Computers Network Laboratory Week #5

Name – B.Pravena **Sec -** B

SRN – PES2UG19CS076

Simple Client-Server Application using Network Socket Programming

Objective:

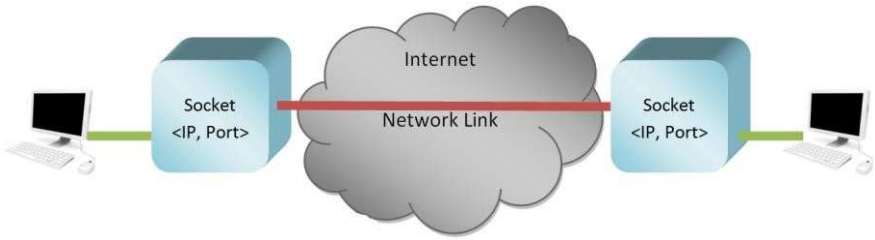
To develop a simple Client-Server application using TCP and UDP.

Pre requisites:

- Basic understanding of networking concepts and socket programming
- Knowledge of python

Sockets

Sockets are just the **endpoints of a two-way communication link** in a network. Socket helps in the communication of two processes/programs on a network (eg. Internet). The programs can communicate by reading/writing via their sockets. A socket comprises of: ***IP Address & Port number***



Task 1: (Mandatory for all students)

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).

Socket Programming with UDP

UDPClient.py

UDPClient.py (~/Documents) - gedit

Open Save

TCPClient.py UDPClient.py

```
from socket import*
serverName = "10.1.10.115"
serverPort = 12000
clientSocket = socket(AF_INET, SOCK_DGRAM)
message=input('Input lowercase sentence')
message=message.encode()
clientSocket.sendto(message,(serverName,serverPort))
modifiedMessage,serverAddress = clientSocket.recvfrom(2048)
print (modifiedMessage.decode())
clientSocket.close()
```

UDPServer.py

udpserver.py

```
from socket import *
serverPort = 12000
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)
```

Outputs -:

```
pesu@SYS-03:~/Documents$ python3 UDPClient.py
Input lowercase sentence hello, welcome
 HELLO, WELCOME
pesu@SYS-03:~/Documents$
```

```
pesse@SYS-04:~/Documents/week5snehal$ python3 udpserver.py
The server is ready to receive
```

Socket Programming with

TCP *TCPClient.py*

```
TCPClient.py
from socket import*
serverName = "10.1.10.115"
serverPort = 12000
clientSocket = socket(AF_INET,SOCK_STREAM)
clientSocket.connect((serverName,serverPort))
sentence = input('Input lowercase sentence')
sentence=sentence.encode()
clientSocket.send(sentence)
modifiedSentence=clientSocket.recv(1024)
print('from Server:',modifiedSentence.decode())
clientSocket.close()
```

TCPServer.py

```
from socket import *
serverPort = 12000
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()
```

Outputs-:

```
pesse@SYS-04:~/Documents/week5snehal$ python3 tcpserver.py
The server is ready to receive
```

```
pesu@SYS-03:~/Documents$ python3 TCPClient.py
Input lowercase sentence hello, welocome to lab
from Server:  HELLO, WELOCOME TO LAB
pesu@SYS-03:~/Documents$
```

Problems:

Install and compile the Python programs TCPClient and UDPClient on one host and TCPServer and UDPServer on another host.

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

If we run TCP Client before TCP server, we get an error that says Connection refused because, the client tries to establish a connection with a non-existent server (server program is not running). Hence, we get the error.

```
pesu@SYS-03:~/Documents$ python3 TCPClient.py
Traceback (most recent call last):
  File "TCPClient.py", line 5, in <module>
    clientSocket.connect((serverName,serverPort))
ConnectionRefusedError: [Errno 111] Connection refused
```

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

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.

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

This will lead to Connection Refused error for 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.

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.

Task 2: Web Server

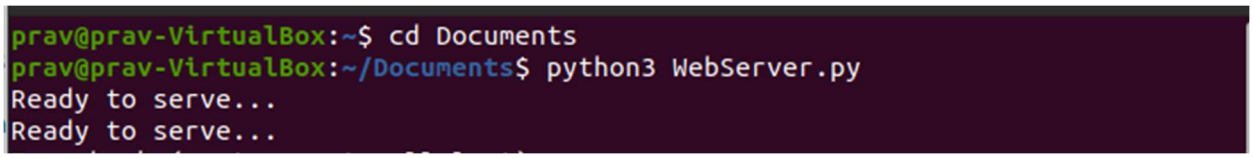
In this assignment, you will develop a simple Web server in Python that is capable of processing only one request. Specifically, your Web server will

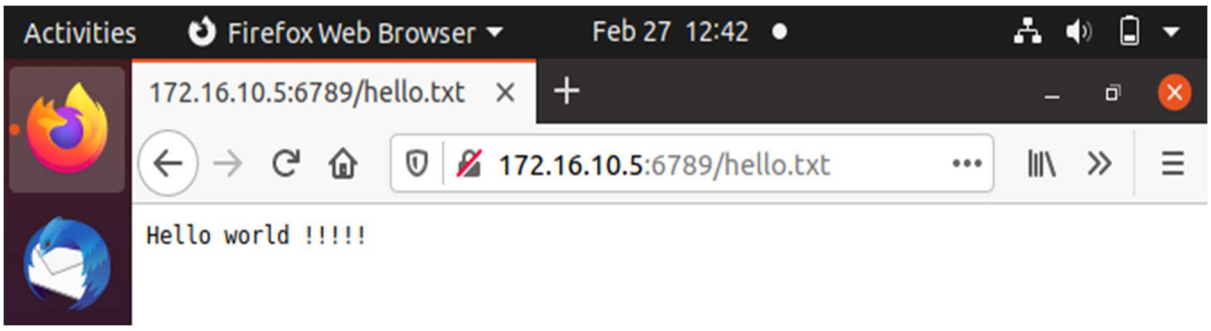
a) create a connection socket when contacted by a client (browser);
b) receive the HTTP request from this connection;
c) parse the request to determine the specific file being requested;
d) get the requested file from the server's file system;
e) create an HTTP response message consisting of the requested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

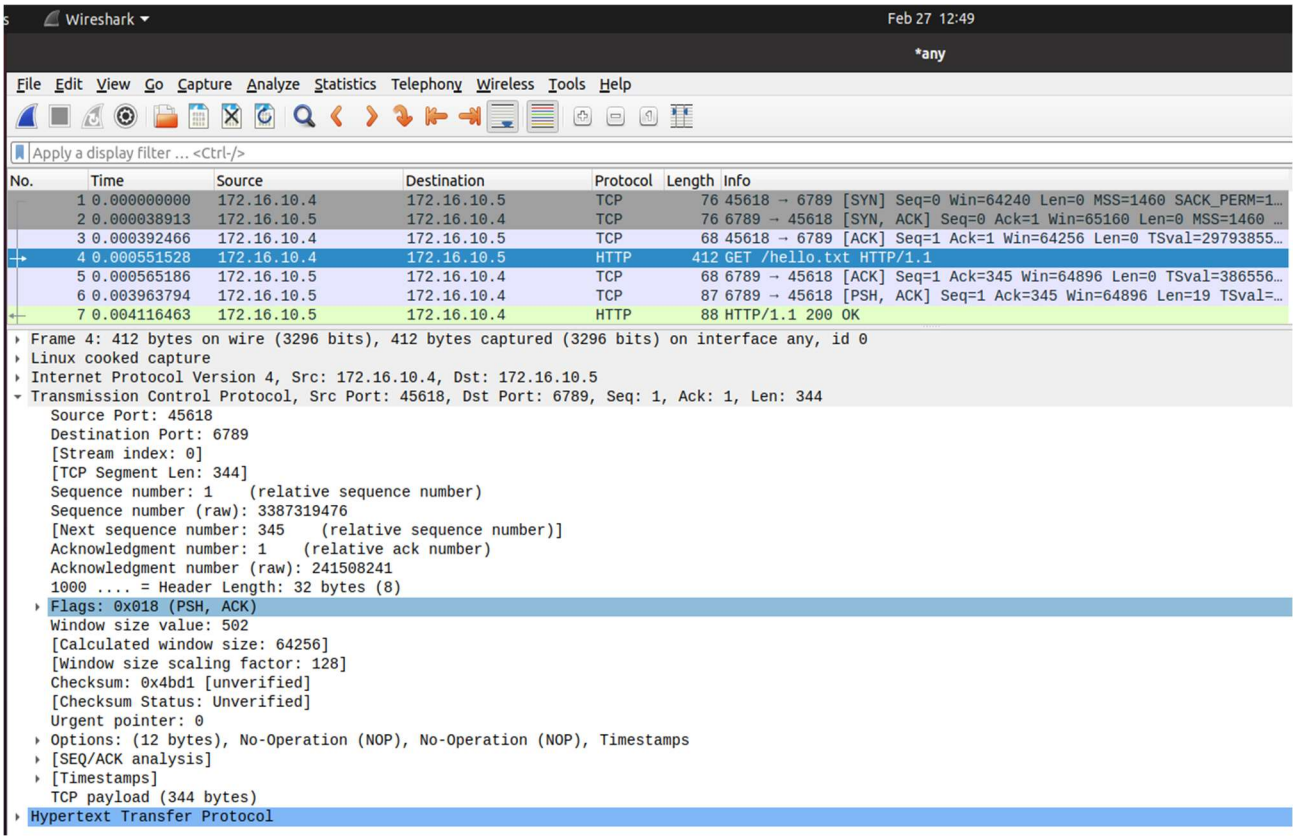
Not Found" error message.

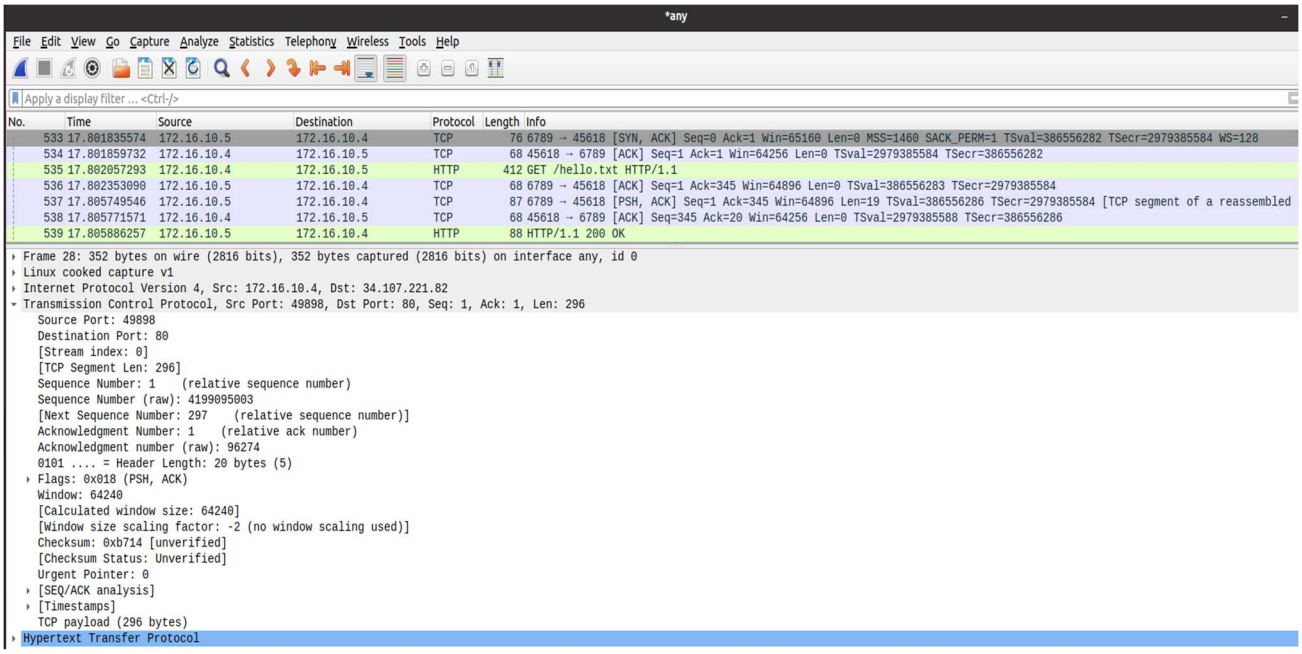
For this assignment, the companion Web site provides the skeleton code for your server. Your job is to complete the code, run your server, and then test your server by sending requests from browsers running on different hosts. If you run your server on a host that already has a Web server running on it, then you should use a different port than port 80 for your Web server.





Wireshark outputs -:





If file not found -: