Derek Yin
113251504

For Part A, each packet is filtered by source port in a dictionary where source ports are keys. The values of this dictionary contain packets with the designated source port. There were a total of 3 TCP flows and this is found by counting all the unique source ports, minus the receiver port. To calculate the sequence number, ack number, and window size of the first two transactions after the three-way handshake for each flow, we first retrieve either the last ACK packet of the three-way handshake if there is a PSH, ACK packet directly proceeding it, or we retrieve the first ACK packet directly after the three-way handshake if there is no PSH, ACK packet. This becomes part one of our first transaction. To find the packet that is sent back to this flow to complete the first transaction, my code looks through the previously defined dictionary for all values with the key equalling the receiver port, in this case, 80. We find the first packet in this dictionary with seq number equalling the ack number of part 1 of transaction 1. Values such as window size, seq number, and ack number are all accessible through the TCP object of the dpkt library. Transaction 2 is similarly found for each flow by taking the ack number of part 2 of transaction 1 and finding a packet with sequence number = that ack number. The procedure is repeated to find part 2 of transaction 2, except we take the SECOND packet in the dictionary with seq number equalling the ack number of part 1 of transaction 2.

To find the throughput of each TCP flow, my code accumulates the length of each TCP object starting from the SYN/ACK packet to the FIN packet of the specific source port, and this represents the total number of bytes received. To find the amount of time that passed, take the timestamp of the last packet and subtract it from the timestamp of the first measured packet. The throughput is then represented in bytes / second.

To estimate the congestion window size of each TCP flow, my code first finds the RTT of each flow by finding the timestamp of the SYN packet and subtracting it from the timestamp of the SYN/ACK packet. This time is then used to calculate how many packets are sent per RTT, for each flow. This estimates the congestion window size and it is observed that the number of packets sent per RTT changes as the congestion window size changes at roughly RTT intervals.

To estimate the total retransmissions, both due to triple duplicate acks and due to timeout, my code accumulates a counter every time a sequence number belonging to an ACK packet is seen more than 3 times during looping. My RTO is = 2 * RTT and I used this to calculate how many of the total retransmissions were due to timeout.