

NAME - GAUTAM KUMAR MAHAR(2103114)

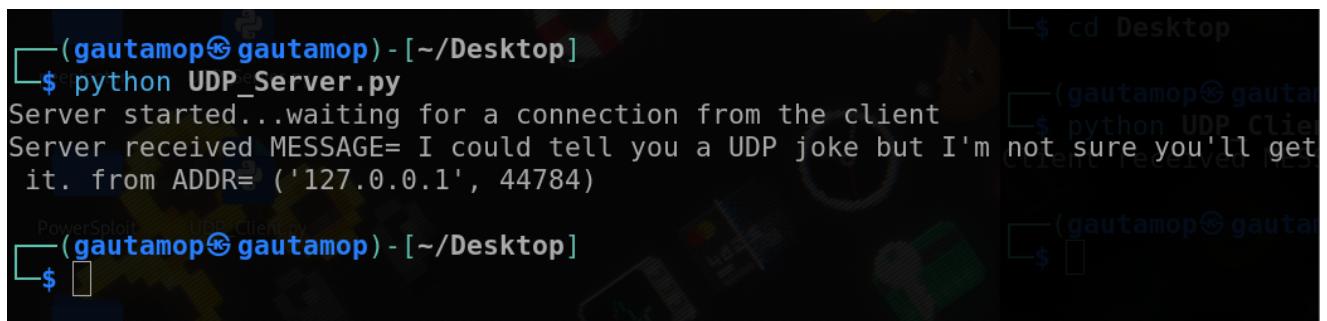
LAB 2 (SOCKTES)

PART 1: UDP SOCKETS

Question 1.

Study the template provided for using UDP sockets in Python (files UDP_Server.py and UDP_Client.py). Make sure that you understand the purpose of UDP sockets, and the steps for creating and using these sockets. Run the Client and Server processes and observe the output.

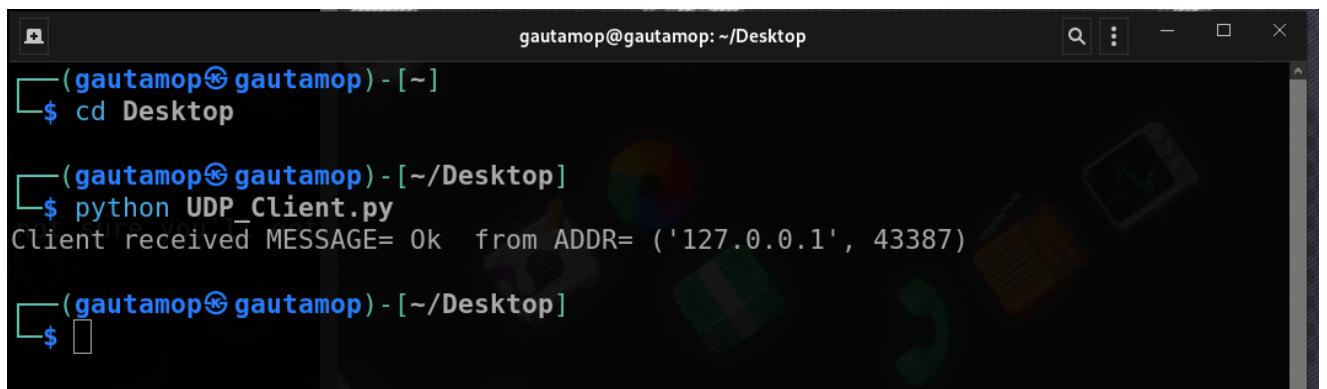
UDP Server:



```
(gautamop㉿gautamop) - [~/Desktop]
$ python UDP_Server.py
Server started...waiting for a connection from the client
Server received MESSAGE= I could tell you a UDP joke but I'm not sure you'll get
it. from ADDR= ('127.0.0.1', 44784)

(gautamop㉿gautamop) - [~/Desktop]
$
```

UDP Client:



```
gautamop@gautamop: ~/Desktop
(gautamop㉿gautamop) - [~]
$ cd Desktop

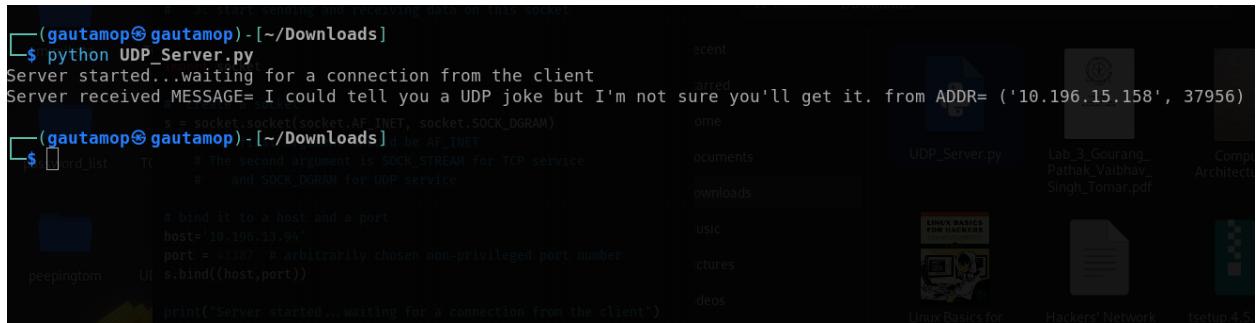
(gautamop㉿gautamop) - [~/Desktop]
$ python UDP_Client.py
Client received MESSAGE= Ok  from ADDR= ('127.0.0.1', 43387)

(gautamop㉿gautamop) - [~/Desktop]
$
```

