CS 456 Assignment 3

Design

The assignment was written in C++, using the built-in sockets library.

Packets were represented in the program as basic structs, holding on to their type (DAT, ACK, EOT, or invalid), sequence number, length (when serialized), and payload. We have invalid packet types because packets only ever exist as a value type and never on the heap, so some sort of "null" is necessary.

The base sender and receiver are both classes which handle opening, reading, and writing to files, as well as creating any necessary sockets. These classes are included in both the go-back-n and selective repeat implementations, to reduce code reuse. The specific implementations for go-back-n and selective repeat are implemented in the sender and receiver subclasses, which override the download_file and upload_file methods.

In go-back-n on the sender side, we use utimer to jump to a handler function when we reach our given timeout. This handler is registered for the SIGALRM signal at the beginning of file upload. After registering it we read in a full frame window from the source file, send each packet in the frame window, then wait for an ACK from the client. Before waiting for a packet reply, we start the ualarm so that if we don't receive a packet within the timeout period, we jump to our handler which sends the frame window again and restarts the waiting. If we do get a reply in time, we process the ACK and advance our frame window as necessary.

On the receiver side, we create a frame window using std::array with a fixed size of the maximum window size. This lets us insert packets that fit in our window as we receive them into their correct positions, without having to worry about reordering them. After receiving an data packet, we go through our buffer and process as many packets as we can, until we reach an empty spot in our buffer, then we shift the buffer so that our next requested packet is at the beginning. On each loop we always send an ACK for the most recently received data packet, in case it was not received the last time.

Testing

The assignment was tested using the given channel emulator and attempting to transfer files of various sizes. Different timeouts and discard probabilities were used, and files up to around 200 MB were tested.