CSE 476/575 – Mobile Communication Networks

Term Project Report

Name: Rahmet Ali Olmez

Number: 1801042623

1. Results

1.1. Web Server

I have tested the project by running the server in a different machine, connected to the same network. While writing and testing the code, I have discovered that the browser was automatically requesting for an icon file named favicon.ico, so I put an icon file to the directory of the server.

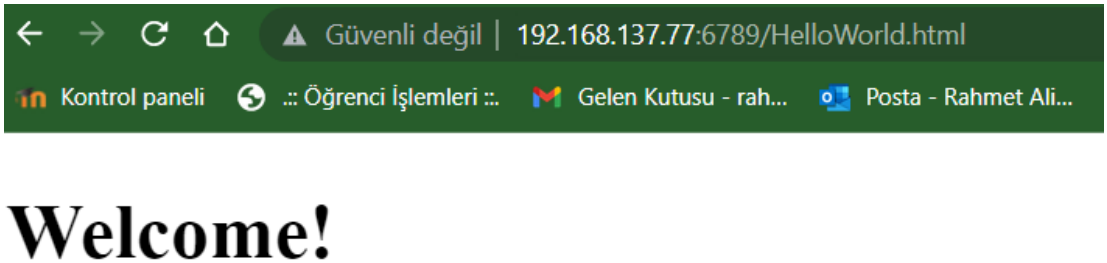

FIG – 1 The client receives the requested file.

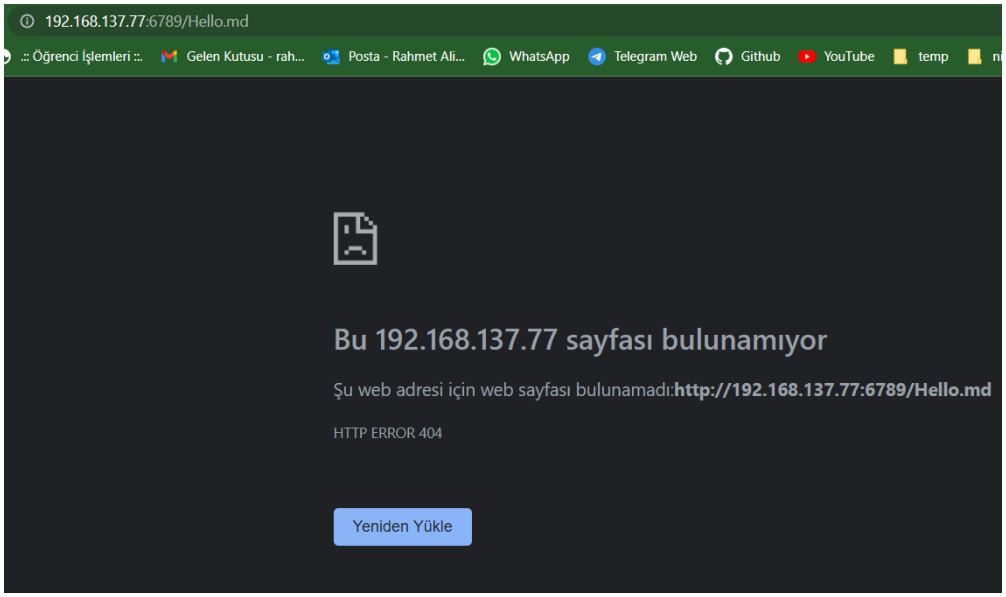

FIG – 2 The client receives error code 404.

1.2. UDP Pinger

This part of the project was also tested on a different machine connected to the same network.

```
C:\Users\Rahmet\Desktop\mobile_communication_networks\term_project\p2>python udp_pinger_client.py
PING 1 16:13:52
PACKET 1 RTT: 0.082909 SECONDS
REQUEST TIMED OUT. PACKET  2 LOST
REQUEST TIMED OUT. PACKET  3 LOST
PING 4 16:13:54
PACKET 4 RTT: 0.1127035 SECONDS
REQUEST TIMED OUT. PACKET  5 LOST
PING 6 16:13:55
PACKET 6 RTT: 0.2196514 SECONDS
REQUEST TIMED OUT. PACKET  7 LOST
PING 8 16:13:56
PACKET 8 RTT: 0.2215319 SECONDS
REQUEST TIMED OUT. PACKET  9 LOST
REQUEST TIMED OUT. PACKET  10 LOST
```

