Project 2 Report

CS 118: Computer Network Fundamentals

Kaitlyne Chan, 004493871
Richard Yu, 304464688

Term: Winter 2018

Prof. Songwu Lu

TA: Zhaowei Tan

University of California,
Los Angeles

1 Implementation Description

1.1 Header Format

The packet header is 8 bytes long and is stored in a vector of characters. The first three bytes are padding, and the fourth byte includes a binary value that represents the packet type. The fifth and sixth bytes store the packet number, and the seventh and eighth bytes store the window size. All the values are stored in binary format as unsigned chars.

1.2 Messages

The client and server first connect via a three-way handshake. After connecting, the client requests a filename from the server, and the server responds with whether the file is found or not. If it's found, the server sends the size of the file. The server then sends packets containing the file data, and the client sends ACKs after receiving them. If any packets are lost, the sender retransmits the packet. After the client successfully receives all the file data, the connection is closed. Upon closing, the client and server send each other a FIN and an ACK.

1.3 Timeouts

For each packet, we store the sequence number and the time that it was sent in an unordered_map. We check if a packet times out by comparing the current time to the packet's timestamp, and if they differ by more than 500 ms, we resend the packet and update its timestamp.

1.4 Window-based Protocol

We implemented a window-based protocol using Selective Repeat, and we used a window size of five packets. The window limits the number of packets sent at once, and it also limits the number of unACKed packets allowed. With Selective Repeat, the receiver sends an ACK for each individual packet it receives. If the ACK is equal to the send base, we move the window to the next lowest sequence number that has not been ACKed. We have a list of the packets' sequence numbers, and once we receive an ACK for a packet, we remove that packet's sequence number from the list so that we maintain a list of unACKed packets. When an individual unACKed packet times out, we retransmit only that packet.

2 Difficulties

Binary files were not transmitted correctly because we stored packet data in a string. The C++ string library functions, such as size(), did not work properly because the functions would only read the string up to a null byte. However, binary data can contain many null bytes. To fix this issue, we used a vector of characters to store the packet data instead.

We also had a problem with overflow when the size was too high. The sequence number would overflow to a value much lower than the window limit, and we would end up transmitting many bytes across the network. This would take so much time that the packets would timeout before we finished filling the window – resulting in an infinite loop of retransmissions. To remedy this, we moved sequence number and the window limit from uint32_t to uint64_t.