CS 5470 Homework 3

Due: April 10, 2024 (4:30 PM, class time)

Please review the plagiarism policies in the class syllabus!

This is a group homework. You may work in teams of up to 2 people. Please make only a single submission with the names of all team members.

Please read all instructions stated in each question carefully. All instructions must be followed for full credit. Please review the class plagiarism and cheating policies, including the class AI usage policy. Note that any violation of these policies will result in a zero.

A Reliable UDP Protocol

Use the UDP Echo program you implemented for HW 2 as the starting point for this homework. For HW 3, you are required to convert UDP Echo to a general-purpose UDP client-server application that can support reliable data transmission using acknowledgements (ACK), sequence numbers (SEQ#), and retransmissions.

The application should allow the client to reliably send N messages to the server, where N is specified by a user. The contents of each message may be the same, i.e., each message could be the same string of random numbers generated once in your program. Note that the testing and validation of this program requires running the client and the server on two separate laptops located within the same local network. Please extend UDP Echo socket() calls to support this.

Your client-server program must be able to do the following:

- Communicate using a sliding-window protocol. As a simplification, you may arbitrarily pick a window size, e.g. 5, 10, or 15. Note that you will need to devise how the window size must be shared between the client and the server.

- Send application packets along with a packet SEQ# from the client to the server. As a simplification, the packet SEQ# should be sequential and indicate the packet number, not the number of bytes transmitted.

- Send ACKs from the server to the client, including the packet SEQ# being ACK'd.

- Implement in-order delivery of packets at the server-side of the application using a mechanism of your own choice. For example, you may based this mechanism on Go-Back-N or Selective Repeat.

- Implement a mechanism to handle timeouts for ACKs and retransmissions. You may use a mechanism of your choice for this.

- At the client, please print the following:

  - SEQ# of each message being sent using the following string : "Message with SEQ# ___ sent".
  - SEQ# of each acknowledges message using the following string: "Message with SEQ# ___ ACKd".
  - Window size before every transmission is sent and after every acknowledgement is received using the following string: "Current window size: ___"
  - Any timeout event and event handling using the following string: "Timeout event occurred. Message SEQ# ____ resent"

- At the server, please print a string to indicate if an out-of-order packet is received and what your program did to handle out-of-order packet delivery.

Testing and Validation:

Please test and validate your code by running the client and server on two different laptops, M1 and M2, on the same local network. Test your reliability mechanisms under the following conditions:

- Condition 1 (No latency or loss): Run UDP Server on M1 and UDP Client on M2. Send 100 application messages from client to server. Copy-paste your output log in your submission document.

- Condition 2 (Latency and Loss): Please use NetEm to create artificial latency and loss in your client machine (M1). NetEm (Network Emulator) is an enhancement of the Linux traffic control facilities that allow adding delay, packet loss, duplication and other characteristics to packets outgoing from a selected network interface. Please refer to the following link for helpful instructions on installing NetEm and using it to create Conditions 2.1(Artificial Latency, No Loss) and 2.2 (No Latency, Artificial Loss) : https://srtlab.github.io/srt-cookbook/how-to-articles/using-netem-to-emulate-networks.html

  - Condition 2.1 (Artificial Latency, No Loss) - Add enough delay time using NetEm to trigger your timeout timer and generate conditions for retransmission. Run UDP Server on M1 and UDP Client on M2. Send 100 application messages from client to server. Copy-paste your output log in your submission document. Your output should show timeout and retransmissions.

- Condition 2.2 (No Latency, Artificial Loss) - Add 10% packet loss using NetEm to trigger retransmissions. Run UDP Server on M1 and UDP Client on M2. Send 100 application messages from client to server. Copy-paste your output log in your submission document. Your output should show retransmissions timeout and retransmissions.

Submission Guidelines:

Please submit the following as a part of your submission:

1. Source code of your client-server code, with good coding practices and explanatory comments.

2. A 2-3 page document describing the design of your reliable communication protocol, including how you implemented SEQ#, ACKs, timeouts, retransmission, and in-order delivery.

3. Appendix (in addition to the 2-3 page document): Add an appendix to your document clearly demonstrating results of your testing and validation. Please add your code outputs, as well as a short discussion of what the outputs show in the Appendix.