

- LAB 1: Web Server

I learned the basics of socket programming for TCP connections in Python.

Web server create a connection socket when contacted by a client .

web server accept and parse the HTTP request and get the requested file from the server's file system.

It create an HTTP response message.

It send the response over the TCP connection to the requesting browser.

If the requested file is not present in the server, the server send an HTTP "404 Not Found" message back to the client.

#### CODE:

Server tcp port : 5000

It created server socket and bind it to address and tcp port with using bind() function

```
# Create a TCP server socket
#(AF_INET is used for IPv4 protocols)
#(SOCK_STREAM is used for TCP)
serverPort = 5000
serverSocket = socket(AF_INET, SOCK_STREAM)
#Prepare a sever socket
#Fill in start
serverSocket.bind(('', serverPort))
serverSocket.listen(1)
print ('port:', serverPort)
```

The connection is created using the accept() function.

```
while True:
    #Establish the connection
    print ('Ready to serve...')
    #Fill in start
    connectionSocket, addr =serverSocket.accept() #Fill in end
    try:
        message = connectionSocket.recv(1024) #Fill in start #Fill in end
        filename = message.split()[1]
        f = open(filename[1:])
        outputdata =f.read() #Fill in start #Fill in end
        print (outputdata)

        #Send one HTTP header line into socket
        #Fill in start#
        connectionSocket.send('\nHTTP/1.1 200 OK\n\n'.encode())
        #Fill in end
```

An HTTP header line is sent to the socket with using send() function. If the file is not found, a response message is sent to the client with send() function.

Client socket is closed with using close() function.

```
# Send the content of the requested file to the connection socket
for i in range(0, len(outputdata)):
    connectionSocket.send(outputdata[i].encode())
connectionSocket.send("\r\n".encode())
connectionSocket.close()

except IOError:
    #Send response message for file not found
    #Fill in start
    mes = "HTTP/1.1 404 Not Found"
    connectionSocket.send(mes.encode())
    mes = "<html><body><h1>404 Not Found!</h1></body></html>\r\n"
    connectionSocket.send(mes.encode())
    #Fill in end
    #Close client socket
    #Fill in start
    connectionSocket.close()
    #Fill in end
```

#### OUTPUT:

I created an html file.



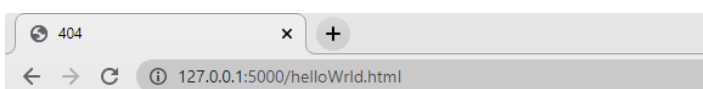
helloWorld.html

The IP address of the host running the server has been determined. Opened a browser from another host and entered the corresponding URL.



LAB 1 WEB SERVER  
hello world  
Tuba TOPRAK

Non-existent file tried



**404 Not Found!**

- Lab 2: UDP Pinger Lab

Client send a simple 10 ping message to a server, and it receive corresponding message back from the server.

Then, The Round Trip Time is calculated. If no reply is received ,The client assume that the packet was lost and print a message

Code:

Added necessary libraries.

Create UDP socket on localhost. Timeout value is set.

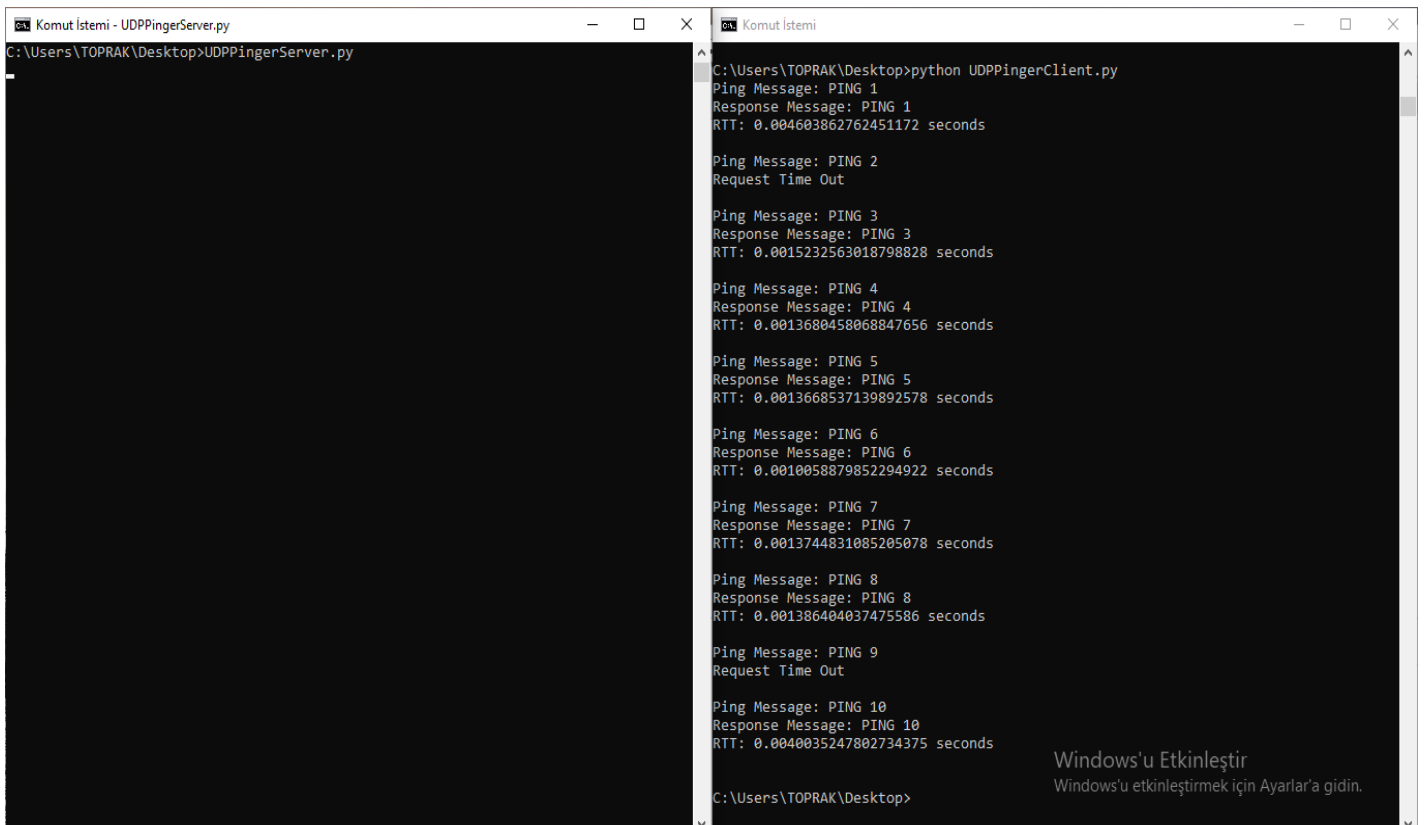
The client waits up to one second for a response from the server.

```
from socket import *
from time import *
clientSocket = socket(AF_INET, SOCK_DGRAM)
serverAddr = ('localhost', 12000)
clientSocket.settimeout(1)
ping = 0
```

A loop was used for the client to send 10 ping messages to the server. The counter is initialized using the time() function to calculate the RTT. The message is encrypted using the encode () function. The ping message is sent to the server using the sendto () function, The client gets the response message from socket with recvfrom() function. Then ,Message is printed on the screen. Rtt is calculated and printed on the screen. If there is no response from the server, it gives an error message.

```
while (ping < 10):
    try:
        start = time()
        message = "PING " + str(ping+1)
        clientSocket.sendto(message.encode(), serverAddr)
        print("Ping Message: " + message)
        data, server = clientSocket.recvfrom(1024)
        print("Response Message: " + data.decode())
        end = time()
        elapsed = end - start
        print("RTT: "+ str(elapsed) + " seconds\n")
    except timeout:
        print("Request Time Out\n")
    ping += 1
clientSocket.close()
```

Output:



The image shows two terminal windows. The left window, titled 'Komut İstemi - UDPPingerServer.py', shows the execution of the server script. The right window, titled 'Komut İstemi', shows the execution of the client script. The client script sends 10 ping messages. The first, third, fourth, fifth, sixth, seventh, eighth, and tenth messages receive responses with RTT values. The second and ninth messages result in 'Request Time Out'.

```
Komut İstemi - UDPPingerServer.py
C:\Users\TOPRAK\Desktop>UDPPingerServer.py

Komut İstemi
C:\Users\TOPRAK\Desktop>python UDPPingerClient.py
Ping Message: PING 1
Response Message: PING 1
RTT: 0.004603862762451172 seconds

Ping Message: PING 2
Request Time Out

Ping Message: PING 3
Response Message: PING 3
RTT: 0.0015232563018798828 seconds

Ping Message: PING 4
Response Message: PING 4
RTT: 0.0013680458068847656 seconds

Ping Message: PING 5
Response Message: PING 5
RTT: 0.0013668537139892578 seconds

Ping Message: PING 6
Response Message: PING 6
RTT: 0.0010058879852294922 seconds

Ping Message: PING 7
Response Message: PING 7
RTT: 0.0013744831085205078 seconds

Ping Message: PING 8
Response Message: PING 8
RTT: 0.001386404037475586 seconds

Ping Message: PING 9
Request Time Out

Ping Message: PING 10
Response Message: PING 10
RTT: 0.0040035247802734375 seconds

C:\Users\TOPRAK\Desktop>
```

- Lab 3: SMTP Lab

Mail port is 50.I created socket.

Client socket recieves certain amount of data. Prints error if message is not received.

```
from socket import *
msg = "\r\n I love computer networks!"
endmsg = "\r\n. \r\n"
mailServer = "localhost"
mailPort = 50
#Fill in start
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((mailServer, mailPort))
#Fill in end
recv = clientSocket.recv(1024)
print (recv)
if recv[:3] != "220":
    print ("220 reply not received from server.")
heloCommand = "HELO Alice\r\n"
clientSocket.send(heloCommand)
recv1 = clientSocket.recv(1024)
print (recv1)
if recv1[:3] != "250":
    print ("250 reply not received from server.")
```

Sender address is entered. Sent RCPT TO command and print server response.

```
clientSocket.send("RCPT TO: <tbtoprakk@gmail.com> \r\n")
recv1 = clientSocket.recv(1024)
print (recv1)
if recv1[:3] != "250":
    print ("250 reply not received from server.")

clientSocket.send("DATA\r\n")
recv1 = clientSocket.recv(1024)
print (recv1)
if recv1[:3] != "354":
    print ("250 reply not received from server.")
clientSocket.send("\r\n")
clientSocket.send("deneme\r\n")

clientSocket.send(".\r\n")
recv1 = clientSocket.recv(1024)
print (recv1)
print (recv1)
if recv1[:3] != "250":
    print ("250 reply not received from server.")
```

Sent QUIT command and get server response.T tells the server that the message should be terminated.

```
clientSocket.send("QUIT\r\n")
clientSocket.cl
```