FIG – 3 UDP Pinger Test 1

```
C:\Users\Rahmet\Desktop\mobile_communication_networks\term_project\p2>python udp_pinger_client.py
REQUEST TIMED OUT. PACKET  1 LOST
REQUEST TIMED OUT. PACKET  2 LOST
PING 3 16:15:28
PACKET 3 RTT: 0.0182983 SECONDS
REQUEST TIMED OUT. PACKET  4 LOST
REQUEST TIMED OUT. PACKET  5 LOST
PING 6 16:15:30
PACKET 6 RTT: 0.1343888 SECONDS
REQUEST TIMED OUT. PACKET  7 LOST
PING 8 16:15:31
PACKET 8 RTT: 0.0666325 SECONDS
PING 9 16:15:31
PACKET 9 RTT: 0.0020221 SECONDS
PING 10 16:15:31
PACKET 10 RTT: 0.0020627 SECONDS
```

FIG – 4 UDP Pinger Test 2

```
C:\Users\Rahmet\Desktop\mobile_communication_networks\term_project\p2>python udp_pinger_client.py
PING 1 16:16:00
PACKET 1 RTT: 0.2049487 SECONDS
PING 2 16:16:01
PACKET 2 RTT: 0.0023801 SECONDS
REQUEST TIMED OUT. PACKET  3 LOST
REQUEST TIMED OUT. PACKET  4 LOST
PING 5 16:16:03
PACKET 5 RTT: 0.1215622 SECONDS
PING 6 16:16:03
PACKET 6 RTT: 0.0022973 SECONDS
REQUEST TIMED OUT. PACKET  7 LOST
REQUEST TIMED OUT. PACKET  8 LOST
REQUEST TIMED OUT. PACKET  9 LOST
REQUEST TIMED OUT. PACKET  10 LOST
```

FIG – 5 UDP Pinger Test 3

1.3. Mail Client

For this project to work properly, I had to take some extra steps. I tried to send a mail using smtp.gmail.com, but it required SSL or TLS. So I used the python ssl library.

Also, the username and password of my account was also required for authorization. The authorization string is generated using:

```
echo -ne “\0username\0password”|base64
```

```
C:\Users\Rahmet\Desktop\mobile_communication_networks\term_project\p3>python smtp_mail_client.py
b'220 smtp.gmail.com ESMTP b73sm6278285edf.37 - gsmtp\r\n'
220 reply not received from server.
b'250 smtp.gmail.com at your service\r\n'
250 reply not received from server.
AUTH RESPONSE:  b'235 2.7.0 Accepted\r\n'
MAIL FROM RESPONSE:  b'250 2.1.0 OK b73sm6278285edf.37 - gsmtp\r\n'
RCPT TO RESPONSE:  b'250 2.1.5 OK b73sm6278285edf.37 - gsmtp\r\n'
DATA RESPONSE:  b'354  Go ahead b73sm6278285edf.37 - gsmtp\r\n'
PERIOD RESPONSE:  b'250 2.0.0 OK  1640620661 b73sm6278285edf.37 - gsmtp\r\n'
QUIT RESPONSE:  b'221 2.0.0 closing connection b73sm6278285edf.37 - gsmtp\r\n'
```

FIG – 6 SMTP client in action

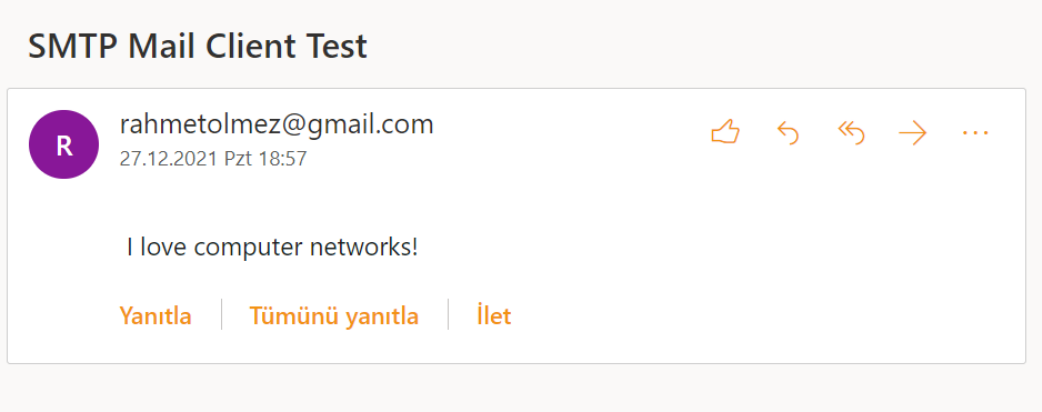

FIG – 7 Mail sent with SMTP client

2. Source Code

2.1. Web Server

```
#import socket module
from socket import *
serverSocket = socket(AF_INET, SOCK_STREAM)

#Prepare a sever socket
#Fill in start
port = 6789
serverSocket.bind(("192.168.137.77", port))
serverSocket.listen(1)
#Fill in end

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

```
                f = open(filename[1:])
                output = f.read()
        #Fill in end

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

                #Send the content of the requested file to the client
                for i in range(0, len(output)):
                        connectionSocket.send(output[i].encode())
                connectionSocket.close()
        except IOError as e:
                #Send response message for file not found
                #Fill in start
                connectionSocket.send('\nHTTP/1.1 404 Not Found\n\n'.encode())
                #Fill in end

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

2.2. UDP Pinger

```
### UDP Pinger Client
### Sends message to server and calculates
### the round trip time of returning message
### Author: Rahmet Ali Olmez
### December 2021

import random
from socket import *
import time
from datetime import datetime

# Set the timeout
timeout = 1
setdefaulttimeout(timeout)

serverSocket = socket(AF_INET, SOCK_DGRAM)

address = ('192.168.137.77', 12000)

for sequence_number in range(1, 11):
      start_time = time.time_ns()
      message = 'Ping ' + str(sequence_number) + ' ' +
str(datetime.now().strftime("%H:%M:%S"))
      message = str.encode(message)
      serverSocket.sendto(message, address)
      # Get message from server
      try:
            message, address = serverSocket.recvfrom(1024)
```

```
            end_time = time.time_ns()
            print(message.decode('utf-8'))
            print('PACKET', sequence_number, 'RTT:', (end_time -
start_time) / 1000000000, 'SECONDS')
      except:
            print('REQUEST TIMED OUT. PACKET ', sequence_number, 'LOST')
```

2.3. Mail Client

```
from socket import *
import ssl # Added
msg = "\r\n I love computer networks!"
endmsg = "\r\n.\r\n"
# Choose a mail server (e.g. Google mail server) and call it mailserver
mailserver = ('smtp.gmail.com', 465) #Fill in end
# Create socket called clientSocket and establish a TCP connection with
mailserver
#Fill in start
with ssl.wrap_socket(socket(AF_INET, SOCK_STREAM)) as clientSocket:
      clientSocket.connect(mailserver)
#Fill in end
      recv = clientSocket.recv(1024)
      print(recv) # Added parentheses
      if recv[:3] != '220':
            print('220 reply not received from server.') # Added
indentation and parens

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

      # Send MAIL FROM command and print server response.
      # Fill in start
      # To generate login input: echo -ne "\0username\0password"|base64
      # The input is edited just for safety
      login_input = 'a098dsufa9G0weDasCjdfX80asdfA'.encode()
      auth_plain_command = 'AUTH PLAIN
'.encode()+login_input+'\r\n'.encode()
      clientSocket.send(auth_plain_command)
      response = clientSocket.recv(1024)
      print('AUTH RESPONSE: ', response)

      mail_from_command = 'MAIL FROM: <rahmetolmez@gmail.com>\r\n'
      clientSocket.send(mail_from_command.encode())
      response = clientSocket.recv(1024)
      print('MAIL FROM RESPONSE: ', response)
      # Fill in end

      # Send RCPT TO command and print server response.
      # Fill in start
      send_rcpt_command = 'RCPT TO: <r.olmez2018@gtu.edu.tr>\r\n'
```

```
      clientSocket.send(str.encode(send_rcpt_command))
      response = clientSocket.recv(1024)
      print('RCPT TO RESPONSE: ', response)
      # Fill in end

      # Send DATA command and print server response.
      # Fill in start
      data_command = 'DATA\r\n'
      clientSocket.send(str.encode(data_command))
      response = clientSocket.recv(1024)
      print('DATA RESPONSE: ', response)
      # Fill in end

      # Send message data.
      # Fill in start
      mail_subject = 'Subject: SMTP Mail Client Test\r\n\r\n'
      clientSocket.send(str.encode(mail_subject))
      #mail_content = 'Hello, Rahmet. This mail is sent using
smtp_mail_client.py!\r\n'
      clientSocket.send(str.encode(msg))
      # Fill in end

      # Message ends with a single period.
      # Fill in start
      clientSocket.send(str.encode(endmsg))
      response = clientSocket.recv(1024)
      print('PERIOD RESPONSE: ', response)
      # Fill in end

      # Send QUIT command and get server response.
      # Fill in start
      period = 'QUIT\r\n'
      clientSocket.send(str.encode(period))
      response = clientSocket.recv(1024)
      print('QUIT RESPONSE: ', response)
      # Fill in end
```