

ACN Programming Assignment 1 Report

Part 1: UDP PINGERS

- Server & Client on the same machine (localhost)

- Server & Client on different machines

CLIENT (ISHA):

The screenshot shows a Visual Studio Code (VS Code) interface. On the left is a dark-themed sidebar with various icons for file operations like copy, search, and refresh. The main area has two tabs open: 'UDP_client.py' and 'README.md 1'. The code editor displays a Python script named 'UDP_client.py' with the following content:

```
1 # importing essential libraries
2 from socket import *
3 import random
4 import time
5 from datetime import datetime
6
7 # store Server's IP address & port number
8 server_addr = ('172.21.133.124',12001)
9 # Create a UDP socket
10 client_socket = socket(AF_INET, SOCK_DGRAM)
11 client_socket.settimeout(1)
12
13 n = int(input("Enter number of pings (N): "))
14 sequence_number = 1
```

Below the code editor are several tabs: PROBLEMS (1), OUTPUT, DEBUG CONSOLE, and TERMINAL. The TERMINAL tab is active, showing the following terminal output:

```
● (base) ishakumar@Ishas-MacBook-Pro ACN_Assignment_1_My_Pingers % /usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/UDP_Pingers/UDP_client.py
Enter number of pings (N): 6
Packet 1 is lost
Response received is: PING 2 2024-09-10 13:15:30.056468
RTT (in milliseconds) of packet 2 is 203.226089
Packet 3 is lost
Response received is: PING 4 2024-09-10 13:15:31.260358
RTT (in milliseconds) of packet 4 is 6.886244
Response received is: PING 5 2024-09-10 13:15:31.267271
RTT (in milliseconds) of packet 5 is 10.108232
Response received is: PING 6 2024-09-10 13:15:31.277402
RTT (in milliseconds) of packet 6 is 5.508900
Maximum RTT (in milliseconds) = 203.226089
Minimum RTT (in milliseconds) = 5.508900
Average RTT (in milliseconds) = 56.432366
Packet loss rate: 33.333%
○ (base) ishakumar@Ishas-MacBook-Pro ACN_Assignment_1_My_Pingers %
```

At the bottom of the terminal window, there are status indicators: 1△0, UTF-8, LF, { MagicPython 3.9.6 64-bit, Go Live, Stylelint+, Go Live, Prettier, and a few other icons.

SERVER (RAKESH):

The screenshot shows a terminal window with the following content:

```
home > rakesh > Documents > group > UdpServer.py > start_sever
  1  from socket import *
  2  import random
  3
  4
  5  def start_sever():
  6
  7      server_addr = ('172.21.133.124',12001) #isha
  8      server_socket = socket(AF_INET, SOCK_DGRAM)
  9      server_socket.bind(server_addr)
 10
 11      while True:
 12          msg, client_addr = server_socket.recvfrom(1024)
 13          msg = msg.decode()
 14          print(f'Message sent = {msg} from {client_addr}')
 15          response = msg.upper()
 16          rand = random.randint(1,10)
 17          if rand>8:
 18              continue
 19          print(f'Response sent back = {msg}')
 20          server_socket.sendto(response.encode(), client_addr)
 21
 22 start_sever()

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(base) rakesh@rakesh:~/Documents/group$ python3 UdpServer.py
Message sent = ping 1 2024-09-10 13:15:29.055551 from ('172.21.134.130', 63481)
Message sent = ping 2 2024-09-10 13:15:30.056468 from ('172.21.134.130', 63481)
Response sent back = ping 2 2024-09-10 13:15:30.056468
Message sent = ping 3 2024-09-10 13:15:30.259767 from ('172.21.134.130', 63481)
Message sent = ping 4 2024-09-10 13:15:31.260358 from ('172.21.134.130', 63481)
Response sent back = ping 4 2024-09-10 13:15:31.260358
Message sent = ping 5 2024-09-10 13:15:31.267271 from ('172.21.134.130', 63481)
Response sent back = ping 5 2024-09-10 13:15:31.267271
Message sent = ping 6 2024-09-10 13:15:31.277402 from ('172.21.134.130', 63481)
Response sent back = ping 6 2024-09-10 13:15:31.277402
```

- Multiple clients try to access a single server simultaneously
SERVER: (ISHA)

The screenshot shows a Python development environment with a dark theme. On the left is a sidebar with various icons: file, search, connection, run, file browser, terminal, Python logo, and settings. A notification badge '1' is visible on the terminal icon.

The main area has tabs for 'UDP_server.py' and 'start_sever'. The code for 'start_sever' is shown:

```
1 from socket import *
2 import random
3
4 def start_sever():
5     server_addr = ('172.21.134.130', 12001)
6     server_socket = socket(AF_INET, SOCK_DGRAM)
```

The terminal window below shows the execution of the script and its output:

```
Message sent = ping 1 2024-09-10 13:02:06.949274 from ('172.21.134.130', 51442)
Response sent back = ping 1 2024-09-10 13:02:06.949274
Message sent = ping 2 2024-09-10 13:02:06.949758 from ('172.21.134.130', 51442)
Response sent back = ping 2 2024-09-10 13:02:06.949758
Message sent = ping 3 2024-09-10 13:02:06.950062 from ('172.21.134.130', 51442)
Response sent back = ping 3 2024-09-10 13:02:06.950062
Message sent = ping 4 2024-09-10 13:02:06.950291 from ('172.21.134.130', 51442)
Message sent = ping 5 2024-09-10 13:02:07.950701 from ('172.21.134.130', 51442)
Response sent back = ping 5 2024-09-10 13:02:07.950701
Message sent = ping 1 2024-09-10 13:02:12.026915 from ('172.21.133.124', 43148)
Message sent = ping 2 2024-09-10 13:02:13.028450 from ('172.21.133.124', 43148)
Response sent back = ping 2 2024-09-10 13:02:13.028450
Message sent = ping 3 2024-09-10 13:02:13.128759 from ('172.21.133.124', 43148)
Response sent back = ping 3 2024-09-10 13:02:13.128759
Message sent = ping 4 2024-09-10 13:02:13.135821 from ('172.21.133.124', 43148)
Response sent back = ping 4 2024-09-10 13:02:13.135821
Message sent = ping 5 2024-09-10 13:02:13.141011 from ('172.21.133.124', 43148)
```

The status bar at the bottom shows: 0 notifications, 0 errors, MagicPython 3.9.6 64-bit, Go Live, Go Live, and a bell icon.

CLIENT 1:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** On the left, showing files `UDP_client.py` and `README.md`.
- Code Editor:** The main area displays the Python code for a UDP client. The code imports necessary libraries, defines a server address, creates a UDP socket, and prompts the user for the number of pings.
- Terminal:** Below the code editor, the terminal window shows the execution of the script and its output. It receives 5 pings and calculates RTT statistics.
- Status Bar:** At the bottom, it shows 1 warning, the code version (3.9.6 64-bit), and various live preview and linting tools like Go Live, Stylelint+, Prettier, and a GitHub icon.

```
1 # importing essential libraries
2 from socket import *
3 import random
4 import time
5 from datetime import datetime
6
7 # store Server's IP address & port number
8 server_addr = ('172.21.134.130', 12001)
9 # Create a UDP socket
10 client_socket = socket(AF_INET, SOCK_DGRAM)
11 client_socket.settimeout(1)
12
13 n = int(input("Enter number of pings (N): "))
```

PROBLEMS 1 TERMINAL ...

/usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/UDP_Pingers/UDP_client.py

● (base) ishakumar@Ishas-MacBook-Pro ACN_Assignment_1_My_Pingers % /usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/UDP_Pingers/UDP_client.py

Enter number of pings (N): 5

Response received is: PING 1 2024-09-10 13:02:06.949274

RTT (in milliseconds) of packet 1 is 0.451803

Response received is: PING 2 2024-09-10 13:02:06.949758

RTT (in milliseconds) of packet 2 is 0.290871

Response received is: PING 3 2024-09-10 13:02:06.950062

RTT (in milliseconds) of packet 3 is 0.215054

Packet 4 is lost

Response received is: PING 5 2024-09-10 13:02:07.950701

RTT (in milliseconds) of packet 5 is 0.370026

Maximum RTT (in milliseconds) = 0.451803

Minimum RTT (in milliseconds) = 0.215054

Average RTT (in milliseconds) = 0.331938

Packet loss rate: 20.000%

○ (base) ishakumar@Ishas-MacBook-Pro ACN_Assignment_1_My_Pingers %

⊗ 1 ⚠ 0 3.9.6 64-bit ⚡ Go Live Stylelint+ ⚡ Go Live ⚡ Prettier ⚡ ⚡

CLIENT 2:

```

❶ UdpClient.py ✘ ❷ UdpServer.py ❸ TcpServer.py ❹ tcpmodifiedserver.py ❺ TcpClient.py
home > rakesh > Documents > group > ❻ UdpClient.py > ...
10   client_socket.settimeout(1)
11   n = int(input("Enter number of pings (N): "))
12   sequence_number = 1
13   total_rtt=0
14   max_rtt=float('-inf')
15   min_rtt=float('inf')
16   success_pings=0
17   copy=
18   while n>0:
19     try:
20       time_stamp = time.time()
21       readable_time = datetime.fromtimestamp(time_stamp).strftime('%Y-%m-%d %H:%M:%S.%f')
22       msg = f'ping {sequence_number} {readable_time}'
23       client_socket.sendto(msg.encode(), server_addr)
24       reponse, _ = client_socket.recvfrom(1024)
25       print(f'Response received is: {reponse.decode()}')
26       rtt = time.time() - time_stamp
27       rtt=rtt*1000
28       print(f'RTT (in milliseconds) of packet {sequence_number} is {rtt:.3f}')
29       total_rtt += rtt
30       max_rtt = max(max_rtt,rtt)
31       min_rtt = min(min_rtt,rtt)
32       success_pings += 1
33     except timeout:
34       print(f'Packet {sequence_number} is lost')
35     finally:
36       sequence_number += 1
37       n -= 1
38   print(f'Maximum RTT = {max_rtt:.3f}')
39   print(f'Minimum RTT = {min_rtt:.3f}')
40   if success_pings>0:
41     print(f'Average RTT = {total_rtt/success_pings:.3f}')
42   else:
43
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
● (base) rakesh@rakesh:~/Documents/group$ python3 UdpClient.py
Enter number of pings (N): 5
Packet 1 is lost
Response received is: PING 2 2024-09-10 13:02:13.028450
RTT (in milliseconds) of packet 2 is 100.265
Response received is: PING 3 2024-09-10 13:02:13.128759
RTT (in milliseconds) of packet 3 is 7.020
Response received is: PING 4 2024-09-10 13:02:13.135821
RTT (in milliseconds) of packet 4 is 5.147
Packet 5 is lost
Maximum RTT = 100.265
Minimum RTT = 5.147
Average RTT = 37.477
Packet loss rate: 40.000%
❻ (base) rakesh@rakesh:~/Documents/group$ █

```

- Modified UDP Server Code (UDPPingerModifiedServer.py): emulate packet loss at the network interface card (NIC) level by using tc (traffic control) netem utility in Linux. (<https://man7.org/linux/man-pages/man8/tc-netem.8.html>)

CLIENT:

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows two files: `UDP_client.py` and `README.md`.
- Terminal:** Displays the execution of `UDP_client.py` and its output. The output shows a ping sequence from the client to a server at `172.21.133.124`, port `12001`. The client sends 10 pings. The server responds with PING messages and RTT measurements. The client also calculates statistics: Maximum RTT = 116.750717 ms, Minimum RTT = 7.130146 ms, Average RTT = 23.060381 ms, and Packet loss rate = 20.000%.
- Status Bar:** Shows the file `MagicPython` is active, along with other extensions like `Stylelint+` and `Prettier`.

```
1 # importing essential libraries
2 from socket import *
3 import random
4 import time
5 from datetime import datetime
6
7 # store Server's IP address & port number
8 server_addr = ('172.21.133.124',12001)

(base) ishakumar@Ishas-MacBook-Pro ACN_Assignment_1_My_Pingers % /usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/UDP_Pingers/UDP_client.py
Enter number of pings (N): 10
Response received is: PING 1 2024-09-10 13:07:35.512666
RTT (in milliseconds) of packet 1 is 16.738653
Response received is: PING 2 2024-09-10 13:07:35.529452
RTT (in milliseconds) of packet 2 is 7.632732
Response received is: PING 3 2024-09-10 13:07:35.537119
RTT (in milliseconds) of packet 3 is 7.130146
Packet 4 is lost
Response received is: PING 5 2024-09-10 13:07:36.545453
RTT (in milliseconds) of packet 5 is 116.750717
Response received is: PING 6 2024-09-10 13:07:36.662245
RTT (in milliseconds) of packet 6 is 7.130861
Response received is: PING 7 2024-09-10 13:07:36.669398
RTT (in milliseconds) of packet 7 is 8.692980
Response received is: PING 8 2024-09-10 13:07:36.678131
RTT (in milliseconds) of packet 8 is 8.223057
Packet 9 is lost
Response received is: PING 10 2024-09-10 13:07:37.687685
RTT (in milliseconds) of packet 10 is 12.183905
Maximum RTT (in milliseconds) = 116.750717
Minimum RTT (in milliseconds) = 7.130146
Average RTT (in milliseconds) = 23.060381
Packet loss rate: 20.000%
(base) ishakumar@Ishas-MacBook-Pro ACN_Assignment_1_My_Pingers %
```

