Computer Network

UDP Socket Programming

Ritika Thakur(2022408) Swarnima Prasad (2022525)

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 conenction.
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

Server Code

```
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:
    rand = random.randint(0, 10)
    message, address = serverSocket.recvfrom(1024)
    # Capitalize the message from the client
    message = message.upper()

    if rand < 4:
        continue
    # Otherwise, the server responds
    serverSocket.sendto(message, address)
```

Client Code

```python
import time
from socket import *
class UDPPingerClient:
    tot_packets = 10
    lost_packets = 0
    RTTs = []
    min_rtt = max_rtt = avg_rtt = None

    def __init__ (self, serverName, serverPort):
        self.serverName = serverName
        self.serverPort = serverPort
        self.clientSocket = socket(family = AF_INET, type = SOCK_DGRAM, proto=0, fileno=None)
        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(1024)
                end_time = time.time()

                rtt = self.calculateRTT(end_time, start_time)
                self.RTTs.append(rtt)

                print(f"Reply from {self.serverName}, address: {serverAddress}: {response.decode()}")
                print(f"RTT: {rtt:.6f} seconds")

            except timeout:
                print("Request timed out")
                self.lost_packets += 1

        if self.RTTs:
            self.min_rtt = min(self.RTTs)
            self.max_rtt = max(self.RTTs)
            self.avg_rtt = sum(self.RTTs) / len(self.RTTs)

    def print_stats(self):
        print(f"\n--- Ping statistics ---")
        print(f"Packets: Sent = {self.tot_packets}, Received = {self.tot_packets - self.lost_packets},
            Lost = {self.lost_packets} ({(self.lost_packets / self.tot_packets) * 100}% packet loss)")
        if self.RTTs:
            print(f"RTTs: Min = {self.min_rtt:.6f}s, Max = {self.max_rtt:.6f}s, Avg = {self.avg_rtt:.6f}s"

    def close(self):
        self.clientSocket.shutdown(SHUT_RDWR)
        self.clientSocket.close()

serverName = '127.0.0.1'
serverPort = 12000

client = UDPPingerClient(serverName, serverPort)

client.ping(client.tot_packets)
client.print_stats()
client.close()
```

calculating RTT for a ping and adding it to the list for finding overall statistics

Output

```
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: 0.000999 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 = 0.000999s, Avg = 0.000143s
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 UDPPingerServer.py
```

```
--- 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>
```

Output

```
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 1726417310.4528308
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
>>
```

Output across different machine

```
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3> python UDPPingerClient.py
Reply from 192.168.32.232: PING 1 1726776725.7596622
RTT: 0.031438 seconds
Request timed out
Reply from 192.168.32.232: PING 3 1726776726.806588
RTT: 0.002959 seconds
Reply from 192.168.32.232: PING 4 1726776726.8095474
RTT: 0.012480 seconds
Reply from 192.168.32.232: PING 5 1726776726.8220272
RTT: 0.007075 seconds
Reply from 192.168.32.232: PING 6 1726776726.8291018
RTT: 0.012595 seconds
Reply from 192.168.32.232: PING 7 1726776726.8416972
RTT: 0.007022 seconds
Request timed out
Request timed out
Reply from 192.168.32.232: PING 10 1726776728.8525276
RTT: 0.005920 seconds

--- Ping statistics ---
Packets: Sent = 10, Received = 7, Lost = 3 (30.0% packet loss)
RTTs: Min = 0.002959s, Max = 0.031438s, Avg = 0.011356s
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_Assignments\Ass3>
```

Change made in server code

```
serverName = '192.168.32.232'
```

UDP Heartbeat

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:

• Speed is critical: For example, in real-time systems like video games, VoIP, or monitoring tools, UDP is preferred because of its low latency.
• 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.

Server Code

```python
import random
import time
from socket import *

serverSocket = socket(AF_INET, SOCK_DGRAM)
serverSocket.bind(('', 12000))
print("The server is ready to receive heartbeats...")

while True:
    rand = random.randint(0, 10)
    message, clientAddress = serverSocket.recvfrom(1024)

    if rand < 4:
        continue

    decoded_message = message.decode()
    print(f"Received message: {decoded_message}")

    _, sequence_number, sent_time = decoded_message.split()

    receive_time = time.time()
    time_diff = receive_time - float(sent_time)

    response_message = f"{sequence_number} {time_diff:.6f}"
    serverSocket.sendto(response_message.encode(), clientAddress)
```

Client Code

```python
import time
from socket import *

class UDPPingerClient:
    tot_packets = 1000
    missed_heartbeats = 0
    consecutive_misses = 0
    total_sent=0

    def __init__ (self, serverName, serverPort):
        self.serverName = serverName
        self.serverPort = serverPort
        self.clientSocket = socket(family = AF_INET, type = SOCK_DGRAM)
        self.clientSocket.settimeout(1)

    def ping(self, tot_packets):
        for i in range(1, tot_packets + 1):
            send_time = time.time()
            self.total_sent+=1
            message = f"Ping {i} {send_time}"
            try:
                self.clientSocket.sendto(message.encode(), (self.serverName, self.serverPort))

                response, serverAddress = self.clientSocket.recvfrom(1024)
                recv_time = time.time()
                response_message = response.decode()
                sequence_number, time_diff = response_message.split()
                time_diff = float(time_diff)

                print(f"Received heartbeat response {sequence_number}: Time difference = {time_diff:.6f} seconds")
                self.consecutive_misses = 0
            except timeout:
                self.consecutive_misses += 1
                self.missed_heartbeats += 1
                print(f"Heartbeat {i}: Request timed out")
            if self.consecutive_misses == 3:
                print("Server is down! 3 consecutive heartbeat responses were missed.")
                break

    def print_stats(self):
        print(f"\n--- Heartbeat statistics ---")
        print(f"Packets: Sent = {self.total_sent}, Received = {self.total_sent - self.missed_heartbeats},
            Lost = {self.missed_heartbeats} ({(self.missed_heartbeats / self.total_sent) * 100}% packet loss)")

    def close(self):
        self.clientSocket.shutdown(SHUT_RDWR)
        self.clientSocket.close()

serverName = '127.0.0.1'
serverPort = 12000

client = UDPPingerClient(serverName, serverPort)
client.ping(client.tot_packets)
client.print_stats()
client.close()
```

Output for 1000 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\CN_ass\CN_Assignments> cd Ass3
PS C:\Users\swarnima prasad\OneDrive\Desktop\CN_ass\CN_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 1726417469.435608
Received message: Ping 26 1726417469.435608
Received message: Ping 27 1726417469.435608
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 1726417472.465085
Received message: Ping 38 1726417473.4804332
Received message: Ping 39 1726417473.4814355
```

Output for 100 packets

TERMINAL

```
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>
```

powershell

```
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
```

Output for 10 packets

```
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>
```

Output across different machine

```
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
```

Change made in server code

```
serverName = '192.168.32.232'
```

```
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)
```

100 packets were also sent without having 3 consecutive packet loss

Observations and Conclusions

UDP Pinger:

- In the first output 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.
- The second output 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
- . Running UDPPingerServer.py and UDPPingerClient.py on different machines does not impact the packet loss and we can still simulate a 30% packet loss.

UDP Heartbeat:

- In testing, we sent 1000 packets to simulate a long-term connection and observe behavior over time.
- 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.
- While timeout is kept for 1 second the server on average gets down within 50 packets are sent
- For different machines we notice that with timeout for 1 second we are able to send 100 packets without 3

References

- https://docs.python.org/3/search.html?q=socket : Python Socket Documentation
- https://docs.python.org/3/howto/sockets.html : Python Socket Howto
- https://github.com/AkankshaSingal8/socket_programming/tree/main : Socket Programming GitHub Repository
- Socket Programming 2 - Tutorial slides