Lab Exercise 4 Answers

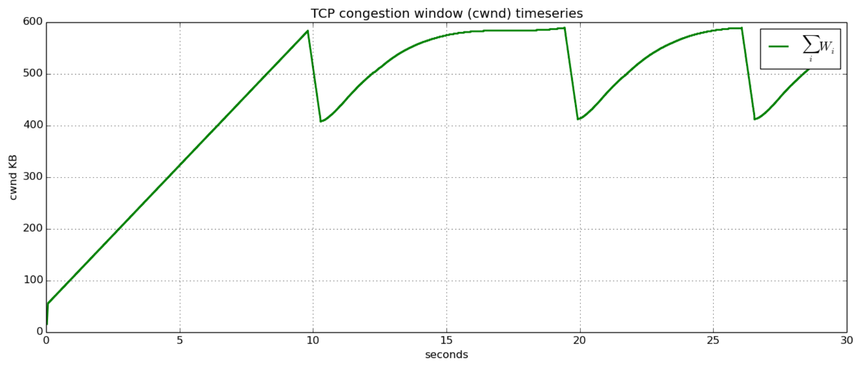
Marking: 12 marks, Exercise 1: 5 marks, Exercise 2: 7 marks.

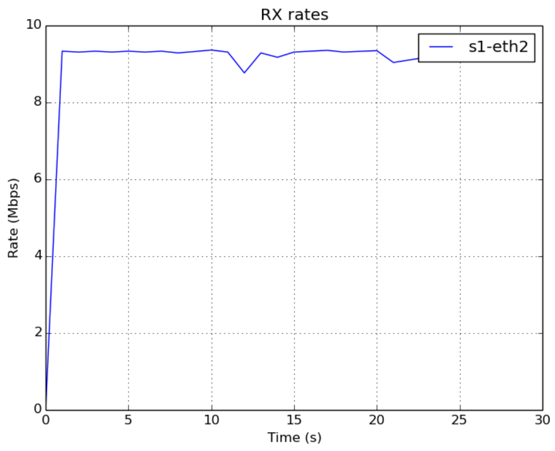
Exercise 1: Parking Lot

Question 1. (1 mark)

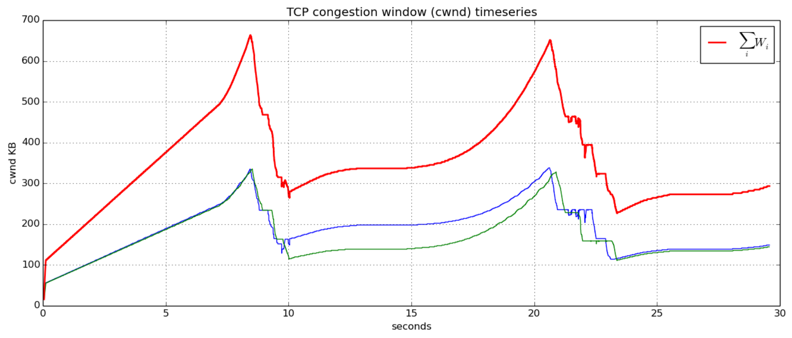
Submit the various plots for different topologies (N = 1 to 5).

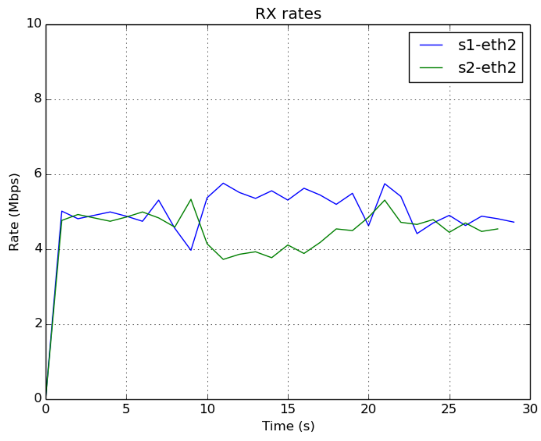
N = 1



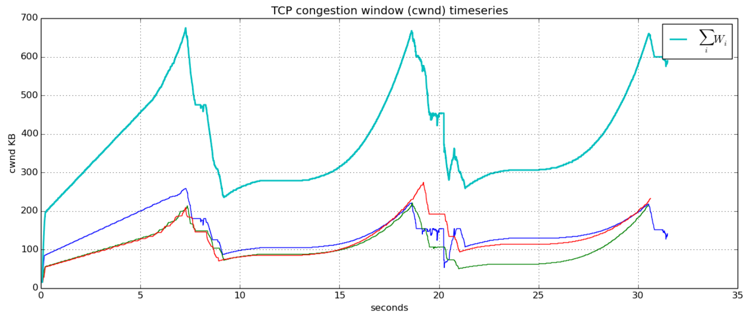


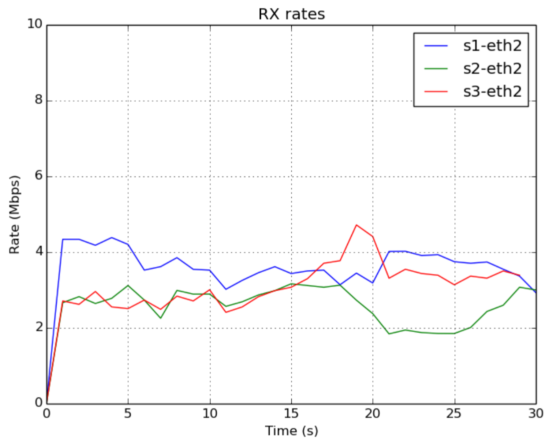
N = 2



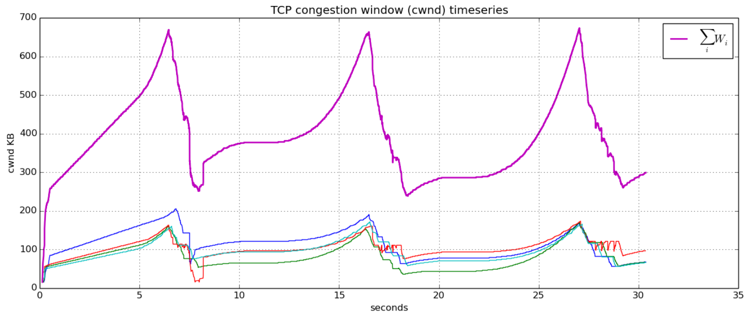

RX rates
s1-eth2
s2-eth2
Rate (Mbps)
Time (s)

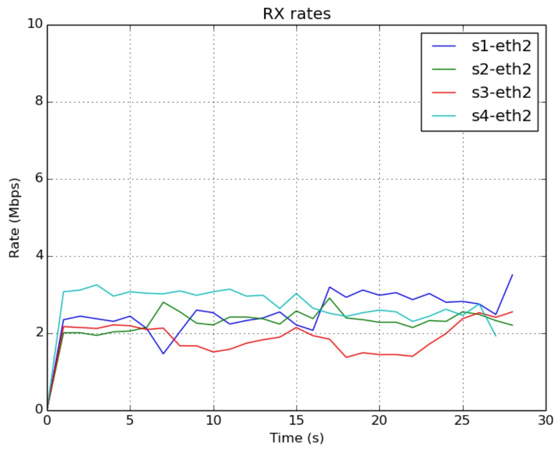
N = 3


TCP congestion window (cwnd) timeseries
$\sum_i w_i$
cwnd KB
seconds

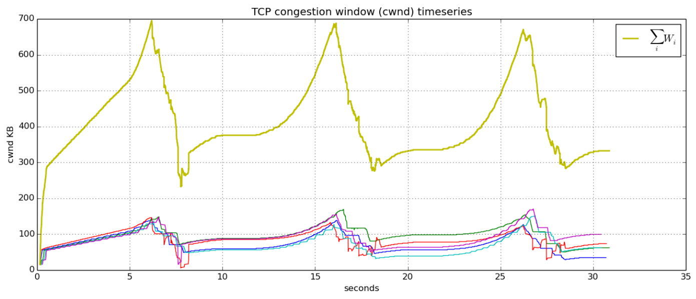

RX rates
s1-eth2
s2-eth2
s3-eth2
Rate (Mbps)
Time (s)

