

CSE476-575
MOBILE
COMMUNICATION
NETWROK

Term Project Report

1801042005
Oğuzhan SEZGİN

Assignment 1: Web Server

I create a simple web server with using python. This web server can handle only one HTTP request. Server creates the TCP socket for connect the client(browser). It listens the 8084 port (Figure 1).

```
serverSocket = socket(AF_INET, SOCK_STREAM)
port = 8084
# Prepare a server socket
# Fill in start
serverSocket.bind(('', port))
serverSocket.listen(1)
# Fill in end
```

Figure 1: Socket creation

Then client tries to connect this socket server accepts this connection(Figure 2).

```
while True:
    # Establish the connection
    print("Ready to serve...")
    # Fill in start
    connectionSocket, addr = serverSocket.accept()
    # Fill in end
```

Figure 2: Establish connection

After that client send request to this port, it receives request. Then it parses request, for the determine the file name. After that it finds the file and read. Even if client sends empty message it ignores (Figure 3).

```
try:
    # Fill in start
    message = connectionSocket.recv(2048)
    # Fill in end
    if message == "":
        connectionSocket.close()
        continue

    filename = message.split()[1]
    f = open(filename[1:])
    # Fill in start
    outputdata = f.read()
    f.close()
```

Figure 3: Receive request, extract filename and read file

Then it sends the HTTP header with 200 code and sends the file content (Figure 4).

```
# Fill in start
respHeader = "HTTP/1.1 200 OK\r\n\r\n"
connectionSocket.send(respHeader.encode())
# Fill in end

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

Figure 4: Send success header and file content

If it cannot find the file, then it sends the HTTP header with 404 code and 404 Not Found message to client (Figure 5).

```
except IOError:
    # Send response message for file not found
    # Fill in start
    respHeader = "HTTP/1.1 404 Not Found\r\n\r\n"
    connectionSocket.send(respHeader.encode())

    respMes = "<html><head><title>File Not Found</tit
    connectionSocket.send(respMes.encode())
    # Fill in end
```

Figure 5: File cannot find header and error

Demo

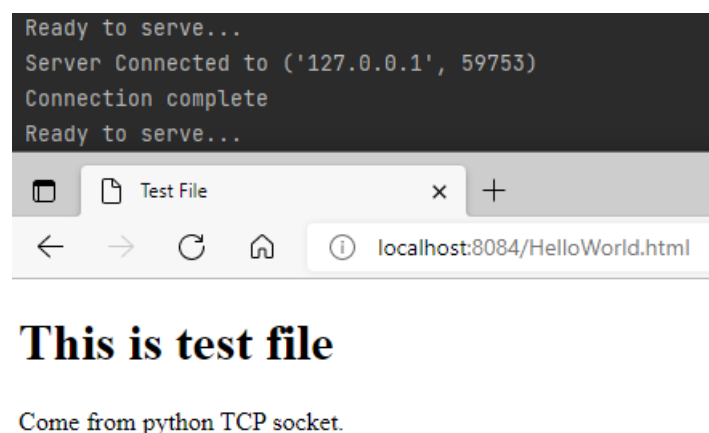
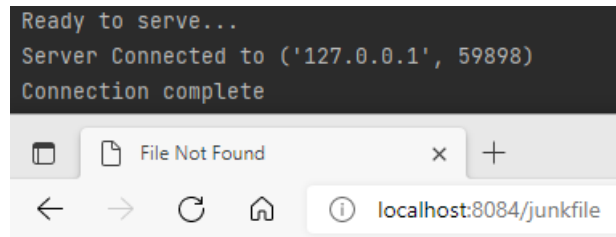


Figure 6: Successful response



404 Not Found!

Figure 7: Not found error

Assignment 2: UDP Pinger

I create simple client in python. It has a simple role; it only sends 10 ping message and waits the response then calculates the RTT times. If response do not come in 1 seconds it assumes response timed out.

First of all, server is started and then Clients creates socket (Figure 7).

```
clientSocket = socket(AF_INET, SOCK_DGRAM)
serverAddr = ('localhost', 12000)
clientSocket.settimeout(1)
```

Figure 7: Clients socket creation

After creation, client takes the current time and put time into the message. Time is in microseconds (Figure 8). Message format is simple. It is “Ping *sequence_number* *time*”. Then it sends message to the server (Figure 9).

```
def current_micro_time():
    return round(time.time() * 1000000)
```

Figure 9: Current time in microsecond

```
try:
    reqTime = current_micro_time()
    message = "Ping " + str(i) + " " + str(reqTime)
    clientSocket.sendto(message.encode(), serverAddr)
    print("Ping Message : " + message)
```

Figure 8: Client sends ping message

Then client waits the server message. If the message comes in 1 seconds it receives message and prints the screen, after print calculates the RTT (Figure 10).

```
data, server = clientSocket.recvfrom(1024)
print("Response Message: " + data.decode())
respTime = current_micro_time()

passTime = respTime - reqTime
print("RTT: {} microseconds".format(passTime))
print("-----")
```

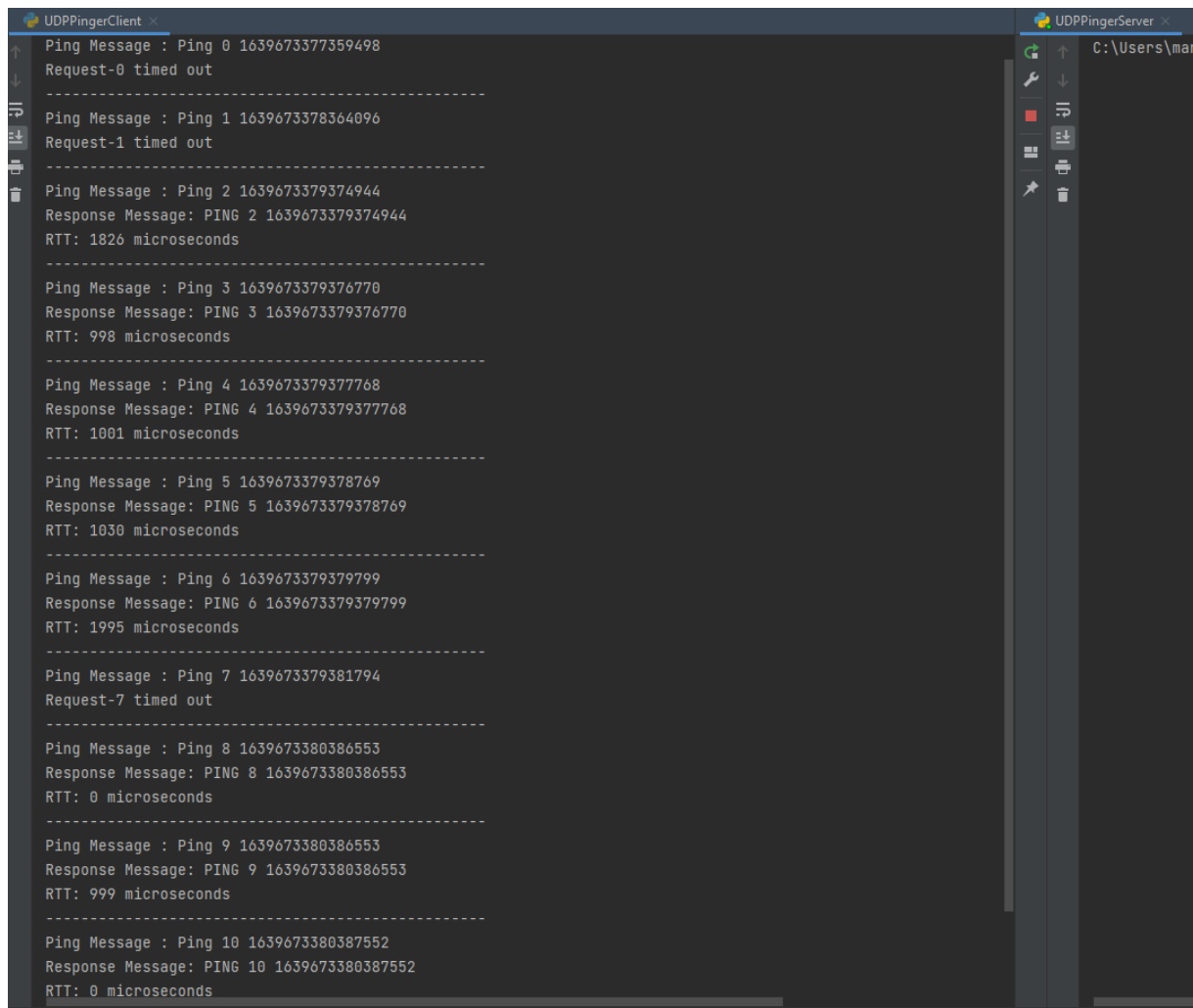
Figure 10: Receive server response and calculate RTT

If the RTT time is greater than 1 seconds then client assume request is timed out (Figure 11).

