

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

Harsh Rajput (2022201)
Aryan Singla (2022112)

Part 1: **UDP Ping Client System**

This system simulates the functionality of the common network utility tool *Ping* using the User Datagram Protocol (UDP). The primary purpose is to send a series of "ping" messages to a server, measure the round-trip time (RTT) for each response, and calculate statistics such as minimum RTT, maximum RTT, average RTT, and packet loss rate. The client sends 10 ping requests and waits for a response from the server.

Server code

UDPPingerServer.py X

▶ ▾ □ ...

UDPPingerServer.py > ...

```
1  # We will need the following module to generate randomized lost packets
2  import time
3  import random
4  from socket import *
5
6  serverName = "localhost"
7  serverPort = 12000
8  serverAddress = (serverName, serverPort)
9  # Create a UDP socket
10 # Notice the use of SOCK_DGRAM for UDP packets
11 serverSocket = socket(AF_INET, SOCK_DGRAM)
12 # Assign IP address and port number to socket
13 serverSocket.bind(('', serverPort))
14 while True:
15     # Generate random number in the range of 0 to 10
16     rand = random.randint(0, 10)
17     # Receive the client packet along with the address it is coming from
18     message, clientAddress = serverSocket.recvfrom(1024)
19     # Capitalize the message from the client
20     modifiedMessage = message.decode('utf-8').upper().encode('utf-8')
21     # If rand is less is than 4, we consider the packet lost and do not respond
22     if rand < 4:
23         continue
24     # Otherwise, the server responds
25     serverSocket.sendto(modifiedMessage, clientAddress)
```

Client code

UDPPingerClient.py

UDPPingerClient.py > ...

```
1  import time
2  from socket import *
3
4  serverName = "localhost"
5  serverPort = 12000
6  serverAddress = (serverName, serverPort)
7
8  # Define the timeout value for the client to wait for a response from the server
9  TIME_OUT = -1
10 serverResponses = []
11
12 # Create a UDP clientSocket
13 clientSocket = socket(AF_INET, SOCK_DGRAM) # Notice the use of SOCK_DGRAM for UDP packets
14 # Set the timeout of to 1 second
15 clientSocket.settimeout(1)
16 print(f"PING {serverName} 10 times")
17 total_ping = 10
18 for sequence_number in range(1, total_ping+1):
19     time_sent = time.time()
20     message = f"Ping {sequence_number} {time.ctime(time_sent)}".encode('utf-8')
21     # Send the UDP packet with the ping message
22     clientSocket.sendto(message, serverAddress)
23     # Wait for the server to respond
24     try:
25         modifiedMessage, serverAddress = clientSocket.recvfrom(1024)
26         modifiedMessage = modifiedMessage.decode('utf-8')
27         time_received = time.time()
28         rtt = (time_received - time_sent)*1000
29         print(f"udp_seq={sequence_number} message=\"{modifiedMessage}\" rtt={rtt:.3f}ms")
30         serverResponses.append(rtt)
31     except timeout:
32         print(f"udp_seq={sequence_number} Request timed out ")
33         serverResponses.append(TIME_OUT)
34
35 clientSocket.close()
```

Code to calculate metrics

UDPPingerClient.py •



UDPPingerClient.py > ...

```
37 # Calculate the statistics
38 # Report the minimum RTT, maximum RTT, average RTT, and packet loss rate at the end of all pings from the client
39 total_time = 0
40 packet_loss = 0
41 rtt_min = float('inf')
42 rtt_max = float('-inf')
43 for rtt in serverResponses:
44     if rtt == TIME_OUT:
45         packet_loss += 1
46     else:
47         total_time += rtt
48         rtt_min = min(rtt_min, rtt)
49         rtt_max = max(rtt_max, rtt)
50
51 packet_transmitted = len(serverResponses)
52 packet_received = packet_transmitted - packet_loss
53 packet_loss_rate = (packet_loss / packet_transmitted) * 100
54 if packet_received == 0:
55     rtt_avg = 0
56 else:
57     rtt_avg = total_time / packet_received
58
59 print(f"\n--- {serverName} ping statistics ---")
60 print(f"{packet_transmitted} packets transmitted, {packet_received} received, {packet_loss_rate:.2f}% packet loss")
61 print(f"rtt min/avg/max = {rtt_min:.3f}/{rtt_avg:.3f}/{rtt_max:.3f} ms")
62
63
```