SERVER:

```
(base) rakesh@rakesh:~/Documents/group$ sudo tc qdisc add dev wlp0s20f3 root netem loss 20%
(base) rakesh@rakesh:~/Documents/group$
```

```

  UdpClient.py  UdpServer.py X  TcpServer.py  tcpmodifiedserver.py  TcpClient.py
home > rakesh > Documents > group > UdpServer.py > ...
1   from socket import *
2   import random
3
4
5   def start_sever():
6
7       server_addr = ('172.21.133.124',12001) #isha
8       server_socket = socket(AF_INET, SOCK_DGRAM)
9       server_socket.bind(server_addr)
10
11      while True:
12          msg, client_addr = server_socket.recvfrom(1024)
13          msg = msg.decode()
14          print(f'Message sent = {msg} from {client_addr}')
15          response = msg.upper()
16          print(f'Response sent back = {msg}')
17          server_socket.sendto(response.encode(), client_addr)
18      start_sever()
19

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS
○ (base) rakesh@rakesh:~/Documents/group$ python3 UdpServer.py
Message sent = ping 1 2024-09-10 13:07:35.512666 from ('172.21.134.130', 60579)
Response sent back = ping 1 2024-09-10 13:07:35.512666
Message sent = ping 2 2024-09-10 13:07:35.529452 from ('172.21.134.130', 60579)
Response sent back = ping 2 2024-09-10 13:07:35.529452
Message sent = ping 3 2024-09-10 13:07:35.537119 from ('172.21.134.130', 60579)
Response sent back = ping 3 2024-09-10 13:07:35.537119
Message sent = ping 4 2024-09-10 13:07:35.544272 from ('172.21.134.130', 60579)
Response sent back = ping 4 2024-09-10 13:07:35.544272
Message sent = ping 5 2024-09-10 13:07:36.545453 from ('172.21.134.130', 60579)
Response sent back = ping 5 2024-09-10 13:07:36.545453
Message sent = ping 6 2024-09-10 13:07:36.662245 from ('172.21.134.130', 60579)
Response sent back = ping 6 2024-09-10 13:07:36.662245
Message sent = ping 7 2024-09-10 13:07:36.669398 from ('172.21.134.130', 60579)
Response sent back = ping 7 2024-09-10 13:07:36.669398
Message sent = ping 8 2024-09-10 13:07:36.678131 from ('172.21.134.130', 60579)
Response sent back = ping 8 2024-09-10 13:07:36.678131
Message sent = ping 9 2024-09-10 13:07:36.686382 from ('172.21.134.130', 60579)
Response sent back = ping 9 2024-09-10 13:07:36.686382
Message sent = ping 10 2024-09-10 13:07:37.687685 from ('172.21.134.130', 60579)
Response sent back = ping 10 2024-09-10 13:07:37.687685

```

Part 2: TCP PINGERS

- Server & Client on the same machine (localhost)

The screenshot shows two terminal panes in VS Code. The left pane displays the execution of `TCPpingClient.py`, which prompts for the number of pings (N) and then performs six pings to the local host. The right pane displays the execution of `TCPpingServer.py`, which listens on port 12001 and responds to the client's pings.

```

TCPpingClient.py
from socket import *
import time
import time
from datetime import datetime
server_addr = ('127.0.0.1', 12001)
n = int(input("Enter number of pings (N): "))
TERMINAL ... Python - ishakumar + ~ □ ^ ×
/usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/TCP_Pingers/TCPpingClient.py
(base) ishakumar@ihas-MacBook-Pro ~ % /usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/TCP_Pingers/TCPpingClient.py
Enter number of pings (N): 6
Client Connection Successful..
Response received is: PING 1 2024-09-10 13:20:31.648889
RTT (in milliseconds) of packet 1 is 0.272989
Client Connection Successful..
Packet 2 lost
Client Connection Successful..
Packet 3 lost
Client Connection Successful..
Packet 4 lost
Client Connection Successful..
Response received is: PING 5 2024-09-10 13:20:31.650771
RTT (in milliseconds) of packet 5 is 0.240088
Client Connection Successful..
Response received is: PING 6 2024-09-10 13:20:31.651277
RTT (in milliseconds) of packet 6 is 0.288963
Maximum RTT (in milliseconds) = 0.288963
Minimum RTT (in milliseconds) = 0.240088
Average RTT (in milliseconds) = 0.267347
Packet loss rate: 50.000000%
(base) ishakumar@ihas-MacBook-Pro ~ %

TCPpingServer.py
from socket import *
import random
server_addr = ('127.0.0.1', 12001)
server_socket = socket(AF_INET, SOCK_STREAM)
server_socket.bind(server_addr)
print(f"Server Started Listening on {server_addr}")
server_socket.listen(0)
while True:
    client_socket, client_addr = server_socket.accept()
    while True:
        msg = client_socket.recv(1024)
        if not msg.decode():
            break
TERMINAL ... Python + ~ □ ^ ×
/usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/TCP_Pingers/TCPpingServer.py
(base) ishakumar@ihas-MacBook-Pro ~ % /usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/TCP_Pingers/TCPpingServer.py
Server Started Listening on 127.0.0.1:12001
Message sent by client = b'ping 1 2024-09-10 13:20:31.648889'
Response sent back = b'PING 1 2024-09-10 13:20:31.648889'
Packet lost
Packet lost
Message sent by client = b'ping 5 2024-09-10 13:20:31.650771'
Response sent back = b'PING 5 2024-09-10 13:20:31.650771'
Message sent by client = b'ping 6 2024-09-10 13:20:31.651277'
Response sent back = b'PING 6 2024-09-10 13:20:31.651277'

```

- Server & Client on different machines

CLIENT (ISHA):

The screenshot shows the Visual Studio Code (VS Code) interface. On the left is the sidebar with various icons for file operations like copy, search, and refresh. The main area displays a Python script named `TCPpingerClient.py`. The code imports `socket`, `time`, and `datetime`, defines a server address, and prompts the user for the number of pings. It then establishes a connection, sends pings, and calculates RTT statistics. The terminal tab shows the execution of the script and its output.

```
from socket import *
import time
import time
from datetime import datetime
server_addr = ('172.21.133.124', 12001)
n = int(input("Enter number of pings (N): "))
sequence_number = 1
total_rtt=0

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python - ishakumar + v ⌂ ⌄ ⌈ ×

/usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/TCP_Pingers/TCPpingerClient.py
(base) ishakumar@Ishas-MacBook-Pro ~ % /usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/TCP_Pingers/TCPpingerClient.py
Enter number of pings (N): 7
Client Connection Successful..
Response received is: PING 1 2024-09-10 13:22:17.721851
RTT (in milliseconds) of packet 1 is 7.383108
Client Connection Successful..
Packet 2 lost
Packet 3 is lost
Client Connection Successful..
Packet 4 lost
Client Connection Successful..
Response received is: PING 5 2024-09-10 13:22:19.045702
RTT (in milliseconds) of packet 5 is 8.157015
Packet 6 is lost
Client Connection Successful..
Response received is: PING 7 2024-09-10 13:22:20.076387
RTT (in milliseconds) of packet 7 is 306.046963
Maximum RTT (in milliseconds) = 306.046963
Minimum RTT (in milliseconds) = 7.383108
Average RTT (in milliseconds) = 107.195695
Packet loss rate: 57.142857%
(base) ishakumar@Ishas-MacBook-Pro ~ %
```

SERVER (RAKESH):

