WEEK 5 – CN LAB

NAME: - DIVYANSHU SHARMA

PES1UG20CS806

Simple Client-Server Application using Network Socket Programming

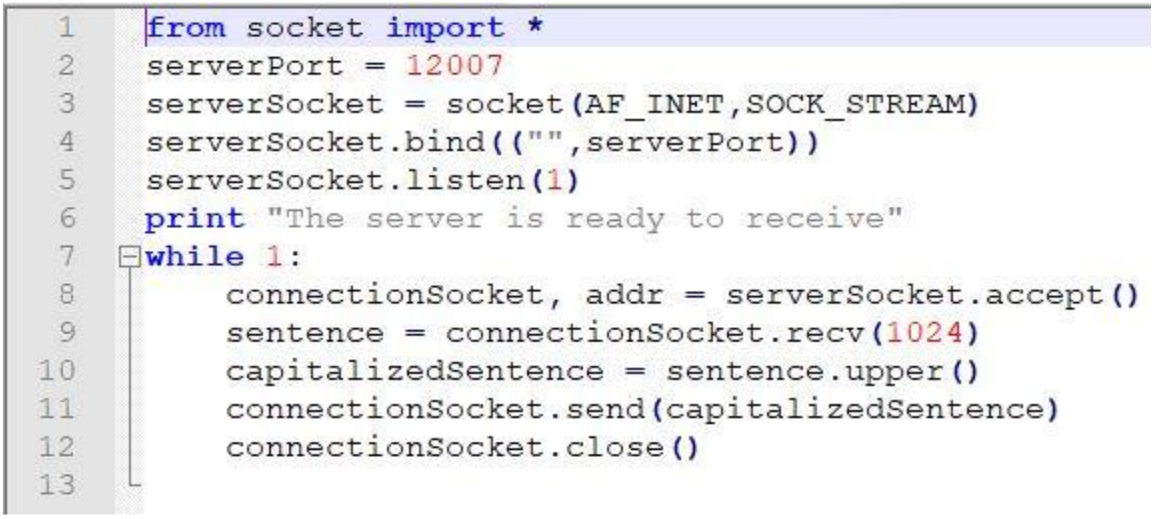
Task 1:

1. Create an application that will
   a. Convert lowercase letters to uppercase
      • e.g. [a…z] to [A…Z]
      • code will not change any special characters, e.g. &*!
   b. If the character is in uppercase, the program must not alter
2. Create Socket API both for client and server.
3. Must take the server address and port from the Command Line Interface (CLI).

1.1 TCP Connection

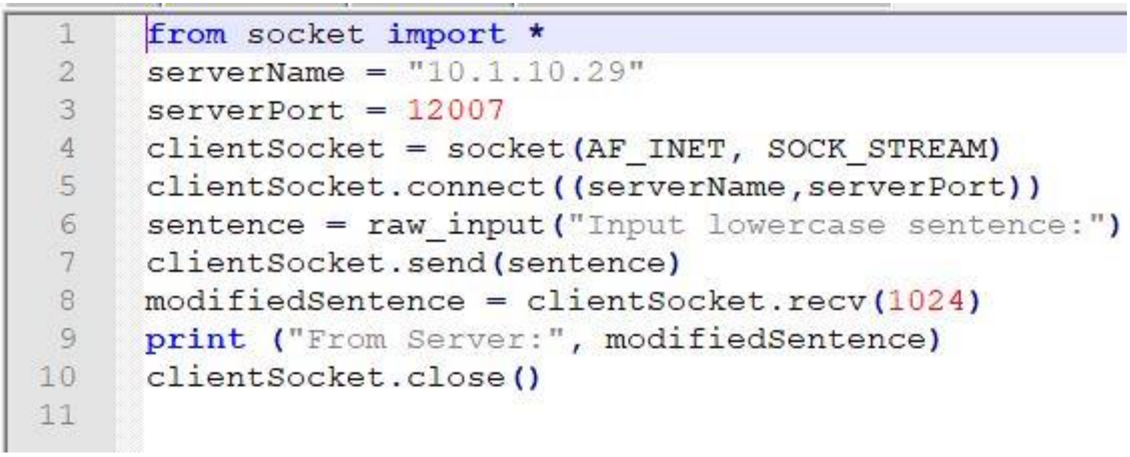
- A TCP connection can be made between two machines with the help of a socket interface using the socket library on Python.
- To create a TCP socket interface, the type of socket needs to be set as SOCK_STREAM.
- The type of addresses needs to be set as AF_INET which corresponds to IPv4.
- Once the server socket application is created, it needs to be hosted and hence needs to bind to a host IP and port number using the bind () function.
- Similarly, the client socket application needs to connect to a host using the IP address and port number.
- The socket can now listen for incoming connections as well as send messages to connected host machines.

1.1.1 TCP Server



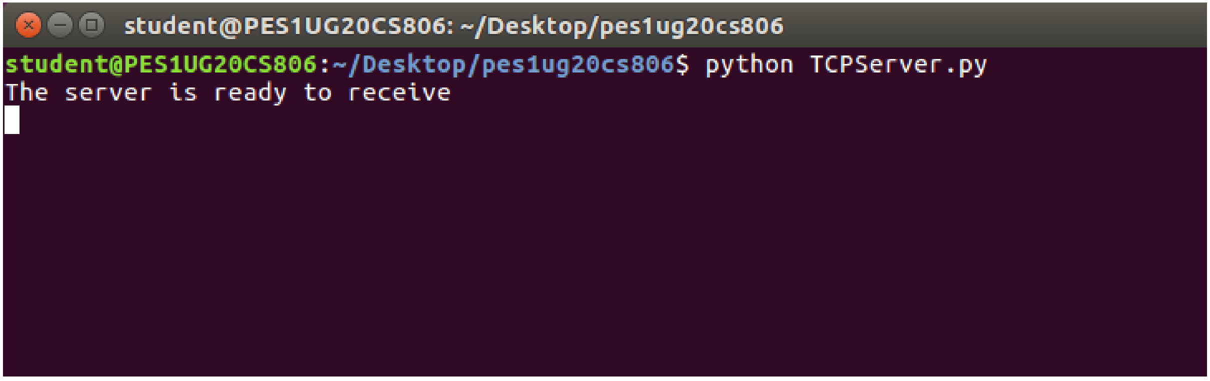
```
from socket import *
serverPort = 12007
serverSocket = socket(AF_INET,SOCK_STREAM)
serverSocket.bind(("",serverPort))
serverSocket.listen(1)
print "The server is ready to receive"
while 1:
    connectionSocket, addr = serverSocket.accept()
    sentence = connectionSocket.recv(1024)
    capitalizedSentence = sentence.upper()
    connectionSocket.send(capitalizedSentence)
    connectionSocket.close()

