ECE 3700 Lab 4 Report – Winter 2019

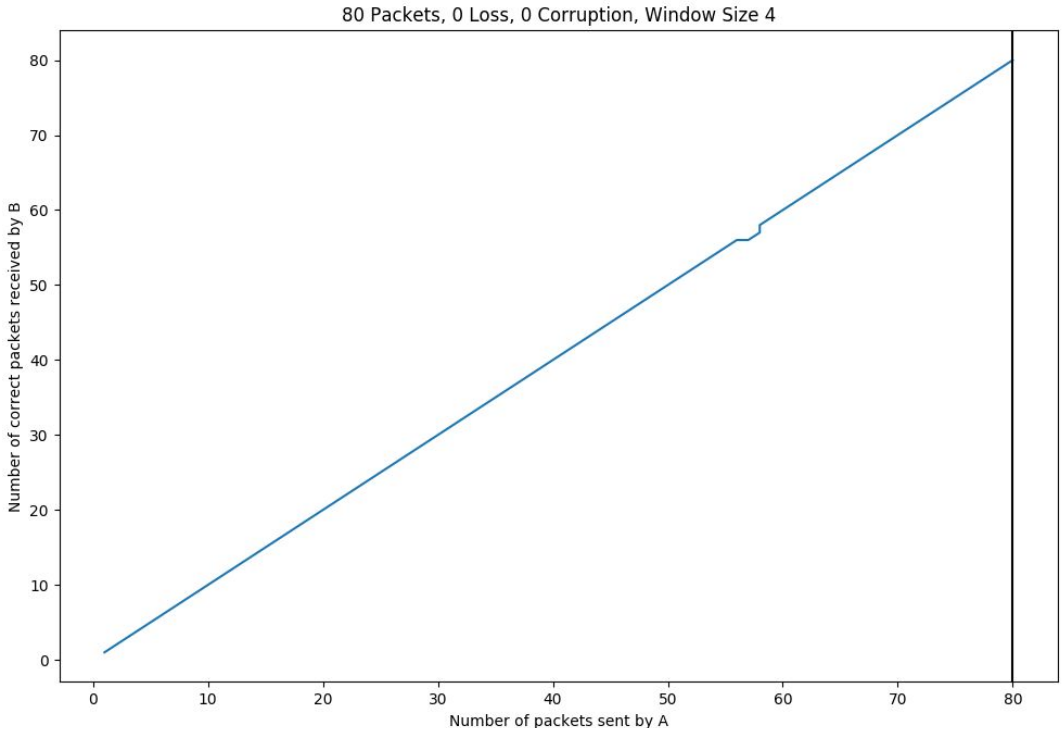
Daniel Lovegrove (7763168)

OUTPUT GRAPHS

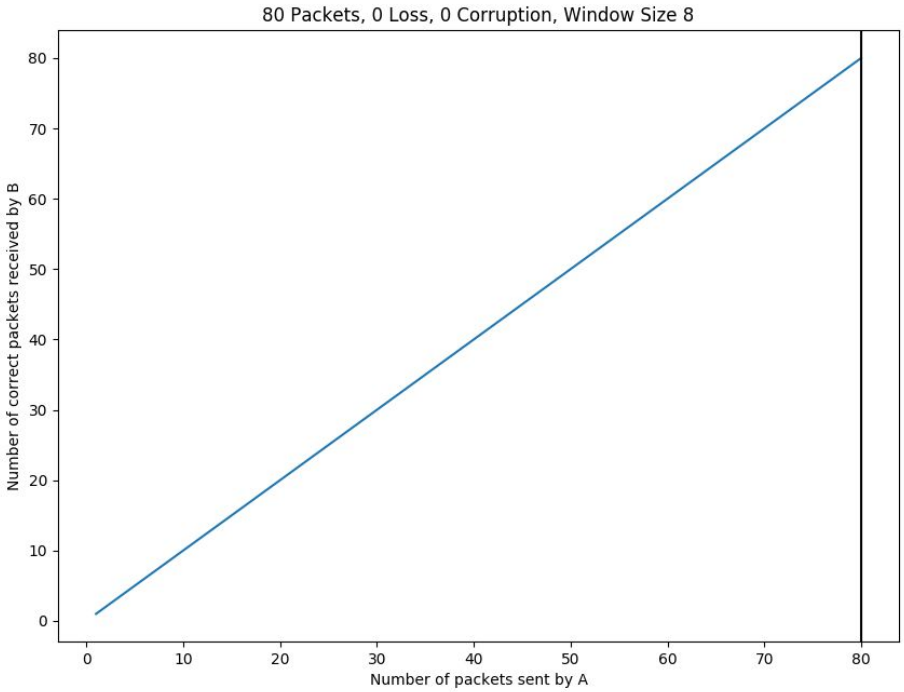
The following are the graphs I obtained from running the GBN code I wrote for lab 4. In each case, 80 packets were sent from A to B. Appendix A contains the script used to generate the graphs.

Each graph plots the number of packets sent by A against the number of packets received by B. The black vertical line at the end of each graph indicates the number of packets required to be sent by A for all packets to be received by B correctly.

Figures 1 and 2 below show that when there is no loss or corruption, B receives each packet from A correctly, since A has to send 80 packets for B to receive the full 80 packets. The window size does not affect the total throughput when there is no loss or corruption.

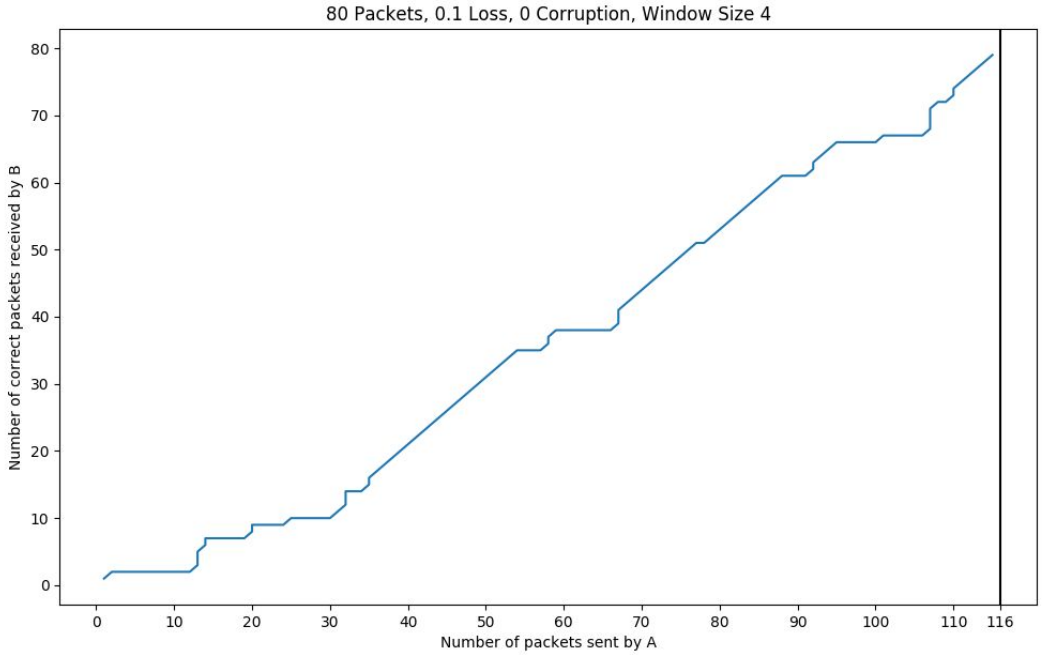


(Figure 1: 80 Packets, 0 Loss, 0 Corruption, Window Size 4 – Daniel Lovegrove)