```
except timeout:
    print("Request-{} timed out".format(i))
    print("-----")
```

Figure 11: Timed out response

Demo



```
UDPPingerClient x
Ping Message : Ping 0 1639673377359498
Request-0 timed out
-----
Ping Message : Ping 1 1639673378364096
Request-1 timed out
-----
Ping Message : Ping 2 1639673379374944
Response Message: PING 2 1639673379374944
RTT: 1826 microseconds
-----
Ping Message : Ping 3 1639673379376770
Response Message: PING 3 1639673379376770
RTT: 998 microseconds
-----
Ping Message : Ping 4 1639673379377768
Response Message: PING 4 1639673379377768
RTT: 1001 microseconds
-----
Ping Message : Ping 5 1639673379378769
Response Message: PING 5 1639673379378769
RTT: 1030 microseconds
-----
Ping Message : Ping 6 1639673379379799
Response Message: PING 6 1639673379379799
RTT: 1995 microseconds
-----
Ping Message : Ping 7 1639673379381794
Request-7 timed out
-----
Ping Message : Ping 8 1639673380386553
Response Message: PING 8 1639673380386553
RTT: 0 microseconds
-----
Ping Message : Ping 9 1639673380386553
Response Message: PING 9 1639673380386553
RTT: 999 microseconds
-----
Ping Message : Ping 10 1639673380387552
Response Message: PING 10 1639673380387552
RTT: 0 microseconds
```

Figure 12: RTT times

Assignment 3: Mail Client

I create a simple mail client that sends email to any receipt. First of all, I define sender and receiver email, sender password and messages (Figure 13).

```
senderMail = "emailfornetwork1@gmail.com"
password = "abc123456+"
receiverMail = "sezginoguzhn@gmail.com"
subject = "SMTP Mail Test"

msg = "\r\n I love computer networks!\r\n(This is test email!!!)"
endmsg = "\r\n.\r\n"
```

Figure 13: Mail information definition

Then I use google mail server with SMTP protocol for send email (Figure 14).

```
# Choose a mail server (e.g. Googlemailserver) and call it mailserver
# Fill in start
port = 587
mailserver = ("smtp.gmail.com", port)
# Fill in end
```

Figure 14: Google mail server

After these definitions, client establish TCP connection with google mail server (Figure 15).

```
# Fill in start
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(mailserver)
# Fill in end

recv = clientSocket.recv(1024)
print(recv)
if recv[:3] != b'220':
    print('220 reply not received from server.')
```

Figure 15: Establish connection

Then it sends **HELO** command to mail server with using TCP socket and prints received message (Figure 16).

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

Figure 16: Send HELO command

After **HELO** command, it sends **TLS** command. This command use for google mail security protocol and prints received message and. It also makes sockets to ssl wrapped socket for the same reason. (Figure 17).

