Project 3: STCP - Reliable Transport Layer

Yongun Seong 2017-19937

For this assignment, I implemented a reliable TCP-like transport layer on top of the provided STCP API, which behaves correctly even when the underlying network is unreliable.

I also implemented a rudimentary RTO mechanism, following RFC 6298 recommendations. I chose 0.1ms as the minimum RTO instead of the RFC recommended 1s to improve performance over unreliable networks, as we only test on local networks (low latency) and our window size is limited to 3072 bytes. Without this modification, transferring even smaller (~1 MiB) files would take too long on unreliable networks. I also limited the maximum RTO to 10 seconds, for similar reasons.

At the core of my implementation I have an internal `STCPSegment` data structure. It contains the relevant fields of the STCP header (segment number, acknowledgment number, and flags) and the data. These are kept in a linked list sorted in increasing sequence number order. This simplifies handling both ACKs and retransmissions, and lets me reuse the same code throughout the connection lifecycle, from setup, transfer, and teardown.

My code suffers from some limitations arising from the implementation details:

First, it can use a lot more memory than needed when queuing out-of-order segments. Specifically, it can use memory in the order of `3072 * 536` bytes instead of just 3071. However, this will not result in wrong behavior; the application will receive the correctly flow-reconstructed data, and the sender will observe the expected sequence of ACK packets.

Second, my sender implementation assumes that the receiver window is always fixed at 3072 bytes. That is, the sender will correctly limit the in-flight unacked segments to 3072 bytes, but it will break if the receiver shrinks or modifies the window size, making the assumption that that the receiver immediately consumes any ACKed segments.

References

- RFC 792: https://www.rfc-editor.org/rfc/rfc793
- RFC 6298: https://www.rfc-editor.org/rfc/rfc6298