Question 2

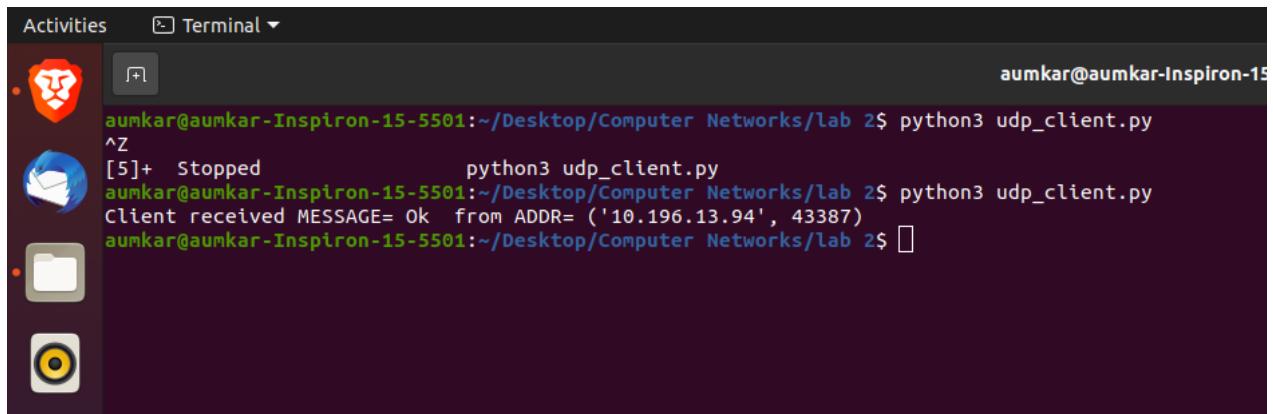
Run the Client and the Server processes on different computers and check if they work as expected (you will need to specify the server's IP address).

Server Screen:



```
(gautamop@gautamop) - [~/Downloads]
$ python UDP_Server.py
Server started...waiting for a connection from the client
Server received MESSAGE= I could tell you a UDP joke but I'm not sure you'll get it. from ADDR= ('10.196.15.158', 37956)
(gautamop@gautamop) - [~/Downloads]
```

Client Screen:



```
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 2$ python3 udp_client.py
^Z
[5]+  Stopped                  python3 udp_client.py
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 2$ python3 udp_client.py
Client received MESSAGE= Ok  from ADDR= ('10.196.13.94', 43387)
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 2$
```

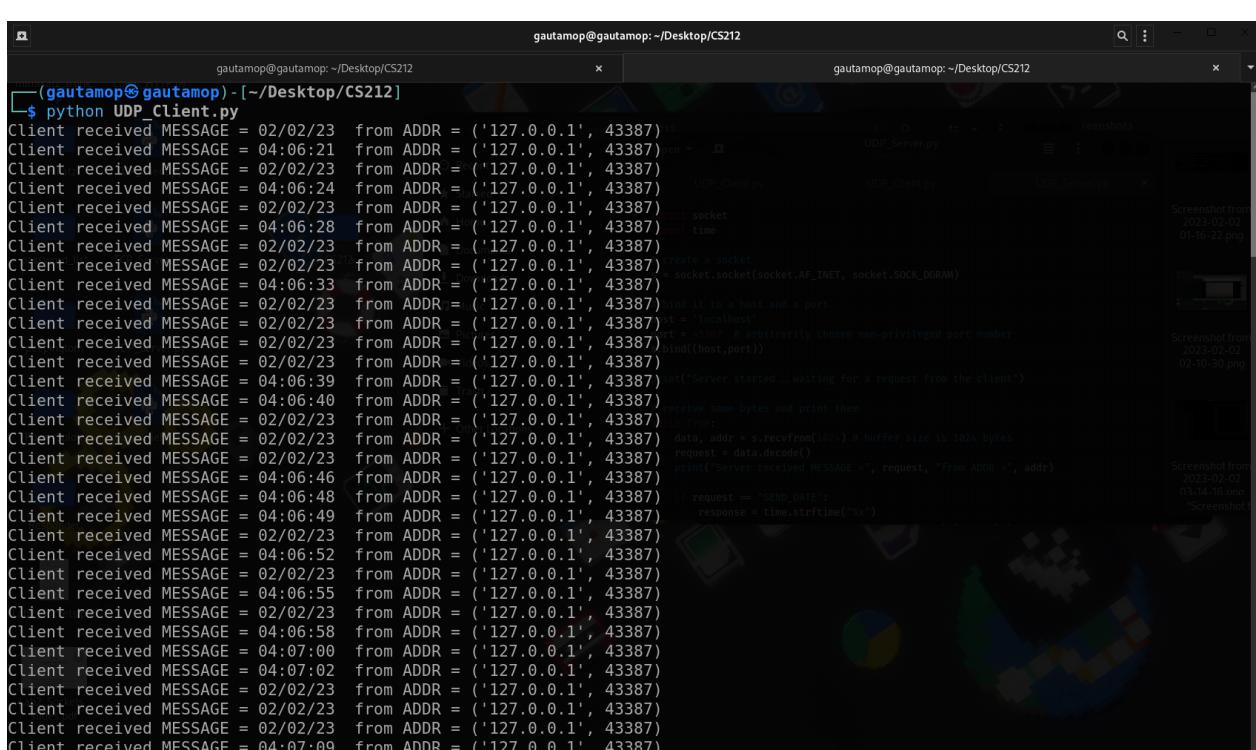
Question 3

Design and implement a Client-Server system that uses UDP sockets to do the following:

- The client sends the server a request. The request string can either be: “SEND_DATE” or “SEND_TIME”.
- The server runs in a infinite loop where it keeps waiting for requests. Whenever it sees a request, it responds by sending either the current DATE or the current TIME in (HH:MM:SS) format as specified in the request.
- When the Client receives a response, it prints it.
- The Client runs in a loop where it generates multiple such requests, and the time between successive requests varies randomly between 1-2 seconds.