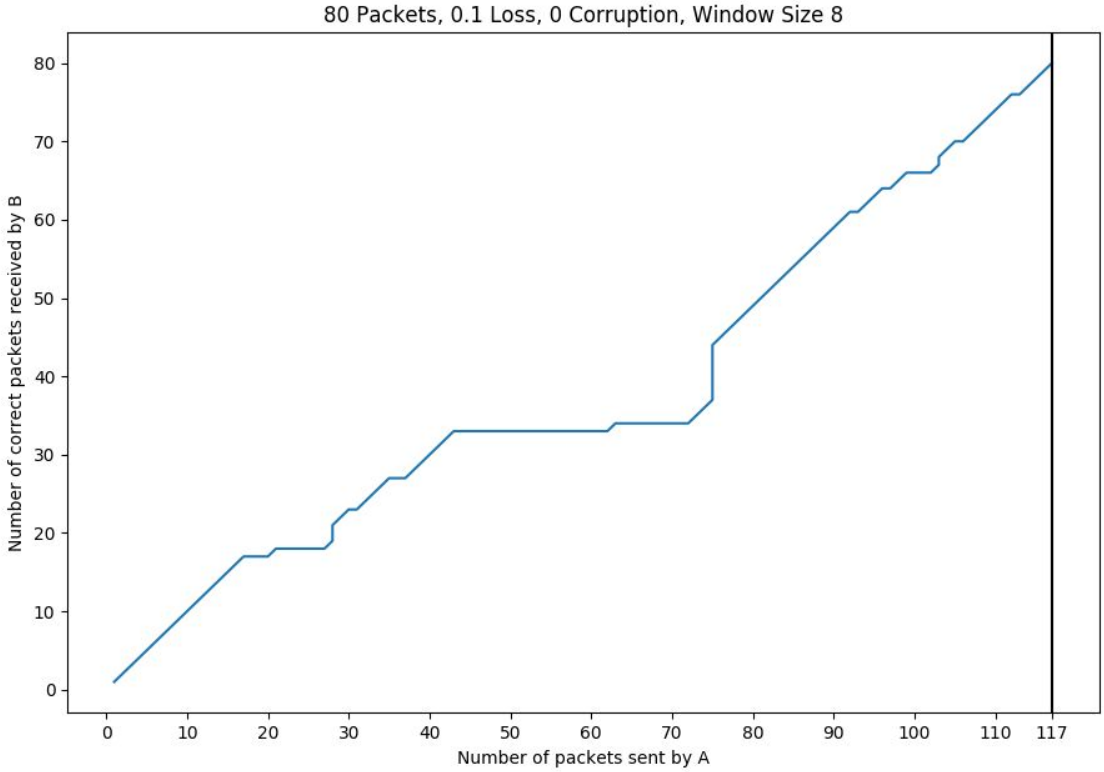


(Figure 2: 80 Packets, 0 Loss, 0 Corruption, Window Size 8 – Daniel Lovegrove)

Figures 3 and 4 below show that when there is a 10% chance of corruption A has to send more packets for B to receive them all correctly. Figure 3 uses a window size of 4 and Figure 4 uses a window size of 8. The window size did not greatly affect the total throughput.



(Figure 3: 80 Packets, 10% Chance of Loss, 0 Corruption, Window Size 4 – Daniel Lovegrove)



(Figure 4: 80 Packets, 10% Chance of Loss, 0 Corruption, Window Size 4 – Daniel Lovegrove)

The large flat lines where A sends multiple messages before B receives them correspond to the times when A gets a timeout from not receiving ACKs from B. When a timeout occurs, A sends multiple messages that correspond to the unACKed messages remaining in the current window. Figure 4 has larger flat lines compared to Figure 3 because the window size in Figure 4 is larger.

APPENDIX A: GRAPH GENERATION SCRIPT

The following script is written using Python 3.

```
import matplotlib
matplotlib.use('TkAgg') # If TkAgg doesn't work, try QT4Agg
import matplotlib.pyplot as plt
from pathlib import Path

# Set these based on the file to process
filename = 'Run Output/80Packets_L0.1_C0.1.txt'
graph_title = '80 Packets, 0.1 Loss, 0.1 Corruption'


filelines = []
with open(Path(filename)) as f:
    filelines = f.readlines()

packets_sent_A = []
packets_received_B = []
curr_num_packets_sent_A = 0
curr_num_packets_received_B = 0
last_event_was_A = False

for line in filelines:
    if "(A): Sending a message with" in line:
        if last_event_was_A:
            packets_sent_A.append(curr_num_packets_sent_A)
            packets_received_B.append(curr_num_packets_received_B)
        curr_num_packets_sent_A += 1
        last_event_was_A = True

    if "(B): Received the correct packet" in line:
        curr_num_packets_received_B += 1
        last_event_was_A = False
        packets_sent_A.append(curr_num_packets_sent_A)
        packets_received_B.append(curr_num_packets_received_B)

x_ticks = list(range(0, curr_num_packets_sent_A, 10))
x_ticks.append(curr_num_packets_sent_A)

plt.plot(packets_sent_A, packets_received_B)
plt.title(graph_title)
plt.axvline(curr_num_packets_sent_A, color="black")
plt.xticks(x_ticks)
plt.xlabel("Number of packets sent by A")
plt.ylabel("Number of correct packets received by B")
plt.show()
```