Client end output

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE COMMENTS

 Code     ... < X

```
PS F:\CN\A1> python "f:\CN\A1\UDPpingerClient.py"
PING localhost 10 times
udp_seq=1 message="PING 1 FRI SEP 20 20:17:39 2024" rtt=2.054ms
udp_seq=2 Request timed out
udp_seq=3 Request timed out
udp_seq=4 Request timed out
udp_seq=5 message="PING 5 FRI SEP 20 20:17:42 2024" rtt=0.000ms
udp_seq=6 message="PING 6 FRI SEP 20 20:17:42 2024" rtt=1.005ms
udp_seq=7 message="PING 7 FRI SEP 20 20:17:42 2024" rtt=0.000ms
udp_seq=8 Request timed out
udp_seq=9 Request timed out
udp_seq=10 message="PING 10 FRI SEP 20 20:17:44 2024" rtt=0.000ms
```

```
--- localhost ping statistics ---
10 packets transmitted, 5 received, 50.0% packet loss
rtt min/avg/max = 0.000/0.612/2.054 ms
```

```
PS F:\CN\A1> python "f:\CN\A1\UDPpingerClient.py"
PING localhost 10 times
udp_seq=1 Request timed out
udp_seq=2 Request timed out
udp_seq=3 Request timed out
udp_seq=4 message="PING 4 FRI SEP 20 20:18:01 2024" rtt=0.000ms
udp_seq=5 Request timed out
udp_seq=6 Request timed out
udp_seq=7 Request timed out
udp_seq=8 message="PING 8 FRI SEP 20 20:18:04 2024" rtt=0.000ms
udp_seq=9 message="PING 9 FRI SEP 20 20:18:04 2024" rtt=0.000ms
udp_seq=10 message="PING 10 FRI SEP 20 20:18:04 2024" rtt=2.091ms
```

```
--- localhost ping statistics ---
10 packets transmitted, 4 received, 60.0% packet loss
rtt min/avg/max = 0.000/0.523/2.091 ms
```

```
PS F:\CN\A1> 
```

Part 1: **UDP Heartbeat Client and Server System**

Another similar application to the UDP Ping would be the UDP Heartbeat. The Heartbeat can be used to check if an application server is up and running or not. The client sends a sequence number and current timestamp in the UDP packet to the server, which is listening for the Heartbeat (i.e., the UDP packets) from the client. Upon receiving the packets, the server calculates the time difference and reports it back. If the server response to client Heartbeat packets is missing for some specified number of times (say 3), the client can assume that the server application has stopped. Here again simulate a UDP packet loss at rate 30%.

Server code

```
UDPHearbeatServer.py X
UDPHearbeatServer.py > ...
1  # We will need the following module to generate randomized lost packets
2  import time
3  import random
4  from socket import *
5
6  serverName = "localhost"
7  serverPort = 12000
8  serverAddress = (serverName, serverPort)
9  # Create a UDP socket
10 # Notice the use of SOCK_DGRAM for UDP packets
11 serverSocket = socket(AF_INET, SOCK_DGRAM)
12 # Assign IP address and port number to socket
13 serverSocket.bind('', serverPort)
14 while True:
15     # Generate random number in the range of 0 to 10
16     rand = random.randint(0, 10)
17     # Receive the client packet along with the address it is coming from
18     message, clientAddress = serverSocket.recvfrom(1024)
19     # time received
20     time_received = time.time()
21     # decode the message and get the sequence number and time sent
22     message = message.decode('utf-8').split()
23     sequence_number = int(message[1])
24     # convert the time sent form ctime to float
25     time_sent = float(message[2])
26     # delay in ms
27     time_difference = (time_received - time_sent)*1000
28     modifiedMessage = f"Ping {sequence_number} time difference={time_difference:.3f}ms".encode('utf-8')
29     # If rand is less is than 4, we consider the packet lost and do not respond
30     if rand < 4:
31         continue
32     # Otherwise, the server responds
33     serverSocket.sendto(modifiedMessage, clientAddress)
```


Client code

```
UDPHeartbeatClient.py •
UDPHeartbeatClient.py > ...
1  import time
2  from socket import *
3
4  serverName = "localhost"
5  serverPort = 12000
6  serverAddress = (serverName, serverPort)
7
8  # Define the timeout value for the client to wait for a response from the server
9  TIME_OUT = -1
10 serverResponses = []
11
12 # Create a UDP clientSocket
13 clientSocket = socket(AF_INET, SOCK_DGRAM) # Notice the use of SOCK_DGRAM for UDP packets
14 # Set the timeout of to 1 second
15 clientSocket.settimeout(1)
16 Heartbeat = 3
17 print(f"PING {serverName} 10 times")
18 total_ping = 10
19 for sequence_number in range(1, total_ping+1):
20     time_sent = time.time()
21     message = f"Ping {sequence_number} {time_sent}".encode('utf-8')
22     # Send the UDP packet with the ping message
23     clientSocket.sendto(message, serverAddress)
24     # Wait for the server to respond
25     try:
26         modifiedMessage, serverAddress = clientSocket.recvfrom(1024)
27         modifiedMessage = modifiedMessage.decode('utf-8')
28         time_received = time.time()
29         rtt = (time_received - time_sent)*1000
30         print(f"udp_seq={sequence_number} message=\"{modifiedMessage}\" rtt={rtt:.3f}ms")
31         serverResponses.append(rtt)
32         # reset the heartbeat
33         Heartbeat = 3
34     except timeout:
35         print(f"udp_seq={sequence_number} Request timed out ")
36         serverResponses.append(TIME_OUT)
37         # decrement the heartbeat and check if the server application has stopped
38         Heartbeat -= 1
39         if (Heartbeat == 0):
40             print("Server is down")
41             break
42
43 clientSocket.close()
```

Client end Output

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SQL CONSOLE COMMENTS

 Code    ... < X

```
PS F:\CN\A1> python "f:\CN\A1\UDPHeartbeatClient.py"
PING localhost 10 times
udp_seq=1 message="Ping 1 time difference=4.649ms" rtt=5.644ms
udp_seq=2 Request timed out
udp_seq=3 message="Ping 3 time difference=0.000ms" rtt=0.000ms
udp_seq=4 message="Ping 4 time difference=0.000ms" rtt=0.993ms
udp_seq=5 Request timed out
udp_seq=6 message="Ping 6 time difference=0.000ms" rtt=0.000ms
udp_seq=7 Request timed out
udp_seq=8 message="Ping 8 time difference=0.000ms" rtt=0.000ms
udp_seq=9 message="Ping 9 time difference=0.000ms" rtt=0.000ms
udp_seq=10 Request timed out

--- localhost ping statistics ---
10 packets transmitted, 6 received, 40.0% packet loss
rtt min/avg/max = 0.000/1.106/5.644 ms
PS F:\CN\A1> python "f:\CN\A1\UDPHeartbeatClient.py"
PING localhost 10 times
udp_seq=1 Request timed out
udp_seq=2 message="Ping 2 time difference=1.049ms" rtt=1.049ms
udp_seq=3 message="Ping 3 time difference=0.000ms" rtt=0.000ms
udp_seq=4 message="Ping 4 time difference=0.000ms" rtt=0.000ms
udp_seq=5 message="Ping 5 time difference=0.000ms" rtt=0.000ms
udp_seq=6 message="Ping 6 time difference=0.000ms" rtt=0.000ms
udp_seq=7 Request timed out
udp_seq=8 Request timed out
udp_seq=9 Request timed out
Server is down

--- localhost ping statistics ---
9 packets transmitted, 5 received, 44.44444444444444% packet loss
rtt min/avg/max = 0.000/0.210/1.049 ms
PS F:\CN\A1> 
```