Computer Networks Lab

Week 5

Abhishek Aditya BS
PES1UG19CS019
Section A

1. Socket Programming

1. Create an application that will
    - Convert lowercase letters to uppercase e.g. [a…z] to [A…Z] code will not change any special characters, e.g. &*!
    - 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 UDP Connection

- A UDP connection can be made between two machines with the help of a socket interface using the `socket` library on Python.
- To create a UDP socket interface, the type of socket needs to be set as `SOCK_DGRAM`.
- 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 UDP Server

UDPServer.py (~/Desktop) - gedit

```
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)
```

1.1.2 UDP Client

UDPClient.py (~/Desktop) - gedit

```
from socket import *
serverName = '10.2.20.252'
serverPort = 12000
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()
```

1.1.3 UDP Connection between Server and Client

student@pesu-OptiPlex-3070: ~/Desktop

```
student@pesu-OptiPlex-3070:~/Desktop$ python2 UDPServer.py
The server is ready to receive
```

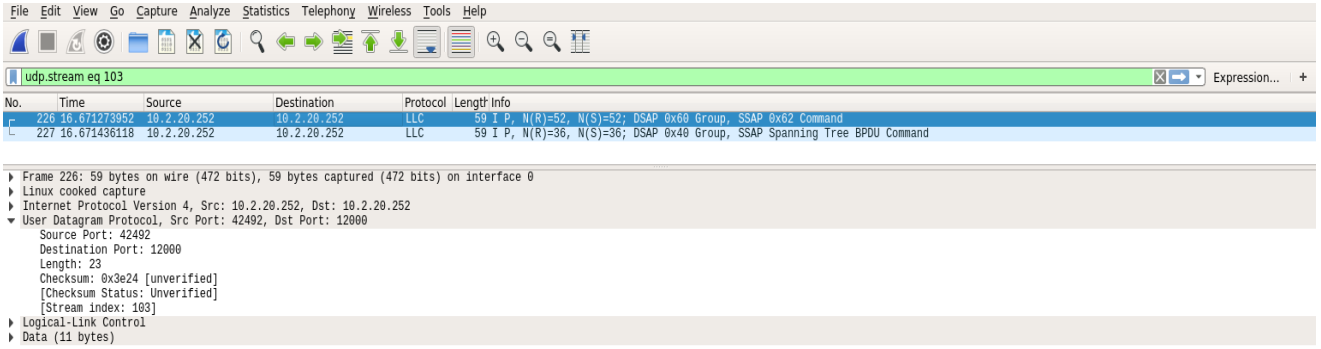
UDP Server

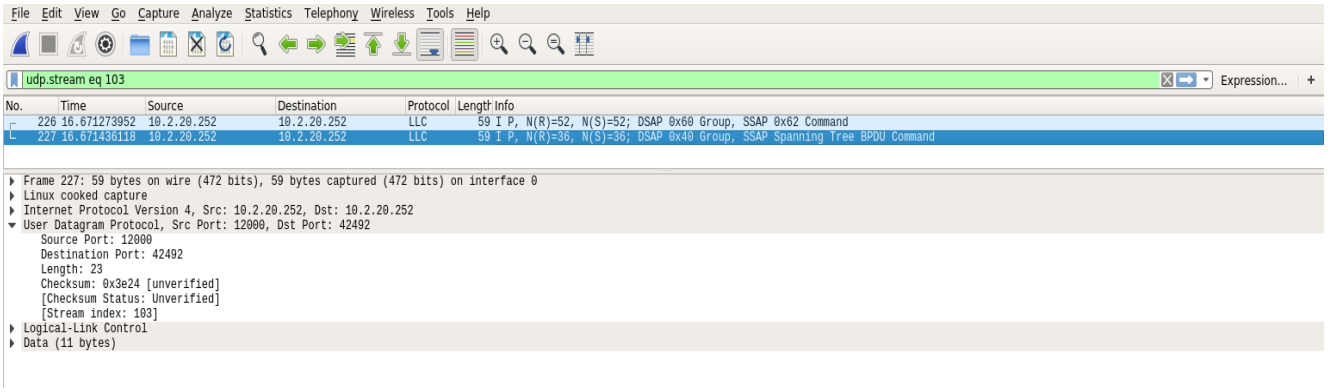
student@pesu-OptiPlex-3070: ~/Desktop

```
student@pesu-OptiPlex-3070:~/Desktop$ python2 UDPClient.py
Input lowercase sentence:abhishek aditya
ABHISHEK ADITYA
```

UDP Client

1.1.4 Wireshark Packet Capture for UDP Connection





1.2 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.

