NETWORKS LABORATORY ASSIGNEMENT 1

JEREMIAH THOMAS

106119055 | CSE -A

Do Note:

- Kindly find all the codes in the zip folder as requested
- Do cd into the right directory while running the programs.
- It is preferred to run the programs on a Linux environment.
- Before running the codes for ethernet related interfaces, do ensure that the particular IP address on the local machine matches that of the code.
- To change the IP address to match the code, one can look up : https://unix.stackexchange.com/questions/152331/how-can-i-create-a-virtual-ethernet-interface-on-a-machine-without-a-physical-ad/152334#152334?newreg=fae89b45bb1c4960ae05e6eea9554922

Question 1:-

1) Write a server program for TCP using Python to do the following:

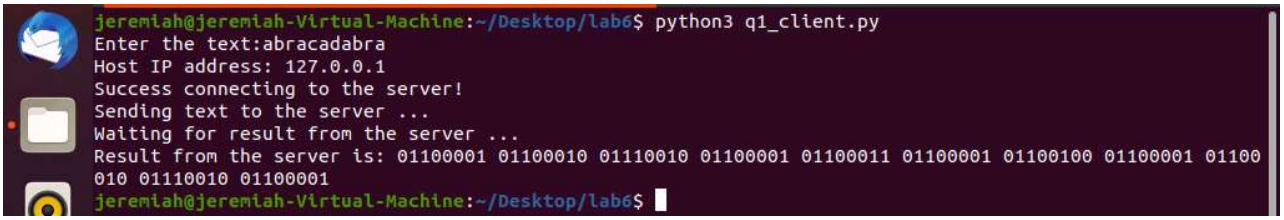
   a. Server returns the binary value of the text sent by the client. Example: for a text string "comnetsii", the client should receive "01100011 01101111 01101101 01101110 01100101 01110100 01110011 01101001 01101001"

   b. The server should be running at the localhost interface

   c. You are free to choose any port

Output:-

Client Side:-



```
jeremiah@jeremiah-Virtual-Machine:~/Desktop/lab6$ python3 q1_client.py
Enter the text:abracadabra
Host IP address: 127.0.0.1
Success connecting to the server!
Sending text to the server ...
Waiting for result from the server ...
Result from the server is: 01100001 01100010 01110010 01100001 01100011 01100001 01100100 01100001 01100
010 01110010 01100001
jeremiah@jeremiah-Virtual-Machine:~/Desktop/lab6$
```

Server Side:-

```
jeremiah@jeremiah-Virtual-Machine:~/Desktop/lab6$ python3 q1_server.py
Waiting for client to send data ...
Connection successful!
From Client: abracadabra
Modified data: 01100001 01100010 01110010 01100001 01100011 01100001 01100100 01100001 01100010 01110010
 01100001
Finished sending to client!
jeremiah@jeremiah-Virtual-Machine:~/Desktop/lab6$
```

Observations and Inferences:-

- Since it is TCP, a connection must be established before sending/receiving data and the outputs above stay true to that as you can see that first, a connection is established between the server and client, following which a text message is sent for processing.
- Also, the client sends his text across to the server which is processed by a strToBinary() fn. This is a simple illustration of the client-server paradigm with a backend where all the backend code is private and secured.
- The client has no access to this function strToBinary() but can only make use of the platform that the server provides and in the way the server wants.
- Thus, the server logic (could be metaphorically called "Intellectual Property") is safe, secured and abstracted from the client.

