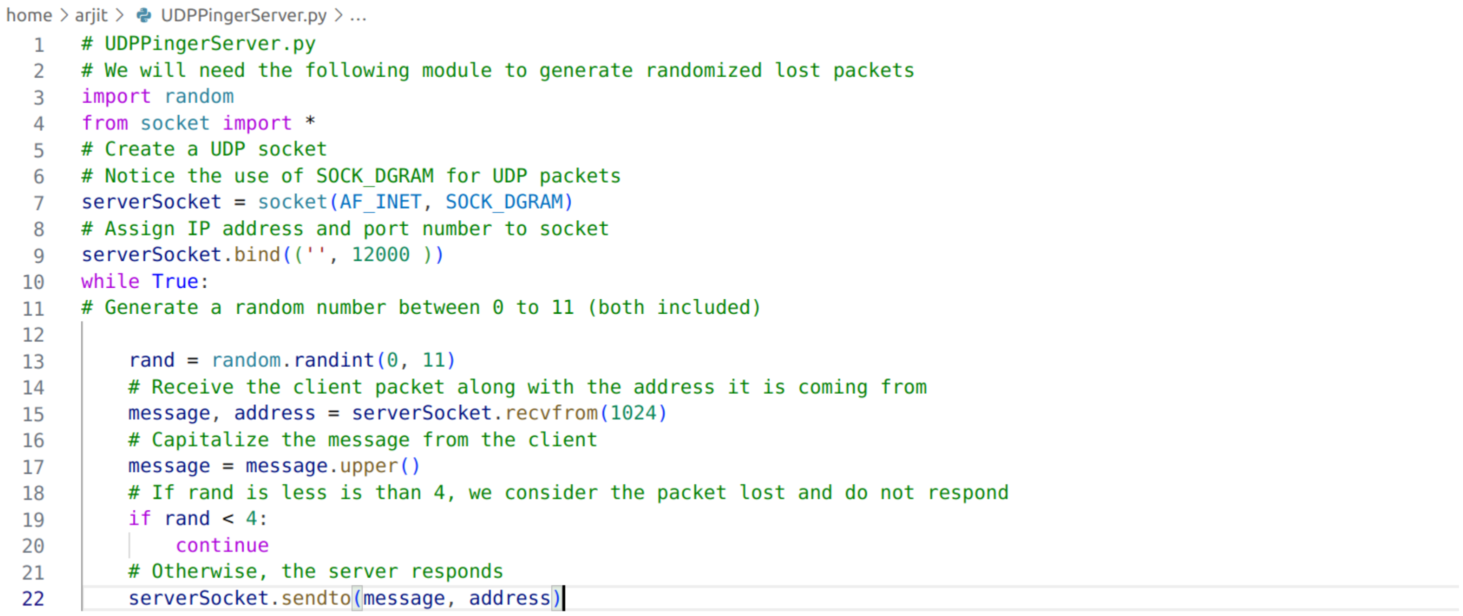
# Programming Assignment 1 - cs23mtech12001

## Task 1- Ping using UDP

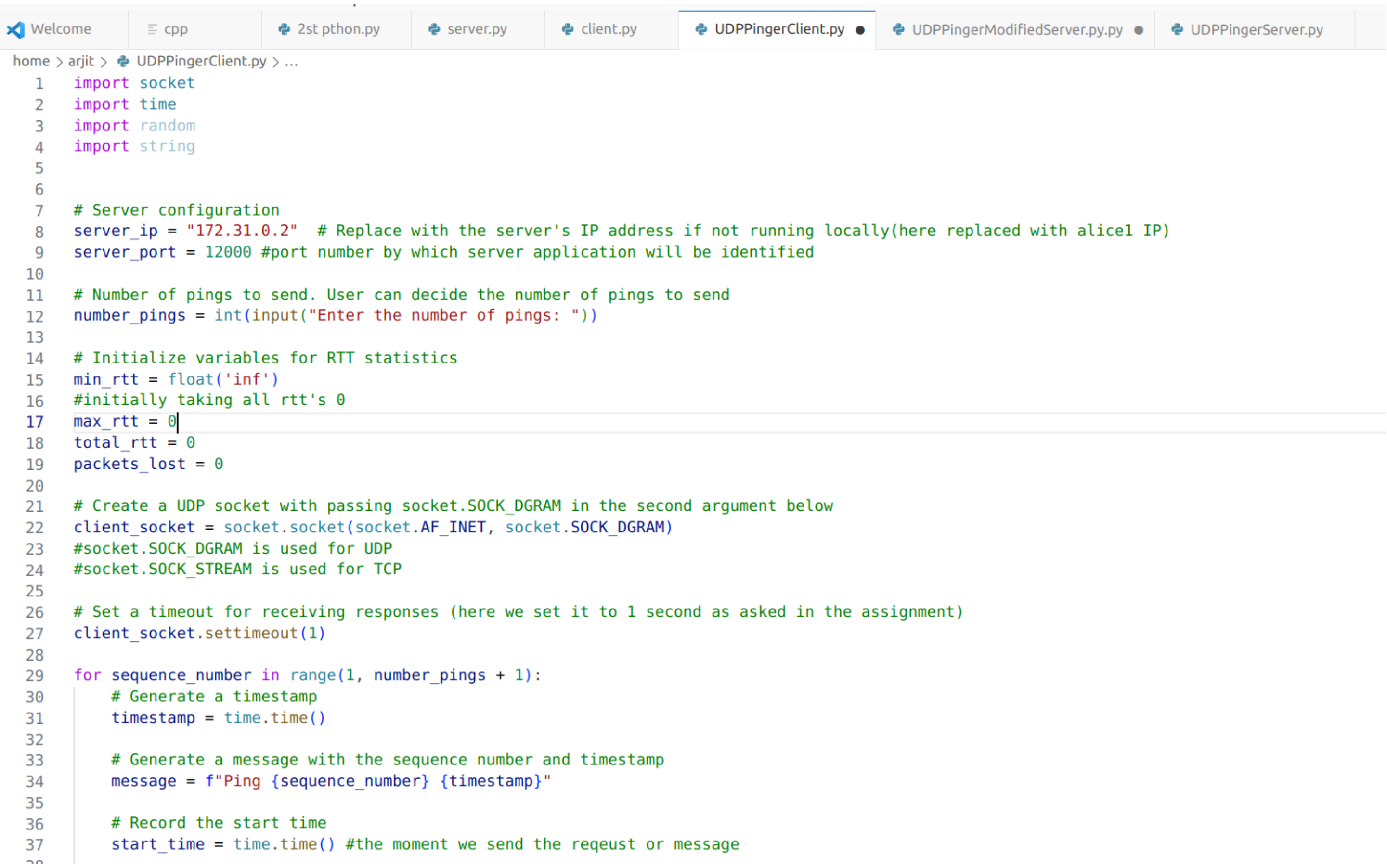
UDPPingerServer.py

Below is the UDP Server code.

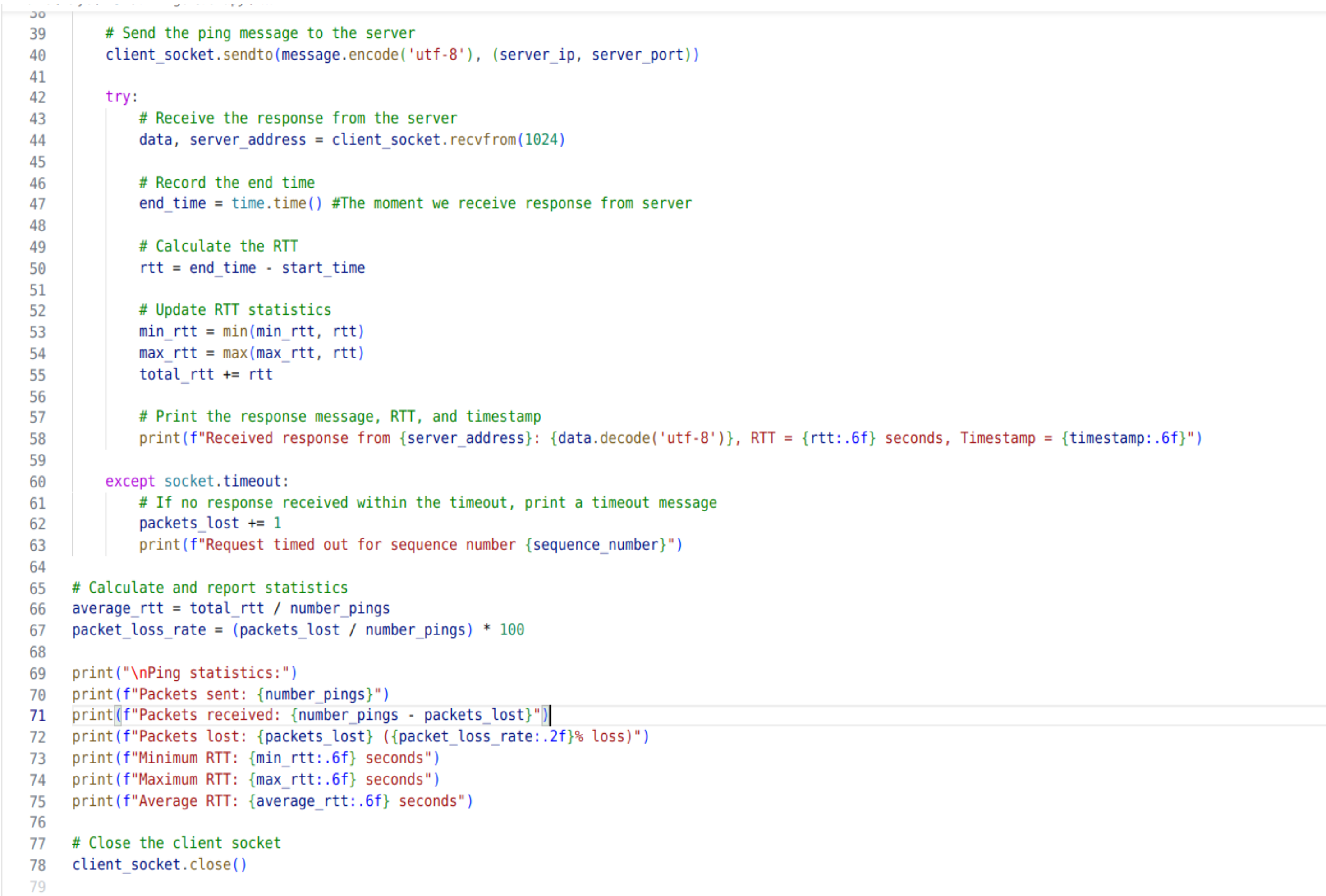


```python
# UDPPingerServer.py
# We will need the following module to generate randomized lost packets
import random
from socket import *
# Create a UDP socket
# Notice the use of SOCK_DGRAM for UDP packets
serverSocket = socket(AF_INET, SOCK_DGRAM)
# Assign IP address and port number to socket
serverSocket.bind(('', 12000 ))
while True:
# Generate a random number between 0 to 11 (both included)

    rand = random.randint(0, 11)
    # Receive the client packet along with the address it is coming from
    message, address = serverSocket.recvfrom(1024)
    # Capitalize the message from the client
    message = message.upper()
    # If rand is less is than 4, we consider the packet lost and do not respond
    if rand < 4:
        continue
    # Otherwise, the server responds
    serverSocket.sendto(message, address)
```