The screenshot shows a code editor interface with several tabs at the top: UdpClient.py, UdpServer.py, TcpServer.py (highlighted in blue), tcpmodifiedserver.py, and TcpClient.py. Below the tabs is a code editor window containing the following Python script:

```
home > rakesh > Documents > group > TcpServer.py > ...
9     server_socket.listen(0)
10
11
12 while True:
13     client_socket, client_addr = server_socket.accept()
14     while True:
15         msg = client_socket.recv(1024)
16         if not msg.decode():
17             break
18         rand = random.randint(1,10)
19         if rand>8:
20             print('Packet lost')
21             client_socket.send('LOST'.encode())
22             break
23         print(f'Message sent by client = {msg} from {client_addr}')
24         response = msg.upper()
25         print(f'Response sent back = {response}')
26         client_socket.send(response)
27     client_socket.close()
28
```

Below the code editor is a terminal window showing the execution of the script and its interaction with clients:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
KeyboardInterrupt
(base) rakesh@rakesh:~/Documents/group$ python3 TcpServer.py
Server Started Listening on 172.21.133.124:12001
Message sent by client = b'ping 1 2024-09-10 13:22:17.721851' from ('172.21.134.130', 49272)
Response sent back = b'PING 1 2024-09-10 13:22:17.721851'
Packet lost
Packet lost
Message sent by client = b'ping 5 2024-09-10 13:22:19.045702' from ('172.21.134.130', 49276)
Response sent back = b'PING 5 2024-09-10 13:22:19.045702'
Message sent by client = b'ping 7 2024-09-10 13:22:20.076387' from ('172.21.134.130', 49278)
Response sent back = b'PING 7 2024-09-10 13:22:20.076387'
```

- Multiple clients try to access a single server simultaneously (MODIFIED TCP)

SERVER: (RAKESH)

```
home > rakesh > Documents > threads > tcpmodifiedserver.py > ...
1  from socket import *
2  import threading
3  # Server address and port
4  server_address = ['172.21.133.124', 12001] # Empty string in the address field means bind
5
6  # Function to handle a client connection
7  def multiple_client_handle(client_socket, client_address):
8      print(f'Connected to {client_address}')
9
10     while True:
11         data = client_socket.recv(1024).decode("utf-8")
12
13         if not data:
14             break
15
16         response = data.upper()
17
18         # Send the response data back to the client
19         client_socket.send(response.encode("utf-8"))
20
21     # Close the client socket
22     client_socket.close()
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Connection with ('172.21.134.130', 49302) closed
Connected to ('172.21.133.124', 41402)
Connection with ('172.21.133.124', 41402) closed
Connected to ('172.21.133.124', 41406)
Connection with ('172.21.133.124', 41406) closed
Connected to ('172.21.134.130', 49309)
Connection with ('172.21.134.130', 49309) closed
Connected to ('172.21.134.130', 49310)
Connection with ('172.21.134.130', 49310) closed
Connected to ('172.21.134.130', 49311)
Connection with ('172.21.134.130', 49311) closed
Connected to ('172.21.134.130', 49312)
Connection with ('172.21.134.130', 49312) closed
Connected to ('172.21.134.130', 49313)
Connection with ('172.21.134.130', 49313) closed
Connected to ('172.21.134.130', 49314)
Connection with ('172.21.134.130', 49314) closed
Connected to ('172.21.133.124', 41426)
Connection with ('172.21.133.124', 41426) closed
Connected to ('172.21.133.124', 41432)
Connected to ('172.21.133.124', 41446)
Connection with ('172.21.133.124', 41432) closed
Connection with ('172.21.133.124', 41446) closed
Connected to ('172.21.133.124', 41472)
Connection with ('172.21.133.124', 41472) closed
```

CLIENT 1: (ISHA)

```
 1  from socket import *
 2  import time
 3  import time
 4  from datetime import datetime
 5
 6  server_addr = ('172.21.133.124', 12001)
 7
 8  n = int(input("Enter number of pings (N): "))
 9  sequence_number = 1

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
[Python - ishakumar] + v  ^ x

● (base) ishakumar@Ishas-MacBook-Pro ~ % /usr/bin/python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pingers/TCP_Pingers/TCPpingClient.py
Enter number of pings (N): 6
Client Connection Successful..
Response received is: PING 1 2024-09-10 13:30:15.520813
RTT (in milliseconds) of packet 1 is 6.935120
Client Connection Successful..
Response received is: PING 2 2024-09-10 13:30:15.533392
RTT (in milliseconds) of packet 2 is 5.114079
Client Connection Successful..
Response received is: PING 3 2024-09-10 13:30:15.542689
RTT (in milliseconds) of packet 3 is 216.471910
Client Connection Successful..
Response received is: PING 4 2024-09-10 13:30:15.763743
RTT (in milliseconds) of packet 4 is 219.401121
Client Connection Successful..
Response received is: PING 5 2024-09-10 13:30:15.987738
RTT (in milliseconds) of packet 5 is 4.595041
Client Connection Successful..
Response received is: PING 6 2024-09-10 13:30:15.997828
RTT (in milliseconds) of packet 6 is 217.492104
Maximum RTT (in milliseconds) = 219.401121
Minimum RTT (in milliseconds) = 4.595041
Average RTT (in milliseconds) = 111.668229
Packet loss rate: 0.00000%
(base) ishakumar@Ishas-MacBook-Pro ~ %
```

CLIENT 2: (RAKESH)

```
● (base) rakesh@rakesh:~/Documents/group$ python3 TcpClient.py
Enter number of pings (N): 10
Response received is: PING 1 2024-09-10 13:30:15.015621
RTT (in milliseconds) of packet 1 is 0.696
Response received is: PING 2 2024-09-10 13:30:15.016489
RTT (in milliseconds) of packet 2 is 207.131
Packet 3 is lost
Response received is: PING 4 2024-09-10 13:30:16.224681
RTT (in milliseconds) of packet 4 is 439.109
Response received is: PING 5 2024-09-10 13:30:16.664037
RTT (in milliseconds) of packet 5 is 0.765
Response received is: PING 6 2024-09-10 13:30:16.664951
RTT (in milliseconds) of packet 6 is 878.797
Packet 7 is lost
Packet 8 is lost
Packet 9 is lost
Response received is: PING 10 2024-09-10 13:30:20.549603
RTT (in milliseconds) of packet 10 is 426.738
Maximum RTT (in milliseconds) = 878.797
Minimum RTT (in milliseconds) = 0.696
Average RTT (in milliseconds) = 325.539
Packet loss rate: 40.00%
○ (base) rakesh@rakesh:~/Documents/group$
```

- Modified TCP Server Code (TCPPingerModifiedServer.py): emulate packet loss at the network interface card (NIC) level by using tc (traffic control) netem utility in Linux. (<https://man7.org/linux/man-pages/man8/tc-netem.8.html>)

CLIENT: (ISHA)

```

TCP_Pingers > TCPpingerClient.py > ...
1  from socket import *
2  import time
3  import time
4  from datetime import datetime
5
6  server_addr = ('172.21.133.124', 12001)
7
8  n = int(input("Enter number of pings (N): "))
9  sequence_number = 1

