Gebze Technical University
Computer Engineering

CSE 476 - 2020 Autumn

TERM PROJECT REPORT

LEMYE CEREN GUMUS
151044071

Course Assistant:Abdullah Salih BAYRAKTAR

1 INTRODUCTION

1.1 Problem Definition

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while other socket reaches out to the other to form a connection. Server forms the listener socket while client reaches out to the server.

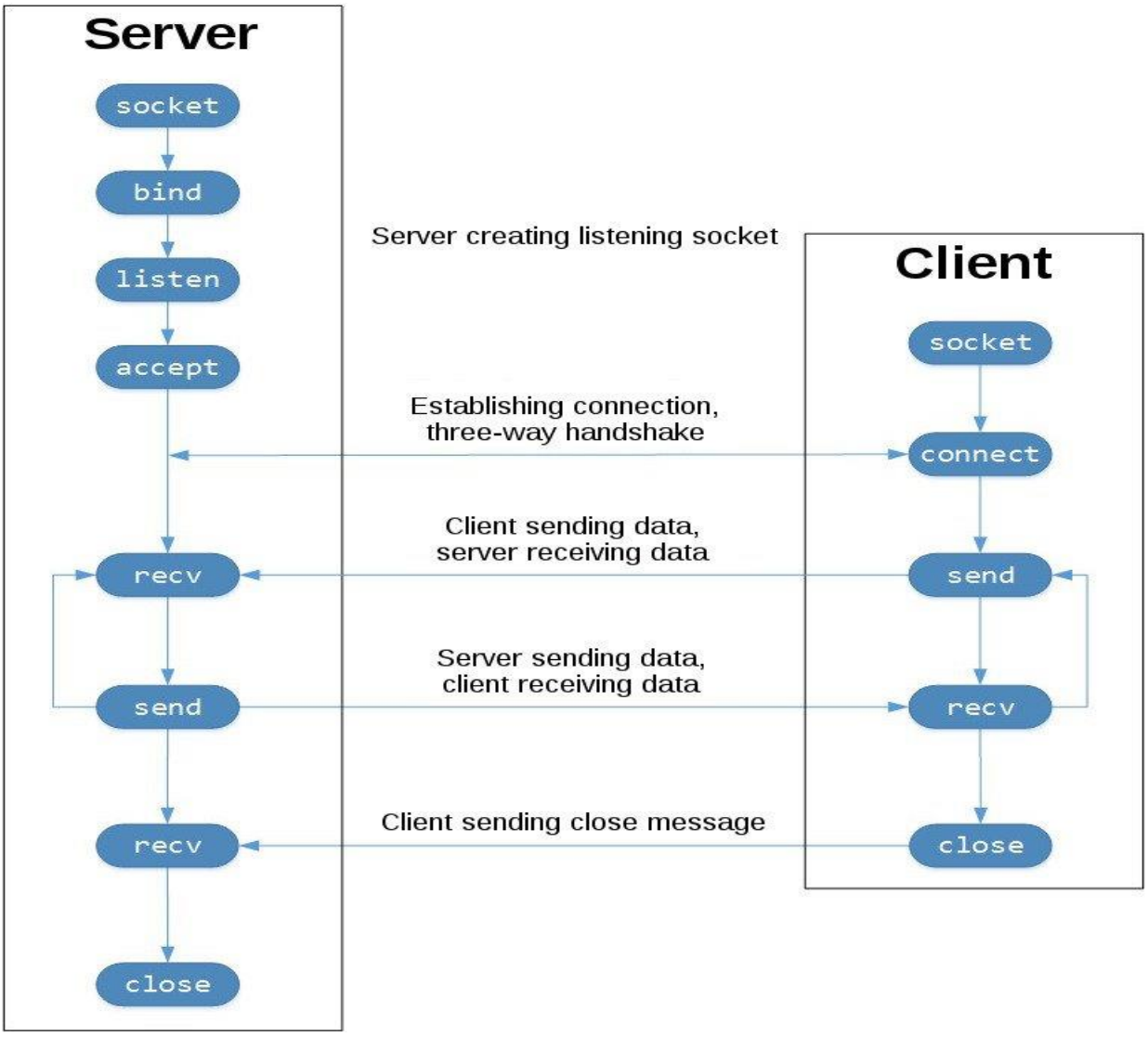
Socket programming is started by importing the socket library and making a simple socket.

```
import socket
serverSocket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Here we made a socket instance and passed it two parameters.Now we can connect to a server using this socket.

- AF_INET refers to the address family ipv4.
- The SOCK_STREAM means connection oriented TCP protocol.

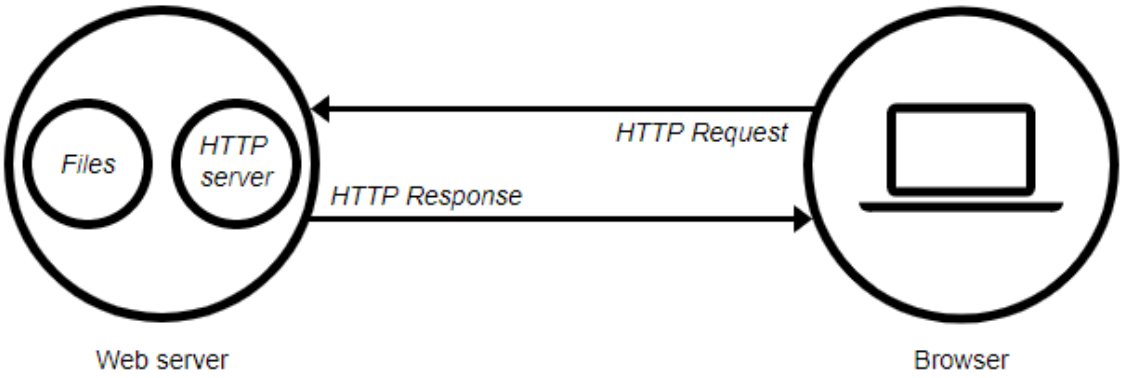
Server – Client Program:



It should be complete three Socket Programming using Phyton.

1. Web Server
2. UDP Pinger
3. Mail Client

Web Server : A web server is a computer that stores web server software and a website's component files. (for example, HTML documents, images, CSS stylesheets, and JavaScript files) A web server connects to the Internet and supports physical data interchange with other devices connected to the web. On the software side, a web server includes several parts that control how web users access hosted files. At a minimum, this is an *HTTP server*. An HTTP server is software that understands URLs (web addresses) and HTTP (the protocol your browser uses to view webpages). An HTTP server can be accessed through the domain names of the websites it stores, and it delivers the content of these hosted websites to the end user's device.



UDP Pinger : The client should send 10 pings to a UDP server. Because UDP is an unreliable protocol, a packet sent from the client to the server may be lost in the network, or vice versa. The client wait up to one second for a reply; if no reply is received within one second, theclient program should assume that the packet was lost during transmission across the network. The client should also calculate the round trip time for each of the pings sent to the server.

Mail Client: An SMTP server is a computer or an app that is responsible for sending emails. It functions following the Simple Mail Transfer Protocol (SMTP). An SMTP server receives emails from the email client. Then it passes them on to another SMTP server and relays them to the incoming mail server. A mail server is the computerized equivalent of your friendly neighborhood mailman. Every email that is sent passes through a series of mail servers along its way to its intended recipient. Although it may seem like a message is sent instantly - zipping from one PC to another in the blink of an eye - the reality is that a complex series of transfers takes place. Without this series

of mail servers, you would only be able to send emails to people whose email address domains matched your own

1.2 System Requirements

For all program , it must be used Python Programming language

Web Server : Python standard library comes with a in-built webserver which can be invoked for simple web client server communication. The port number can be assigned programmatically and the web server is accessed through this port. Though it is not a full featured web server which can parse many kinds of file, it can parse simple static html files and serve them by responding them with required response codes.