Question 2:-

2) The Binary Decoding Server: Write another server running on eth1 interface to do the following:

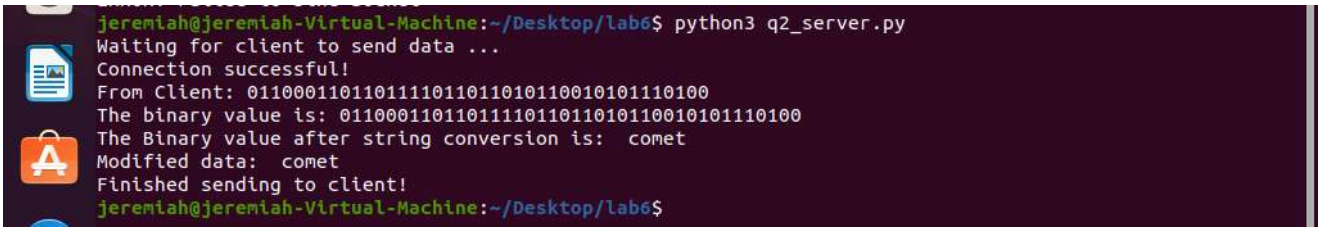
   a. Server returns the string value of a binary input. Example: the binary string "01100011 01101111 01101101 01101110 01100101 01110100 01110011 01101001 01101001" sent from the client should return "comnetsii"

Output:-

(Do Scroll Down)

Client Side:-

Server Side:-



Observations and Inferences:-

- To be honest, this is very closely linked to Q1. So, the findings for q1 are nearly the same here as well.
- However, in this program, the connecting interface is the ethernet interface which I've set using the [sudo ip address change dev] command to "13.3.3.3."
- With the knowledge I possess as of now and also from a high-level view, it is hard to say where the difference lies.
- One low-level difference might be the speed that they possess. Ethernet must obviously be much faster than the former as it's a physical connection.
- However, we would need some tools to measure the speed between sockets.

(Do Scroll Down)

Question 3:-

3) Write two client programs one for each of the server.

   a. What happens if the client aborts (e.g., when the input is CTRL/D)?

   b. Can you run two clients against the same server? Why or why not? Hint: Multithreaded socket server in Python

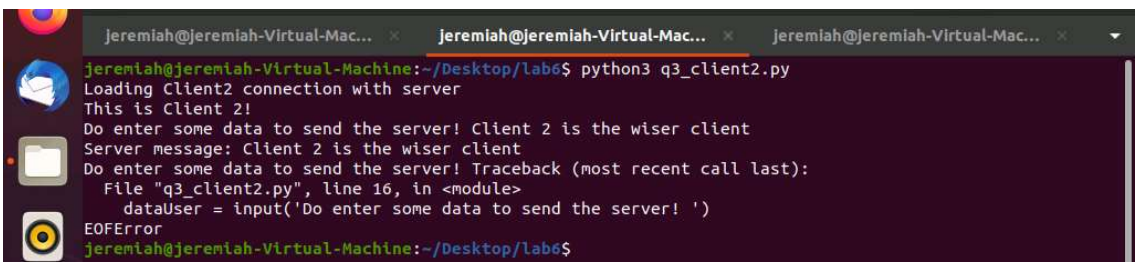
   c. What happens when you try to connect client1 to server2 by passing a wrong network address (e.g., connecting to localhost instead of eth1 IP)?

Output & Answer:-

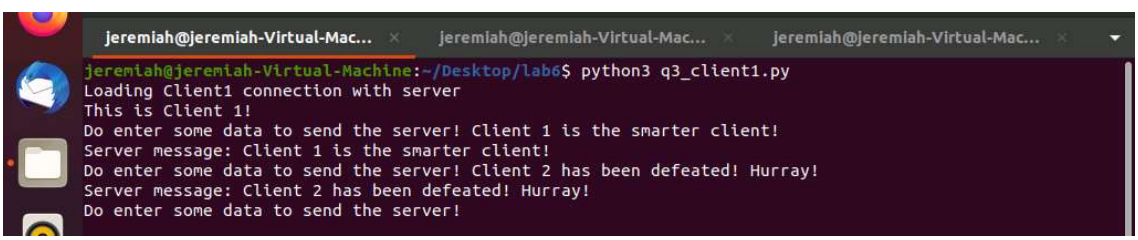
a)

To test this, I have taken Client 2 as the client to abort. This is what I observed: -

