

# ACN Programming-Assignment-1-My Pingers: My Pingers

## Students:

1. Vinaykumar Kadari (CS24MTECH14008)
2. P.Rushi Keswar Reddy (CS24MTECH11018)
3. Kishor Kumar Patro (SM24MTECH14001)

## Part 1: UDP Pinger

### Implementation of UDP Pinger Client :

The UDP pinger client program is implemented by using socket programming. The program takes N as input, which indicates the number of pings the client wants to send to the server. A UDP socket is created, which works on an IPv4 address. To avoid waiting a long time to receive a response from the server, a timeout is set to the client socket for 1 second. An array rtt\_time is declared to store the Round trip times of each ping request response. Using Exception Handling in Python, errors are handled by printing out exceptions.

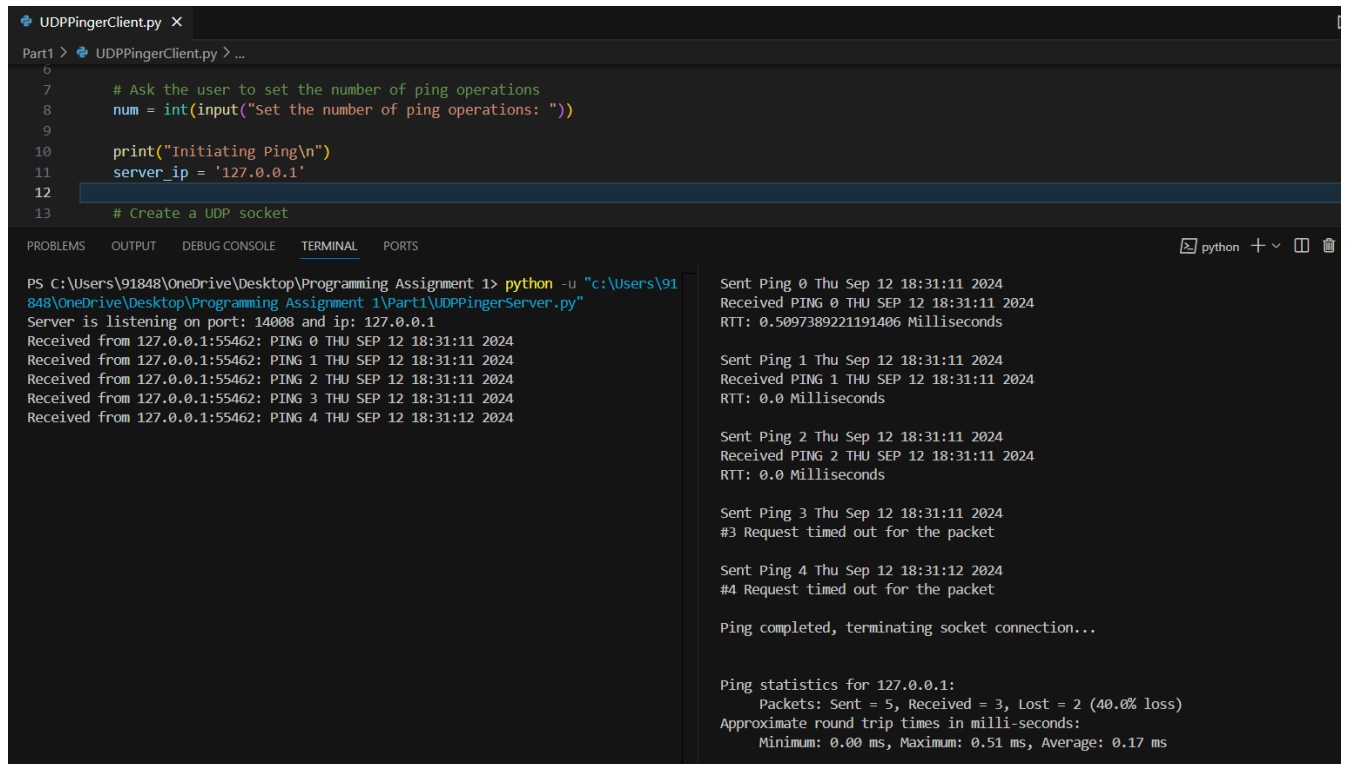
- The internal implementation of sending each ping Request:
- For loop is used to iterate N times. The messages are sent in the following format:

**Ping + ith ping + time\_stamp**

- For calculating the RTT of each ping request, the time module is utilized to get timestamps before sending the request(start) and after getting the response(end). Using sendto method of client socket, encode message is sent to server\_address using UDP protocol.
- Using recvfrom method of client socket, response of the request sent is received and time elapsed is determined by subtracting the time stamps start from end. Finally, rtt is calculated and appended into the list.
- If, during the above process, any ping response faces a timeout, an exception is raised, and that request is considered packet loss.
- After n ping requests, the created client socket is closed.
- The minimum, maximum, and average RTTs are calculated from the data.