```
port = 12345
Next bind to the port we have not typed any IP in the IP field instead we have
inputted an empty string   this makes the server listen to requests
coming from other computers on the network
serverSocket.bind(('', port))

and put the socket into listening mode
serverSocket.listen(5)

Establish the connection
While(true):
        Try:
        Except:
        .
        .
        .
Establish connection with client.
c, addr = serverSocket.accept()
print ('Got connection from',addr)

Send one HTTP header line into socket
connectSocket.send('\nHTTP/1.1 200 OK\n\n'.encode())

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

Send HTTP response message for file not found
```

```
connectionSocket.send("\nHTTP/1.1 404 Not Found\n\n".encode())
```

UDP Pinger :

Server :

Assign IP address and port number to socket

```
serverSocket.bind(('', 12000))
while True:
    Generate random number in the range of 0 to 10
    rand = random.randint(0, 10)
    Receive the client packet along with the address it is coming from
    msg, addr = serverSock.recvfrom(1024)
    Capitalize the message from the client
    msg = msg.upper()
    If rand is less is than 4, we consider the packet lost and do not respond
    if rand < 4:        continue
   Otherwise, the server responds
   serverSock.sendto(msg, addr)
```

client :

```
msg = 'ping'
addr = ("localhost", 12000)
for i in range(1,11):
    start = time.time()
    clientSock.sendto(msg.encode(), addr)
     try:
    response, server = clientSock.recvfrom(1024)
    end = time.time()
     RTT = end - start
```

Mail Client:

Create socket called clientSocket and establish a TCP connection with mailserver

```
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect(mailserver) recv = clientSocket.recv(1024)

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

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] != '250': print('250 reply not received from server.')

Info for username and password
username = "cerengumus0932@gmail.com"
the username for your server
password = "151044071"
the password for your server, changed here
base64_str = ("\x00"+username+"\x00"+password).encode() base64_str = base64.b64encode(base64_str) authMsg ="message".encode()+base64_str+"\r\n".encode() clientSocket.send(authMsg) recv_auth = clientSocket.recv(1024)

if recv1[:3] != '250': print('250 reply not received from server.')

command and print server response.
mailFrom = "MAIL FROM: <cerengumus0932@gmail.com> \r\n"
clientSocket.send(mailFrom.encode())
recv2 = clientSocket.recv(1024)
if recv1[:3] != '250':
# Send RCPT TO command and print server response.
rcptTo = "RCPT TO: <cerengumus0932@gmail.com> \r\n"
clientSocket.send(rcptTo.encode()) recv3 = clientSocket.recv(1024)

if recv1[:3] != '250': print('250 reply not received from server.')
command and print server response. data = "DATA\r\n"clientSocket.send(data.encode())

recv4 = clientSocket.recv(1024)

if recv1[:3] != '250': # Send message data.

subject = "Subject: SMTP mail client testing \r\n\r\n"
clientSocket.send(subject.encode()) message = raw_input("Enter your message: \r\n") clientSocket.send(message.encode()) clientSocket.send(endmsg.encode()) recv_msg = clientSocket.recv(1024) print("Response after sending message body:"+recv_msg.decode()) if recv1[:3] != '250': print('250 reply not received from server.')

Send QUIT command and get server response.
clientSocket.send("QUIT\r\n".encode()) message=clientSocket.recv(1024) print (message) clientSocket.close()

2 METHOD

2.1 Use Case

Web server : *Python Web_server.py*
Browser : *http://127.0.0.1:6789/HelloWorld.html*
optional Exercises2: *Python client.py http://127.0.0.1 6789 HelloWorld.html*

UDP Pinger:
Python UDP_Pinger_server.py
Python client_UDP.py

Mail Client:
Python MailClient.py

2.2 Problem Solution Approach

Web Server :

(i) Create a connection socket when contacted by a client (browser);
(ii) Receive the HTTP request from this connection;
(iii) Parse the request to determine the specific file being requested;
(iv) Get the requested file from the server's file system;
(v) Create an HTTP response message consisting of the requested file preceded by header lines;
(vi) Send the response over the TCP connection to the requesting browser. If a browser requests a file that is not present in your server, your server should return a "404 Not Found" error message.
(vii) Browsers running on different hosts. If you run your server on a host that already has a Web server running on it, then you should use a different port than port 80 for your Web server.

UDP Pinger :

(i) Client will send a simple ping message to a server
(ii) Receive a corresponding pong message back from the server
(iii) Determine the delay between when the client sent the ping message and received the pong message.
(iv) Ping program is to send 10 ping messages to the target server over UDP.
(v) For each message, your client is to determine and print the RTT when the corresponding pong message is returned.
(vi) The client cannot wait indefinitely for a reply to a ping message.