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL

Enter number of pings (N): 7
Packet 1 is lost
Client Connection Successful..
Response received is: PING 2 2024-09-10 14:23:02.198075
RTT (in milliseconds) of packet 2 is 5.348206
Client Connection Successful..
Response received is: PING 3 2024-09-10 14:23:02.208631
RTT (in milliseconds) of packet 3 is 221.537113
Client Connection Successful..
Response received is: PING 4 2024-09-10 14:23:02.435421
RTT (in milliseconds) of packet 4 is 6.169081
Client Connection Successful..
Response received is: PING 5 2024-09-10 14:23:02.446076
RTT (in milliseconds) of packet 5 is 5.523205
Client Connection Successful..
Response received is: PING 6 2024-09-10 14:23:02.456530
RTT (in milliseconds) of packet 6 is 5.006075
Client Connection Successful..
Response received is: PING 7 2024-09-10 14:23:02.468267
RTT (in milliseconds) of packet 7 is 9.060860
Maximum RTT (in milliseconds) = 221.537113
Minimum RTT (in milliseconds) = 5.006075
Average RTT (in milliseconds) = 42.107423
Packet loss rate: 14.285714%

```

SERVER: (RAKESH)

```
(base) rakesh@rakesh:~/Documents/group$ sudo tc qdisc add dev wlp0s20f3 root netem loss 20%
(base) rakesh@rakesh:~/Documents/group$
```

```

UdpClient.py UdpServer.py TcpServer.py tcpmodifiedserver.py TcpClient.py

home > rakesh > Documents > threads > tcpmodifiedserver.py > ...
1  from socket import *
2  import threading
3  #import random
4  # Server address and port
5  server_address = ('172.21.133.124', 12001) # Empty string in the address field means bind to all available interfaces
6
7  # Function to handle a client connection (parameter) client_address: Any
8  def multiple_client_handle(client_socket, client_address):
9      print(f'Connected to {client_address}')
10
11     while True:
12         data = client_socket.recv(1024).decode("utf-8")
13
14         if not data:
15             break
16
17         response = data.upper()
18
19         # Send the response data back to the client
20         client_socket.send(response.encode("utf-8"))
21
22     # Close the client socket

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

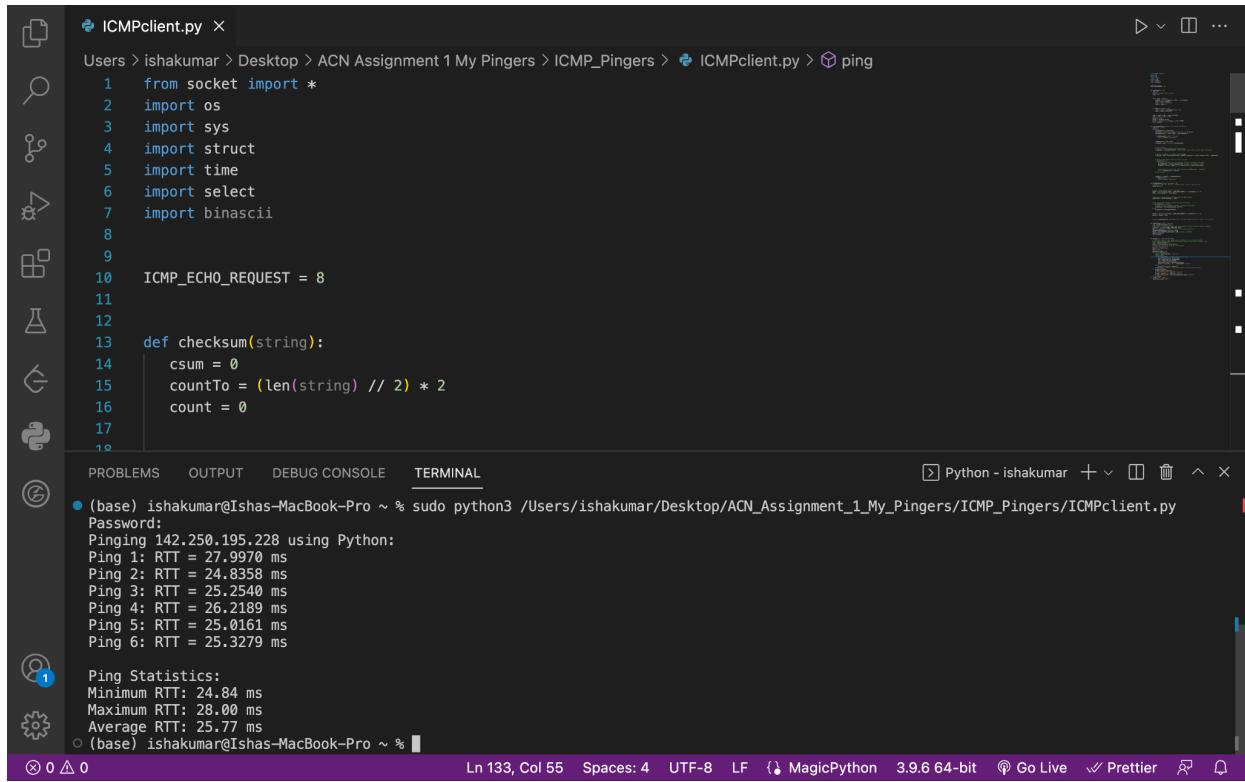
Connection with ('172.21.133.124', 53254) closed
Connected to ('172.21.133.124', 53262)
Connection with ('172.21.133.124', 53262) closed
Connected to ('172.21.133.124', 53272)
Connection with ('172.21.133.124', 53272) closed
Connected to ('172.21.133.124', 53288)
Connection with ('172.21.133.124', 53288) closed
Connected to ('172.21.133.124', 53296)
Connection with ('172.21.133.124', 53296) closed
Connected to ('172.21.133.124', 53298)
Connection with ('172.21.133.124', 53298) closed
Connected to ('172.21.133.124', 53304)
Connection with ('172.21.133.124', 53304) closed
Connected to ('172.21.134.130', 49561)
Connection with ('172.21.134.130', 49561) closed
Connected to ('172.21.134.130', 49562)
Connection with ('172.21.134.130', 49562) closed
Connected to ('172.21.134.130', 49563)
Connection with ('172.21.134.130', 49563) closed
Connected to ('172.21.134.130', 49564)
Connection with ('172.21.134.130', 49564) closed
Connected to ('172.21.134.130', 49565)
Connection with ('172.21.134.130', 49565) closed
Connected to ('172.21.134.130', 49566)
Connection with ('172.21.134.130', 49566) closed