## 1) Local Host

The given UDPPingerServer and implemented UDPPingerClient programs are executed on the same machine with the local host having an address as “127.0.0.1” for debugging purposes. Therefore the round trip time (RTTs) will be low and consistent.



```
UDPPingerClient.py X
Part1 > UDPPingerClient.py > ...
6
7 # Ask the user to set the number of ping operations
8 num = int(input("Set the number of ping operations: "))
9
10 print("Initiating Ping\n")
11 server_ip = '127.0.0.1'
12
13 # Create a UDP socket

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\91848\OneDrive\Desktop\Programming Assignment 1> python -u "c:\Users\91848\OneDrive\Desktop\Programming Assignment 1\Part1\UDPPingerServer.py"
Server is listening on port: 14008 and ip: 127.0.0.1
Received from 127.0.0.1:55462: PING 0 THU SEP 12 18:31:11 2024
Received from 127.0.0.1:55462: PING 1 THU SEP 12 18:31:11 2024
Received from 127.0.0.1:55462: PING 2 THU SEP 12 18:31:11 2024
Received from 127.0.0.1:55462: PING 3 THU SEP 12 18:31:11 2024
Received from 127.0.0.1:55462: PING 4 THU SEP 12 18:31:12 2024

Sent Ping 0 Thu Sep 12 18:31:11 2024
Received PING 0 THU SEP 12 18:31:11 2024
RTT: 0.5097389221191406 Milliseconds

Sent Ping 1 Thu Sep 12 18:31:11 2024
Received PING 1 THU SEP 12 18:31:11 2024
RTT: 0.0 Milliseconds

Sent Ping 2 Thu Sep 12 18:31:11 2024
Received PING 2 THU SEP 12 18:31:11 2024
RTT: 0.0 Milliseconds

Sent Ping 3 Thu Sep 12 18:31:11 2024
#3 Request timed out for the packet

Sent Ping 4 Thu Sep 12 18:31:12 2024
#4 Request timed out for the packet

Ping completed, terminating socket connection...

Ping statistics for 127.0.0.1:
    Packets: Sent = 5, Received = 3, Lost = 2 (40.0% loss)
    Approximate round trip times in milli-seconds:
        Minimum: 0.00 ms, Maximum: 0.51 ms, Average: 0.17 ms
```

## 2) Single Client

### Simulation of UDP packet Loss:

A random module in Python is used to simulate the packet loss at the server end.

### Packet loss simulation using random module:

Random integers between 1 to 10 are generated, if the generated integer is more than 8, then it is considered as packet loss, and the server won't send the response packet back to the client. Given server code is modified a little bit to achieve the above simulation.

## Server Output:

```
Part1 > UDPPingerServer.py > ...
1 # UDPPingerServer.py
2 # We will need the following module to generate randomized lost packets
3 import random
4 from socket import *
5
6 # Create a UDP socket
7 # Notice the use of SOCK_DGRAM for UDP packets
8 serverSocket = socket(AF_INET, SOCK_DGRAM)
9 server_ip='172.21.132.171'
10 # Assign IP address and port number to socket
11 serverSocket.bind((server_ip, 14008))
12
13 print(f"Server is listening on port: {14008} and ip: {server_ip}")
14
15 while True:
16     # Generate a random number between 1 to 10 (both inclusive)
17     rand = random.randint(1, 10)
18
19     # Receive the client packet along with the address it is coming from
20     message, address = serverSocket.recvfrom(1024)
21     message=message.decode('utf-8')
22
23     # Capitalize the message from the client
24     message = message.upper()
25     print(f"Received from {address[0]}:{address[1]}: {message}")
26     # If rand is greater than 8, we consider the packet lost and do not respond
27     if rand > 8:
28         continue
29
30     # Otherwise, the server response
31     serverSocket.sendto(message.encode('utf-8'), address)
```

```
Assmns\Programming Assignment 1> & "C:/Program Files/Python310/python.exe" "c:/Users/prush/Desktop/sem-1/Advanced Computer Networks/Computer_networks/Assmns/Programming Assignment 1/Part1/UDPPingerServer.py"
Server is listening on port: 14008 and ip: 172.21.132.171
Received from 172.21.135.35:62614: PING 0 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 1 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 2 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 3 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 4 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 5 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 6 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 7 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 8 THU SEP 12 17:34:00 2024
Received from 172.21.135.35:62614: PING 9 THU SEP 12 17:34:01 2024
Received from 172.21.135.35:62614: PING 10 THU SEP 12 17:34:02 2024
Received from 172.21.135.35:62614: PING 11 THU SEP 12 17:34:02 2024
Received from 172.21.135.35:62614: PING 12 THU SEP 12 17:34:03 2024
Received from 172.21.135.35:62614: PING 13 THU SEP 12 17:34:04 2024
Received from 172.21.135.35:62614: PING 14 THU SEP 12 17:34:04 2024
```

## Client Output:

```
UDPPingerClient.py > ...
1 # UDPPingerClient.py
2 import socket
3 import time # Import time library
4
5 while True:
6
7     # Ask the user to set the number of ping operations
8     num = int(input("Set the number of ping operations: "))
9
10    print(f"Initiating Ping's")
11
12    # ... (rest of the client code) ...
```

PROBLEMS OUTPUT DEBUG\_CONSOLE TERMINAL PORTS

```
Sent Ping 10 Thu Sep 12 17:34:02 2024
Received PING 10 THU SEP 12 17:34:02 2024
RTT: 114.65334892272949 Milliseconds

Sent Ping 11 Thu Sep 12 17:34:02 2024
#11 Request timed out for the packet

Sent Ping 12 Thu Sep 12 17:34:03 2024
#12 Request timed out for the packet

Sent Ping 13 Thu Sep 12 17:34:04 2024
Received PING 13 THU SEP 12 17:34:04 2024
RTT: 7.237672805786133 Milliseconds

Sent Ping 14 Thu Sep 12 17:34:04 2024
Received PING 14 THU SEP 12 17:34:04 2024
RTT: 4.721879959106445 Milliseconds

Ping completed, terminating socket connection...
```

```
Ping statistics for 172.21.132.171:
Packets: Sent = 15, Received = 11, Lost = 4 (26.66666666666668% loss)
Approximate round trip times in milli-seconds:
Minimum: 3.61 ms, Maximum: 114.65 ms, Average: 18.78 ms
```

### 3) Client Killed

```
UDPPingerClient.py X
Part1 > UDPPingerClient.py > ...
6
7     # Ask the user to set the number of ping operations
8     num = int(input("Set the number of ping operations: "))
9
10    print("Initiating Ping\n")
11    server_ip = '172.21.149.80'
12
13    # Create a UDP socket
14    client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
15
16    server_address = (server_ip, 14008) # Set IP Address and Port Number of Socket

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Sent Ping 7 Thu Sep 12 19:47:25 2024
Received PING 7 THU SEP 12 19:47:25 2024
RTT: 19.92177963256836 Milliseconds

Sent Ping 8 Thu Sep 12 19:47:25 2024
Received PING 8 THU SEP 12 19:47:25 2024
RTT: 9.031534194946289 Milliseconds

Sent Ping 9 Thu Sep 12 19:47:25 2024
Ping completed, terminating socket connection...
Traceback (most recent call last):
  File "c:\Users\91848\OneDrive\Desktop\Programming Assignment 1\Part1\UDPPingerClient.py", line 31, in <module>
    data, server = client.recvfrom(4096) # Maximum data received 4096 bytes
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
TimeoutError: timed out

During handling of the above exception, another exception occurred:

Traceback (most recent call last):
  File "c:\Users\91848\OneDrive\Desktop\Programming Assignment 1\Part1\UDPPingerClient.py", line 38, in <module>
    print("#" + str(i) + " Request timed out for the packet\n")
    ^^^^^^
KeyboardInterrupt
PS C:\Users\91848\OneDrive\Desktop\Programming Assignment 1> ^C
```

### 4) Emulation of UDP packet loss at NIC by tc-netem

To emulate packet loss at NIC of the machine where server code is running, the following commands are

**sudo tc qdisc add dev enp0s3 root netem loss 20%**

This command sets up 20% packet loss on the wifi (ethernet) interface enp0s3 of the machine running server code.

**tc qdisc show dev enp0s3**

This command verifies the traffic control rule at the Wi-Fi interface

**sudo tc qdisc del dev enp0s3 root**

This command is finally used to delete the setup loss at NIC.

```
vinaykadari@vinaykadari: ~  
vinaykadari@vinaykadari:~$ sudo tc qdisc add dev lo root netem loss 20%  
vinaykadari@vinaykadari:~$
```

## Server Output:

```
UDPPingerModifiedServer.py x  
Desktop > Computer_Networks > Program_A1 (2) > Program_A1 > Part1 > UDPPingerModifiedServer.py > ...  
6 # Create a UDP socket  
7 # Notice the use of SOCK_DGRAM for UDP packets  
8 serverSocket = socket(AF_INET, SOCK_DGRAM)  
9 server_ip='172.21.132.191'  
10 # Assign IP address and port number to socket  
11 serverSocket.bind((server_ip, 14000))  
12  
13 print(f"Server is listening on port: {14000} and ip: {server_ip}")  
14  
15 while True:  
16     # Generate a random number between 1 to 10 (both inclusive)  
17     # rand = random.randint(1, 10)  
18  
19     # Receive the client packet along with the address it is coming from  
20     message, address = serverSocket.recvfrom(1024)  
21  
22     # Capitalize the message from the client  
23     message = message.upper()  
24     print(f"Received from {address[0]}:{address[1]}: {message.decode('utf-8')}")  
25     # If rand is greater than 8, we consider the packet lost and do not respond to the client  
26     # if rand > 8:  
27     #     continue  
28  
29     # Otherwise, the server response  
30     serverSocket.sendto(message, address)  
  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
/bin/python3 "/home/rushi/Desktop/Computer_Networks/Program_A1 (2)/Program_A1/Part1/UDPPingerModifiedServer.py"  
rushi@rushi-VirtualBox:~$ /bin/python3 "/home/rushi/Desktop/Computer_Networks/Program_A1 (2)/Program_A1/Part1/UDPPingerModifiedServer.py"  
Server is listening on port: 14000 and ip: 172.21.132.191  
Received from 172.21.132.192:38230: PING 0 WED SEP 11 20:01:26 2024  
Received from 172.21.132.192:38230: PING 1 WED SEP 11 20:01:26 2024  
Received from 172.21.132.192:38230: PING 2 WED SEP 11 20:01:26 2024  
Received from 172.21.132.192:38230: PING 3 WED SEP 11 20:01:26 2024  
Received from 172.21.132.192:38230: PING 4 WED SEP 11 20:01:26 2024  
Received from 172.21.132.192:38230: PING 5 WED SEP 11 20:01:26 2024  
Received from 172.21.132.192:38230: PING 6 WED SEP 11 20:01:26 2024  
Received from 172.21.132.192:38230: PING 7 WED SEP 11 20:01:27 2024  
Received from 172.21.132.192:38230: PING 8 WED SEP 11 20:01:27 2024  
Received from 172.21.132.192:38230: PING 9 WED SEP 11 20:01:27 2024
```

## Client Output:

```
UDPPingerClient.py X
Part1 > UDPPingerClient.py > ...
13 # Create a UDP socket
14 client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
15
16 server_address = (server_ip, 14000) # Set IP Address and Port Number of Socket
17
18 client.settimeout(1) # Sets a timeout value of 1 second
19
20 rtt_time = [] # List to store Round-Trip Times (RTTs)
21 packet_lost = 0 # Count of lost (timed-out) pings
22
23 try:
24     # Loop to ping the server 'num_pings' times
25     for i in range(num):
26         start = time.time() # Start time when message is sent to server
27         message = 'Ping ' + str(i) + " " + time.ctime(start)
28         try:
29             sent = client.sendto(message.encode("utf-8"), server_address)
30             print("Sent " + message)
31             data, server = client.recvfrom(4096) # Maximum data received 4096 bytes
32             print("Received " + str(data.decode("utf-8")))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Ping completed, terminating socket connection...

Ping statistics for 172.21.132.191:  
Packets: Sent = 10, Received = 9, Lost = 1 (10.0% loss)  
Approximate round trip times in milli-seconds:  
Minimum: 4.22 ms, Maximum: 57.62 ms, Average: 10.95 ms

## 5) Error Handling

As done in packet loss simulation, the random module simulates ICMP error responses. Randint function generates integers from 1 to 10. If the value generated is less than or equal to 6 then the response is received successfully. If the value is greater than 6 and less than or equal to 8 then it is considered as "ICMP Destination Unreachable," and if the value is greater than 8 then it is considered as "ICMP Port Unreachable". For these cases, a raw socket is created to send an ICMP error packet to the client.

## Server output

