Wyatt Chen
Kaito Hakamata
CSE 160 - Computer Networks

Project 3: Reliable Transportation and Congestion Control

Design Process and Design Decisions

Our design follows with two main parts: setup/teardown and sending data. For setup/teardown, we have the sockets operate according to the three-way handshake algorithm that is default for TCP. The three-way handshake works by the client sending a SYN to the server, the server sends an SYN+ACK back if it received SYN, and the client sends an ACK back to the server for acknowledgement. We update the sequence numbers accordingly. After the three-way handshake is successfully done, the connection is established. If we want to close a connection, it will work like connection establishment with a three-way handshake as well.

For sending data, we had the sender and receiver constantly sending and reading data when possible. Meaning the data was popped off when it was read and popped off with it was acknowledged. The only time this constant routine will be stopped is when the window size is zero, so the sender will wait for a short period until sending another packet of data. The acknowledgement of this packet will update the window size if there was any changes. The choice was to make the sender handle learning the window size and not the receiver due to implementation issues. The data will be analyzed using last byte sent and next byte sent in order to see if there was data transaction or not.

Discussion Questions

1. When the transport protocol implementation picks an initial sequence number to establish a new connection choosing a random number is better. Let's say a node was establishing a new connection, sends the data and crashes. Upon reboot that node tries to establish the connection again. If the sequence number picked was 1, the previous data could be mixed up with the new connection. Choosing random numbers mitigates the chances of this happening.
2. The receive buffer should be at least 2 times the size of the sending buffer because then the receiver can, after obtaining the full send window, tell the sender that it can get another window. This format can technically set up the sender to send a full window constantly.
3. If a SYN flood attack occurs, one of two things would occur. One, if the source address and port ID are the same, there will just be overwrites after overwrites of the old connections. Two, if there are differing source addresses and port ID's, various different connections will be live until a timeout. Limiting the number of SYN that can be accepted from a host during a set time frame can prevent this attack.
4. For a FIN attack, the connections won't time out so the connection never closes. Prevent having too many connections and timing out functionality after not getting data for a while will be a better way to handle this attack.