CSE 476 TERM PROJECT – FINAL REPORT

Fatih Selim YAKAR - 161044054

PROBLEM DEFINITION:

For Part1 (Web Server)

I need to write a python web server that can process 1 request within the 1st part of the project. This server should simply meet the following expectations in order:

1) When the server receives a request from a client, it will create a socket.
2) This will receive HTTP request using the opened socket.
3) To find the requested file, it must reserve the incoming request.
4) It must open the requested file from the server's location. If it cannot find the file, it should give a "404 not found" error.
5) It should generate an HTTP response containing this file.
6) It must send the generated HTTP response to the client over the TCP connection.

For Part2 (UDP pinger)

Within the 2nd part of the project, a client will be written and this client will send 10 consecutive ping messages to a server via UDP. It will keep and print the time from each ping message to the pong message that comes as a response. In line with the delays or packet losses caused by the nature of UDP, the response pongs that exceed 1 second timeout will print "request timed out" message.

For Part3 (Mail Client)

Within the 3.part of the project, a mail client will be written and this client will connect to a mail server and send a message to another e-mail using the SMTP protocol via TCP. After the message is sent, the connection will be closed.

IMPLEMENTATION AND RUNNING:

For Part1 (Web Server)

Source Code Part:

First of all, I wrote a very simple html code to provide response to a client named “sample.html”. Then I opened a TCP type socket in the server code. I connected this socket to the required port and then put it in listening mode. Then, while true, it always accepted over the socket opened in a loop. If the data received as a result of Accept matches an html file, it parses this information and returns that file to the client. If it doesn't match, 404 not found error printed.

```
1   <html>
2
3       <head>
4
5           <title>Python HTTP Server</title>
6
7       </head>
8
9       <body>
10
11          <h1>Simple HTTP Server</h1>
12
13          <p>The HTTP Server is working! Hello World!</p>
14
15      </body>
16
17  </html>
```

Second, I filled in the blanks with the required code fragments.

```
#import socket module
from socket import *

PORT = 8080
serverSocket = socket(AF_INET, SOCK_STREAM)
#Prepare a server socket

#Fill in start
serverSocket.bind(('', PORT))
serverSocket.listen(1)
#Fill in end

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

        #Fill in start
        connectionSocket.send('HTTP/1.0 200 OK\r\n\r\n')
        #Fill in end

        #Send the content of the requested file to the client
        for i in range(0, len(outputdata)):
            connectionSocket.send(outputdata[i])
        connectionSocket.close()

    except IOError:
        #Send response message for file not found

        #Fill in start
        print "404 Not Found"
        connectionSocket.send('HTTP/1.1 404 Not Found\r\n\n')
        #Fill in end

        #Close client socket

        #Fill in start
        connectionSocket.close()
        #Fill in end
serverSocket.close()
```

Running and Hardcoded Parameters:

There is only one hardcoded parameter in Part1 and this parameter is the PORT parameter. The port number is determined according to this parameter and the server is switched on accordingly.

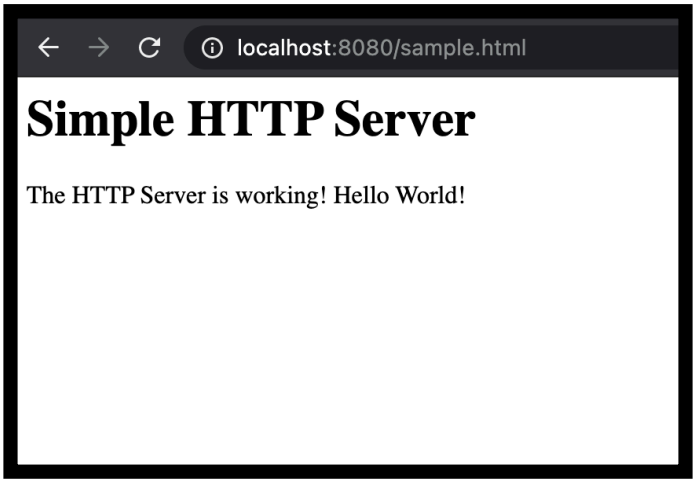
```
#import socket module
from socket import *

PORT = 8080
serverSocket = socket(AF_INET, SOCK_STREAM)
#Prepare a server socket
```

The command to run is as follows: “python2 WebServer.py” or “python WebServer.py”

