Lab Exercise 5: TCP Congestion Control and Fairness

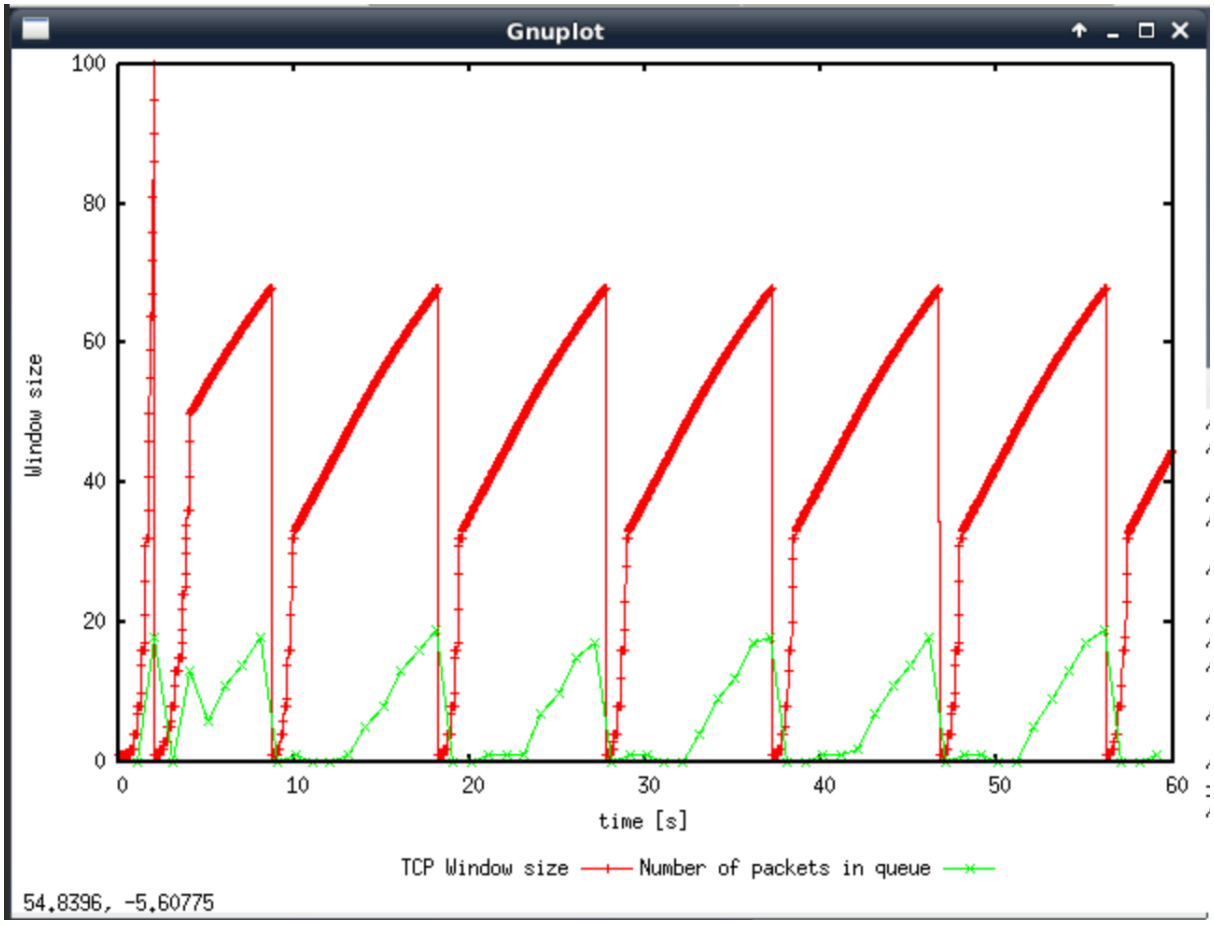
Exercise 1: Understanding TCP Congestion Control using ns-2

Question 1 : 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.

Solution 1 :

Commands : $ns tpWindow.tcl 150 100ms and $gnuplot Window.plot gives us the following output (Figure 1).

Figure 1 : The gnuplot for 150 packets at 100ms delay



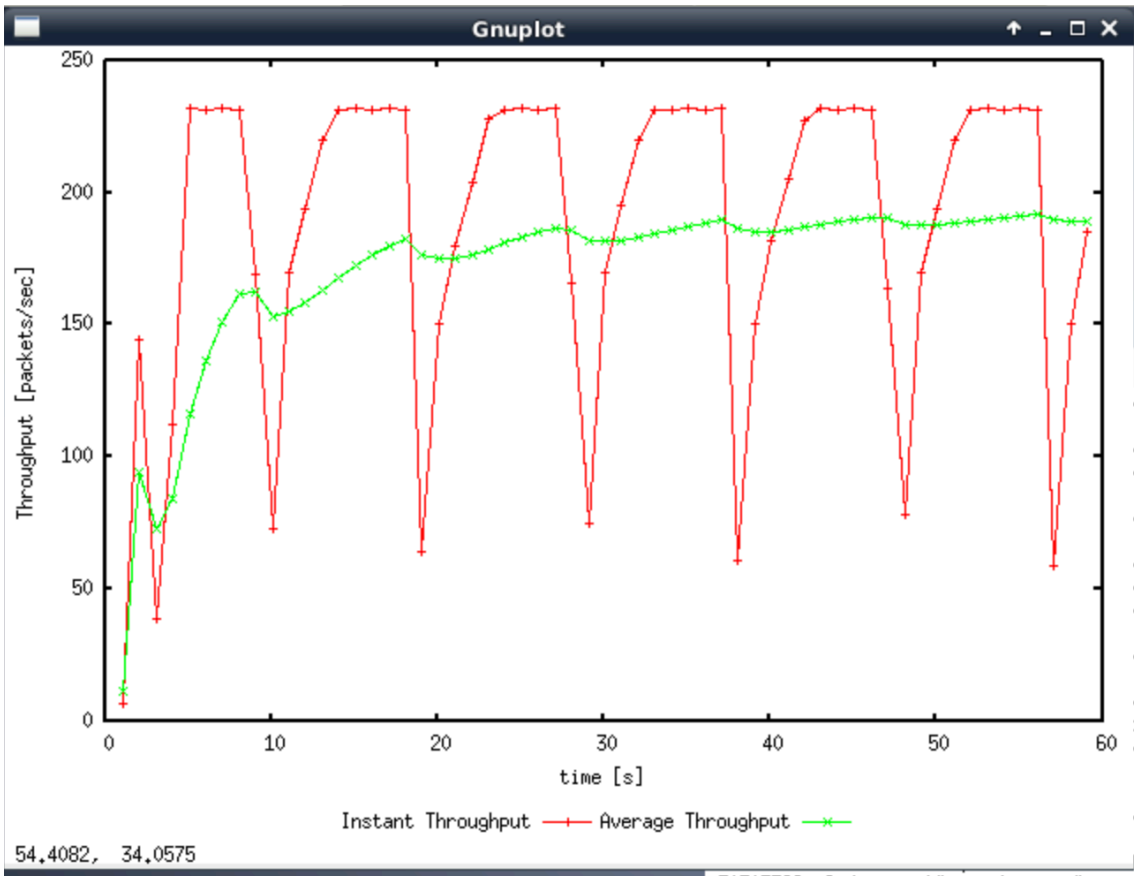
We can observe from the graph that the maximum size of congestion (y-axis) flow in this case reaches to 100 packets. Some packets are dropped because the 100 packet congestion is greater than the size of the queue i.e. 20 packets (green lines).When the congestion window reaches this value 100 ,it will keep alternating back to the slow start phase from packet congestion / dropping of packets.

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

Solution 2 :

Commands : $ns tpWindow.tcl 150 100ms and $gnuplot WindowTPut.plot gives us the following output (Figure 2).

Figure 2 : The gnuplot for 150 packets at 100ms delay



Packets per second throughput(pps) = ~190 pps (from observations) IP

TCP Headers = 20 + 20 = 40 bytes
Payload of the packet = 500 Bytes
Packets per second throughput(bps) = (500+40)*8*190

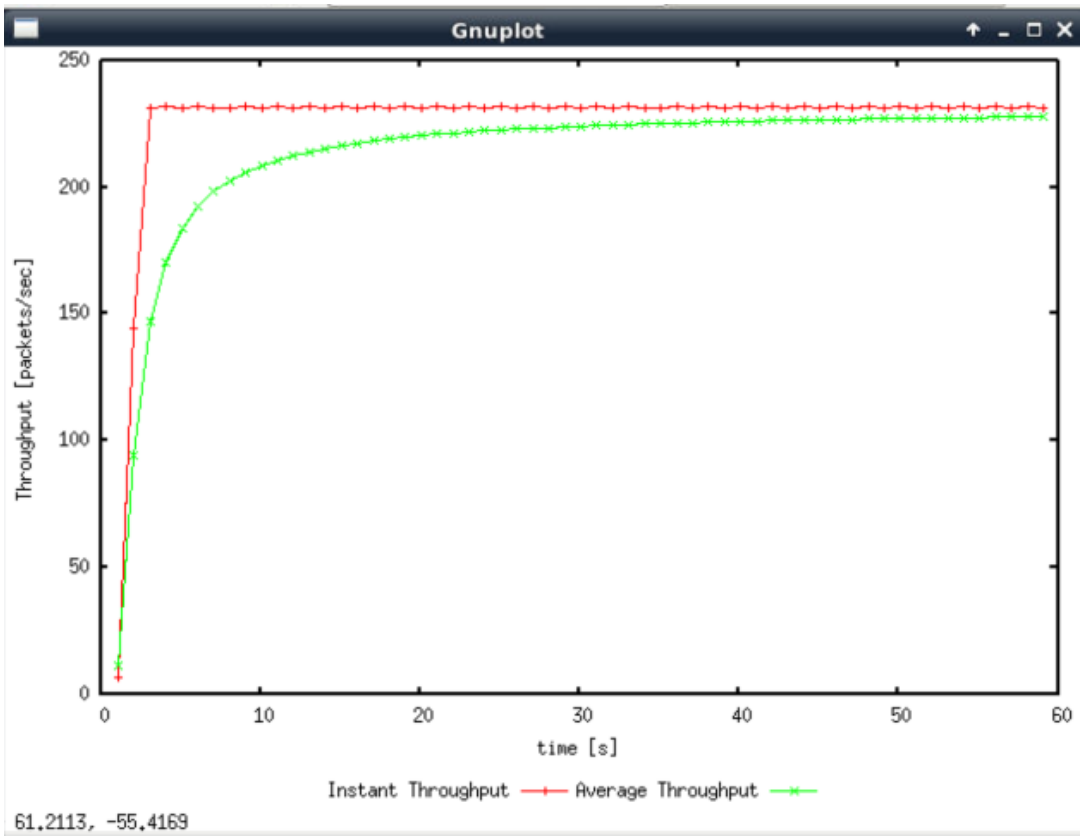
= 820,800 bps

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)?

Solution 3 :

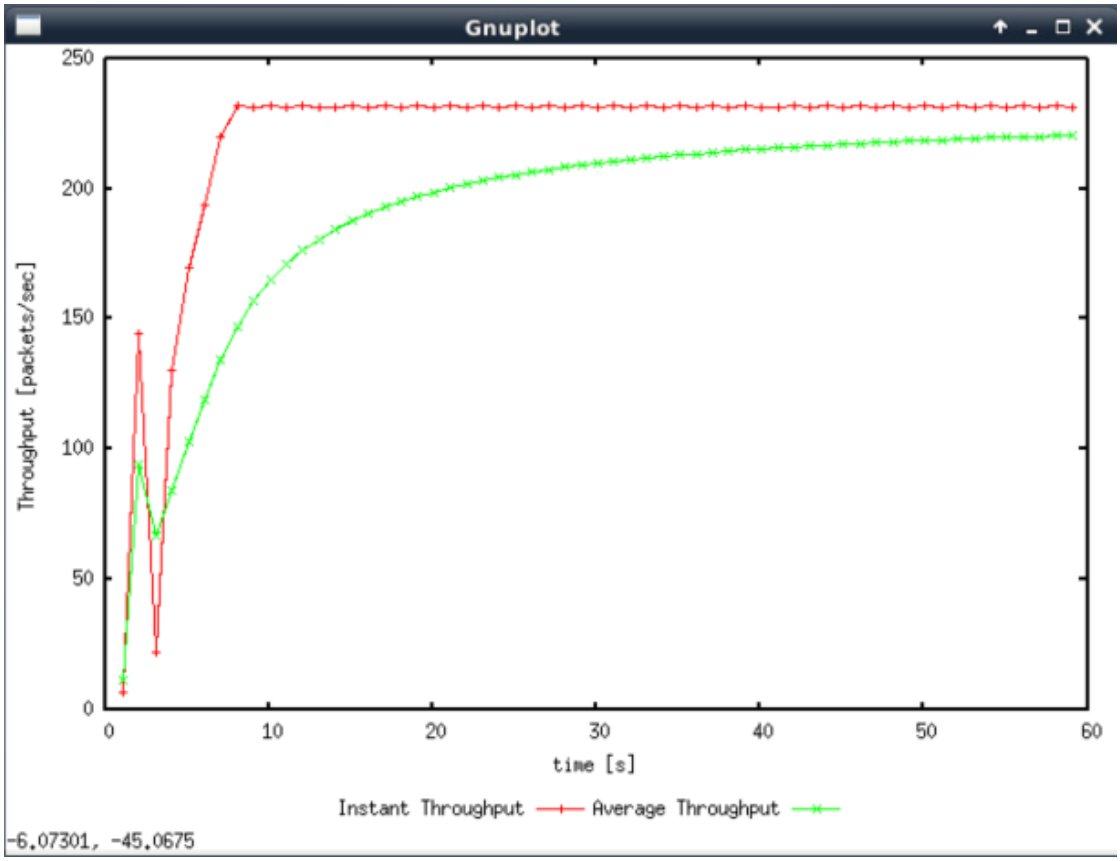
$ns tpWindow.tcl 50 100ms

$gnuplot WindowTPut.plot

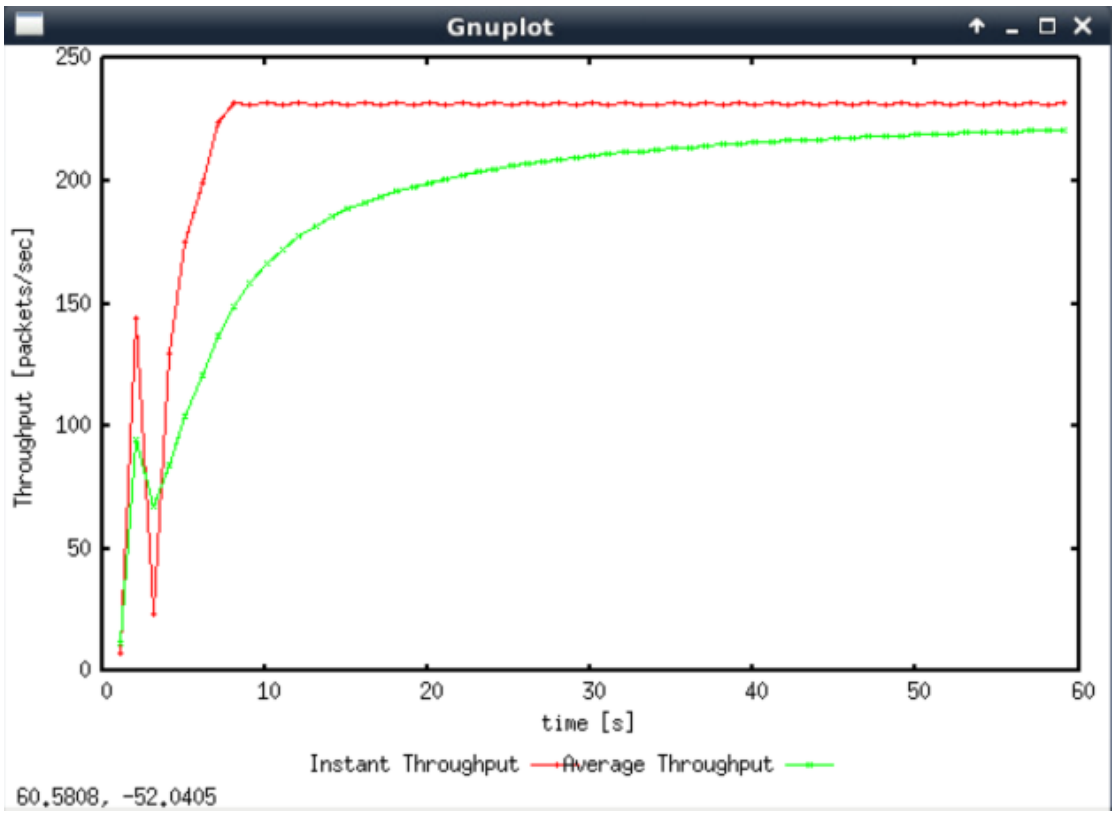


$ns tpWindow.tcl 65 100ms

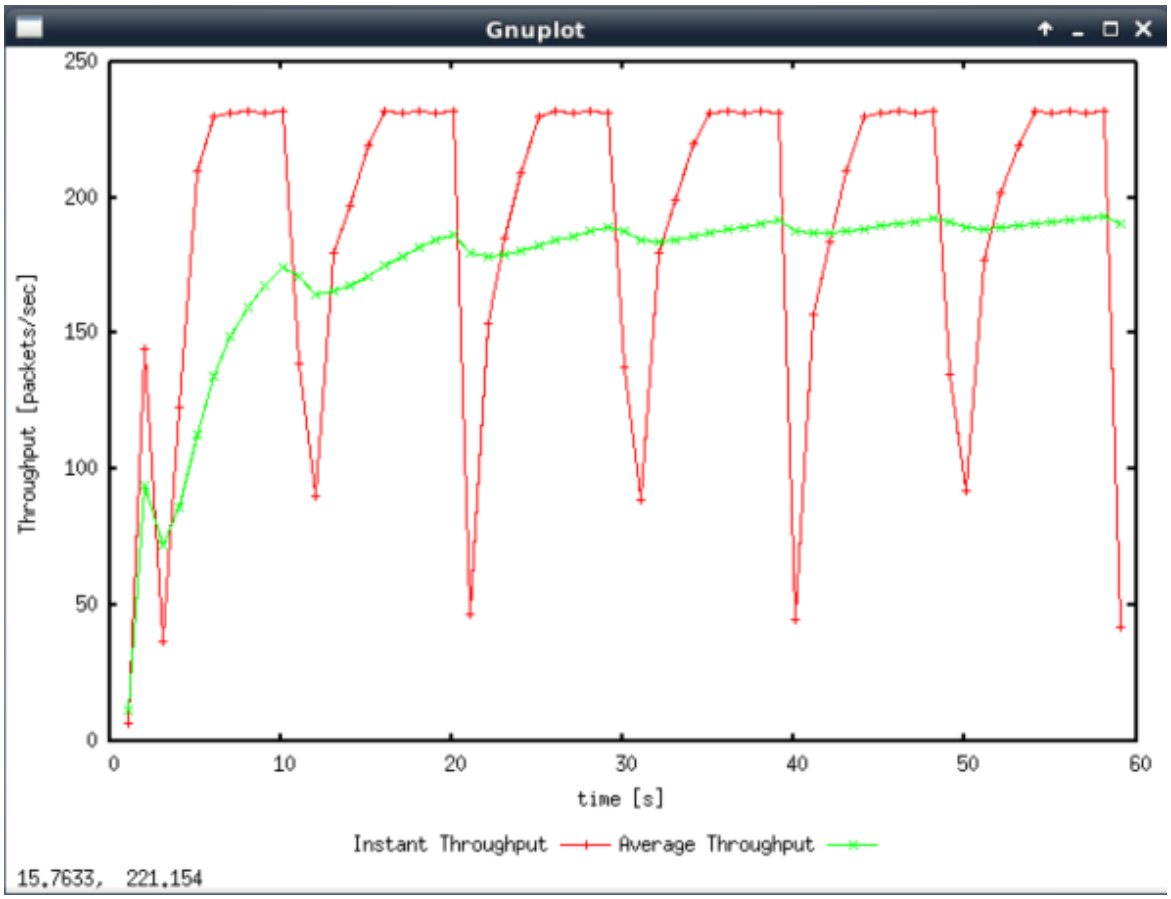
$gnuplot WindowTPut.plot



$ns tpWindow.tcl 66 100ms $gnuplot WindowTPut.plot



$ns tpWindow.tcl 80 100ms $gnuplot WindowTPut.plot



After trying various combinations, it can concluded that the maximum window size is 66 at which TCP stops oscillating and any value larger than 66 the graph for TCP oscillates constantly.
The average throughput at this point = 220 packets/sec

= IP + TCP

Headers = 20 + 20 = 40 bytes

Payload of the packet = 500 Bytes

Packets per second throughput(bps) = (500+40)*8*220

= 1038400 bps

The actual throughput 1038400 bps =~ 1.038400 Mbps which is slightly larger than link capacity.

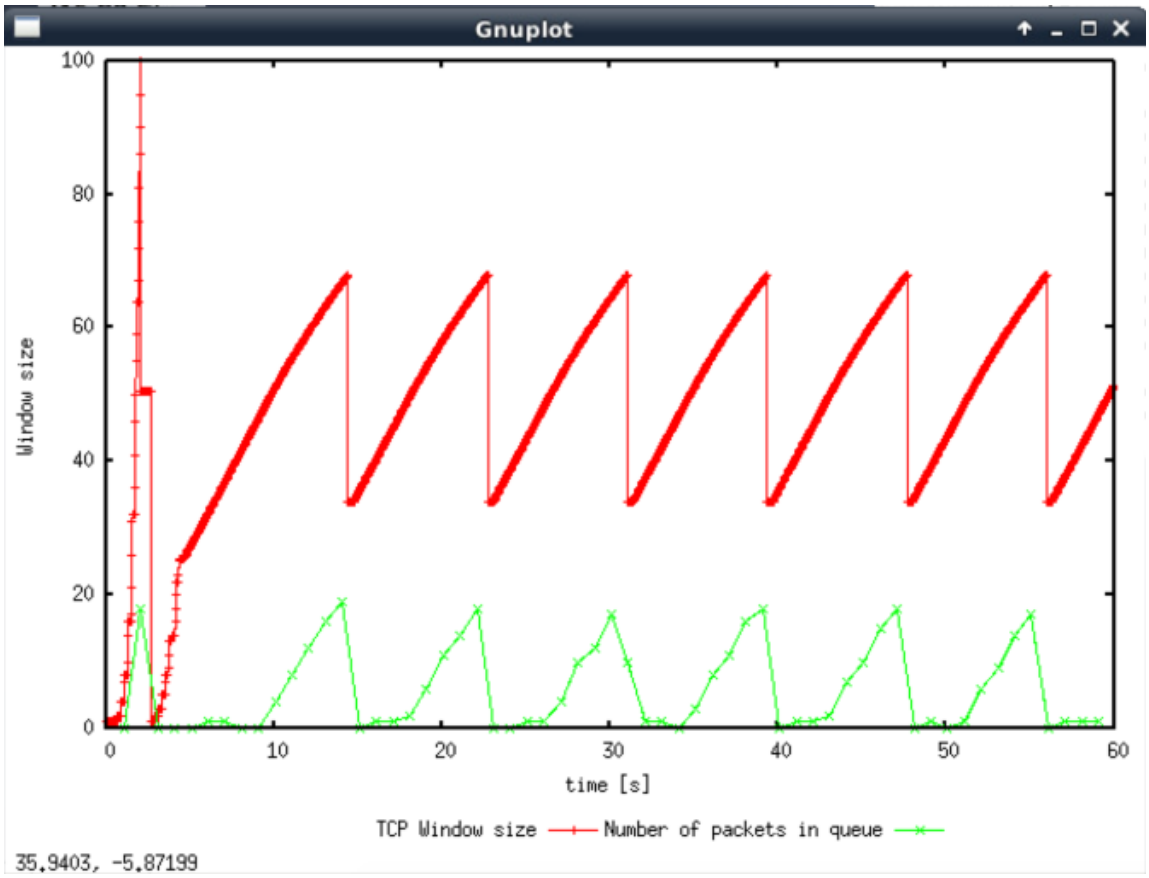
Question 4 : Repeat the steps outlined in Question 1 and 2 (NOT Question 3) but for TCP Reno. Compare the graphs for the two implementations and explain the differences. (Hint: compare the number of times the congestion window goes back to zero in each case). How does the average throughput differ in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

Solution 4 :

By running the below commands,

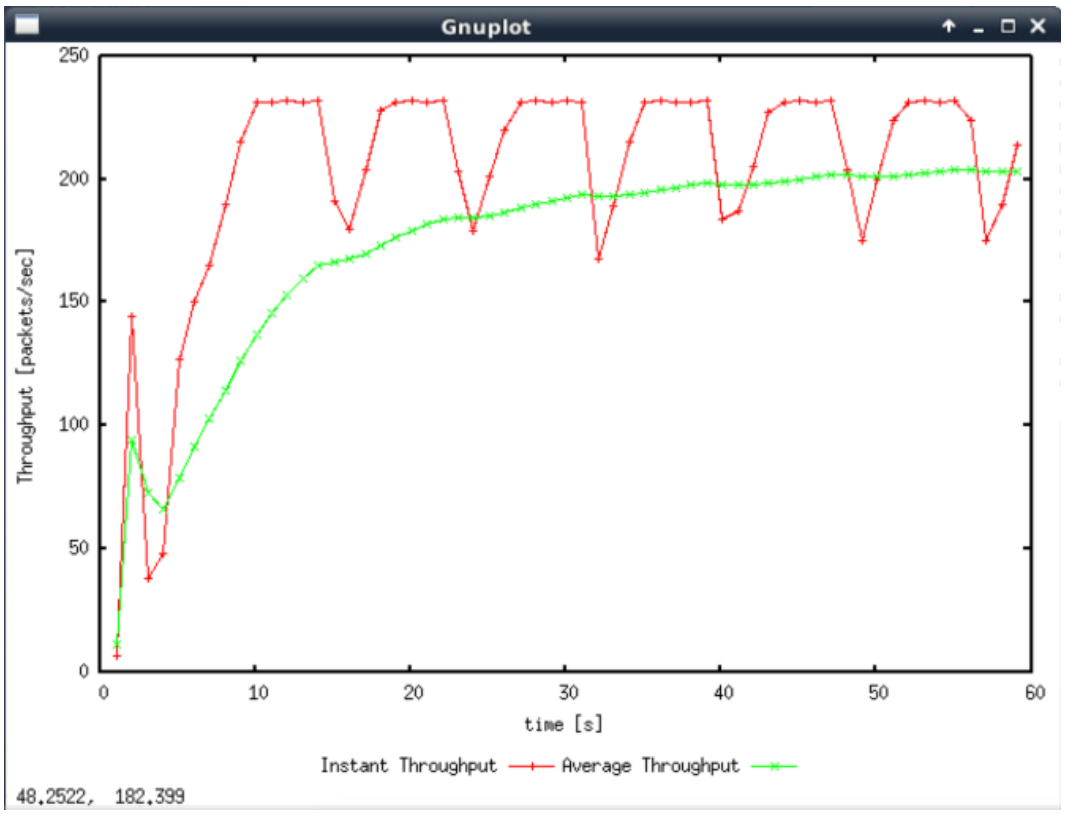
$ns tpWindow.tcl 150 100ms

$gnuplot Window.plot



$ns tpWindow.tcl 150 100ms

$gnuplot WindowTPut.plot



With TCP Reno, it does not re-enter a slow-start phase when loss occurs TCP Reno throughput ~= 200 pps

(500 + 40) * 8 bits * 200 = 864,000 bps

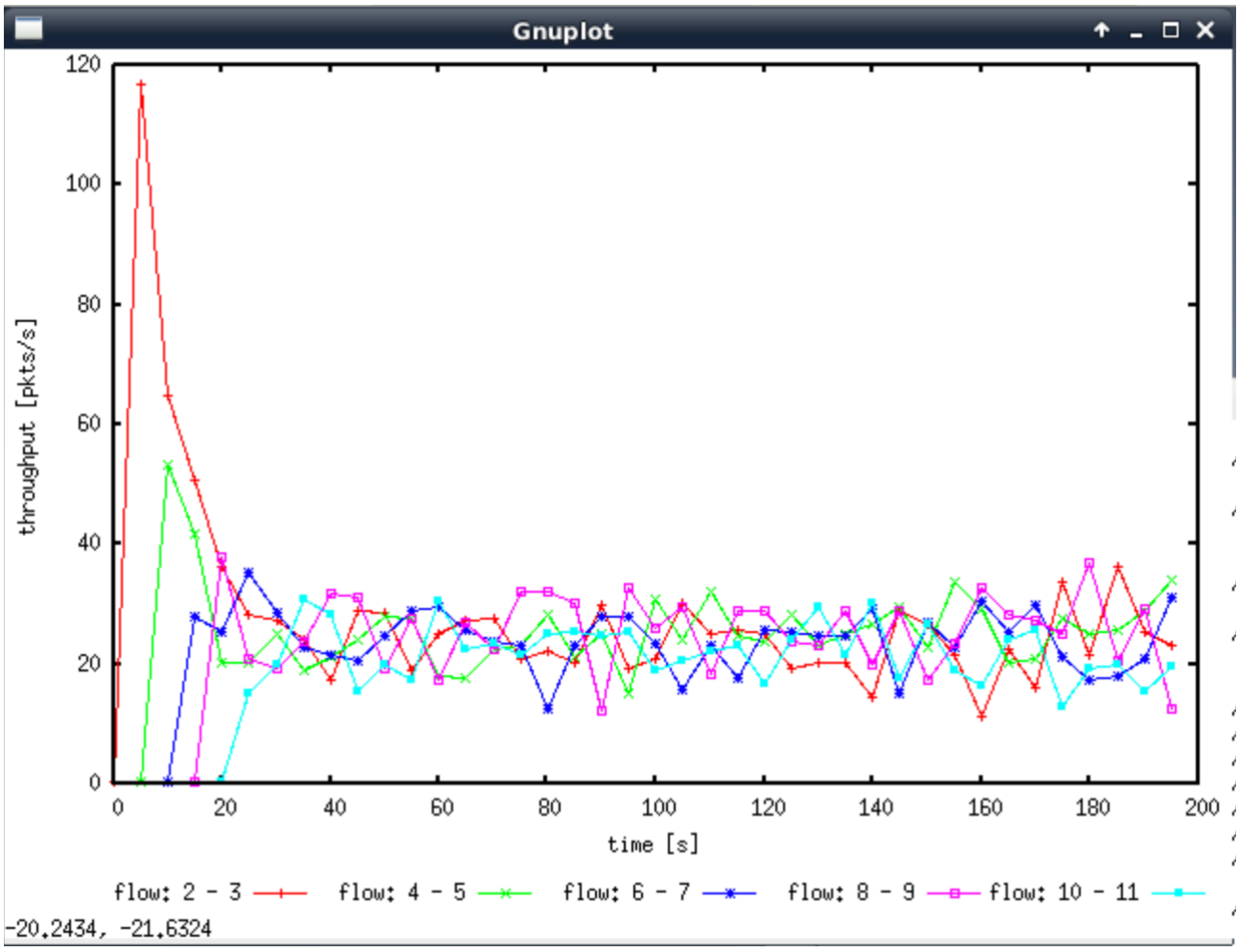
Throughput previously with TCP Tahoe was 820,800 bps, therefore TCP Reno performs better.

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.

Solution 1 :

Yes, each flow does get an equal share of the capacity of the common link.



As observed in the graph, although throughput for the first few connections start off higher than the others that aren't connected yet, it averages out to be fairly equal with some expected fluctuations in throughput(different colour lines).

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.

Solution 2 :

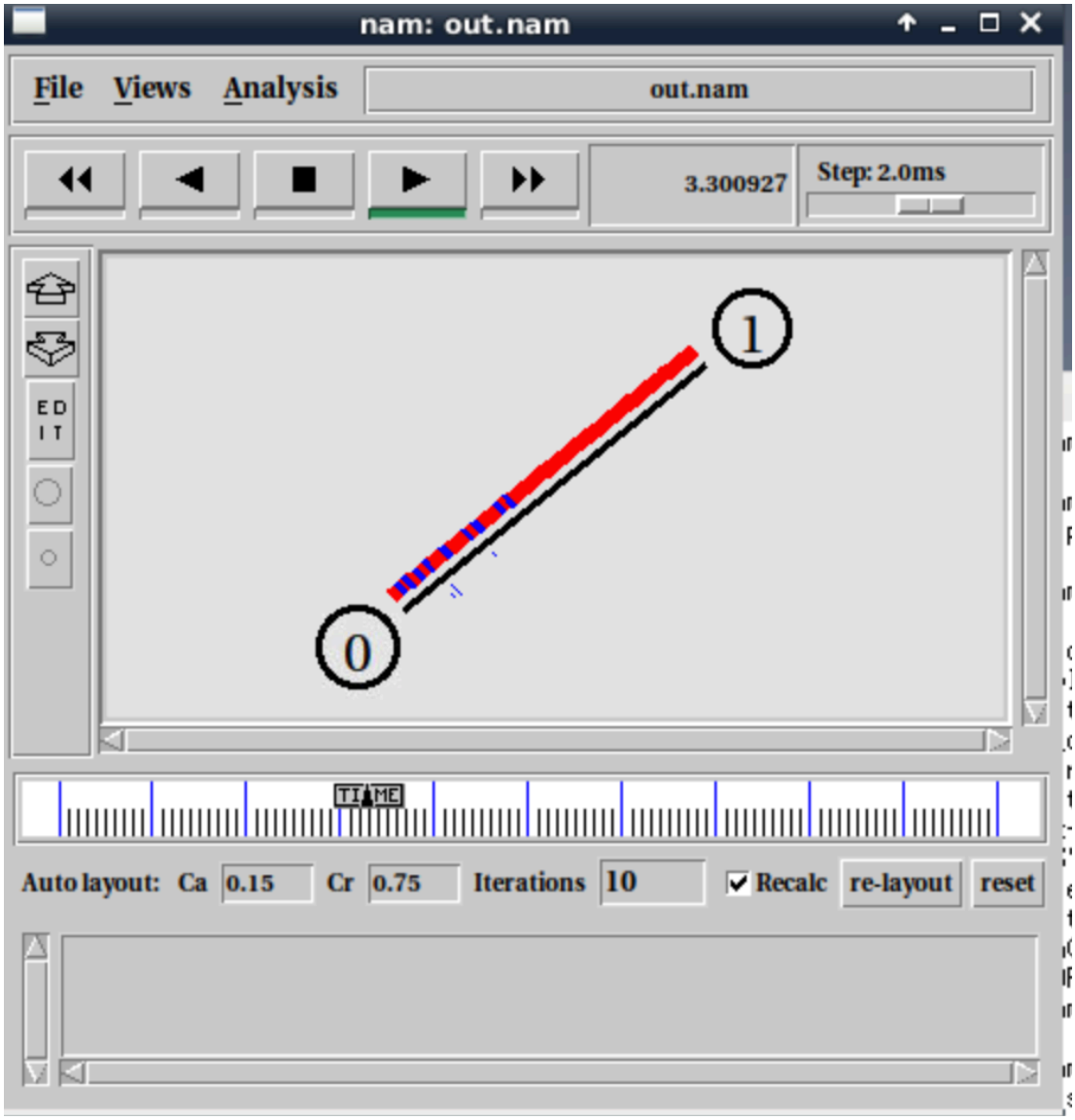
As observed in the previous graph, throughput for the first few connections start off higher and then significantly decreases for each new observation that is created. This is due to the congestion control mechanisms.I consider this to be fair behaviour, as each connection adjusts the size of the connection window when a new connection comes in, to allow sharing of the common 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 ?

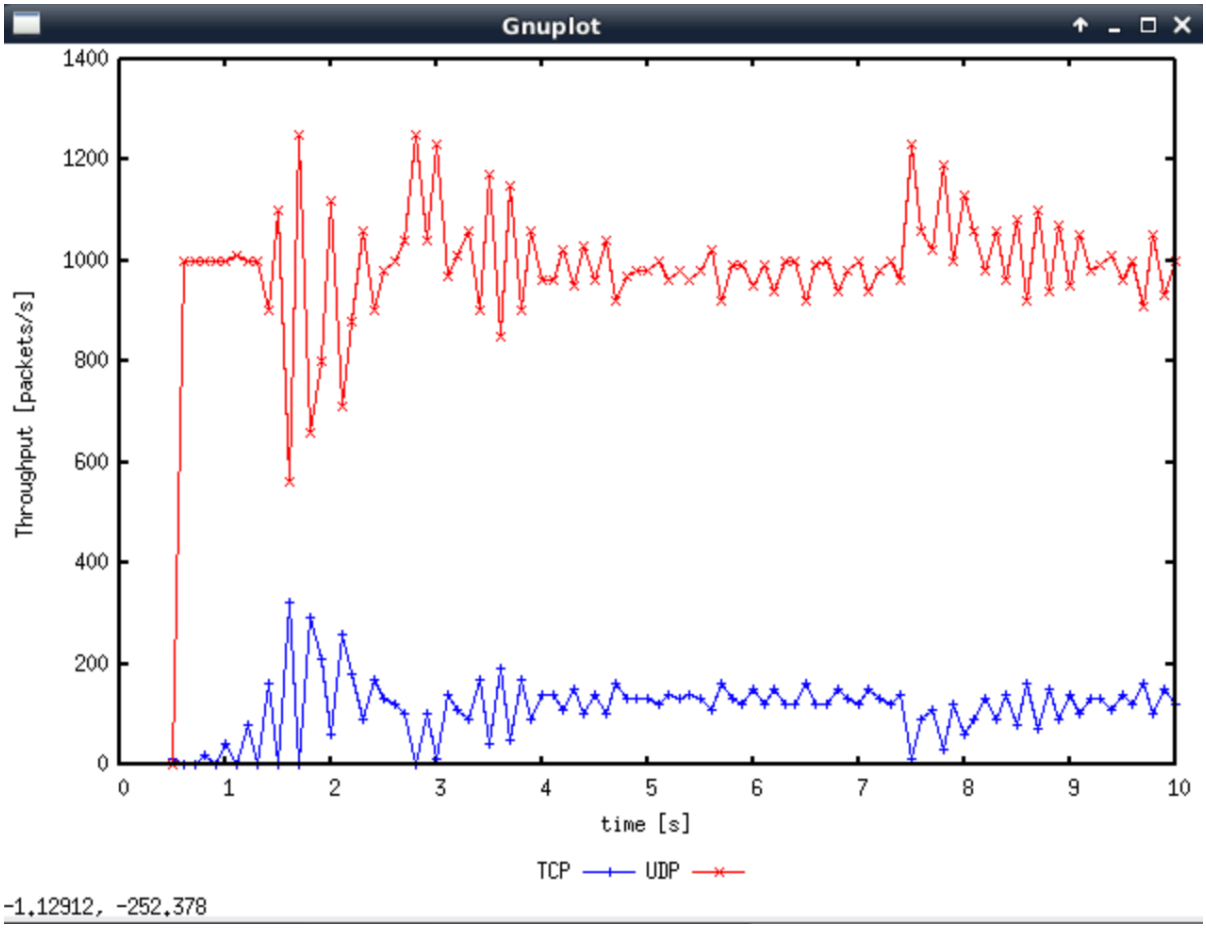
Solution 1 :

The script opened the NAM window and we observed packets with two different colours(blue and red) where



TCP = blue output and UDP = red output.

Throughput of the two flows can be observed in the graph below,



The TCP flow is expected to be lower than the UDP due to congestion control.

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.

Solution 2 :

One flow achieve higher(UDP) throughput than the other(TCP) because UDP has no congestion control feature and there is no connection setup time. So, as soo. As a UDP connection is established it straight away transmits packets.

In case of TCP, there is a connection setup time involved and also due to the congestion control there is lot of time wasted.

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?

Solution 3 :

Typically, UDP is used in applications where speed is more critical than reliability.

UDP is faster that TCP because error recovery is not attempted. It is a "best effort" protocol.
UDP does not provide any guarantee that the messages or packets sent would reach at all. UDP is lightweight so there is no ordering of messages, no tracking connections, etc. and it does not handle reliability and congestion control. It does not have an option for flow control as well.

If everyone started using UDP instead of TCP, the network could just collapse.