```
TLSCCommand = "STARTTLS\r\n"
clientSocket.send(TLSCCommand.encode())
recv2 = clientSocket.recv(1024)
print(recv2)
if recv2[:3] != b'220':
    print('220 reply not received from server.')
```

Figure 17: TLS command for google mail server

Then it sends credential of sender mail for authenticate the server (Figure 18).

```

clientSocket.send("AUTH LOGIN\r\n".encode())
recv3 = clientSocket.recv(1024)
print(recv3)
if recv3[:3] != b'334':
    print('334 reply not received from server.')

clientSocket.send(b64encode(senderMail.encode()) + "\r\n".encode())
recv4 = clientSocket.recv(1024)
print(recv4)
if recv4[:3] != b'334':
    print('334 reply not received from server.')

clientSocket.send(b64encode(password.encode()) + "\r\n".encode())
recv5 = clientSocket.recv(1024)
print(recv5)
if recv5[:3] != b'235':
    print('235 reply not received from server.')

```

Figure 18: Authenticate google mail server

After that it sends **MAIL FROM** command for the determine sender information (Figure 19).

```

# Send MAIL FROM command and print server response.
# Fill in start
mailFrom = "MAIL FROM: <" + senderMail + ">\r\n"
clientSocket.send(mailFrom.encode())
recv6 = clientSocket.recv(1024)
print(recv6)
if recv6[:3] != b'250':
    print('250 reply not received from server.')
# Fill in end

```

Figure 19: MAIL FROM command

Then it sends **RCPT TO** command for determine receiver information (Figure 20).

```

# Send RCPT TO command and print server response.
# Fill in start
rcptTo = "RCPT TO: <" + receiverMail + ">\r\n"
clientSocket.send(rcptTo.encode())
recv7 = clientSocket.recv(1024)
print(recv7)
if recv7[:3] != b'250':
    print('250 reply not received from server.')
# Fill in end

```

Figure 20: RCPT TO command

After **RCPT TO** command it sends **DATA** command for determine mail content to be sent (Figure 21).


```

# Send DATA command and print server response.
# Fill in start
data = "DATA\r\n"
clientSocket.send(data.encode())
recv8 = clientSocket.recv(1024)
print(recv8)
if recv8[:3] != b'354':
    print('354 reply not received from server.')
# Fill in end

```

Figure 21: DATA command

After determining the mail content, it sends the message data (Figure 22).

```

# Send message data.
# Fill in start
message = "SUBJECT: " + subject + "\r\n" + msg + endmsg
clientSocket.send(message.encode())
recv9 = clientSocket.recv(1024)
print(recv9)
if recv9[:3] != b'250':
    print('250 reply not received from server.')
# Fill in end

```

Figure 22: Send message data

Finally, it sends **QUIT** command for terminate connection with google mail server and then it closes socket at the end (Figure 23).

```

# Send QUIT command and get server response.
Quit = "QUIT\r\n"
clientSocket.send(Quit.encode())
recv10 = clientSocket.recv(1024)
print(recv10)
if recv10[:3] != b'221':
    print('221 reply not received from server.')

clientSocket.close()

```

Figure 23: QUIT Command

Demo

```
C:\Users\mance\PycharmProjects\mobile_project\venv\Scripts\python.exe C:/U
b'220 smtp.gmail.com ESMTP 14sm2108960ejj.156 - gsmtpl\r\n'
b'250 smtp.gmail.com at your service\r\n'
b'220 2.0.0 Ready to start TLS\r\n'
b'334 VXNlcm5hbWU6\r\n'
b'334 UGFzc3dvcmQ6\r\n'
b'235 2.7.0 Accepted\r\n'
b'250 2.1.0 OK 14sm2108960ejj.156 - gsmtpl\r\n'
b'250 2.1.5 OK 14sm2108960ejj.156 - gsmtpl\r\n'
b'354 Go ahead 14sm2108960ejj.156 - gsmtpl\r\n'
b'250 2.0.0 OK 1639687808 14sm2108960ejj.156 - gsmtpl\r\n'
b'221 2.0.0 closing connection 14sm2108960ejj.156 - gsmtpl\r\n'

Process finished with exit code 0
```

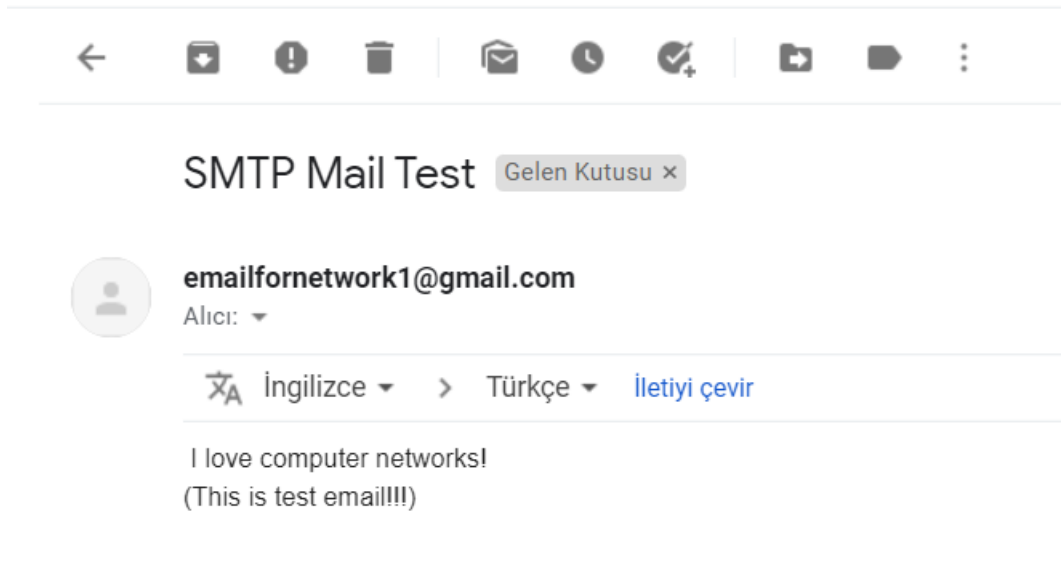


Figure 24: Test-1; sender: emailfornetwork1@gmail.com, receiver: sezginoguzhn@gmail.com

```
C:\Users\mance\PycharmProjects\mobile_project\venv\Scripts\python.exe C:/Users/mance/Py
b'220 smtp.gmail.com ESMTP sh33sm2198992ejc.56 - gsmtpl\r\n'
b'250 smtp.gmail.com at your service\r\n'
b'220 2.0.0 Ready to start TLS\r\n'
b'334 VXNlcm5hbWU6\r\n'
b'334 UGFzc3dvcmQ6\r\n'
b'235 2.7.0 Accepted\r\n'
b'250 2.1.0 OK sh33sm2198992ejc.56 - gsmtpl\r\n'
b'250 2.1.5 OK sh33sm2198992ejc.56 - gsmtpl\r\n'
b'354 Go ahead sh33sm2198992ejc.56 - gsmtpl\r\n'
b'250 2.0.0 OK 1639688117 sh33sm2198992ejc.56 - gsmtpl\r\n'
b'221 2.0.0 closing connection sh33sm2198992ejc.56 - gsmtpl\r\n'

Process finished with exit code 0
```

SMTP Mail Test

- ⓘ Bu ileti gereksiz e-posta olarak tanımlandı. İletiyi 30 gün sonra silenecek. [Gereksiz e-posta değil](#)
- ⓘ Etiket: Junk Email (30 gün) Sona erme zamanı: 15.01.2022 Cmt 23:55



emailfornetwork1@gmail.com

16.12.2021 Per 23:55



I love computer networks!
(This is test email!!!)

[Yanıtla](#)

[Tümünü yanıtla](#)

[İlet](#)

Figure 25: Test-2; sender: emailfornetwork1@gmail.com, receiver: oguzhansezgin@gtu.edu.tr