Comp9331

Lab05

Changyue Tan

Z5325184

Revision of TCP Congestion Control

Question 1:

- Loss events that occur at time 1 is dupACK, which set CWND /= 2, indicating network capable of delivering some segments, take Multiplicative Decrease.
- Loss events that occur at time 2 is timeout, which sets cwnd to 1, as not enough ACKs is suggesting serious congestion, start from 1 to avoid Congestion Collapse

Question 2:

- Slow Start, Bandwidth discovery, double cwnd every RTT, cwnd += 1 for each ACK.

Question 3:

- Additive increase: For each successful RTT (all ACKs), cwnd = cwnd +1.

Question 4:

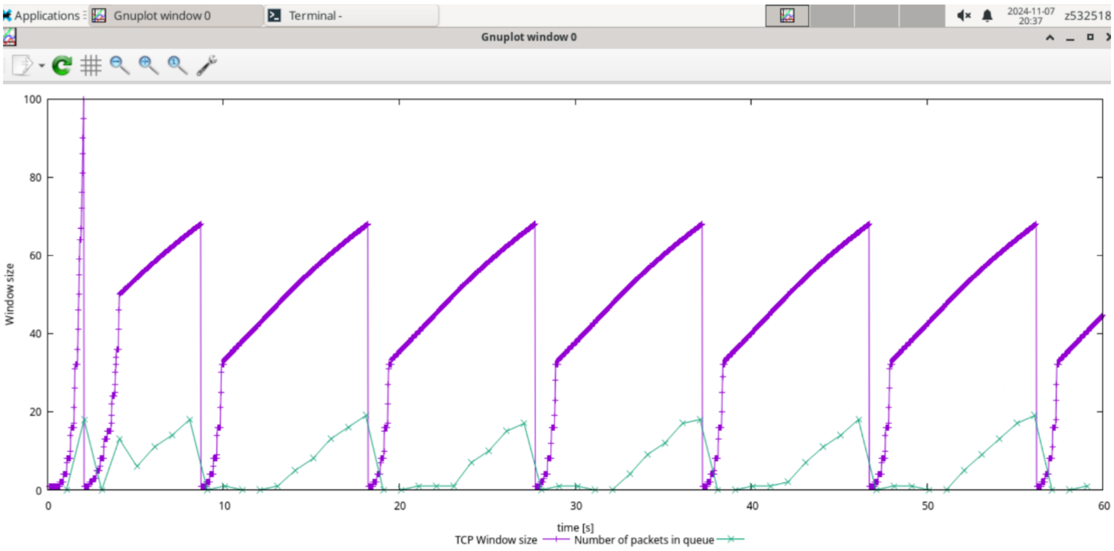
- Because the host is just starting to send data and no congestion was detected, hance we want to discover what is the largest available bandwidth quickly.

Question 5:

- Cwn was set to 1, and Slow Start Threshold was set to cwn/2. Beginning slow start.

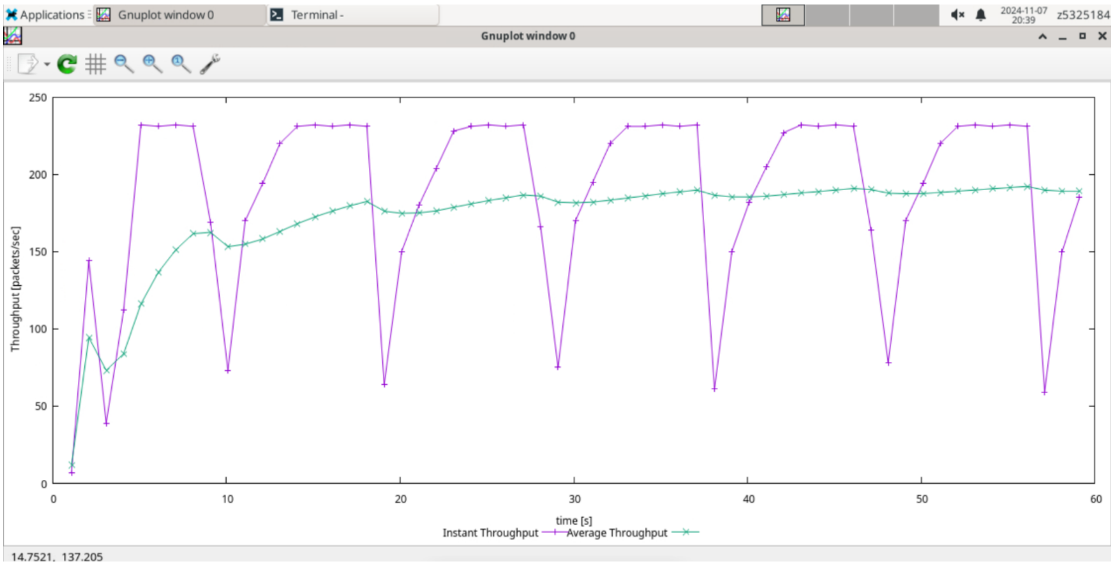
Exercise 1: Understanding TCP Congestion Control using ns-2

