I4IKN - Øvelse 8

UDP/IP Socket Programming

## UDP server

Vi vil udvikle en iterativ server som kan modtage en kommando fra en client. Kommandoerne som genkendes er ”U”, ”u”, ”L” og ”l”. Vi håndterer at der modtages andre kommandoer.

Snippet 7 viser initiering af vores socket. SOCK\_DGRAM gør den til en UDP forbindelse. Vi sætter socket options til at være genbrugelig for at den ikke er lukket i en periode efter vores program crasher. Dette er primært brugbart under tests med kodeændring. Vi fejlhåndterer og udskriver fejlmeddelelse hvis dette sker.

|  |
| --- |
| **def** main**(**argv**):**  **try:**  s **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_DGRAM**)**  s**.**setsockopt**(**socket**.**SOL\_SOCKET**,** socket**.**SO\_REUSEADDR**,** 1**)**  s**.**bind**((**HOST**,** PORT**))**      **except** socket**.**error **as** msg**:**  **print** 'Error 1 encountered: '**,** msg  s**.**close**()**  sys**.**exit**()** |

Snippet 7 – Initiering af socket.

Snippet 8 viser vores while-loop som gør vores server iterativ. Der bliver også her fejlhåndteret. Den kan primært 3 ting. Enten anmodes om en af 2 filer eller også kan den svare med ukendt anmodning.

|  |
| --- |
| **try:**  **while(**1**):**  **print** '\nServer ready to receive data.'  data**,** address **=** s**.**recvfrom**(**BUFSIZE**)**  **print** 'Message from: '**,** address  **print** 'Message received: '**,** data    **if** data **==** "L" **or** data **==** "l"**:**  sendFile**(**"/proc/loadavg"**,** address**,** s**)**  **elif** data **==** "U" **or** data **==** "u"**:**  sendFile**(**"/proc/uptime"**,** address**,** s**)**  **else:** **print** 'Command not recognized: '**,** data  socket**.**sendto**(**"Command not recognized"**,** address**)**    **except** socket**.**error **as** msg**:**  **print** 'Error 2 encountered: '**,** msg  s**.**close**()**  sys**.**exit**()** |

Snippet 8 – Iterativ server der ikke lukker ned efter end filoverførsel til client.

|  |
| --- |
| **def** sendFile**(**file\_**,** address**,** socket**):**  text **=** openFile**(**file\_**)**  socket**.**sendto**(**text**,** address**)**  **print** 'Returning '**,** file\_      **def** openFile**(**file\_**):**  **with** open**(**file\_**,** "rb"**)** **as** file\_obj**:**  text **=** file\_obj**.**read**()**  **return** text |

Snippet 9 – Vi ser her vores to funktioner til at åbne en fil og sende en fil.

## Test

Figur 1 – Vi kan modtage 4 kommandoer og der returneres det korrekte. Serveren er iterativ.

## UDP Client

Vi vil udvikle en client som sender ved hjælp af UDP. Den modtager to argumenter – en kommando som skal sendes og hvor denne skal sendes til.

|  |
| --- |
| **def** main**(**argv**):**  #Create UDP socket  sock **=** socket**.**socket**(**socket**.**AF\_INET**,** socket**.**SOCK\_DGRAM**)**  sock**.**bind**((**UDP\_HOST**,** UDP\_PORT**))**  cmd **=** argv**[**0**]**  addr **=** argv**[**1**]**    #Send data  sock**.**sendto**(**cmd**,** **(**addr**,** UDP\_PORT**))**    #Recieve response  data**,** addr **=** sock**.**recvfrom**(**BUFFER\_SIZE**)**  **print** 'Recieved: '**,** data  #Closing Socket  **print** 'Closing socket'  sock**.**close**()** |

Snippet 10 – Koden for vores UDP client. Vi initiere en socket. Der modtages to argumenter som behandles. Data modtaget udskrives i terminalen.

## Test

Figur 3 – Vi ser at alle 4 kommandoer returnerer hvad de skal. Programmet lukker derefter ned og er klar til at køres igen.

## Konklusion

Vi har nu lavet både en UDP server og en TCP server som vi har fået til at virke. Vi har prøvet at håndtere fejl der kan opstå.

Fuld kode er vedhæftet i bilag.