CS252 Lab 6 - Comparing TCP variants + Wireshark

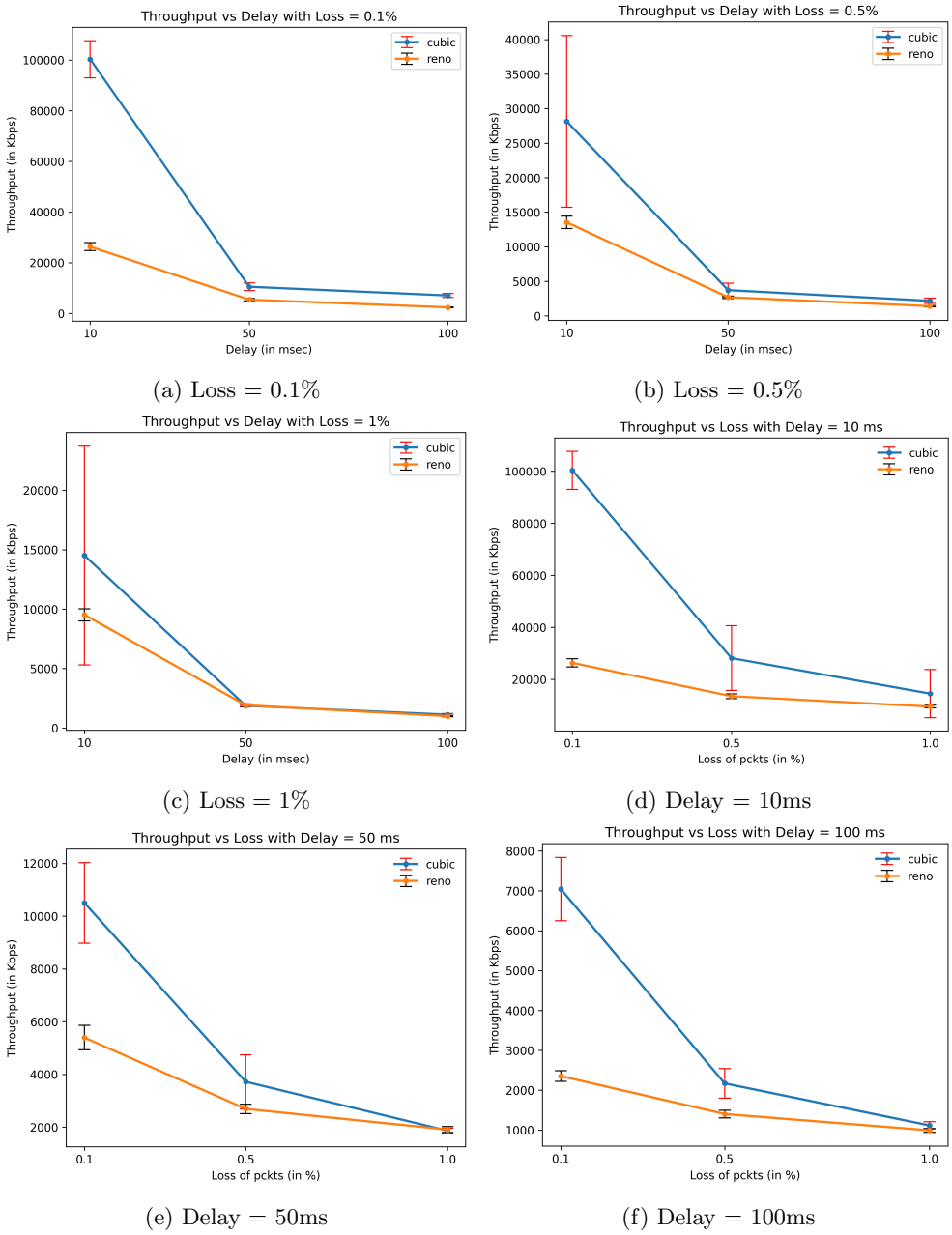
G Akash Reddy (190050038) - M Satwik (190050107) - A Kranthi (190050022)

May 8, 2021

Contents

1 Tcp-Cubic vs Tcp-Reno :

The following plots show the throughput obtained using TCP Cubic and TCP Reno vs delay(loss) keeping loss(delay) factor constant :



(a) Loss = 0.1%

(b) Loss = 0.5%

(c) Loss = 1%

(d) Delay = 10ms

(e) Delay = 50ms

(f) Delay = 100ms

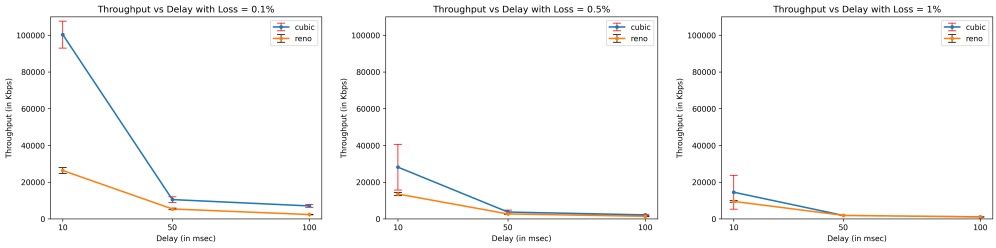
Figure 1: Throughput vs delay(loss) keeping loss(delay) constant. Note that here the y-axis scales are different for each graph.

