COMPUTER NETWORKS LABORATORY – UE19CS255

WEEK5 – Simple Client-Server Application using Network Socket Programming

Date : 27/2/2021

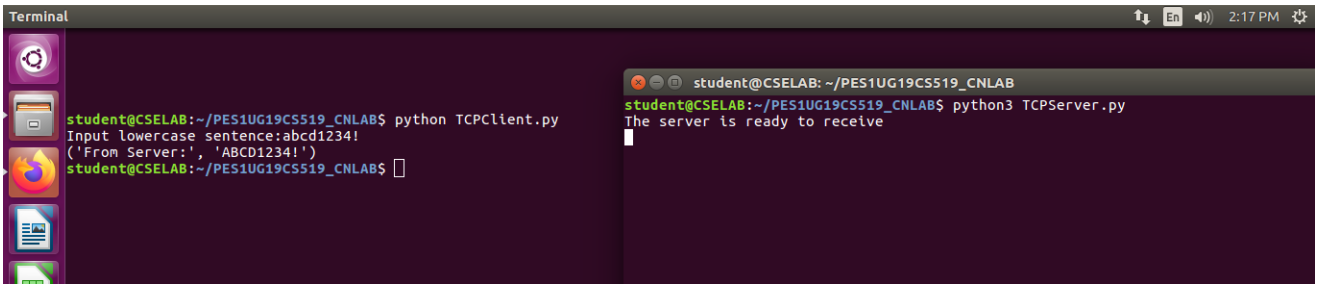
Name	SRN	Section
Sumukh Raju Bhat	PES1UG19CS519	H

TASK 1:

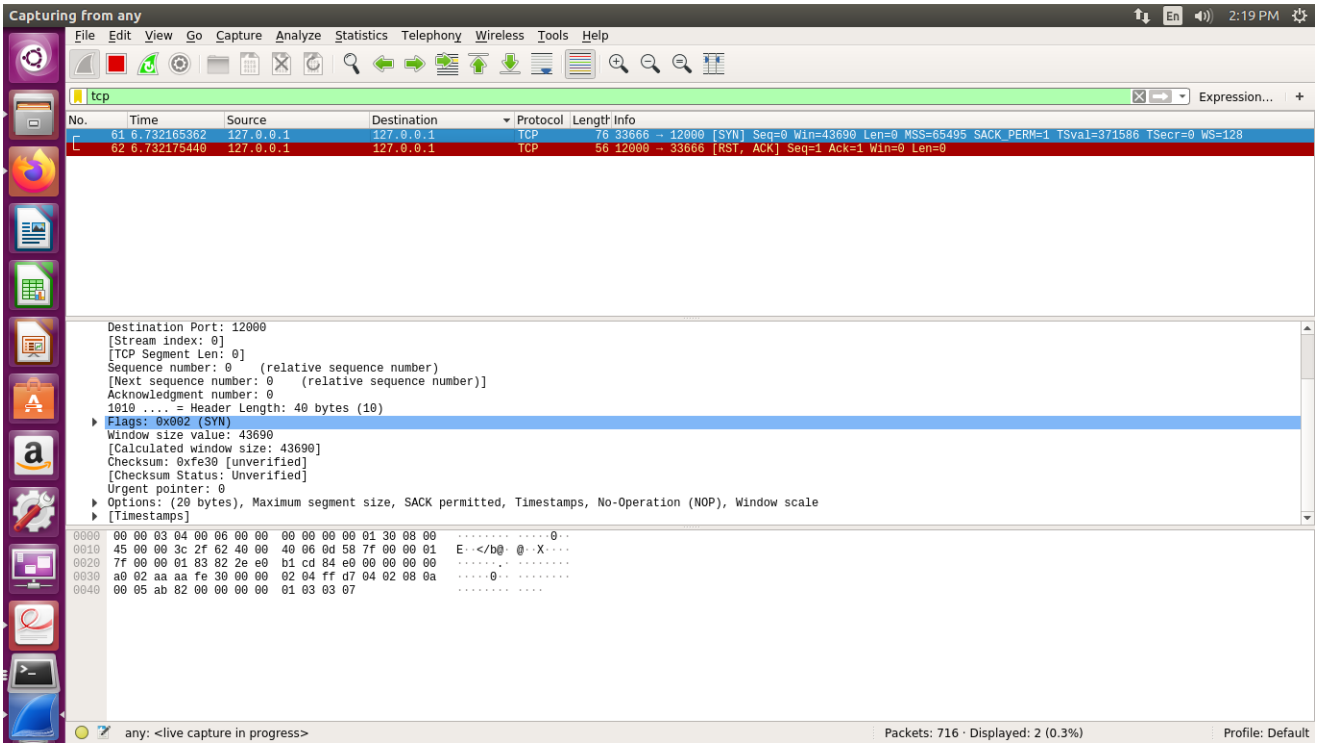
We set the servername/hostname as **localhost** and server script was run on one terminal and client on another.

TCP:

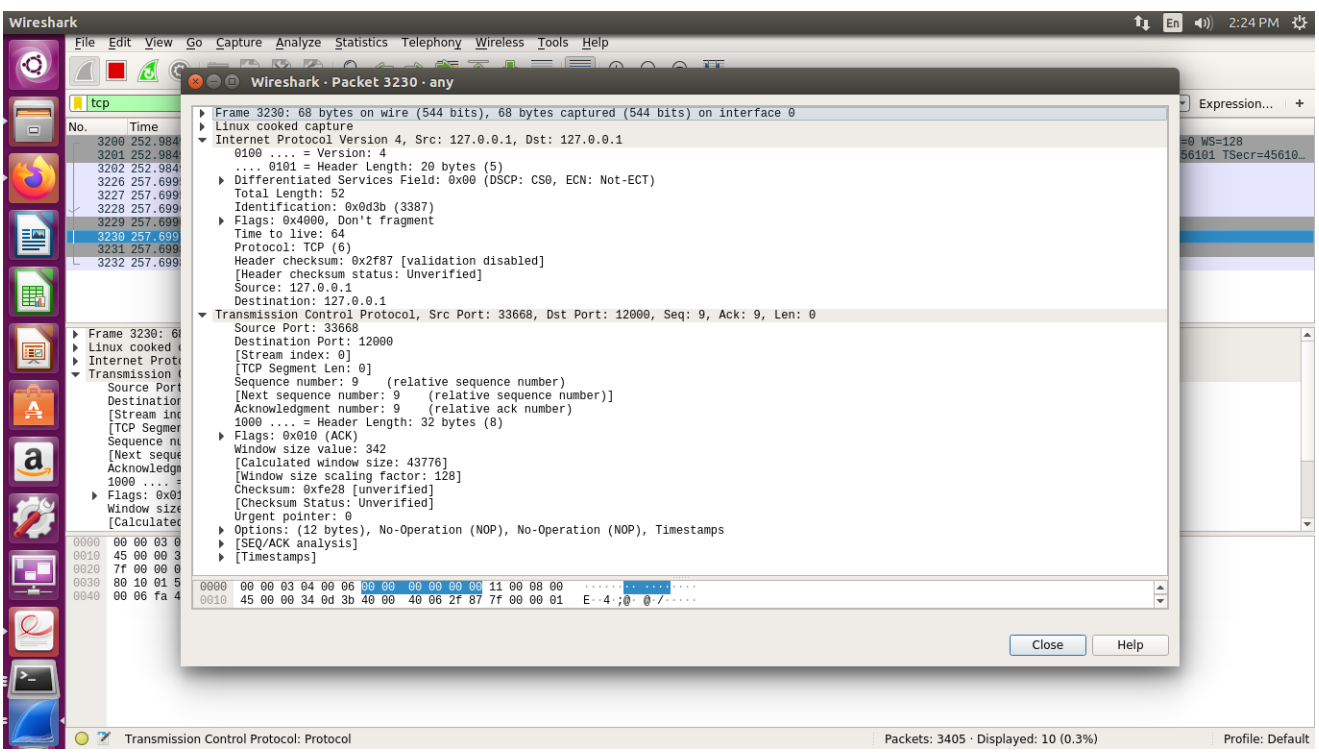
To create a TCP socket interface, we need to use **SOCK_STREAM** as type of socket of python socket library. Type of address is set as **AF_INET which is IPv4**. We use **bind**() to bind a host IP and port number.



When server script is run on one terminal and client script on another, the characters written on client side gets converted to uppercase in the server side and gets printed on client. Both in my case via **localhost(127.0.0.1) and port 12000.**



Request packet:



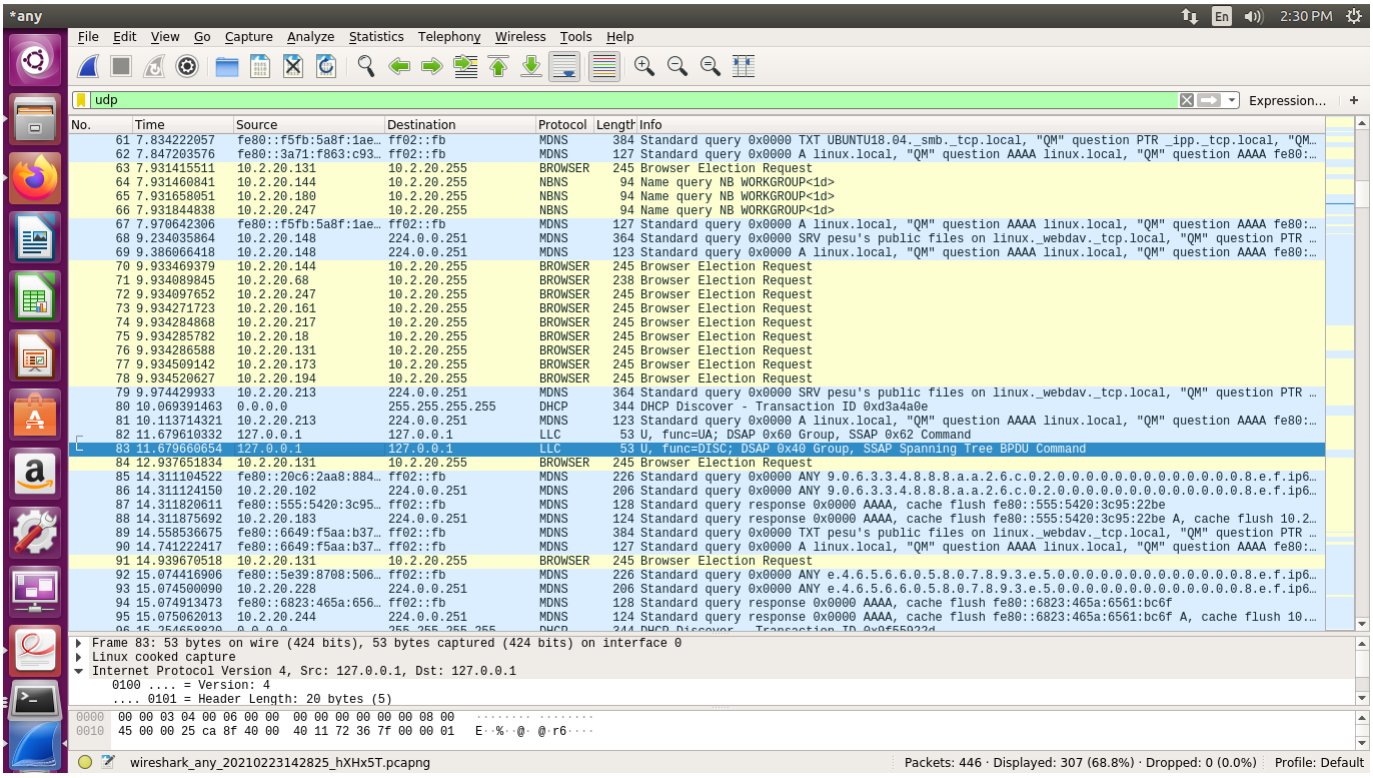
We can see both source and destination being 127.0.0.1(localhost), and serverport/dst port being 12000. TCP is being used. Similarly the response packet, but src port will be 12000 [As set in our python code as serverPort]

UDP:

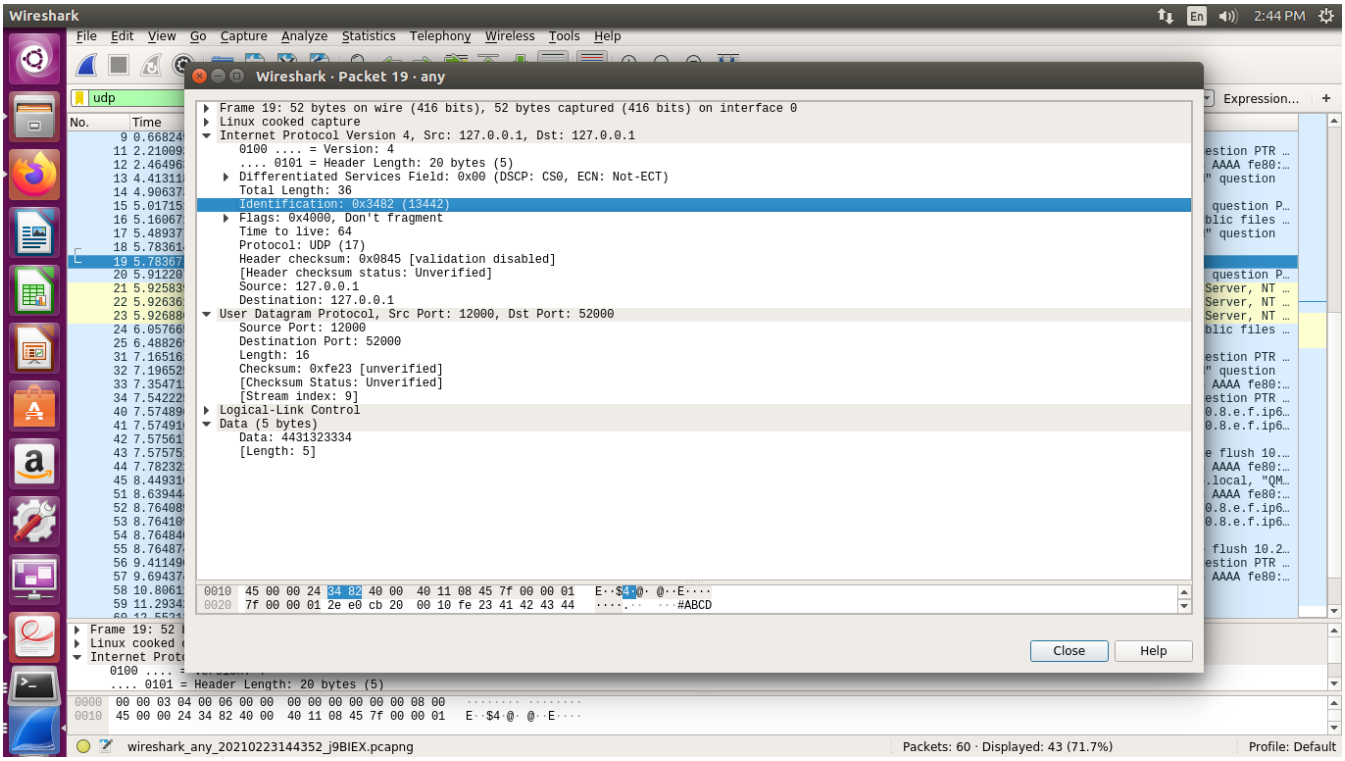
To create a **UDP** socket interface, we need to use **SOCK_DGRAM** as type of socket of python socket library. Type of address is set as **AF_INET** which is IPv4. We use **bind**() to bind a host IP and port number.

We set the servername/hostname as localhost and server script was run on one terminal and client on another.

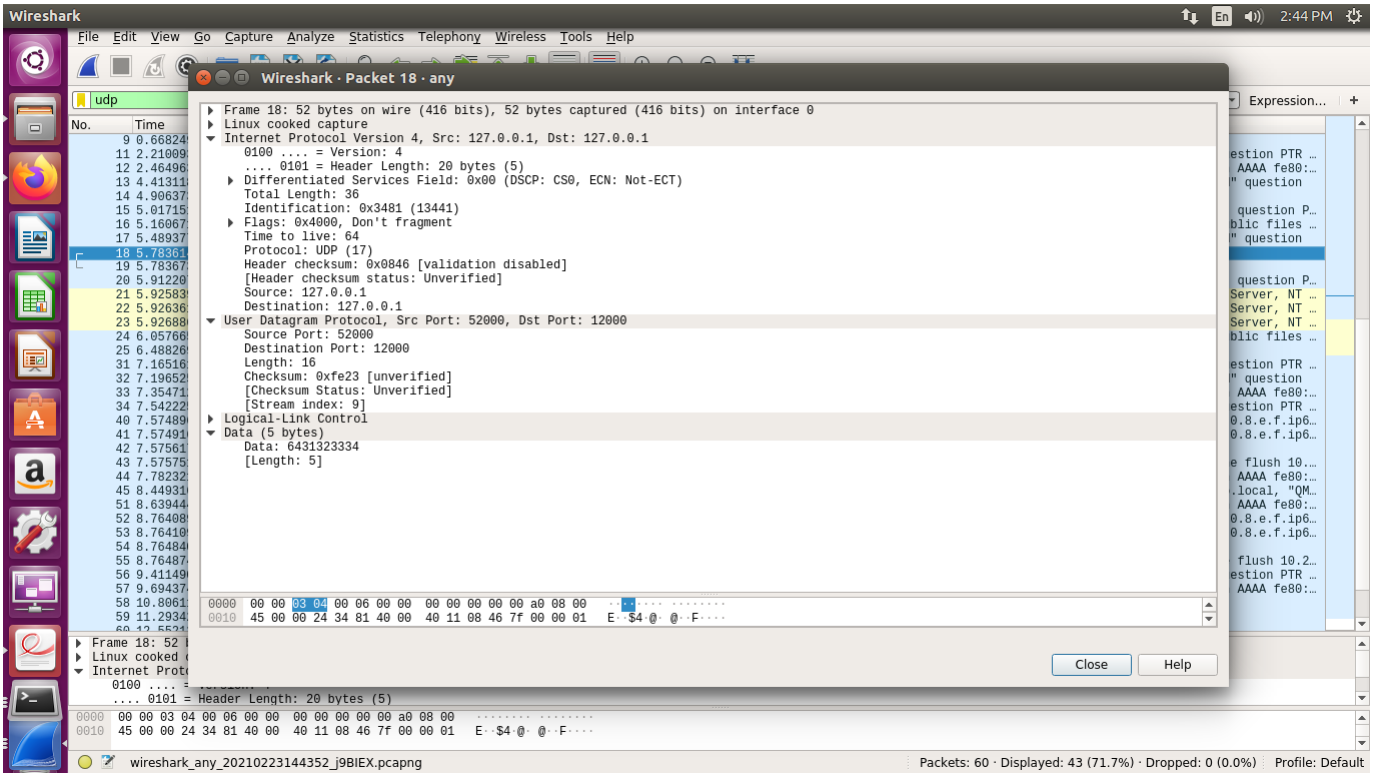
When server script is run on one terminal and client script on another, the characters written on client side gets converted to uppercase in the server side and gets printed on client. Both in my case via **localhost(127.0.0.1) and port 12000.**



Response packet:



Request packet:

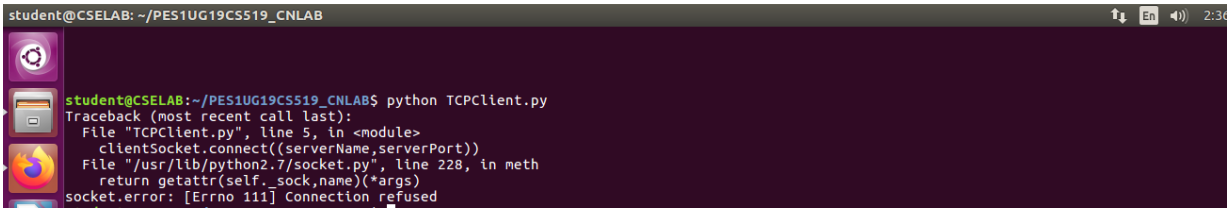


We can see both source and destination being 127.0.0.1(localhost). We can see that the dst port for request is 12000 and src port of response packet is 12000 as set in python code as serverPort. UDP is used and we can also see a data field in the packet inspection.

Problems:

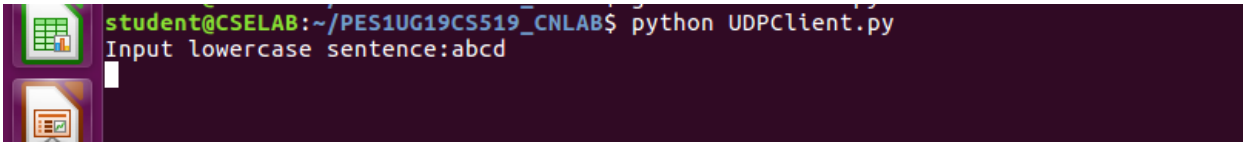
1. Suppose you run TCPClient before you run TCPServer. What happens? Why?

We get a **connection refused error**. A TCP connection can be established only when a host machine listens to a request for a client on a given IP address and port number. Then it accepts connections only at the specified address and port number.



2. Suppose you run UDPClient before you run UDPServer. What happens? Why?

There is **no error** as UDP is a **connectionless protocol** which is prone to data integrity issues like loss of packets. Hence here the packets are lost forever and we don't get any required response.



3. What happens if you use different port numbers for the client and server sides?

There is a **connection refused error** for TCP setup. The reason is same as Q1 and similar screenshot to Q1 is seen.

There is **no error** for UDP setup, but we don't get any response as there is no as such server mentioned in client and hence no connection was established and packets are lost forever. The reason is same as Q2 and similar screenshot to Q2 is seen.

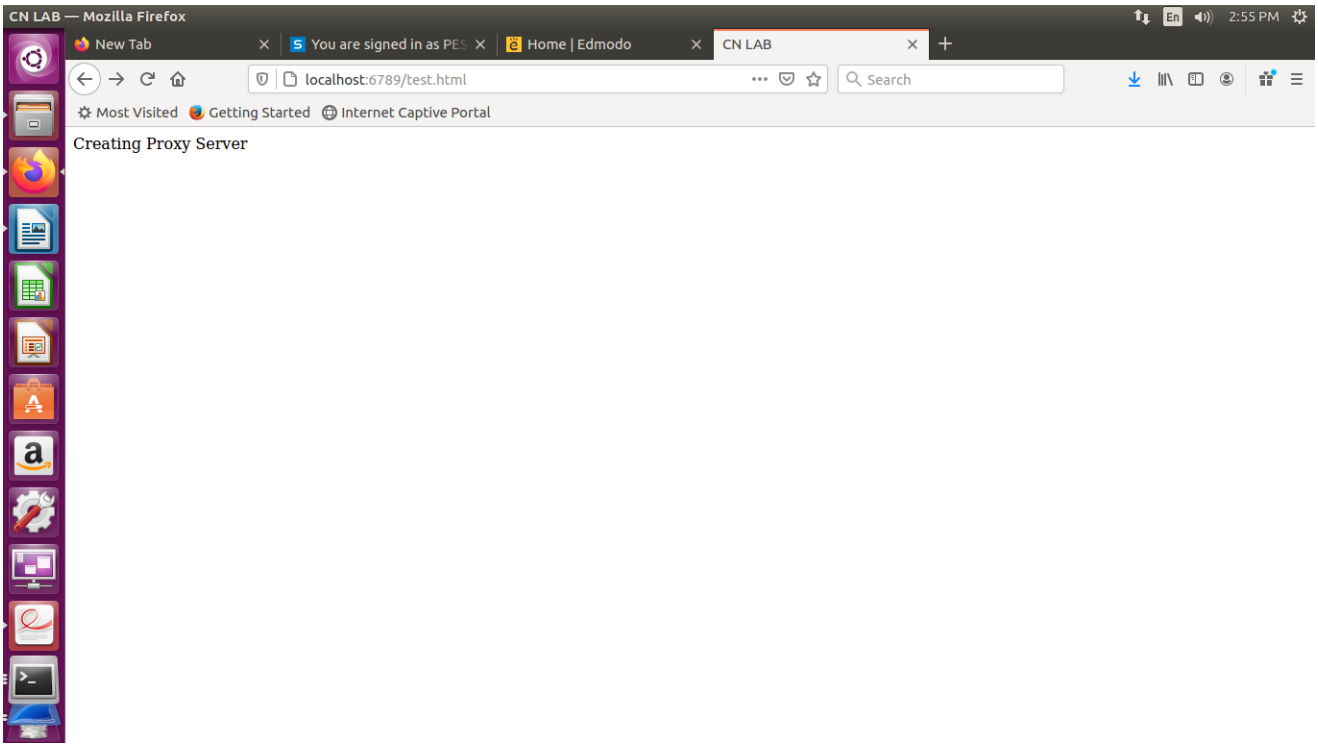
<u>Task2: Web Server</u>

We write a python script to setup a server by creating a **TCP** socket setup using python's socket module. We use **AF_INET for IPv4 use, SOCK_STREAM for TCP**. Then we use **localhost(127.0.0.1) as server IP and 6789 as portnumber** where the server should listen for requests from client. We use **bind()** to put together the port and host/servername and build a socket. Then we continiously listen to request using a **while true loop** which runs infinitely. Then we use a **try except block**. In the try block if we don't get any error try block will be

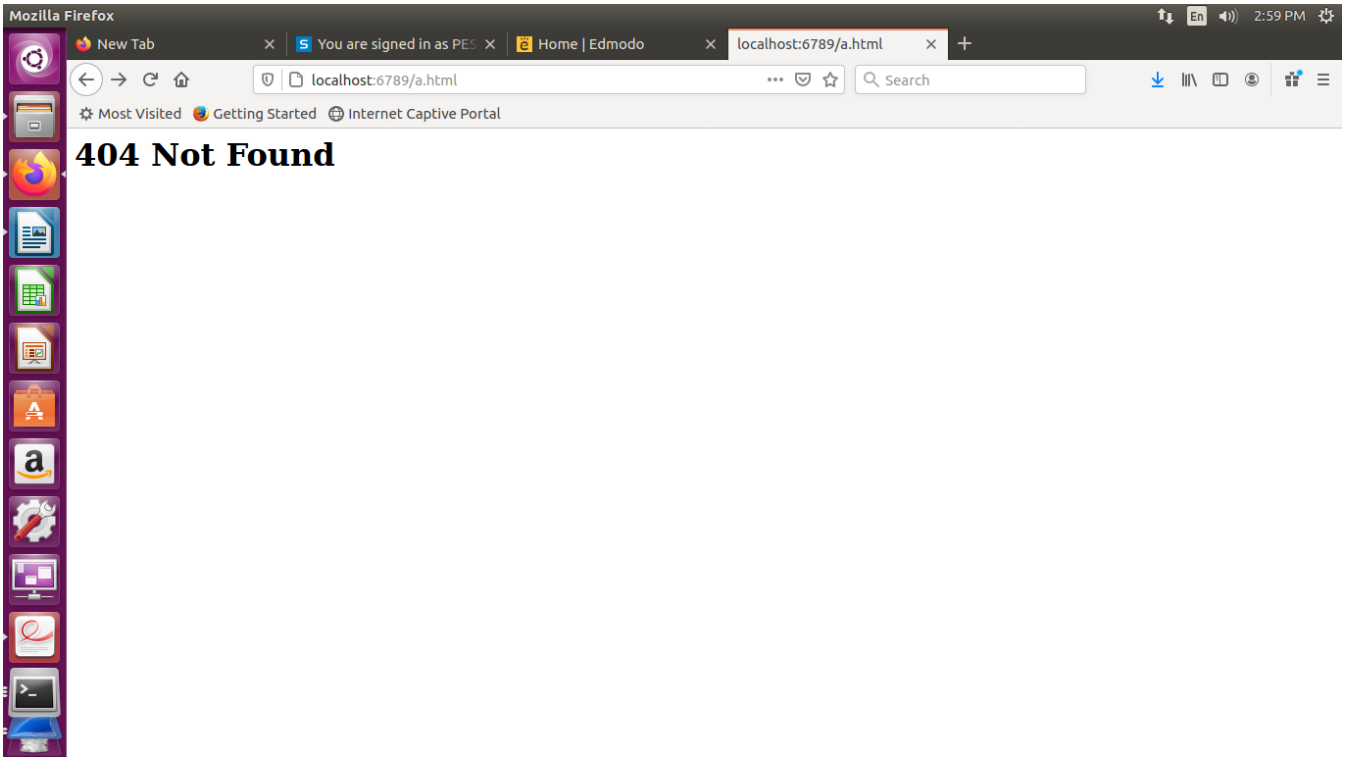
executed and we extract the path of the file requestion using **split()** function from the GET header's first line and open the file on the server side and render it with a success response of **"HTTP/1.1 200 OK\r\n\r\n"**. If we get any errors like file not found on the server etc, except block will be executed and we send a **"HTTP/1.1 404 Not Found\r\n\r\n"** response and render a html page with content as **"<html><head></head><body><h1>404 Not Found</h1></body></html>\r\n"**. Then again the server goes on listening mode until we interrupt.

When we print the request on the terminal we get the following:

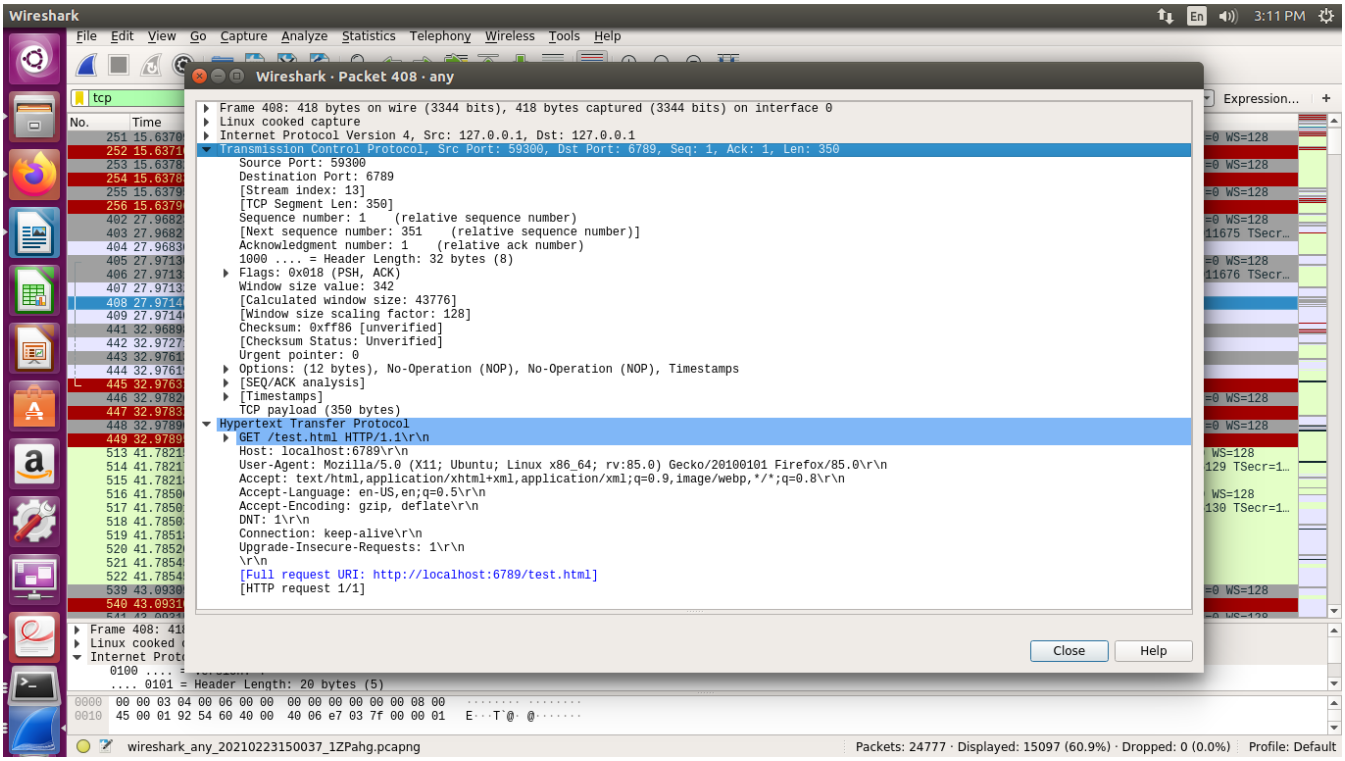
When we request for **test.html** with random content – **Creating Proxy Server** and title **CN Lab** which is present on the server side we get:



When we request a file **a.html** that is not there in the server side:



We capture the following get request packet on successful rendering on wireshark:



We see the **destination port** being **6789**(as it is a request packet) which we have set and it used **TCP** transmission. We see the IP to be **127.0.0.1** which is localhost as set by us. Similarly we get a response packet with **200 OK** response with **6789** as **src port**.

We capture the following response packet on wireshark on unsuccessful request:

```
▾ Transmission Control Protocol, Src Port: 6789, Dst Port: 60258, Seq: 1, Ack: 432, Len: 26
    Source Port: 6789
    Destination Port: 60258
    [Stream index: 40]
    [TCP Segment Len: 26]
    Sequence number: 1    (relative sequence number)
    [Next sequence number: 27    (relative sequence number)]
    Acknowledgment number: 432    (relative ack number)
    1000 .... = Header Length: 32 bytes (8)
  ▸ Flags: 0x018 (PSH, ACK)
    Window size value: 512
    [Calculated window size: 65536]
    [Window size scaling factor: 128]
    Checksum: 0x3ed8 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▸ [SEQ/ACK analysis]
  ▸ [Timestamps]
    TCP payload (26 bytes)
▾ Hypertext Transfer Protocol
  ▾ HTTP/1.1 404 Not Found\r\n
    ▸ [Expert Info (Chat/Sequence): HTTP/1.1 404 Not Found\r\n]
      Response Version: HTTP/1.1
      Status Code: 404
      [Status Code Description: Not Found]
      Response Phrase: Not Found
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.000087285 seconds]
    [Request in frame: 1099]
    [Request URI: http://10.2.21.74:6789/hi.html]
```

We see the **source port** being **6789**(as it is a response packet) which we have set and it used TCP transmission. We get a **404 not found** response as we requested for a file ./a.html which isn't present on the server. Similarly we get a request packet with **6789** as **dst port**.