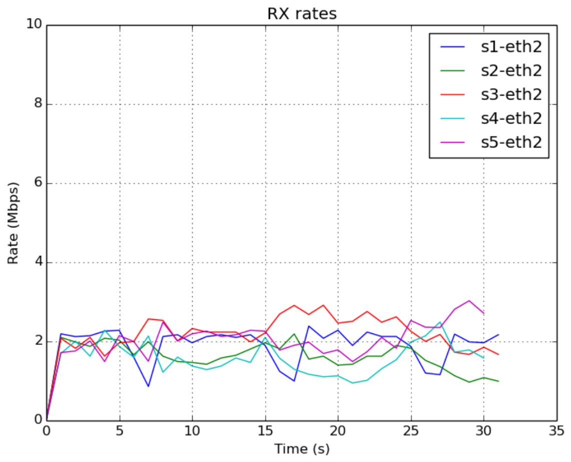
N = 4


TCP congestion window (cwnd) timeseries
$\sum_i w_i$
cwnd KB
seconds



N = 5





Question 2. (1 mark)

Looking at the window size's sawtooth pattern, the sawtooth's increasing part is a curve instead of a straight line. This is because the queuing delay increases due to the buffer filling, which causes the ACK to take longer to get back, which delays the next window increase.

Question 3. (1 mark)

The plots for the cumulative cwnd sawtooth are remarkably similar for N = 2, 3, 4, and 5. The plot for N = 1 varies somewhat from the rest, in that the variance is lower for N = 1 as compared to the others.

Question 4. (1 mark)

The N flows should ideally get an equal and fair share of the bottleneck link, which in the case of the parking lot topology is the link between s1 and the receiver (of capacity 10 Mbps). Thus each

flow should roughly get 10/N Mbps. The rate plots for different values of N do suggest that this is indeed the case on average.
If all hosts use UDP, each flow would still get an equal and fair share of the bottleneck link as above. However, none of the flows would employ congestion control which could lead to significant packet loss.

Question 5. (1 mark)

The bottleneck bandwidth is shared equally among all active TCP connections. Thus if a flow opens multiple TCP connections it can obtain a greater share of the bandwidth. For example, assume that N = 2 and sender 1 opens 2 connections, then each TCP connection would get about 10/3 Mbps (1 connection from sender 2 and 2 connections from sender 1). Thus, the cumulative share for sender 1 would now be 20/3 Mbps.

If one host uses UDP with the other flows using TCP, then the UDP flow will on average get a greater share of the bandwidth. This is because when the TCP flows detect congestion, they will cut their cwnd, while the UDP flow will continue to transmit without pulling back. Thus the UDP flow will get a greater share of the bandwidth during the time periods when the TCP flows cut back their cwnd and go through congestion avoidance.

Exercise 2: Buffer Bloat

Question 6 (0.5 marks)

The average RTT between h1 and h2 is 20.616 msec.

The web page download takes 1 second.

Question 7 (0.5 marks)

The RTT statistics when the long-lived iperf flow is running are as follows:

Min: 415.894 ms, Avg: 627.553 ms, Max: 813.036 ms, mdev: 127.668 ms

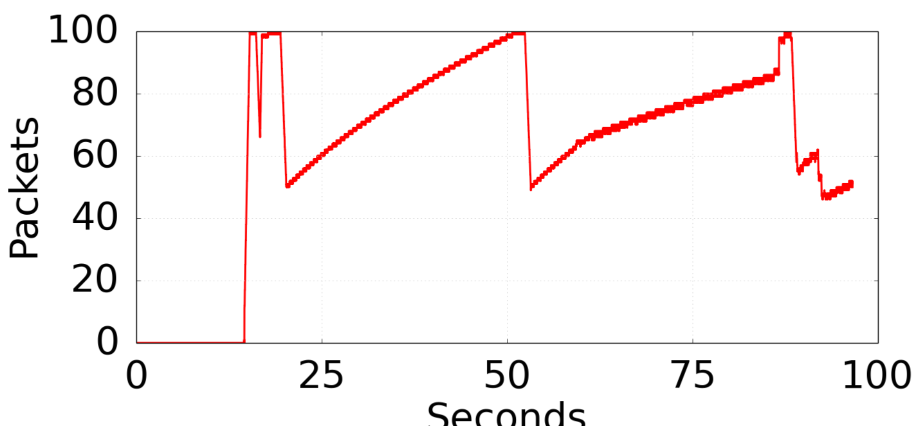
The RTT increases significantly. This is because the ping packets are delayed in the router buffer behind the packets from the iperf flow.

Question 8 (0.5 marks)

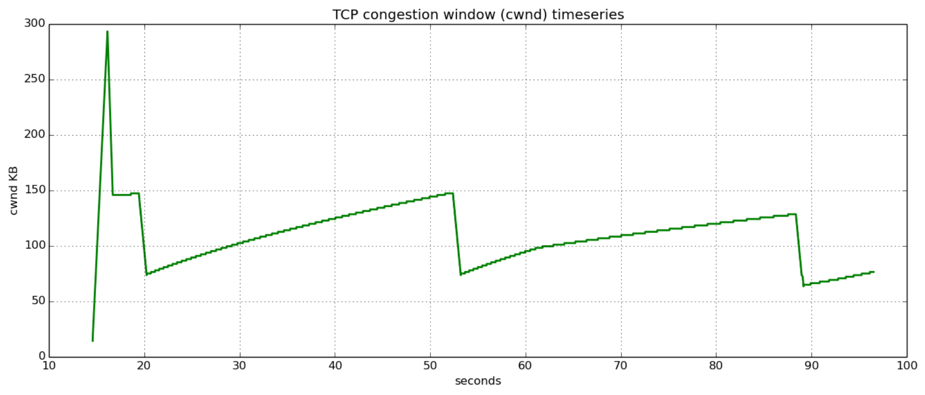
The web page download now takes 6.6 seconds. The reason for the increased delay is that the packets for the web flow need to share the router queues with the iperf flow traffic. Since the iperf flow generates significant traffic, the packets for the web flow are delayed behind the iperf packets leading to increased queueing delays. This in turn increases the total delay for the web download as compared with Question 6.

Question 9 (1.5 marks)

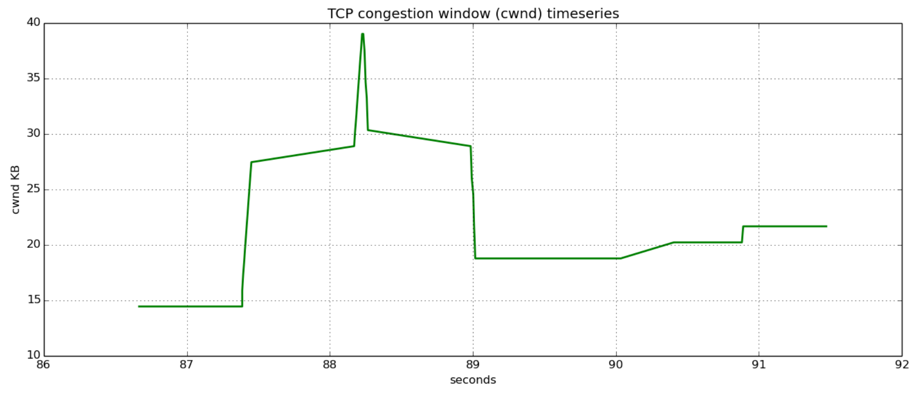
The following plot shows the buffer occupancy.



The following plot shows the cwnd for the long-lived iperf flow



The following plot shows the cwnd for the short-lived wget flow (note that the x-axis scale is different for this plot from the above)



For the first 85 seconds or so, only the long-lived iperf flow is active. Observe the initial slow-start phase, which eventually leads to a congestion event around 16 seconds (or so). This is also reflected by the router buffer occupancy being full. The flow also appears to encounter another congestion event around 20 seconds. This can be inferred by the fact that the cwnd is cut by half and the buffer occupancy is also maximum. Following this, the iperf flow enters congestion avoidance (AIMD) for the rest of the duration. One can readily observe the saw-tooth behavior.

The short-lived wget flow is initiated around 85 seconds. It goes through slow start (reflected in the cwnd plot) and encounters a congestion event just after 88 seconds. Observe a similar congestion event in the cwnd plot for the iperf flow at that time. Also observe that the buffer occupancy is maximum at that time. So both flows cut back their windows and enter congestion avoidance. However, since the wget flow is short-lived it quits within 2-3 seconds of entering this phase.

Question 10 (1 mark)

The average RTT is 20.863ms, which is similar to what was observed earlier. This makes sense because as it reflects the RTT of the underlying network when there is no traffic. This should not change by changing the buffer size in the router.

The webpage download takes 1.5 seconds. This is slightly longer than in Part 1. This is because with a shorter buffer, the cwnd for the wget flow won't increase to the same level as when the buffer size was 100 packets. Thus, the throughput for the TCP connection reduces which in turn increases the download time.

The RTT statistics when the long-lived iperf flow is running are as follows:

Min: 91.232 ms, Avg: 137.115 ms, Max: 170.668 ms, mdev: 20.94 ms

The RTT has increased considerably from 20ms since the ping packets are now queued behind the iperf packets. However, the RTT is much lower than in Part 2. The small buffer reduces the queueing delay significantly.

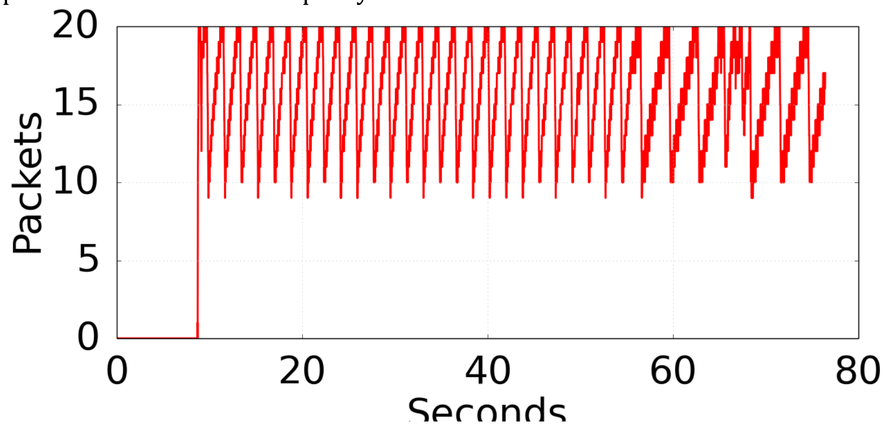
The webpage download now takes 2.9 second. The reason for the reduced delay (compared to Part 2) can again be attributed to the reduced queueing experienced due to the smaller buffer.