Cubic is more aggressive than Reno. This is also **confirmed** by the above plots since we observe the throughput to be higher in the case of TCP-cubic everytime.

Also it is clear from the above plots that **as delay in transmission increases, throughput falls**, i.e. delay and throughput are inversly related. **As loss of packets increases, throughput falls**. This is a straight forward deduction since more packets are to be transmitted as more and more sent out packets are lost. we see that as conditions of transmission (delay and loss) get worse, both cubic and reno give same throughput. This implies that **TCP cubic behaves similar to TCP reno when transmission conditions are bad**.

To compare various situations easily, the following plots (essentially same as above) are drawn keeping the scale of y-axis same for all plots.

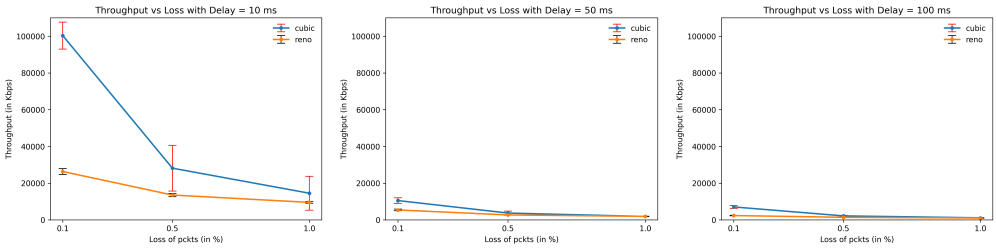
The below three plots show the case where loss of packets is kept constant, with values as 0.1%, 0.5%, 1%.



(a) Loss = 0.1% (b) Loss = 0.5% (c) Loss = 1%

Figure 2: Throughput vs delay keeping loss constant.

The below three show the case where delay in transmission is kept constant, with values as 10ms, 50ms, 100ms.

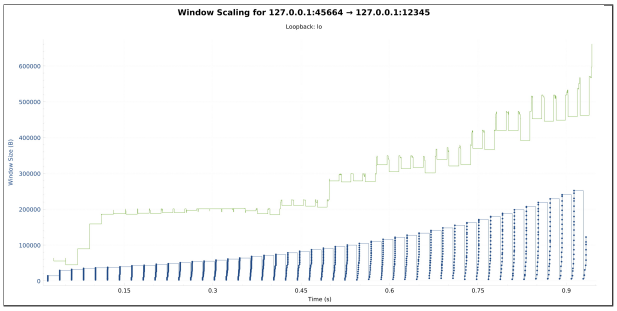


(a) Delay = 10ms (b) Delay = 50ms (c) Delay = 100ms

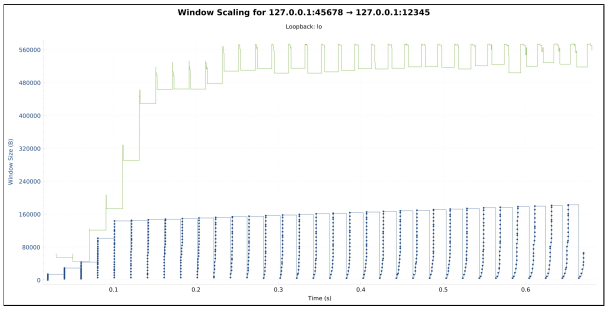
Figure 3: Throughput vs loss keeping delay constant.

2 Window Scaling Graphs :

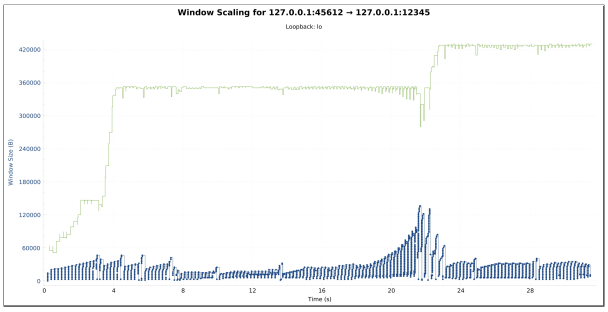
Using wireshark, the following window scaling graphs were obtained :



(a) TCP cubic with delay 10ms and loss 0.1%



(b) TCP reno with delay 10ms and loss 0.1%

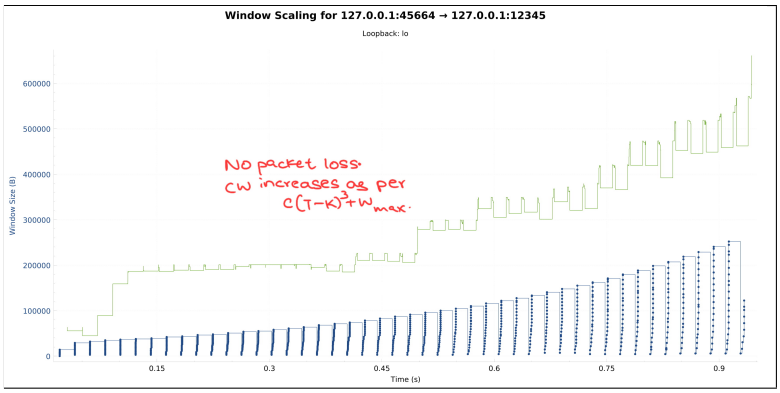


(c) TCP cubic with delay 100ms and loss 1%

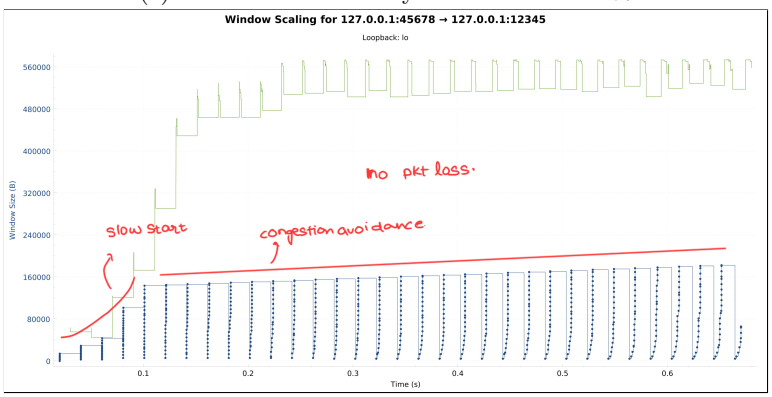
(d) TCP reno with delay 100ms and loss 1%

Figure 4: Window scaling graphs.

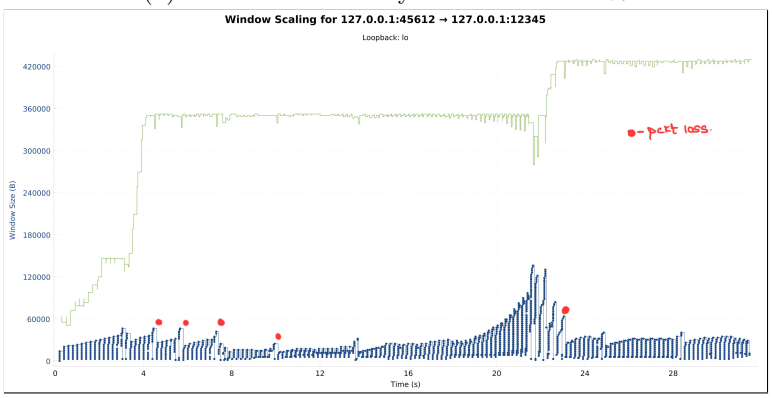
And below are the annotated graphs in which reno's slow start and congestion avoidance is shown. **Packet loss can be inferred by sudden fall in the congestion window size.** In reno, if RTO expiry happens, then CW falls to 1MSS, but when 3 Dup. Acks are detected, CW falls to half of its previous value. In cubic, whenever packet loss is infered, CW falls by a constant factor.



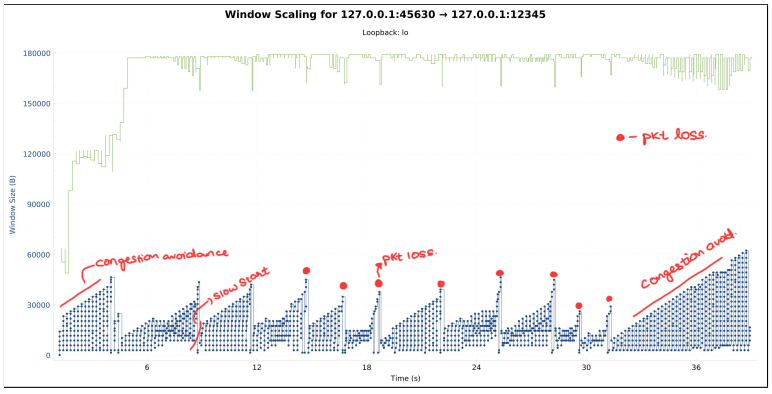
(a) TCP cubic with delay 10ms and loss 0.1%



(b) TCP reno with delay 10ms and loss 0.1%



(c) TCP cubic with delay 100ms and loss 1%



(d) TCP reno with delay 100ms and loss 1%

Figure 5: Window scaling graphs - annotated.