HINT: You can use the following line of code to generate a random amount of delay, uniformly distributed between 1-2 seconds:

Client Screen:



```
gautamop@gautamop:~/Desktop/CS212
(gautamop@gautamop) [~/Desktop/CS212]
$ python UDP_Client.py
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:21 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:24 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:28 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:33 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:39 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:40 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:46 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:48 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:49 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:52 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:55 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:06:58 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:07:00 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:07:02 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 02/02/23 from ADDR = ('127.0.0.1', 43387)
Client received MESSAGE = 04:07:09 from ADDR = ('127.0.0.1', 43387)
```

Here, It sends requests at random between SEND DATE and SEND TIME while running in an indefinite loop. It prints the response on the client screen after getting it from the server.

Server Screen:

Here, It is operating in an endless loop as it waits for requests from the client; everytime one comes in, it prints the request on the server screen and sends the client the response.

PART 2: TCP SOCKETS [5 points]

Question 4 -

Study the template provided for using TCP sockets in Python (files TCP_Server.py and TCP_Client.py). Make sure that you understand the purpose of TCP sockets and the steps for creating and using these sockets. Observe the differences between UDP and TCP sockets and the steps for their use. Run the TCP Client and Server processes and observe the output.

TCP Server Screen:

```
gautamop@gautamop: ~/Desktop
(gautamop@gautamop) - [~/Desktop]
$ python TCP_Server.py
Server started...waiting for a connection from the client
Connection initiated from ('127.0.0.1', 47264)
SERVER RECEIVED: Knock knock..
(gautamop@gautamop) - [~/Desktop]
$ # connect to the server
host='localhost'
port=43389 # this is the server's port number, which the client needs to know
s.connect((host,port))

# send some bytes
s.send("Knock knock..".encode('utf-8'))
```

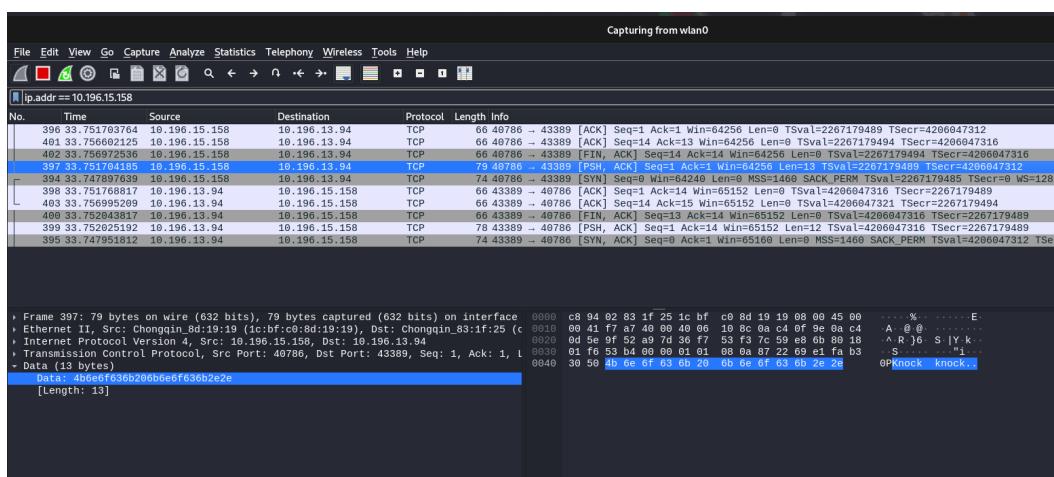
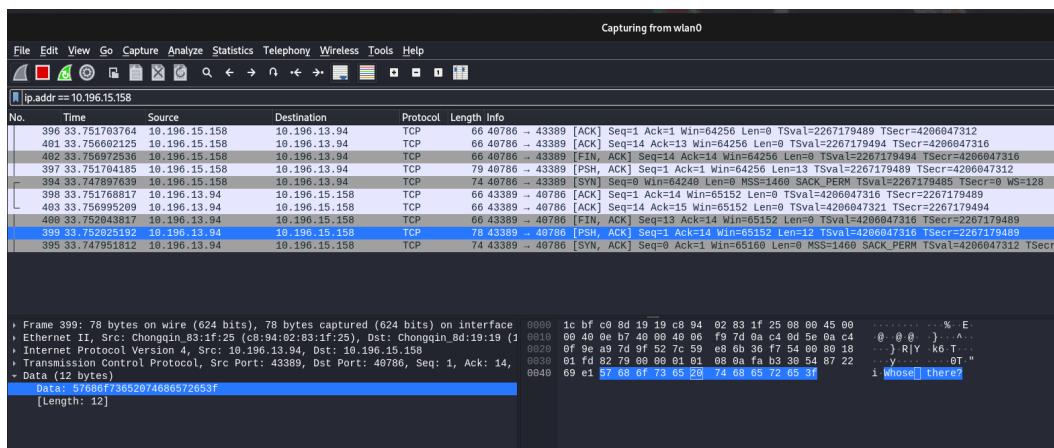
TCP Client Screen:

```
gautamop@gautamop: ~/Desktop
(gautamop@gautamop) - [~/Desktop] PART 2: TCP SOCKETS [5 points]
$ python TCP_Client.py
CLIENT RECEIVED: Who's there?
(gautamop@gautamop) - [~/Desktop]
$ #
```