Below is the UDP Client Code  (Client.py)



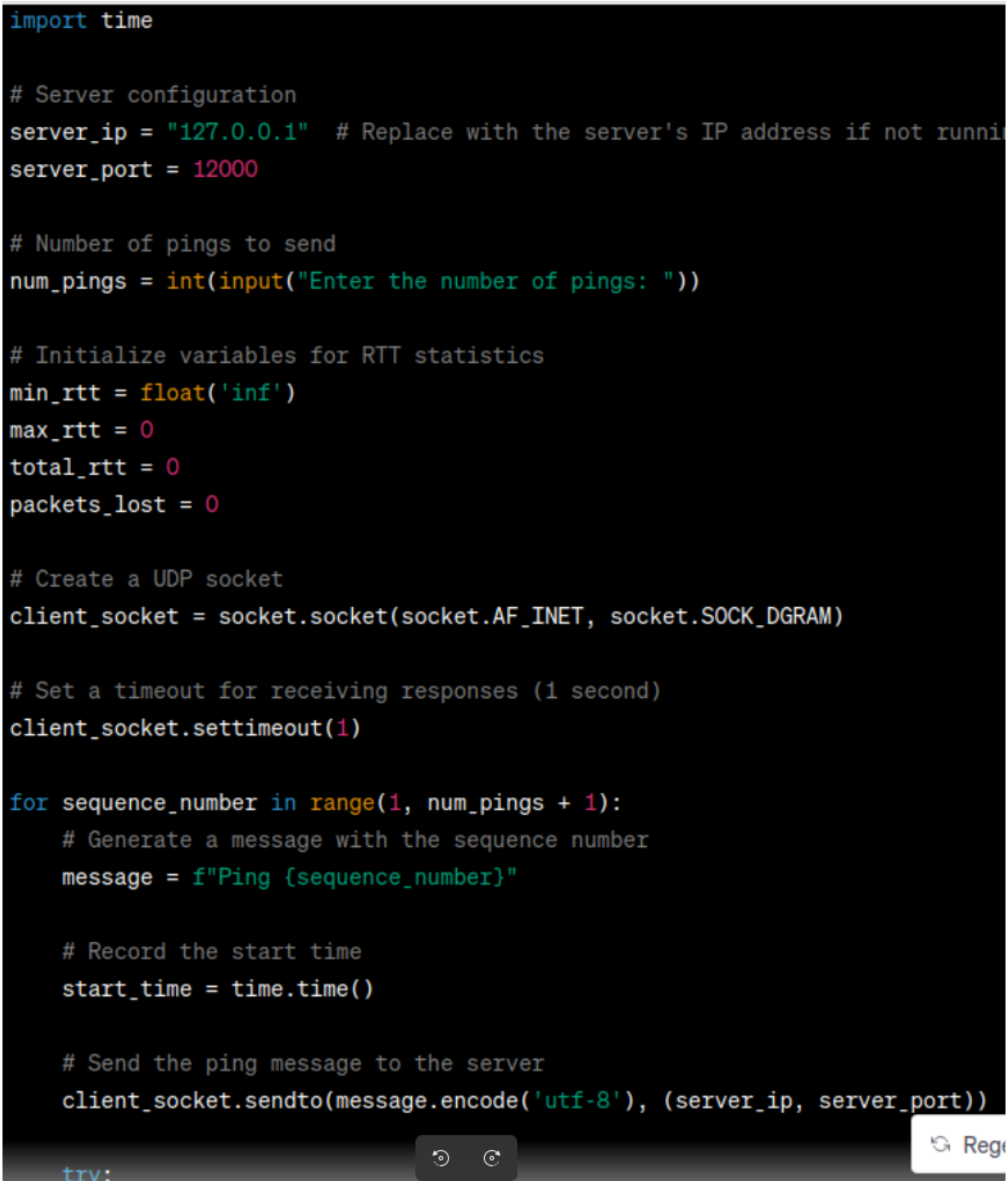
```python
import socket
import time
import random
import string


# Server configuration
server_ip = "172.31.0.2"  # Replace with the server's IP address if not running locally(here replaced with alice1 IP)
server_port = 12000 #port number by which server application will be identified

# Number of pings to send. User can decide the number of pings to send
number_pings = int(input("Enter the number of pings: "))

# Initialize variables for RTT statistics
min_rtt = float('inf')
#initially taking all rtt's 0
max_rtt = 0
total_rtt = 0
packets_lost = 0

# Create a UDP socket with passing socket.SOCK_DGRAM in the second argument below
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
#socket.SOCK_DGRAM is used for UDP
#socket.SOCK_STREAM is used for TCP

# Set a timeout for receiving responses (here we set it to 1 second as asked in the assignment)
client_socket.settimeout(1)

for sequence_number in range(1, number_pings + 1):
    # Generate a timestamp
    timestamp = time.time()

    # Generate a message with the sequence number and timestamp
    message = f"Ping {sequence_number} {timestamp}"

    # Record the start time
    start_time = time.time() #the moment we send the reqeust or message
```



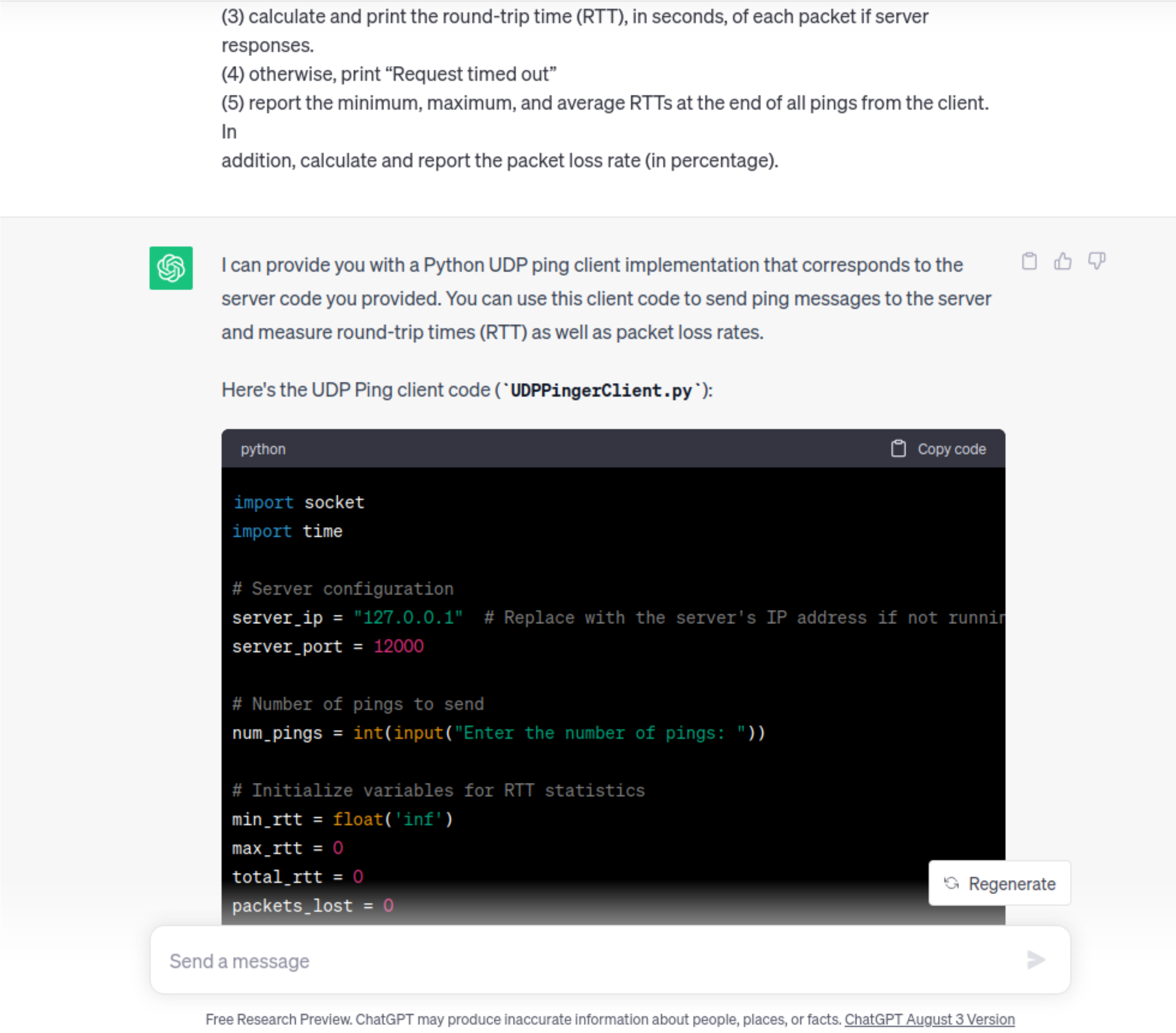
```
    # Send the ping message to the server
    client_socket.sendto(message.encode('utf-8'), (server_ip, server_port))

    try:
        # Receive the response from the server
        data, server_address = client_socket.recvfrom(1024)

        # Record the end time
        end_time = time.time() #The moment we receive response from server

        # Calculate the RTT
        rtt = end_time - start_time

        # Update RTT statistics
        min_rtt = min(min_rtt, rtt)
        max_rtt = max(max_rtt, rtt)
        total_rtt += rtt

        # Print the response message, RTT, and timestamp
        print(f"Received response from {server_address}: {data.decode('utf-8')}, RTT = {rtt:.6f} seconds, Timestamp = {timestamp:.6f}")

    except socket.timeout:
        # If no response received within the timeout, print a timeout message
        packets_lost += 1
        print(f"Request timed out for sequence number {sequence_number}")

# Calculate and report statistics
average_rtt = total_rtt / number_pings
packet_loss_rate = (packets_lost / number_pings) * 100

print("\nPing statistics:")
print(f"Packets sent: {number_pings}")
print(f"Packets received: {number_pings - packets_lost}")
print(f"Packets lost: {packets_lost} ({packet_loss_rate:.2f}% loss)")
print(f"Minimum RTT: {min_rtt:.6f} seconds")
print(f"Maximum RTT: {max_rtt:.6f} seconds")
print(f"Average RTT: {average_rtt:.6f} seconds")

# Close the client socket
client_socket.close()
```