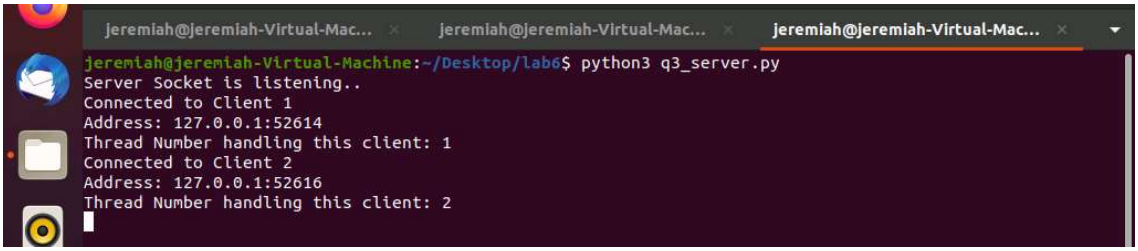
Client 2:-



Client 1:-



Server:-



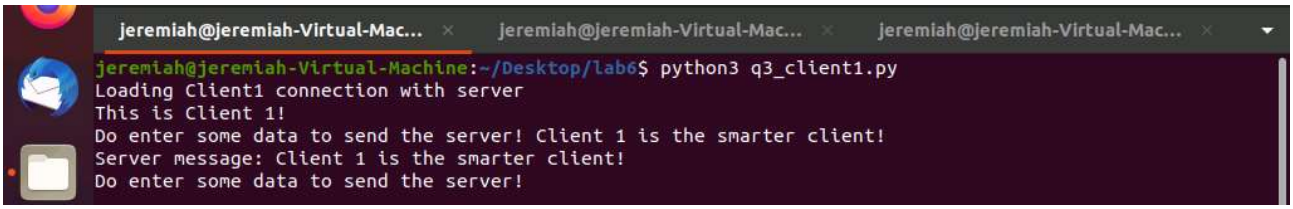
- Client 2 immediately breaks and loses its connection with the server.
- However, client1's connection is unaffected and functions smoothly as you can see. For instance, the data ->(Client 2 has been defeated) was sent after client 1 terminated and the server still echoes it back.
- Thus, because they're handled on 2 separate threads, when one fails, the other still works without a hassle.

---

b)

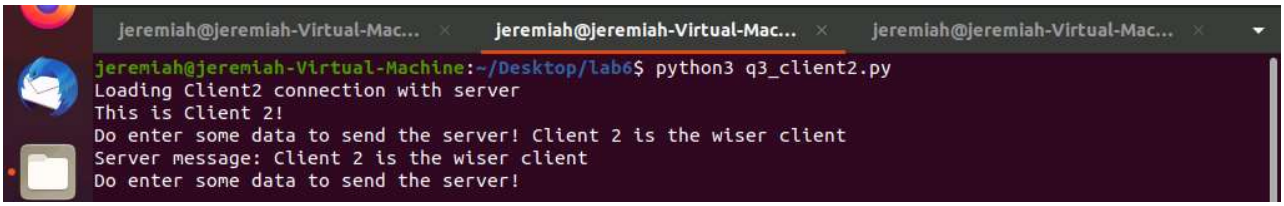
- Yes, you can run two clients against the same server. This is possible through the means of a concept called multithreading and in python, one can make use of Multithreaded Socket Servers to achieve the same.
- For instance, in my code, start_new_thread is used to spawn a new thread that deals with a client on a unique port with the function passed in its argument, which in this is : multiThreaded(), a fn that essentially servers to echo data from client.

