

# 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

```
root@ubuntu:~/Desktop/IKN/Ovelse_8# python server_udp.py  
  
Server ready to receive data.  
Message from: ('10.0.0.2', 9000)  
Message received: l  
Returning /proc/loadavg  
  
Server ready to receive data.  
Message from: ('10.0.0.2', 9000)  
Message received: L  
Returning /proc/loadavg  
  
Server ready to receive data.  
Message from: ('10.0.0.2', 9000)  
Message received: U  
Returning /proc/uptime  
  
Server ready to receive data.  
Message from: ('10.0.0.2', 9000)  
Message received: u  
Returning /proc/uptime  
  
Server ready to receive data.
```

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

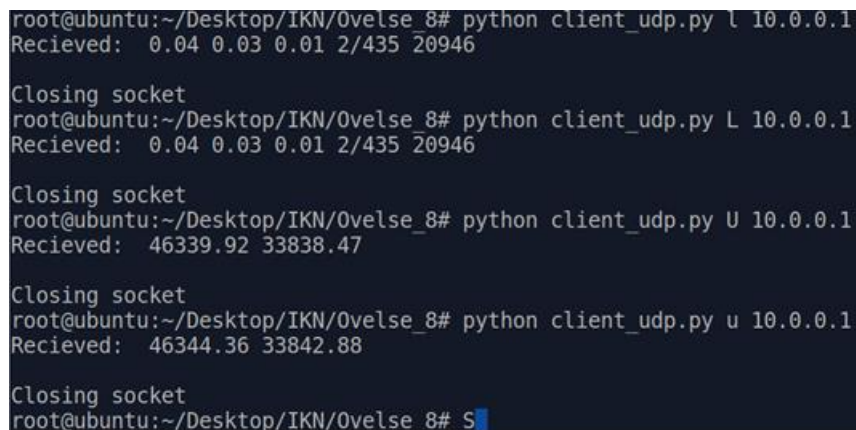
## UDP Client

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



```
root@ubuntu:~/Desktop/IKN/Ovelse_8# python client_udp.py l 10.0.0.1  
Recieved: 0.04 0.03 0.01 2/435 20946  
  
Closing socket  
root@ubuntu:~/Desktop/IKN/Ovelse_8# python client_udp.py L 10.0.0.1  
Recieved: 0.04 0.03 0.01 2/435 20946  
  
Closing socket  
root@ubuntu:~/Desktop/IKN/Ovelse_8# python client_udp.py U 10.0.0.1  
Recieved: 46339.92 33838.47  
  
Closing socket  
root@ubuntu:~/Desktop/IKN/Ovelse_8# python client_udp.py u 10.0.0.1  
Recieved: 46344.36 33842.88  
  
Closing socket  
root@ubuntu:~/Desktop/IKN/Ovelse_8# S
```

Figur 2 – 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.