```

1.1.2 TCP Client



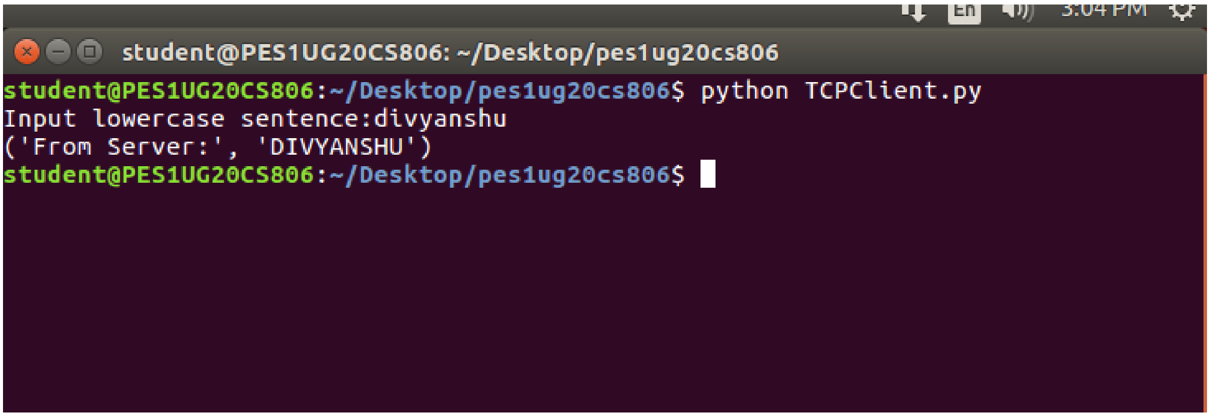
```
from socket import *
serverName = "10.1.10.29"
serverPort = 12007
clientSocket = socket(AF_INET, SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = raw_input("Input lowercase sentence:")
clientSocket.send(sentence)
modifiedSentence = clientSocket.recv(1024)
print ("From Server:", modifiedSentence)
clientSocket.close()
```

1.1.3 TCP Connection between Server and Client



```
student@PES1UG20CS806:~/Desktop/pes1ug20cs806$ python TCPServer.py
The server is ready to receive
```

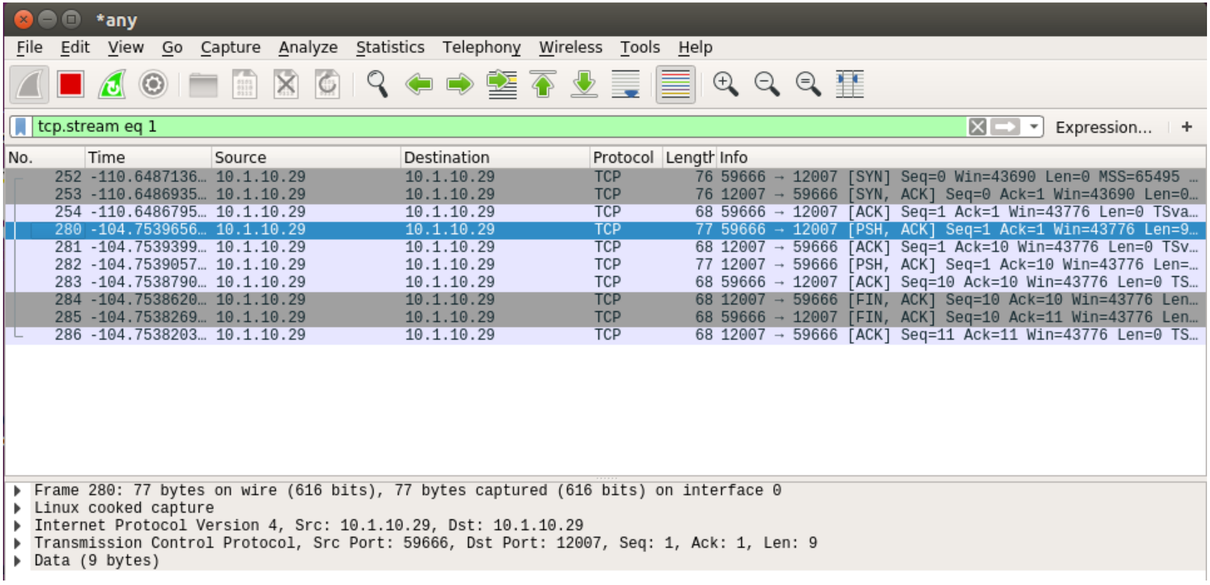
TCP Server



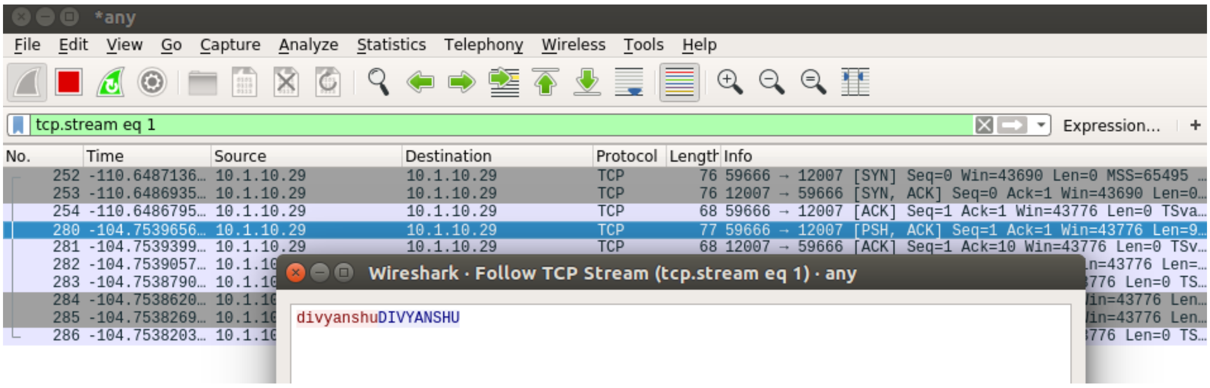
```
student@PES1UG20CS806:~/Desktop/pes1ug20cs806$ python TCPClient.py
Input lowercase sentence:divyanshu
('From Server:', 'DIVYANSHU')
student@PES1UG20CS806:~/Desktop/pes1ug20cs806$
```