I have used LLM (Chat GPT) for code generation and it was working but I made significant changes in the code and put sufficient efforts to understand the flow and how everything is working. Additionally I have add some features which were not there in the basic code.
Chat GPT Version: Default (GPT-3.5)
Website: [https://chat.openai.com/](https://chat.openai.com/)
Chat Conversation link for reference:
[https://chat.openai.com/share/7a3b702d-7704-4320-9e9a-6ee0a55c9403](https://chat.openai.com/share/7a3b702d-7704-4320-9e9a-6ee0a55c9403)

Below are the Screenshots of Chat GPT



```
import time

# Server configuration
server_ip = "127.0.0.1"  # Replace with the server's IP address if not runni
server_port = 12000

# Number of pings to send
num_pings = int(input("Enter the number of pings: "))

# Initialize variables for RTT statistics
min_rtt = float('inf')
max_rtt = 0
total_rtt = 0
packets_lost = 0

# Create a UDP socket
client_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

# Set a timeout for receiving responses (1 second)
client_socket.settimeout(1)

for sequence_number in range(1, num_pings + 1):
    # Generate a message with the sequence number
    message = f"Ping {sequence_number}"

    # Record the start time
    start_time = time.time()

    # Send the ping message to the server
    client_socket.sendto(message.encode('utf-8'), (server_ip, server_port))

    try:
```



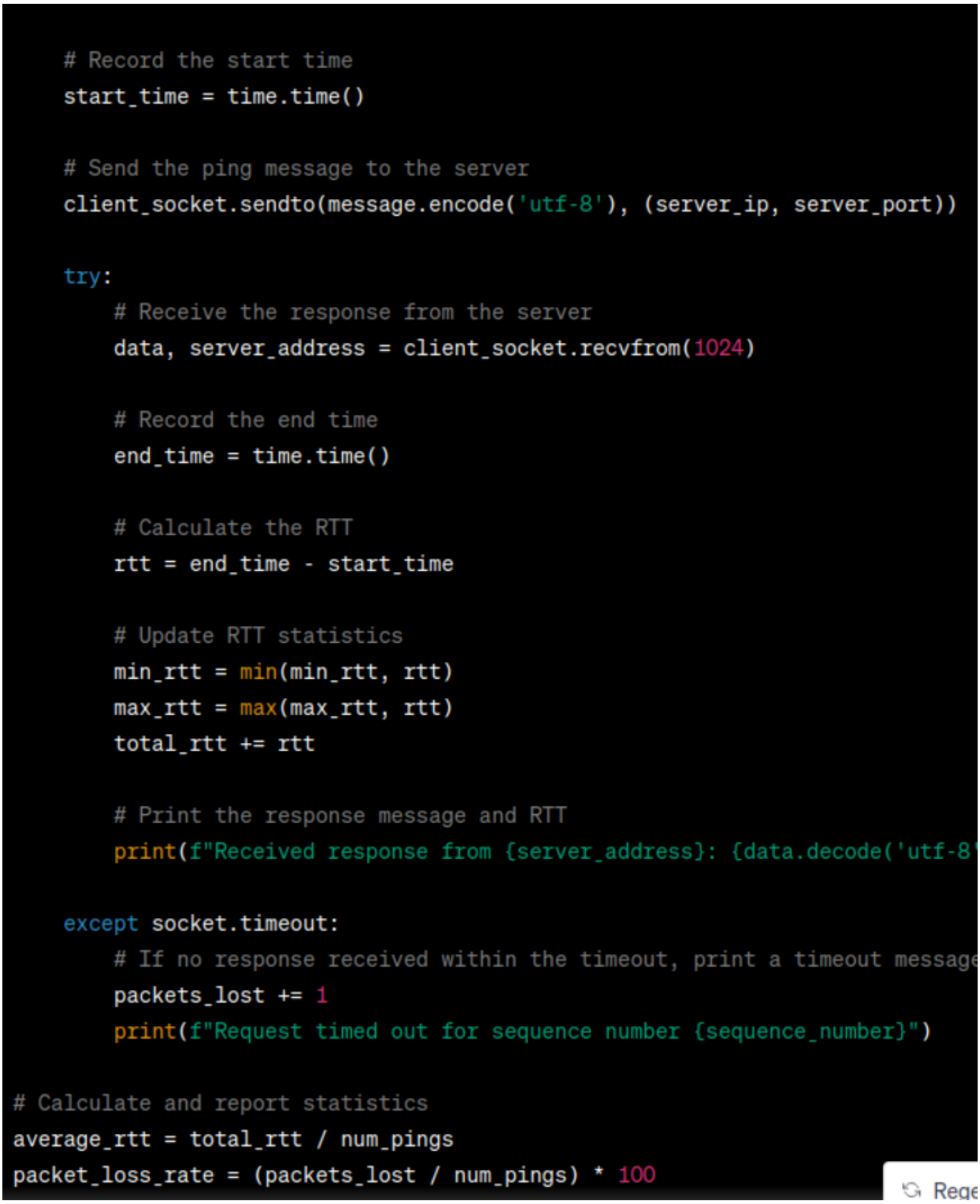
(3) calculate and print the round-trip time (RTT), in seconds, of each packet if server responses.
(4) otherwise, print “Request timed out”
(5) report the minimum, maximum, and average RTTs at the end of all pings from the client. In
addition, calculate and report the packet loss rate (in percentage).

I can provide you with a Python UDP ping client implementation that corresponds to the server code you provided. You can use this client code to send ping messages to the server and measure round-trip times (RTT) as well as packet loss rates.

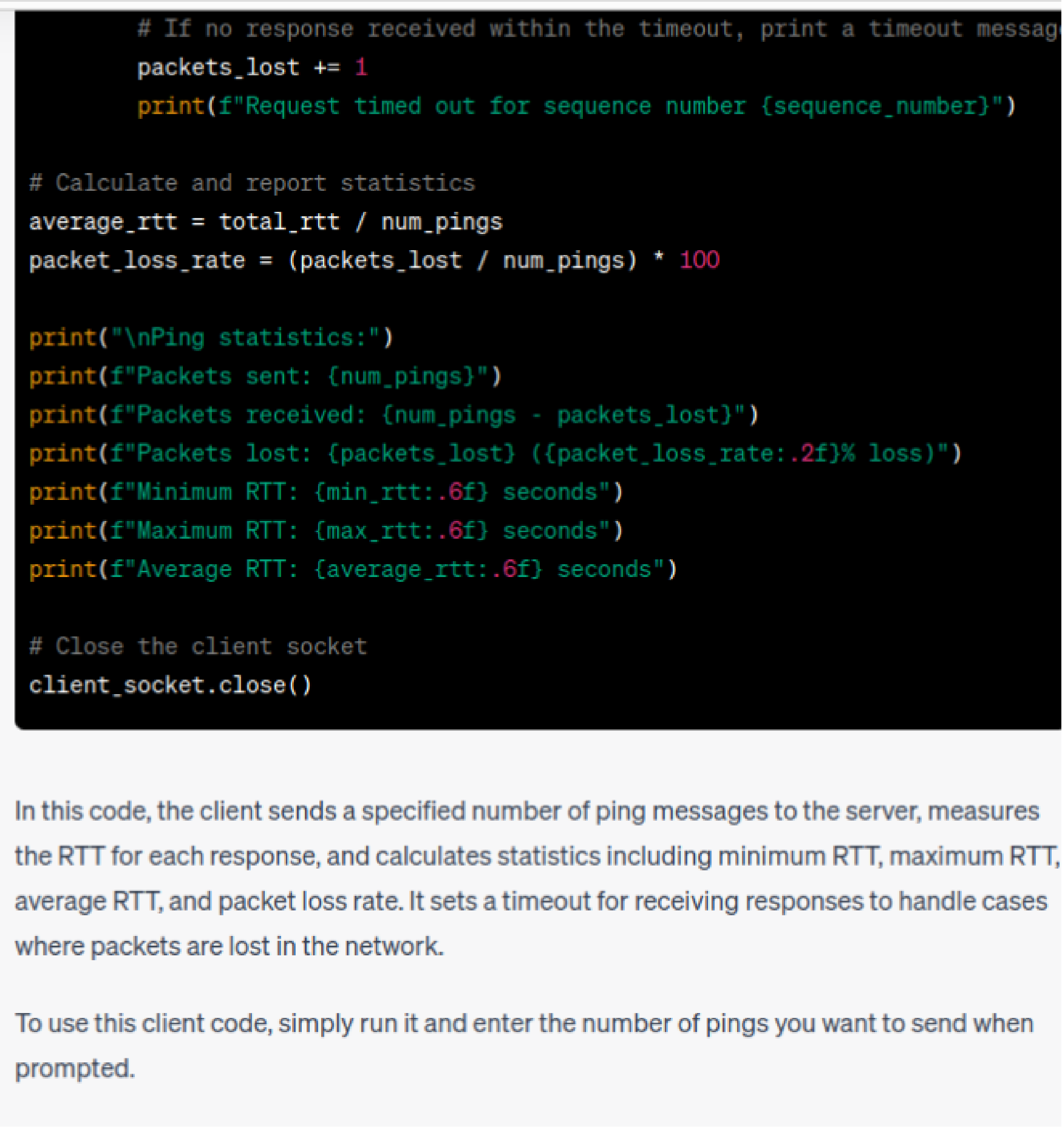
Here's the UDP Ping client code (`UDPPingerClient.py`):

```python
import socket
import time

# Server configuration
server_ip = "127.0.0.1"  # Replace with the server's IP address if not runni
server_port = 12000

# Number of pings to send
num_pings = int(input("Enter the number of pings: "))

# Initialize variables for RTT statistics
min_rtt = float('inf')
max_rtt = 0
total_rtt = 0
packets_lost = 0
```

Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT August 3 Version



```
    # Record the start time
    start_time = time.time()

    # Send the ping message to the server
    client_socket.sendto(message.encode('utf-8'), (server_ip, server_port))

    try:
        # Receive the response from the server
        data, server_address = client_socket.recvfrom(1024)

        # Record the end time
        end_time = time.time()

        # Calculate the RTT
        rtt = end_time - start_time

        # Update RTT statistics
        min_rtt = min(min_rtt, rtt)
        max_rtt = max(max_rtt, rtt)
        total_rtt += rtt

        # Print the response message and RTT
        print(f"Received response from {server_address}: {data.decode('utf-8'

    except socket.timeout:
        # If no response received within the timeout, print a timeout messag
        packets_lost += 1
        print(f"Request timed out for sequence number {sequence_number}")

# Calculate and report statistics
average_rtt = total_rtt / num_pings
packet_loss_rate = (packets_lost / num_pings) * 100
```



```
        # If no response received within the timeout, print a timeout messag
        packets_lost += 1
        print(f"Request timed out for sequence number {sequence_number}")

# Calculate and report statistics
average_rtt = total_rtt / num_pings
packet_loss_rate = (packets_lost / num_pings) * 100

print("\nPing statistics:")
print(f"Packets sent: {num_pings}")
print(f"Packets received: {num_pings - packets_lost}")
print(f"Packets lost: {packets_lost} ({packet_loss_rate:.2f}% loss)")
print(f"Minimum RTT: {min_rtt:.6f} seconds")
print(f"Maximum RTT: {max_rtt:.6f} seconds")
print(f"Average RTT: {average_rtt:.6f} seconds")

# Close the client socket
client_socket.close()
```

In this code, the client sends a specified number of ping messages to the server, measures the RTT for each response, and calculates statistics including minimum RTT, maximum RTT, average RTT, and packet loss rate. It sets a timeout for receiving responses to handle cases where packets are lost in the network.

To use this client code, simply run it and enter the number of pings you want to send when prompted.

## Modified UDP Server

In this, there is no random function used in server code, instead, I used the following command before and after  running the server code to inject 33% packet loss at NIC level.

tc qdisc add dev eth0 root netem loss 33%l

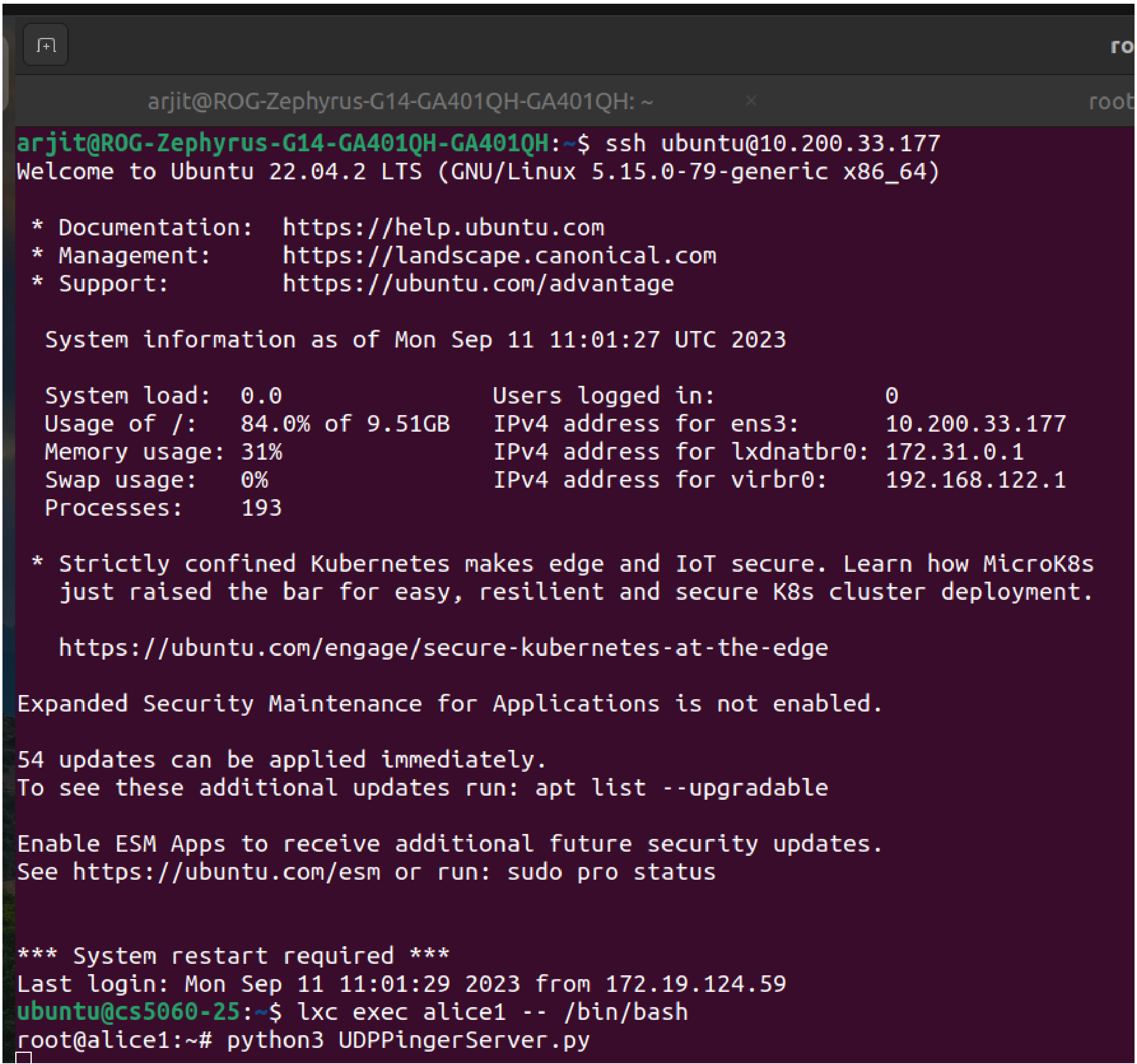
tc qdisc del dev eth0 root netem loss 33%