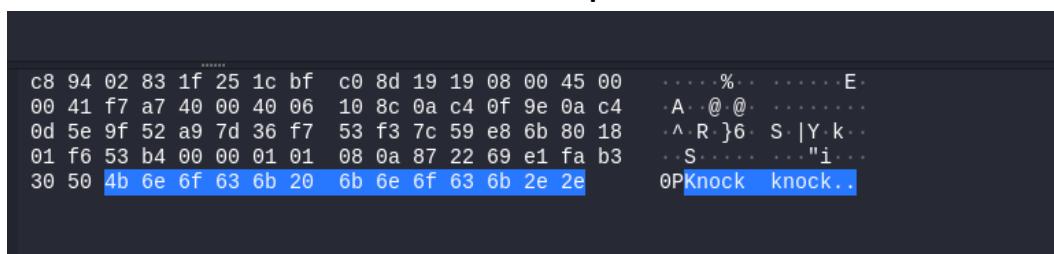
Question 5-

Start up Wireshark and apply a filter such that only the traffic generated by your Client and Server processes are displayed. Identify the messages used by TCP during the Handshake and the actual text sent by the two processes. Are the “contents” of the packet (the message strings) visible within Wireshark? (This is what we’d expect since the strings aren’t encrypted before sending.)

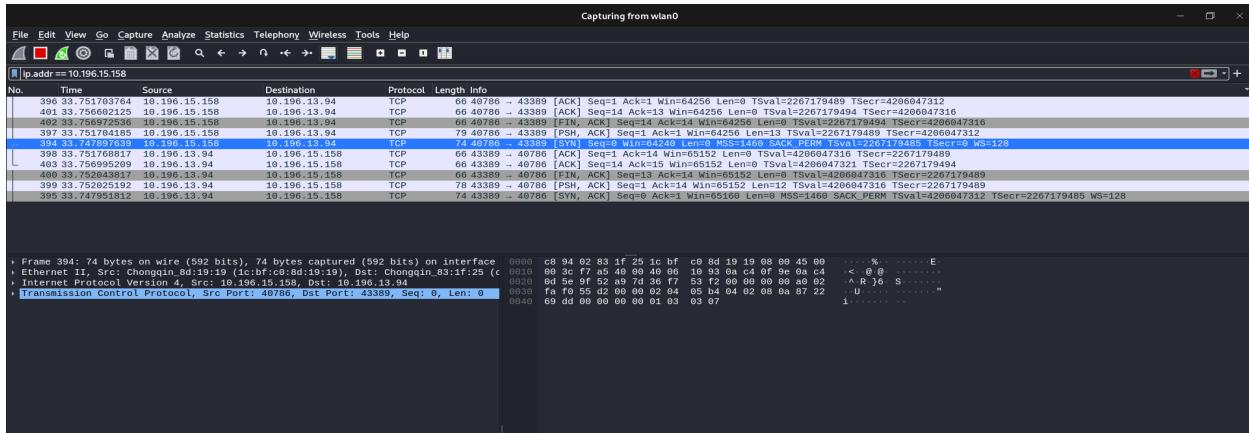
Answer-



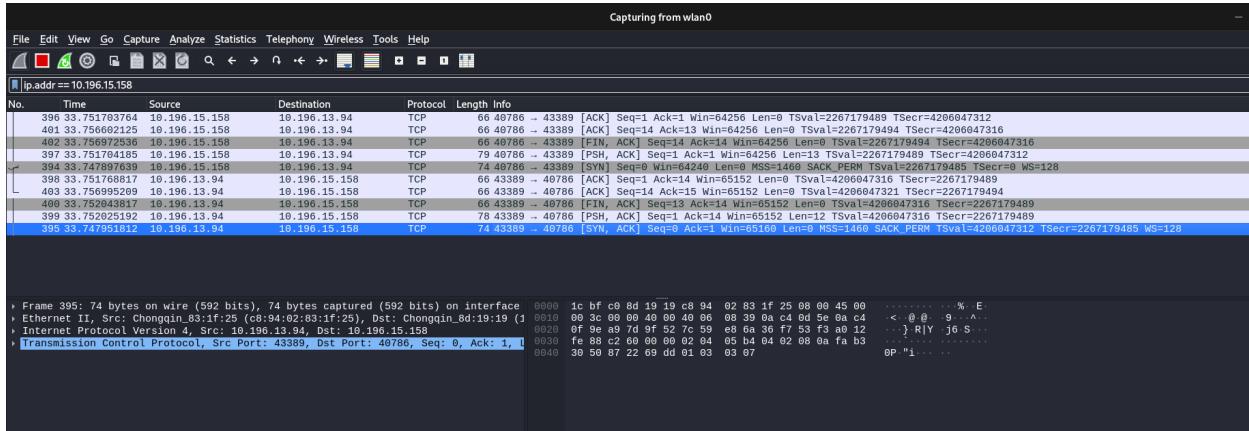
Here we are able to see the contents of the packet are visible.



