UDP Ping Client System Design

1. **Send Pings Using UDP**: The client sends 10 ping messages to the server using UDP. Since UDP is connectionless, no connection setup is needed. Each message contains the sequence number and the timestamp when it was sent.
2. **Message Format**: The ping message is formatted as a single line in ASCII, containing the string `"Ping"`, followed by the `sequence_number` (starting from 1), and the `time` at which the client sent the message. Example: `Ping 1 Wed Sep 20 12:34:56 2024`.
3. **Setting a Timeout**: Since UDP is unreliable, there is a possibility that packets might be lost. The client is programmed to wait for up to 1 second for a response from the server. If no response is received within this timeout period, the client assumes that the packet was lost.
4. **Receiving and Printing the Server's Response**: If the server responds to the ping, the client receives the message and prints it to the console. The message contains the original ping details and the calculated round-trip time (RTT).
5. **Calculating Round-Trip Time (RTT)**: The RTT is computed as the difference between the time the ping was sent and the time the server's response is received. This is done for each individual ping and displayed as part of the response.
6. **Handling Timeout (Packet Loss)**: If no response is received within the 1-second timeout, the client prints a message indicating that the request timed out (i.e., the packet was lost in transmission).
7. **Recording RTT for Statistics**: The RTT of each successfully received response is recorded. If a ping times out, the RTT for that ping is marked as `-1` to indicate packet loss.
8. **Calculating Minimum, Maximum, and Average RTT**: After all 10 pings are sent, the client calculates and prints the minimum, maximum, and average RTT from the successfully received responses. This gives insight into the network latency.
9. **Reporting Packet Loss Rate**: The client calculates and reports the packet loss rate as the percentage of lost packets (those that timed out and received no response) out of the 10 pings sent.
10. **Network Testing**: During testing, the client sends pings to `localhost` or `127.0.0.1`, simulating a local server. After debugging, the program can be tested across a real network by running the client and server on different machines. This tests the behavior of UDP communication across a network.

UDP Heartbeat Client System Design

Changes I made on UDP Pinger client to implement the **UDP Heartbeat Client** functionality in the ping client:

1. **Introduced `Heartbeat` Variable**:
    - A new variable `Heartbeat = 3` was introduced, which represents the number of consecutive missed pings allowed before the client assumes that the server is down.
2. **Heartbeat Decrement on Timeout**:
    - Inside the `except timeout:` block, if a ping times out (i.e., no response from the server within 1 second), the `Heartbeat` value is decremented by 1 (`Heartbeat -= 1`).
3. **Reset Heartbeat on Response**:
    - If a valid response is received from the server, the `Heartbeat` is reset to 3. This means the client assumes the server is functioning properly if it receives a successful ping response.
4. **Check for Server Down Condition**:
    - After each timeout, the code checks if `Heartbeat` has reached 0. If it does, the client concludes that the server is down, prints `"Server is down"`, and breaks the loop to stop further pings.
5. **Early Termination on Server Down**:
    - The loop breaks prematurely if three consecutive pings time out, simulating a loss of connection to the server. This prevents the client from continuing to ping a potentially non-responsive server.

These changes implement a basic heartbeat mechanism to detect server failure based on consecutive missed pings.

UDP Heartbeat Server System Design

Modifications made to the original server code for handling the UDP Heartbeat at Server

1. Time Calculations:
    - The server now records the time when it receives the message using `time_received = time.time()` and calculates the time difference (RTT) using the time sent by the client (`time_sent`), which is extracted from the decoded message.

2. Decoding Message and Extracting Data:
    - Instead of simply capitalizing the message, the server decodes it, splits it to extract the `sequence_number` and `time_sent`, and converts `time_sent` from a string to a float for RTT calculation.
3. Modified Response Message:
    - The server now constructs a new message that includes the `sequence_number` and the calculated RTT (`time_difference`). This message is then encoded and sent back to the client if the packet is not dropped.

These changes enable the server to calculate and report the round-trip time (RTT) for each ping message received.

How many times should you send the UDP Heartbeat packet before you see 3 consecutive responses missing? Is it 10, 100, 1000?

Probability of missing a response, p(missing)
30% or 0.3

Probability of 3 consecutive responses missing
p(missing)*p(missing)*p(missing)
0.3*0.3*0.3
0.027

Therefore We need to send 1/0.027 or 37 packets