CY350X - Computer Networks

Project #1 – UDP Ping Utility

In this programming assignment you will build a ping client and server that provides functionality similar to that provided by standard ping programs available in your operating system. However, the program you develop uses a simpler protocol, UDP, rather than the standard Internet Control Message Protocol (ICMP). The ping utility allows a client machine to send a packet of data to a remote machine, and have the remote machine respond back to the client (an action referred to as echoing). Among other uses, the ping utility allows hosts to determine round-trip times to other machines. Additionally, you will build the ping server.

Because it is unlikely you would experience packet loss on our network, the ping server you build must simulate packet loss. The loss should be random and only occur on about 30% of packets over time. Your server should only respond to properly formatted messages.

Checkpoint due NLT 26 Jan 2023 at 2359.

Completed project due NLT 09 Feb 2023 at 2359.

Requirements

Client:

1. Write a client program in Python 3. Send a series of 10 ping messages to the ping server using UDP. The message must conform to the following protocol: **ping** *sequence_number time*
   The sequence number is an integer value beginning at 1 and ending at 10. Additionally, your time format must be: *day_of_month hrs:mins:secs abbreviated_month year*
   The client should either accept the server's IP and port as a command line argument, or the program should prompt the user to enter the server's IP and port.
   Example message: ping 4 10 10:43:55 Jan 2023
2. For each ping sent, you must calculate the round-trip time of the ping. Wait up to 1 second for a response from the server.
   a. The client socket must timeout if no response is received after 1 second (you must configure this timeout on the socket you create). If a timeout occurs, display a message to the user indicating no response was received for that particular ping.
   b. When a response is received, verify that the response from the server is the expected response (message includes pong and sequence number matches). Display the server response along with the RTT in milliseconds.
3. After all 10 pings have been sent, provide the user with the following information: number of pings sent, number of responses received, success rate (%), maximum round trip time, minimum round trip time, and average round trip time (for minimum, maximum, and average round-trip times, do not include timeouts).
4. You must use only the *socket* module in Python for your socket implementation. You may use other modules for the other requirements.

Server:

1. Write a server program in Python 3 that receives UDP packets. Upon receiving a packet:
    a. Randomly simulate packet loss. If your program determines that packet should be dropped, do not respond to it. Your simulated loss rate should only be about 30% loss.
    b. If the packet is not to be "dropped" and begins with **ping**, respond to the client with the same packet that was received, except replace **ping** with **pong**.
    c. If the packet is not dropped but does not begin with **ping**, then it too should be "dropped".
2. You must use only the *socket* module in Python for your socket implementation. You may use other modules for the other requirements.

Submissions

1. **Checkpoint:** Submit a document that contains a list of modules you intend to use and how you intend to use them. The document should also include a flowchart for your ping client. You can find examples at https://en.wikipedia.org/wiki/Flowchart (The example C-style for loop is a good starting point). The chart should cover from program start to program termination.
2. **Final Submission:** You will electronically submit the code for your ping client and server as two files, named **lastname_firstinitial_pr1_client.py** and **lastname_firstinitial_pr1_server.py**, to Canvas. Use in-line documentation in your code. Additionally, you must complete an e-Acknowledgment statement in CIS.
    a. Along with your code, you will submit a one-page analysis of your design philosophy, problems that you encountered during implementation, and lessons learned.

Resources

We did a basic UDP example in class that may be helpful. You may also reference the Python documentation for modules that will help you meet all requirements.

Pro Tips

- On your VM, python should be run as "python3".
- Start small – get a basic client and server working, then build in each requirement piece by piece.
- Python documentation can help you with text manipulation, time, and date formats.
- Use print statements to help in your debugging efforts and to understand where in code your program is.
- Develop using the loopback interface on your machine (localhost or 127.0.0.1).
- Use Wireshark to look at the traffic you are sending / receiving.