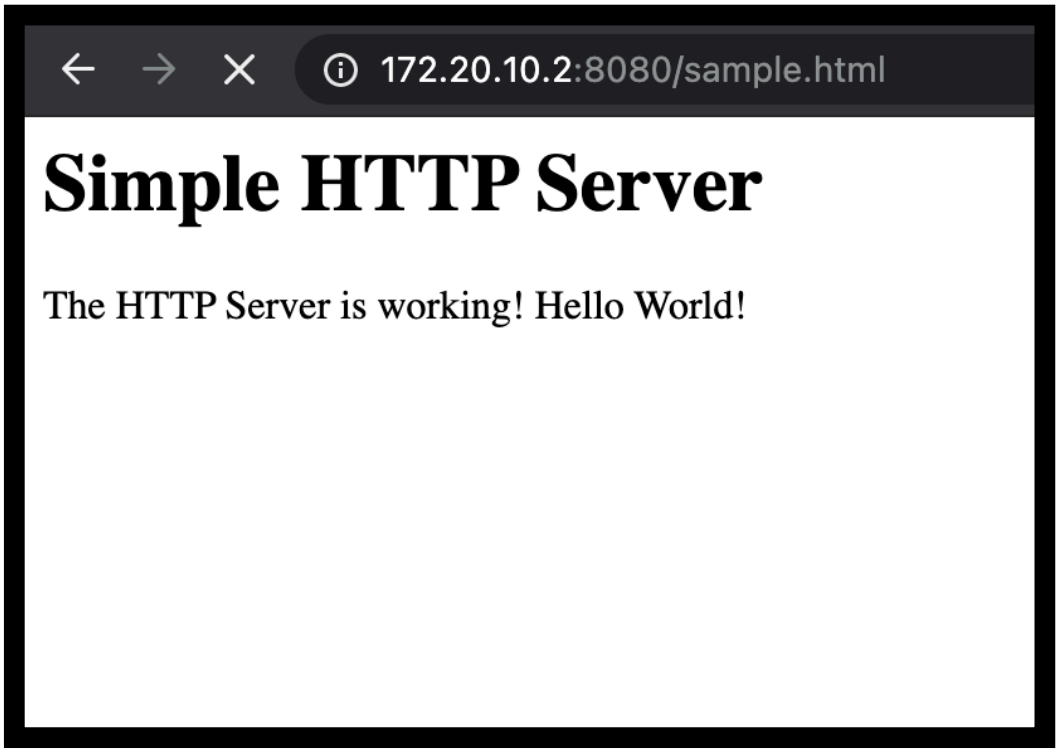
Server Part Running:

```
fatihselimyakar@Fatihs-MacBook-Pro PART1 % python WebServer.py
Ready to serve...
<html>

   <head>

       <title>Python HTTP Server</title>

   </head>

   <body>

       <h1>Simple HTTP Server</h1>

       <p>The HTTP Server is working! Hello World!</p>

   </body>

</html>
Ready to serve...
Ready to serve...
```

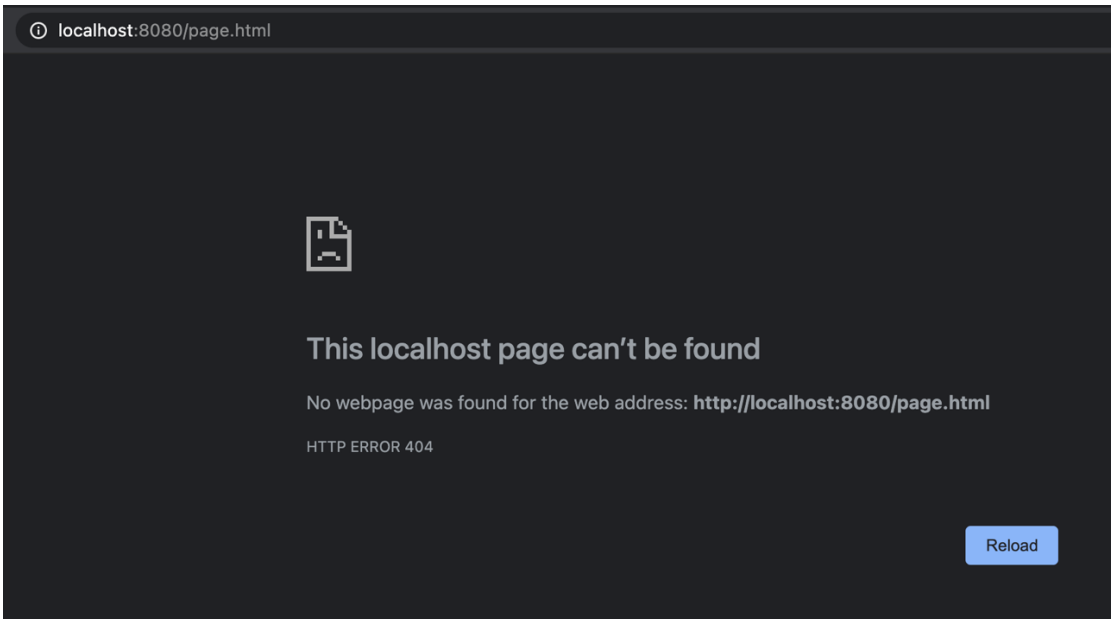
Client Part Running (Same Machine):