## Output-

**alice1 is Server**
**bob1 is Client**

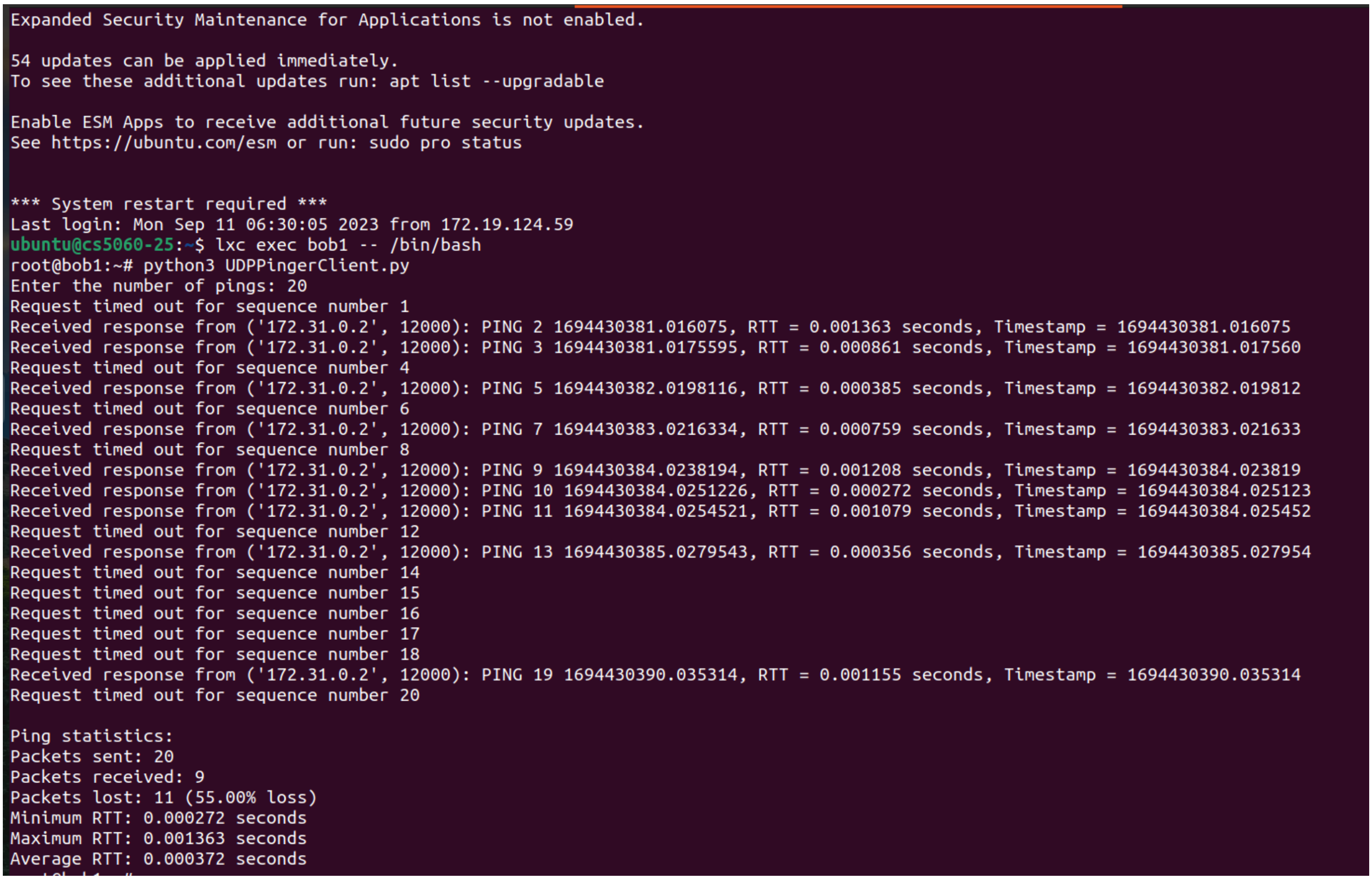
## 1)UDP

### UDPPingerServer.py Running on alice1



```
arjit@ROG-Zephyrus-G14-GA401QH-GA401QH:~$ ssh ubuntu@10.200.33.177
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-79-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Mon Sep 11 11:01:27 UTC 2023

  System load:  0.0               Users logged in:           0
  Usage of /:   84.0% of 9.51GB   IPv4 address for ens3:     10.200.33.177
  Memory usage: 31%               IPv4 address for lxdnatbr0: 172.31.0.1
  Swap usage:   0%                IPv4 address for virbr0:   192.168.122.1
  Processes:    193

 * Strictly confined Kubernetes makes edge and IoT secure. Learn how MicroK8s
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

54 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***
Last login: Mon Sep 11 11:01:29 2023 from 172.19.124.59
ubuntu@cs5060-25:~$ lxc exec alice1 -- /bin/bash
root@alice1:~# python3 UDPPingerServer.py
```

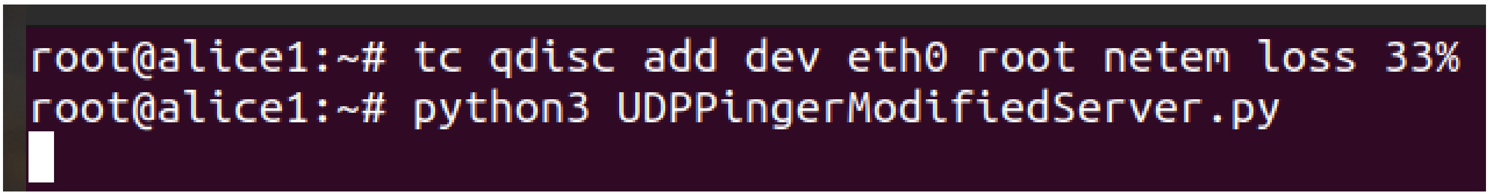
### UDPPingerClient.py Running on bob1



```
Expanded Security Maintenance for Applications is not enabled.

54 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***
Last login: Mon Sep 11 06:30:05 2023 from 172.19.124.59
ubuntu@cs5060-25:~$ lxc exec bob1 -- /bin/bash
root@bob1:~# python3 UDPPingerClient.py
Enter the number of pings: 20
Request timed out for sequence number 1
Received response from ('172.31.0.2', 12000): PING 2 1694430381.016075, RTT = 0.001363 seconds, Timestamp = 1694430381.016075
Received response from ('172.31.0.2', 12000): PING 3 1694430381.0175595, RTT = 0.000861 seconds, Timestamp = 1694430381.017560
Request timed out for sequence number 4
Received response from ('172.31.0.2', 12000): PING 5 1694430382.0198116, RTT = 0.000385 seconds, Timestamp = 1694430382.019812
Request timed out for sequence number 6
Received response from ('172.31.0.2', 12000): PING 7 1694430383.0216334, RTT = 0.000759 seconds, Timestamp = 1694430383.021633
Request timed out for sequence number 8
Received response from ('172.31.0.2', 12000): PING 9 1694430384.0238194, RTT = 0.001208 seconds, Timestamp = 1694430384.023819
Received response from ('172.31.0.2', 12000): PING 10 1694430384.0251226, RTT = 0.000272 seconds, Timestamp = 1694430384.025123
Received response from ('172.31.0.2', 12000): PING 11 1694430384.0254521, RTT = 0.001079 seconds, Timestamp = 1694430384.025452
Request timed out for sequence number 12
Received response from ('172.31.0.2', 12000): PING 13 1694430385.0279543, RTT = 0.000356 seconds, Timestamp = 1694430385.027954
Request timed out for sequence number 14
Request timed out for sequence number 15
Request timed out for sequence number 16
Request timed out for sequence number 17
Request timed out for sequence number 18
Received response from ('172.31.0.2', 12000): PING 19 1694430390.035314, RTT = 0.001155 seconds, Timestamp = 1694430390.035314
Request timed out for sequence number 20

Ping statistics:
Packets sent: 20
Packets received: 9
Packets lost: 11 (55.00% loss)
Minimum RTT: 0.000272 seconds
Maximum RTT: 0.001363 seconds
Average RTT: 0.000372 seconds
```

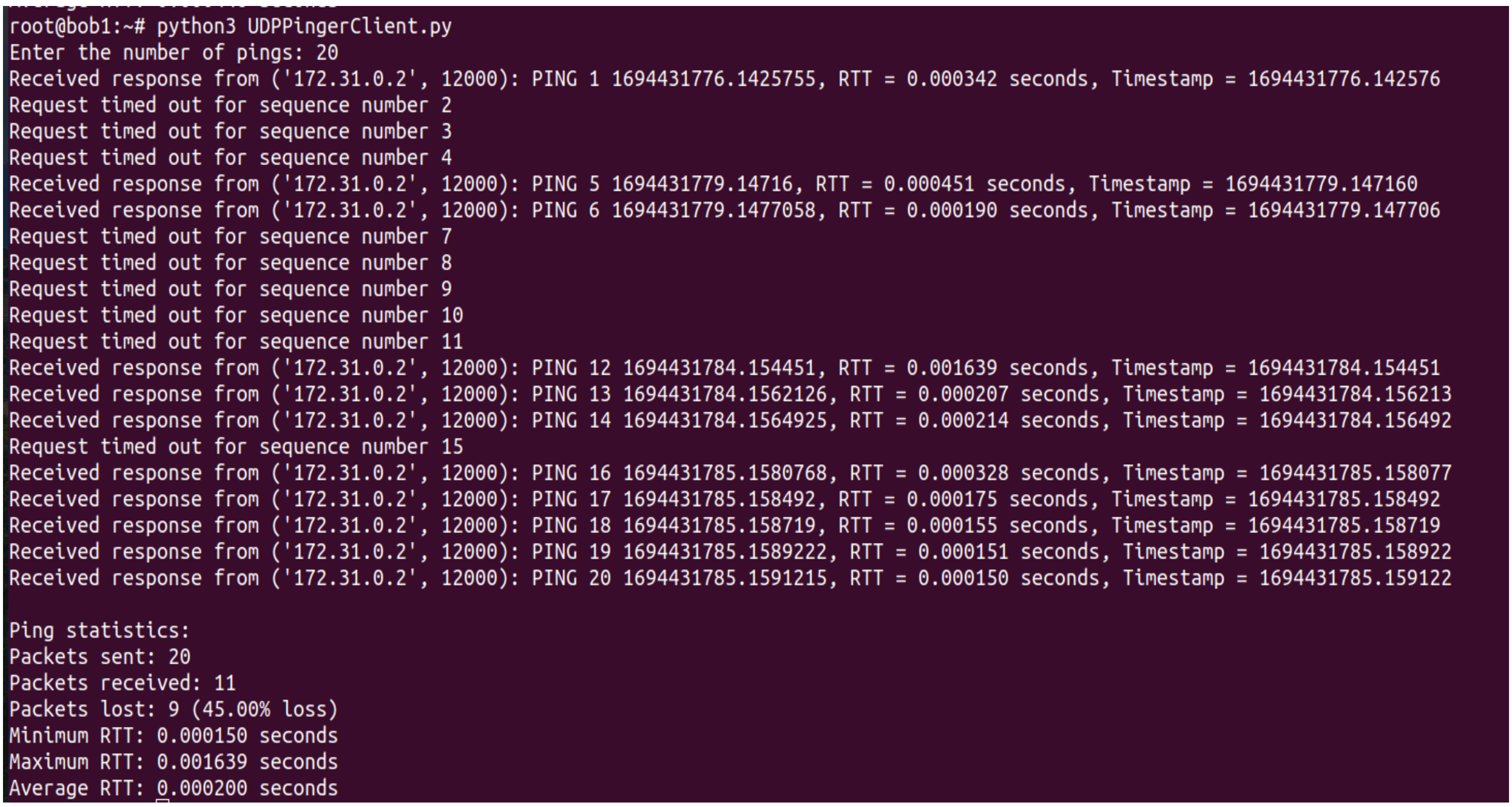
## UDPPingerModifiedServer

### UDPPingerModifiedServer.py Running on alice1



```
root@alice1:~# tc qdisc add dev eth0 root netem loss 33%
root@alice1:~# python3 UDPPingerModifiedServer.py
```

## UDPPingerClient.py Running on bob1



```
root@bob1:~# python3 UDPPingerClient.py
Enter the number of pings: 20
Received response from ('172.31.0.2', 12000): PING 1 1694431776.1425755, RTT = 0.000342 seconds, Timestamp = 1694431776.142576
Request timed out for sequence number 2
Request timed out for sequence number 3
Request timed out for sequence number 4
Received response from ('172.31.0.2', 12000): PING 5 1694431779.14716, RTT = 0.000451 seconds, Timestamp = 1694431779.147160
Received response from ('172.31.0.2', 12000): PING 6 1694431779.1477058, RTT = 0.000190 seconds, Timestamp = 1694431779.147706
Request timed out for sequence number 7
Request timed out for sequence number 8
Request timed out for sequence number 9
Request timed out for sequence number 10
Request timed out for sequence number 11
Received response from ('172.31.0.2', 12000): PING 12 1694431784.154451, RTT = 0.001639 seconds, Timestamp = 1694431784.154451
Received response from ('172.31.0.2', 12000): PING 13 1694431784.1562126, RTT = 0.000207 seconds, Timestamp = 1694431784.156213
Received response from ('172.31.0.2', 12000): PING 14 1694431784.1564925, RTT = 0.000214 seconds, Timestamp = 1694431784.156492
Request timed out for sequence number 15
Received response from ('172.31.0.2', 12000): PING 16 1694431785.1580768, RTT = 0.000328 seconds, Timestamp = 1694431785.158077
Received response from ('172.31.0.2', 12000): PING 17 1694431785.158492, RTT = 0.000175 seconds, Timestamp = 1694431785.158492
Received response from ('172.31.0.2', 12000): PING 18 1694431785.158719, RTT = 0.000155 seconds, Timestamp = 1694431785.158719
Received response from ('172.31.0.2', 12000): PING 19 1694431785.1589222, RTT = 0.000151 seconds, Timestamp = 1694431785.158922
Received response from ('172.31.0.2', 12000): PING 20 1694431785.1591215, RTT = 0.000150 seconds, Timestamp = 1694431785.159122

Ping statistics:
Packets sent: 20
Packets received: 11
Packets lost: 9 (45.00% loss)
Minimum RTT: 0.000150 seconds
Maximum RTT: 0.001639 seconds
Average RTT: 0.000200 seconds
```

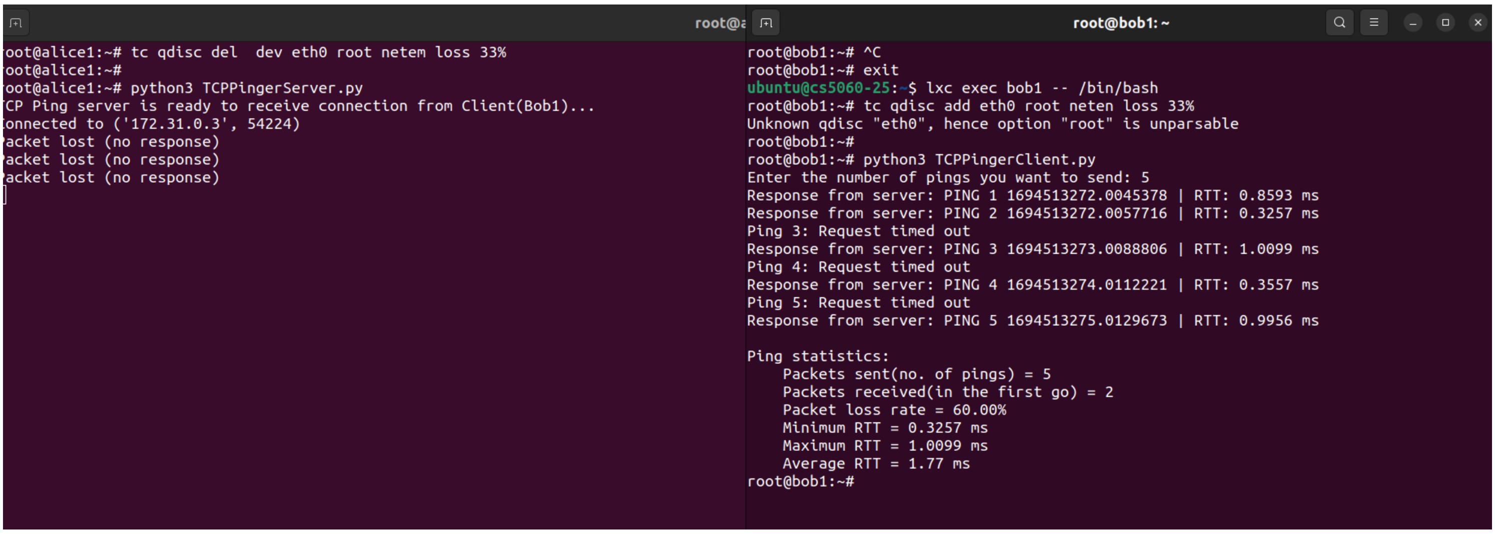
UDP Complete

---

# Task 2- TCP

```
I used Chat GPT for the basic code generation and then edited the code accordingly where changes were needed.
```

a) **TCP Server running on alice1**
b) **TCP Client running in bob1**



```
oot@alice1:~# tc qdisc del  dev eth0 root netem loss 33%
oot@alice1:~#
oot@alice1:~# python3 TCPPingerServer.py
CP Ping server is ready to receive connection from Client(Bob1)...
onnected to ('172.31.0.3', 54224)
acket lost (no response)
acket lost (no response)
acket lost (no response)
```

```
root@bob1:~# ^C
root@bob1:~# exit
ubuntu@cs5060-25:~$ lxc exec bob1 -- /bin/bash
root@bob1:~# tc qdisc add eth0 root neten loss 33%
Unknown qdisc "eth0", hence option "root" is unparsable
root@bob1:~#
root@bob1:~# python3 TCPPingerClient.py
Enter the number of pings you want to send: 5
Response from server: PING 1 1694513272.0045378 | RTT: 0.8593 ms
Response from server: PING 2 1694513272.0057716 | RTT: 0.3257 ms
Ping 3: Request timed out
Response from server: PING 3 1694513273.0088806 | RTT: 1.0099 ms
Ping 4: Request timed out
Response from server: PING 4 1694513274.0112221 | RTT: 0.3557 ms
Ping 5: Request timed out
Response from server: PING 5 1694513275.0129673 | RTT: 0.9956 ms

Ping statistics:
    Packets sent(no. of pings) = 5
    Packets received(in the first go) = 2
    Packet loss rate = 60.00%
    Minimum RTT = 0.3257 ms
    Maximum RTT = 1.0099 ms
    Average RTT = 1.77 ms
root@bob1:~#
```

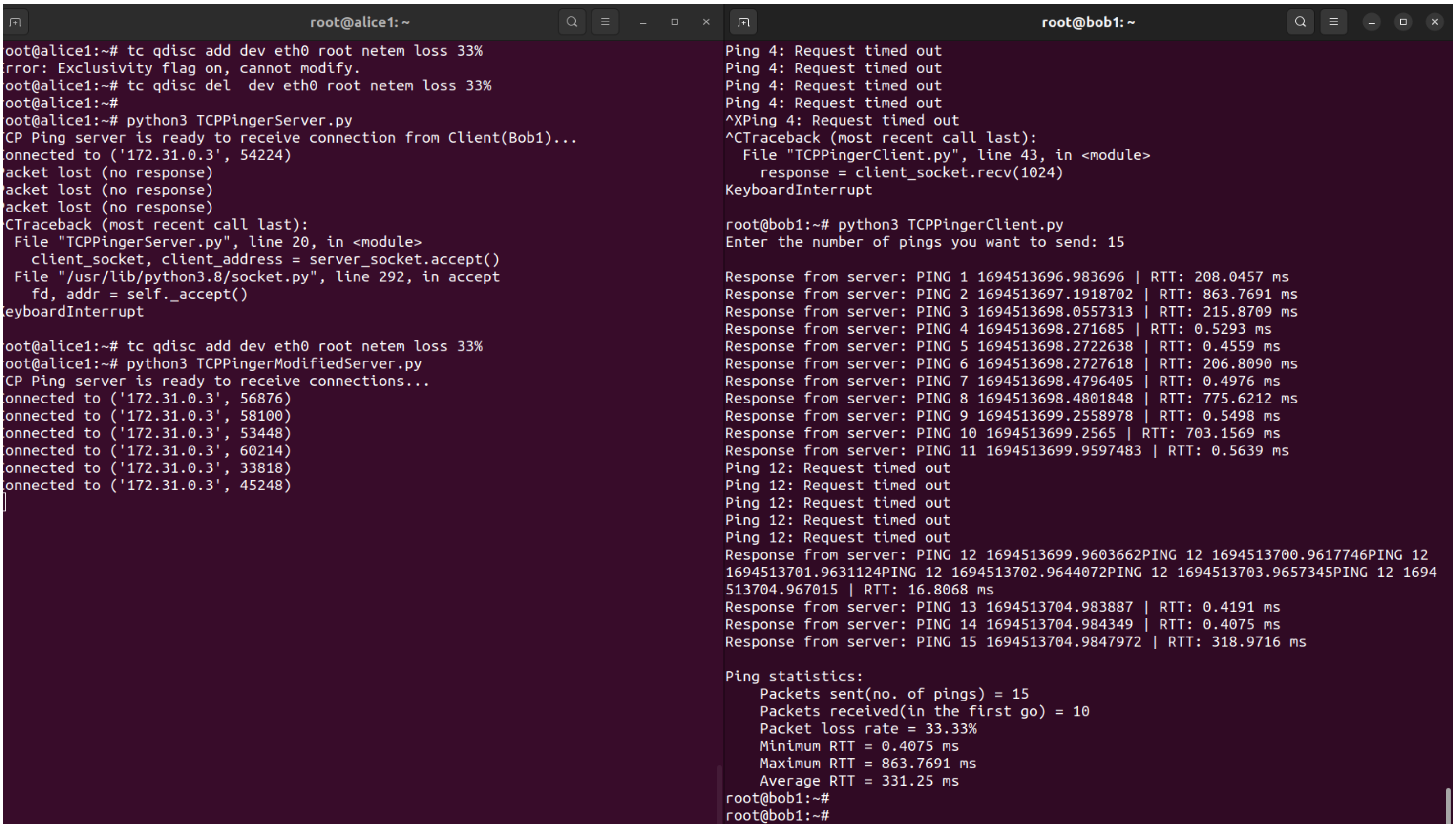
## c) Modified Server

## Including the following command

**tc qdisc add dev eth0 root netem loss 33%**

**tc qdisc del dev eth0 root netem loss 33%**

## Below image shows the output of TCP Modified Server

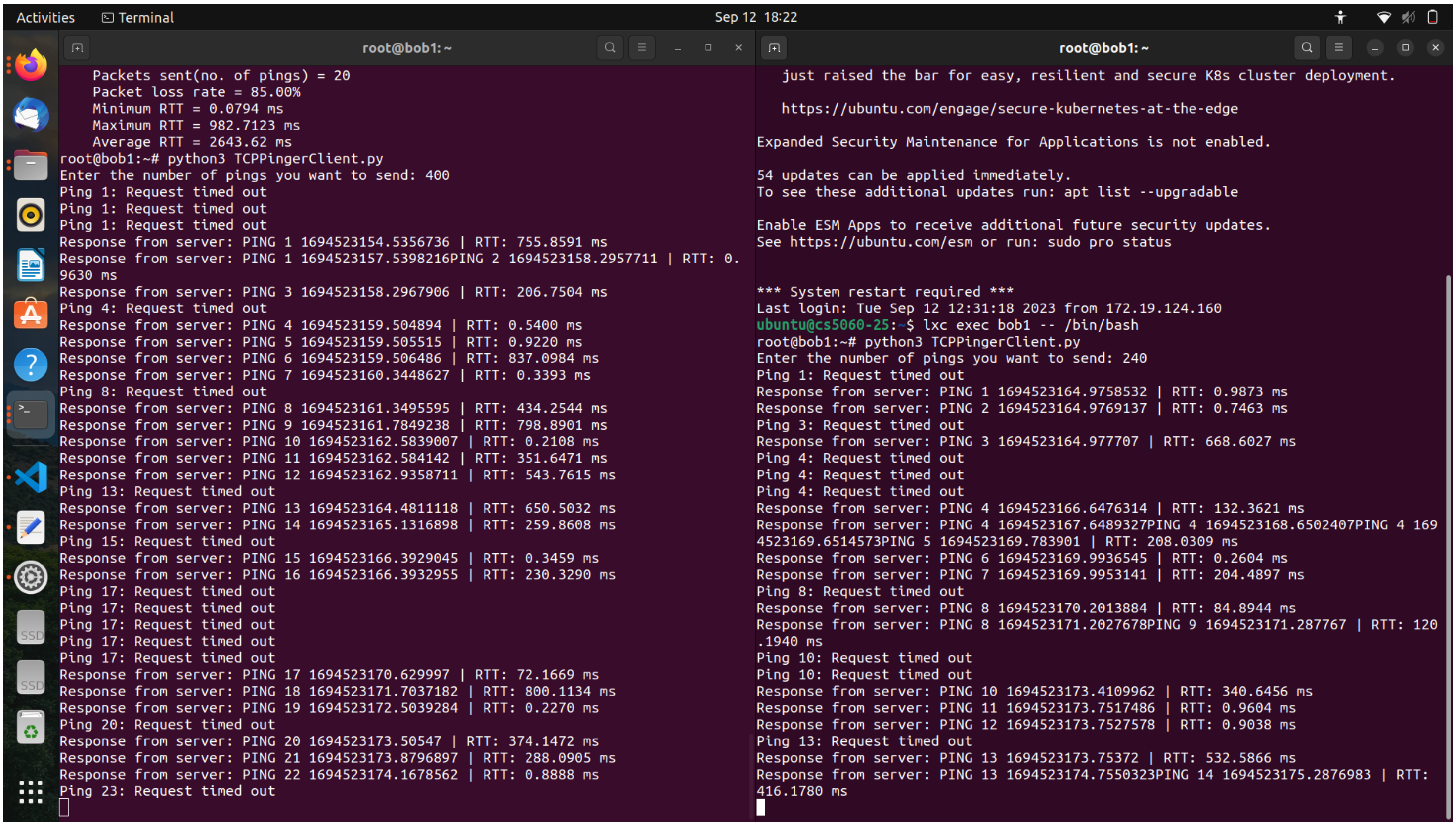


```
oot@alice1:~# tc qdisc add dev eth0 root netem loss 33%
rror: Exclusivity flag on, cannot modify.
oot@alice1:~# tc qdisc del  dev eth0 root netem loss 33%
oot@alice1:~#
oot@alice1:~# python3 TCPPingerServer.py
CP Ping server is ready to receive connection from Client(Bob1)...
onnected to ('172.31.0.3', 54224)
acket lost (no response)
acket lost (no response)
acket lost (no response)
CTraceback (most recent call last):
 File "TCPPingerServer.py", line 20, in <module>
   client_socket, client_address = server_socket.accept()
 File "/usr/lib/python3.8/socket.py", line 292, in accept
   fd, addr = self._accept()
eyboardInterrupt

oot@alice1:~# tc qdisc add dev eth0 root netem loss 33%
oot@alice1:~# python3 TCPPingerModifiedServer.py
CP Ping server is ready to receive connections...
onnected to ('172.31.0.3', 56876)
onnected to ('172.31.0.3', 58100)
onnected to ('172.31.0.3', 53448)
onnected to ('172.31.0.3', 60214)
onnected to ('172.31.0.3', 33818)
onnected to ('172.31.0.3', 45248)
```

```
Ping 4: Request timed out
Ping 4: Request timed out
Ping 4: Request timed out
Ping 4: Request timed out
^XPing 4: Request timed out
^CTraceback (most recent call last):
  File "TCPPingerClient.py", line 43, in <module>
    response = client_socket.recv(1024)
KeyboardInterrupt

root@bob1:~# python3 TCPPingerClient.py
Enter the number of pings you want to send: 15

Response from server: PING 1 1694513696.983696 | RTT: 208.0457 ms
Response from server: PING 2 1694513697.1918702 | RTT: 863.7691 ms
Response from server: PING 3 1694513698.0557313 | RTT: 215.8709 ms
Response from server: PING 4 1694513698.271685 | RTT: 0.5293 ms
Response from server: PING 5 1694513698.2722638 | RTT: 0.4559 ms
Response from server: PING 6 1694513698.2727618 | RTT: 206.8090 ms
Response from server: PING 7 1694513698.4796405 | RTT: 0.4976 ms
Response from server: PING 8 1694513698.4801848 | RTT: 775.6212 ms
Response from server: PING 9 1694513699.2558978 | RTT: 0.5498 ms
Response from server: PING 10 1694513699.2565 | RTT: 703.1569 ms
Response from server: PING 11 1694513699.9597483 | RTT: 0.5639 ms
Ping 12: Request timed out
Ping 12: Request timed out
Ping 12: Request timed out
Ping 12: Request timed out
Ping 12: Request timed out
Response from server: PING 12 1694513699.9603662PING 12 1694513700.9617746PING 12
1694513701.9631124PING 12 1694513702.9644072PING 12 1694513703.9657345PING 12 1694
513704.967015 | RTT: 16.8068 ms
Response from server: PING 13 1694513704.983887 | RTT: 0.4191 ms
Response from server: PING 14 1694513704.984349 | RTT: 0.4075 ms
Response from server: PING 15 1694513704.9847972 | RTT: 318.9716 ms

Ping statistics:
    Packets sent(no. of pings) = 15
    Packets received(in the first go) = 10
    Packet loss rate = 33.33%
    Minimum RTT = 0.4075 ms
    Maximum RTT = 863.7691 ms
    Average RTT = 331.25 ms
root@bob1:~#
root@bob1:~#
```

## TCP Concurrent

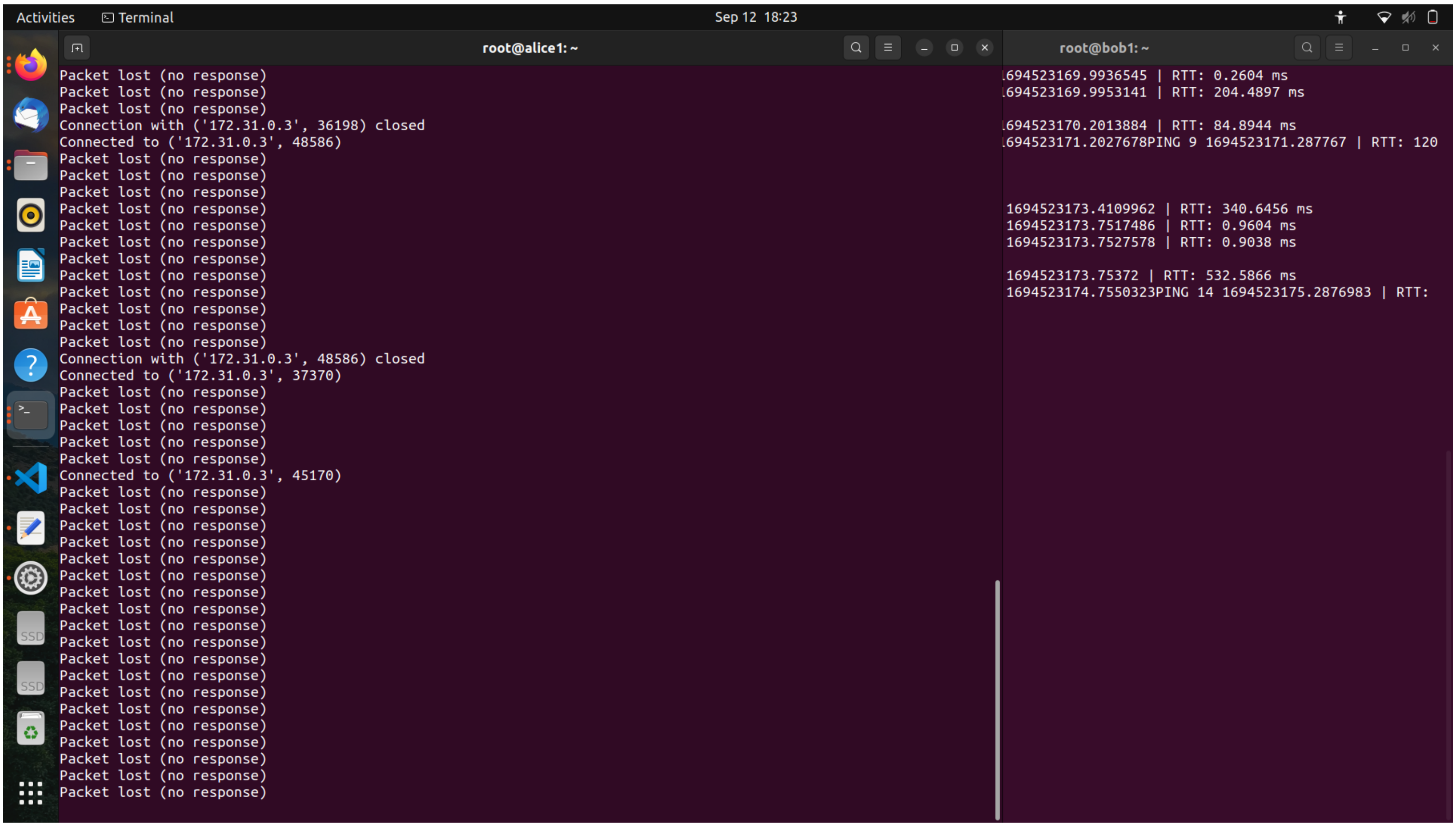
## The below image shows two concurrent clients running on bob1



```
    Packets sent(no. of pings) = 20
    Packet loss rate = 85.00%
    Minimum RTT = 0.0794 ms
    Maximum RTT = 982.7123 ms
    Average RTT = 2643.62 ms
root@bob1:~# python3 TCPPingerClient.py
Enter the number of pings you want to send: 400
Ping 1: Request timed out
Ping 1: Request timed out
Ping 1: Request timed out
Response from server: PING 1 1694523154.5356736 | RTT: 755.8591 ms
Response from server: PING 1 1694523157.5398216PING 2 1694523158.2957711 | RTT: 0.
9630 ms
Response from server: PING 3 1694523158.2967906 | RTT: 206.7504 ms
Ping 4: Request timed out
Response from server: PING 4 1694523159.504894 | RTT: 0.5400 ms
Response from server: PING 5 1694523159.505515 | RTT: 0.9220 ms
Response from server: PING 6 1694523159.506486 | RTT: 837.0984 ms
Response from server: PING 7 1694523160.3448627 | RTT: 0.3393 ms
Ping 8: Request timed out
Response from server: PING 8 1694523161.3495595 | RTT: 434.2544 ms
Response from server: PING 9 1694523161.7849238 | RTT: 798.8901 ms
Response from server: PING 10 1694523162.5839007 | RTT: 0.2108 ms
Response from server: PING 11 1694523162.584142 | RTT: 351.6471 ms
Response from server: PING 12 1694523162.9358711 | RTT: 543.7615 ms
Ping 13: Request timed out
Response from server: PING 13 1694523164.4811118 | RTT: 650.5032 ms
Response from server: PING 14 1694523165.1316898 | RTT: 259.8608 ms
Ping 15: Request timed out
Response from server: PING 15 1694523166.3929045 | RTT: 0.3459 ms
Response from server: PING 16 1694523166.3932955 | RTT: 230.3290 ms
Ping 17: Request timed out
Ping 17: Request timed out
Ping 17: Request timed out
Ping 17: Request timed out
Ping 17: Request timed out
Response from server: PING 17 1694523170.629997 | RTT: 72.1669 ms
Response from server: PING 18 1694523171.7037182 | RTT: 800.1134 ms
Response from server: PING 19 1694523172.5039284 | RTT: 0.2270 ms
Ping 20: Request timed out
Response from server: PING 20 1694523173.50547 | RTT: 374.1472 ms
Response from server: PING 21 1694523173.8796897 | RTT: 288.0905 ms
Response from server: PING 22 1694523174.1678562 | RTT: 0.8888 ms
Ping 23: Request timed out
```

```
   just raised the bar for easy, resilient and secure K8s cluster deployment.

   https://ubuntu.com/engage/secure-kubernetes-at-the-edge

Expanded Security Maintenance for Applications is not enabled.

54 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


*** System restart required ***
Last login: Tue Sep 12 12:31:18 2023 from 172.19.124.160
ubuntu@cs5060-25:~$ lxc exec bob1 -- /bin/bash
root@bob1:~# python3 TCPPingerClient.py
Enter the number of pings you want to send: 240
Ping 1: Request timed out
Response from server: PING 1 1694523164.9758532 | RTT: 0.9873 ms
Response from server: PING 2 1694523164.9769137 | RTT: 0.7463 ms
Ping 3: Request timed out
Response from server: PING 3 1694523164.977707 | RTT: 668.6027 ms
Ping 4: Request timed out
Ping 4: Request timed out
Ping 4: Request timed out
Response from server: PING 4 1694523166.6476314 | RTT: 132.3621 ms
Response from server: PING 4 1694523167.6489327PING 4 1694523168.6502407PING 4 169
4523169.6514573PING 5 1694523169.783901 | RTT: 208.0309 ms
Response from server: PING 6 1694523169.9936545 | RTT: 0.2604 ms
Response from server: PING 7 1694523169.9953141 | RTT: 204.4897 ms
Ping 8: Request timed out
Response from server: PING 8 1694523170.2013884 | RTT: 84.8944 ms
Response from server: PING 8 1694523171.2027678PING 9 1694523171.287767 | RTT: 120
.1940 ms
Ping 10: Request timed out
Ping 10: Request timed out
Response from server: PING 10 1694523173.4109962 | RTT: 340.6456 ms
Response from server: PING 11 1694523173.7517486 | RTT: 0.9604 ms
Response from server: PING 12 1694523173.7527578 | RTT: 0.9038 ms
Ping 13: Request timed out
Response from server: PING 13 1694523173.75372 | RTT: 532.5866 ms
Response from server: PING 13 1694523174.7550323PING 14 1694523175.2876983 | RTT:
416.1780 ms
```

### Here we can see two connections served concurrently



```
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Connection with ('172.31.0.3', 36198) closed
Connected to ('172.31.0.3', 48586)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Connection with ('172.31.0.3', 48586) closed
Connected to ('172.31.0.3', 37370)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Connected to ('172.31.0.3', 45170)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
Packet lost (no response)
```

```
694523169.9936545 | RTT: 0.2604 ms
694523169.9953141 | RTT: 204.4897 ms

694523170.2013884 | RTT: 84.8944 ms
694523171.2027678PING 9 1694523171.287767 | RTT: 120



 1694523173.4109962 | RTT: 340.6456 ms
 1694523173.7517486 | RTT: 0.9604 ms
 1694523173.7527578 | RTT: 0.9038 ms

 1694523173.75372 | RTT: 532.5866 ms
 1694523174.7550323PING 14 1694523175.2876983 | RTT:
```

## ANTI-PLAGIARISM Statement

I certify that this assignment/report is my own work, based on my personal study and/or research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, packages, datasets, reports, lecture notes, and any other
kind of document, electronic or personal communication. I also certify that this assignment/report has not previously been submitted for assessment/project in any other course lab, except where specific permission has been granted from all course instructors involved, or at any other time in this course, and that I have not copied in part or whole or otherwise plagiarised the work of other students and/or persons. Additionally, I acknowledge that I may have used AI tools, such as language models (e.g., ChatGPT, Bard), for assistance in generating and refining my assignment, and I have made all reasonable efforts
to ensure that such usage complies with the academic integrity policies set for the course. I pledge to uphold the principles of honesty and responsibility at CSE@IITH. In addition, I understand my responsibility to report honour violations by other students if I become aware
of it.

Name: Arjit Gupta
Roll No: cs23mtech12001
Date: 12/09/2023
Signature: AG