Exercise 1: Understanding TCP Congestion Control using ns-2

Question 1: Run the script with the max initial window size set to 150 packets and the delay set to 100ms (be sure to type "ms" after 100). In other words, type the following:

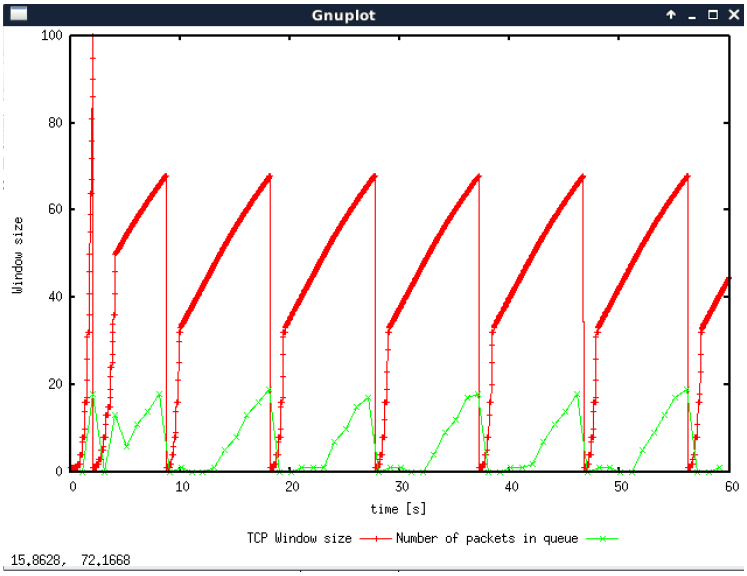
```
$ns tpWindow.tcl 150 100ms
```

In order to plot the size of the TCP window and the number of queued packets, we use the provided gnuplot script Window.plot as follows:

```
$gnuplot Window.plot
```

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.

Here is the graph plotted.



The maximum size of the congestion window that the TCP flow reaches in this case is 100. After that, a congestion event happens. Some packets are dropped when the queue is full. Then the window size drops to 1 and the threshold equals to CWND/2. The TCP then enter the slow start phase.

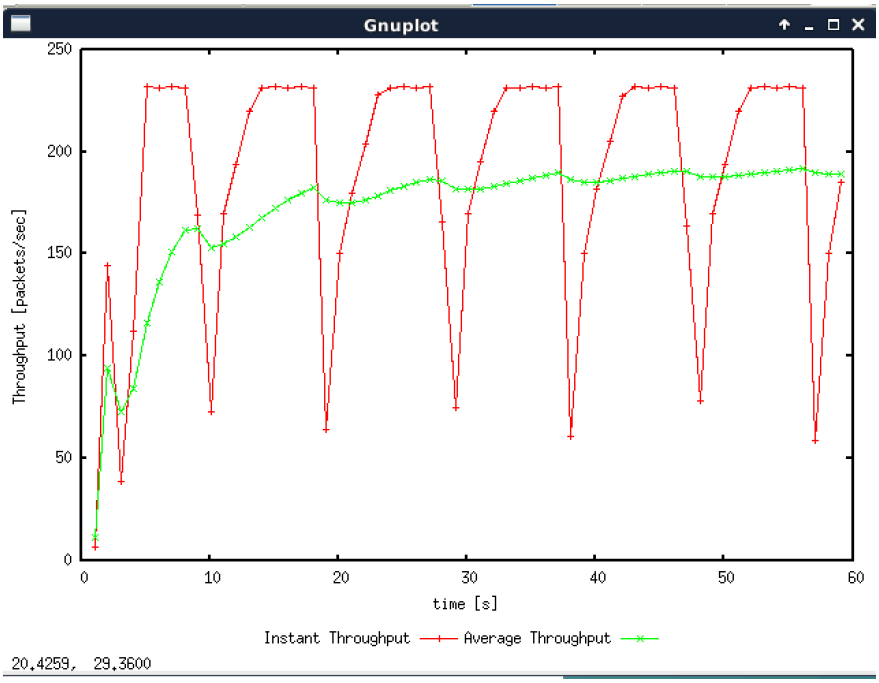
Question 2: From the simulation script we used, we know that the payload of the packet is 500 Bytes. Keep in mind that the size of the IP and TCP headers is 20 Bytes, each. Neglect any other headers. What is the average throughput of TCP in this case? (both in number of packets per second and bps)

You can plot the throughput using the provided gnuplot script WindowTPut.plot as follows:

```
$gnuplot WindowTPut.plot
```

This will create a graph that plots the instantaneous and average throughput in packets/sec. Include the graph in your submission report.

Here is the plotted graph



The average output should be about 190 packets/sec. In form of bps,
190 * (500 + 20 * 2) * 8= 820800 bps

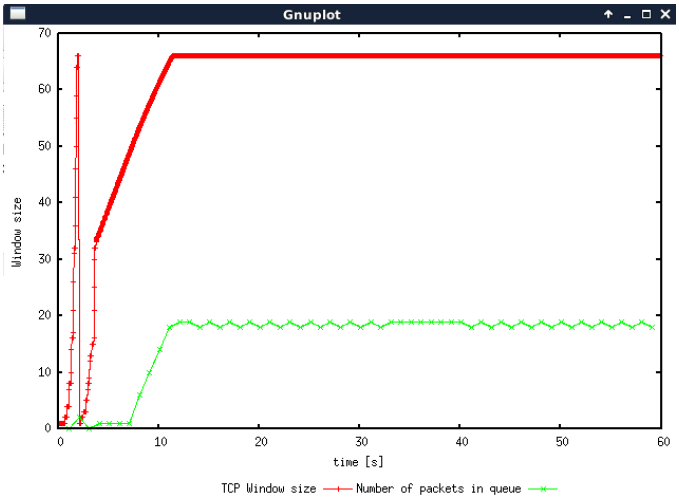
Question 3: Rerun the above script, each time with different values for the max congestion window size but the same RTT (i.e. 100ms). How does TCP respond to the

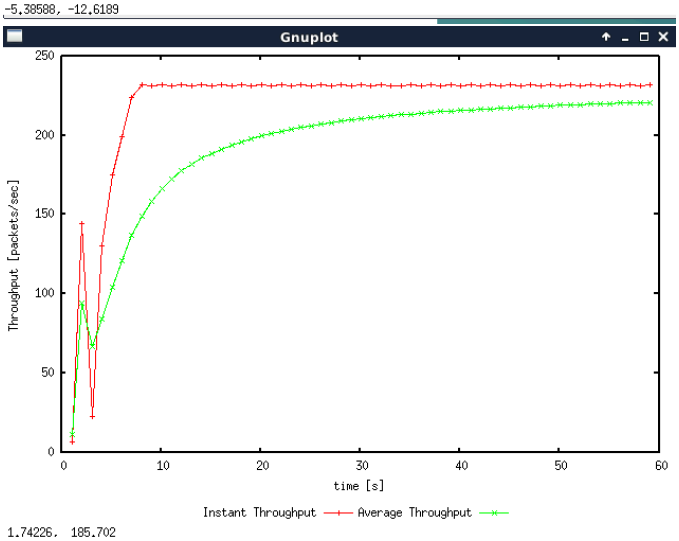
variation of this parameter? Find the value of the maximum congestion window at which TCP stops oscillating (i.e., does not move up and down again) to reach a stable behaviour. What is the average throughput (in packets and bps) at this point? How does the actual average throughput compare to the link capacity (1Mbps)?

TCP Tahoe vs TCP Reno

Recall that, so far we have observed the behaviour of TCP Tahoe. Let us now observe the difference with TCP Reno. As you may recall, in TCP Reno, the sender will cut the window size to 1/2 its current size if it receives three duplicate ACKs. The default version of TCP in ns-2 is TCP Tahoe. To change to TCP Reno, modify the Window.tcl OTcl script. Look for the following line:

The maximum congestion window is 66. Here are the plotted graphs:



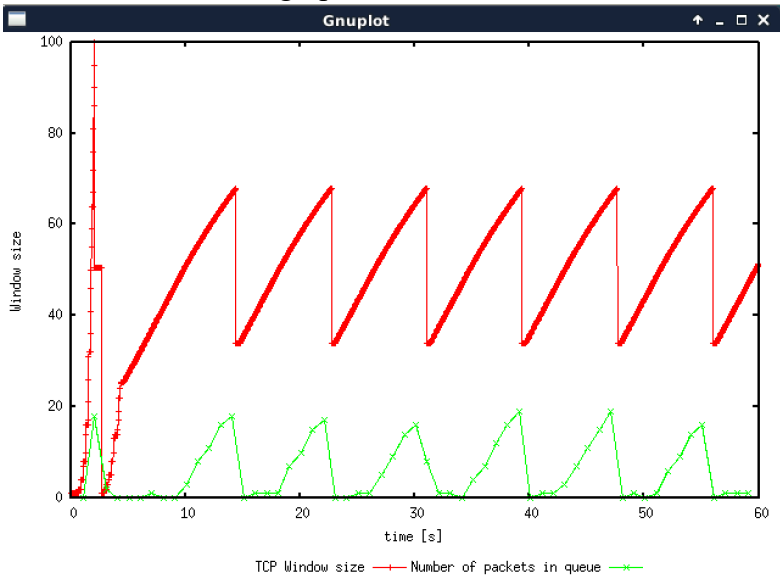


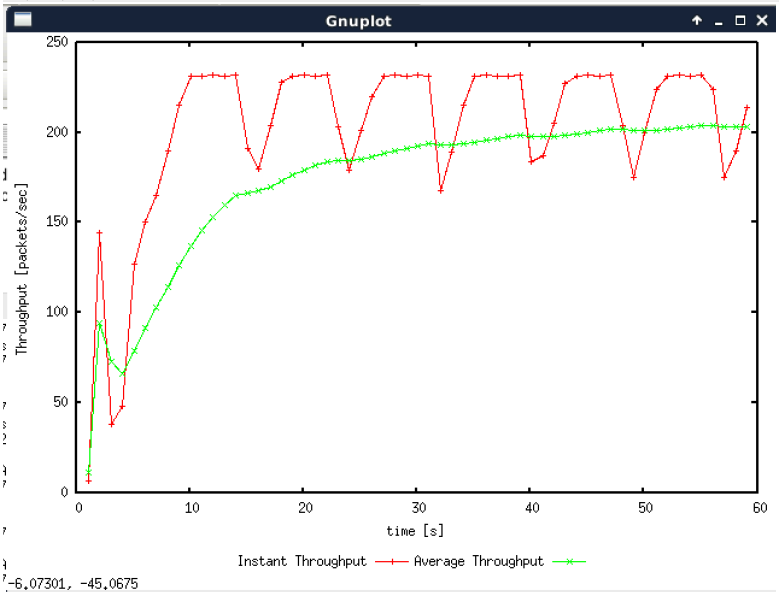
The average output at this point is 220 packets/sec. In terms of bps, that is:
220 * 8 *540 = 950400 bps, about 950.4 kbps, slightly smaller than the link capacity

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?

In Reno, here are the graphs:

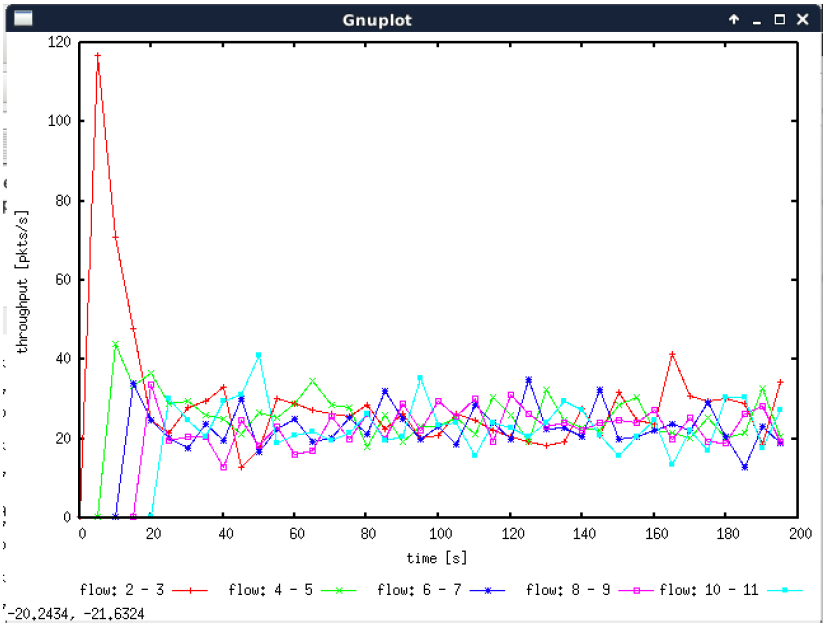




For Reno, the congestion windows drop back to 0 only once, that is when the first congestion event occurs. However, for Tahoe, every time when there are Triple ACK or packets drop, the congestion windows drop to 0. For average throughput, it is about 200 packet per second, slightly higher than that of Tahoe

Exercise 2: Flow Fairness with TCP

Question 1: Does each flow get an equal share of the capacity of the common link (i.e., is TCP fair) ? Explain which observations lead you to this conclusion.



Yes, each flow get an equal share of the capacity of the common link. From the graph, I notice that the line of flows merge to a close value after 40 seconds.

Question 2. What happens to the throughput of the pre-existing TCP flows when a new flow is created? Explain the mechanisms of TCP which contribute to this behaviour. Argue about whether you consider this behaviour to be fair or unfair.

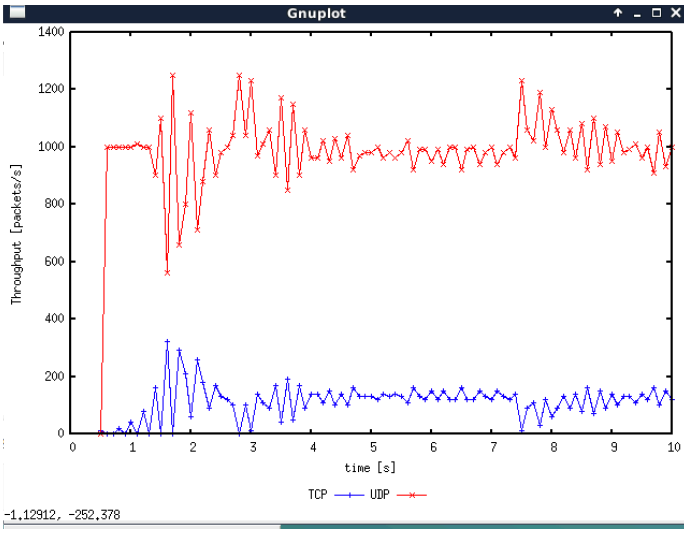
The throughput of all pre-existing flows will reduce when a new flow enters the network. The congestion window of new flow would increase exponentially at the beginning and that lead to congestion. Because of it, other flows would meet congestion in the network and then adjust their own CWND. I think it is fair enough, because each flow can get a similar share of the link

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 throughput of UDP flow should be much higher than TCP flow. And because UCP doesn't need to build connection, the beginning phase should grow much faster.

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.



The red line represent the UDP while the blue one represent the TCP. The reason why the throughput of UDP is greater than TCP is that its non-existent acknowledge packet (ACK) that permits a continuous packet stream, instead of TCP that acknowledges a set of packets, calculated by using the TCP window size and round-trip time (RTT). As a result, the UDP stay at a higher level in the graph.

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?

Advantage:
1. the speed of UDP is higher
2. the header of UDP is smaller. That helps to save space.
3. UDP doesn't need to build a connection with the other end.
4. the utilization of network link is higher

Disadvantage:
1. UDP is unreliability
2. UDP doesn't provide congestion control

If everyone start to use UDP, as it doesn't provide congestion control, the network link may be more and more congested.