Computer Networks Assignment:3

Aman Vishnoi
12040090

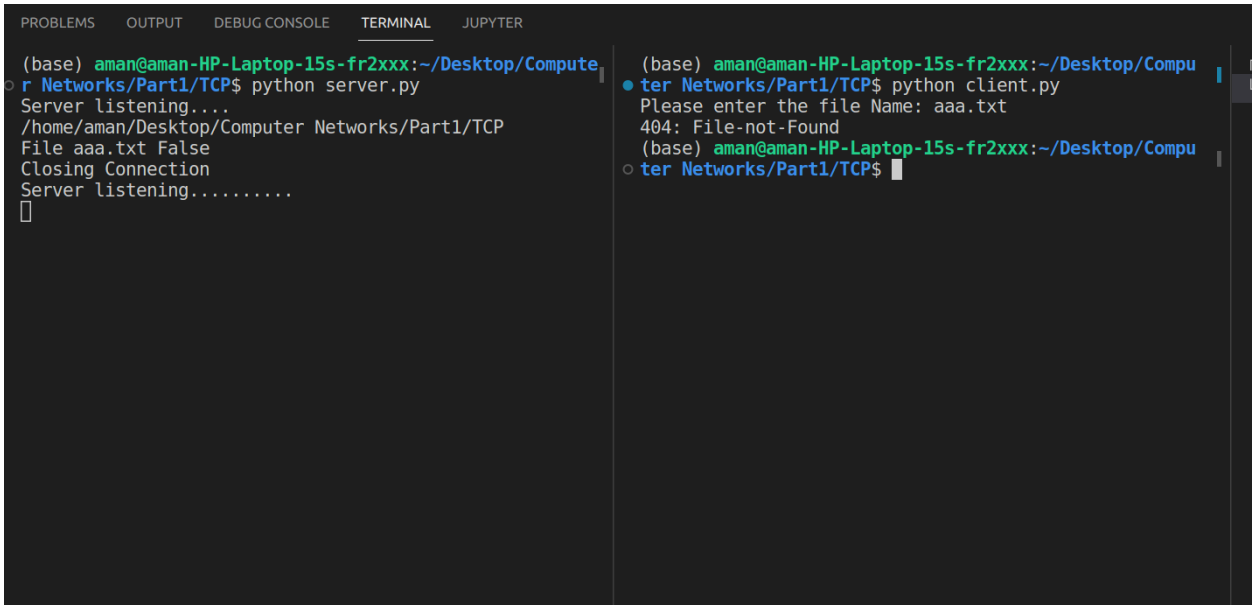
To run any code please find the readme file attached in respective folders

Question 1:

I have used python and socket (Low level programming) modules to make the codes.

TCP:

In the case of TCP first we bind the server to an IP(localhost) and a not reserved port number. Then we create the client which first connects to the server. The server will acknowledge the incoming request and establish a connection with the client. The client then sends the file name for the server. The server checks if the file is in the working directory or not. If not then it will send “404: File Not Found Error”.



And closes the client connection and again waits for a new connection to establish.

If the file is present in the working directory then it will send the confirmation to the client that file is present which word do you want and client starts a loop asking for Word#iteration_number. The server extracts the contents of the file and returns the word in binary format to the client. The client then again asks for another word and this continues until the word is EOF. When EOF comes the client closes the connection and the server again starts listening for new connections. The received contents are saved in a new file called received_file.txt.

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Compute
r Networks/Part1$ c
d TCP/
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Compute
r Networks/Part1/TCP$ python server.py
Server listening....
/home/aman/Desktop/Computer Networks/Part1/TCP
File file.txt True
Server received 'Word_#1'
Done sending Word_#1
Server received 'Word_#2'
Done sending Word_#2
Server received 'Word_#3'
Done sending Word_#3
Server received 'Word_#4'
Done sending Word_#4
Server received 'Word_#5'
Done sending Word_#5
Server received 'Word_#6'
Done sending Word_#6
Server received 'Word_#7'
Done sending Word_#7
Server received 'Word_#8'
Done sending Word_#8
Closing Connection
Server listening..........
```

```
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Compu
ter Networks/Part1/TCP$ python client.py
Please enter the file Name: file.txt
Successfully found file 1
data=%s b'Aman'
data=%s b'Vishnoi'
data=%s b'should'
data=%s b'get'
data=%s b'B'
data=%s b'in'
data=%s b'CN'
data=%s b'EOF'
Successfully get the file
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Compu
ter Networks/Part1/TCP$
```

UDP:

For the UDP client server everything is the same except for a few changes in the code base such as UDP doesn't require the connection to be established first and closing the connection part. So in case of UDP we just pass in the file name, if the file exists it will return the content of the file and store them in received_file.txt

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Compute
r Networks/Part1/UPD$ python server.py
Server listening....
Server received 'Word_#1'
Done sending Word_#1
Server received 'Word_#2'
Done sending Word_#2
Server received 'Word_#3'
Done sending Word_#3
Server listening..........
```

```
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Compu
ter Networks/Part1/ PD$ python client.py
data=%s b'Aman'
data=%s b'Vishnoi'
data=%s b'EOF'
Successfully get the file
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Compu
ter Networks/Part1/ PD$
```

Question 2:

a) I have created a UDP server and a UDP client.The client creates a UDP socket and takes arguments as time_interval between two packets, number of packets and packet_size. I have appended a timestamp in each packet which contains the details when the packet is sent and upon receiving the packet by echo server I calculate the difference between the present time and the time stamp from the packet to calculate the RTT. Since the packets are transferred from my machine to my machine itself there is no packet loss, so I have to simulate packet loss like in a real world environment by randomly dropping the packet with the probability of 10%. Based on timeout packet loss percentage is calculated at the end

```
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Networks/Part2/Parta$ python udp_echo_server.py
UDP server up and listening
```

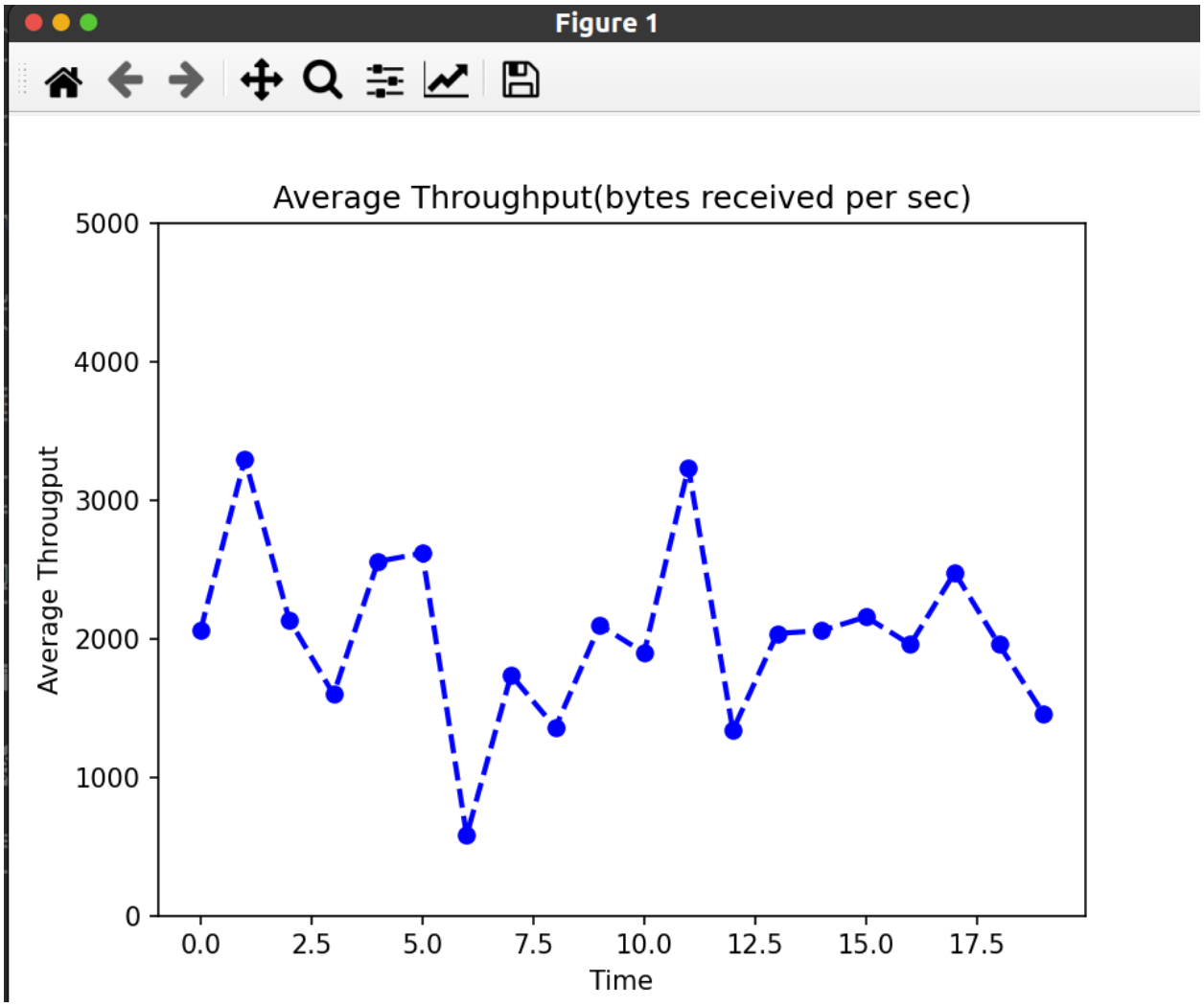
```
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Networks/Part2/Parta$ python udp_echo_client.py --time_interval 2 --num_packets 15 --packet_size 10
PacketSent
RTT observed 0.0032099999999999997
PacketSent
RTT observed 0.001432
PacketSent
RTT observed 0.001184
PacketSent
RTT observed 0.001065
PacketSent
RTT observed 0.000966
PacketSent
RTT observed 0.001029
PacketSent
RTT observed 0.000995
PacketSent
RTT observed 0.001044
PacketSent
Packet lost.
PacketSent
RTT observed 0.001013
PacketSent
RTT observed 0.000932
PacketSent
RTT observed 0.000865
PacketSent
RTT observed 0.001013
PacketSent
RTT observed 0.00095
PacketSent
RTT observed 0.00116
Percentage of Packets Lost 6.666666666666667%
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Networks/Part2/Parta$
```

b) For the part b, I have reduced the time interval between two consecutive packets to machine's capability and I have taken the packet size as an argument and I have sent as many packets as machine can and append average delay and average throughput in a list and have plotted them in the graph

```
(NLP) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Networks/Part2/Partb$ python udp_echo_client.py  --packet_size 20
Time t= 0 sec
Time t= 1 sec
Time t= 2 sec
Time t= 3 sec
Time t= 4 sec
Time t= 5 sec
Time t= 6 sec
Time t= 7 sec
Time t= 8 sec
Time t= 9 sec
Time t= 10 sec
Time t= 11 sec
Time t= 12 sec
Time t= 13 sec
Time t= 14 sec
Time t= 15 sec
Time t= 16 sec
Time t= 17 sec
Time t= 18 sec
Time t= 19 sec
```

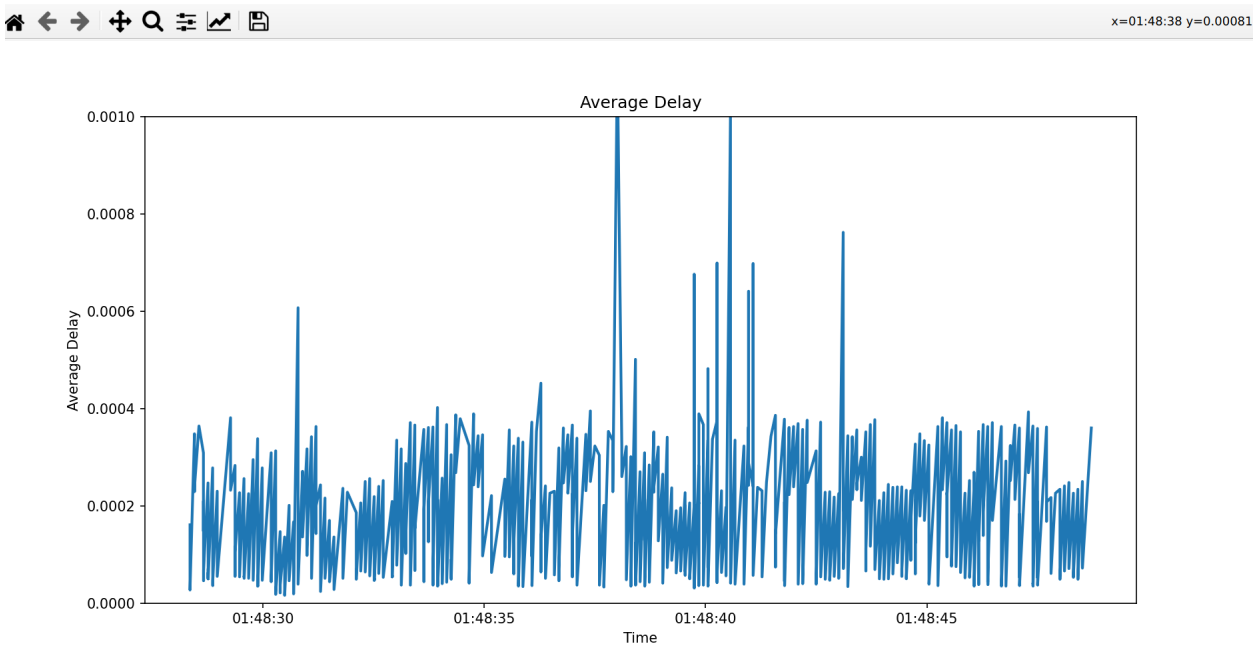
```
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Networks/Part2/Partb$ python udp_echo_server.py
UDP server up and listening
```

Average Throughput:



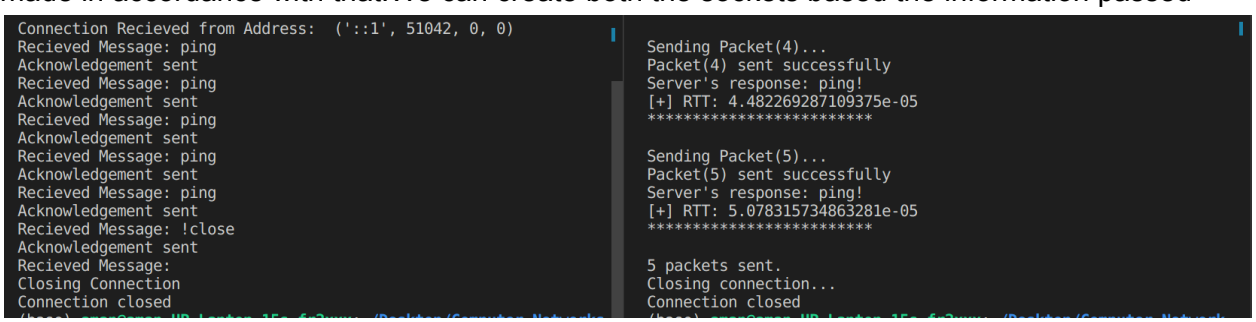
The y-axis is bytes and x-axis is time in seconds

Average Delay:



Question 3:

Question3 is the same as question 2 except for protocol independence. We can view the version of the IP address via the socket.getaddrinfo(server ip, server port) function in the Python socket library, which delivers information about the socket address. The socket can then be made in accordance with that.We can create both the sockets based the information passed



Question 4:

I have created a simple functionality of google drive where a user can view, download, upload, delete files on the server.

This works on TCP protocol so a connection has to be established first with the server. And you cannot write any unrecognized command.

```
Connect to server first


Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do:
```

```
/bin/python3 "/home/aman/Desktop/Computer Networks/Part4/server.p
y"
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Networks
/Part4$ /bin/python3 "/home/aman/Desktop/Computer Networks/Part4/
server.py"
Server ready to accept connections
```

```
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Network
s/Part4$ python client.py


Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: ^[OQ
Command not recognised; please try again


Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do:
```

After connecting with the server you can do pretty much all the operations that you want. The detailed information for all the operations along with command line arguments is given.
If you try to do any action without actually connecting with the server

```
Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: DELF file1.txt
Connection not established
```

It will prompt you to establish the connection first with the server.

After successful connection it will show something like

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    JUPYTER

/bin/python3 "/home/aman/Desktop/Computer Networks/Part4/server.p
y"
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Networks
/Part4$ /bin/python3 "/home/aman/Desktop/Computer Networks/Part4/
server.py"
Server ready to accept connections
Connection Esatblished Successfully by ('127.0.0.1', 57166)


Waiting for instruction
```

```
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: LIST


Connect to server first


Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: CONN
Connection sucessful


Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do:
```

The server will start listening for instructions from the client.

A) LIST:

```
QUIT            : Exit
Enter what you want to do: LIST
All files in the folder are
 client.txt,file1.txt,filee.txt


Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do:
```

It will list all the files which the server has at the moment. It will first get all the files in the working directory and then one file at a time to the server to prevent BUFFER overflow

B) Download

This will let you download files from the server into your current directory. First the server check if the file exists if not it will return -1

```
Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: DWLD file1.txt
File Size -1
File does not exist on the server
```

If it exists on the server it will read it in the binary format break it into chunk send the file size to the client and start sending the file chunk by chunk so that there is not loss of the packet due to buffer overflow

```
Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: DWLD Files/file1.txt
File Size 12

Downloading the file
Successfully downloaded Files/file1.txt
Time elapsed: 0.005459s
File size: 12b
```

C) Upload

This is the same as downloading except that the file is sent by the client and the server is the receiver.First the client sends the fileSize name and gets the info regarding the server BUFFER size. System check if the client has the file it is trying to upload if it exists then it will split it up into chunks depending on server BUFFER Size and starts

sending it.

```
Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: UPLD filee.txt

Uploading file: filee.txt...
File_Size 13

Sending...
filee.txt successfully sent. Size: 13 time taken 0.005597
```

D) DELETE

This is used for deleting a file from the server by the client. First the client sends the request to the server for deletion of the file. If the file doesn't exist on the server, the server responds by FileNotFound Error and if the file exists the server asks for confirmation from the client whether he wants to delete the file or not. If the client picks no then the operation is abandoned and if it presses Yes the file is deleted from the server.

```
Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: DELF file1.txt
The file does not exist on server


Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do:
```

```
Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: DELF file1.txt
Are you sure you want to delete file1.txt? (Y/N)
```

```
Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: DELF file1.txt
Are you sure you want to delete file1.txt? (Y/N)
Y
File successfully deleted
```

E) QUIT

This will closes the connection between the client and the server and the server again starts listening for new connections

```
Waiting for instruction
Instructions From Client DWLD
filename passed Files/file1.txt
File Exists on the server


Waiting for instruction
Instructions From Client QUIT
Connection over by the client ('127.0.0.1', 41192)
Closing Connection
Server Listening
```

```
Welcome to the FTP client.

Call one of the following functions:
CONN           : Connect to server
UPLD file_path : Upload file
LIST           : List files
DWLD file_path : Download file
DELF file_path : Delete file
QUIT           : Exit
Enter what you want to do: QUIT
b'Connection Over'
Server connection ended
(base) aman@aman-HP-Laptop-15s-fr2xxx:~/Desktop/Computer Net
s/Part4$
```