CSLR52 – Networks Laboratory

Report: Assignment-1

Name: Saloni Rakholiya

Roll number: 106119109

Question 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**

Solution:

Explanation: We create a server that is based on localhost interface and port 7777, and bind the socket to this address. Once it is binded, it listens on this port for clients. Once the client is founded and server accepts the connection, the string is received, and then we encode the string and send it back to the client.

Code (converting text to binary in server):

```
#imports
import socket

#define constants
HOST='localhost'
PORT=7777

addr = (HOST, PORT)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
   s.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR |
socket.SO_REUSEPORT, 1)
   s.bind(addr)
   # print(s.getsockname())
   #listening for clients
   s.listen()
   print("Listening...")
   #connect to the client
   conn, addr = s.accept()
   print("Found a client...")
   #get data from server
   string = conn.recv(1024).decode()
   print("Received string from client "+str(string)+" ...")
   try:
       #encode the server data
       result = ' '.join(format(ord(x), 'b') for x in string)
       print("The encoded string is "+str(result))
   except:
       result = "The input string sent is invalid!"
       print("The input string sent is invalid!")
```

```
    print("Result sent!")
    #send result to client
    conn.sendall(result.encode())
```

Question 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"**

Solution:

Explanation: I have implemented the code in Linux and since I did not have physical NIC for ethernet interface, I created a virtual one. We create a virtual eth1 interface first, as this is the interface that we will use. We do this using the following commands on the Linux terminal:

$ sudo modprobe dummy
$ sudo lsmod | grep dummy
$ sudo ip link add eth1 type dummy
$ sudo ip address change dev eth1 10.0.0.10
$ ip address

If the last command shows the eth1 interface with IP address, it means the interface is successfully created. And then we get the IP address of this interface, and along with port 7777, we form the address and bind the socket with this address. We then listen for clients, and once the connection is found and accepted from the client side, we get a string from the server. This is in binary input form and we decode it back to the text string, if possible and send it back to the client.

After we are done with the program, we delete the interface with the following commands:
$ sudo ip link delete eth1 type dummy

Code (converting binary input to text):

```
# import libraries
import socket
import fcntl
import struct

# get the ip address of the interface
def interface_to_ip_address(interface_name):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,
        struct.pack('256s', bytes(interface_name[:15],'utf-8'))
    )[20:24])

# define constants
HOST=interface_to_ip_address('eth1')
PORT=7777

addr = (HOST, PORT)
```

```
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
   # s.setsockopt(socket.SOL_SOCKET, 25, ('eth1'+'\0').encode())
   s.bind(addr)
   s.listen()
   print("Listening...")
   conn, addr = s.accept()
   print("Found a client...")
   string = conn.recv(1024).decode()
   print("Received input from client "+str(string))
   try:
       result = ''.join(chr(int(x, 2)) for x in string.split())
       print("The decoded string is "+str(result))
   except:
       result = 'The input string sent is invalid!'
       print("The input string sent is invalid!")
   print("Bye!")
   conn.sendall(result.encode())
```

Question 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)?**

Solution:

Explanation: We create 2 client programs, one for the encoding server and one for the decoding server. The general procedure of creating a client is to know the interface and port and then connect the client to server address. Once connected we send the input and get back an output from the server and display it. Then the connection ends.

Encode-client code (text to binary):

```
#import libraries
import socket

#define constants
HOST='localhost'
PORT=7777

addr = (HOST,PORT)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
   # s1.setsockopt(socket.SOL_SOCKET, 25, ('eth1'+'\0').encode())
   #connect to server
   s.connect(addr)
   print("Connected to server...")
   string = input('Enter string to send to encoding server: ')
   #send string to server
   s.send(string.encode())
   #get result from server
```

```
    result = s.recv(1024).decode()
    print("Received from server: "+str(result))
```

Decode-client code (binary to text):

```
#import libraries
import socket
import fcntl
import struct

#convert interface name to ip address
def interface_to_ip_address(interface_name):
    s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    return socket.inet_ntoa(fcntl.ioctl(
        s.fileno(),
        0x8915,
        struct.pack('256s', bytes(interface_name[:15],'utf-8'))
    )[20:24])

#define constants
HOST=interface_to_ip_address('eth1')
PORT=7777

addr = (HOST,PORT)

with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    # s1.setsockopt(socket.SOL_SOCKET, 25, ('eth1'+'\0').encode())
    # connect to server
    s.connect(addr)
    print("Connected to server...")
    string = input('Enter binary input to send to the decoding server:
')
    #send string to server
    s.send(string.encode())
    #get data from server
    result = s.recv(1024).decode()
    print("Received from server: "+str(result))
```

Output results:

Encoding server:

```
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$ python encode_server.py
Listening...
Found a client...
Received string from client comnetsii ...
The encoded string is 1100011 1101111 1101101 1101110 1100101 1110100 1110011 1101001 1101001
Result sent!
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$
```

Encoding client:

```
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$ python encode_client.py
Connected to server...
Enter string to send to encoding server: comnetsii
Received from server: 1100011 1101111 1101101 1101110 1100101 1110100 1110011 1101001 1101001
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$
```

Decoding Server:

```
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$ python decode_server.py
Listening...
Found a client...
Received input from client 01100011 01101111 01101101 01101110 01100101 01110100 01110011 01101001
01101001
The decoded string is comnetsii
Bye!
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$
```

Decoding client:

```
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$ python decode_client.py
Connected to server...
Enter binary input to send to the decoding server: 01100011 01101111 01101101 01101110 01100101 011
10100 01110011 01101001 01101001
Received from server: comnetsii
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$
```

Answer a. CTRL/D is the command to abort and the client process gets killed but server is active. When we press CTRL/D is pressed on client side, the client is aborted, and there is an interrupt. It shows EOFError on the terminal and it essentially still sends a message whatever exists on the terminal to the server but it is empty, sending 0 characters to the server. This stops the server and aborts the client too, as our server handles only 1 client. EOFError is raised when there is an end of file encountered without any data being read, similar to our case, where there is no data but EOF is encountered.

```
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$ python decode_client.py
Connected to server...
Enter binary input to send to the decoding server: Traceback (most recent call last):
  File "decode_client.py", line 26, in <module>
    string = input('Enter binary input to send to the decoding server: ')
EOFError
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$
```

```
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$ python decode_server.py
Listening...
Found a client...
Received input from client
The decoded string is
Bye!
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$
```

Answer b. Yes, we can run two or even more clients against one server using multithreading. Thread depicts a lightweight process. Threads are cheaper in terms of memory overhead. Multithreading refers to implementing multiple threads at the same time in a process. There are python libraries used for creating multithreaded servers. Using multithreading in python, we can connect multiple clients to a server at the same time, by creating threads. One example is written by me below, which is the code for multithreaded server with 2 clients which are connected to it simultaneously. The server is able to handle both because of multiple threads. The server listens indefinitely for clients and we need to press CTRL/C to stop the server. We can also set a limit to the number of clients it can connect to, but I have implemented it for any number of servers in the below code.

Multithreaded server:

```
#import
import socket
from _thread import *
```

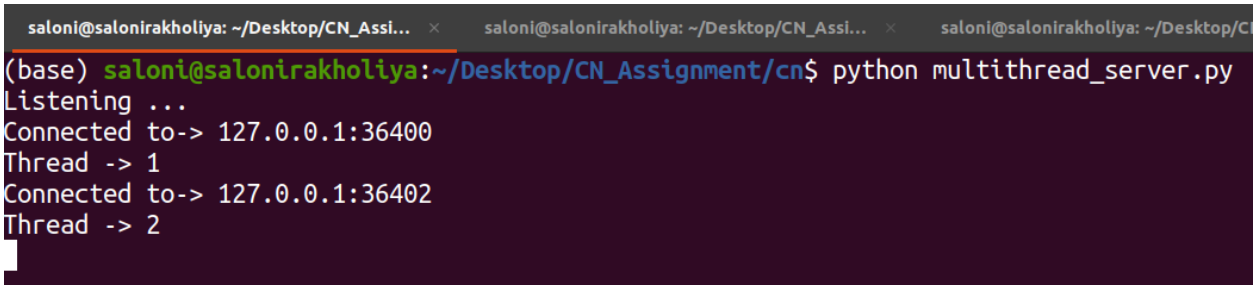
```
server_socket = socket.socket()

#define constants
host = '127.0.0.1'
port = 7777
thread_no = 0

#binding server to port
try:
   server_socket.bind((host, port))
except socket.error as err:
   print(str(err))

print('Listening ...')
server_socket.listen(2)

def new_threading_client(connection):
   while True:
       data = connection.recv(2048)
       server_msg =data.decode('utf-8')
       if not data:
           break
       connection.sendall(str.encode(server_msg))
   connection.close()

# keep accepting clients and threading
while True:
   Client, address = server_socket.accept()
   print('Connected to-> ' + address[0] + ':' + str(address[1]))
   #start thread for client connected
   start_new_thread(new_threading_client, (Client, ))
   thread_no+=1
   print('Thread -> ' + str(thread_no))
server_socket.close()
```

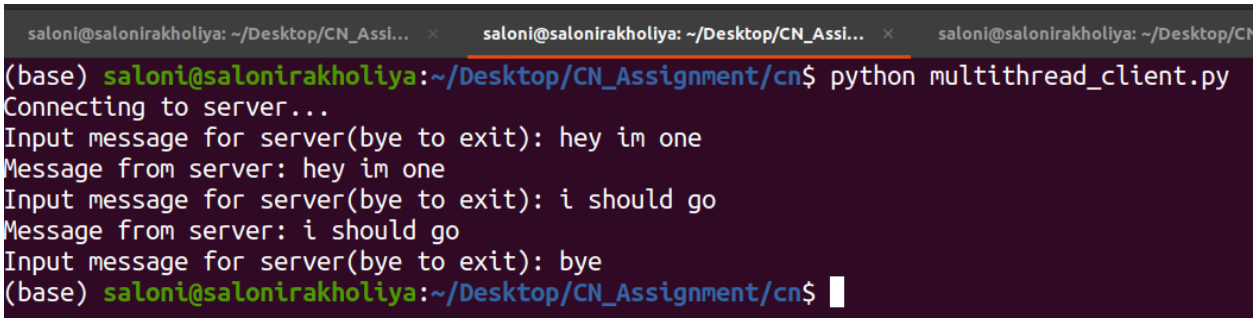
Client(run multiple times on different terminals):

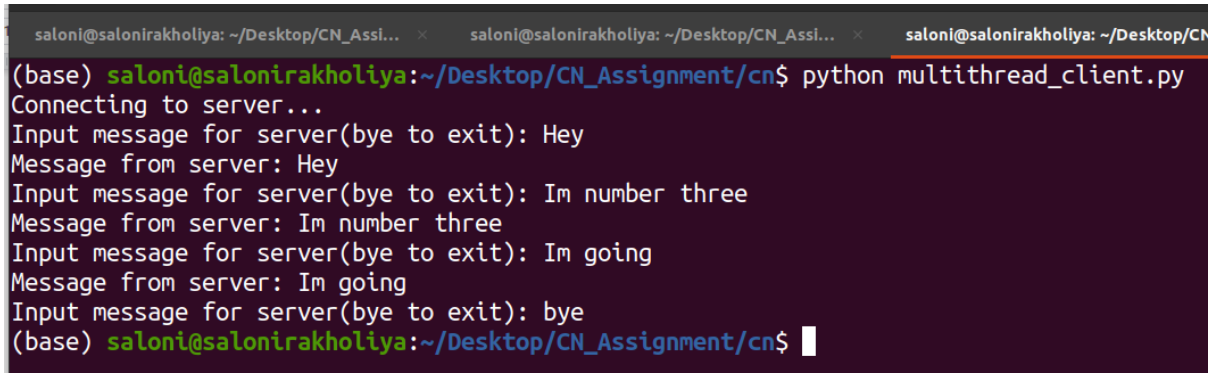
```
import socket

client_socket = socket.socket()
host = '127.0.0.1'
port = 7777

print('Connecting to server...')
try:
   client_socket.connect((host, port))
except socket.error as err:
   print(str(err))

while True:
   our_string = input('Input message for server(bye to exit): ')
   client_socket.send(str.encode(our_string))
   if our_string=="bye":
       break
   Response = client_socket.recv(1024)
```

```
    print("Message from server: "+str(Response.decode('utf-8')))

client_socket.close()
```

Outputs:







Answer c. When we try connecting client1 to server 2 by passing wrong network address, client does not connect to the server. Even if we compulsorily use same IP of server2, then also the communication occurs in different format and thus it crashes. We get a ConnectionRefusedError on the client side while the server continues listening for the clients. It is an exception raised when a connection that was attempted is declined by the server. Thus the error is encountered at the **s.connect(addr)** line. The address is different and interfaces too hence the connection is refused.

```
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$ python encode_client.py
Traceback (most recent call last):
  File "encode_client.py", line 13, in <module>
    s.connect(addr)
ConnectionRefusedError: [Errno 111] Connection refused
(base) saloni@salonirakholiya:~/Desktop/CN_Assignment/cn$
```