Question 1:



a) The maximum size of the congestion window reached is 100
b) When the congestion window reaches 100 the cwn was reset to 1, indicating a timeout. And slow start threshold is set to half of the cwn at the time it cwn was rest to 1, which is 100/2 = 50.
c) Next, a new slow start begins, which an exponential growth function. After cwn reaches the slow start threshold, it transitioned into additive increase, which is a linear growth function. This additive increase continuous until another timeout occurs (or a dupACK occurs), and cwn will be reset to 1, ssthresh will be set to half of cwn again.

Question 2:



From the plot we can see that the average throughput is:

around 180 packets per second.

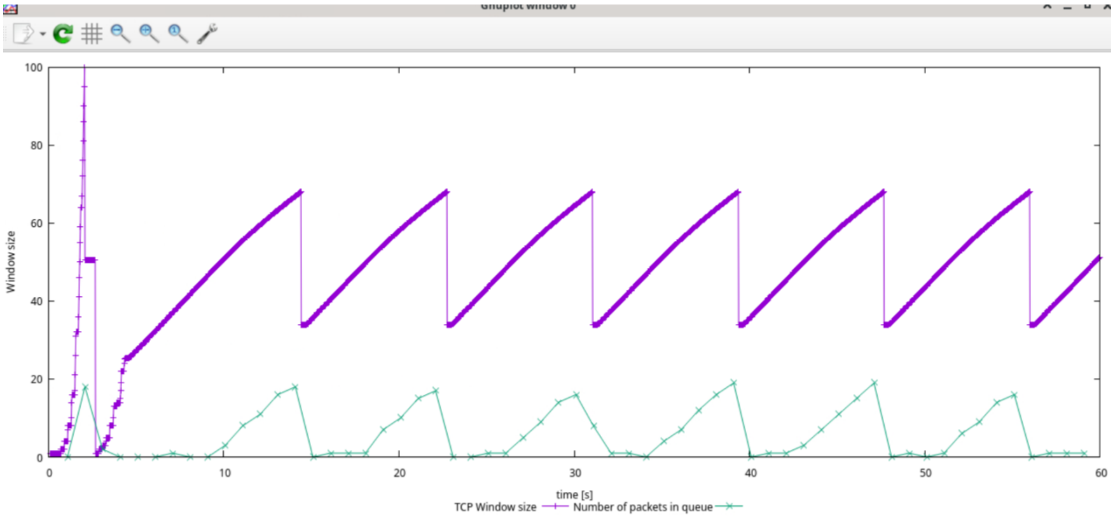
Considering the payload only, the throughput is:

180 * 500 = 90000 bytes per second

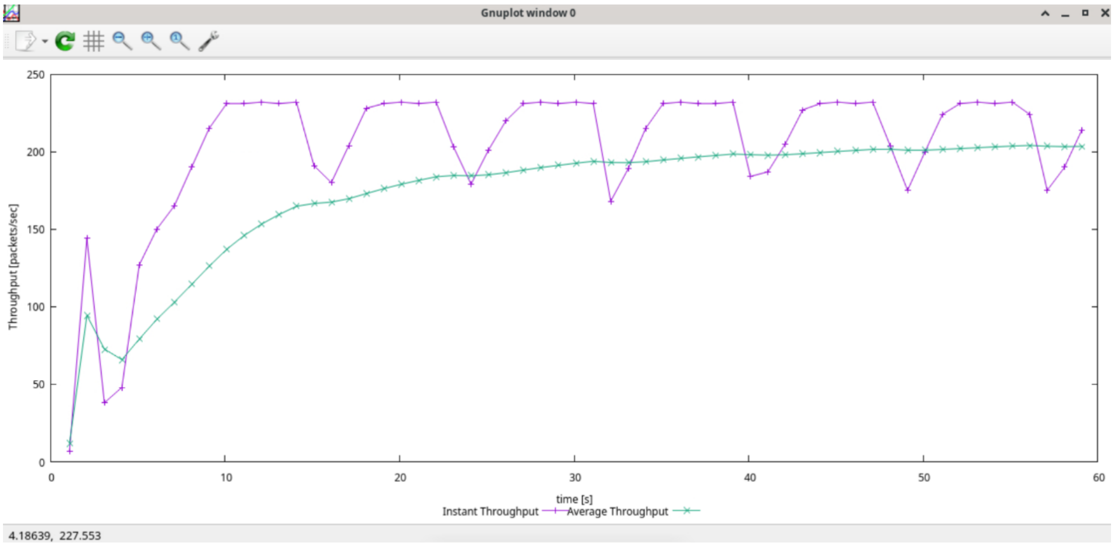
Considering the payload plus the TCP and IP header sizes, the throughput is:

180 * (500 + 20 + 20) = 97200 bytes per second

Question 3:

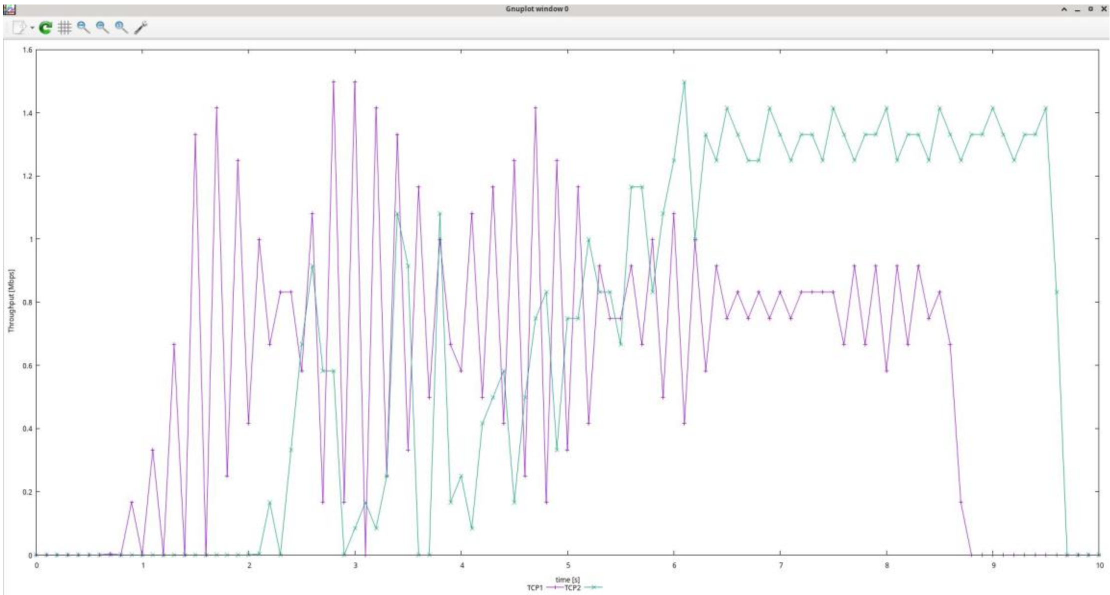


The congestion window only returns to one once. TCP Reno begins slow start, reaches a cwn of 100, timeout occurs, rest cwn to 1 and ssthresh to 50, begins slow start again, reaches ssthresh of 50 and switched to additive increase, just as in TCP Tahoe. However, whenever a dupACK occurs, in TCP Reno, cwn was cut by half, from around 70 to 35, instead of resting to 1, as in TCP Tahoe. Additionally, TCP Reno continuous with additive increase after cutting cwn by half, unlike in TCP Tahoe, changing back to slow start after resting cwn to 1.



The average throughput of TCP Reno is higher than TCP Tahoe. For TCP Reno, average throughput reached to above 200 packets per second, while for TCP Tahoe, the average throughput was only around 180 packets per second. This likely because the cwn was kept at a relatively high level for TCP Reno, as it does not reset cwn to 1 after dupACK, it makes more use of the available bandwidth to transmit packets, hence, higher throughput.

Exercise 2: Setting up NS2 simulation for measuring TCP throughput (3.5 marks)



Question 1:

The throughput achieved by flow tcp2 higher than tcp1 between 6 sec to 8 sec because the link from n1 to n2 is congested during this period of time, and packets are dropped due to buffer overflow at n1. For tcp1, the fastest way for it to send packets from n0 to n5 is through this link n1->n2 (the simulation chose this path over from n6->n4 randomly). For tcp2, it does not use this n1->n2 link, hence is not affected by the congestion at this link. However, n1->2 link is also shared by tcp4, as it sends packets from n3 to n0. After tcp4 starts sending packets, buffer at n1 was slowly filling up from time 4 to time 6, starting a congestion, forcing packets at n1 to be dropped

Question 2:

The throughput for tcp1 fluctuates between a time span of 0.5 sec to 2 sec because it is doing bandwidth discovery, similar to a slow start if there was congestion control. It first sends a small number of packets, wait to see if it can receive ACKs for all of them, and then try to send more packets, wait for ACKs again before sending more packets. The number of packets sent in each round is increasing so that it can discover the available bandwidth quickly.

Exercise 3: Understanding the Impact of Network Dynamics on Routing

Question 1:

Node 0 is sending packets to Node 5, path: n0->n1->n4->n5

Node 2 is sending packets to Node 5, path: n2->n3->n5

The routing does not change over the duration of the simulation

Question 2:

The link between n1 and n4 was blocked during time 1.0 to time 1.2. The route of communication between n2 and n5 was unchanged, as link n1->n4 was not used by this communication. However, link n1->n4 was used by the communication between n0 and n5. The packets sent from n0 was unable to reach beyond n1 between time 1.0 and time 1.2. No alternative routing was used by this communication; hence the communication was interrupted.

Question 3:

Yes, there are additional traffic, they are nodes communicating their Distance vectors with neighbouring nodes to neighbouring nodes.

After the link n1->n4 was blocked, packets from n0 to n5 was routed to new path n0->n1->n3->n5. Hence the communication from n0 to n5 was uninterrupted by the shutdown of link n1->n4.

Question 4:

The change made to the simulation was changed the link cost of n1->n4 to 3. Hence the routing from n0 to n5, even with link n1->n4 available, will be changed to n0->n1->n3->n5, as this path have less total link cost comparing to use the link n1->n4, which now has a link cost of 3.

Question 5:

The routing from n0 to n5 is still n0->n1->n4->n5

The routing from n2 to n5 is either n2->n3->n5 or n2->n1->n4->n5, which have the same total link cost of 4 now, as link cost of n1->n4 is 2, and n3->5 is 3.

The line: "Node set multiPath_ 1" is probably used to enable a communication to take multiple paths. As in the simulation, the communication from n2 to n5 uses path n2->n3->n5 before changing to path n2->n1->n4->n5. And then it switches back and forth between these two paths. Given that these two paths have the same link cost, the reason for switching back and forth is possibly to balance the load of traffic evenly to avoid congestion.