(**"0"**,** clientsocket**)**  Lib**.**writeTextTCP**(**"File " **+** filename **+** "doesnt exist."**,** clientsocket**)**  **print** '3 Closing connection '**,** address  clientsocket**.**close**()** |

Snippet – Iterativ server der ikke lukker ned efter end filoverførsel til client.

Snippet 2 viser vores while løkke som håndtere én connection fra en client for derefter at starte forfra. Dette gør vores server iterativ. Vi bruger funktioner fra det udleverede Lib, som er blevet lavet som et object bibliotek og derfor skal vi kalde objektet Lib.funktion for at gøre brug af disse.

|  |
| --- |
| **def** sendFile**(**fileName**,** fileSize**,** conn**):**  i **=** 0  text **=** "."  #Sending size of requested file  Lib**.**writeTextTCP**(**str**(**fileSize**),** conn**)**  **with** open**(**fileName**,** "rb"**)** **as** file\_obj**:**  **while** 1**:** #text not == "":  text **=** file\_obj**.**read**(**BUFSIZE**)**  i **=** i **+** 1  **if** text **==** ""**:**  conn**.**send**(**text**)**  **break**  conn**.**send**(**text**)**  **print** 'Packets sent: '**,** i  **print** 'File sent: '**,** fileName |

Snippet – Funktion til at sende fil

Snippet 3 viser vores sendFile funktion som vi bruger til at sende den fil der er blevet anmodet om. Den skal bruge filnavn, filstørrelse og en socket at sende på. Vi starter med at sende filstørrelsen som tekststreng og derefter sender vi den anmodede fil i bider på 1000 bytes(BUFSIZE). Den udskriver i terminalen hvor mange packets der er sent og at filen er sent til sidst.

# Client

Vi har udviklet en klient som opfylder at den skal have to argumenter når den kaldes – en filsti og hvor denne skal hentes.

|  |
| --- |
| PORT **=** 9000  HOST **=** ''  BUFSIZE **=** 1000  **def** main**(**argv**):**  #Fil som skal hentes  fileName **=** argv**[**0**]** #Filsti som argument  **print** '1 Filename: '**,** fileName  **if** argv**[**1**]:**  HOST **=** argv**[**1**]**  **print** HOST  #Oprettelse af socket  clientsocket **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_STREAM**)**  clientsocket**.**connect**((**HOST**,** PORT**))**  **print** '2 ' |

Snippet 4 – Oprettelse af clientsocket.

Snippet 4 viser initiering af client socket. SOCK\_STREAM gør at det er TCP.

|  |
| --- |
| #Anmod server om fil gennem socket  Lib**.**writeTextTCP**(**fileName**,** clientsocket**)**  **print** '3 '  #Vent paa at modtage fil  receiveFile**(**fileName**,** clientsocket**)**  **print** '5 File received. Closing connection.\n '  clientsocket**.**close**()** |

Snippet 5 – Client anmoder server om fil gennem socket.

Snipper 5 viser at vi initiere TCP forbindelsen ved at anmode og en fil. Derefter kalder vi funktionen receiveFile(). Når vi har modtaget filen lukkes socketten og programmet lukkes.

|  |
| --- |
| **def** receiveFile**(**fileName**,** conn**):**  size **=** Lib**.**readTextTCP**(**conn**)** # Save recieved message  fileName **=** Lib**.**extractFilename**(**fileName**)** # Remove path  **print** '4 Data size: '**,** long**(**size**)**  **if** long**(**size**)** **==** 0**:**  **print** 'File'**,**fileName**,** 'doesn\'t exist. Closing connection.'  sys**.**exit**()**  text\_obj **=** open**(**fileName**,** "w"**)** # Make new file  i **=** 0  **while** i **<** long**(**size**):**  text **=** conn**.**recv**(**BUFSIZE**)**  text\_obj**.**write**(**text**)** # Write message to file  i **=** i **+** len**(**text**)**  **print** 'Text received: '**,** i**,** long**(**size**)**  **if** i **>=** size**:**  **break**  **print** 'Text obj: '**,** text\_obj**,**  text\_obj**.**close**()** # Save file  filesize **=** Lib**.**check\_File\_Exists**(**fileName**)**  **print** '\nSize of file: '**,** filesize |

Snippet 6 – Modtage fil fra server

Snippet 6 viser vores funktion til at modtage filen. Vi starter med at modtage størrelsen på filen som en tekststreng. Der tages hensyn til om filen ikke eksisterer da filstørrelsen så vil være 0. Derefter opretter vi en fil og skriver alle tegn vi modtager ind i denne indtil den korrekte filstørrelse er opnået. Derefter gemmes filen ved at lukke denne og vi tjekker størrelsen på den nye fil.

# Test

|  |  |
| --- | --- |
| Figur 1 – Succesfuld anmodning og modtagelse af fil i clientens terminal. Der ses at vi kan håndtere hvis filen ikke eksisterer og der vil ikke blive oprettet en tom fil. | Figur 2 – Succesfuld modtagelse af anmodning og afsendelse af fil. Serveren fungere iterativt og kan håndtere at filen ikke eksisterer. Vi kan også se at vores TCP forbindelse ligger på port 9000, og at der oprettes nye porte til at sende på. (57576 og 58070). |
| Figur - Vi downloader den originale fil fra nettet og sammenligner den med den vi har hentet gennem vores TCP. Der er ingen forskel. | |

# Konklusion

Vores TCP server virker fint. Den kan håndtere at mange fejl, deriblandt at filen ikke eksistere. De filer vi modtager er identiske til de oprindelige hvilket betyder at det virker som det skal. Vi kan sende alle slags filer da vi splitter filen op inden den sendes. Vi har her valgt at inkludere testen med at sende et billede samt fejl i stien som derved henviser til en ikke-eksisterende fil.

Det virker mellem to forskellige virtuelle maskiner på samme PC.

Den fulde kode er vedlagt i bilag.