Steven Tung
4/2/2024
CSE 310: Computer Networks
Professor Jain

Questions from the Homework Document Answered

High-level summary of the analysis_pcap_tcp code:
Part A:
When creating the list of the flows, I filtered based on source IP and port and destination IP and port. I created a tuple with those 4 values that serve as my dictionary's key. The dictionary contains all the relevant fields I need to store the flow. The packets are stored in a list in the flows dictionary.

Total bytes were calculated by summing the length of a packet's TCP portion. Only packets not from receiver to sender were summed. Duration is the difference between the end and start timestamps. Throughput is just total bytes divided by duration.

Part B:
For RTT calculation, I stored the timestamps of the SYN, ACK packets of the receiver. With each timestamp, the RTT is estimated by finding the difference between the timestamp of the stored packet and the SYN packet.

Comment on how the congestion window size grows: this was done by finding the number of packets sent since the packet that I initiated the counting, and then comparing the time elapsed with the RTT. If it was greater than RTT then add the count to the cwnd list and reset the packet counter. In summary, I counted the number of packets that were sent from the sender in one RTT. The process is then continued throughout the entire flow.

For retransmissions, I first determined how many retransmissions were done. I had to keep track of the sequence numbers of each packet in the flow, the timestamp, and also the number of instances that were transmitted. Next, you need to ignore PUSH flags. Then you find the number of timeouts by comparing the max(timestamp) - min(timestamp) with 2RTT. If it's greater than 2RTT then it is a timeout. Duplicate ACK is if it was less than 2RTT. In summary, if after compiling the timestamps of duplicate packets from the sender the time interval is greater than 2RTT then that is a timeout, otherwise, it is a duplicate ACK.

Both Parts: Results were validated using WireShark. I kept playing with my code until I matched the Wireshark or got close to it. If any explanation of my process needs to be clarified, I left comments in my code which could be useful.