Part B:

The program used for this part of the assignment was written in analysis_pcap_tcp.py. You can run this code through the command: python analysis_pcap_tcp.py . Running this will ask which file you would like to use and prompt you for what you would like to inspect.

1.

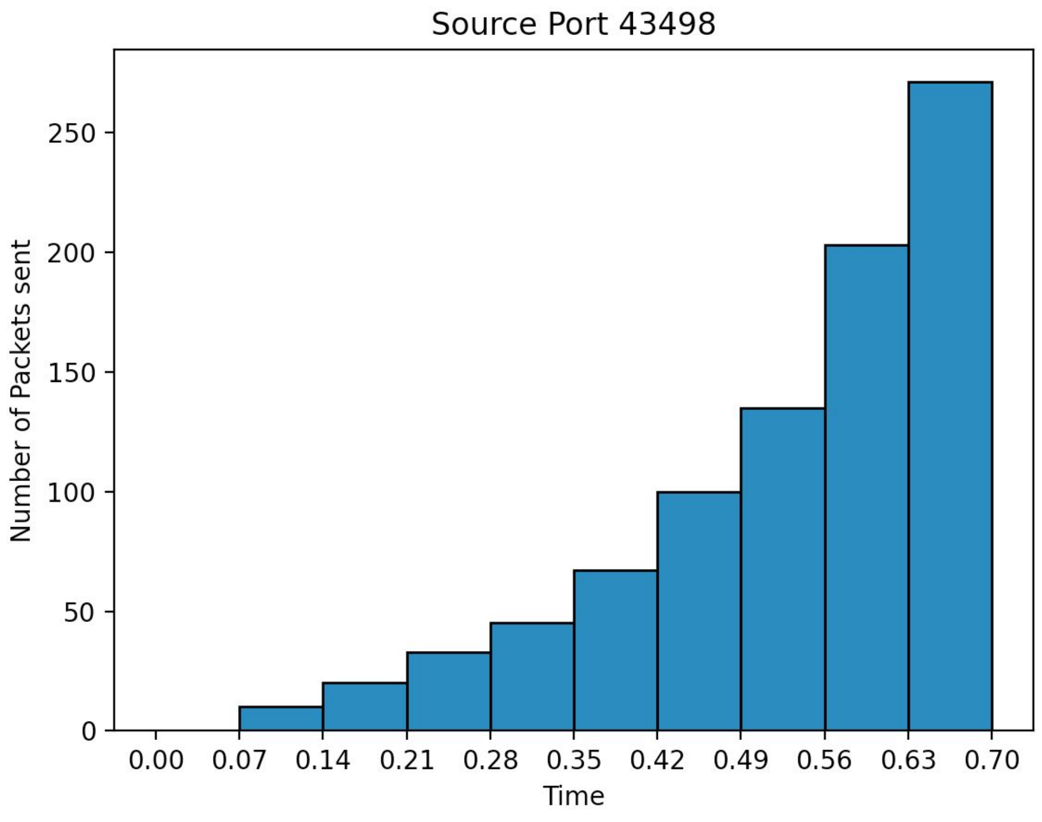
Congestion Window is the amount of data that can be sent before receiving anything back. The estimate should be done from the sender, the sender will control what is being sent based on the acks, or lack of acks that it has received.

The way I calculated the 10 congestion window sizes was that I calculated how many packets got sent within a period of time, multiplied by 1448. Similar to what I said above, the sender won't send any packets out if it knows that the receiver is congested thereby altering the number of packets it will send within this period.

Source Port: 43498

First 10 Congestion Window Sizes: [0, 14480, 28960, 47784, 65160, 97016, 144800, 195480, 293944, 392408]

Growth Factors After First Non-Zero Congestion Window: [2.0, 1.65, 1.36, 1.49, 1.49, 1.35, 1.5, 1.33]