Client Part (Different Machine):



Client Part (404 Not Found):



For Part2 (UDP Pinger)

Source Code Part:

First of all, the given server code was implemented. Then, in each loop of a loop that returns 10 times for the client, firstly the UDP connection is opened. A 1-second timeout was set in this connection, after which the ping message was sent and the timer started. If there was an answer within 1 second, the time it came was printed, if not, timed out was printed.

```
import time
import socket

UDP_IP = "127.0.0.1"
UDP_PORT = 12000

# Creates the UDP(SOCK_DGRAM) Socket with IPv4(AF_INET) type
clientSocket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
# Sets the time out for the created socket
clientSocket.settimeout(1.0)

for i in range(1,11):

    # Gets the start time of the progress
    start = time.time()
    # Send the UDP messages, no binding because connectionless,
    message= "Ping "+ str(i)+" "+str(start)
    clientSocket.sendto(message , (UDP_IP, UDP_PORT))
    # Prints sent message's index and current time
    print "Sent Message(Ping)     :( Data:",message,")"

    try:
        # Gets the received message
        data, server = clientSocket.recvfrom(1024)
        # Gets the end time of the progress
        end = time.time()
        # Find the difference
        passingTime = end - start
        # Prints the received message data and passing time
        print "Received Message(Pong) :( Data:",data,"Sequence Number:",i,"Round Trip Time(RTT):",passingTime,")"

    except socket.timeout:
        print "Request Timed Out"
```

Running and Hardcoded Parameters:

There are three different parameters as hardcoded in client. These parameters are UDP_IP and UDP_PORT parameters. UDP parameters are used to UDP connection's socket information.

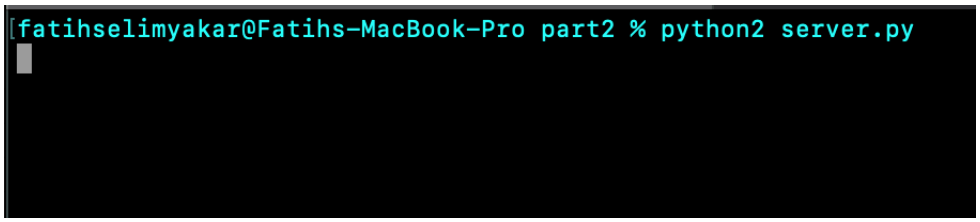
```
UDP_IP = "127.0.0.1"
UDP_PORT = 12000
```

The command to run is as follows: "python client.py", "python server.py" or "python2 client.py" ,"python2 server.py"

Client Part Running:

```
[fatihselimyakar@Fatihs-MBP PART2 % python client.py
Sent Message(Ping)     :( Data: Ping 1 1609680565.83 )
Received Message(Pong) :( Data: PING 1 1609680565.83 Sequence Number: 1 Round Trip Time(RTT): 0.000195026397705 )
Sent Message(Ping)     :( Data: Ping 2 1609680565.83 )
Received Message(Pong) :( Data: PING 2 1609680565.83 Sequence Number: 2 Round Trip Time(RTT): 0.000114917755127 )
Sent Message(Ping)     :( Data: Ping 3 1609680565.83 )
Request Timed Out
Sent Message(Ping)     :( Data: Ping 4 1609680566.83 )
Received Message(Pong) :( Data: PING 4 1609680566.83 Sequence Number: 4 Round Trip Time(RTT): 0.000396013259888 )
Sent Message(Ping)     :( Data: Ping 5 1609680566.83 )
Received Message(Pong) :( Data: PING 5 1609680566.83 Sequence Number: 5 Round Trip Time(RTT): 0.000239133834839 )
Sent Message(Ping)     :( Data: Ping 6 1609680566.83 )
Request Timed Out
Sent Message(Ping)     :( Data: Ping 7 1609680567.84 )
Request Timed Out
Sent Message(Ping)     :( Data: Ping 8 1609680568.84 )
Received Message(Pong) :( Data: PING 8 1609680568.84 Sequence Number: 8 Round Trip Time(RTT): 0.000392913818359 )
Sent Message(Ping)     :( Data: Ping 9 1609680568.84 )
Received Message(Pong) :( Data: PING 9 1609680568.84 Sequence Number: 9 Round Trip Time(RTT): 0.000190019607544 )
Sent Message(Ping)     :( Data: Ping 10 1609680568.84 )
Request Timed Out
```

Server Part Running:



For Part3 (Mail Client)

Source Code Part:

First of all, I defined the strings required to forward the mail as global. then I chose google's mail server as mailserver. Then I created a TCP connection sock that is clientSocket. After these, data such as HeloCommand, starttls (a kind of security protocol), auth login, Mail from, Rcpt to, Data, Subject were sent over the socket respectively. With this sent data, the mail was sent after the configurations were established and authentication was provided.

```python
import ssl
import base64
from socket import *
import sys

subject = "SMTP mail client testing"
msg = "\r\n I love Computer Networks"
endmsg = "\r\n.\r\n"
# You must fill the mail and password in your mail infos.
fromMail = "fatihselimyakar@gmail.com"
rcptMail = "fatihselim.yakar2016@gtu.edu.tr"
password ="22071998"

mailserver = ("smtp.gmail.com", 587) #Fill in start #Fill in end

# Create socket called clientSocket and establish a TCP connection with mailserver
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(mailserver)
recv = clientSocket.recv(1024)
print recv
if recv[:3] != '220':
    print '220 reply not received from server.'

# Send HELO command then print server response.
heloCommand = 'HELO Alice\r\n'
clientSocket.send(heloCommand.encode())
recv1 = clientSocket.recv(1024)
print recv1
if recv1[:3] != '250':
    print '250 reply not received from server.'

# Send starttls command then print server response.
clientSocket.send(('starttls\r\n').encode())
recv2=clientSocket.recv(1024)
print recv2
if recv2[:3] != '220':
    print '220 reply not received from server.'


#Wrap socket and send info for username and password then print server response
clientSocketWrapped = ssl.wrap_socket(clientSocket, ssl_version=ssl.PROTOCOL_SSLv23)

clientSocketWrapped.send(('auth login\r\n').encode())
print(clientSocketWrapped.recv(1024).decode())

clientSocketWrapped.send((base64.b64encode(fromMail.encode()))+('\r\n').encode())
print(clientSocketWrapped.recv(1024).decode())

clientSocketWrapped.send((base64.b64encode(password.encode()))+('\r\n').encode())
print(clientSocketWrapped.recv(1024).decode())
```

```
52
53  # Send MAIL FROM command then print server response.
54  mailFrom = "MAIL FROM: <"+fromMail+"> \r\n"
55  clientSocketWrapped.send(mailFrom.encode())
56  recv3 = clientSocketWrapped.recv(1024)
57  print recv3
58  if recv3[:3] != '250':
59      print '250 reply not received from server.'
60
61  # Send RCPT TO command then print server response.
62  rcptTo = "RCPT TO: <"+rcptMail+"> \r\n"
63  clientSocketWrapped.send(rcptTo.encode())
64  recv4 = clientSocketWrapped.recv(1024)
65  print recv4
66  if recv4[:3] != '250':
67      print '250 reply not received from server.'
68
69  # Send DATA command then print server response.
70  data = "DATA\r\n"
71  clientSocketWrapped.send(data.encode())
72  recv5 = clientSocketWrapped.recv(1024)
73  print recv5
74  if recv5[:3] != '354':
75      print '354 reply not received from server.'
76
77  # Send message data then print server response.
78  clientSocketWrapped.send(("Subject: "+subject+" \r\n\r\n").encode())
79  clientSocketWrapped.send(("From: "+fromMail + '\r\n').encode())
80  clientSocketWrapped.send(("To: "+rcptMail + '\r\n').encode())
81  clientSocketWrapped.send(msg.encode())
82  clientSocketWrapped.send(endmsg.encode())
83  recv_msg = clientSocketWrapped.recv(1024)
84  print recv_msg.decode()
85  if recv_msg[:3] != '250':
86      print '250 reply not received from server.'
87
88  # Send QUIT command and get server response then print.
89  clientSocketWrapped.send("QUIT\r\n".encode())
90  message=clientSocketWrapped.recv(1024)
91  print message
92  if message[:3] != '221':
93      print '221 reply not received from server.'
94  clientSocketWrapped.close()
```

