Lab 4: TCP Congestion Control

Tan Shin Jie 1003715

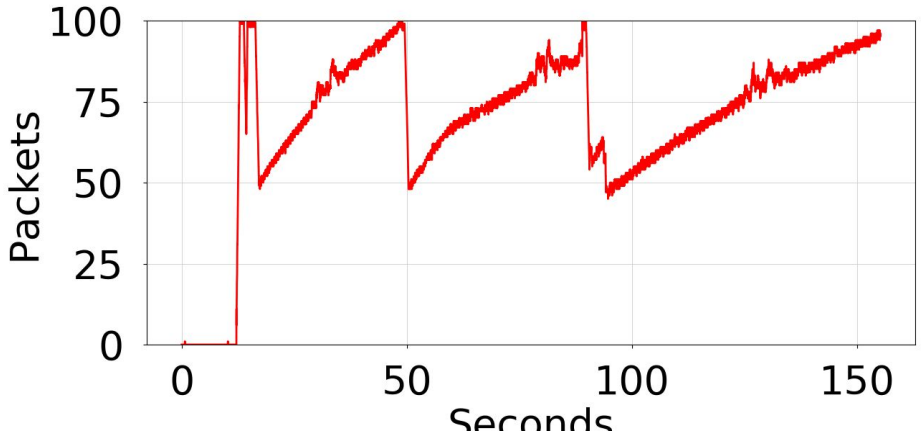
– What is the normal time required to download the webpage on h1 from h2?
1 second

– What was your initial expectation for the congestion window size over time?
The bottleneck link has bandwidth 1.5Mb/s(187.5 kB/s).
With the index.html having a size of 177.7kB, I expect the congestion window to grow exponentially because it's always in the slow start state until the last ACK packet of the file is received. After that, the congestion window will remain constant because there are no more activity from the client to the server.

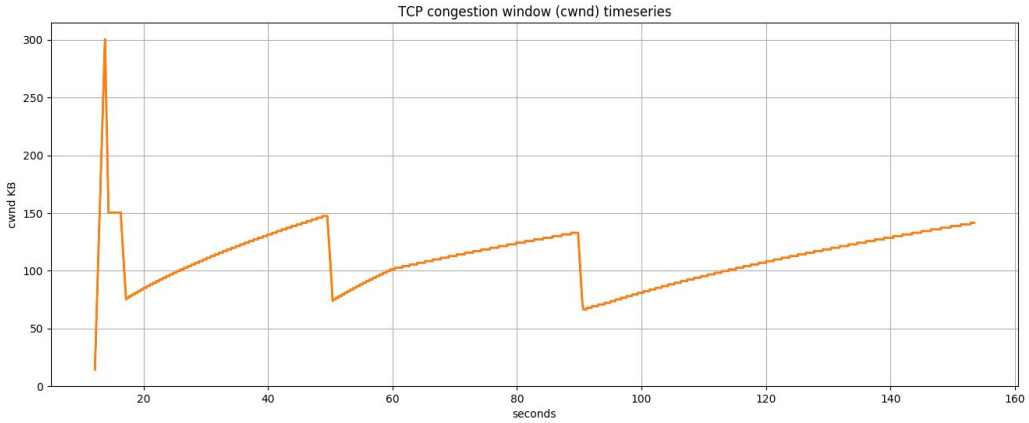
– After starting iperf on h1, did you observe something interesting in the ping RTT?
Before iperf, RTT is on average 38.936ms.
After iperf, RTT is on average 516.116s (~13 times longer).

– After starting iperf on h1, why does the web page take so much longer to download?
iperf client bloated the buffer of the switch between h1 and h2. Packets are lost due to the overflowed buffer. Retransmission of the packet caused the total time to download to increase.

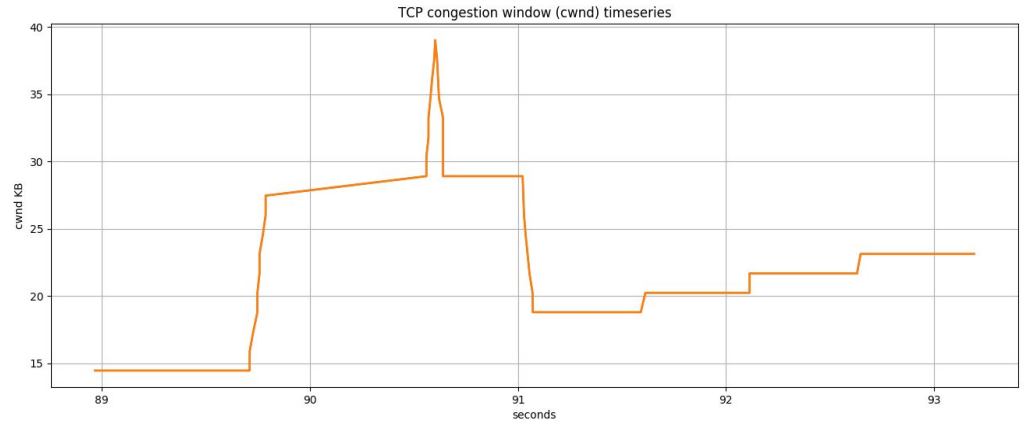
– Please provide the figures for the first experiment (with qlen 100)
Buffer size:



cwnd for iperf:



cwnd for wget:



* Please comment on what you can see in the figures

Time taken to download index.html is 4.5 seconds.

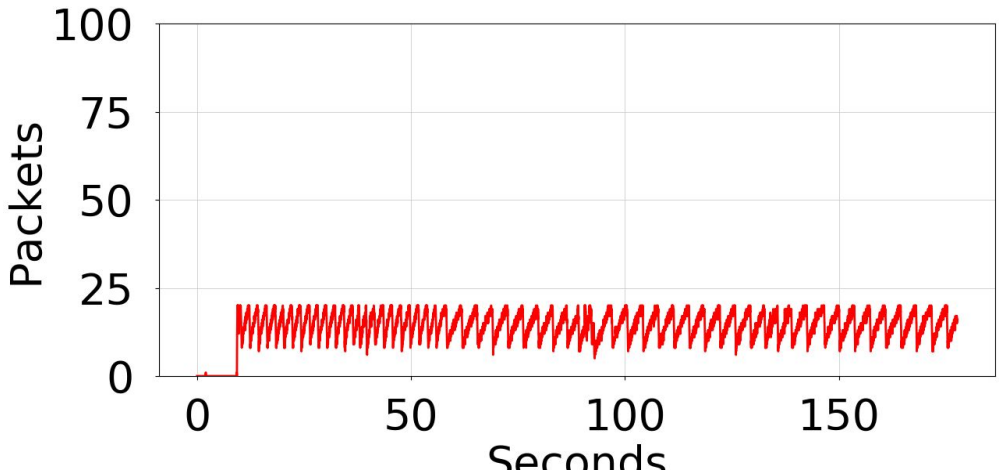
Buffer graph has a sparse saw-tooth pattern with peaks at 100.

cwnd for iperf first increased rapidly immediately readjusted to a saw-tooth pattern with a peak at around 150kB.

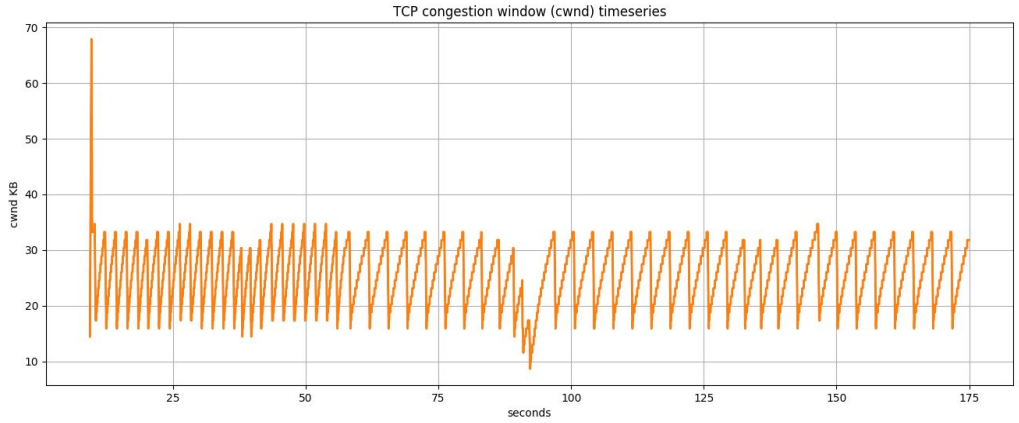
cwnd for wget had two huge steps increase from 15kB to a peak at slightly less than 40kB and then had two huge steps decrease to 20kB and followed by a steady increase.

– Please provide the figures for the second experiment (with qlen 20)

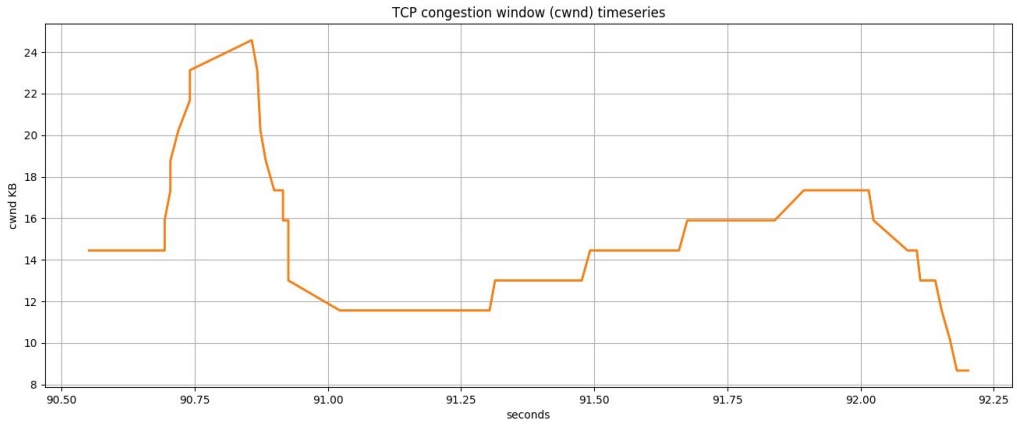
Buffer size:



cwnd for iperf:



cwnd for wget:



* Please comment on what you can see in the figures, and what is different (and why)

Time taken to download index.html is 2.2 seconds.

Buffer graph has a denser saw-tooth pattern with a peak at 20.

cwnd for iperf first increased rapidly and readjusted to a saw-tooth pattern with a peak at around 35kB.
cwnd for wget increases from 14kB to the highest peak 24kB and followed by a huge drop to 12kB. Then, cwnd increases gradually to peak at around 18kB before it drops again.

Comparing the cwnd graphs of qlen=100 and qlen=20, we can see a similar pattern in the cwnd of iperf, that is, a huge spike at the beginning and then followed by a saw-tooth pattern. Both cwnd for iperf also experience a drop when the wget cwnd is introduced.
Interestingly, qlen=20 gives a better performance for wget despite the fact that the highest peak of cwnd in qlen=100 is larger. This can be explained by the queueing delay. A larger buffer allows more packets to queue in the switch. However, with the iperf constantly bloating the networks with packets, the majority of the buffer spaces are taken up by the iperf packets, hence, wget packets have to wait much longer when it manages to squeeze in the buffer. With a smaller buffer, it provides a “fairer” treatment to both clients because packets from the bloating client are dropped instead and therefore packets from the other client do not have to queue too long before it gets sent out.