Forester Vosburgh
Kevin Oyler
CS3251

Design Report RELDAT

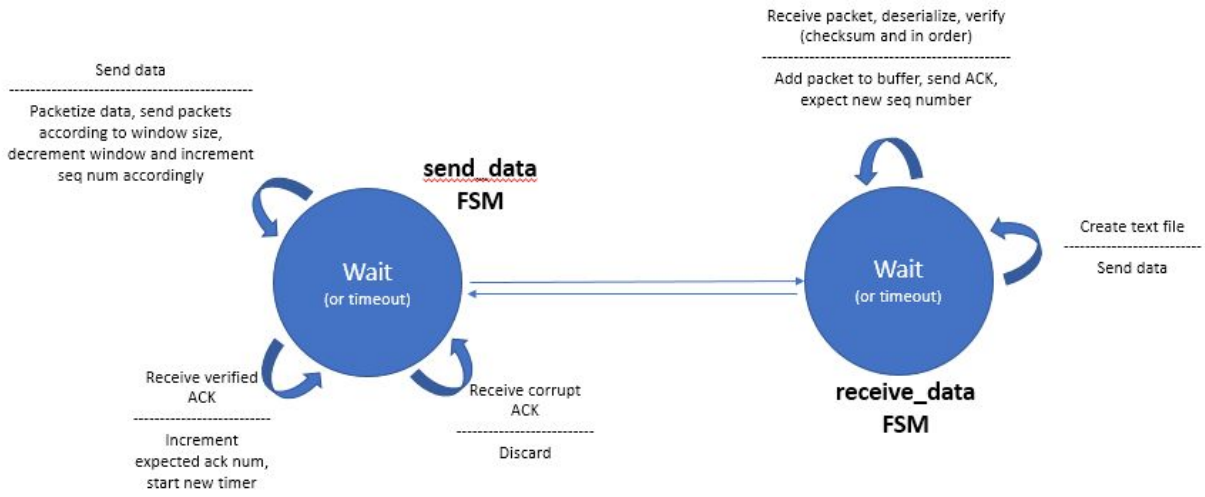
High Level Description

Reldat works similarly to TCP in that it starts with a 3-way handshake. The client will send a SYN packet with no payload, the server will receive this and send a SYN ACK, then the client will send an ACK with no payload. Afterwards, the client can issue a transform command to send the text file to the server. The server will batch receive as many packets as its window size allows, process what it needs to do with each packet and ACK them accordingly. MD5 hashing checks for corruption in packets and timeouts account for lost packets, these two packet types will be dropped. ACKs are only sent on in-order, verified packets.

Header Structure

Each packet's header is defined by the PacketHeader class.
The header consists of these fields (all integers):
Source port, destination port, sequence number, ack number, SYN flag, ACK flag, FIN flag, window size, checksum, payload length

FSM



Non-trivial Functions

- Data sending function
    - Works similarly to a Go-back-N reliable transfer. Find how many packets are necessary to send the file through a factor of bytes and payload size, then buffer these packet-size byte strings of info into an empty hash per packet. Loop over

each entry in the hash, extract the data, and packetize it (pads packets to the same size as well, this just made it easier for our sending). Keep track of each sequence number as it relates to the hash entry. Send number of packets that correlate with the window size (batch send). If seq numbers 0-4 have been sent and an ACK number of 4 has been received back, you know 0-4 were all successful (like Go-back-N). If timeouts are reached (lost packets) or packets are corrupted, resend. If a max number of retries is met, send an ending connection packet to the other side and close the connection.

- Data receiving function
    - Works similarly to Go-back-N reliable transfer, sliding window implemented. Packets that are in order are added to a data buffer, out of order packets are discarded. Deserialize, checksum, and verify type of incoming packets while decrementing the window size for each packet received. ACK verified, non-corrupted packets by sending a cumulative ACK number (i.e. if seq num 1-3 were successful packets but 4 and 5 were corrupt/lost, send ACK number 3). If the other side of the connection times out, close the connection after some max timeout time. Move sliding window accordingly for next receive batch of packets.

Questions

- How does RELDAT perform connection establishment and connection termination?
    - RELDAT performs a 3-way handshake as described in the high level description for connection establishment between a client and the server. Various timeouts can close the connection from the client or server end, as well as the client executing the disconnect command
- How does RELDAT detect and deal with corrupted/lost packets?
    - RELDAT uses timeouts and checksums to deal with corrupted and lost packets. If a packet's acknowledgement isn't received within a given time, it is considered lost. If a received packet fails its checksum, it is considered a mangled packet that has been corrupted (if initial packet checksum and after-sent packet checksum do not match). We checksum with a standard md5 hash from an imported Python library.
- How does RELDAT support bi-directional data transfers?
    - RELDAT batch sends packets and batch receives packets. Once the receiving side's window has been filled, it processes the packets and then sends another batch. It will not receive while it is sending and vice versa.
- How does RELDAT provide byte-stream semantics?
    - Byte-stream semantics are provided by making each packet an object which is serialized before being sent through the socket. After deserializing and verifying the received packet, we can process its header fields and type.
- Are there any special values or parameters in your design (such as minimum packet size)?
    - Packets are all padded to the same size to make sequence numbers easy. Max payload size is 1000 bytes, max packet size is 3200 bytes, max timeouts is 5.