Adam Alberski

112890087

CSE 310

Programming Assignment #2

High-Level Summary

Introduction:

The purpose of this program is to parse any pcap file as input, and return various statistics about the flows in the file such as the source and destination IP addresses, the total amount of bytes and time taken to complete a flow, and estimated congestion window sizes.

Libraries used:

- dpkt: Most functions in this program rely on dpkt, including grabbing data from packets and finding ip and tcp data for example.
- socket: Used to fetch a readable form of IP addresses from bytes provided by dpkt
- datetime: Provides the means to make timestamps from dpkt readable by humans

General code:

The program initially requests user input to determine what pcap file to parse. Once the file is opened, a for loop iterates through the file, skipping any packet not under a TCP connection and dissects packets to grab data such as their IP, TCP data, and flags, and uses these to construct a 'Packet' object which is appended to a list in order to store every single packet. The program then uses a for loop to iterate through the new list of packets, and compares the ports and addresses of each other until a match is found and the information from those two packets are used to construct a 'Transaction' object which is appended to a list of transactions. FInally, in a similar fashion to checking packets, all transactions with 'SYN' or 'FIN' packets in the list are compared until they find matches (transactions with 'SYN' will be paired with 'FIN' when they're on the same port). These values are then used to construct 'Flow' objects added to a list, which each represent the entirety of TCP connections over specific ports. After all code is processed, formatted output is printed out for each flow.

Part A

(a) To receive an accurate list of the source/destination ports and addresses, I gathered the data by making a reference to the first ‘SYN’ packet of each flow. From there, I used socket functions to print out the values for source and destination IP addresses taken from functions I created in the Packet class, and retrieved port numbers from the sport and dport attributes of the TCP objects in the packet.

(b) To find the first of two transactions in a flow, I started iterating through the list of packets and grabbed the first packet that had a matching port and that appeared chronologically after the first ‘SYN’ response packet of the flow. Then, I scanned through the packets again with a nested for loop to find a matching packet with this one in order to create a transaction, which is then appended to a list (‘first_two’) in the Flow object to store the first two transactions. After the first transaction is received, I iterated through the packets list again, only now making sure this first packet comes after the second packet of the first transaction, and again used a nested for loop to find a matching packet to create the transaction. Once both of the transactions were in the list, I went through each packet in the transactions and printed the source and destination addresses in a similar fashion to (a), and referenced values from the tcp attribute to get sequence, acknowledgement and window size numbers. For time, datetime was used to print out the timestamp for each packet of the transaction.

(c) To receive values for throughput, I began by iterating through the list of packets, and adding to a counter only if the packet came chronologically between the starting and ending packets of the flow, and only if the packet belonged to the same port as the flow. This counter was used to describe the total number of packets in a flow. For the total data sent, another counter was used that added the length of the tcp attribute for each packet that belonged to the flow. To get the time period, I calculated the difference between the timestamp of the first packet sent in the flow with the last packet received in the flow. All of these values were created as attributes of the Flow class in order to be able to call them when printing. For bytes per second, I calculated the total amount of bytes divided by the time period to get the rate. No object was created for bytes per second, and is only established when printing output for the flow.

Part B

1) To calculate the congestion window size of each flow, I began by calculating the RTT per flow by finding the difference between timestamps of the SYN and SYN/ACK packets, also known as the first packet sent and its response. Then, I iterated through the list of packets with a while loop that ended once a list containing congestion windows was full. This loop starts at the first packet once a SYN is responded to in the flow (the SYN/ACK), and continues through the list adding to a counter to track the congestion size. The loop only adds to the congestion counter if the packet belongs to the flow, and is cut off once the time difference between the current and starting packet is greater than the RTT. Once the loop is finished, the counter for congestion size is added to a list, and the starting value is set to the current position in order to repeat the process and find the next congestion size. I was able to observe that the congestion window size continued to grow exponentially from the former window, meaning that the deeper the flow runs, the more packets that are able to be transferred in one RTT. This could either be due to the size of a packet in bytes being diminished, or the speed of the connection increases.