ACN Programming-Assignment-1
My Pingers

Submitted by:
Abdulla Ovais - CS24MTECH12014
Sakshi Badole - CS24MTECH11008
Simeon Sahasramsa Gajula - SM24MTECH11002

భారతీయ సాంకేతిక విజ్ఞాన సంస్థ హైదరాబాద్
भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

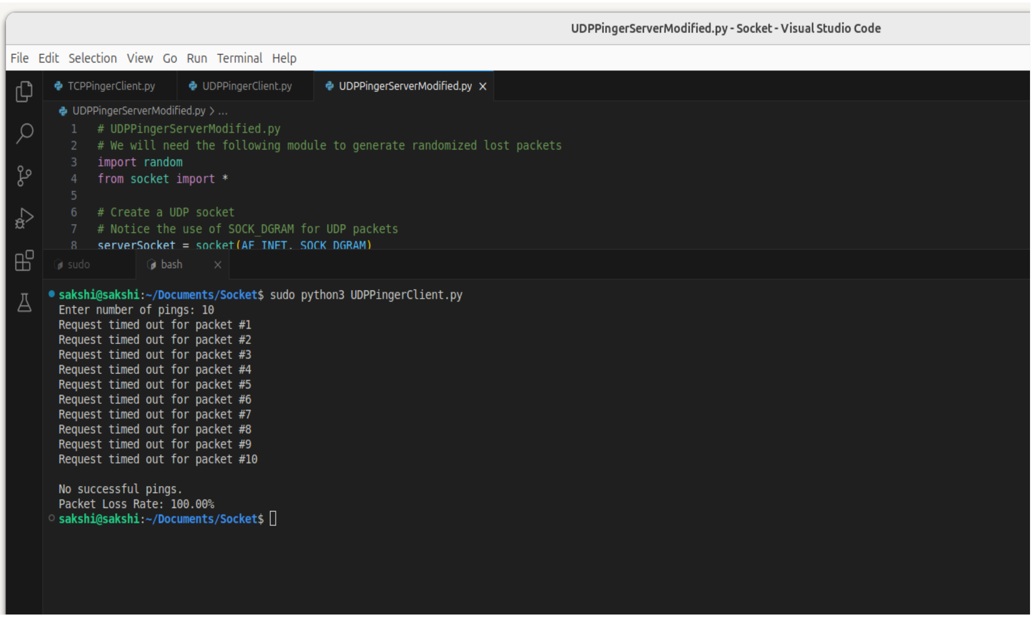
My Pingers

PART-1 : UDP Pinger

In the case of UDP Pinger, we are creating a UDP server and a UDP client, here the client pings the UDP server and the client calculates the delay or the RTT of each delay, as well as it (the client) calculates the maximum, minimum, average RTT and the packet loss percent.

Case 1: UDP Client to UDP Server(Simulated packet loss using random integer)

In the case of UDP Pinger, we have created a UDPPingerClient.py which uses UDP sockets to ping the UDP Server. The UDPServer works replies with a certain probability otherwise the client prints the request timed out.


UDPPingerServerModified.py - Socket - Visual Studio Code
File Edit Selection View Go Run Terminal Help
TCPPingerClient.py
UDPPingerClient.py
UDPPingerServerModified.py
UDPPingerServerModified.py > ...
sudo
bash

```
# UDPPingerServerModified.py
# We will need the following module to generate randomized lost packets
import random
from socket import *

# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets
serverSocket = socket(AF_INET, SOCK_DGRAM)
```

```
sakshi@sakshi:~/Documents/Socket$ sudo python3 UDPPingerClient.py
Enter number of pings: 10
Request timed out for packet #1
Request timed out for packet #2
Request timed out for packet #3
Request timed out for packet #4
Request timed out for packet #5
Request timed out for packet #6
Request timed out for packet #7
Request timed out for packet #8
Request timed out for packet #9
Request timed out for packet #10

No successful pings.
Packet Loss Rate: 100.00%
sakshi@sakshi:~/Documents/Socket$
```

Fig: Window displaying request timed out for the UDP Client when pings the UDP server having simulated packet

Workflow of UDP: The program is sending N pings to the server and calculating RTT for each ping. Then it adds the RTT of each paclet into a list and calculate Minumum RTT, Maximum RTT, Average RTT and Packet Loss Rate.
In the case of running multiple clients, we made some changes to the Server program because when a particular client while it’s running stops abruptly (or closed forcefully) then the Server was not responding to the other Clients which sent ping request. So in the Server code we added an except case in which the server will close the pervious socket and again create a socket, the server starts listening again.

We created a UDP CLIENT ping program which sends a ping to the UDP server in lowercase and the server send the ping back in uppercase.

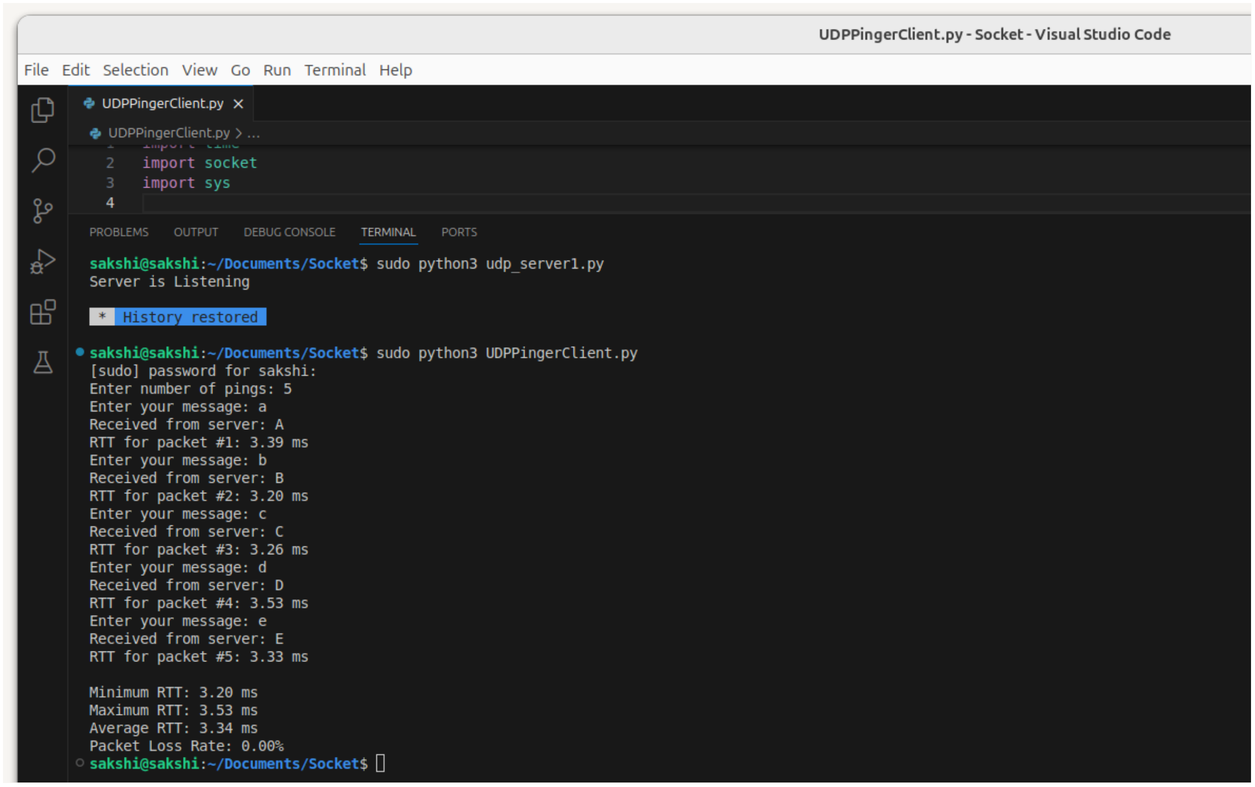
CASE 2: UDP Client to Modified UDP Server:

Here the modified UDP server does not simulate packet drop using random integer but a linux utility for traffic control on the server side, using the following command:

sudo tc qdisc add dev <interface> root netem loss 20%

To delete this rule the following command is used :

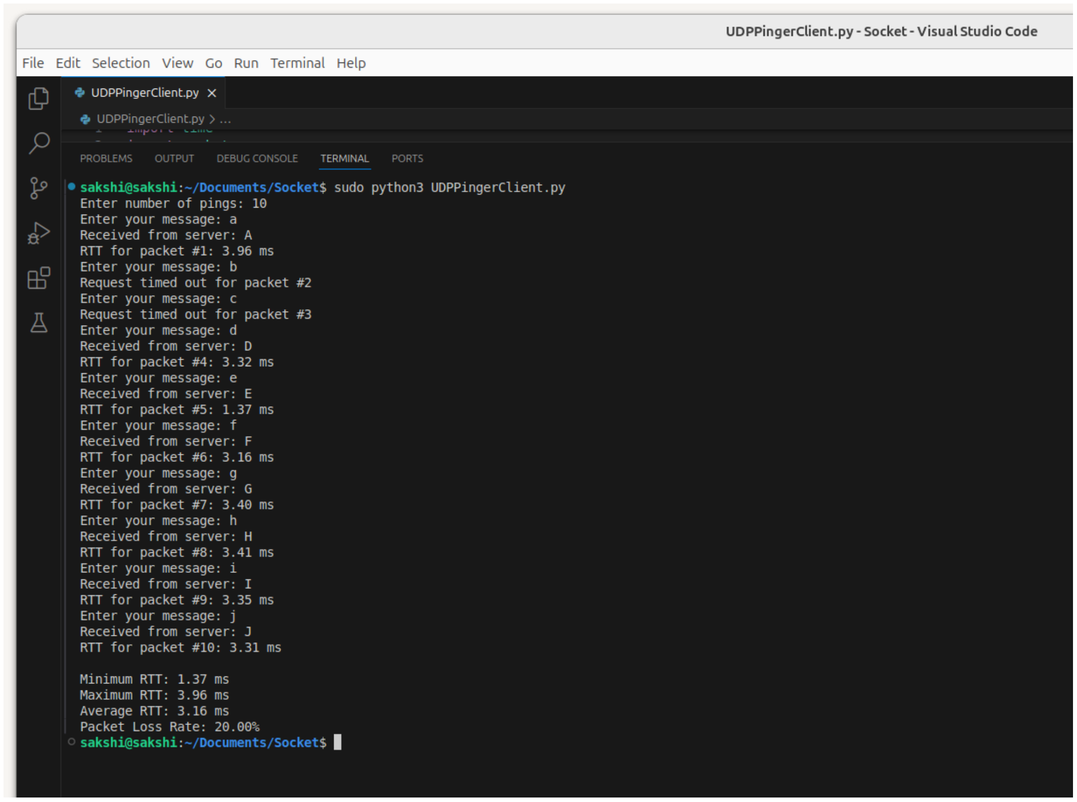
sudo tc qdisc del dev <interface> root


UDPPingerClient.py - Socket - Visual Studio Code
File Edit Selection View Go Run Terminal Help
UDPPingerClient.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
import socket
import sys
```

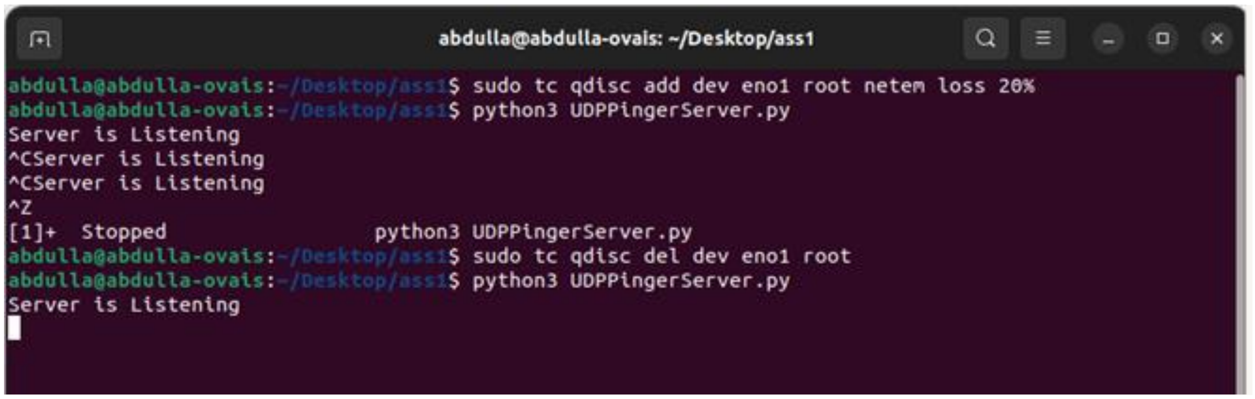
```
sakshi@sakshi:~/Documents/Socket$ sudo python3 udp_server1.py
Server is Listening

 *  History restored

sakshi@sakshi:~/Documents/Socket$ sudo python3 UDPPingerClient.py
[sudo] password for sakshi:
Enter number of pings: 5
Enter your message: a
Received from server: A
RTT for packet #1: 3.39 ms
Enter your message: b
Received from server: B
RTT for packet #2: 3.20 ms
Enter your message: c
Received from server: C
RTT for packet #3: 3.26 ms
Enter your message: d
Received from server: D
RTT for packet #4: 3.53 ms
Enter your message: e
Received from server: E
RTT for packet #5: 3.33 ms

Minimum RTT: 3.20 ms
Maximum RTT: 3.53 ms
Average RTT: 3.34 ms
Packet Loss Rate: 0.00%
sakshi@sakshi:~/Documents/Socket$
```

Fig: Snapshot displaying UDP Pinger Client output when the UDP server was run without any simulated packet loss i.e. before running the sudo tc qdisc add dev <interface> root netem loss 20% command.


UDPPingerClient.py - Socket - Visual Studio Code
File Edit Selection View Go Run Terminal Help
UDPPingerClient.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
sakshi@sakshi:~/Documents/Socket$ sudo python3 UDPPingerClient.py
Enter number of pings: 10
Enter your message: a
Received from server: A
RTT for packet #1: 3.96 ms
Enter your message: b
Request timed out for packet #2
Enter your message: c
Request timed out for packet #3
Enter your message: d
Received from server: D
RTT for packet #4: 3.32 ms
Enter your message: e
Received from server: E
RTT for packet #5: 1.37 ms
Enter your message: f
Received from server: F
RTT for packet #6: 3.16 ms
Enter your message: g
Received from server: G
RTT for packet #7: 3.40 ms
Enter your message: h
Received from server: H
RTT for packet #8: 3.41 ms
Enter your message: i
Received from server: I
RTT for packet #9: 3.35 ms
Enter your message: j
Received from server: J
RTT for packet #10: 3.31 ms

Minimum RTT: 1.37 ms
Maximum RTT: 3.96 ms
Average RTT: 3.16 ms
Packet Loss Rate: 20.00%
sakshi@sakshi:~/Documents/Socket$
```

Fig: Snapshot displaying UDP Pinger Client output when the UDPModifiedServer was run having codefor simulated packet loss using command“sudo tc disc add dev <interface> root netem loss <percent>”

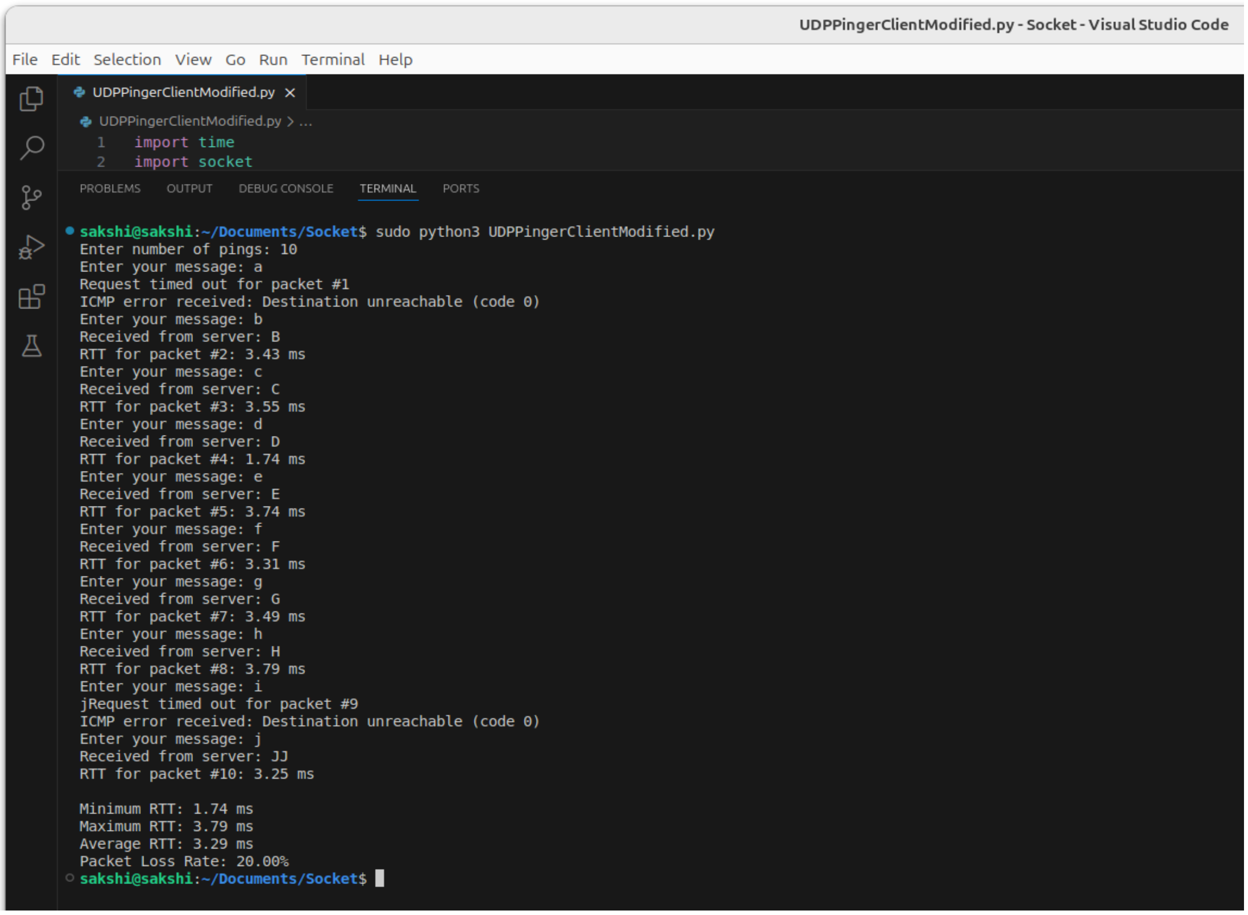


```
abdulla@abdulla-ovais:~/Desktop/ass1$ sudo tc qdisc add dev eno1 root netem loss 20%
abdulla@abdulla-ovais:~/Desktop/ass1$ python3 UDPPingerServer.py
Server is Listening
^CServer is Listening
^CServer is Listening
^Z
[1]+  Stopped                 python3 UDPPingerServer.py
abdulla@abdulla-ovais:~/Desktop/ass1$ sudo tc qdisc del dev eno1 root
abdulla@abdulla-ovais:~/Desktop/ass1$ python3 UDPPingerServer.py
Server is Listening
```

Fig: Displaying terminal window running UDPPingerServer after running command “sudo tc disc add dev <interface> root netem loss <percent>”.

CASE 3: Modified UDP Pinger Client to Modified UDP Pinger Server

Here, the UDP client and the server has been modified so as to receive and send an ICMP error message such as destination network unreachable. For this an ICMP socket has been created in both client and server codes. The server packs the icmp error message and sends it via ICMP socket whenever it randomly drops the client's ping. The client receives and unpacks the ICMP packet through the ICMP socket in the section where it previously displayed the Request-timed-out.


UDPPingerClientModified.py - Socket - Visual Studio Code
File Edit Selection View Go Run Terminal Help
UDPPingerClientModified.py
UDPPingerClientModified.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
import time
import socket
```

```
sakshi@sakshi:~/Documents/Socket$ sudo python3 UDPPingerClientModified.py
Enter number of pings: 10
Enter your message: a
Request timed out for packet #1
ICMP error received: Destination unreachable (code 0)
Enter your message: b
Received from server: B
RTT for packet #2: 3.43 ms
Enter your message: c
Received from server: C
RTT for packet #3: 3.55 ms
Enter your message: d
Received from server: D
RTT for packet #4: 1.74 ms
Enter your message: e
Received from server: E
RTT for packet #5: 3.74 ms
Enter your message: f
Received from server: F
RTT for packet #6: 3.31 ms
Enter your message: g
Received from server: G
RTT for packet #7: 3.49 ms
Enter your message: h
Received from server: H
RTT for packet #8: 3.79 ms
Enter your message: i
jRequest timed out for packet #9
ICMP error received: Destination unreachable (code 0)
Enter your message: j
Received from server: JJ
RTT for packet #10: 3.25 ms

Minimum RTT: 1.74 ms
Maximum RTT: 3.79 ms
Average RTT: 3.29 ms
Packet Loss Rate: 20.00%
sakshi@sakshi:~/Documents/Socket$
```

Fig: Displaying modified UDP Pinger Client being run showing ICMP error message with error code.

PART-2: TCP Pinger

In the case of TCP Pinger, we are creating a TCP server socket and a TCP client socket, here the client pings the TCP server and the client calculates the delay or the RTT of each delay, as well as it (the client) calculates the maximum, minimum, average RTT and the packet loss percent.

Workflow of TCP-CASE-1:

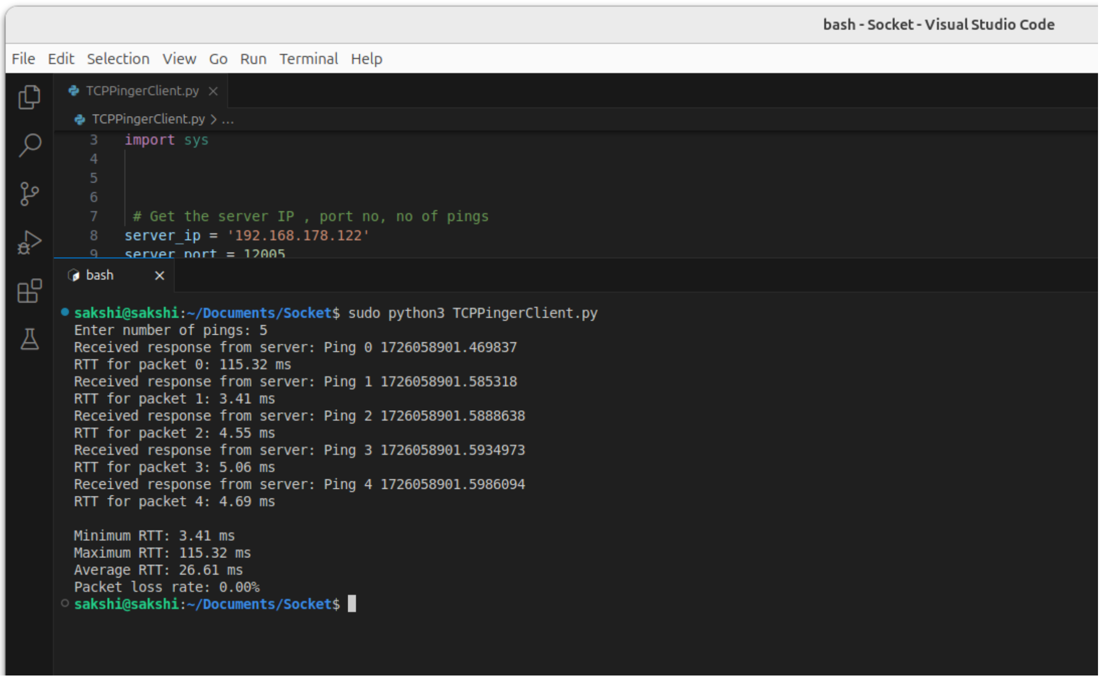
1. First the client and server connects with each others.

2. Then the client send the packet to the server.

3. Now the server sends the response to the client, and we calculate the RTT for each packet min.RTT and max.RTT and avg.RTT but the packet loss rate does not increase due to no packet loss.

CASE 1: TCP Client to TCP Server(No concurrent connections)

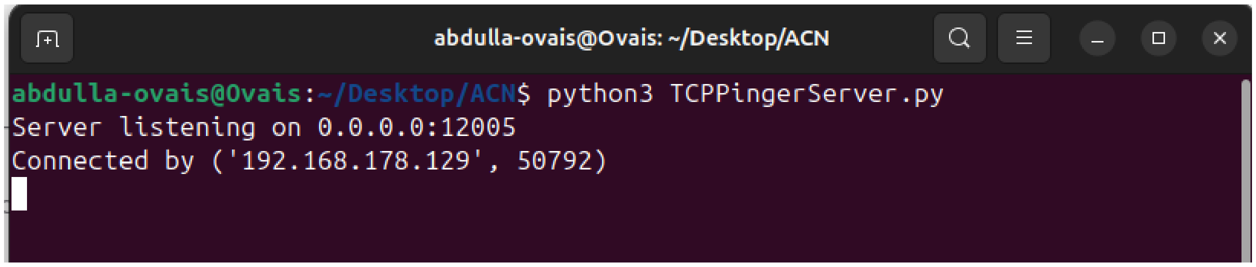
In the case of TCP Pinger, the TCP server does not manage concurrent requests it services one client at a time.


bash - Socket - Visual Studio Code
File Edit Selection View Go Run Terminal Help
TCPPingerClient.py
TCPPingerClient.py > ...
bash

```
import sys



 # Get the server IP , port no, no of pings
server_ip = '192.168.178.122'
server_port = 12005
```

```
sakshi@sakshi:~/Documents/Socket$ sudo python3 TCPPingerClient.py
Enter number of pings: 5
Received response from server: Ping 0 1726058901.469837
RTT for packet 0: 115.32 ms
Received response from server: Ping 1 1726058901.585318
RTT for packet 1: 3.41 ms
Received response from server: Ping 2 1726058901.5888638
RTT for packet 2: 4.55 ms
Received response from server: Ping 3 1726058901.5934973
RTT for packet 3: 5.06 ms
Received response from server: Ping 4 1726058901.5986094
RTT for packet 4: 4.69 ms

Minimum RTT: 3.41 ms
Maximum RTT: 115.32 ms
Average RTT: 26.61 ms
Packet loss rate: 0.00%
sakshi@sakshi:~/Documents/Socket$
```

Fig: Displaying TCP Pinger Client running and calculating the minimum, maximum, average RTTs and Packet loss percent.



```
abdulla-ovais@Ovais:~/Desktop/ACN$ python3 TCPPingerServer.py
Server listening on 0.0.0.0:12005
Connected by ('192.168.178.129', 50792)
```

Fig: Displaying TCP Pinger Server running.

Workflow of TCP-CASE-2:

1. First the multiple clients and server connect with each other using multithreading.

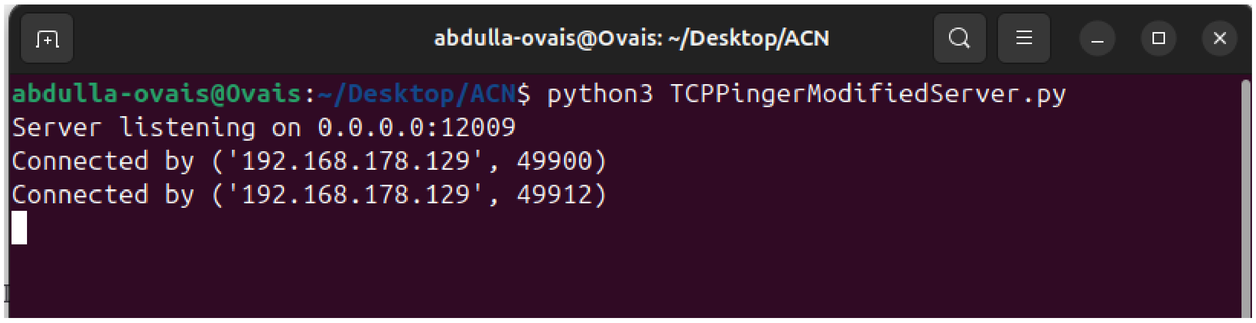
2. Then the multiple clients send the packet to the server.

3. Server sends the response to the multiple clients and calculates the RTT,min.RTT,max.RTT,avg.RTT and packet loss Rate.

Below in fig: It shows multiple clients are connected with server.

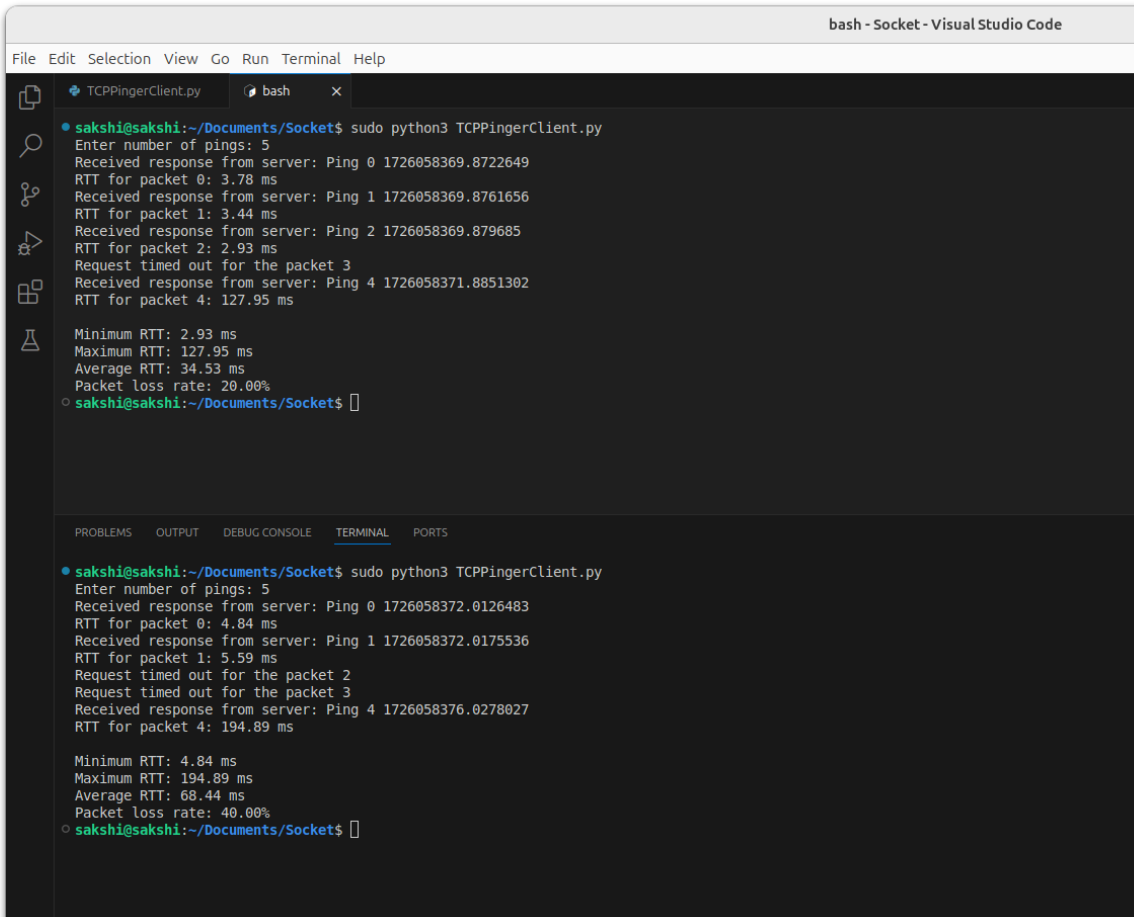
CASE 2: TCP Client to Modified TCP server:

Here the TCP server is modified to handle concurrent client connection, using multithreading in TCP server socket.



```
abdulla-ovais@Ovais:~/Desktop/ACN$ python3 TCPPingerModifiedServer.py
Server listening on 0.0.0.0:12009
Connected by ('192.168.178.129', 49900)
Connected by ('192.168.178.129', 49912)
```

Fig: Displaying Modified TCP Pinger Server running, where two different clients are connected to the given server.


bash - Socket - Visual Studio Code
File Edit Selection View Go Run Terminal Help
TCPPingerClient.py
bash
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
sakshi@sakshi:~/Documents/Socket$ sudo python3 TCPPingerClient.py
Enter number of pings: 5
Received response from server: Ping 0 1726058369.8722649
RTT for packet 0: 3.78 ms
Received response from server: Ping 1 1726058369.8761656
RTT for packet 1: 3.44 ms
Received response from server: Ping 2 1726058369.879685
RTT for packet 2: 2.93 ms
Request timed out for the packet 3
Received response from server: Ping 4 1726058371.8851302
RTT for packet 4: 127.95 ms

Minimum RTT: 2.93 ms
Maximum RTT: 127.95 ms
Average RTT: 34.53 ms
Packet loss rate: 20.00%
sakshi@sakshi:~/Documents/Socket$ 
```

```
sakshi@sakshi:~/Documents/Socket$ sudo python3 TCPPingerClient.py
Enter number of pings: 5
Received response from server: Ping 0 1726058372.0126483
RTT for packet 0: 4.84 ms
Received response from server: Ping 1 1726058372.0175536
RTT for packet 1: 5.59 ms
Request timed out for the packet 2
Request timed out for the packet 3
Received response from server: Ping 4 1726058376.0278027
RTT for packet 4: 194.89 ms

Minimum RTT: 4.84 ms
Maximum RTT: 194.89 ms
Average RTT: 68.44 ms
Packet loss rate: 40.00%
sakshi@sakshi:~/Documents/Socket$ 
```

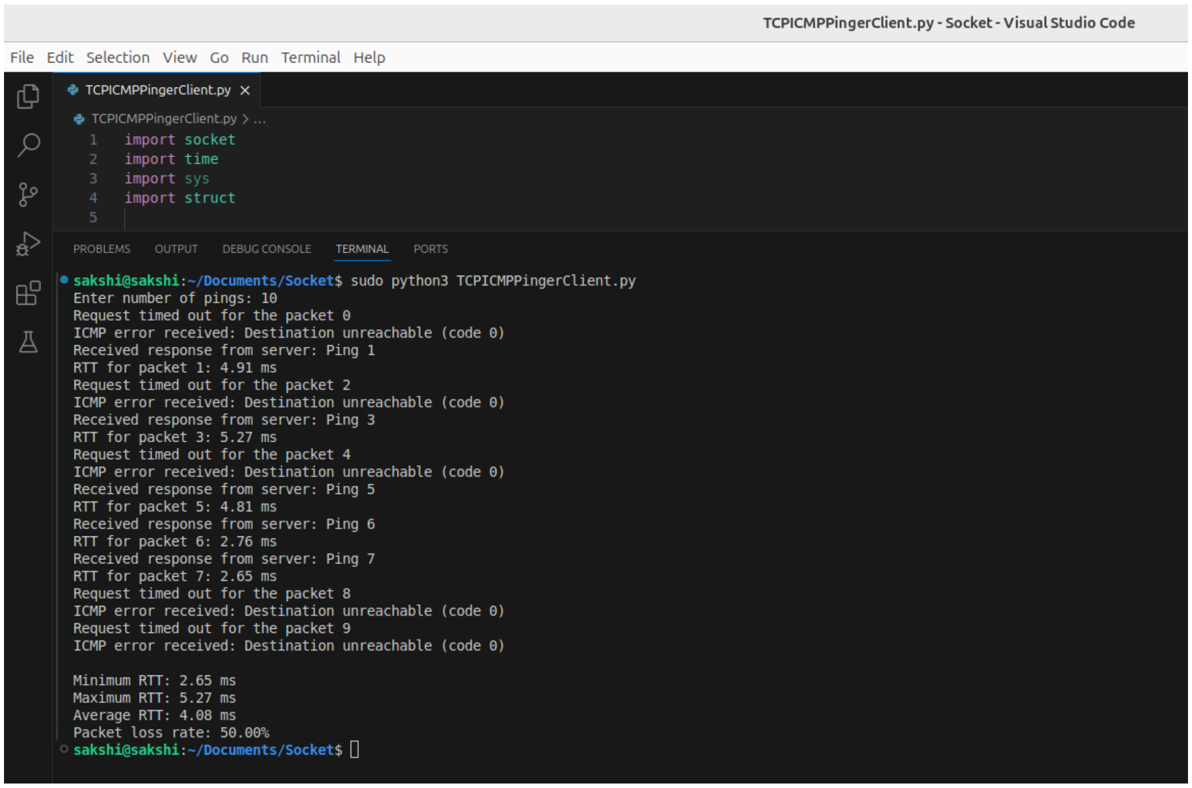
Fig: Displaying TCP Pinger Clients running in two separate terminals pinging a single TCP Server.

Workflow of TCP-CASE-3:

1. In this case First we made some changes in server-side code and client-side code in which the server sends the ICMP response message to the client with error code if any packet is lost.
2. Now connect the client with the server.
3. client sends the packet, and server sends the response in which calculates the RTT,min.RTT,max.RTT,avg.RTT and packet loss Rate.

CASE 3: Modified TCP Client to Modified TCP server:

Here, the client and server are modified so that they can receive and send ICMP error messages. An ICMP socket is added to both the codes so as to aid the ICMP error message communication.
The server packs and sends an ICMP error message through an ICMP socket and the client receives and unpacks the ICMP error message using its code and displays the error message only in the case where before there used to be request-timed-out.


TCPICMPPingerClient.py - Socket - Visual Studio Code
File Edit Selection View Go Run Terminal Help
TCPICMPPingerClient.py
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```python
import socket
import time
import sys
import struct
```

```
sakshi@sakshi:~/Documents/Socket$ sudo python3 TCPICMPPingerClient.py
Enter number of pings: 10
Request timed out for the packet 0
ICMP error received: Destination unreachable (code 0)
Received response from server: Ping 1
RTT for packet 1: 4.91 ms
Request timed out for the packet 2
ICMP error received: Destination unreachable (code 0)
Received response from server: Ping 3
RTT for packet 3: 5.27 ms
Request timed out for the packet 4
ICMP error received: Destination unreachable (code 0)
Received response from server: Ping 5
RTT for packet 5: 4.81 ms
Received response from server: Ping 6
RTT for packet 6: 2.76 ms
Received response from server: Ping 7
RTT for packet 7: 2.65 ms
Request timed out for the packet 8
ICMP error received: Destination unreachable (code 0)
Request timed out for the packet 9
ICMP error received: Destination unreachable (code 0)

Minimum RTT: 2.65 ms
Maximum RTT: 5.27 ms
Average RTT: 4.08 ms
Packet loss rate: 50.00%
sakshi@sakshi:~/Documents/Socket$
```

Fig: Displaying TCP Pinger Client running with ICMP error message displayed only for the dropped packets, otherwise it calculates the min., max., avg. RTTs as well as packet loss percent.

PART 3: ICMP Pinger

Here, we only needed to create an ICMP pinger client as almost every OS has built-in ICMP functionality.

WORKFLOW FOR ICMP:

The ICMP pinger client calls ping function with the ip address of the server and the number of pings, the ping function calls doOnePing function for n = number of pings times. DoOnePing function creates a raw socket and calls sendOnePing function with the parameters icmp_socket, destination address and process ID(known by using os.getpid()).

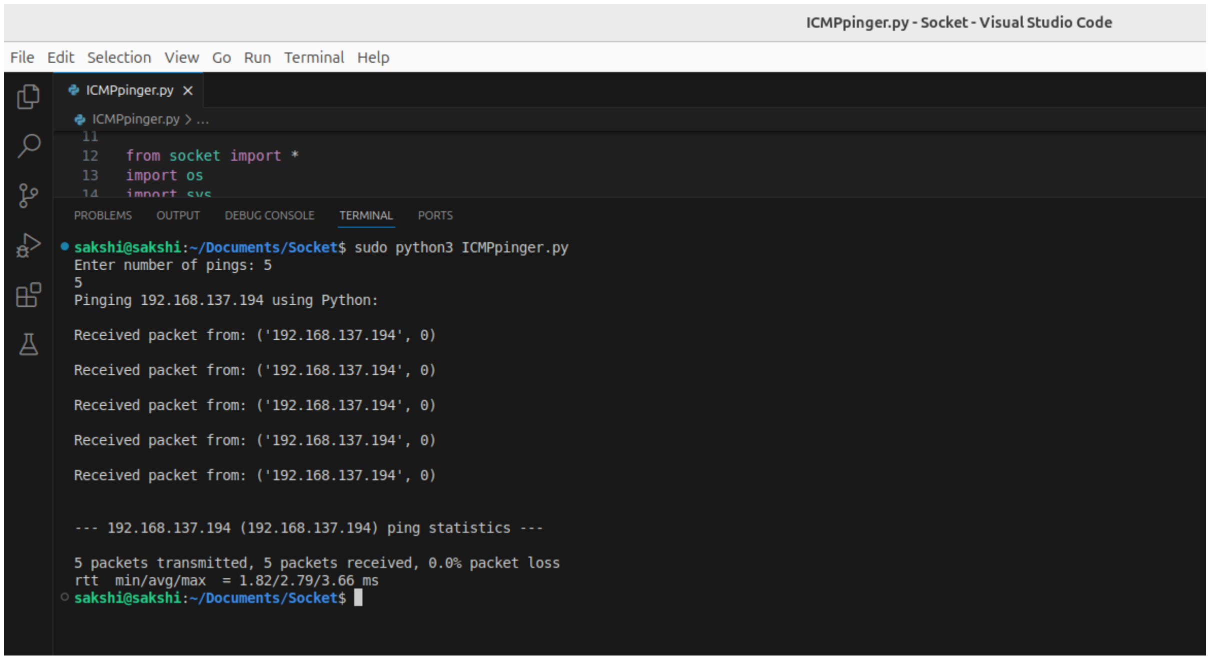
SendOnePing creates an ICMP packet with header(with format “bbHHh”) and data(“d” and timestamp). For the header checksum is calculated using checksum function which is then added to the header and the icmp packet is sent through the icmp socket to the server with packet, and destination address.
Here, the bbHHh refers to b-signed integer of 1 byte(ICMP type and code), H – unsigned short of 2 bytes(checksum and process id), h – signed char 2 bytes(sequence).

ReceiveOnePing function accepts parameters icmp socket, process ID, time-out and destination address. It receives the ICMP packet and unpacks it and abstract the data from it, i.e. ICMP type, ICMP code and it also calculates the delay for successful ICMP responses and returns the delay or the error response according to the received packet.

Ping function receives this delay for each ping and appends it to a global list and when there's an AttributeError(when a string is returned such as "Request timed-out" or when TypeError(None is received in case of ICMP error response) it does not append to the delay list but just increases the packet loss count. It prints the Maximum, Minimum, Average RTTs using the delay list and prints the packet loss percentage using the packet loss variable which increments whenever the delay returned goes to Attribute error.

CASE 1: ICMP client (min, max, avg RTTs, packet loss percentage calculated)

ICMP pinger client skeleton has been modified so that now it calculates min RTT, max RTT, avg RTT and packet loss percentage for the user input value for the number of pings.


ICMPpinger.py - Socket - Visual Studio Code
File Edit Selection View Go Run Terminal Help
ICMPpinger.py
ICMPpinger.py > ...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
from socket import *
import os
import sys
```

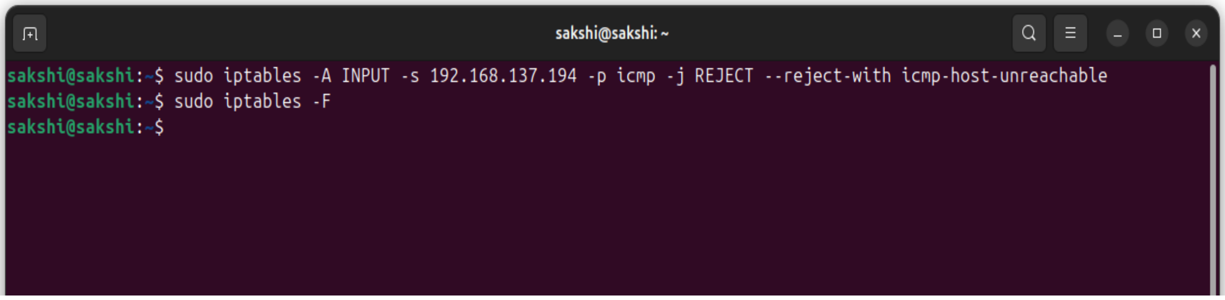
```
sakshi@sakshi:~/Documents/Socket$ sudo python3 ICMPpinger.py
Enter number of pings: 5
5
Pinging 192.168.137.194 using Python:

Received packet from: ('192.168.137.194', 0)

Received packet from: ('192.168.137.194', 0)

Received packet from: ('192.168.137.194', 0)

Received packet from: ('192.168.137.194', 0)

Received packet from: ('192.168.137.194', 0)


--- 192.168.137.194 (192.168.137.194) ping statistics ---

5 packets transmitted, 5 packets received, 0.0% packet loss
rtt  min/avg/max  = 1.82/2.79/3.66 ms
sakshi@sakshi:~/Documents/Socket$
```

Fig: Displaying ICMP Pinger Client running with calculated min, max, avg RTTs and packet loss percentage .

CASE 2:

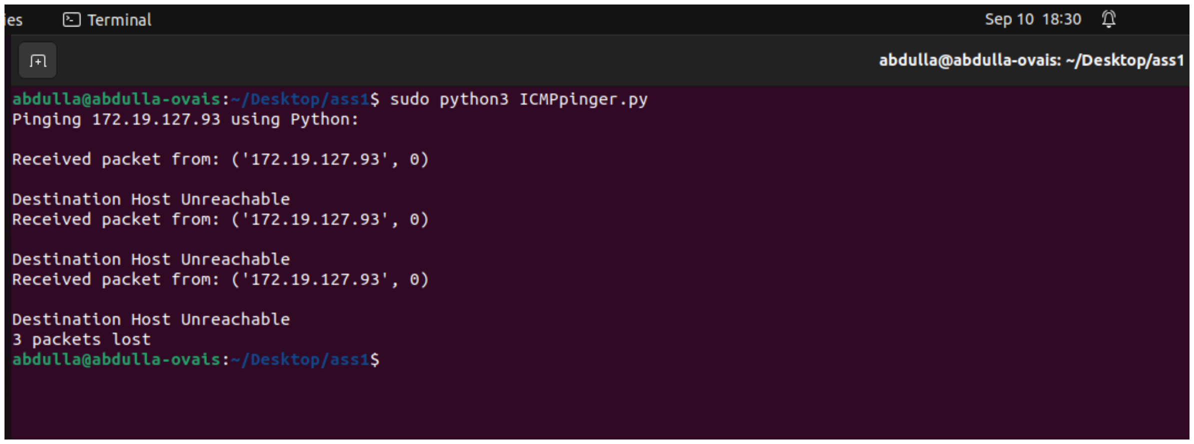
The second modification was to capture and display the ICMP error messages received through the ICMP raw socket by our client when the following command is run over on the server machine:
sudo iptables -A INPUT -s <source_ip_address> -p icmp -j REJECT --reject-with <error_response>

This command configures the IP table rules in the server side machine and now replies with the selected ICMP error message.



```
sakshi@sakshi:~$ sudo iptables -A INPUT -s 192.168.137.194 -p icmp -j REJECT --reject-with icmp-host-unreachable
sakshi@sakshi:~$ sudo iptables -F
sakshi@sakshi:~$
```

Fig: Displaying terminal with iptables command being run on the server side terminal , where 192.168.137.194 is the client IP address and the error message is icmp-host-unreachable.



```
abdulla@abdulla-ovais:~/Desktop/ass1$ sudo python3 ICMPpinger.py
Pinging 172.19.127.93 using Python:

Received packet from: ('172.19.127.93', 0)

Destination Host Unreachable
Received packet from: ('172.19.127.93', 0)

Destination Host Unreachable
Received packet from: ('172.19.127.93', 0)

Destination Host Unreachable
3 packets lost
abdulla@abdulla-ovais:~/Desktop/ass1$
```

Fig: ICMP pinger client with ICMP error message being displayed on the client side terminal, the server runs on IP 172.19.127.93.

ANTI-PLAGIARISM Statement

We certify that this assignment/report is the result of our collaborative work, based on our collective study and research. All sources, including books, articles, software, datasets, reports, and communications, have been properly acknowledged. This work has not been previously submitted for assessment in any other course unless specific permission was granted by all involved instructors.

We also acknowledge the use of AI tools, such as LLMs (e.g., ChatGPT), for assistance in refining this assignment, if used. We have ensured that their usage complies with the academic integrity policies of this course. We pledge to uphold the principles of honesty, integrity, and responsibility at CSE@IITH. Additionally, we understand our duty to report any violations of academic integrity by others if we become aware of them.

Abdulla Ovais - CS24MTECH12014
Sakshi Badole - CS24MTECH11008
Simeon Sahasramsa Gajula - SM24MTECH11002
Date: 11/09/2024
Signatures: A.O. , S.B. , S.S.G

Task/Section	**Group Member <Enter Name of Student>**	**Contribution <describe the work done, in brief>**	**Challenges faced (if any).**
Research & Info gathering	Sakshi Badole	Went through all the tutorial links, provided more python socket tutorials for the team.	Learning and finding about raw sockets.
Code Development	Simeon	Coded UDP client and modified UDP server (without random integer)	
	Sakshi Badole	Coded ICMP Client, UDP ICMP error client, UDP ICMP error server	Issue with the python global list, exception thrown when the string/None value was returned
	Abdulla Ovais	Coded TCP client, TCP Server, TCP Modified server with concurrent processes, TCP ICMP server, TCP ICMP client.	
Testing & Debugging	Sakshi Badole Abdulla Ovais	Tested and noted outputs for all the programs	
Documentation & Report Writing	Abdulla Ovais	README file Report – “Workflow” of UDP and TCP	
	Sakshi Badole	Report – Brief description of PART 1,2 and 3, arrangement of screenshots, their labels and outputs. “Workflow” of ICMP.	
Final Review & Submission			