Question 11 (0.5 mark)

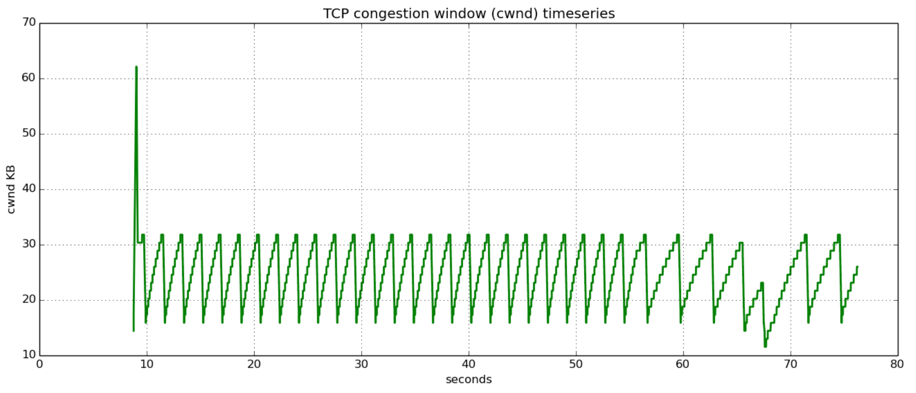
The webpage download now takes only 2.8 seconds. This is significantly shorter than the previous experiment. The reason can be attributed to the shorter buffer. The wget packets still share this short buffer with the iperf flow. However, this time around the queueing delay is shorter due to the reduced size of the buffer.

Question 12 (1.5 marks)

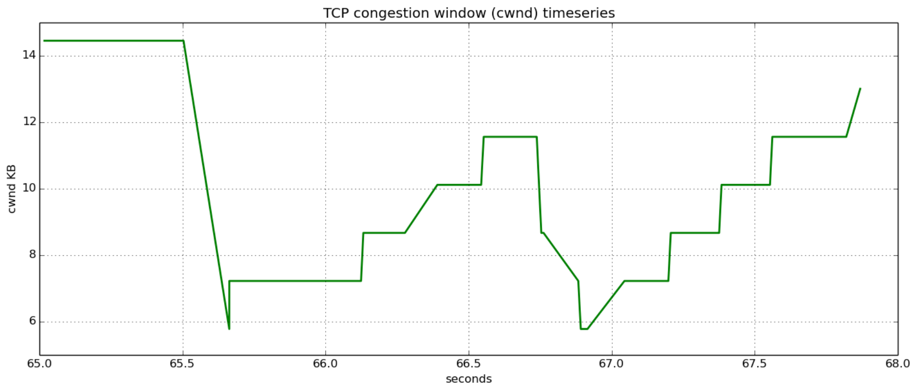
The following plot shows the buffer occupancy:



The following plot shows the cwnd for the long-lived iperf flow:



The following plot shows the cwnd for the short-lived wget flow (note that the x-axis scale is different for this plot from the above)



For the first 65 seconds or so, only the long-lived iperf flow is active. The cwnd behavior is similar to that in the previous experiment. The difference being that the size of the cwnd is much smaller. Observe the initial slow-start phase, which eventually leads to a congestion event around 9 seconds (or so). This is also reflected by the router buffer occupancy being full. Following this the flow enters congestion avoidance which is observed in the saw-tooth pattern. Notice that the cwnd consistently hits a peak of around 31-32 KB when it encounters a congestion event.

The wget flow is initiated around 65 seconds. It immediately encounters a congestion event and then transitions to congestion avoidance. Another congestion event is observed around 66.75 second which is followed by another congestion avoidance phase. The wget flow ends around 68 seconds. Also observe that the saw-tooth for the iperf flow shows a slightly different pattern during this time period, in that, the peak of the sawtooth during the 65-68 second period is smaller than when the wget flow is not active.

Question 13. (1 mark)

The RTT in the absence of traffic is still around 20ms.

The webpage is downloaded in 1 second as before.

Once the long-lived flow is initiated, the RTT is still around 20ms. This is because in this experiment the ping packets are directed to a separate queue and thus are not buffered behind the iperf packets. Moreover, since fair scheduling is used, the ping packets can essentially bypass the iperf packet. Thus the RTT is similar to that when there was no other traffic on the network.

The webpage download time is also 1 second. The reason is exactly as above. The wget packets are buffered in a separate queue and can thus completely bypass the iperf traffic.