TCP Client

1.1.4 Wireshark Capture for TCP Connection


*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression... +

```
tcp.stream eq 1

No.     Time            Source          Destination     Protocol Length Info
    252 -110.6487136…  10.1.10.29      10.1.10.29      TCP        76 59666 → 12007 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 …
    253 -110.6486935…  10.1.10.29      10.1.10.29      TCP        76 12007 → 59666 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0…
    254 -110.6486795…  10.1.10.29      10.1.10.29      TCP        68 59666 → 12007 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSva…
    280 -104.7539656…  10.1.10.29      10.1.10.29      TCP        77 59666 → 12007 [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=9…
    281 -104.7539399…  10.1.10.29      10.1.10.29      TCP        68 12007 → 59666 [ACK] Seq=1 Ack=10 Win=43776 Len=0 TSv…
    282 -104.7539057…  10.1.10.29      10.1.10.29      TCP        77 12007 → 59666 [PSH, ACK] Seq=1 Ack=10 Win=43776 Len=…
    283 -104.7538790…  10.1.10.29      10.1.10.29      TCP        68 59666 → 12007 [ACK] Seq=10 Ack=10 Win=43776 Len=0 TS…
    284 -104.7538620…  10.1.10.29      10.1.10.29      TCP        68 12007 → 59666 [FIN, ACK] Seq=10 Ack=10 Win=43776 Len…
    285 -104.7538269…  10.1.10.29      10.1.10.29      TCP        68 59666 → 12007 [FIN, ACK] Seq=10 Ack=11 Win=43776 Len…
    286 -104.7538203…  10.1.10.29      10.1.10.29      TCP        68 12007 → 59666 [ACK] Seq=11 Ack=11 Win=43776 Len=0 TS…

▸ Frame 280: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface 0
▸ Linux cooked capture
▸ Internet Protocol Version 4, Src: 10.1.10.29, Dst: 10.1.10.29
▸ Transmission Control Protocol, Src Port: 59666, Dst Port: 12007, Seq: 1, Ack: 1, Len: 9
▸ Data (9 bytes)
```

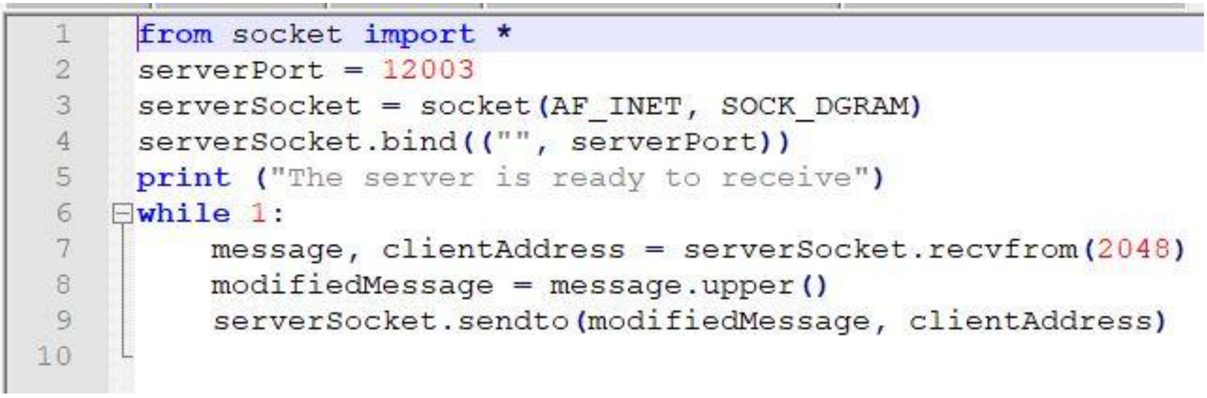

*any
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Expression... +

```
tcp.stream eq 1

No.     Time            Source          Destination     Protocol Length Info
    252 -110.6487136…  10.1.10.29      10.1.10.29      TCP        76 59666 → 12007 [SYN] Seq=0 Win=43690 Len=0 MSS=65495 …
    253 -110.6486935…  10.1.10.29      10.1.10.29      TCP        76 12007 → 59666 [SYN, ACK] Seq=0 Ack=1 Win=43690 Len=0…
    254 -110.6486795…  10.1.10.29      10.1.10.29      TCP        68 59666 → 12007 [ACK] Seq=1 Ack=1 Win=43776 Len=0 TSva…
    280 -104.7539656…  10.1.10.29      10.1.10.29      TCP        77 59666 → 12007 [PSH, ACK] Seq=1 Ack=1 Win=43776 Len=9…
    281 -104.7539399…  10.1.10.29      10.1.10.29      TCP        68 12007 → 59666 [ACK] Seq=1 Ack=10 Win=43776 Len=0 TSv…
    282 -104.7539057…  10.1.10                                                                             n=43776 Len=…
    283 -104.7538790…  10.1.10                                                                             776 Len=0 TS…
    284 -104.7538620…  10.1.10                                                                             in=43776 Len…
    285 -104.7538269…  10.1.10                                                                             in=43776 Len…
    286 -104.7538203…  10.1.10                                                                             776 Len=0 TS…

Wireshark · Follow TCP Stream (tcp.stream eq 1) · any

divyanshuDIVYANSHU
```

1.2 UDP Connection

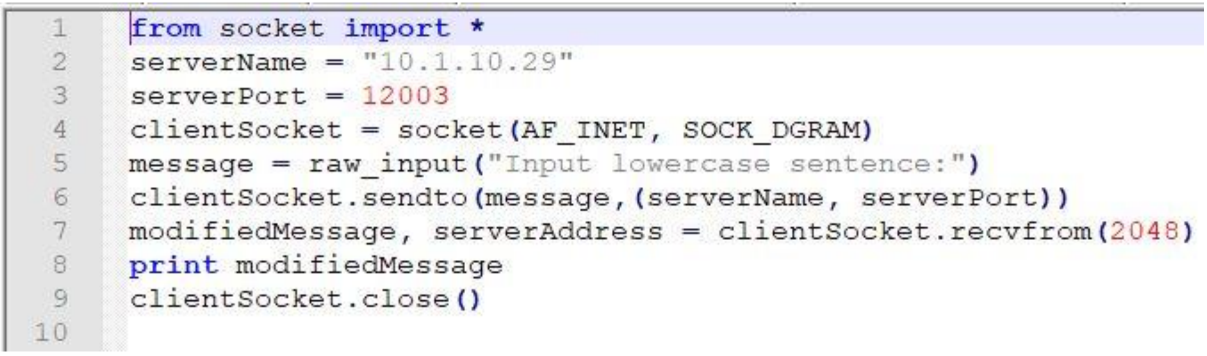
- A UDP connection can be made between two machines with the help of a socket interface using the socket library on Python.
- A UDP connection can be made between two machines with the help of a socket interface using the socket library on Python3.
- The type of addresses needs to be set as AF_INET which corresponds to IPv4.
- Once the server socket application is created, it needs to be hosted and hence needs to bind to a host IP and port number using the bind () function.
- Similarly, the client socket application needs to connect to a host using the IP address and port number.
- The socket can now listen for incoming connections as well as send messages to connected host machines.

1.2.1 UDP Server



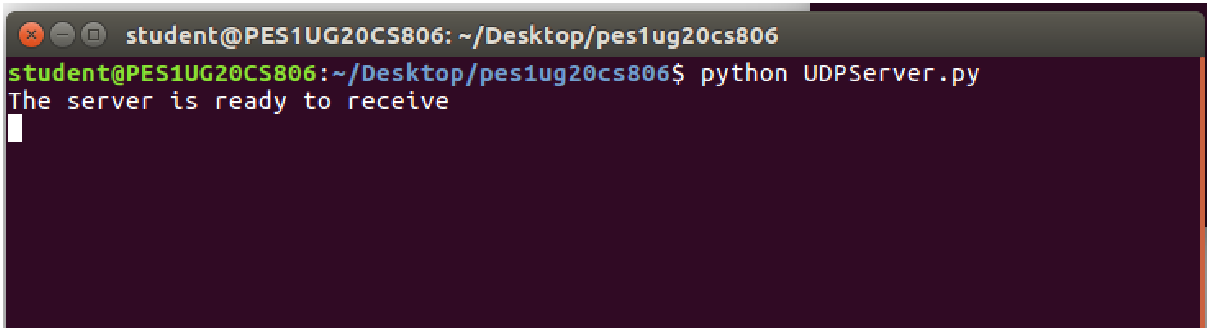
```python
from socket import *
serverPort = 12003
serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(("", serverPort))
print ("The server is ready to receive")
while 1:
    message, clientAddress = serverSocket.recvfrom(2048)
    modifiedMessage = message.upper()
    serverSocket.sendto(modifiedMessage, clientAddress)

```

1.2.2 UDP Client



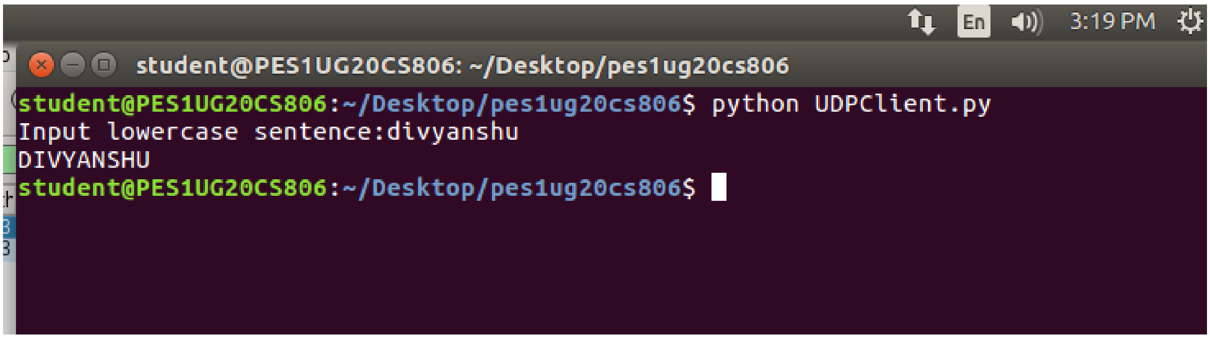
```python
from socket import *
serverName = "10.1.10.29"
serverPort = 12003
clientSocket = socket(AF_INET, SOCK_DGRAM)
message = raw_input("Input lowercase sentence:")
clientSocket.sendto(message,(serverName, serverPort))
modifiedMessage, serverAddress = clientSocket.recvfrom(2048)
print modifiedMessage
clientSocket.close()

```

1.2.3 UDP Connection between Server and Client



```
student@PES1UG20CS806:~/Desktop/pes1ug20cs806$ python UDPServer.py
The server is ready to receive
```

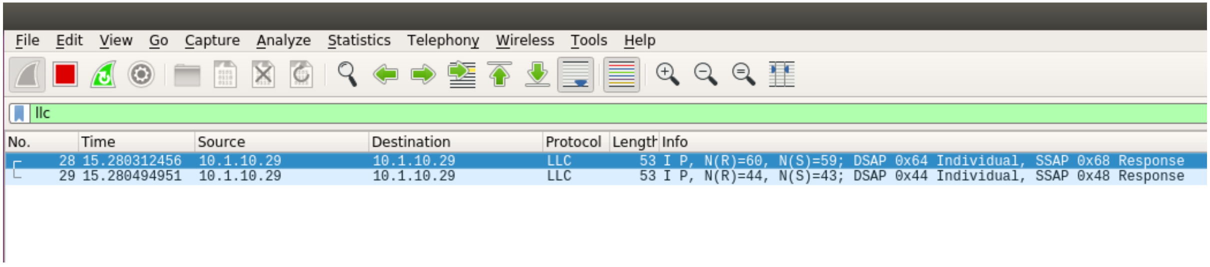
UDP Server



```
student@PES1UG20CS806:~/Desktop/pes1ug20cs806$ python UDPClient.py
Input lowercase sentence:divyanshu
DIVYANSHU
student@PES1UG20CS806:~/Desktop/pes1ug20cs806$ 
```

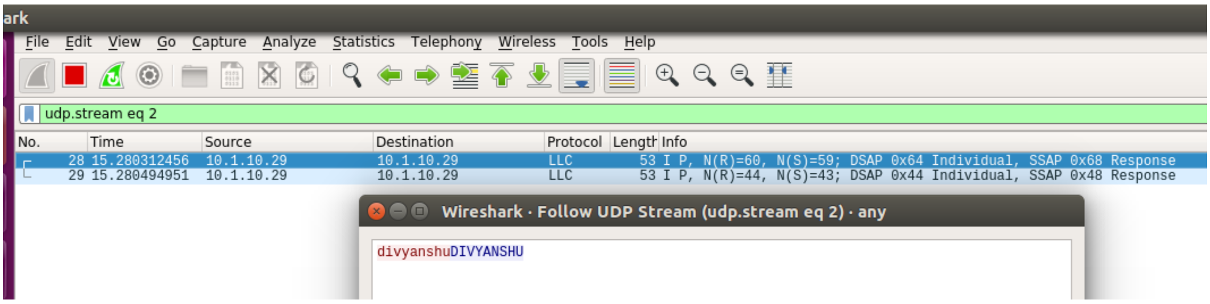
UDP Client

1.2.4 Wireshark Capture for UDP Connection


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

llc

No.	Time	Source	Destination	Protocol	Length	Info
28	15.280312456	10.1.10.29	10.1.10.29	LLC	53	I P, N(R)=60, N(S)=59; DSAP 0x64 Individual, SSAP 0x68 Response
29	15.280494951	10.1.10.29	10.1.10.29	LLC	53	I P, N(R)=44, N(S)=43; DSAP 0x44 Individual, SSAP 0x48 Response


ark
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

udp.stream eq 2

No.	Time	Source	Destination	Protocol	Length	Info
28	15.280312456	10.1.10.29	10.1.10.29	LLC	53	I P, N(R)=60, N(S)=59; DSAP 0x64 Individual, SSAP 0x68 Response
29	15.280494951	10.1.10.29	10.1.10.29	LLC	53	I P, N(R)=44, N(S)=43; DSAP 0x44 Individual, SSAP 0x48 Response

Wireshark · Follow UDP Stream (udp.stream eq 2) · any

```
divyanshuDIVYANSHU
```

1.3 Problems

Q1. ***Suppose you run TCPClient before you run TCPServer. What happens? Why?***

Answer: This will lead to a ***ConnectionRefusedError***, since the server socket application we are trying to connect has not been initiated and is not listening for connections on the given port number. Therefore, any connection requests sent by a client machine at that IP address and port number immediately fail since the connection gets refused. A TCP connection can be established between two socket interfaces only when a host machine listens to requests on a given IP address and port number and accepts connections made by another machine at the same address and port.

Q2. ***Suppose you run UDPClient before you run UDPServer. What happens? Why?***

Answer: No error will be obtained since UDP does not require a prior connection to be set up between the host machines for data transfer to begin. It is a connectionless protocol which transfers packets of data to a destination IP and port number without verifying the existence of the connection. Hence, it is prone to data integrity issues such as loss of packets. If any packets of data are sent before the server is executed, the packets are lost forever and will not reach the server socket application. However, if any packets of data are sent after the server is executed, the client will be able to send packets to a destination server and also receive response packets in return.

Q3. ***What happens if you use different port numbers for the client and server sides?***

Answer: This will lead to a ***ConnectionRefusedError*** for a TCP connection, since the server socket application we are trying to connect to is not listening for requests at the same port number as the one the client socket application is trying to connect with. Hence, the connection between the two socket interfaces is never setup and the connection are downright refused. However, on a UDP connection, since no prior connection is required to be established between the host machines for data transfer to take place, no error as such is obtained. Any messages sent by the client are lost since the destination server does not exists.

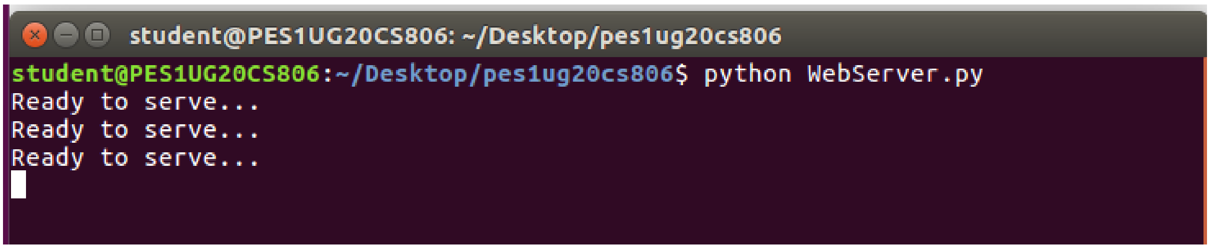
Task 2: Web Server

In this assignment, you will develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will

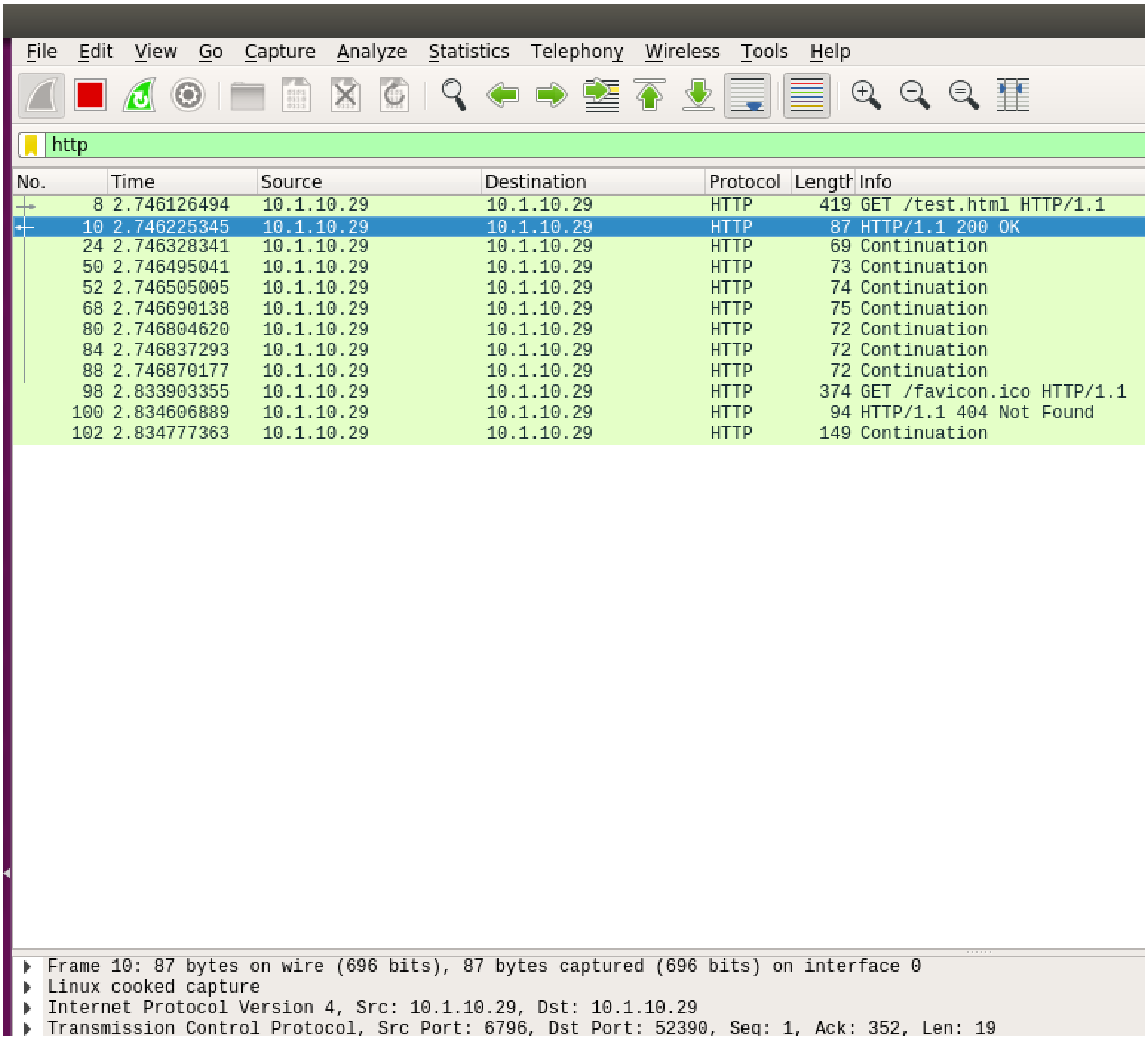
a) create a connection socket when contacted by a client (browser);
b) receive the HTTP request from this connection;
c) parse the request to determine the specific file being requested;
d) get the requested file from the server's file system;
e) create an HTTP response message consisting of the requested file preceded by header lines; and
f) 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.

For this assignment, the companion Web site provides the skeleton code for your server. Your job is to complete the code, run your server, and then test your server by sending requests from 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.

SUCCESSFULLY ACCESS THE .html FILE FROM THE SERVER



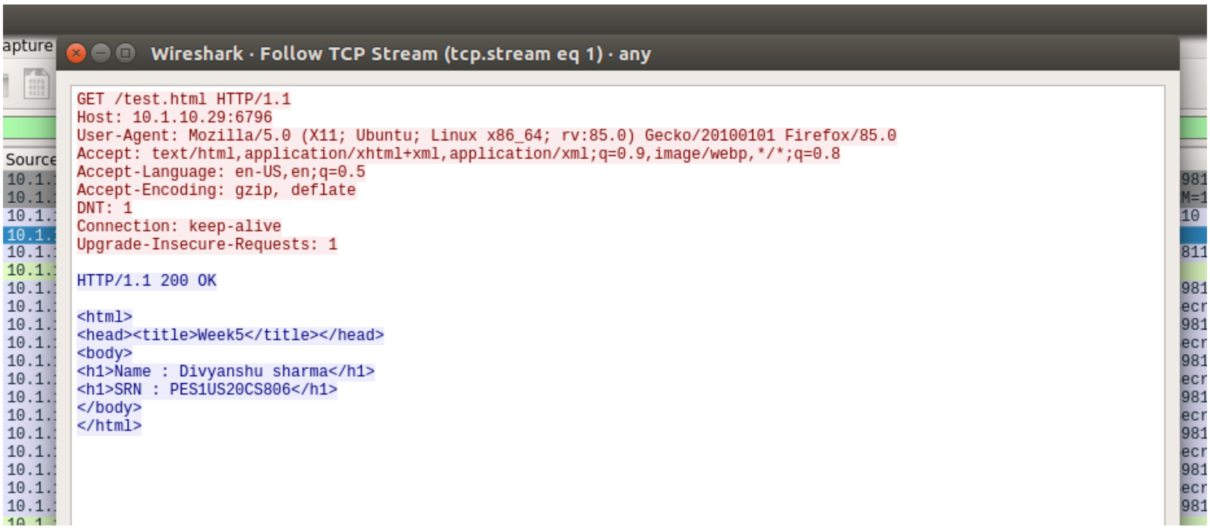
```
student@PES1UG20CS806:~/Desktop/pes1ug20cs806$ python WebServer.py
Ready to serve...
Ready to serve...
Ready to serve...
```


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
http

```
No.     Time           Source        Destination   Protocol Length Info
      8 2.746126494    10.1.10.29    10.1.10.29    HTTP      419 GET /test.html HTTP/1.1
     10 2.746225345    10.1.10.29    10.1.10.29    HTTP       87 HTTP/1.1 200 OK
     24 2.746328341    10.1.10.29    10.1.10.29    HTTP       69 Continuation
     50 2.746495041    10.1.10.29    10.1.10.29    HTTP       73 Continuation
     52 2.746505005    10.1.10.29    10.1.10.29    HTTP       74 Continuation
     68 2.746690138    10.1.10.29    10.1.10.29    HTTP       75 Continuation
     80 2.746804620    10.1.10.29    10.1.10.29    HTTP       72 Continuation
     84 2.746837293    10.1.10.29    10.1.10.29    HTTP       72 Continuation
     88 2.746870177    10.1.10.29    10.1.10.29    HTTP       72 Continuation
     98 2.833903355    10.1.10.29    10.1.10.29    HTTP      374 GET /favicon.ico HTTP/1.1
    100 2.834606889    10.1.10.29    10.1.10.29    HTTP       94 HTTP/1.1 404 Not Found
    102 2.834777363    10.1.10.29    10.1.10.29    HTTP      149 Continuation
```

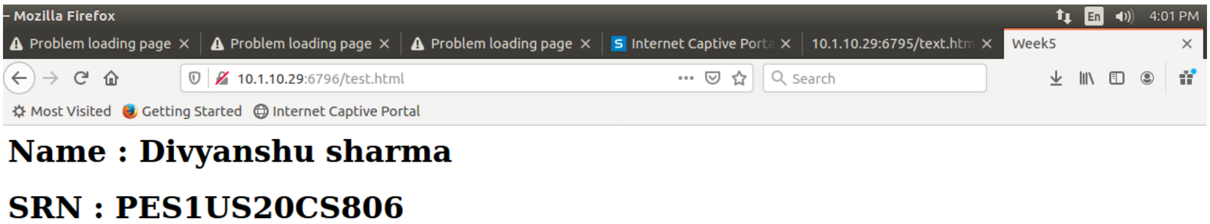
```
▶ Frame 10: 87 bytes on wire (696 bits), 87 bytes captured (696 bits) on interface 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 10.1.10.29, Dst: 10.1.10.29
▶ Transmission Control Protocol, Src Port: 6796, Dst Port: 52390, Seq: 1, Ack: 352, Len: 19
```

Wireshark Capture



```
GET /test.html HTTP/1.1
Host: 10.1.10.29:6796
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 200 OK

<html>
<head><title>Week5</title></head>
<body>
<h1>Name : Divyanshu sharma</h1>
<h1>SRN : PES1US20CS806</h1>
</body>
</html>
```

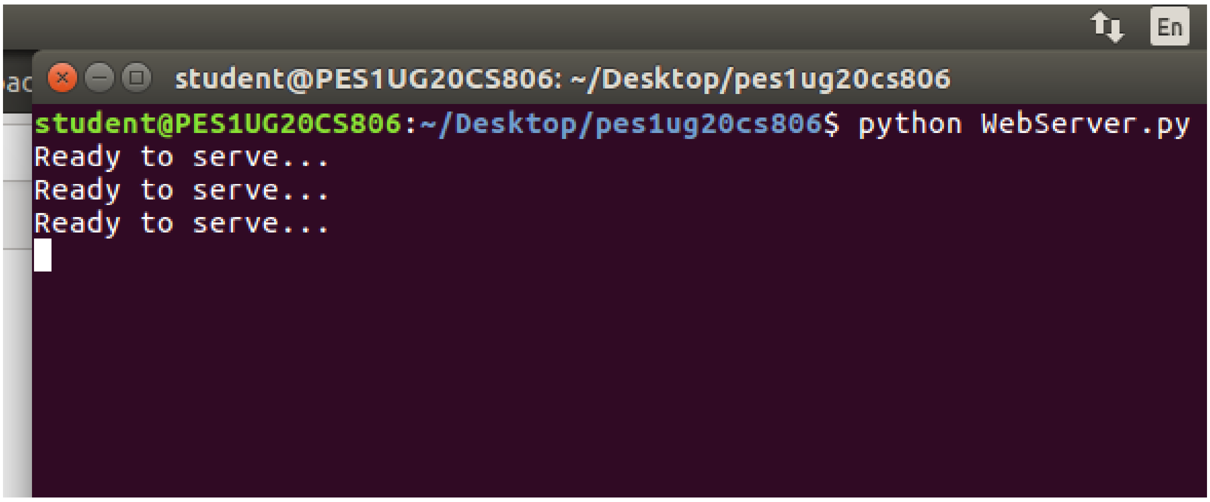
TCP PROTOCOL



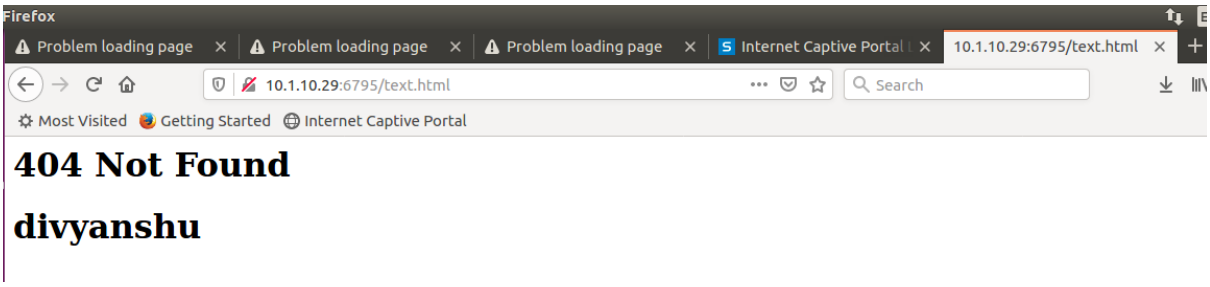
Name : Divyanshu sharma

SRN : PES1US20CS806

UNSUCCESSFUL ACCESS OF FILE RETURNING *404 NOT FOUND*

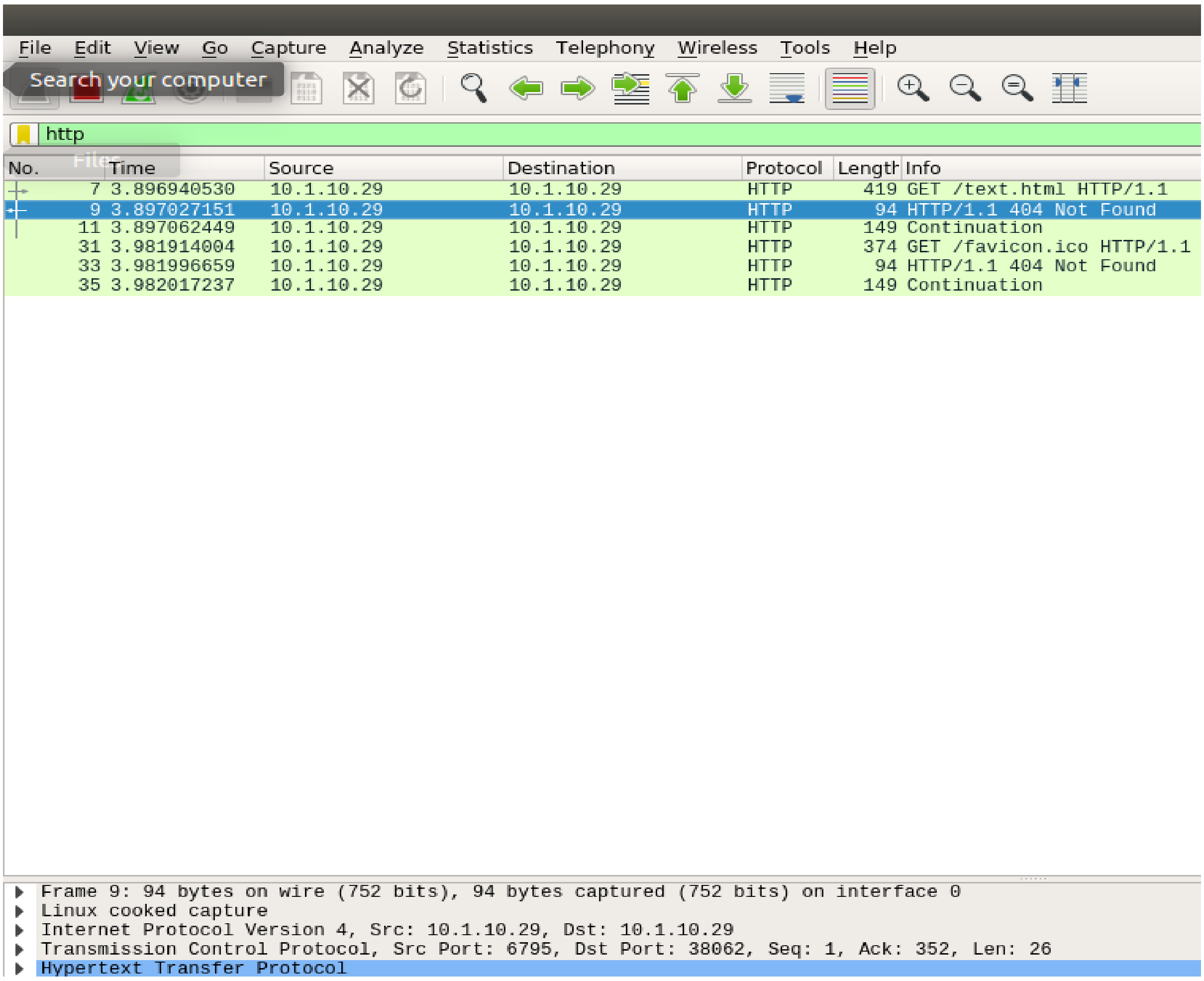


```
student@PES1UG20CS806:~/Desktop/pes1ug20cs806$ python WebServer.py
Ready to serve...
Ready to serve...
Ready to serve...
```



404 Not Found

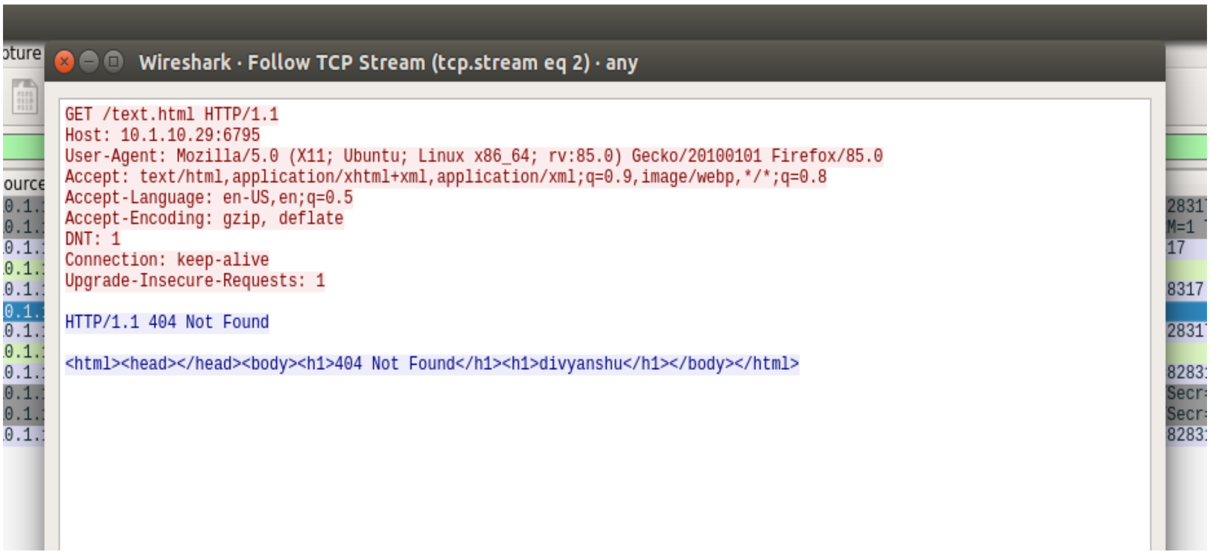
divyanshu



```
http
No.   Time         Source        Destination   Protocol Length Info
    7 3.896940530  10.1.10.29    10.1.10.29    HTTP      419 GET /text.html HTTP/1.1
    9 3.897027151  10.1.10.29    10.1.10.29    HTTP       94 HTTP/1.1 404 Not Found
   11 3.897062449  10.1.10.29    10.1.10.29    HTTP      149 Continuation
   31 3.981914004  10.1.10.29    10.1.10.29    HTTP      374 GET /favicon.ico HTTP/1.1
   33 3.981996659  10.1.10.29    10.1.10.29    HTTP       94 HTTP/1.1 404 Not Found
   35 3.982017237  10.1.10.29    10.1.10.29    HTTP      149 Continuation
```

```
▶ Frame 9: 94 bytes on wire (752 bits), 94 bytes captured (752 bits) on interface 0
▶ Linux cooked capture
▶ Internet Protocol Version 4, Src: 10.1.10.29, Dst: 10.1.10.29
▶ Transmission Control Protocol, Src Port: 6795, Dst Port: 38062, Seq: 1, Ack: 352, Len: 26
▶ Hypertext Transfer Protocol
```

Wireshark Capture



```
GET /text.html HTTP/1.1
Host: 10.1.10.29:6795
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:85.0) Gecko/20100101 Firefox/85.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
DNT: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1

HTTP/1.1 404 Not Found

<html><head></head><body><h1>404 Not Found</h1><h1>divyanshu</h1></body></html>
```

TCP PROTOCOL