CSCD58 Assignment 2
Derek Lai
999035020
Laihoche

1. Short Answers

1.1. rdt 3.0 will still function correctly even if packets and ACKs are delayed. In the event a packet was delayed, nothing special happens as the server is waiting for the client to send a packet to it, the client just sends the packet as it is suppose to just delayed. In the case an ACK is delayed, say ACK1, the client would time out from not receiving an ACK1 for the packet, PKT1, it sent and resend the packet. The client would resend PKT1 and at the same time the delayed ACK1 would be recieved by the client. The server would recieve PKT1 and notice that it is a duplicate and ignore it and a second ACK1 would be sent by the server for the repeated PKT1 even though it already sent an ACK1. When the client finally receives the first delayed ACK1, it responds by sending a delayed PKT2. If all the PKTs and ACKs send are delayed, this essentially forces double of everything to be sent and received due to the delay but no data is explicitly lost due to the delay. The file transfer speed would be greatly reduced due to the number of duplicate ACKs and PKTs exchanged.

1.2. TCP can still detect duplicate packets despite fast network speeds due to a sliding window size. The sending host is only allowed to send up to the size of the window before it must wait for ACKs from all the packets send, at which point the window size is allowed to move up. In addition, each octet of data send is given a sequence number helping the server and client align the packets in the correct order in the event they get out of order. The sequence number also helps the server detect duplicate packets if there are duplicate sequence numbers. If an ACK is not recieved for the packets sent within the timeout period, the data is resent. Duplicate packets are ignored.

1.3.
   i. At 100% utilization, an window size of 8333 packets is required.
   1Gb/s = 1000000000 b/s, (1000000000 bs $\times$ 0.100s) $\div$ (1500B * 8) = 8333
   ii. If a packet is lost utilizing fast recovery, cwnd = cwnd / 2. From then, cwnd increases to max from there. In this case 8333 / 2 = 4166. Then of transmission * RTT, 4167 * .100 s = 416.7 seconds. Utilizing the old model, the window size is reset to 1, and exponentially rises until 50% and at which point it linearly grows. At a window size of 1 to 4166, it takes $\log_2(4166-1) \times 0.1s = 1.20$ to reach 50% of window size, then $1.20s + 416.7s = 417.90s$
   iii. On a 10Gb/s network, window size would increase to 83333 packets, Fast recovery time would increase to 4166.6 s to return to 100%. On the old model, 4168.1 s.
   iv. If the segment size increased to 10KB, the window size would decrease to 1250 packets. Fast recovery time would decrease to 62.5 s and slow recovery would be 63.43s.

2. .2
   Blue = HTTPServer
   Red = ThreadHTTPServer
   Yellow = ThreadPipeHTTPServer
   Green = NIOHTTPServer