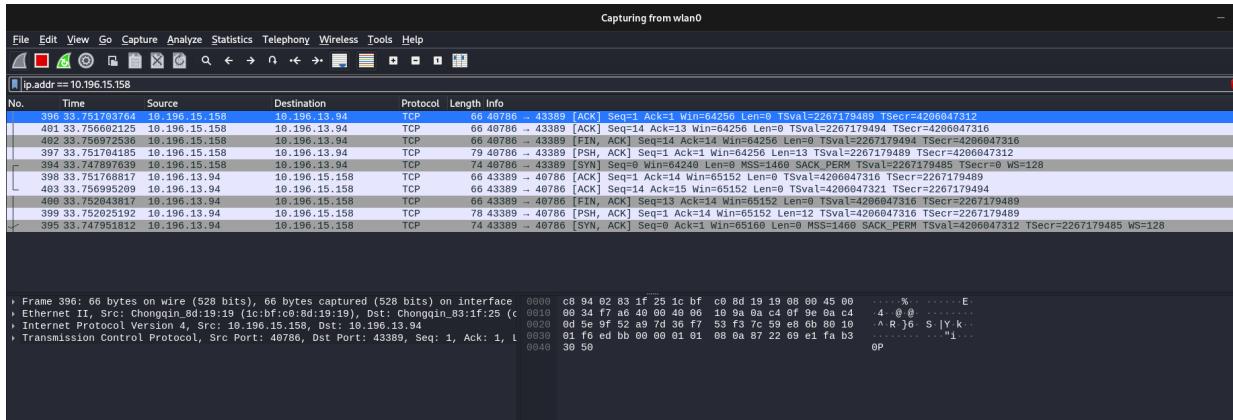
SYC



SYC ACK



ACK

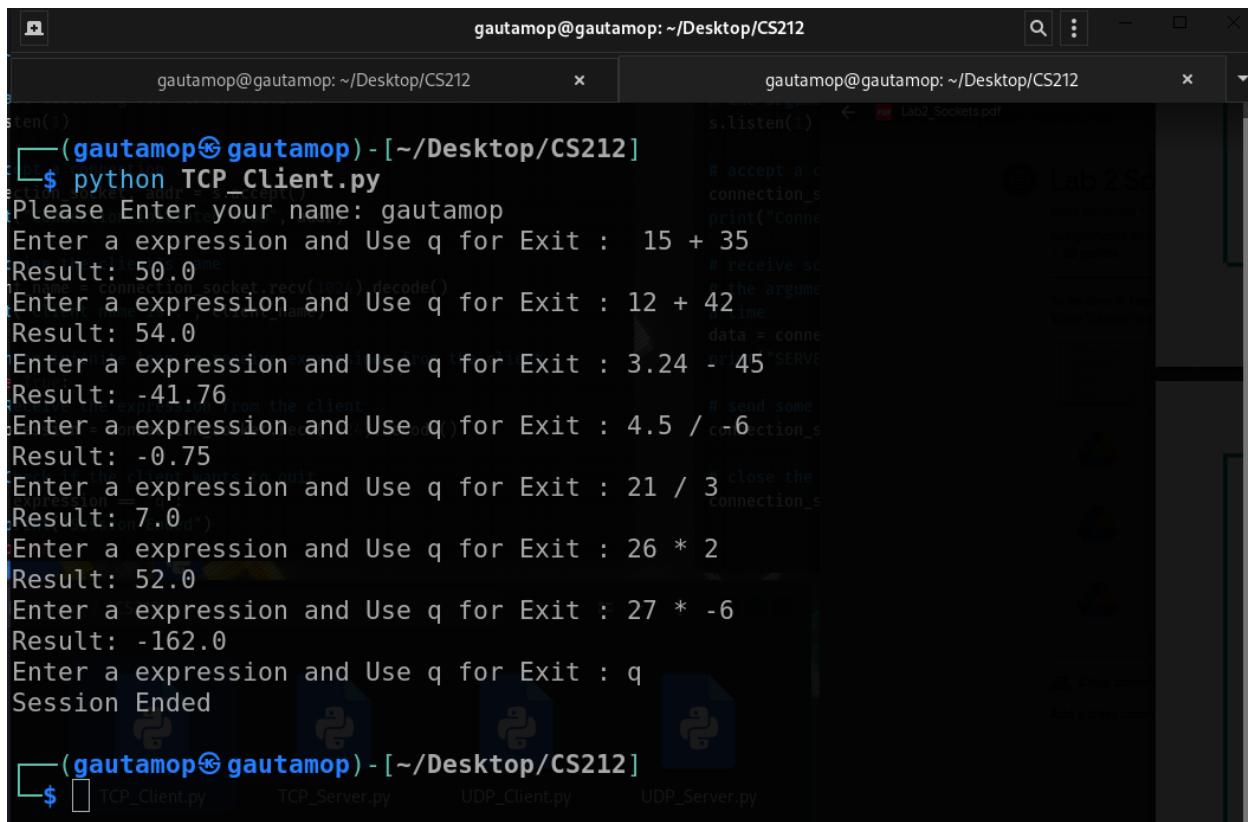


Question 6-

Design and implement a Client-Server system that uses TCP sockets to do the following:

- The client initiates communication with a Server by sending the server its name. (Choose some name for your client process). The Server remembers this name for the entire duration of the communication session.
- The client then runs in an infinite loop where it accepts a line of input from the user. The user is expected to enter a string consisting of two numbers and a simple arithmetic operation (separated by spaces), for example: “12 + 42” or “3.24 - 45” or “4.5 / -6”. If the input is not correctly formatted, a warning is displayed to the user. If correctly formatted, the Client sends this string to the Server.
- The Server runs an infinite loop where it keeps waiting for requests from this client. Upon receiving a request, the server prints the received message, computes the answer by performing the arithmetic operation and sends it back as a string. The Client prints the answer it received from the server.
- When the user wishes to stop, they enter “q”. The client process forwards the “q” to the Server upon which the server ends the communication session and prints “Session ended”. The Client processes stops.

