Lee Gui An 1002651
50.012 Networks Lab 4

Part 1: Initial State

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

Initially, the delay between the two hosts is about 30ms:

rtt min/avg/max/mdev: 30.185/30.236/30.288/0.181ms

– What was your initial expectation for the congestion window size over time?

Expectation of cwnd: cwnd == available bandwidth

– After starting iperf on h1, did you observe something interesting in the ping RTT?

Ping RTT dramatically increased: avg ping was 1275.110ms

– After starting iperf on h1, why does the web page take so much longer to download?

The bandwidth of the link is consumed by the iperf file. This means that there is little available packet link rate for the web page to be loaded from, and when packets are lost, the congestion control takes over, reducing the effective bandwidth for wget(since both processes fight for the same bandwidth).

Lee Gui An 1002651
50.012 Networks Lab 4

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

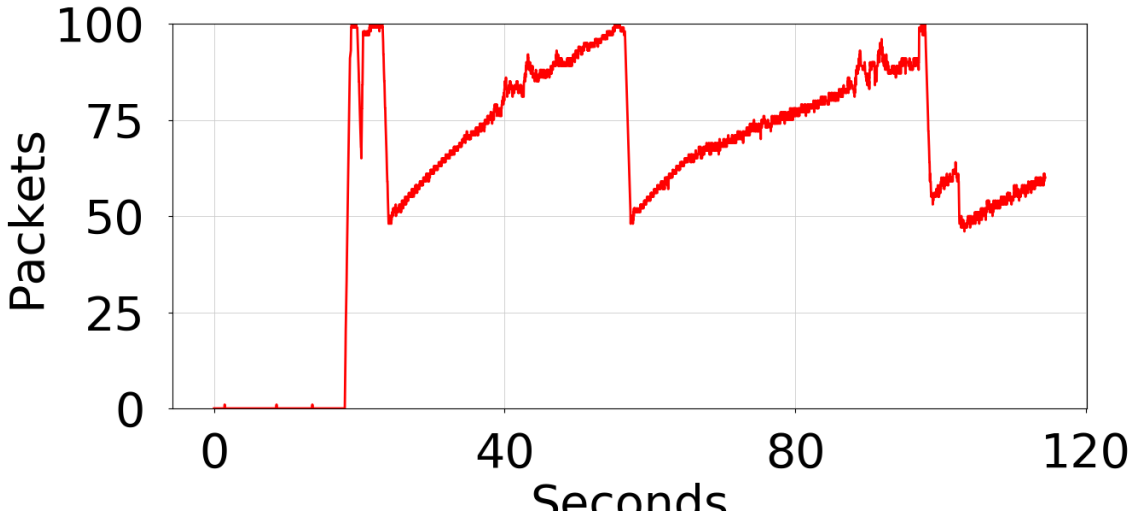


FIG 1.1: Packet Occupancy v. Time

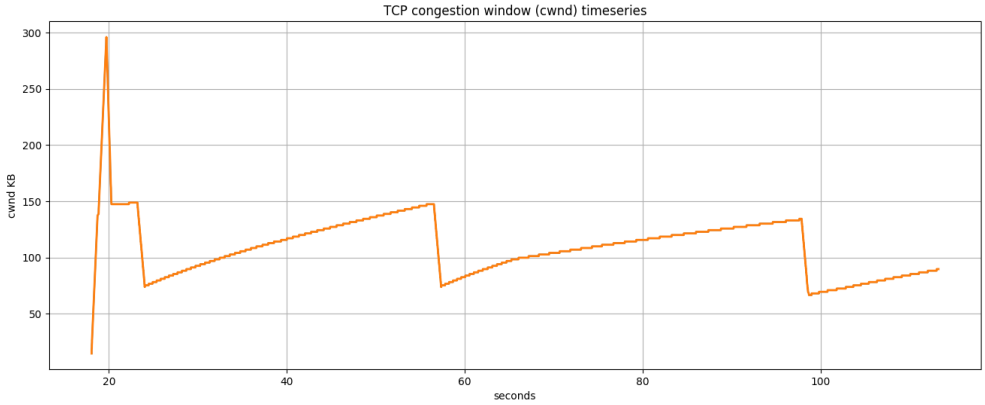


FIG 1.2 iperf cwnd over time

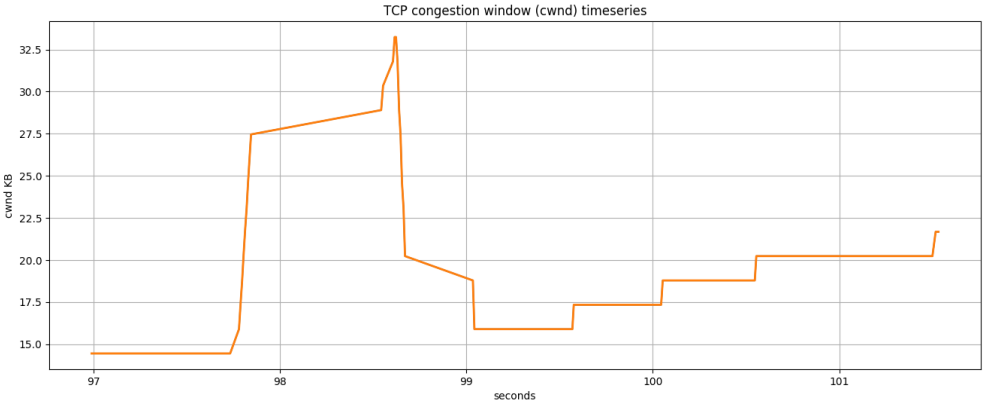


FIG 1.3

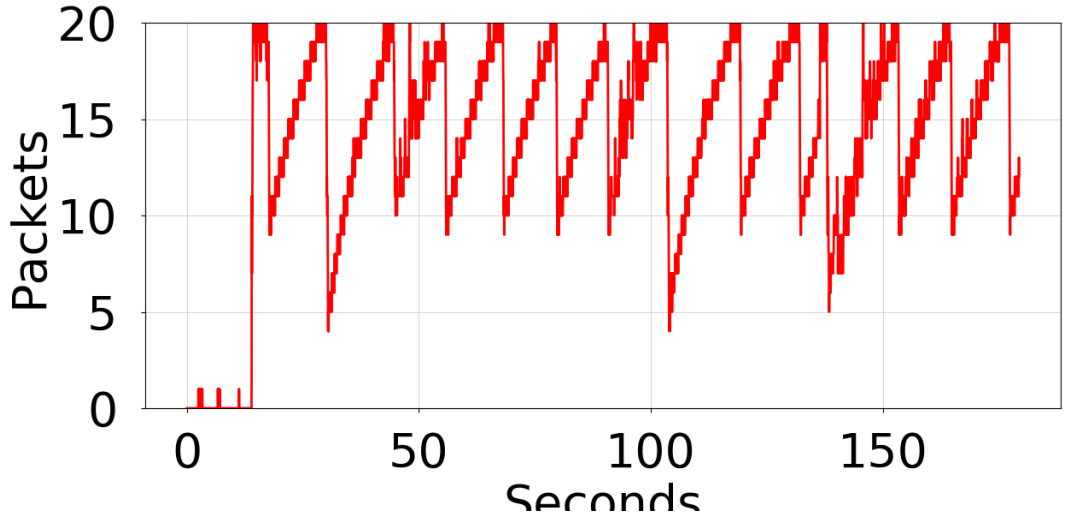
Lee Gui An 1002651
50.012 Networks Lab 4

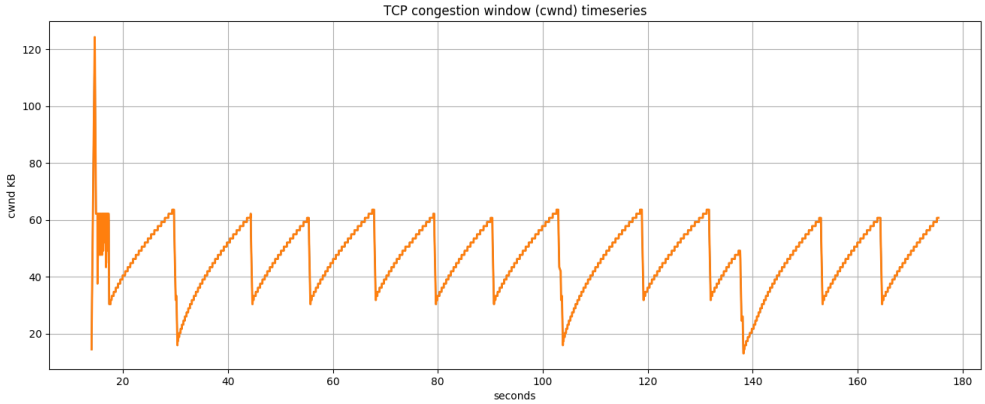
*** Please comment on what you can see in the figures**

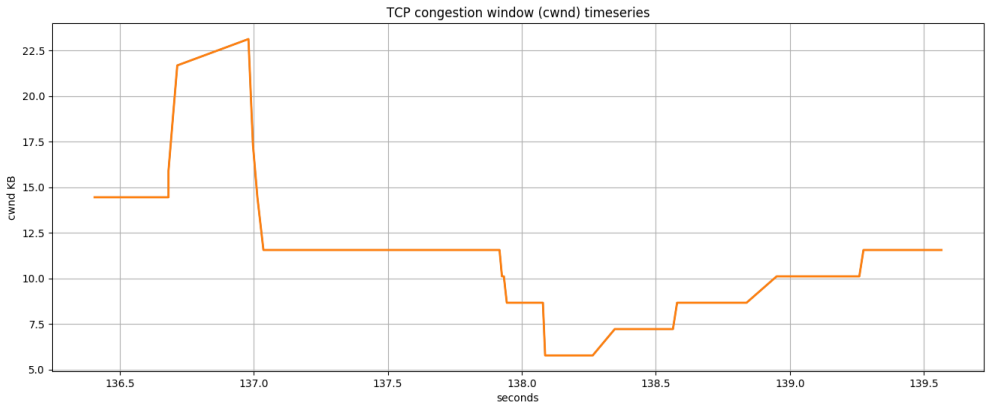
As we observe here, we can clearly identify that in the initial state of the cwnd of the *iperf* transfer, there is a sharp exponential spike in the slow start phase as cwnd seeks capacity, followed by a steady climb as it hits congestion-avoidance. This corresponds with the spike towards full bandwidth capacity seen in the packet monitor.

After this point, we see that when we attempt the *wget* program, we too see a spike in the packet utilization and the cwnd value of *wget*. As both programs maximize the packet rate, they both fall into collision avoidance, halving their cwnd rate, but since *iperf* has more packets in flight, we see that soon after *wget* hits congestion again and halves itself, to slowly climb back up again with repeated fast-recovery phases. This is matched by the decreased max cwnd before congestion for *iperf*.

Lee Gui An 1002651
50.012 Networks Lab 4







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

In this image, we can clearly see that there are a lot more congestion peaks and cwnd resets than in the previous image. This is because with a decreased buffer size, less packets can be held in queue before congestion occurs, and thus *iperf* will hit saturation much faster, causing there to be more reset peaks. However, *wget* is less affected – since it starts with a much more limited bandwidth, it has less packets in flight, and hence can almost continuously increase its bandwidth – until it comes into packet congestion with

the rising *iperf*. This contest for network resources is the reason why we see the double curve where each process eats into the other – but the reduced packets allows for *wget* to acquire a larger share by exploiting the ‘halving’ loop of *iperf* to acquire more resources.