(vii) It should have the client wait up to one second for a reply from the server; if no reply is received, the client should assume that the packet was lost and print a message accordingly.

Mail Client:

(i) Create mail client that sends email to any recipient.

(ii) Client will need to establish a TCP connection with a mail server (e.g., a Google mail server), dialogue with the mail server using the SMTP protocol, send an email message to a recipient (e.g., your friend) via the mail server

(iii) Close the TCP connection with the mail server.

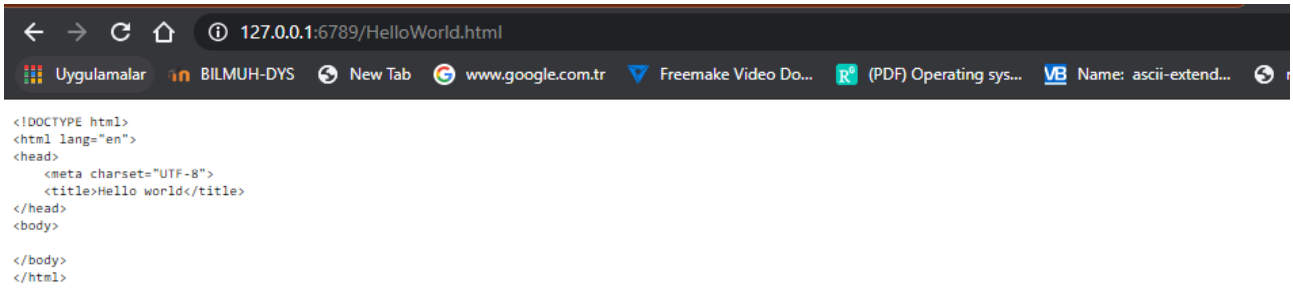
PROJECT CODE RUNNING RESULT:

Web Server:

Web_server.py

```
C:\Users\crngm\AppData\Local\Programs\Python\Python38-32\python.exe C:/Users/crngm/Desktop/151044071_Final/Web_server.py
socket binded to 6789
socket is listening
the web server is up on port: 6789
Ready to serve...
```

Browser :



Optional Exercises – Web Server – Exercises1 :

Server.py

```
Terminal:  Local  Local (2)  +
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\crngm\Desktop\151044071_Final\optionalExercisesWebServer\Exercises1>PYTHON MultiThreadServer.py
Ready to serve...
```

Client.py

```
Terminal:  Local  Local (2)  Local (3)  +
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\crngm\Desktop\151044071_Final\optionalExercisesWebServer\Exercises1>python client.py
Ready to serve...

<!DOCTYPE html>
<html lang="">
<head>
        <meta charset="utf-8">
        <title>Hello World</title>
</head>
<style>
        body {background-color: cornsilk;}
        h1      {color:saddlebrown;font-family:calibri; opacity:0.4;text-align: center}
</style>
<body>
        <div>
                <h1>Hello World</h1>
        </div>
</body>
</html>
Ready to serve...
Ready to serve...
```

Browser:

127.0.0.1/HelloWorld.html

Uygulamalar | BILMUH-DYS | New Tab | www.google.com.tr | Freemake Video Do... | (PDF) Operating sys...

```
<!DOCTYPE html>
<html lang="">
<head>
        <meta charset="utf-8">
        <title>Hello World</title>
</head>
<style>
        body {background-color: cornsilk;}
        h1      {color:saddlebrown;font-family:calibri; opacity:0.4;text-align: center}
</style>
<body>
        <div>
                <h1>Hello World</h1>
        </div>
</body>
</html>
```

Optional Exercises – Web Server – Exercises2 :

Server.py :

```
C:\Users\crngm\Desktop\151044071_Final\optionalExercisesWebServer\Exercises2>python server.py
Ready to serve...
addr:
 ('127.0.0.1', 49333)
data :
<!DOCTYPE html>
<html lang="">
<head>
        <meta charset="utf-8">
        <title>Hello World</title>
</head>
<style>
        body {background-color: cornsilk;}
        h1      {color:saddlebrown;font-family:calibri; opacity:0.4;text-align: center}
</style>
<body>
        <div>
                <h1>Hello World</h1>
        </div>
</body>
</html>
Ready to serve...
addr:
 ('127.0.0.1', 49333)

C:\Users\crngm\Desktop\151044071_Final\optionalExercisesWebServer\Exercises2>
```

client.py

```
151044071_Final 〉 optionalExercisesWebServer 〉 Exercises2 〉 server.py
Project    Exercises2\server.py    Exercises2\client.py    optionalExercisesWebServer\client.py    optionalExercisesWebServer
Terminal:  Local (2)    Local    +
data :  b'\n<!DOCTYPE html>\n<html lang="">\n<head>\n\t<meta charset="utf-8">\n\t<title>Hello World</title>\n</head>\n<style>\
\th1 \t{color:saddlebrown;font-family:calibri; opacity:0.4;text-align: center}\n</style>\n<body>\n\t<div>\n\t\t<h1>Hello World

C:\Users\crngm\Desktop\151044071_Final\optionalExercisesWebServer\Exercises2>python client.py 127.0.0.1 6791 HelloWorld.html
server host : 127.0.0.1
server port : 6791
filename : HelloWorld.html
host_port : 127.0.0.1:6791
data :
<!DOCTYPE html>
<html lang="">
<head>
        <meta charset="utf-8">
        <title>Hello World</title>
</head>
<style>
        body {background-color: cornsilk;}
        h1      {color:saddlebrown;font-family:calibri; opacity:0.4;text-align: center}
</style>
<body>
        <div>
                <h1>Hello World</h1>
        </div>
</body>
</html>

C:\Users\crngm\Desktop\151044071_Final\optionalExercisesWebServer\Exercises2>
Run    TODO    Problems    Terminal    Python Console
System clipboard is unavailable (moments ago)    7:23    CRLF    UTF-
Type here to search
```

UDP PINGER:

UDP_pinger_server.py

```
151044071_Final  C:\Users\crngm\Desktop\151044071_Final
Terminal:  Local    Local (2)    +
Microsoft Windows [Version 10.0.19041.685]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\crngm\Desktop\151044071_Final>pyhton UDP_pinger_server.py
'pyhton' is not recognized as an internal or external command,
operable program or batch file.

C:\Users\crngm\Desktop\151044071_Final>python UDP_pinger_server.py
PING SERVER
random number: 2
random number: 4
random number: 6
random number: 3
random number: 0
random number: 8
random number: 3
random number: 3
random number: 5
random number: 3
random number: 1
```

client_UDP.py

```
151044071_Final C:\Users\crngm\Desktop\151044071_Final
Terminal: Local  Local (2)  +

C:\Users\crngm\Desktop\151044071_Final>PYTHON client_UDP.py
#1
Request timed out
#2
Response :b'PING'
RTT:0.001004sec
#3
Response :b'PING'
RTT:0.001000sec
#4
Request timed out
#5
Request timed out
#6
Response :b'PING'
RTT:0.002982sec
#7
Request timed out
#8
Request timed out
#9
Response :b'PING'
RTT:0.001114sec
#10
Request timed out
```

Optional Exercises –UDP PINGER– Exercises1 :

UDPPingServer.py:

```
151044071_Final 〉 optionalExercisesUDP 〉 Exercises1 〉 UDPPingClient.py
Project  Web_server.py  MultiThreadServer.py  HelloWorld.ht
151044071_Final C:\Users\crngm\Desktop\151044071_Final   while seq <= number_pings 〉 if type(recieved) == byte
Terminal: Local  Local (2)  Local (3)  +
C:\Users\crngm\Desktop\151044071_Final\optionalExercisesUDP\Exercises1>PYTHON UDPPingServer.py
Recieve:  b'PING 1 1609635359003'
Send:  b'PING 1 1609635359003'
Recieve:  b'PING 2 1609635359769'
Recieve:  b'PING 2 1609635360773'
Send:  b'PING 2 1609635360773'
Recieve:  b'PING 3 1609635361132'
Send:  b'PING 3 1609635361132'
Recieve:  b'PING 4 1609635361511'
Recieve:  b'PING 4 1609635362522'
Send:  b'PING 4 1609635362522'
Recieve:  b'PING 5 1609635363508'
Send:  b'PING 5 1609635363508'
Recieve:  b'PING 6 1609635363743'
Send:  b'PING 6 1609635363743'
Recieve:  b'PING 7 1609635364171'
Send:  b'PING 7 1609635364171'
Recieve:  b'PING 8 1609635364678'
Recieve:  b'PING 8 1609635365694'
Recieve:  b'PING 8 1609635366714'
Send:  b'PING 8 1609635366714'
Recieve:  b'PING 9 1609635367151'
Send:  b'PING 9 1609635367151'
Recieve:  b'PING 10 1609635367806'
Send:  b'PING 10 1609635367806'
```

UDPPingClient.py:

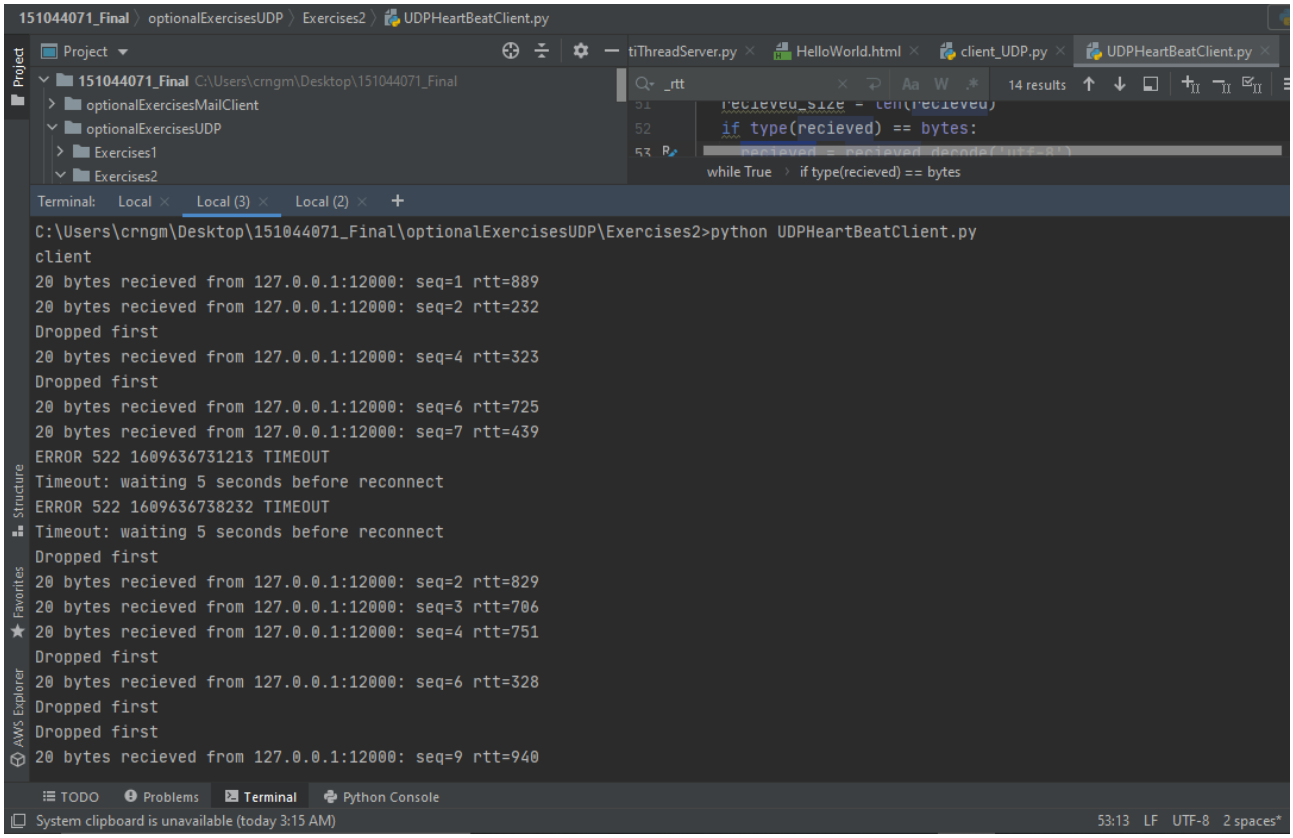
```
C:\Users\crngm\Desktop\151044071_Final\optionalExercisesUDP\Exercises1>python UDPPingClient.py
UDP PING CLIENT
message: PING 1 1609635359003
received: b'PING 1 1609635359003'
['PING', '1', '1609635359003']
20 bytes recieved from 127.0.0.1:12000: seq=1 rtt=766
message: PING 2 1609635359769
received: ERROR 522 1609635360770 TIMEOUT
['ERROR', '522', '1609635360770', 'TIMEOUT']
ERROR 522 1609635360770 TIMEOUT
message: PING 2 1609635360773
received: b'PING 2 1609635360773'
['PING', '2', '1609635360773']
20 bytes recieved from 127.0.0.1:12000: seq=2 rtt=358
message: PING 3 1609635361132
received: b'PING 3 1609635361132'
['PING', '3', '1609635361132']
20 bytes recieved from 127.0.0.1:12000: seq=3 rtt=378
message: PING 4 1609635361511
received: ERROR 522 1609635362520 TIMEOUT
['ERROR', '522', '1609635362520', 'TIMEOUT']
ERROR 522 1609635362520 TIMEOUT
message: PING 4 1609635362522
received: b'PING 4 1609635362522'
['PING', '4', '1609635362522']
20 bytes recieved from 127.0.0.1:12000: seq=4 rtt=986
```