```
UDP_ICMP_Server.py X
Part1 > UDP_ICMP_Server.py > ...
12 print(f"Server is listening on port: {server_port} and IP: {server_ip}")
13
14 # Function to send ICMP Destination Unreachable or Port Unreachable
15 def send_icmp_error(dest_addr, error_type, code):
16     sock = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)
17     sock.setsockopt(socket.IPPROTO_IP, socket.IP_HDRINCL, 1)
18
19     packet_id = os.getpid() & 0xFFFF
20     header = struct.pack('!BBHH', error_type, code, 0, packet_id, 1)
21
22     # Include dummy IP header for compatibility with some systems
23     ip_header = struct.pack('!BBHH4s4s',
24                             69, 0, 84, 54321, 0, 255, socket.IPPROTO_ICMP, 0,
25                             socket.inet_aton(server_ip),
26                             socket.inet_aton(dest_addr))
27
28     # Send ICMP error packet to client
29     sock.sendto(ip_header + header, (dest_addr, 0))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
sock = socket.socket(socket.AF_INET, socket.SOCK_RAW, socket.IPPROTO_ICMP)
File "/usr/lib/python3.8/socket.py", line 231, in _init_
socket.socket._init_(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted
● rushi@rushy-VirtualBox:~/Desktop/Computer_Networks/My_Pingers_Files/Program_A1$ cd Part1
○ rushi@rushy-VirtualBox:~/Desktop/Computer_Networks/My_Pingers_Files/Program_A1/Part1$ sudo python3 UDP_ICMP_Server.py
[sudo] password for rushi:
Server is listening on port: 14009 and IP: 172.21.132.191
Received from 172.21.132.192:48122: PING 0 THU SEP 12 17:47:31 2024
Received from 172.21.132.192:48122: PING 1 THU SEP 12 17:47:32 2024
Received from 172.21.132.192:48122: PING 2 THU SEP 12 17:47:32 2024
Sending ICMP Port Unreachable to 172.21.132.192
Received from 172.21.132.192:48122: PING 3 THU SEP 12 17:47:33 2024
Received from 172.21.132.192:48122: PING 4 THU SEP 12 17:47:33 2024
Sending ICMP Port Unreachable to 172.21.132.192
Received from 172.21.132.192:48122: PING 5 THU SEP 12 17:47:34 2024
Sending ICMP Destination Unreachable to 172.21.132.192
Received from 172.21.132.192:48122: PING 6 THU SEP 12 17:47:35 2024
Received from 172.21.132.192:48122: PING 7 THU SEP 12 17:47:35 2024
Received from 172.21.132.192:48122: PING 8 THU SEP 12 17:47:35 2024
Sending ICMP Port Unreachable to 172.21.132.192
Received from 172.21.132.192:48122: PING 9 THU SEP 12 17:47:36 2024
□
```

## Client Output:

```
UDP_icmp_client.py X
Part1 > UDP_icmp_client.py > ...
45 # Check for ICMP error messages
46 icmp_data, icmp_addr = raw_socket.recvfrom(1024)
47 icmp_type, icmp_code = parse_icmp_packet(icmp_data)
48 if icmp_type == ICMP_DEST_UNREACHABLE:
49     if icmp_code == 0:
50         print(f"ICMP Error: Destination Unreachable from {icmp_addr[0]}")
51         packet_lost += 1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Sent Ping 6 Thu Sep 12 17:47:35 2024
Received PING 6 THU SEP 12 17:47:35 2024 from ('172.21.132.191', 14009)
RTT: 98.725 ms

Sent Ping 7 Thu Sep 12 17:47:35 2024
Received PING 7 THU SEP 12 17:47:35 2024 from ('172.21.132.191', 14009)
RTT: 5.432 ms

Sent Ping 8 Thu Sep 12 17:47:35 2024
ICMP Error: Port Unreachable from 172.21.132.191
RTT: 1005.226 ms

Sent Ping 9 Thu Sep 12 17:47:36 2024
Received PING 9 THU SEP 12 17:47:36 2024 from ('172.21.132.191', 14009)
RTT: 10.831 ms

Ping completed, terminating socket connection...
Ping statistics for 172.21.132.191:
    Packets: Sent = 10, Received = 6, Lost = 4 (40.0% loss)
Approximate round trip times in milli-seconds:
    Minimum = 5.43 ms, Maximum = 1005.23 ms, Average = 427.13 ms
```

## **Part 2: TCP Pinger**

### **Implementation of TCP Pinger Client and Server:**

In this packet we are using TCP socket instead of UDP. TCP provides a **reliable** and **connection-oriented communication** channel. Both client and server pingers are implemented. Along with this, multithreading is also incorporated into server code to deal with multiple clients concurrently.

#### **Outline of the TCP pinger client Implementation:**

- a) Creation of socket.
- b) Establishment of the TCP connection with server address and port number.
- c) Sending Ping requests to the server.
- d) Response receiving and Error handling in case of Exception
- e) Calculation of minimum, maximum, and Average RTT values. Packet loss rate is also calculated based on the number of packets lost.
- f) Closing Connection and Socket.

#### **Outline of the TCP server Implementation:**

- a) Creation of Server Socket.
- b) Binding server address and Port and start listening for incoming connection.
- c) Connection acceptance and processing of request
- d) Sending response and Error Handling
- e) Closing connection

#### **Results:**

##### **1) Local Host**

Running both TCP pinger server and TCP pinger client both on same local machine with local server address "127.0.0.1".



```
TCPPingerClient.py X
Part2 > TCPPingerClient.py > ...
5 client = socket(AF_INET, SOCK_STREAM)
6
7 # Set a timeout of 1 second
8 client.settimeout(1)
9 server_ip='127.0.0.1'
10 # Server address and port
11 server_address = (server_ip, 14008)
12

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS powershell + - [ ] [ ]

PS C:\Users\91848\OneDrive\Desktop\Programming Assignment 1> python -u "c:\Users\91848\OneDrive\Desktop\Programming Assignment 1\Part2\TCPPingerServer.py"
TCP server up and listening...
Connection established with ('127.0.0.1', 53826)
Packet from ('127.0.0.1', 53826) responded: PING 0 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 1 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 2 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 3 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 4 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 5 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 6 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 7 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 8 THU SEP 12 18:33:30 2024
Packet from ('127.0.0.1', 53826) responded: PING 9 THU SEP 12 18:33:30 2024
Connection with ('127.0.0.1', 53826) closed.

Sent Ping 5 Thu Sep 12 18:33:30 2024
Received PING 5 THU SEP 12 18:33:30 2024

Sent Ping 6 Thu Sep 12 18:33:30 2024
Received PING 6 THU SEP 12 18:33:30 2024

Sent Ping 7 Thu Sep 12 18:33:30 2024
Received PING 7 THU SEP 12 18:33:30 2024

Sent Ping 8 Thu Sep 12 18:33:30 2024
Received PING 8 THU SEP 12 18:33:30 2024

Sent Ping 9 Thu Sep 12 18:33:30 2024
Received PING 9 THU SEP 12 18:33:30 2024

Ping statistics for 127.0.0.1:
    Packets: Sent = 9, Received = 9, Lost = 0 (0.0% loss)
    Approximate round trip times in milli-seconds:
        Minimum: 0.00 ms, Maximum: 0.00 ms, Average: 0.00 ms
```

## 2) Single Client

Server code and client code are executed on two different machines connected to the same network. The packet loss is simulated by random() function of random module in python.

### Server Output:

```
PS C:\Users\prush\Desktop\sem-1\Advanced Computer Networks\Computer_networks\Assmns> python -u "c:\Users\prush\Desktop\sem-1\Advanced Computer Networks\Computer_networks\Assmns\Program_A1\Part2\TCPPingerServer.py"
TCP server up and listening...
Connection established with ('172.21.135.35', 60903)
Packet from ('172.21.135.35', 60903) responded: PING 0 WED SEP 11 18:09:36 2024
Packet from ('172.21.135.35', 60903) responded: PING 1 WED SEP 11 18:09:36 2024
Packet from ('172.21.135.35', 60903) responded: PING 2 WED SEP 11 18:09:36 2024
Packet from ('172.21.135.35', 60903) responded: PING 3 WED SEP 11 18:09:36 2024
Packet from ('172.21.135.35', 60903) responded: PING 4 WED SEP 11 18:09:37 2024
Packet from ('172.21.135.35', 60903) lost (rand=10)
Packet from ('172.21.135.35', 60903) responded: PING 6 WED SEP 11 18:09:38 2024
Connection with ('172.21.135.35', 60903) closed.
```

## Client Output:

As observed below, there is one packet lost occurred. The RTT statistics of all Ping Packets are displayed.

```
TCPPingerClient.py X
Part2 > TCPPingerClient.py > ...
1  import time
2  from socket import *
3
4  # Create a TCP socket
5  client = socket(AF_INET, SOCK_STREAM)
6
7  # Set a timeout of 1 second
8  client.settimeout(1)
9  server_ip='172.21.132.171'
10 # Server address and port
11 server_address = (server_ip, 14008)
12
13 # Establish a connection to the server
14 client.connect(server_address)
15
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
#5 Request timed out for the packet

Sent Ping 6 Wed Sep 11 18:09:38 2024
Received PING 6 WED SEP 11 18:09:38 2024

Ping statistics for 172.21.132.171:
    Packets: Sent = 6, Received = 5, Lost = 1 (16.666666666666664% loss)
Approximate round trip times in milli-seconds:
    Minimum: 0.00 ms, Maximum: 0.04 ms, Average: 0.01 ms
```

### 3) Emulation of TCP packet loss at NIC by tc-netem

Instead of using a random module in Python, tc-netem utility in linux is used to inject a loss of 20% at NIC of machine at which server code is running. The following packet losses are observed.

#### Server Output:

```
TCP server up and listening...
Connection established with ('192.168.167.16', 51397)
Packet from ('192.168.167.16', 51397) responded: PING 1 THU SEP 12 20:39:46 2024
Packet from ('192.168.167.16', 51397) responded: PING 2 THU SEP 12 20:39:46 2024
Packet from ('192.168.167.16', 51397) responded: PING 3 THU SEP 12 20:39:46 2024
Packet from ('192.168.167.16', 51397) responded: PING 4 THU SEP 12 20:39:46 2024
Packet from ('192.168.167.16', 51397) responded: PING 5 THU SEP 12 20:39:46 2024
Packet from ('192.168.167.16', 51397) responded: PING 6 THU SEP 12 20:39:46 2024
Packet from ('192.168.167.16', 51397) responded: PING 10 THU SEP 12 20:39:49 2024
Connection with ('192.168.167.16', 51397) closed.
```

#### Client Output:

Out of 10 packets, nearly 4 packets are lost, which is simulated with tc-netem in linux.

```
TCPPingerClient.py X
Part2 > TCPPingerClient.py > ...
1  import time
2  from socket import *
3
4  # Create a TCP socket
5  client = socket(AF_INET, SOCK_STREAM)
6
7  # Set a timeout of 1 second
8  client.settimeout(1)
9  server_ip='192.168.167.187'
10 # Server address and port
11 server_address = (server_ip, 14008)
12
13 # Establish a connection to the server
14 client.connect(server_address)
15

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

Sent Ping 5 Thu Sep 12 20:39:46 2024
Received PING 5 THU SEP 12 20:39:46 2024

Sent Ping 6 Thu Sep 12 20:39:46 2024
Received PING 6 THU SEP 12 20:39:46 2024

Sent Ping 7 Thu Sep 12 20:39:46 2024
#7 Request timed out for the packet

Sent Ping 8 Thu Sep 12 20:39:47 2024
#8 Request timed out for the packet

Sent Ping 9 Thu Sep 12 20:39:48 2024
#9 Request timed out for the packet

Sent Ping 10 Thu Sep 12 20:39:49 2024
Received PING 10 THU SEP 12 20:39:49 2024

Ping statistics for 192.168.167.187:
    Packets: Sent = 10, Received = 6, Lost = 4 (40.0% loss)
    Approximate round trip times in milli-seconds:
        Minimum: 0.00 ms, Maximum: 0.22 ms, Average: 0.05 ms
```

## 4) Multi-Client TCP Server using Threading

To handle multiple clients, threading is used at the server end. Each thread is used to handle a single client. Based on the number of clients, those many threads are created by the os of the server to manage all connections concurrently

### Server

```
TCPPingerThreading.py
Program_A1 > Part2 > TCPPingerThreading.py > handle_client
6 def handle_client(connectionSocket, address):
28     print(f'Packet from {address} lost (rand={rand})')
29 else:
30     # Otherwise, the server responds
31     connectionSocket.send(message.encode('utf-8'))
32     print(f'Packet from {address} responded: {message}')
33
34 except Exception as e:
35     print(f'Error handling request from {address}: {e}')
36
37 finally:
38     # Close the connection with the client after the loop ends
39     connectionSocket.close()
40     print(f'Connection with {address} closed.')
41
42 # Create a TCP socket
43 serverSocket = socket(AF_INET, SOCK_STREAM)
44
45 # Bind the socket to the address and port
46 serverSocket.bind(('172.21.132.171', 14008))
47
48 # Start listening for incoming connections
49 serverSocket.listen(5)
50
51 print("TCP server up and listening...")
52 cnt = 0
53
54 while cnt < 10:
55     cnt += 1
56     # Accept a new client connection
57     connectionSocket, address = serverSocket.accept()
58
59     # Start a new thread to handle the client
60     client_thread = threading.Thread(target=handle_client, args=(connectionSocket, address))
61     client_thread.start()
62
63
```

```
PS C:\Users\prush\Desktop\sem-1\Advanced Computer Networks\Computer_networks\Assms> python -u "C:\Users\prush\Desktop\sem-1\Advanced Computer Networks\Computer_networks\Assms\Program_A1\Part2\TCPPingerThreading.py"
TCP server up and listening...
Connection established with ('172.21.135.35', 57616)
Packet from ('172.21.135.35', 57616) responded: PING 0 WED SEP 11 18:21:53 2024
Packet from ('172.21.135.35', 57616) responded: PING 1 WED SEP 11 18:21:53 2024
Packet from ('172.21.135.35', 57616) lost (rand=9)
Packet from ('172.21.135.35', 57616) responded: PING 3 WED SEP 11 18:21:54 2024
Packet from ('172.21.135.35', 57616) responded: PING 4 WED SEP 11 18:21:54 2024
Packet from ('172.21.135.35', 57616) responded: PING 5 WED SEP 11 18:21:54 2024
Packet from ('172.21.135.35', 50742) responded: PING 0 WED SEP 11 18:22:14 2024
Packet from ('172.21.135.35', 50742) lost (rand=10)
Packet from ('172.21.135.35', 50742) responded: PING 2 WED SEP 11 18:22:15 2024
Packet from ('172.21.135.35', 50742) responded: PING 3 WED SEP 11 18:22:15 2024
Packet from ('172.21.135.35', 50742) responded: PING 4 WED SEP 11 18:22:15 2024
Packet from ('172.21.135.35', 50742) lost (rand=10)
Packet from ('172.21.135.35', 50742) responded: PING 6 WED SEP 11 18:22:16 2024
Packet from ('172.21.135.35', 50742) lost (rand=10)
Packet from ('172.21.135.35', 50742) lost (rand=9)
Connection with ('172.21.135.35', 50742) closed.
```

**Client :** Multiple client requests through different port numbers to the server.

```
Client1.py
Part2 > Client1.py > ...
1 import time
2 from socket import *
3
4 # Create a TCP socket
5 client = socket(AF_INET, SOCK_STREAM)
6
7 # Set a timeout of 1 second
8 client.settimeout(1)
9 server_ip='172.21.132.171'
10 # Server address and port
11 server_address = (server_ip, 14008)
12
13 # Establish a connection to the server
14 client.connect(server_address)
15
16 # Ask the user to set the number of ping operations
17 num = int(input("Set the number of ping operations: "))

Client2.py
Part2 > Client2.py > ...
1 import time
2 from socket import *
3
4 # Create a TCP socket
5 client = socket(AF_INET, SOCK_STREAM)
6
7 # Set a timeout of 1 second
8 client.settimeout(1)
9 server_ip='172.21.132.171'
10 # Server address and port
11 server_address = (server_ip, 14008)
12
13 # Establish a connection to the server
14 client.connect(server_address)
15
16 # Ask the user to set the number of ping operations
17 num = int(input("Set the number of ping operations: "))
```

```
Sent Ping 5 Wed Sep 11 18:21:54 2024
#6 Request timed out for the packet
#6 Request timed out for the packet

Ping statistics for 172.21.132.171:
    Packets: Sent = 6, Received = 4, Lost = 2 (33.33333333333333% loss)
    Approximate round trip times in milli-seconds:
        Minimum: 0.00 ms, Maximum: 0.02 ms, Average: 0.01 ms

#7 Request timed out for the packet

Sent Ping 8 Wed Sep 11 18:22:17 2024
#8 Request timed out for the packet

Ping statistics for 172.21.132.171:
    Packets: Sent = 8, Received = 4, Lost = 4 (50.0% loss)
    Approximate round trip times in milli-seconds:
        Minimum: 0.00 ms, Maximum: 0.02 ms, Average: 0.01 ms
```

## 5) TCP with ICMP error handling

### Server

```
TCP_icmp_server.py X
Part2 > TCP_icmp_server.py > ...
21
22 # Bind the TCP socket
23 serverSocket = socket(AF_INET, SOCK_STREAM)
24 serverSocket.bind(('172.21.132.192', 14008))
25 serverSocket.listen(5)
26
27 # Wait for TCP connection and listen for it
28