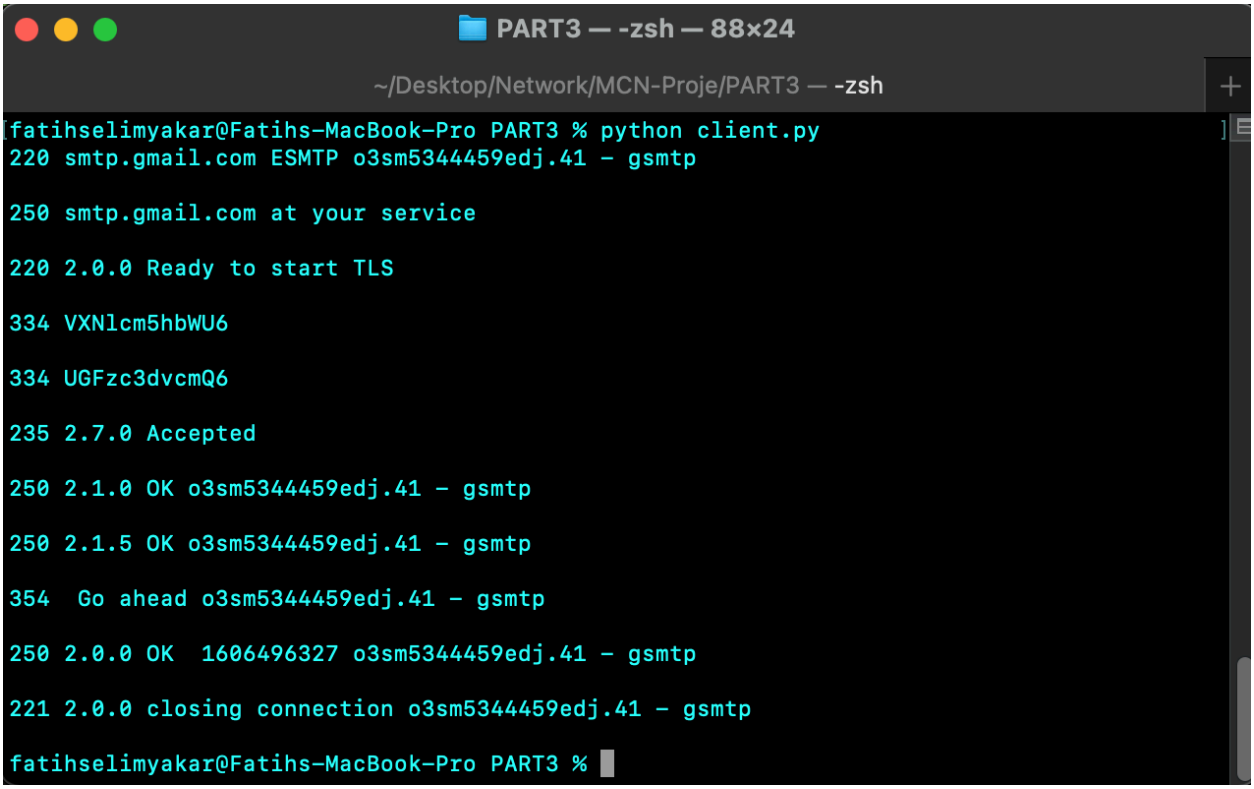
Running and Hardcoded Parameters:

There is only one hardcoded parameter in Part3 and this parameters are the subject,msg,endmsg,fromMail,rcptMail,password parameter. The e-mail messages are determined according to these parameter. Also the data required for the authentication and e-mail address in the second part must be filled in. (OPTIONAL-fromMail must be Google e-mail address and password must filled correctly)

```
# You must type the message in these parameters
subject = "SMTP mail client testing"
msg = "\r\n I love Computer Networks"
endmsg = "\r\n.\r\n"
# You must fill the mail and password in your mail infos.
fromMail = "fatihselimyakar@gmail.com"
rcptMail = "fatihselim.yakar2016@gtu.edu.tr"
password ="**********"
```

The command to run is as follows: “python client.py” or “python2 client.py”

Client part:



Server part: