Term Project Final Report

Elif Akgün
1801042251

1. Web Server

1.1 Assignment Definition

In this assignment, a simple Web server is developed in Python that is capable of processing only one request. Specifically, the Web server creates a connection socket when contacted by a client (browser), receives the HTTP request from this connection, parse the request to determine the specific file being requested, get the requested file from the server's file system, create an HTTP response message consisting of the requested file preceded by header lines, and send the response over the TCP connection to the requesting browser. If a browser requests a file that is not present in the server, the server returns a "404 Not Found" error message.

1.2 Program Code

```
# Import socket module
from socket import *

serverSocket = socket(AF_INET, SOCK_STREAM) #TCP server socket

# Prepare a server socket
serverPort = 2525
serverSocket.bind(('', serverPort))
serverSocket.listen(1) #listen 1 connction

while True:
  #Establish the connection
  print ("Ready to serve...")

  # Set up a new connection from the client
  connectionSocket, addr = serverSocket.accept()

  try:
    # Receives the request message from the client
    message =  connectionSocket.recv(1500)
    filename = message.split()[1] #path of request
    f = open(filename[1:], 'rb')
    outputdata = f.read()
```

```
24      #print(outputdata)
25
26      #Send one HTTP header line into socket
27      connectionSocket.send(str("HTTP/1.1 200 OK\r\n\r\n").encode())
28
29      #Send the content of the requested file to the client
30      for i in range(0, len(outputdata)):
31          connectionSocket.send(chr(outputdata[i]).encode())
32
33      connectionSocket.close()
34
35    except IOError:
36      #Send response message for file not found
37      connectionSocket.send(str("HTTP/1.1 404 Not Found\r\n\r\n").encode()
        )
38      connectionSocket.send(str("<html><head></head><body><h1>404 Not
        Found</h1></body></html>\r\n").encode())
39
40      #Close client socket
41      connectionSocket.close()
42
43  serverSocket.close()
```

1.3 Tests

Put an HTML file (e.g., HelloWorld.html) in the same directory that the server is in. Run the server program. Determine the IP address of the host that is running the server (e.g., 128.238.251.26). From another host, open a browser and provide the corresponding URL. For example: http://128.238.251.26:6789/HelloWorld.html 'HelloWorld.html' is the name of the file you placed in the server directory. The browser should then display the contents of HelloWorld.html. If the file is not present at the server, you should get a "404 Not Found" message.

Figure 1: Server runs with HelloWorld.html that is in the same directory

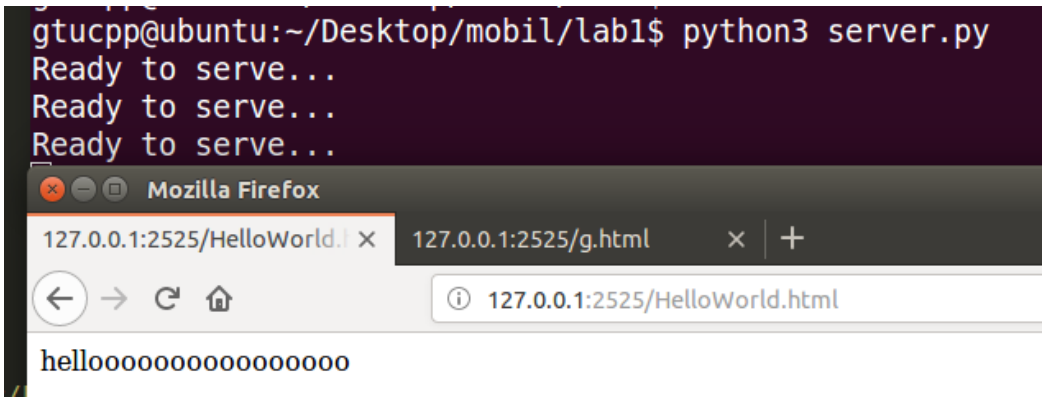
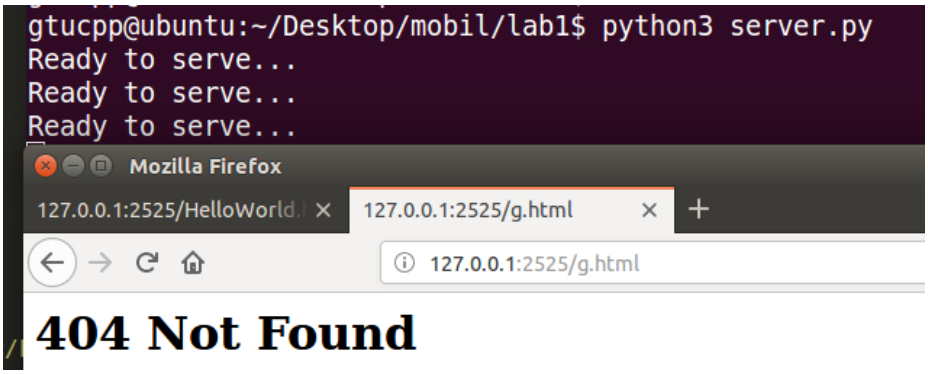


Figure 2: Server runs with g.html that is not present



2. UDP Pinger

2.1 Assignment Definition

In this assignment, the client sends 10 ping messages to the target server over UDP. For each message, determine and print the delay between when the client sent the ping message and received the pong message. This delay is called the Round Trip Time (RTT). Because UDP is an unreliable protocol, a packet sent by the client or server may be lost. For this reason, the client cannot wait indefinitely for a reply to a ping message. The client waits up to one second for a reply from the server; if no reply is received, it assumes that the packet was lost and print a message accordingly.

2.2 Program Code

We have the complete code for the server, so I write only client code. I took IP address and port number as command line argument. Then I create socket. In a for loop, I sent Ping message 10 times. I calculate RTT and if response does not come in one second, it means packet was lost.

```
# UDPPingerServer.py
# We will need the following module to generate randomized lost packets
import random
from socket import *

# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets
serverSocket = socket(AF_INET, SOCK_DGRAM)

# Assign IP address and port number to socket
serverSocket.bind(('', 2525))

while True:
  # Generate random number in the range of 0 to 10
  rand = random.randint(0, 10)
```

```
16
17   # Receive the client packet along with the address it is coming from
18   message, address = serverSocket.recvfrom(1024)
19
20   # Capitalize the message from the client
21   message = message.upper()
22
23   # If rand is less is than 4, we consider the packet lost and do not
       respond
24   if rand < 4:
25     continue
26   # Otherwise, the server responds
27   serverSocket.sendto(message, address)
```

```
1 #UDPPingerClient.py
2 from socket import *
3 from time import time, ctime
4 import sys
5
6 #there sohuld be 3 command line arguments
7 if (len(sys.argv) != 3):
8     print("Usage: UDPPingClient.py <server_host> <server_portNum>")
9     sys.exit()
10
11 ip_addr = sys.argv[1]
12 portNum = sys.argv[2]
13
14 #create client socket
15 clientSocket = socket(AF_INET, SOCK_DGRAM)
16 #wait one second for reply
17 clientSocket.settimeout(1)
18
19 #send and receive 10 ping messages
20 for i in range(0, 10):
21     t0 = time() #start time
22     message = "Ping " + str(i+1) + " " + ctime(t0)[11:19]
23
24     try:
25         #sends and receives the message
26         clientSocket.sendto(message.encode(),(ip_addr, int(portNum)))
27         response, server_addr = clientSocket.recvfrom(1024)
28
29         t1 = time() #end time
30
31         #response message from server
32         print("Response: " + response.decode())
33
34         print("Round Trip Time: %.3fs \n" % (t1 - t0))
35     except:
36         print("Ping " + str(i+1) + " request timed out\n")
37 clientSocket.close()
```

2.3 Tests

1)Run the UDPPingerServer.py
2)Run the UDPPingerClient.py as "UDPPingClient.py $< server\_host >< server\_portNum >$"

Figure 3: Sample Output-1

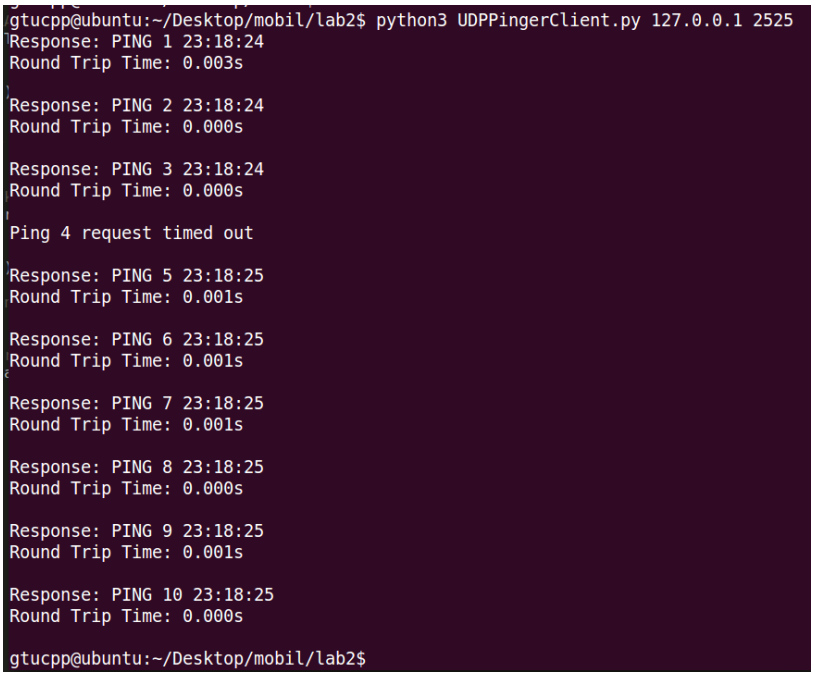
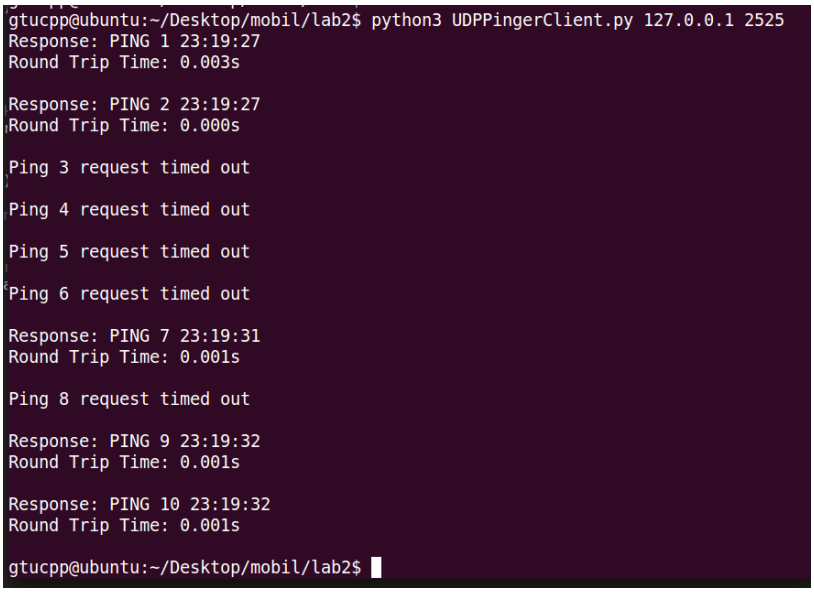


Figure 4: Sample Output-2



3. Mail Client

3.1 Assignment Definition

In this assignment, the task is to develop a simple mail client that sends email to any recipient. The client needs to connect to a mail server, dialogue with the mail server using the SMTP protocol, and send an email message to the mail server. Python provides a module, called smtplib, which has built in methods to send mail using SMTP protocol. However, this module is not used in this assignment, because it hides the details of SMTP and socket programming.

3.2 Program Code

First I set mail server and port. I chose Google mail server and port 587. Other port numbers may occurs error. We should send some commands to send mail. I send following commands respectively.
-HELO command
-STARTTLS command for secure connection
-AUTH LOGIN command
-Mail address and password
-MAIL FROM command with mail address
-RCPT TO command with target mail address
-DATA command
-Subject and message
-QUIT command to quit connection

```
from socket import *
import ssl
import base64

mailAddress = "mail@gmail.com"
password = "password"
targetMail = "targetmail@mail.com"
subject = "Enter mail's subject"
message = "Enter message"

#set mail server
mailServer = ("smtp.gmail.com", 587)
#create socket for TCP
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(mailServer)
#get response and print
response = clientSocket.recv(1024).decode()
print("response: ", response)
if(response[:3] != "220"):
    print("220 reply not received from server.")

```

```
#send HELO command
command = "HELO Alice\r\n"
clientSocket.send(command.encode())
#get response and print
response1 = clientSocket.recv(1024).decode()
print("response1: ", response1)
if(response1[:3] != "250"):
    print("250 reply not received from server.")

#send STARTTLS command
command = "STARTTLS\r\n"
clientSocket.send(command.encode())
#get response and print
response2 = clientSocket.recv(1024).decode()
print("response2: ", response2)
if(response2[:3] != "220"):
    print("220 reply not received from server.")

#wrap socket for security
tlsSocket = ssl.wrap_socket(clientSocket)

#send AUTH LOGIN command
command = "AUTH LOGIN\r\n"
tlsSocket.send(command.encode())
#get response and print
response3 = tlsSocket.recv(1024).decode()
print("response3: ", response3)
if(response3[:3] != "334"):
    print("334 reply not received from server.")

#send mail address
tlsSocket.send(base64.b64encode(mailAddress.encode()))
tlsSocket.send(("\r\n").encode())
#get response and print
response4 = tlsSocket.recv(1024).decode()
print("response4: ", response4)
if(response4[:3] != "334"):
    print("334 reply not received from server.")

#send password
tlsSocket.send(base64.b64encode(password.encode()))
tlsSocket.send(("\r\n").encode())
#get response and print
response5 = tlsSocket.recv(1024).decode()
print("response5: ", response5)
if(response5[:3] != "235"):
    print("235 reply not received from server.")

#send MAIL FROM command
command = "MAIL FROM:<" + mailAddress + ">\r\n"
tlsSocket.send(command.encode())
```

```
#get response and print
response6 = tlsSocket.recv(1024).decode()
print("response6: ", response6)
if(response6[:3] != "250"):
    print("250 reply not received from server.")

#send RCPT TO command
command = "RCPT TO:<" + targetMail + ">\r\n"
tlsSocket.send(command.encode())
#get response and print
response7 = tlsSocket.recv(1024).decode()
print("response7: ", response7)
if(response7[:3] != "250"):
    print("250 reply not received from server.")

#send DATA command
command = "DATA\r\n"
tlsSocket.send(command.encode())
#get response and print
response8 = tlsSocket.recv(1024).decode()
print("response8: ", response8)
if(response8[:3] != "354"):
    print("354 reply not received from server.")

#send subject and message
subject = "Subject: "+ subject + "\r\n\r\n"
message = "\r\n" + message + "\r\n.\r\n"
tlsSocket.send(subject.encode())
tlsSocket.send(message.encode())
#get response and print
response9 = tlsSocket.recv(1024).decode()
print("response9: ", response9)
if(response9[:3] != "250"):
    print("250 reply not received from server.")

#send QUIT command
command = "QUIT\r\n"
tlsSocket.send(command.encode())
#get response and print
response10 = tlsSocket.recv(1024).decode()
print("response10: ", response10)
if(response10[:3] != "221"):
    print("221 reply not received from server.")
```

3.3 Test

Figure 5: Sample Output

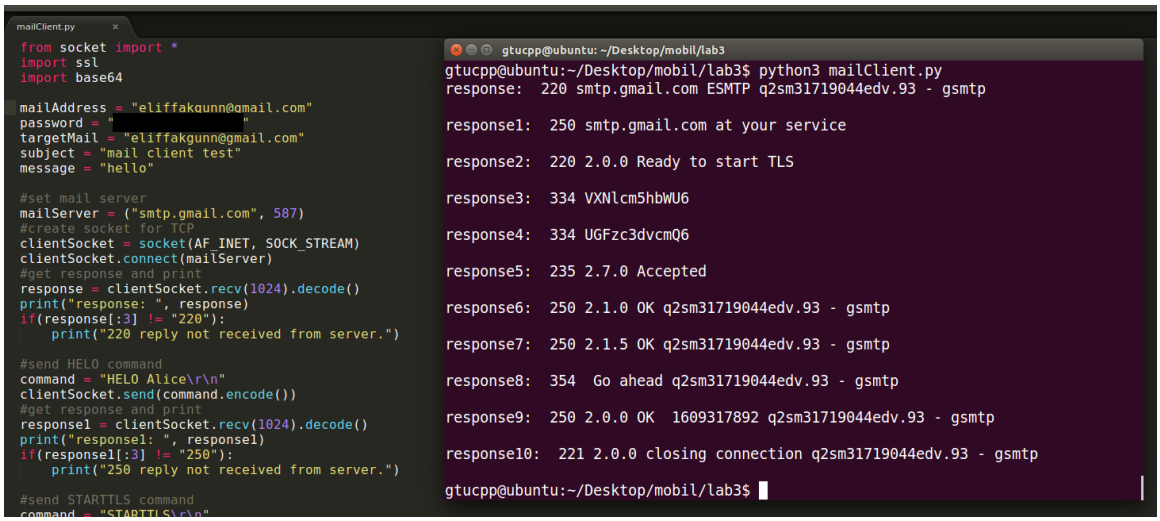


Figure 6: Inbox