Optional Exercises –UDP PINGER– Exercises2 :

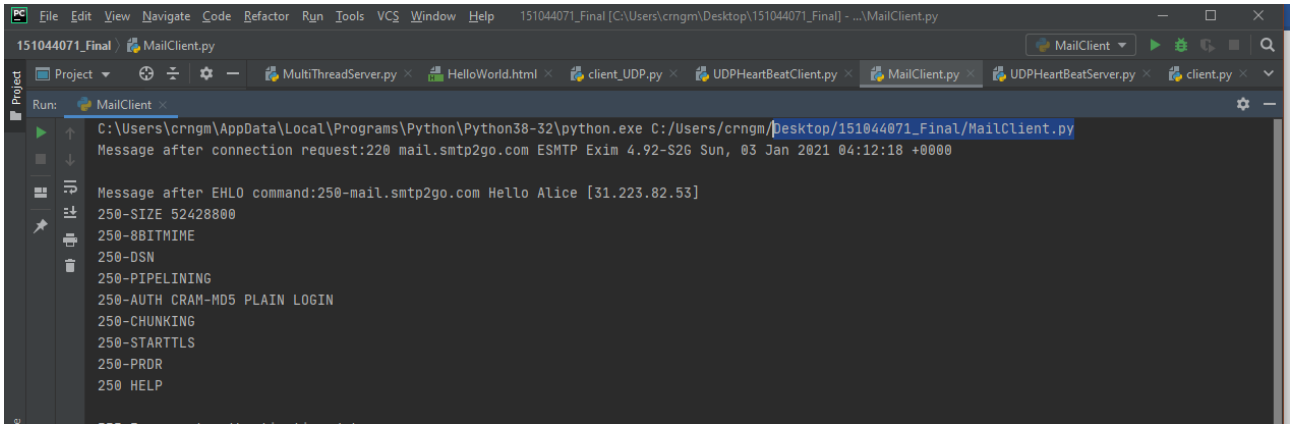
UDPHeartBeatServer.py:

```
C:\Users\crngm\Desktop\151044071_Final\optionalExercisesUDP\Exercises2>python UDPHeartBeatServer.
ready to server...
1609636720541
message: b'PING 1 1609636720541'
Recieve: PING 1 1609636720541
Send: PING 1 1609636720541
1901
message: b'PING 2 1609636722442'
Client connect.
Recieve: PING 2 1609636722442
Send: PING 2 1609636722442
2251
message: b'PING 4 1609636724693'
Dropped Packet:2
Dropped Packet:3
Recieve: PING 4 1609636724693
Send: PING 4 1609636724693
2342
message: b'PING 6 1609636727035'
Dropped Packet:4
Dropped Packet:5
Recieve: PING 6 1609636727035
```

UDPHeartBeatClient.py:



MAIL CLIENT:



Optional Exercises – MAIL CLIENT – Exercises :

Smtp_TLS.py: