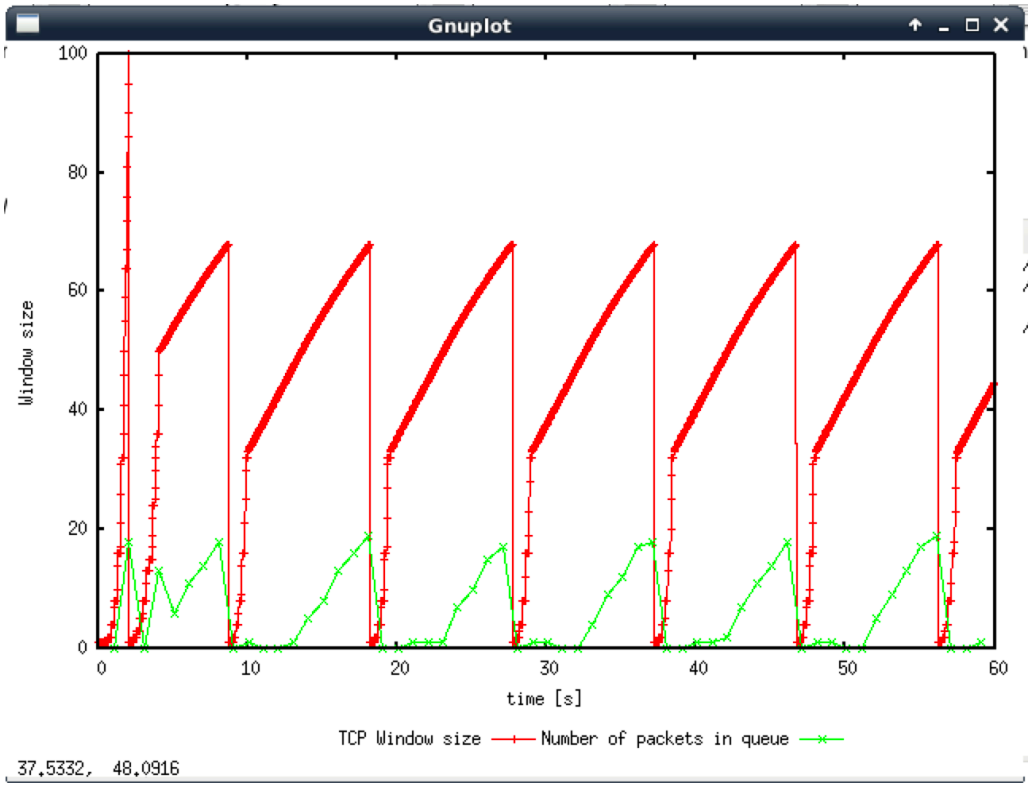
Exercise 1: Understanding TCP Congestion Control using ns-2

Question1: What is the maximum size of the congestion window that the TCP flow reaches in this case? What does the TCP flow do when the congestion window reaches this value? Why? What happens next? Include the graph in your submission report.

- The following graph shows the congestion window and number of packets in the queue as a function of time.



- Although the maximum size of the congestion window is 150 packets, the congestion window grows up to **100** packets during the slow start phase. This is because the maximum size of queue is 20 packets and any additional packets would be dropped.

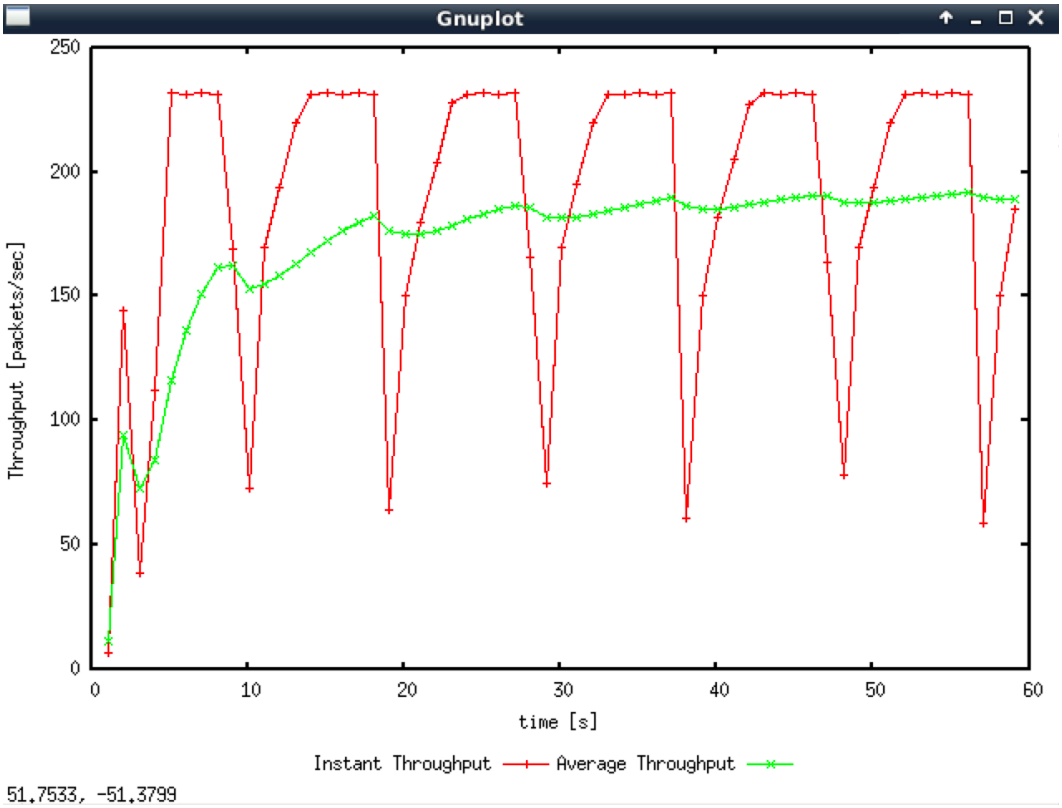
- Thus, when the window is ramping up and the queue becomes full, packets would be dropped. This might cause the congestion event. There are two things that could actually cause the congestion event, it is either timeout or triple duplicate Acts.

- So, the sender needs to fix this situation by decreasing the congestion window to 1 and threshold to 0.5 the size of the window. The connection enters slow start and ramps up the window rapidly until it hits the threshold and then this connection transitions to congestion avoidance phase.

- Finally, the queue becomes full again and hence packet get loss. The connection transitions back to slow stage again.

Question2: What is the average throughput of TCP in this case? (both in number of packets per second and bps)

- The following graph shows the the instantaneous and average throughput in packets/sec for the TCP connection.



The average throughput of TCP in packets/sec is 190.

As we know, payload is 500 bytes. IP, TCP header is 20 bytes(neglect other headers).

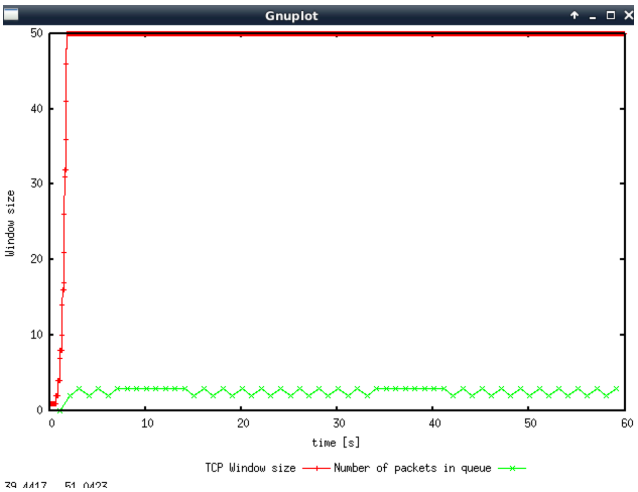
There is something that we need to consider: the header is included in the data or not. So, I've done 2 calculations:

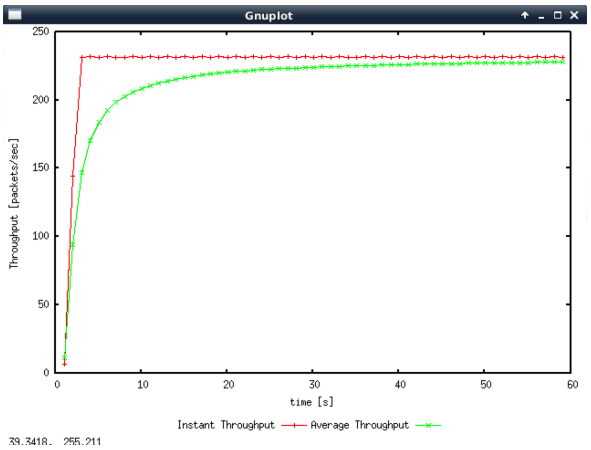
1) Include header, the throughput is 190 * (500 + 40) * 8 = 802.8 kbps

2) Not included header, the throughput is 190 * 500 * 8 = 760 kbps

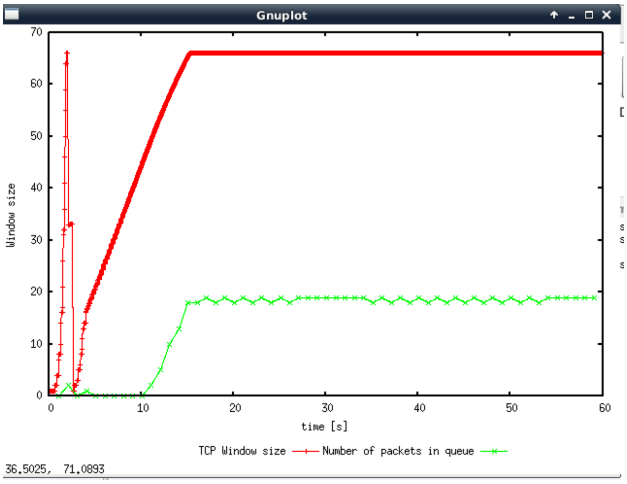
Question 3: Rerun the above script, each time with different values for the max congestion window size but the same RTT (i.e. 100ms). How does TCP respond to the variation of this parameter? Find the value of the maximum congestion window at which TCP stops oscillating (i.e., does not move up and down again) to reach a stable behaviour. What is the average throughput (in packets and bps) at this point? How does the actual average throughput compare to the link capacity (1Mbps)?

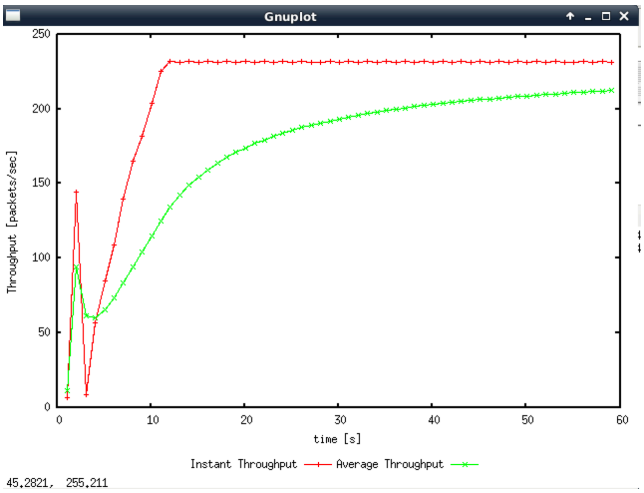
- The following graphs show **the window size** and the **throughput** when the initial max congestion is set to **50**.





- The following graphs show **the window size** and the **throughput** when the initial max congestion is set to **66**.





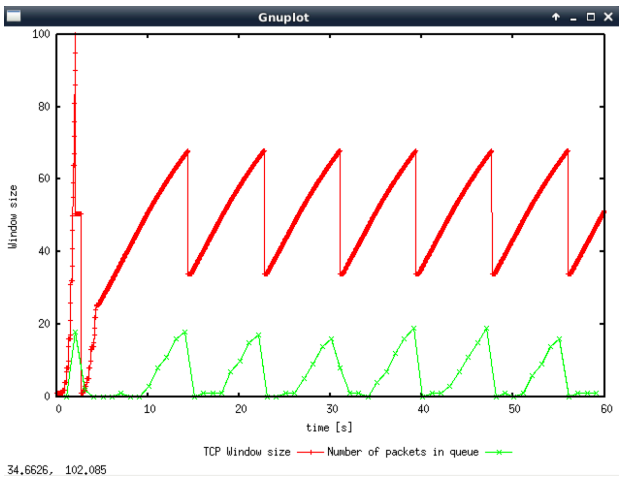
- When the maximum congestion window size becomes greater than the maximum queue size, some packets will get dropped and TCP will go back to slow start.

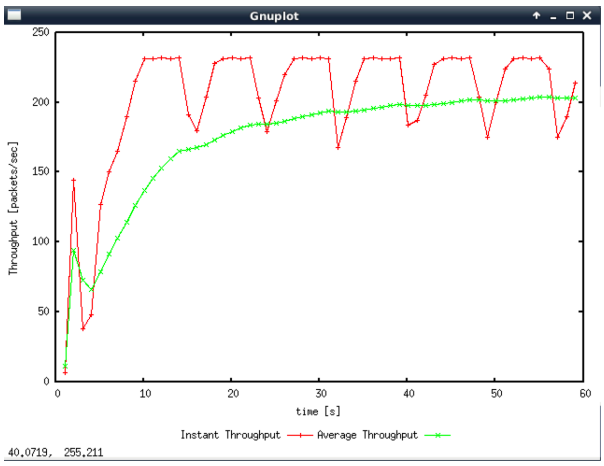
- When the initial maximum congestion window is set to be less than **50** packets: TCP stabilises immediately. The average throughput is close to 225 packets/sec at this point. And hence the average throughout is 225*500*8 = 900 Kbps, which is almost equal to 1 Mbps.

- When the initial maximum congestion window is set to be less than **66** packets: the oscillation stop after the first return to slow start. When reduce the congestion window to half its size, this is enough to stabilise the number of packets in the sending queue; the queue never gets full, and so packets are not dropped anymore.

- **Question 4:** Repeat the steps outlined in Question 1 and 2 (NOT Question 3) but for TCP Reno. Compare the graphs for the two implementations and explain the differences. (Hint: compare the number of times the congestion window goes back to zero in each case). How does the average throughput differ in both implementations?

- The following graphs show **the window size(left graph)** and the **throughput(right graph)** when the initial max congestion is set to 150 for TCP Reno .





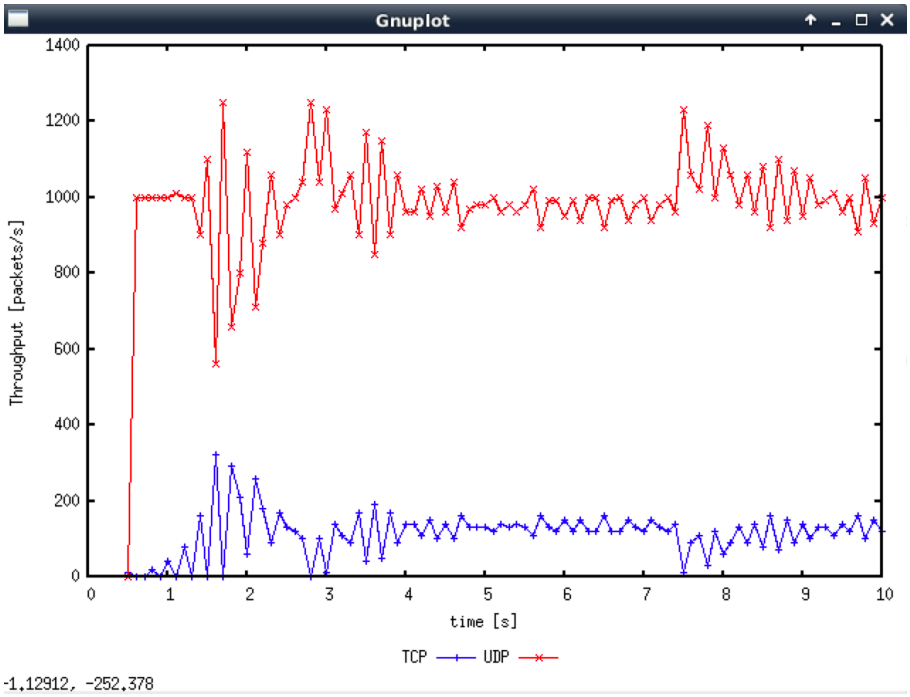
- **Window size graph:** For the most part, the TCP connection does not enter slow start. Instead, the sender reduces its current congestion window to half and increase it linearly, until losses starts taking place again. This pattern repeats. This implies that most of the losses are detected due to triple duplicate ACKs. This behaviour is different to TCP Tahoe where the window is reduced to 1 after each congestion event.

- **Throughput graph:** The throughout of TCP Reno is around 200 packets/sec which is slightly higher than TCP Tahoe(which was 190 packets). This is because TCP Reno does not have to initiate slow start after each congestion event.

Exercise 3: TCP competing with UDP

Question 1: How do you expect the TCP flow and the UDP flow to behave if the capacity of the link is 5 Mbps ?

- The following graph shows the throughput for the two flows.



- UDP does not have any control congestion unlike TCP. This means that a UDP flow will not reduce its transmission rate in response to congestion. Hence, we would except that UDP would continue to transmit at its scheduled rate which TCP will try to settle in the leftover capacity.

Question 2: Why does one flow achieve higher throughput than the other? Try to explain what mechanisms force the two flows to stabilise to the observed throughput.

- As the above graph shows, UDP achieve higher throughput than TCP, because it is not restrained by congestion control. UDP transmits packets at a constant rate, regardless of whether any of them get dropped. However, for TCP connection, it will decrease its transmission rate when it detects congestion in the network.

Question 3: List the advantages and the disadvantages of using UDP instead of TCP for a file transfer, when our connection has to compete with other flows for the same link. What would happen if everybody started using UDP instead of TCP for that same reason?

- **Advantages:** the sender could keep transmitting unrestrained, irrespective of congestion. This may potentially reduce the delay for transferring files. **Disadvantage:** the file transfer protocol running on UDP would need to implement reliable data transfer as UDP does not provide this service.

- If everybody started using UDP instead of TCP, then if faced with congestion, none of the flows would throttle their sending rate. Thus, there is a high possibility that parts of the internet would encounter congestion collapse.