1.2.1 TCP Server

TCPServer.py (~/Desktop/PES1UG19CS019/TCP) - gedit

```
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()
```

1.2.2 TCP Client

TCPClient.py (~/Desktop/PES1UG19CS019/TCP) - gedit

```
from socket import *
serverName = '10.2.20.252'
serverPort = 12000
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()
```

1.2.3 TCP Connection between Server and Client

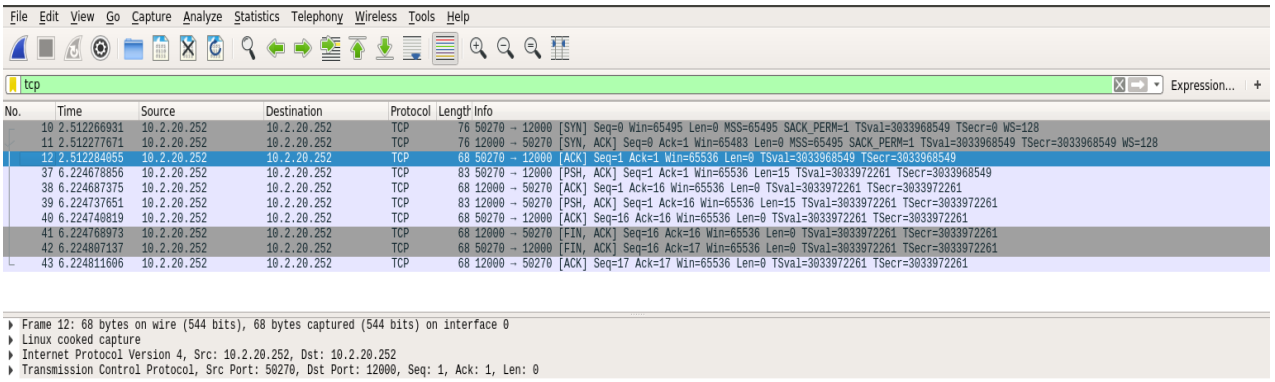
student@pesu-OptiPlex-3070: ~/Desktop

```
student@pesu-OptiPlex-3070:~/Desktop$ python2 TCPServer.py
The server is ready to receive
```

student@pesu-OptiPlex-3070: ~/Desktop

```
student@pesu-OptiPlex-3070:~/Desktop$ python2 TCPClient.py
Input lowercase sentence:abhishek aditya
From Server: ABHISHEK ADITYA
student@pesu-OptiPlex-3070:~/Desktop$
```

1.2.4 Wireshark Packet Capture for TCP Connection



1.3 Questions

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 to has not been initiated and is not listening for connections on the given port number.* Hence, any connection requests sent by a client machine at that IP 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 set up and the connection is 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 exist.

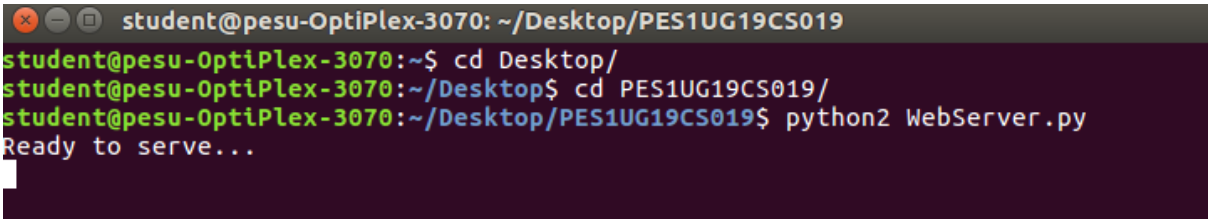
2. Task 2 - Web Server

A simple Web server in Python that is capable of processing only one request

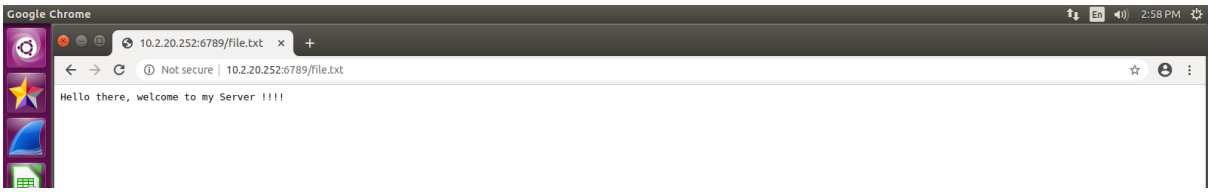
```
# Import socket module
from socket import *

# Create a TCP server socket
#(AF_INET is used for IPv4 protocols)
#(SOCK_STREAM is used for TCP)

serverSocket = socket(AF_INET, SOCK_STREAM)

# Assign a port number
serverPort = 6789

# Bind the socket to server address and server port
serverSocket.bind(("", serverPort))

# Listen to at most 1 connection at a time
serverSocket.listen(1)

# Server should be up and running and listening to the incoming connections
while True:
        print 'Ready to serve...'

        # Set up a new connection from the client
        connectionSocket, addr = serverSocket.accept()

        # If an exception occurs during the execution of try clause
        # the rest of the clause is skipped
        # If the exception type matches the word after except
        # the except clause is executed
        try:
                # Receives the request message from the client
                message =  connectionSocket.recv(1024)
                # Extract the path of the requested object from the message
                # The path is the second part of HTTP header, identified by [1]
                filename = message.split()[1]
```

```
            # Because the extracted path of the HTTP request includes
            # a character '\', we read the path from the second character
            f = open(filename[1:])
            # Store the entire content of the requested file in a temporary buffer
            outputdata = f.read()
            # Send the HTTP response header line to the connection socket
            connectionSocket.send("HTTP/1.1 200 OK\r\n\r\n")

            # Send the content of the requested file to the connection socket
            for i in range(0, len(outputdata)):
                    connectionSocket.send(outputdata[i])
            connectionSocket.send("\r\n")

            # Close the client connection socket
            connectionSocket.close()

        except IOError:
            # Send HTTP response message for file not found
            connectionSocket.send("HTTP/1.1 404 Not Found\r\n\r\n")
            connectionSocket.send("<html><body><h1>404 NotFound</h1></body></html>\r\n")
            # Close the client connection socket
            connectionSocket.close()

serverSocket.close()
```

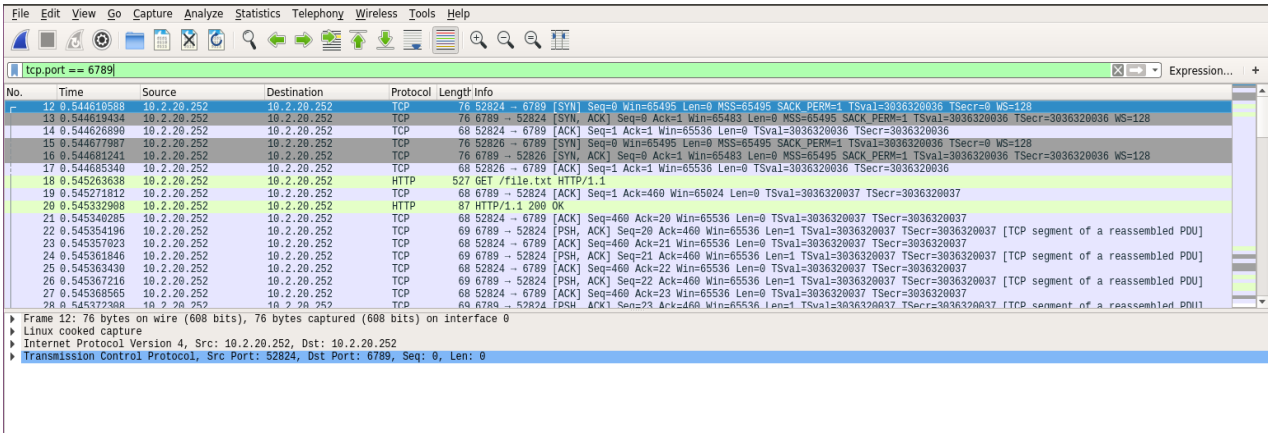
2.1 Running the Web Server



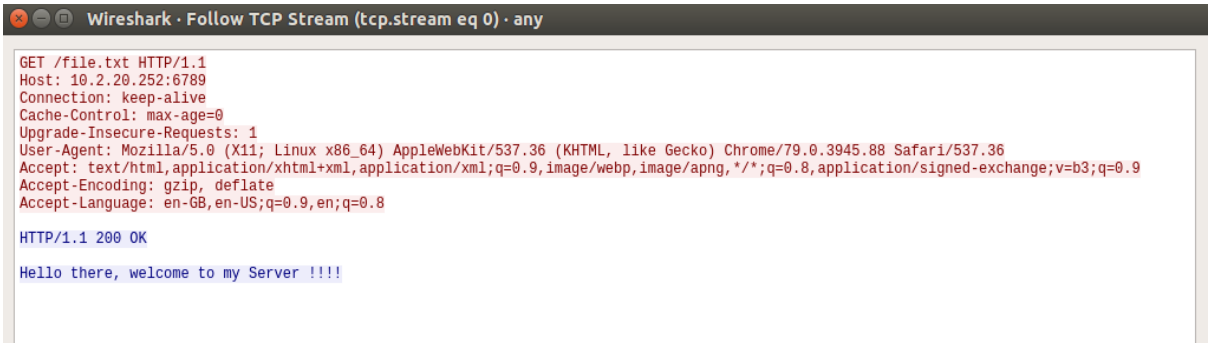
2.2 Accessing the file from Server's file system



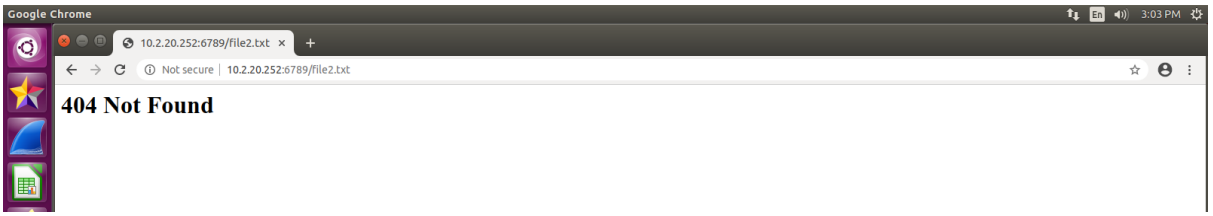
2.3 Wireshark Capture



2.4 TCP Stream of the request to the server



2.5 Requesting a file not Present in the Server



Server returns a “404 Not Found” error message