Sung Mo Yang

11280117

High-level summary

I opened the pcap file through Python's basic file opener. I used the dpkt library to convert the contents of the pcap file into processable data. I iterated through the data, saving each packet into a dictionary, where the keys are the tuple: (source port, destination port) and the contents are an array of TCP packets associated with the tuple. The dpkt library has its own structure of creating TCP packet objects and so these TCP packet objects were saved into the array. Using this method, I had a dictionary entry for each flow, and each array associated with a key was a series of TCP packets in order of transaction of that flow.

The remainder of my code is how I processed each flow. I could retrieve the number of TCP flows initiated from the number of unique entries in the dictionary. For each TCP flow, its source port, source IP address, destination port, and destination IP address could be retrieved from the first TCP packet object because this information was guaranteed to exist in initial the three-way handshake.

Identifying the first two transactions was simply going through the array of TCP packet objects, storing the first two packets with unique SEQ and ACK numbers, and then finding the TCP packets from the receiver that acknowledged those two unique packets.

The sender throughput for each flow was calculated by adding every single byte of each TCP packet and then dividing it by the difference between the first timestamp and the last timestamp.

The first three congestion windows were identified by counting the number of packets sent from the sender to the receiver until an acknowledgement was finally sent back to the sender.

The number of times a retransmission occurred due to a triple duplicate ACK was found by isolating and iterating through just the ACK TCP packets sent by the receiver. Whenever three consecutive, identical ACKs were sent, another retransmission was counted.

The number of times a retransmission occurred due to timeout was found by isolating the TCP data packets sent by the sender. Whenever two consecutive, identical SEQs were sent, a retransmission was counted. This was the total number of retransmissions that occurred, so I subtracted the number of times retransmission due to triple duplicate ACKs occurred. The remaining difference is the number of times a retransmission occurred due to timeout. This method was approved by piazza post @177.

Congestion window size growth

The first section of part B requires a comment on how congestion window size grew in the pcap file. From my observations, congestion window size started at 10, then 20, then 33. It grew by an estimated addition of 10 each time. This is indicative of AIMD, Additive Increase Multiplicative Decrease, a property of TCP to expand upon sending rate. TWhethere, it can be assumed that for each loss event (whether it be due to timeout or triple duplicate ACKs), the congestion window would be cut in half, before increasing by an addition of 10 again for each successful packet sent.