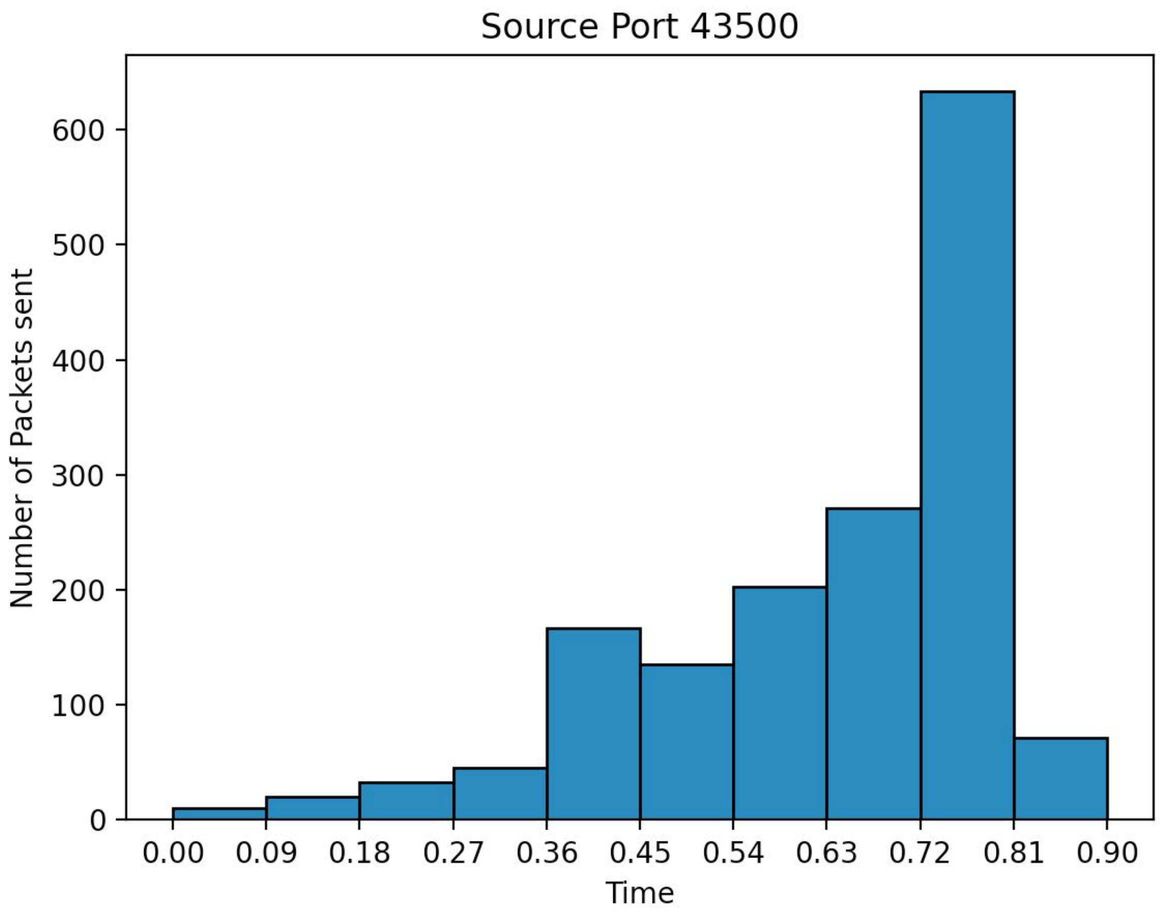
The first congestion window size was 0, however, this is because no packets have yet to be sent within that period of time. After such, the next congestion window is understandably 14480.

As you can see from the graph, the congestion window steadily increases, meaning that during this period of time it has not reached the max congestion window size.

Source Port: 43500

First 10 Congestion Window Sizes: [14480, 28960, 47784, 65160, 241816, 195480, 293944, 392408, 916584, 102808]

Growth Factors After First Congestion Window: [2.0, 1.65, 1.36, 3.71, 0.808, 1.5, 1.33, 2.34, 0.112]

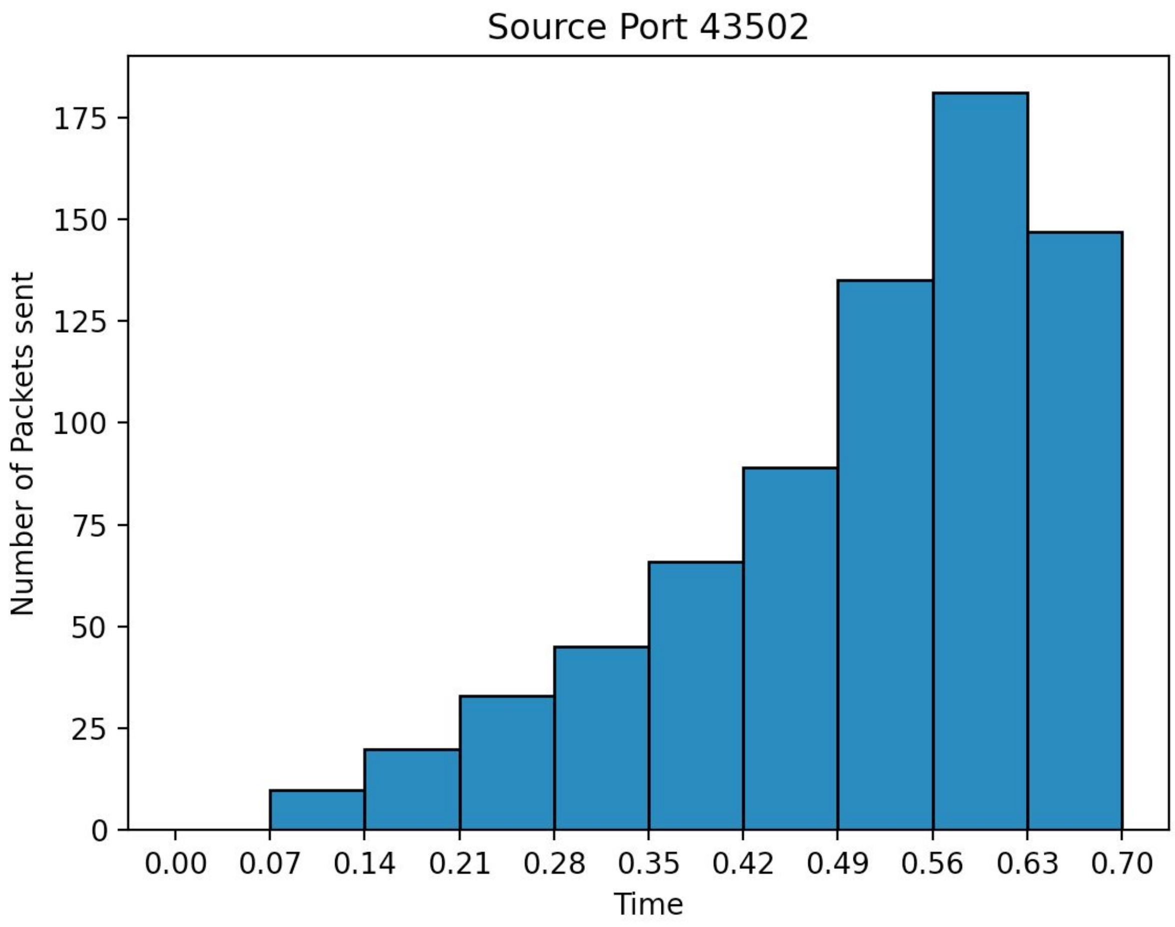


The initial congestion window size is 14480. Looking at the graph produced, it is likely that there was a delay in retrieval of acknowledgement packets, thereby holding things up. When the sender finally got the acknowledgements it needed, it then was able to send several packets (.72-.81). This in turn meant that that the sender didn't receive many packets back in the next period of time (.81-.90) and had to delay sending more out until there was more room.

Source Port: 43502

First 10 Congestion Window Sizes: [0, 14480, 28960, 47784, 65160, 95568, 128872, 195480, 262088, 212856]

Growth Factors After First Congestion Window: [2.0, 1.65, 1.36, 1.47, 1.35, 1.52, 1.34, 0.812]



The first congestion window size was 0, however, this is because no packets have yet to be sent within that period of time. After such, the next congestion window is understandably 14480.

As you can see from the graph, the number of packets steadily increases, meaning that the max congestion window has yet to be reached. However, between .63 and .70, the number of packets sent are less, meaning that it is likely that we experienced some congestion. During the next period of time (.7-.77), we can expect to send out a small number of packets, only in response to the acks we received.

2. A duplicate ack is sent when the receiving end receives an out of order packet. Three in a row of these is a good indication that a packet has been lost. These duplicate acks are sent from the receiver to the original sender, therefore it is ideal to look for these duplicate acks at the original sender.

Source Port 43498:
- Total Number of Retransmissions: 3
- Number Due to Triple Dup Acks: 2
- Number Due to Timeouts: 1

Source Port 43500:
- Total Number of Retransmissions: 94
- Number Due to Triple Dup Acks: 4
- Number Due to Timeouts: 90

Source Port 43502:
- Total Number of Retransmissions: 0
- Number Due to Triple Dup Acks: 0
- Number Due to Timeouts: 0

The way I calculated this was by counting the number of corresponding acks between retransmissions. If this number was greater than or equal to 3, I counted the retransmission as one due to Triple Duplicate Acks. I had previously calculated the total number of transmissions, and used this number, minus the number of triple duplicate acks, to calculate the number of retransmissions due to timeouts.