Client Screen:



The screenshot shows a terminal window with two tabs. The left tab is titled '(gautamop@gautamop) - [~/Desktop/CS212]' and the right tab is also '(gautamop@gautamop) - [~/Desktop/CS212]'. The left tab contains the following text:

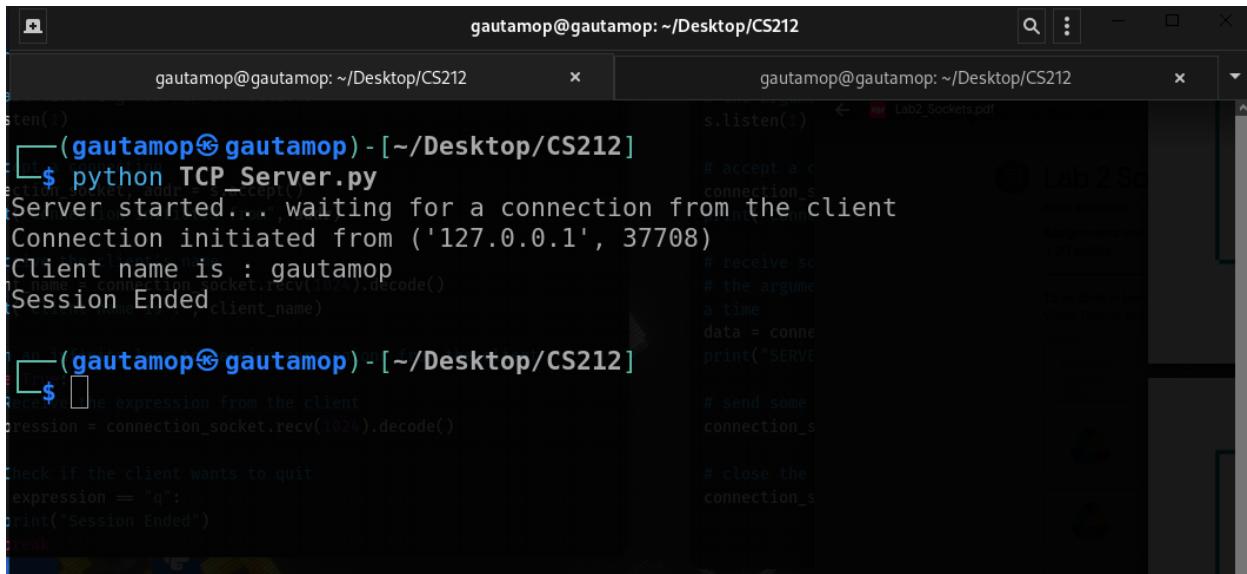
```
gautamop@gautamop:~/Desktop/CS212
$ python TCP_Client.py
Please Enter your name: gautamop
Enter a expression and Use q for Exit : 15 + 35
Result: 50.0
Enter a expression and Use q for Exit : 12 + 42
Result: 54.0
Enter a expression and Use q for Exit : 3.24 - 45
Result: -41.76
Enter a expression and Use q for Exit : 4.5 / -6
Result: -0.75
Enter a expression and Use q for Exit : 21 / 3
Result: 7.0
Enter a expression and Use q for Exit : 26 * 2
Result: 52.0
Enter a expression and Use q for Exit : 27 * -6
Result: -162.0
Enter a expression and Use q for Exit : q
Session Ended
```

The right tab shows the source code of the TCP Client:

```
open()
s.listen(1)
# accept a connection
connection_s
print("Connection established")
# receive something
data = connection_s.recv(1024).decode()
# send some
connection_s.sendall(str(result).encode())
# close the connection
connection_s.close()
```

At the bottom of the terminal, there are icons for Python files: TCP_Client.py, TCP_Server.py, UDP_Client.py, and UDP_Server.py.

Server Screen:



```
gautamop@gautamop: ~/Desktop/CS212
gautamop@gautamop: ~/Desktop/CS212
(gautamop@gautamop) [~/Desktop/CS212]
$ python TCP_Server.py
Server started... waiting for a connection from the client
Connection initiated from ('127.0.0.1', 37708)
Client name is : gautamop
Session Ended

(gautamop@gautamop) [~/Desktop/CS212]
$
```

```
# accept a connection
connection_socket, client_name = s.accept()
# receive some data
data = connection_socket.recv(1024).decode()
print("RECEIVED: " + data)

# send some data
connection_socket.sendall("Hello Client".encode())
# close the connection
connection_socket.close()
```

Question 7-

When you decide upon a “format” for the messages that can be correctly understood by the Client/Server processes, you have implicitly designed a “Protocol”. A protocol can be “Stateful” or “Stateless”. Find out and understand what these terms mean.

- Is the application-layer protocol designed by you for question 3 Stateless?

Yes, It is stateless as every request from a client is understood in isolation.

- Is the application-layer protocol designed by you for question 6 Stateless?

Yes, It is stateless as every request from a client is understood in isolation.

- Is TCP a Stateless protocol? TCP is not stateless.

Yes, It is stateful.

- Is UDP a Stateless protocol?

Yes, UDP is stateless.

Question 8-

For systems such as those described in question 6, think of how you could modify the Server so that it can handle multiple clients at the same time. For each connection (using TCP sockets), the Server should provide the same service as described.

There are several ways to implement this. One way is to use multiple threads for the Server, one per connection. A skeleton for a multi-threaded server program can be found in this answer on StackOverflow: <https://stackoverflow.com/a/40351010/1329325>

In this question, we wish to design such a multi-Client system to implement a “Chat Room” as follows:

- The Client process acts like a chat window. It takes user input, sends appropriate requests to the Server, and displays the messages sent by the server to the human user.
- The Server process acts like a chat room manager. It allows client processes to login to the chat room (each client needs to have a unique name). The server keeps track of all the clients that are currently logged in. Whenever any interesting event happens, (such as new user logging in or leaving the chat room) the status is broadcast to all connected clients. Also, whatever each user types is broadcast to all clients.
- At the beginning, the user is requested for a “login name”. The client process then sends a login request to the chat room (Server) with this name.
- After logging in, whatever lines the user types is broadcast to all clients along with the sender’s name. The following lines show an example of output that might be displayed to two different clients. Client 1 is the first to join the chat room.

When first person joined ! Server get notified

The image shows four terminal windows arranged in a 2x2 grid, illustrating the communication between a TCP Server and two clients.

- Top Left Terminal:** Shows the TCP Server process running. It prints "Server started... waiting for a connection from the client" and "Connected ('127.0.0.1', 37390)". It also shows the message "new user name is :Alon Musk:alanop!".
- Top Right Terminal:** Shows Client 1 (alanop) connecting. It prompts "Please, Enter Your Name:Alon Musk" and "Please, Enter Your ig name". The user enters "alanop". The server responds with "new user :Alon Musk:alanop connected!" and "new user :Alon Musk:alanop joined".
- Bottom Left Terminal:** Shows Client 2 (gautamop) connecting. It prints the prompt "\$".
- Bottom Right Terminal:** Shows Client 2 (gautamop) connected. It prints the prompt "\$".

When Other two person joined and then server also get notified

The image shows four terminal windows arranged in a 2x2 grid. The top-left window shows the server starting and accepting connections from three clients. The top-right window shows the first client (Alan Musk) entering his name and ig name. The bottom-left window shows the second client (Jeff Bezos) entering his name and ig name. The bottom-right window shows the third client (Gautam) entering his name and ig name. All clients are listed as connected users.

```
(gautamop@gautamop) [~/Desktop/BONUS]
$ python TCP_Server_Q8.py
Server started... waiting for a connection from the client
Connected ('127.0.0.1', 37390)
new user name is :Alan Musk:alanop!
Connected ('127.0.0.1', 33856)
new user name is :Jeff Bezos:jeffop!
Connected ('127.0.0.1', 60598)
new user name is :Gautam:gautamop!
```

```
(gautamop@gautamop) [~/Desktop/BONUS]
$ cd Desktop
(gautamop@gautamop) [~/Desktop]
$ cd BONUS
(gautamop@gautamop) [~/Desktop/BONUS]
$ python TCP_Client_Q8.py
Please, Enter Your Name:Alan Musk
Please, Enter Your ig name
alanop
new user :Alan Musk:alanop connected!
new user :Alan Musk:alanop joined

new user :Jeff Bezos:jeffop joined

new user :Gautam:gautamop joined
```

```
(gautamop@gautamop) [~/Desktop]
$ cd Desktop
(gautamop@gautamop) [~/Desktop]
$ cd BONUS
(gautamop@gautamop) [~/Desktop/BONUS]
$ python TCP_Client_Q8.py
Please, Enter Your Name:Jeff Bezos
Please, Enter Your ig name
jeffop
new user :Jeff Bezos:jeffop connected!

new user :Jeff Bezos:jeffop joined

new user :Gautam:gautamop joined
```

```
(gautamop@gautamop) [~/Desktop]
$ cd Desktop
(gautamop@gautamop) [~/Desktop]
$ cd BONUS
(gautamop@gautamop) [~/Desktop/BONUS]
$ python TCP_Client_Q8.py
Please, Enter Your Name:Gautam
Please, Enter Your ig name
gautamop
new user :Gautam:gautamop connected!

new user :Gautam:gautamop joined
```

And finally all are chatting with eachother ...

The image shows four terminal windows arranged in a 2x2 grid. The top-left window shows the server accepting connections from three clients. The top-right window shows the first client (Alan Musk) entering his name and ig name. The bottom-left window shows the second client (Jeff Bezos) entering his name and ig name. The bottom-right window shows the third client (Gautam) entering his name and ig name. All clients are listed as connected users. In the bottom-right window, the clients begin to chat with each other, with messages like "Hey Guys", "What's Up Guys?", and "How are you guys ?".

```
(gautamop@gautamop) [~/Desktop/BONUS]
$ python TCP_Server_Q8.py
Server started... waiting for a connection from the client
Connected ('127.0.0.1', 37390)
new user name is :Alan Musk:alanop!
Connected ('127.0.0.1', 33856)
new user name is :Jeff Bezos:jeffop!
Connected ('127.0.0.1', 60598)
new user name is :Gautam:gautamop!
```

```
(gautamop@gautamop) [~/Desktop/BONUS]
$ cd BONUS
(gautamop@gautamop) [~/Desktop/BONUS]
$ python TCP_Client_Q8.py
Please, Enter Your Name:Alan Musk
Please, Enter Your ig name
alanop
new user :Alan Musk:alanop connected!
new user :Alan Musk:alanop joined

new user :Jeff Bezos:jeffop joined

new user :Gautam:gautamop joined

:Gautam:Hey Guys
:Jeff Bezos:What's Up Guys?
:Gautam:How are you guys ?
```

```
(gautamop@gautamop) [~/Desktop]
$ cd Desktop
(gautamop@gautamop) [~/Desktop]
$ cd BONUS
(gautamop@gautamop) [~/Desktop/BONUS]
$ python TCP_Client_Q8.py
Please, Enter Your Name:Jeff Bezos
Please, Enter Your ig name
jeffop
new user :Jeff Bezos:jeffop connected!

new user :Jeff Bezos:jeffop joined

new user :Gautam:gautamop joined

Hey Guys
:Gautam:Hey Guys
:Jeff Bezos:What's Up Guys?
How are you guys ?
:Gautam:How are you guys ?
```

```
(gautamop@gautamop) [~/Desktop]
$ cd Desktop
(gautamop@gautamop) [~/Desktop]
$ cd BONUS
(gautamop@gautamop) [~/Desktop/BONUS]
$ python TCP_Client_Q8.py
Please, Enter Your Name:Gautam
Please, Enter Your ig name
gautamop
new user :Gautam:gautamop connected!

new user :Gautam:gautamop joined
```

