Internet Technology Project 3 Report

Collaborators: Christopher Hellriegel (cmh375), Bhavya Patel (bbp53)

Implementation Summary

Port / framing. UDP port 30038. RUDP header (big-endian): type:1B | seq:4B | len:2B | payload…. Type codes: SYN=1, SYN-ACK=2, ACK=3, DATA=4, DATA-ACK=5, FIN=6, FIN-ACK=7.

Three-way handshake. Client sends SYN(seq=0) → waits for SYN-ACK → sends ACK. On success we print: [CLIENT] Connection established and [SERVER] Received ACK – Handshake complete / [SERVER] Connection established.

Reliable data (stop-and-wait). Message is split into 200-byte chunks with seq = 0,1,2…. For each chunk the client sends DATA(seq=n) and waits for DATA-ACK(seq=n) with RTO = 0.5 s and RETRIES = 10. If the timer fires, the client retransmits the same DATA; the server treats duplicates as out-of-order and re-ACKs the last in-order seq.

Random ACK delay (server). To force retransmissions for evidence, the server sleeps a uniform random 100–1000 ms before replying to each DATA with DATA-ACK.

Teardown. Client sends FIN; server replies FIN-ACK; both print closure: [CLIENT] Connection closed, [SERVER] FIN received, closing … Connection closed.

Capture Analysis:

Handshake.pcap:
These three frames illustrate the simple handshake of a SYN, SYN-ACK, and ACK. This 3-step exchange ensures reliable, synchronized communication between both endpoints before data transmission begins. Sequence and acknowledgment numbers establish initial synchronization.

Data.pcap:
Here, there are UDP protocols that signify the moments of Data Acknowledgments and data acknowledgments that were retransmitted. Analyzing this pcap, we can see that the first and second sets of 200-byte messages are fine. This is for the “Hello from student…” and the lines 1-30. The next set of lines, 31-54, is retransmitted, and we can observe this because the time exceeded the RTO of 0.5 milliseconds. The next lines, 55-80, are also retransmitted with an additional UDP re-acknowledgment. The last following set of lines, 81-100, replicated the actions of lines 55-80.

Teardown.pcap:

The last two frames in the pcap show the final FIN, followed by a FIN-ACK, as indicated by the brief time difference between the FIN and the ACK.

Discussion:
Our client uses a stop-and-wait ARQ: it transmits one DATA(seq=n) and waits for DATA-ACK(seq=n) before sending the next chunk. The server intentionally introduces a random 100–1000 ms delay before replying, while the client's RTO is 0.5 s. Whenever the server's delay exceeds the RTO, the client's timer fires and it retransmits the same DATA(seq=n). The server treats such duplicates as out-of-order and re-ACKs the last in-order sequence (you can see this in our logs, e.g., "Sent DATA-ACK seq=3 (delay 917 ms)" followed by "Re-ACK seq=3 (received 3)"). Reliability is guaranteed because the sender only advances to the next sequence after receiving the correct DATA-ACK(seq=n); duplicates are harmless (idempotent at receiver) and losses are recovered via timeout + retransmit. The three pcaps corroborate this: the handshake file shows only SYN/SYN-ACK/ACK; the data file shows multiple frames with the same sequence value (retransmissions) and matching ACKs; the teardown file shows just FIN/FIN-ACK.