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

• vinaykadari@vinaykadari:~/Downloads/Program_A1 (2)/Program_A1$ cd Part1
• vinaykadari@vinaykadari:~/Downloads/Program_A1 (2)/Program_A1/Part1$ cd ..
• vinaykadari@vinaykadari:~/Downloads/Program_A1 (2)/Program_A1$ cd Part2
• vinaykadari@vinaykadari:~/Downloads/Program_A1 (2)/Program_A1/Part2$ sudo python3 TCP_icmp_server.py
[sudo] password for vinaykadari:
TCP server up and listening...
Connection established with ('172.21.132.191', 52836)
Packet from ('172.21.132.191', 52836) responded: PING 0 THU SEP 12 18:06:42 2024
Packet from ('172.21.132.191', 52836) responded: PING 1 THU SEP 12 18:06:42 2024
Packet from ('172.21.132.191', 52836) responded: PING 2 THU SEP 12 18:06:42 2024
Packet from ('172.21.132.191', 52836) responded: PING 3 THU SEP 12 18:06:42 2024
Packet from ('172.21.132.191', 52836) responded: PING 4 THU SEP 12 18:06:42 2024
Sending ICMP Destination Unreachable to ('172.21.132.191', 52836) (rand=8)
Packet from ('172.21.132.191', 52836) responded: PING 6 THU SEP 12 18:06:43 2024
Packet from ('172.21.132.191', 52836) responded: PING 7 THU SEP 12 18:06:43 2024
Sending ICMP Destination Unreachable to ('172.21.132.191', 52836) (rand=7)
Sending ICMP Port Unreachable to ('172.21.132.191', 52836) (rand=9)
Sending ICMP Destination Unreachable to ('172.21.132.191', 52836) (rand=8)
Connection with ('172.21.132.191', 52836) closed.
```

### Client

```
TCP_ICMP_Client.py X
Part2 > TCP_ICMP_Client.py > ...
1 import time
2 from socket import *
3
4 # Create a TCP socket
5 client = socket(AF_INET, SOCK_STREAM)
6
7 # Set a timeout of 1 seconds
8 client.settimeout(1)
9 server_ip = '172.21.132.192'
10 # Server address and port
11 server_address = (server_ip, 14008)
12
13 # Establish a connection to the server
14 client.connect(server_address)
15
16 # Ask the user to set the number of ping operations
17 num = int(input("Set the number of ping operations: "))
18
19 print("Initiating Ping\n")
20

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Sent Ping 7 Thu Sep 12 18:06:43 2024
Received PING 7 THU SEP 12 18:06:43 2024

Sent Ping 8 Thu Sep 12 18:06:43 2024
ICMP Error: Port Unreachable

Sent Ping 9 Thu Sep 12 18:06:44 2024
ICMP Error: Port Unreachable

Sent Ping 10 Thu Sep 12 18:06:45 2024
ICMP Error: Port Unreachable

Ping statistics for 172.21.132.192:
Packets: Sent = 10, Received = 6, Lost = 4 (40.0% loss)
Approximate round trip times in milli-seconds:
Minimum: 0.01 ms, Maximum: 0.08 ms, Average: 0.04 ms

• rushi@rush-VirtualBox:~/Desktop/Computer_Networks/My_Fingers_Files/Program_A1$
```

## Part 3: ICMP Pinger

### 1. ICMP to google without packet loss :

Sending ICMP ping requests to the google.com server, and the following responses are received. As observed below there is one packet loss.

```
ICMP_Pinger.py X
Part3 > ICMP_Pinger.py > receiveOnePing
32 def receiveOnePing(mySocket, ID, timeout, destAddr, sequence):
46     # Fetch the ICMP header from the IP packet
47     icmpHeader = recPacket[20:28]
48     type, code, checksum, packetID, sequence_received = struct.unpack("bbHHh", icmpHeader)
49
50     if packetID == ID: # If the packet matches the ID
51         bytesInDouble = struct.calcsize("d")
52
53         timeSent = struct.unpack("d", recPacket[28:28 + bytesInDouble])[0]
54         rtt = timeReceived - timeSent
55         return rtt * 1000, sequence_received # Return the RTT in milliseconds
56
57     timeLeft = timeLeft - howLongInSelect
58     if timeLeft <= 0:
59         return f"Request timed out. Seq = {sequence}"
60

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
gnment\ACN\Program_A1\Part3\ICMP_Pinger.py"
Pinging 142.250.193.142 using Python:

Reply from 142.250.193.142: Seq=1, time=460.45ms
Reply from 142.250.193.142: Seq=2, time=371.44ms
Reply from 142.250.193.142: Seq=3, time=382.11ms
Request timed out. Seq = 4

Ping statistics for 142.250.193.142:
    Packets: Sent = 4, Received = 3, Lost = 1 (25% loss)
    Approximate round trip times in milli-seconds:
        Minimum = 371.44ms, Maximum = 460.45ms, Average = 404.66ms
PS C:\Users\91848\Downloads\IIT Hyderabad\IITH Assignment\ACN\Program_A1>
```

### 2. ICMP Pinger to another machine without packet loss

```
ICMP_Pinger.py
ICMP_Pinger.py > ...
128 def ping(host,numPing, timeout=1):
129     min_rtt = min(rtt)
130     max_rtt = max(rtt)
150     avg_rtt = sum(rtt) / len(rtt)
151     packet_loss_rate = ((numPing - len(rtt)) / numPing) * 100
152
153     print(f"\nMinimum RTT: {min_rtt:.2f} ms")
154     print(f"Maximum RTT: {max_rtt:.2f} ms")
155     print(f"Average RTT: {avg_rtt:.2f} ms")
156     print(f"Packet loss rate: {packet_loss_rate:.2f}%")
157 else:
158     print("No RTTs recorded.")
159     print("Packet loss rate: 100%")
160 numPing=int(input("Enter number of pings :"))
161 ping("192.168.167.187",numPing)
162
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\91848\Downloads\IIT Hyderabad\mp> python -u "c:\Users\91848\Downloa
Enter number of pings :5
Pinging 192.168.167.187 using Python:

Reply from 192.168.167.187: time=76.32ms
Reply from 192.168.167.187: time=230.20ms
Reply from 192.168.167.187: time=25.61ms
Reply from 192.168.167.187: time=223.30ms
Reply from 192.168.167.187: time=229.51ms

Minimum RTT: 25.61 ms
Maximum RTT: 230.20 ms
Average RTT: 156.99 ms
Packet loss rate: 0.00%
PS C:\Users\91848\Downloads\IIT Hyderabad\mp> 
```

### 3. ICMP with handling error



```

vinaykadari@vinaykadari:~/Downloads/Program_A1 (2)/Program_A1/Part3$
PING 192.168.167.187 (192.168.167.187) 56(84) bytes of data.
From 192.168.41.46 icmp_seq=1 Destination Host Unreachable
From 192.168.41.46 icmp_seq=10 Destination Host Unreachable
From 192.168.41.46 icmp_seq=13 Destination Host Unreachable
From 192.168.41.46 icmp_seq=16 Destination Host Unreachable
From 192.168.41.46 icmp_seq=19 Destination Host Unreachable
From 192.168.41.46 icmp_seq=22 Destination Host Unreachable
From 192.168.41.46 icmp_seq=25 Destination Host Unreachable
From 192.168.41.46 icmp_seq=28 Destination Host Unreachable
From 192.168.41.46 icmp_seq=31 Destination Host Unreachable
From 192.168.41.46 icmp_seq=34 Destination Host Unreachable
From 192.168.41.46 icmp_seq=37 Destination Host Unreachable
From 192.168.41.46 icmp_seq=40 Destination Host Unreachable
From 192.168.41.46 icmp_seq=43 Destination Host Unreachable
From 192.168.41.46 icmp_seq=46 Destination Host Unreachable
From 192.168.41.46 icmp_seq=49 Destination Host Unreachable
From 192.168.41.46 icmp_seq=52 Destination Host Unreachable
From 192.168.41.46 icmp_seq=55 Destination Host Unreachable
From 192.168.41.46 icmp_seq=58 Destination Host Unreachable
From 192.168.41.46 icmp_seq=61 Destination Host Unreachable
From 192.168.41.46 icmp_seq=64 Destination Host Unreachable
From 192.168.41.46 icmp_seq=67 Destination Host Unreachable

```

## Work Distribution Summary

**Name:** Vinaykumar Kadari

**Roll No:** CS24MTECH14008

Task/Section	Contribution	Challenges faced
<b>Research &amp; Info gathering</b>	<p>My contribution is on TCP Pinger, ICMP Pinger and Error Handling</p> <p><u>Reference:</u></p> <p>a) Multithreaded TCP servers and clients, <a href="#">Multiple Clients Model Multithreading</a></p> <p>b) Modified TCP without tc-netem <a href="#">NIC level by using tc (traffic control) netem</a></p>	Understanding the structure of ICMP packet and TCP pinger packet and improve it and customize the code takes traffic of time
<b>Code Development</b>	TCP client & TCP server	It is crucial to manage exception and error in socket programming



	TCP_icmp_server TCP_icmp_client ICMP_Pinger	to avoid crashes and work smoothly
<b>Testing &amp; Debugging</b>	TCP client & TCP server TCP_icmp_server TCP_icmp_client ICMP_Pinger	While testing the server gets unresponsive sometime at that time the client might hang or receive incomplete packets.
<b>Documentation &amp; Report Writing</b>	Formatting the documentations and correctly keeping screenshots	
<b>Final Review &amp; Submission</b>	Ensure the code, report and readme file are correctly formatted	

**Name:** P.Rushi Keswar Reddy

**Roll No:** CS24MTECH11018

<b>Task/Section</b>	<b>Contribution</b>	<b>Challenges faced</b>
<b>Research &amp; Info gathering</b>	My contribution is on ICMP_Pinger.py, TCPPingerThreading.py and UDP_icmp_client and server  Reference: <a href="#">TCP Socket Programming</a> <a href="#">ICMP Pinger</a>	In Pinger, ICMP is different from UDP and TCP sockets, and accessing raw sockets might require administrative privileges.
<b>Code Development</b>	TCPPingerThreading.py UDP_icmp_client UDP_icmp_server ICMP_Pinger	It is crucial to manage exception and error in socket programming to avoid crashes and work smoothly
<b>Testing &amp; Debugging</b>	TCPPingerThreading.py UDP_icmp_client UDP_icmp_server ICMP_Pinger	During testing, if the server becomes unresponsive, the client may hang or receive incomplete packets.
<b>Documentation &amp; Report Writing</b>	Explaining the screenshots and correcting the sentences	
<b>Final Review &amp; Submission</b>	Review and test the code thoroughly and check whether the code correctly	

	works or not	
--	--------------	--

**Name:** KISHOR KUMAR PATRO

**Roll No:** SM24MTECH14001

<b>Task/Section</b>	<b>Contribution &lt;describe the work done, in brief&gt;</b>	<b>Challenges faced (if any).</b>
<b>Research &amp; Info gathering</b>	UDPPingerServer.py	Understanding UDP Lack of reliability mechanisms, UDP doesn't have built-in reliability features like acknowledgments and retransmissions.UDP has more speed compare to tcp
<b>Code Development</b>	UDP client UDP server UDP_icmp_server	As UDP doesn't provide mechanism for re-transmission there is a packet loss over congested network, unlike tcp it lacks of feedback
<b>Testing &amp; Debugging</b>	UDP client UDP server UDP_icmp_server	While testing the udp code there is a request timeout error as iith wifi blocks udp icmp packets.Distinguishing host vs network issues: It's difficult to tell if packet loss is due to end-host failure or genuine network problems.
<b>Documentation &amp; Report Writing</b>		
<b>Final Review &amp; Submission</b>		

## 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.

**Names & Roll No:**

**Vinaykumar Kadari** (CS24MTECH14008)

**P.Rushi Keswar Reddy** (CS24MTECH11018)

**Kishor Kumar Patro** (SM24MTECH14001)

**Date:** 12-09-2024

**Signatures:** Vinaykumar, Rushikeswar, Kishor