Client1 Side:-

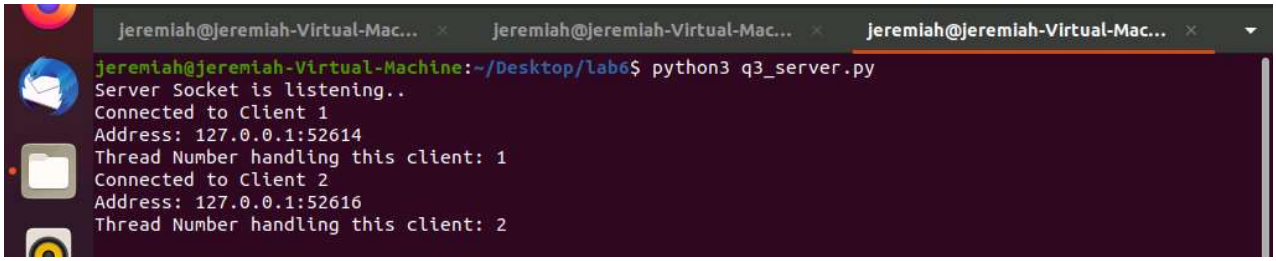


(Do Scroll Down)

Client2 Side:-



Server Side:-



- As you can see vividly, Client 1 is the first to make a request of connection to the server. The server immediately gives Client 1 a unique address with a specific port : 52614.
- Furthermore, Client 1 is handled on a new thread called, Thread 1.
- Then, sometime later Client 2 makes a request and the server spawns a new thread(Thread 2) to deal with Client 2 and this thread operates simultaneously at par with Thread 1.
- Also, Server 1 assigns a specific port to client 2 as well :52616.

C)

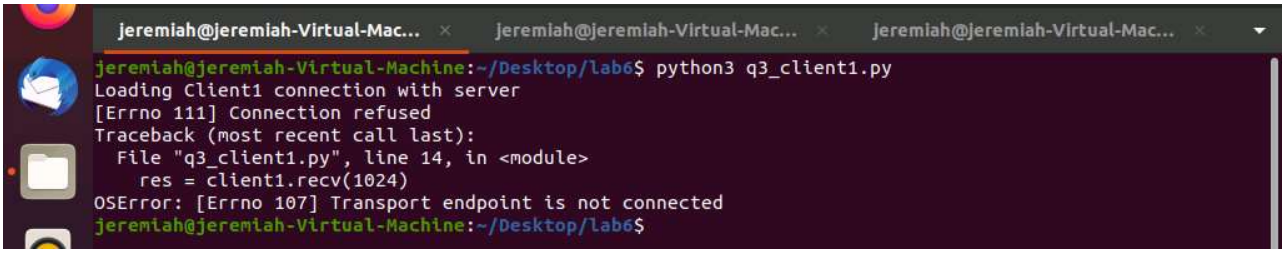
Change in Code To Test 3(c):-

```
L4: eth = '13.3.3.3'
L9:  client1.connect((eth, port))
```

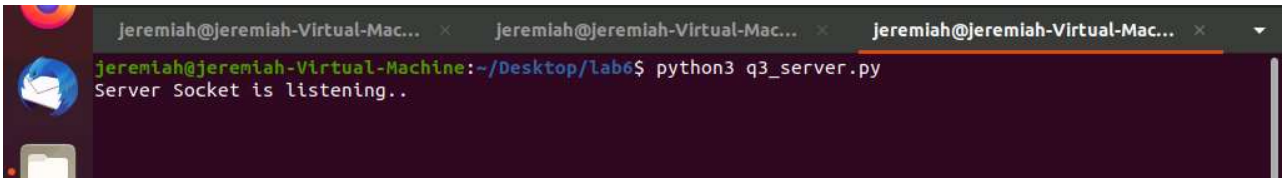
(Do Scroll Down)

Output:-

Client1 Side:-



Server Side:-



- Here as we can see client1 connects to “13.3.3.3” which is the ethernet interface IP in my environment. However, the server socket is based on localhost.
- Thus, there is a clash when client1 requests for a connection and that is why we see the Error, “Connection Refused” and “Transport endpoint is not Connected”
- The server remains idle as seen in the screenshot as it does not handle and its socket is not listening on the IP address, “13.3.3.3”
- To rectify this, we would need to either change the address the server socket is bound to the eth address or change the address from the client side to localhost.

THANK YOU!