Enter number of pings (N): 10
Response received is: PING 1 2024-09-10 14:23:00.255208
RTT (in milliseconds) of packet 1 is 0.666
Response received is: PING 2 2024-09-10 14:23:00.256053
RTT (in milliseconds) of packet 2 is 0.362
Response received is: PING 3 2024-09-10 14:23:00.256585
RTT (in milliseconds) of packet 3 is 0.344
Response received is: PING 4 2024-09-10 14:23:00.257651
RTT (in milliseconds) of packet 4 is 0.362
Response received is: PING 5 2024-09-10 14:23:00.257475
RTT (in milliseconds) of packet 5 is 0.291
Response received is: PING 6 2024-09-10 14:23:00.257886
RTT (in milliseconds) of packet 6 is 0.265
Response received is: PING 7 2024-09-10 14:23:00.258265
RTT (in milliseconds) of packet 7 is 0.253
Response received is: PING 8 2024-09-10 14:23:00.258646
RTT (in milliseconds) of packet 8 is 0.251
Response received is: PING 9 2024-09-10 14:23:00.259010
RTT (in milliseconds) of packet 9 is 0.240
Response received is: PING 10 2024-09-10 14:23:00.259365
RTT (in milliseconds) of packet 10 is 0.430
Maximum RTT (in milliseconds) = 0.666
Minimum RTT (in milliseconds) = 0.240
Average RTT (in milliseconds) = 0.340
Packet loss rate: 0.00%

```

Part 3: ICMP PINGERS

CLIENT:



```
ICMPclient.py ×
Users > ishakumar > Desktop > ACN Assignment 1 My Pingers > ICMP_Pingers > ICMPclient.py > ping
 1  from socket import *
 2  import os
 3  import sys
 4  import struct
 5  import time
 6  import select
 7  import binascii
 8
 9
10 ICMP_ECHO_REQUEST = 8
11
12
13 def checksum(string):
14     csum = 0
15     countTo = (len(string) // 2) * 2
16     count = 0
17
18
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL
(base) ishakumar@Ishas-MacBook-Pro ~ % sudo python3 /Users/ishakumar/Desktop/ACN_Assignment_1_My_Pinglers/ICMP_Pingers/ICMPclient.py
Password:
Pinging 142.250.195.228 using Python:
Ping 1: RTT = 27.9970 ms
Ping 2: RTT = 24.8358 ms
Ping 3: RTT = 25.2540 ms
Ping 4: RTT = 26.2189 ms
Ping 5: RTT = 25.0161 ms
Ping 6: RTT = 25.3279 ms
Ping Statistics:
Minimum RTT: 24.84 ms
Maximum RTT: 28.00 ms
Average RTT: 25.77 ms
(base) ishakumar@Ishas-MacBook-Pro ~ %
Ln 133, Col 55    Spaces: 4    UTF-8    LF    { MagicPython 3.9.6 64-bit    ⚡ Go Live    ✨ Prettier    ⌂    ⌂
```

ERROR RESPONSE:

1. ICMP-host-unreachable
2. ICMP-port-unreachable

UDP:

SERVER:

```

1 import socket
2 import random
3 import struct
4
5 # Function to create a simple ICMP error packet (Destination Host Unreachable)
6 def create_icmp_error_packet(original_message):
7     # Destination Unreachable
8     icmp_type = 3
9     # Host Unreachable
10    icmp_code = 1
11    checksum = 0
12
13    # Create ICMP header (8 bytes: type, code, checksum, ID, sequence)
14    header = struct.pack('bbHHH', icmp_type, icmp_code, checksum, 0, 0)
15    checksum = calculate_checksum(header)
16    header = struct.pack('bbHHH', icmp_type, icmp_code, checksum, 0, 0)
17    icmp_socket.sendto(header, original_message)
18
19
20    # Function to calculate checksum
21    def calculate_checksum(data):
22        checksum = 0
23        length = len(data)
24        for i in range(0, length, 2):
25            word = (data[i] << 8) + (data[i + 1])
26            checksum += word

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Sending valid response to ('192.168.241.144', 49515)
Received message from ('192.168.241.144', 49515): PING 3 2024-09-12 15:43:14.876932
Sending valid response to ('192.168.241.144', 49515)
Received message from ('192.168.241.144', 49515): PING 4 2024-09-12 15:43:14.880671
Sending valid response to ('192.168.241.144', 49515)
Received message from ('192.168.241.144', 49515): PING 5 2024-09-12 15:43:14.883816
Sending valid response to ('192.168.241.144', 49515)
Received message from ('192.168.241.144', 49515): PING 6 2024-09-12 15:43:14.888264
Sending valid response to ('192.168.241.144', 49515)
Received message from ('192.168.241.144', 49515): PING 7 2024-09-12 15:43:14.892838
Sending valid response to ('192.168.241.144', 49515)
Received message from ('192.168.241.144', 49515): PING 8 2024-09-12 15:43:14.898123
Sending ICMP error to ('192.168.241.144', 49515)
Received message from ('192.168.241.144', 49515): PING 9 2024-09-12 15:43:15.899773
Sending valid response to ('192.168.241.144', 49515)
Received message from ('192.168.241.144', 49515): PING 10 2024-09-12 15:43:15.904616
Sending ICMP error to ('192.168.241.144', 49515)

```

CLIENT:

```

1  from socket import *
2  import time
3  from datetime import datetime
4  import struct
5
6  # Define the server address and port
7  server_address = ('192.168.241.110', 12000)
8
9  # Create a UDP socket for sending pings
10 client_socket = socket(AF_INET, SOCK_DGRAM)
11 client_socket.settimeout(1)
12
13 # Create a raw socket to capture ICMP errors
14 icmp_socket = socket(AF_INET, SOCK_RAW, IPPROTO_ICMP)
15 icmp_socket.settimeout(1)
16 # Ask for the number of pings to send
17 N = int(input("Enter the number of pings: "))
18
19 # Variables to track RTT and packet statistics
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
(base) rakesh@rakesh:~$ sudo /bin/python3 /home/rakesh/Documents/final/udpclient.py
Enter the number of pings: 10
Message received from server: PING 1 2024-09-12 15:50:26.991720
RTT of successful packet 1 is: 329.996 ms
Message received from server: PING 2 2024-09-12 15:50:27.321767
RTT of successful packet 2 is: 17.064 ms
Request for packet 3 timed out
ICMP Destination Host Unreachable error received for packet 3
Message received from server: PING 4 2024-09-12 15:50:28.340436
RTT of successful packet 4 is: 293.991 ms
Request for packet 5 timed out
ICMP Destination Host Unreachable error received for packet 5
Message received from server: PING 6 2024-09-12 15:50:29.635922
RTT of successful packet 6 is: 25.728 ms
Message received from server: PING 7 2024-09-12 15:50:29.661686
RTT of successful packet 7 is: 6.068 ms
Message received from server: PING 8 2024-09-12 15:50:29.667830
RTT of successful packet 8 is: 8.810 ms
Message received from server: PING 9 2024-09-12 15:50:29.676694
RTT of successful packet 9 is: 10.964 ms
Message received from server: PING 10 2024-09-12 15:50:29.687706
RTT of successful packet 10 is: 9.293 ms
Ping Statistics
Maximum RTT: 329.996 ms
Minimum RTT: 6.068 ms
Average RTT: 87.739 ms
Packet loss rate: 20.00%
(base) rakesh@rakesh:~$
```

TCP:

SERVER:

```
1 import socket
2 import struct
3 import random
4 import os
5 import time
6
7 # Function to calculate checksum for ICMP packets
8 def checksum(source_string):
9     count_to = (len(source_string) // 2) * 2
10    sum = 0
11    count = 0
12
13    while count < count_to:
14        this_val = source_string[count + 1] * 256 + source_string[count]
15        sum = sum + this_val
16        sum = sum & 0xffffffff
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL TERMINAL PORTS

```
Random value: 18
Sending ICMP 'Port Unreachable' to ('192.168.241.110', 32824)
Random value: 3
Sending TCP response to ('192.168.241.110', 32824)
Random value: 4
Sending TCP response to ('192.168.241.110', 32824)
Random value: 8
Sending TCP response to ('192.168.241.110', 32824)
Random value: 9
Sending TCP response to ('192.168.241.110', 32824)
Random value: 24
Sending ICMP 'Host Unreachable' to ('192.168.241.110', 32824)
Random value: 21
Sending ICMP 'Host Unreachable' to ('192.168.241.110', 32824)
Random value: 13
Sending TCP response to ('192.168.241.110', 32824)
Random value: 14
Sending TCP response to ('192.168.241.110', 32824)
Random value: 6
Sending TCP response to ('192.168.241.110', 32824)
Random value: 2
Sending TCP response to ('192.168.241.110', 32824)
Random value: 13
Sending TCP response to ('192.168.241.110', 32824)
Random value: 22
Sending ICMP 'Host Unreachable' to ('192.168.241.110', 32824)
Random value: 4
Sending TCP response to ('192.168.241.110', 32824)
Random value: 13
Sending TCP response to ('192.168.241.110', 32824)
Random value: 3
```

CLIENT:

```

1 import socket
2 import time
3 import statistics
4 import struct
5
6 # Function to receive ICMP packets
7 def receive_icmp(clientSocket):
8     raw_data, addr = clientSocket.recvfrom(1024)
9     icmp_header = raw_data[20:28] # Skip the IP header (first 20 bytes)
10
11     icmp_type, icmp_code, icmp_checksum, icmp_unused = struct.unpack('!BBHI', icmp_header)
12
13     if icmp_type == 3:
14         if icmp_code == 3:
15             print(f"Received ICMP 'Port Unreachable' from {addr[0]}")
16         elif icmp_code == 1:
17             print(f"Received ICMP 'Host Unreachable' from {addr[0]}")
18         else:
19             print("Received other ICMP message")

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

Request timed out for packet #42. Checking for ICMP error...
Received ICMP 'Port Unreachable' from 192.168.241.144
Sending TCP packet #43
Received TCP response: PING 43 1726122308.7296839 | RTT: 12.64 ms | Packet #43
Sending TCP packet #44
Received TCP response: PING 44 1726122308.7424257 | RTT: 4.58 ms | Packet #44
Sending TCP packet #45
Received TCP response: PING 45 1726122308.7470732 | RTT: 38.38 ms | Packet #45
Sending TCP packet #46
Received TCP response: PING 46 1726122308.7855666 | RTT: 7.64 ms | Packet #46
Sending TCP packet #47
Received TCP response: PING 47 1726122308.793275 | RTT: 3.18 ms | Packet #47
Sending TCP packet #48
Request timed out for packet #48. Checking for ICMP error...
Received ICMP 'Host Unreachable' from 192.168.241.144
Sending TCP packet #49
Received TCP response: PING 49 1726122309.7980316 | RTT: 3.75 ms | Packet #49
Sending TCP packet #50
Received TCP response: PING 50 1726122309.8019056 | RTT: 5.72 ms | Packet #50

Minimum RTT: 0.13 ms
Maximum RTT: 203.34 ms
Average RTT: 22.36 ms
Packet loss rate: 38.00%
Connection closed.

```

ICMP:

SERVER:

```
sudo iptables -A INPUT -s <source_ip_address> -p icmp -j REJECT --reject-with
<error_response>
```

CLIENT:

```

user@user-VirtualBox:~ $ sudo python 3 /home/user/Desktop/ICMPPingerErrorclient.py
Pinging 192.168.184.169 using Python:

Destination Host Unreachable

--- Ping statistics ---
10 packets transmitted, 0 received, 100.0% packet loss
rtt min = 0.000 ms, rtt avg = 0.000 ms, rtt max = 0.000 ms
user@user-VirtualBox:~ $ |

```

Work Distribution Summary

Task/Section	Group Member Name	Contribution	Challenges faced(if any)
Research & Info gathering	Isha Kumar	Researched about how to simulate packet loss using netem command. Learnt about iptables command.	Learnt about bash permissions & system commands for MacOS while searching on LLMs for the errors I encountered while coding.
Code Development	Isha Kumar	Did coding for UDP Client, modified server & Error response test	Tried to understand ICMP skeleton code using LLMs.
Testing & Development	Isha Kumar	<ul style="list-style-type: none"> Tested the code on the same machine (localhost). 	Had problems while testing the codes using Linux based commands, like iptables, on my

		<ul style="list-style-type: none"> • Tested the code on different machines. • Simulated packet loss using random library & netem command. 	Macbook.
Documentation & Report Writing	Isha Kumar	Documented all the README files for the UDP, TCP & ICMP codes.	Had to learn how to code for README.md markdown file from scratch
Final Review and Submission	Isha Kumar	Reviewed the final report at the end.	

Task/Section	Group Member Name	Contribution	Challenges faced(if any)
Research & Info gathering	Achyut Morang	Researched about how to modify the server code for further enhancements	Learnt about socket library, tc commands, searched for alternatives for iptables to work on macOS terminal
Code Development	Achyut Morang	Simulated packet loss using tc netem command. Tried to simulate 20% packet loss.	
Testing & Development	Achyut Morang	Tested the codes on MacOS	MacOS doesn't support tc and iptables
Documentation & Report Writing	Achyut Morang	Helped teammates in documenting in markdown for README files	Tried documenting everything in markdown

Final Review and Submission	Achyut Morang	Used tar to combine multiple files into an archive file	
-----------------------------	---------------	---	--

Task/Section	Group Member Name	Contribution	Challenges faced(if any)
Research & Info gathering	Rakesh	Gathered information and researched about the ICMP packet from various resources	Needed to look at icmp packet details in depth and understanding it was challenging.
Code Development	Rakesh	ICMP coding part is done along with the error code sending and worked on multi-threading part	Needed to learn threading and implementing it with socket programming was challenging but finally implemented it.
Testing & Development	Rakesh	Tested the ICMP codes using various networks and debugged them to get the desired output.	Testing was challenging with IITH wifi because it blocks ICMP error messages so connected to mobile hotspot and got error messages.
Documentation & Report Writing	Rakesh	Code formatting, comments and added screenshots of the outputs in the documentation.	
Final Review and Submission	Rakesh		

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:

ISHA KUMAR <CS24MTECH14001>

ACHYUT MORANG <SM24MTECH11005>

RAKESH <CS24MTECH11011>

Date: 10 September, 2024

Signatures:

ISHA KUMAR : I.K.

ACHYUT : A.Morang

RAKESH : P.Rakesh

References

1. CHAT GPT references:

ChatGPT

How to solve this error while trying to create a raw socket in MacOS:
PermissionError: [Errno 1] Operation not permitted

On macOS (and other Unix-based systems), creating raw sockets requires elevated privileges because they allow direct access to the network layer. This can be a security risk, so the operating system restricts this capability to administrative users or processes with the necessary permissions.

Steps to Resolve the PermissionError: [Errno 1] Operation not permitted :

1. Run as Root or with sudo : The simplest way to allow the creation of raw sockets is to run your Python script with sudo:
bash
Copy code
sudo python your_script.py
- This gives your script the necessary permissions to create raw sockets.
2. Modify System Settings for Specific Permissions: Instead of running the entire script as root, you can allow specific executables (like Python) to use raw sockets. However, this approach is more complex and can vary based on security policies on your machine. You can modify the system's permissions using commands like setcap on Linux, but macOS doesn't support this command directly.
3. Use pfctl for Packet Filtering and Raw Socket Access: macOS uses pfctl for packet filtering and networking tasks. You can configure rules with pfctl to allow ICMP packets or to create permissions for specific use cases. However, this doesn't grant raw socket permissions directly—it's more about traffic control.
4. System Integrity Protection (SIP): macOS includes System Integrity Protection (SIP), which

Get citation

Message ChatGPT

ChatGPT can make mistakes. Check important info.

ChatGPT

Explain this ICMP skeleton code: from socket import *

```
import os
import sys
import struct
import time
import select
import binascii
```

ICMP_ECHO_REQUEST = 8

```
def checksum(string):
    csum = 0
    countTo = (len(string) // 2) * 2
    count = 0

    while count < countTo:
        thisVal = string[count+1] * 256 + string[count]
        csum = csum + thisVal
        csum = csum & 0xffffffff
        count = count + 2

    if countTo < len(string):
        csum = csum + string[len(string) - 1]
        csum = csum & 0xffffffff

    csum = (csum >> 16) + (csum & 0xffff)
    csum = csum + (csum >> 16)
    answer = ~csum
    answer = answer & 0xffff
    answer = answer >> 8 | (answer << 8 & 0xff00)
```

Get citation

Message ChatGPT

ChatGPT can make mistakes. Check important info.

2.

ChatGPT

The provided ICMP client code is a basic implementation of a ping utility in Python. It uses raw sockets to send ICMP Echo Request packets (pings) to a specified host and waits for ICMP Echo Reply packets (pong) from that host. This code is a simplified version of the `ping` command found in many operating systems and is used to test network connectivity and measure the round-trip time (RTT) of packets.

Here's an explanation of the code's structure and functionality:

1. Imports and Constants

```
python
from socket import *
import os
import sys
import struct
import time
import select
import binascii

ICMP_ECHO_REQUEST = 8
```

- **Imports:**
 - `socket`: Provides functions to create and manipulate raw sockets.
 - `os`: Used for accessing system-related functions, such as retrieving the current process ID.
 - `sys`: For platform-specific operations, like adjusting checksum bytes.
 - `struct`: Helps with packing and unpacking binary data, especially for creating ICMP headers.
 - `time`: For timing and calculating round-trip times (RTT).
 - `select`: Allows waiting for socket readiness to read within a timeout period.

Get citation

Message ChatGPT

ChatGPT can make mistakes. Check important info.

ChatGPT

2. Checksum Calculation

```
python
def checksum(string):
    # Calculates the checksum for the ICMP packet to detect errors
    ...
    return answer
```

- **Purpose:** Computes the checksum for the ICMP packet to ensure data integrity.
- **How It Works:** The function processes the data in 16-bit chunks, sums them, handles any carry bits, and takes the one's complement of the sum to produce the checksum.

3. Receiving a Ping Response

```
python
def receiveOnePing(mySocket, ID, timeout, destAddr):
    # Receives an ICMP Echo Reply within the specified timeout period
    ...
    return timeReceived - timeSent
```

- **Purpose:** Waits for and receives an ICMP Echo Reply from the destination address.
- **How It Works:**
 - Uses `select.select()` to wait for data availability on the socket within the timeout period.
 - If data is received, it extracts the ICMP header and checks if the packet ID matches the sent ping request.
 - If the packet matches, it calculates the RTT by comparing the send and receive timestamps.

Get citation

Message ChatGPT

ChatGPT can make mistakes. Check important info.

The screenshot shows the ChatGPT interface with a dark theme. On the left, there's a sidebar with a tree view of recent projects and a "Upgrade plan" section. The main area has a header "ChatGPT" and a sub-header "4. Sending a Ping Request". Below this is a code block in Python:

```
python
def sendOnePing(mySocket, destAddr, ID):
    # Sends an ICMP Echo Request to the destination address
    ...

```

Followed by a bulleted list:

- Purpose: Constructs and sends an ICMP Echo Request packet to the target host.
- How It Works:
 - Creates an ICMP header with a placeholder checksum.
 - Calculates the checksum of the header and data, then reconstructs the header with the correct checksum.
 - Sends the complete packet (header + data) using `sendto`.

Below this is another section "5. Performing a Single Ping" with a code block:

```
python
def doOnePing(destAddr, timeout):
    # Combines sending and receiving a single ICMP ping
    ...
    return delay

```

And another bulleted list:

- Purpose: Handles the process of sending a ping and receiving the corresponding reply.
- How It Works:
 - Creates a raw socket configured for ICMP.
 - Calls `sendOnePing()` to send the request.
 - Calls `receiveOnePing()` to wait for the reply.
 - Closes the socket and returns the measured RTT or a timeout message.

At the bottom right is a "Get citation" button.

The screenshot continues the ChatGPT interface. The sidebar remains the same. The main area now has a header "6. Main Ping Function". Below it is a code block:

```
python
def ping(host, timeout=1, count=5):
    # Sends multiple pings to the specified host and gathers statistics
    ...
    return delay

```

With a bulleted list:

- Purpose: Orchestrates the entire ping process, sending multiple pings to the specified host and calculating statistics like minimum, maximum, and average RTT.
- How It Works:
 - Resolves the host to its IP address.
 - Repeatedly calls `doOnePing()` for the specified number of times (`count`).
 - Collects RTTs of successful replies and counts packet losses.
 - Displays statistics about packet loss and RTT.

Below this is a section "7. Execution Entry Point" with a code block:

```
python
if __name__ == "__main__":
    ping("www.google.com")

```

With a bulleted list:

- Purpose: This line makes the script executable as a standalone program. It starts the `ping` function with `www.google.com` as the target, sending six pings with a one-second timeout for each.

At the bottom right is a "Get citation" button.

3. <https://www.datacamp.com/tutorial/a-complete-guide-to-socket-programming-in-python>
4. <https://www.youtube.com/watch?v=7gek0eCnbHs&authuser=0>