Matt Stropkey   Writeups

For stop and wait, I generated the checksum by adding up the values of the packets sequence number, acknowledgement number, and each character of the payload. Then I took the one's complement of this, to give the checksum. When the receiver received a packet, it first checked to see if the sequence number was correct. Then if this was correct, a checksum was generated and compared to the checksum stored in the packet. Since A was the only one sending actual packets, I set the acknowledgement number for all packets from A to 0. If all of this was correct, then the packet must be valid and a acknowledgement was sent. If any of this was incorrect, either a negative acknowledgement was sent or if the packet was a duplicate of the previous acknowledged packet, a duplicate acknowledgement of the previous packet was sent. This was done by storing the values of the last acknowledged packet in the receiver to compare against a received packet if it was determined that the packet was the correct next packet. For the timer value, I tried to do what TCP does in that the rtt is calculated based off of the variation in time it takes for packets to be sent and acknowledged. If a packet is resent, the rtt was not calculated for that packet.

To compile I used, gcc project2_stop_wait.c -o stopwait

For go back n, I used the same approach for the checksum. Since both sides could now send and receive packets, it was more difficult to keep track of everything. Professor Khattab told me to resend packets on receipt of a NAK message, but this seemed to flood the network too much. Thus, I decided to send resend a message on receipt of a NAK only if the timer for that packet was 70% used. This made it so if a packet was recently resent, it would not be resent immediately if the resent packet was never given a chance to make it to the other side. I also implemented piggybacking acknowledgements on all data packets to help deal with packet loss. Even if a packet arrived out of order, as long as the checksum was valid, the receiver could check the acknowledgement number and see if it acknowledged any previous unacknowledged messages. When a negative acknowledgement was sent, I sent the negative of the next packet expected to be received. When a sender got this, it could figure out that the receiver has received up to that many packets, thus also serving as a acknowledgement. I only sent negative acknowledgements if the checksum was invalid or the sequence number was invalid. If a receiver did not have any unacknowledged sent packets, it simply ignored the negative acknowledgement. Also if a acknowledgement was received for a previous acknowledged packet, it was also ignored. When a acknowledgement came in, I checked to see if the next unacknowledged packet would have a timeout based on it's start time plus the current timer value. If the value was less than the current time, all unacknowledged packets were resent immediately. Otherwise, the timer was reset with the remaining time left.