Then I Try To Connect with another Pc

The screenshot shows two terminal windows side-by-side. The left terminal window is titled '(gautamop@gautamop) - [~/Desktop/BONUS]' and contains the following command and output:

```
gautamop@gautamop:~/Desktop/BONUS
$ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 2965 bytes 263772 (257.5 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 2965 bytes 263772 (257.5 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.196.13.94 netmask 255.255.240.0 broadcast 10.196.15.255
        inet6 fe80::bb03:ab22:c774:5bcf prefixlen 64 scopeid 0x20<link>
            ether c8:94:02:83:1f:25 txqueuelen 1000 (Ethernet)
            RX packets 356271 bytes 301764798 (287.7 MiB)
            RX errors 0 dropped 27 overruns 0 frame 0
            TX packets 228734 bytes 61917117 (59.0 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

(gautamop@gautamop) - [~/Desktop/BONUS]
$ python TCP_Server_Q8.py
Server started... waiting for a connection from the client
Connected ('10.196.13.94', 41610)
new user name is :Gautam:gautamop!
Connected ('10.196.15.158', 42816)
new user name is :Aumkar:aumkar_008!
```

The right terminal window is also titled '(gautamop@gautamop) - [~/Desktop/BONUS]' and contains the following command and output:

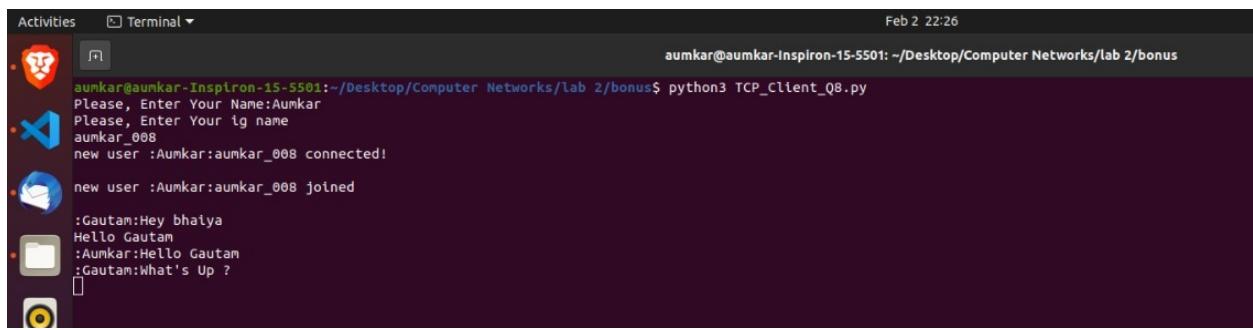
```
gautamop@gautamop:~/Desktop/BONUS
$ cd Desktop
(gautamop@gautamop) - [~/Desktop]
$ cd BONUS
(gautamop@gautamop) - [~/Desktop/BONUS]
$ ls
new.py TCP_Client_Q8.py TCP_Server_Q8.py
$ python TCP_Client_Q8.py
Please, Enter Your Name:Gautam
Please, Enter Your ig name
gautamop
new user :Gautam:gautamop connected!
new user :Gautam:gautamop joined
Client send message
new user :Aumkar:aumkar_008 joined
new user :Aumkar:aumkar_008 joined
Hey bhaiya
:Gautam:Hey bhaiya
:Aumkar:Hello Gautam
What's Up ?
:Gautam:What's Up ?
```

I able to send and receive messages...

This screenshot shows the same two terminal windows as the previous one, but with more exchanged messages. The left terminal window shows the server accepting a connection from '10.196.13.94' port 41610 and another connection from '10.196.15.158' port 42816. It also shows the server receiving messages from both clients.

The right terminal window shows the client sending messages to the server, including 'Hey bhaiya', ':Gautam:Hey bhaiya', ':Aumkar:Hello Gautam', 'What's Up ?', and ':Gautam:What's Up ?'. The server responds to these messages.

Other Pc Screen -



The screenshot shows a terminal window titled "Terminal" with the command "python3 TCP_Client_Q8.py" running. The output of the program is displayed, showing a text-based chat between two users: "Aumkar" and "Gautam". The conversation includes messages like "Please, Enter Your Ig name", "new user :Aumkar:aumkar_008 connected!", "new user :Aumkar:aumkar_008 joined", and "Hello Gautam". Both users respond with "Hello" and ask "What's Up ?".

```
aumkar@aumkar-Inspiron-15-5501:~/Desktop/Computer Networks/lab 2/bonus$ python3 TCP_Client_Q8.py
Please, Enter Your Ig name
aumkar_008
new user :Aumkar:aumkar_008 connected!
new user :Aumkar:aumkar_008 joined
:Gautam:Hey bhalya
Hello Gautam
:Aumkar>Hello Gautam
:Gautam:What's Up ?
```

Thank You !