

# CSE 232 Section B: Computer Networks: Programming Assignment 1: UDP Pinger Lab

Ritika Thakur (2022408)

Swarnima Prasad (2022525)

## 1 About UDP Pinger

A UDP Pinger is a simple application that sends and receives UDP packets to check if a server is reachable and measure the Round Trip Time (RTT). Key Features of UDP:

1. **Connectionless protocol:** UDP does not establish a connection before sending data which makes it fast but unreliable.
2. **Test server responsiveness:** UDP Pinger Client sends a series of messages with timestamps and sequences to the server to test its responsiveness.
3. **Simulate Packet loss:** UDP is unreliable and packet loss may occur depending on the network connection. UDP Pinger simulates such packet losses but randomly selecting which packets to respond to.
4. **Timeout:** Since UDP is unreliable, client waits for response for a specified time before considering the packet lost.
5. **Round-Trip Time (RTT):** The client measures the time it takes for the message to go to the server and return (RTT) providing an estimate of network latency.
6. **Packet Loss Detection:** Packet loss can be detected based on missed responses using:
  - (a) RTT for each packet
  - (b) Packet loss percentage
  - (c) Minimum, maximum and average RTT for all pings

## 2 Part 1: Implement Client code according to given Server code to implement UDP Pinger

### 2.1 Server Code

```
1 serverSocket = socket(AF_INET, SOCK_DGRAM)
2 serverSocket.bind(('', 12000))
3 while True:
4     rand = random.randint(0, 10)
5     message, address = serverSocket.recvfrom(1024)
6     message = message.upper()
7     if rand < 4:
8         continue
9     serverSocket.sendto(message, address)
```

The server code provided to us sits in an infinite loop listening for incoming UDP packets. When a packet comes in and if a randomized integer is greater than or equal to 4, the server simply capitalizes the encapsulated data and sends it back to the client.

## 2.2 Client Code

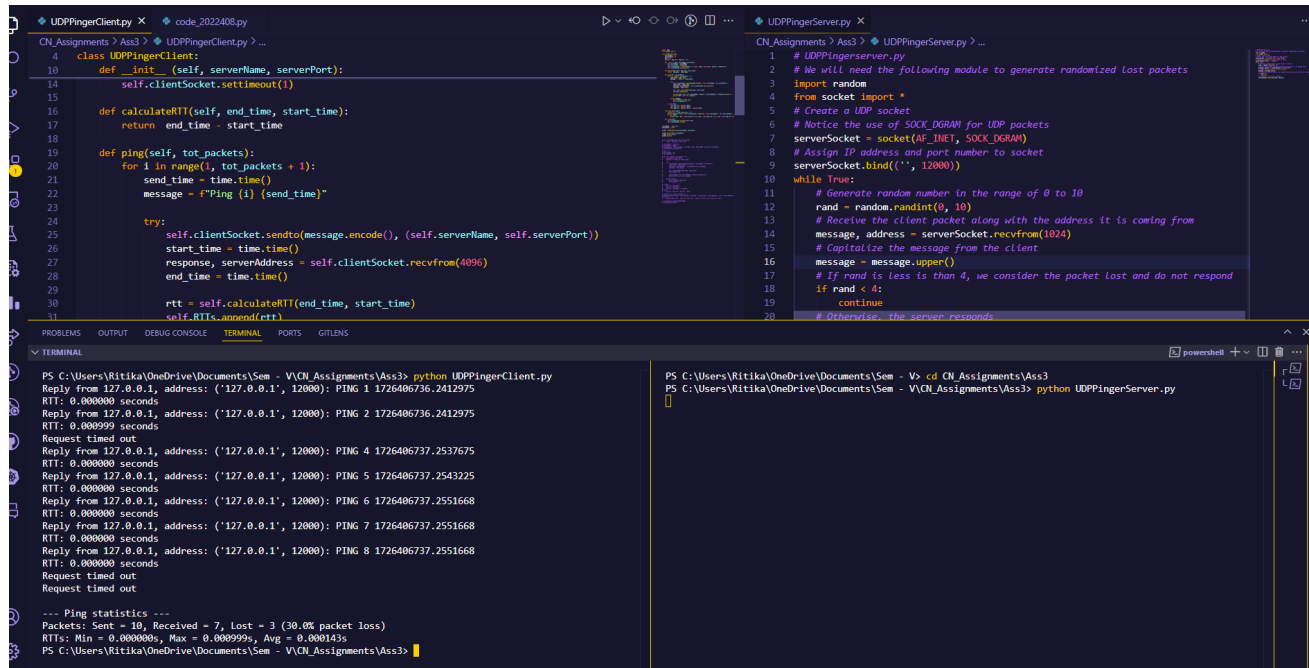
```
1 class UDPPingerClient:
2     tot_packets = 10
3     lost_packets = 0
4     RTTs = []
5     min_rtt = max_rtt = avg_rtt = None
6
7     def __init__(self, serverName, serverPort):
8         self.serverName = serverName
9         self.serverPort = serverPort
10        self.clientSocket = socket(family = AF_INET, type = SOCK_DGRAM, proto=0,
11                                   fileno=None)
12        self.clientSocket.settimeout(1)
13
14    def calculateRTT(self, end_time, start_time):
15        return end_time - start_time
16
17    def ping(self, tot_packets):
18        for i in range(1, tot_packets + 1):
19            send_time = time.time()
20            message = f"Ping {i} {send_time}"
21            try:
22                self.clientSocket.sendto(message.encode(), (self.serverName, self.
23                                                            serverPort))
24                start_time = time.time()
25                response, serverAddress = self.clientSocket.recvfrom(1024)
26                end_time = time.time()
27                rtt = self.calculateRTT(end_time, start_time)
28                self.RTTs.append(rtt)
29                print(f"Reply from {self.serverName}: {response.decode()}")
30                print(f"RTT: {rtt:.6f} seconds")
31            except timeout:
32                print("Request timed out")
33                self.lost_packets += 1
34
35        if self.RTTs:
36            self.min_rtt = min(self.RTTs)
37            self.max_rtt = max(self.RTTs)
38            self.avg_rtt = sum(self.RTTs) / len(self.RTTs)
39
40    def print_stats(self):
41        print(f"\n--- Ping statistics ---")
42        print(f"Packets: Sent = {self.tot_packets}, Received = {self.tot_packets -
43            self.lost_packets}, Lost = {self.lost_packets} ({(self.lost_packets / self
44            .tot_packets) * 100}% packet loss)")
45        if self.RTTs:
46            print(f"RTTs: Min = {self.min_rtt:.6f}s, Max = {self.max_rtt:.6f}s, Avg =
47                {self.avg_rtt:.6f}s")
48
49    def close(self):
50        self.clientSocket.shutdown(SHUT_RDWR)
51        self.clientSocket.close()
```

The UDPPingerClient.py implements a UDP Pinger Client that sends a series of ping messages to a UDP server and measures the round-trip time (RTT) for each message. It sends 10 ping requests, calculates the RTT for each response, and handles packet loss by counting requests that time out after 1 second. After all pings are sent, it prints the number of packets sent, received, lost, and the packet loss percentage, along with the minimum, maximum, and average RTTs. Finally, it closes the client socket.

## 2.3 Outputs

### 2.3.1 Local Host

Below are some outputs when UDPPingerClient.py and UDPPingerServer.py are run on the same machine on different terminals:



```
class UDPPingerClient:
    def __init__(self, serverName, serverPort):
        self.clientSocket.settimeout(1)

    def calculateRTT(self, end_time, start_time):
        return end_time - start_time

    def ping(self, tot_packets):
        for i in range(1, tot_packets + 1):
            send_time = time.time()
            message = f"ping {i} (send_time)"

            try:
                self.clientSocket.sendto(message.encode(), (self.serverName, self.serverPort))
                start_time = time.time()
                response, serverAddress = self.clientSocket.recvfrom(4096)
                end_time = time.time()
                rtt = self.calculateRTT(end_time, start_time)
                self.RTTs.append(rtt)

# 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 # Create a UDP socket
6 # Notice the use of SOCK_DGRAM for UDP packets
7 serverSocket = socket(AF_INET, SOCK_DGRAM)
8 # Assign IP address and port number to socket
9 serverSocket.bind(('', 12000))
10 while True:
11     # Generate random number in the range of 0 to 10
12     rand = random.randint(0, 10)
13     # Receive the client packet along with the address it is coming from
14     message, address = serverSocket.recvfrom(1024)
15     # Capitalize the message from the client
16     message = message.upper()
17     # If rand is less than 4, we consider the packet lost and do not respond
18     if rand < 4:
19         continue
20     # Otherwise, the server responds
```

```
PS C:\Users\Ritika\OneDrive\Documents\Sem - V\CN_Assignments\Ass3> python UDPPingerClient.py
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 1 1726406736.2412975
RTT: 0.000000 seconds
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 2 1726406736.2412975
RTT: 0.000000 seconds
Request timed out
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 4 1726406737.2537675
RTT: 0.000000 seconds
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 5 1726406737.2543225
RTT: 0.000000 seconds
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 6 1726406737.2551668
RTT: 0.000000 seconds
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 7 1726406737.2551668
RTT: 0.000000 seconds
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 8 1726406737.2551668
RTT: 0.000000 seconds
Request timed out
Request timed out

--- Ping statistics ---
Packets: Sent = 10, Received = 7, Lost = 3 (30.0% packet loss)
RTTs: Min = 0.000000s, Max = 0.000000s, Avg = 0.000143s
PS C:\Users\Ritika\OneDrive\Documents\Sem - V\CN_Assignments\Ass3>
```

Figure 1: UDPPingerClient simulating 30% loss

```
--- Ping statistics ---
Packets: Sent = 100, Received = 54, Lost = 46 (46.0% packet loss)
RTTs: Min = 0.000000s, Max = 0.001051s, Avg = 0.000216s
PS C:\Users\Ritika\OneDrive\Documents\Sem - V\CN_Assignments\Ass3>
```

Figure 2: UDPPingerClient for 100 packets

```
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments> cd Ass3
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3> python UDPPingerClient.py
Request timed out
Request timed out
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 3 1726417310.4518144
RTT: 0.000000 seconds
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 4 1726417310.4528388
RTT: 0.000000 seconds
Request timed out
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 6 1726417311.4668863
RTT: 0.000000 seconds
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 7 1726417311.4668863
RTT: 0.000000 seconds
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 8 1726417311.4668863
RTT: 0.000000 seconds
Request timed out
Reply from 127.0.0.1, address: ('127.0.0.1', 12000): PING 10 1726417312.467927
RTT: 0.000000 seconds

--- Ping statistics ---
Packets: Sent = 10, Received = 6, Lost = 4 (40.0% packet loss)
RTTs: Min = 0.000000s, Max = 0.000000s, Avg = 0.000000s
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3>
```

```
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments> cd Ass3
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3> python UDPPingerServer.py
>>
```

Figure 3: UDPPingerClient for 100 packets

### 2.3.2 Different Machines

Below are some outputs when UDPPingerClient.py and UDPPingerServer.py are run on different machines:

```
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3> python UDPPingerClient.py
Reply from 192.168.32.232: PING 1 1726776829.0851982
RTT: 0.006491 seconds
Reply from 192.168.32.232: PING 2 1726776829.092689
RTT: 0.010354 seconds
Reply from 192.168.32.232: PING 3 1726776829.10406
RTT: 0.006994 seconds
Reply from 192.168.32.232: PING 4 1726776829.1110542
RTT: 0.009669 seconds
Reply from 192.168.32.232: PING 5 1726776829.1207235
RTT: 0.007297 seconds
Request timed out
Reply from 192.168.32.232: PING 7 1726776830.1329608
RTT: 0.006393 seconds
Request timed out
Request timed out
Reply from 192.168.32.232: PING 10 1726776832.155001
RTT: 0.008133 seconds

--- Ping statistics ---
Packets: Sent = 10, Received = 7, Lost = 3 (30.0% packet loss)
RTTs: Min = 0.006393s, Max = 0.010354s, Avg = 0.007905s
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3>
```

Figure 4: UDPPingerClient simulating 30% loss across different machines

## 3 About UDP Heartbeat

A UDP heartbeat is a lightweight message sent at regular intervals between a client and a server to check if the connection is still alive and responsive. In networking, it's often used to monitor the status of the server, ensuring that it's up and running, or to keep a connection active without transferring large amounts of data.

UDP heartbeats are typically used when:

1. Speed is critical: For example, in real-time systems like video games, VoIP, or monitoring tools, UDP is preferred because of its low latency.
2. The application can tolerate some packet loss: Losing occasional heartbeats is acceptable as long as it's infrequent, and missing several in a row may signal a problem.

## 4 Part 2: Modify Client and Server code to implement UDP Heartbeat

### 4.1 Server Code

```
1 serverSocket = socket(AF_INET, SOCK_DGRAM)
2 serverSocket.bind(('', 12000))
3 print("The server is ready to receive heartbeats...")
4 while True:
5     rand = random.randint(0, 10)
6     message, clientAddress = serverSocket.recvfrom(1024)
7     if rand < 4:
8         continue
9     decoded_message = message.decode()
10    print(f"Received message: {decoded_message}")
```

```

11     _, sequence_number, sent_time = decoded_message.split()
12     receive_time = time.time()
13     time_diff = receive_time - float(sent_time)
14     response_message = f"{sequence_number} {time_diff:.6f}"
15     serverSocket.sendto(response_message.encode(), clientAddress)

```

## 4.2 Client Code

```

1  class UDPPingerClient:
2      tot_packets = 1000
3      missed_heartbeats = 0
4      consecutive_misses = 0
5      total_sent=0
6
7      def __init__(self, serverName, serverPort):
8          self.serverName = serverName
9          self.serverPort = serverPort
10         self.clientSocket = socket(family = AF_INET, type = SOCK_DGRAM)
11         self.clientSocket.settimeout(1)
12
13     def ping(self, tot_packets):
14         for i in range(1, tot_packets + 1):
15             send_time = time.time()
16             self.total_sent+=1
17             message = f"Ping {i} {send_time}"
18             try:
19                 self.clientSocket.sendto(message.encode(), (self.serverName, self.
20                     serverPort))
21                 response, serverAddress = self.clientSocket.recvfrom(1024)
22                 recv_time = time.time()
23                 response_message = response.decode()
24                 sequence_number, time_diff = response_message.split()
25                 time_diff = float(time_diff)
26                 print(f"Received heartbeat response {sequence_number}: Time difference
27                     = {time_diff:.6f} seconds")
28                 self.consecutive_misses = 0
29             except timeout:
30                 self.consecutive_misses += 1
31                 self.missed_heartbeats += 1
32                 print(f"Heartbeat {i}: Request timed out")
33                 if self.consecutive_misses == 3:
34                     print("Server is down! 3 consecutive heartbeat responses were missed."
35                         )
36                     break
37
38     def print_stats(self):
39         print(f"\n--- Heartbeat statistics ---")
40         print(f"Packets: Sent = {self.total_sent}, Received = {self.total_sent - self.
41             missed_heartbeats}, Lost = {self.missed_heartbeats} ({(self.
42                 missed_heartbeats / self.total_sent) * 100}% packet loss)")
43
44     def close(self):
45         self.clientSocket.shutdown(SHUT_RDWR)
46         self.clientSocket.close()

```

## 4.3 Outputs

### 4.3.1 Local Host

Below are some outputs when UDPHeartbeatClient.py and UDPHeartbeatServer.py are run on the same machine on different terminals:

```
PS C:\Users\Ritika\OneDrive\Documents\Sem - V\CN_Assignments\Ass3> python UDPHeartbeatClient.py
Received heartbeat response 1: Time difference = 0.003571 seconds
Received heartbeat response 2: Time difference = 0.001068 seconds
Received heartbeat response 3: Time difference = 0.000000 seconds
Received heartbeat response 4: Time difference = 0.001006 seconds
Heartbeat 5: Request timed out
Heartbeat 6: Request timed out
Received heartbeat response 7: Time difference = 0.001003 seconds
Received heartbeat response 8: Time difference = 0.000000 seconds
Received heartbeat response 9: Time difference = 0.000000 seconds
Received heartbeat response 10: Time difference = 0.000000 seconds

--- Heartbeat statistics ---
Packets: Sent = 10, Received = 8, Lost = 2 (20.0% packet loss)
PS C:\Users\Ritika\OneDrive\Documents\Sem - V\CN_Assignments\Ass3>
```

Figure 5: UDP Heartbeat with 10 packets

```

PS C:\Users\Ritika\OneDrive\Documents\Sem - V\CN_Assignments\Ass3> python UDPHeartbeatClient.py
Received heartbeat response 1: Time difference = 0.000000 seconds
Heartbeat 2: Request timed out
Received heartbeat response 3: Time difference = 0.000000 seconds
Received heartbeat response 4: Time difference = 0.000000 seconds
Received heartbeat response 5: Time difference = 0.000000 seconds
Heartbeat 6: Request timed out
Received heartbeat response 7: Time difference = 0.000925 seconds
Received heartbeat response 8: Time difference = 0.000000 seconds
Received heartbeat response 9: Time difference = 0.000000 seconds
Heartbeat 10: Request timed out
Received heartbeat response 11: Time difference = 0.000947 seconds
Heartbeat 12: Request timed out
Received heartbeat response 13: Time difference = 0.001048 seconds
Heartbeat 14: Request timed out
Received heartbeat response 15: Time difference = 0.000742 seconds
Received heartbeat response 16: Time difference = 0.000512 seconds
Heartbeat 17: Request timed out
Heartbeat 18: Request timed out
Heartbeat 19: Request timed out
Server is down! 3 consecutive heartbeat responses were missed.

--- Heartbeat statistics ---
Packets: Sent = 19, Received = 11, Lost = 8 (42.10526315789473% packet loss)
PS C:\Users\Ritika\OneDrive\Documents\Sem - V\CN_Assignments\Ass3>

PS C:\Users\Ritika\OneDrive\Documents\Sem - V> cd CN_Assignments\Ass3
PS C:\Users\Ritika\OneDrive\Documents\Sem - V\CN_Assignments\Ass3> python UDPHeartbeatServer.py
The server is ready to receive heartbeats...
Received message: Ping 1 1726414389.9965243
Received message: Ping 3 1726414391.0103593
Received message: Ping 4 1726414391.0112293
Received message: Ping 5 1726414391.0112293
Received message: Ping 7 1726414392.0209162
Received message: Ping 8 1726414392.021841
Received message: Ping 9 1726414392.0228066
Received message: Ping 11 1726414393.028079
Received message: Ping 13 1726414394.0370631
Received message: Ping 15 1726414395.045213
Received message: Ping 16 1726414395.0459552
```

Figure 6: UDP Heartbeat with 100 packets

```

Received heartbeat response 2: Time difference = 0.000000 seconds
Heartbeat 3: Request timed out
Heartbeat 4: Request timed out
Received heartbeat response 5: Time difference = 0.000000 seconds
Received heartbeat response 6: Time difference = 0.001032 seconds
Received heartbeat response 7: Time difference = 0.000000 seconds
Received heartbeat response 8: Time difference = 0.000585 seconds
Heartbeat 9: Request timed out
Heartbeat 10: Request timed out
Received heartbeat response 11: Time difference = 0.000000 seconds
Received heartbeat response 12: Time difference = 0.000000 seconds
Heartbeat 13: Request timed out
Heartbeat 14: Request timed out
Received heartbeat response 15: Time difference = 0.000000 seconds
Heartbeat 16: Request timed out
Received heartbeat response 17: Time difference = 0.000000 seconds
Heartbeat 18: Request timed out
Received heartbeat response 19: Time difference = 0.000000 seconds
Received heartbeat response 20: Time difference = 0.000000 seconds
Received heartbeat response 21: Time difference = 0.000000 seconds
Heartbeat 22: Request timed out
Heartbeat 23: Request timed out
Received heartbeat response 24: Time difference = 0.000477 seconds
Received heartbeat response 25: Time difference = 0.000000 seconds
Received heartbeat response 26: Time difference = 0.000000 seconds
Received heartbeat response 27: Time difference = 0.000000 seconds
Received heartbeat response 28: Time difference = 0.000000 seconds
Received heartbeat response 29: Time difference = 0.000000 seconds
Received heartbeat response 30: Time difference = 0.000000 seconds
Heartbeat 31: Request timed out
Received heartbeat response 32: Time difference = 0.000000 seconds
Heartbeat 33: Request timed out
Received heartbeat response 34: Time difference = 0.000000 seconds
Heartbeat 35: Request timed out
Received heartbeat response 36: Time difference = 0.000540 seconds
Heartbeat 37: Request timed out
Received heartbeat response 38: Time difference = 0.001002 seconds
Received heartbeat response 39: Time difference = 0.000000 seconds
Heartbeat 40: Request timed out
Heartbeat 41: Request timed out
Heartbeat 42: Request timed out
Server is down! 3 consecutive heartbeat responses were missed.

--- Heartbeat statistics ---
Packets: Sent = 42, Received = 24, Lost = 18 (42.857142857142854% packet loss)

```

```

PS C:\Users\swarnima.prasad\OneDrive\Desktop\OL Ass\OL Assignments> cd Ass3
PS C:\Users\swarnima.prasad\OneDrive\Desktop\OL Ass\OL Assignments\Ass3> python Q2_server.py
The server is ready to receive heartbeats...
Received message: Ping 2 1726417459.3829966
Received message: Ping 5 1726417461.3862047
Received message: Ping 6 1726417461.3862047
Received message: Ping 7 1726417461.3872368
Received message: Ping 8 1726417461.3872368
Received message: Ping 11 1726417463.3920503
Received message: Ping 12 1726417463.3920503
Received message: Ping 15 1726417465.4029057
Received message: Ping 17 1726417466.4080868
Received message: Ping 19 1726417467.4121025
Received message: Ping 20 1726417467.4131212
Received message: Ping 21 1726417467.4131212
Received message: Ping 24 1726417469.4351306
Received message: Ping 25 1726417469.4356088
Received message: Ping 26 1726417469.4356088
Received message: Ping 27 1726417469.4356088
Received message: Ping 28 1726417469.4366305
Received message: Ping 29 1726417469.4366305
Received message: Ping 30 1726417469.4366305
Received message: Ping 32 1726417470.4498785
Received message: Ping 34 1726417471.4645836
Received message: Ping 36 1726417472.4650805
Received message: Ping 38 1726417473.4804332
Received message: Ping 39 1726417473.4814355

```

Figure 7: UDP Heartbeat with 1000 packets

#### 4.3.2 Different Machines

Below are some outputs when UDPHeartbeatClient.py and UDPHeartbeatServer.py are run on different machines:

```

Heartbeat 94: Request timed out
Received heartbeat response 95: Time difference = 3.066470 seconds
Received heartbeat response 96: Time difference = 3.068163 seconds
Received heartbeat response 97: Time difference = 3.063143 seconds
Heartbeat 98: Request timed out
Received heartbeat response 99: Time difference = 3.063967 seconds
Heartbeat 100: Request timed out

--- Heartbeat statistics ---
Packets: Sent = 100, Received = 68, Lost = 32 (32.0% packet loss)

```

Figure 8: UDP Heartbeat with 100 packets across different machines not facing 3 consecutive losses



```

PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3> python UDPHeartbeatClient.py
Received heartbeat response 1: Time difference = 3.066142 seconds
Received heartbeat response 2: Time difference = 3.063971 seconds
Heartbeat 3: Request timed out
Received heartbeat response 4: Time difference = 3.063734 seconds
Received heartbeat response 5: Time difference = 3.063859 seconds
Heartbeat 6: Request timed out
Heartbeat 7: Request timed out
Received heartbeat response 8: Time difference = 3.067698 seconds
Received heartbeat response 9: Time difference = 3.064611 seconds
Received heartbeat response 10: Time difference = 3.065667 seconds
Received heartbeat response 11: Time difference = 3.065002 seconds
Heartbeat 12: Request timed out
Received heartbeat response 13: Time difference = 3.068117 seconds
Received heartbeat response 14: Time difference = 3.067029 seconds
Received heartbeat response 15: Time difference = 3.070457 seconds
Received heartbeat response 16: Time difference = 3.065941 seconds
Heartbeat 17: Request timed out
Received heartbeat response 18: Time difference = 3.064579 seconds
Received heartbeat response 19: Time difference = 3.064699 seconds
Heartbeat 20: Request timed out
Heartbeat 21: Request timed out
Received heartbeat response 22: Time difference = 3.069790 seconds
Heartbeat 23: Request timed out
Heartbeat 24: Request timed out
Heartbeat 25: Request timed out
Server is down! 3 consecutive heartbeat responses were missed.

--- Heartbeat statistics ---
Packets: Sent = 25, Received = 15, Lost = 10 (40.0% packet loss)
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3>

```

Figure 9: UDP Heartbeat for 1000 packets

```

serverName = '192.168.32.232'

```

Figure 10: ServerName of the machine on which UDPHeartbeatServer is running

## 5 Observations and Conclusions

For UDP Pinger:

1. In the first output (Figure 1), the UDPPingerClient simulates a 30% packet loss rate. This is reflected by the missing replies for some of the pings, as indicated by the "Request timed out" message. The loss is consistent with the randomness in packet drops by the server, which uses a random number generator to drop packets.



2. The second output (Figure 2) shows a test with 100 packets, demonstrating the ability of the client to handle a larger number of pings. The packet loss percentage does not shoot up by a huge margin, and the RTT statistics are more representative over a larger sample size.
3. Running UDPPingerServer.py and UDPPingerClient.py on different machines does not impact the packet loss and we can still simulate a 30% packet loss.

For UDP Heartbeat:

1. In testing, we sent 1000 packets to simulate a long-term connection and observe behavior over time.
2. Continuously packets are sent until program logic detects 3 consecutive failures, which could happen after any number of packets, depending on the state of the network or server.
3. While timeout is kept for 1 second the server on average gets down within 50 packets are sent.
4. For different machines we notice that with timeout for 1 second we are able to send 100 packets without 3 consecutive misses.

## 6 References

- Python Socket Documentation
- Python Socket Howto
- Socket Programming GitHub Repository
- Socket Programming 2 - Tutorial slides