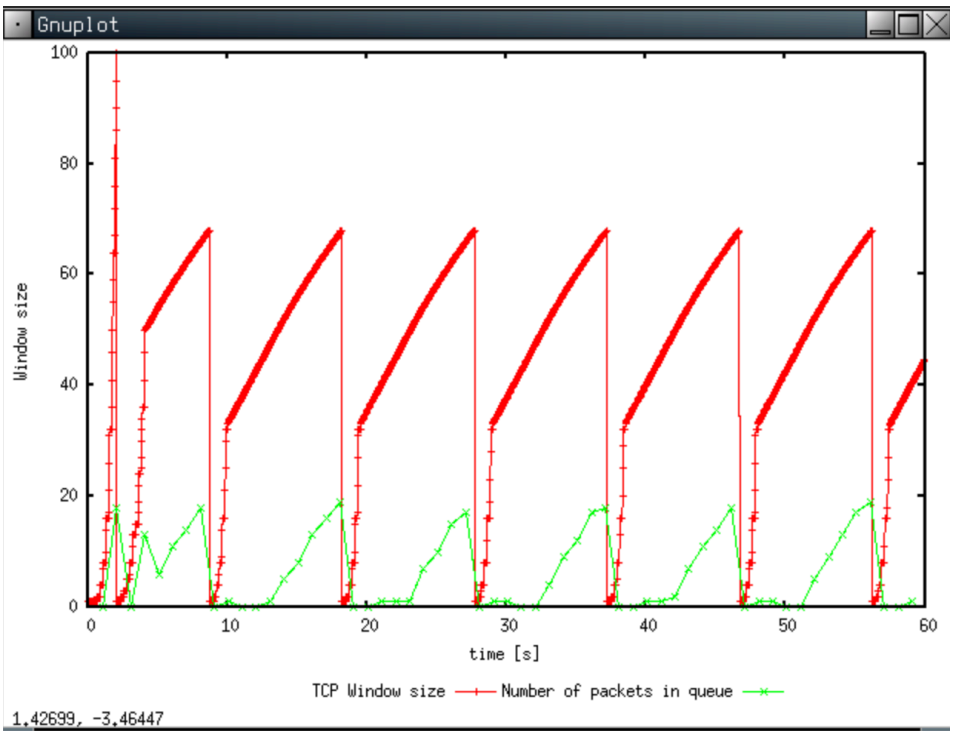
Exercise 1: Understanding TCP Congestion Control using ns-2

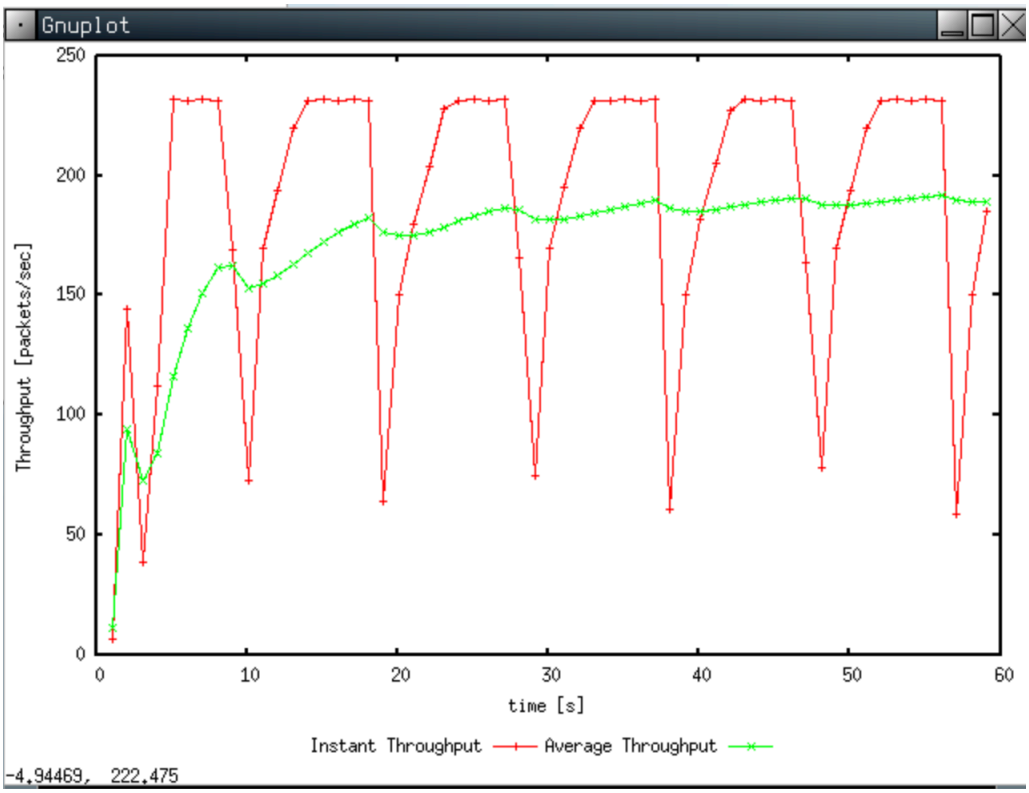
Question 1: Run the script with the max initial window size set to 150 packets and the delay set to 100ms (be sure to type "ms" after 100). In other words, type the following:… What is the maximum size of the congestion window that the TCP flow reaches in this case? What does the TCP flow do when the congestion window reaches this value? Why? What happens next? Include the graph in your submission report.

- Max congestion window size = 100
- Packet loss occurs when the window size reaches this size.
- This is because the simulated network's queue size is only 20.
- Once the threshold has been reached and the network is congested, the sender resets back to the slow start phase and keep repeating every time it detects congestion.



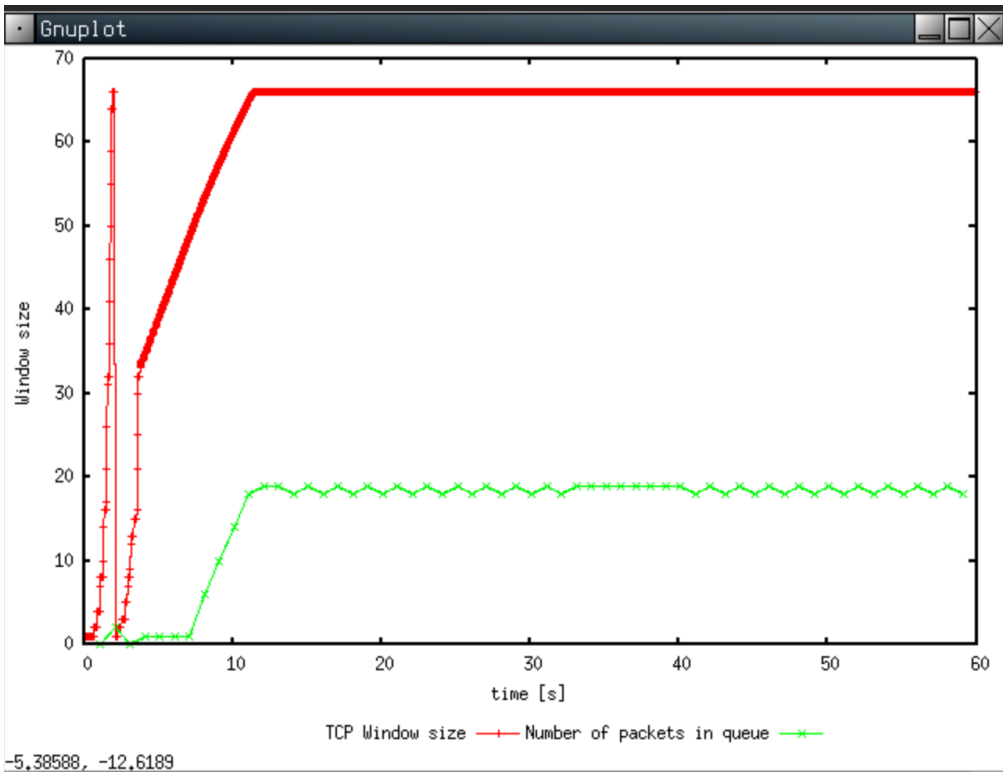
Question 2: From the simulation script we used, we know that the payload of the packet is 500 Bytes. Keep in mind that the size of the IP and TCP headers is 20 Bytes, each. Neglect any other headers. What is the average throughput of TCP in this case? (both in number of packets per second and bps)

- Header + Packet = 20 + 20 + 500 = 540 bytes = (540 x 8) bits
- Throughput rate is approximately 170 packets/sec
- Average Throughput = 4320 bits x 170 = 734400 bps



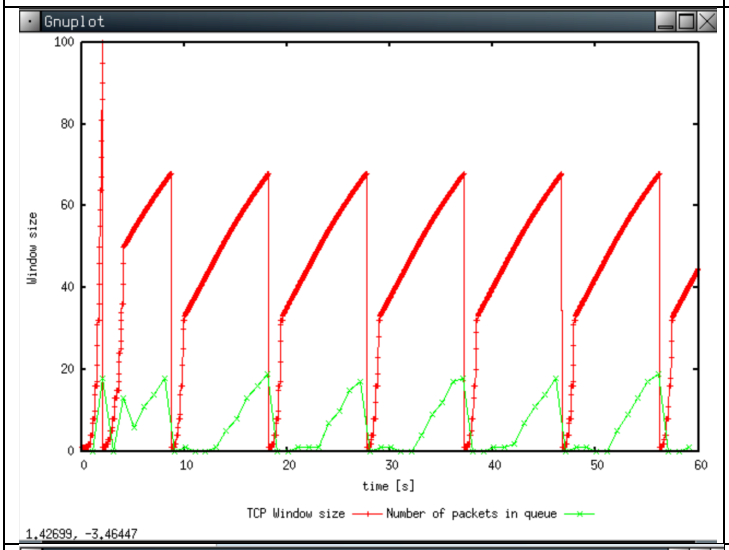
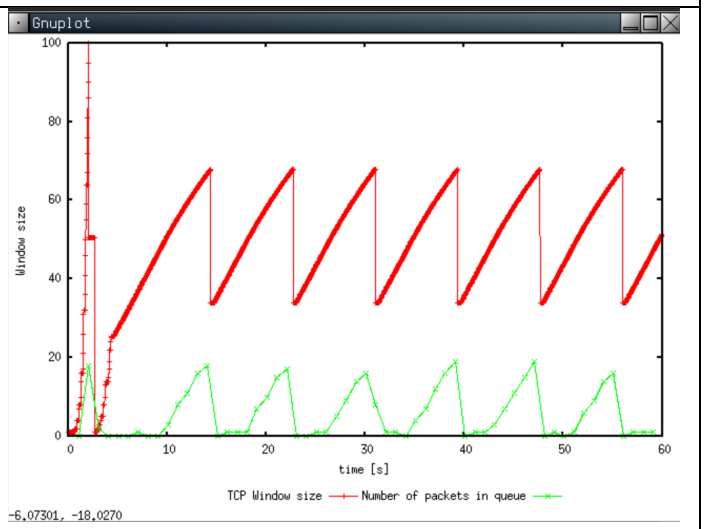
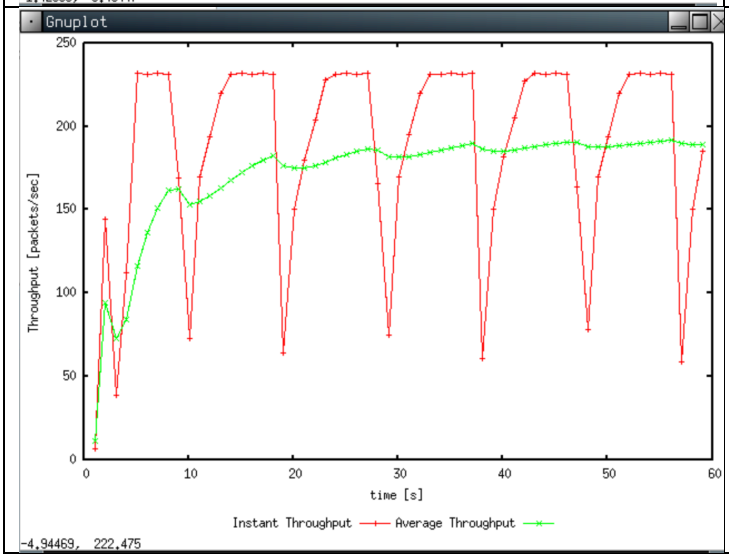
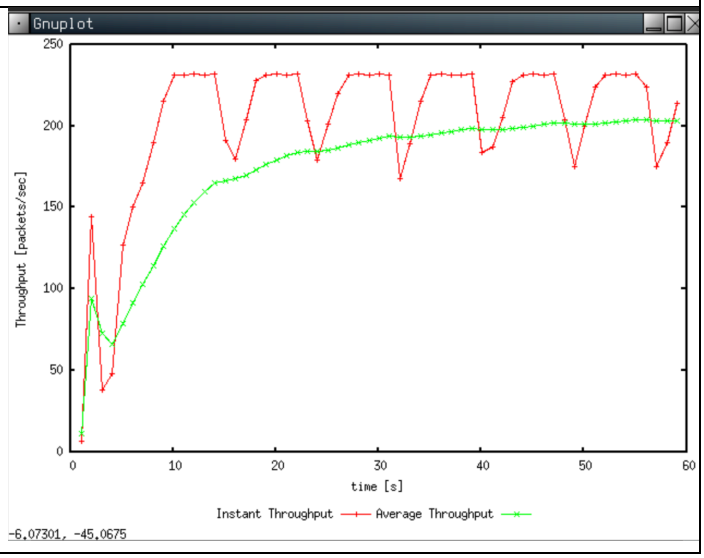
Question 3: Rerun the above script, each time with different values for the max congestion window size but the same RTT (i.e. 100ms). How does TCP respond to the variation of this parameter? Find the value of the maximum congestion window at which TCP stops oscillating (i.e., does not move up and down 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)?

- As the window size decreased, the length between peak kept increasing (ie. started becoming more stable).



- Stopped oscillating at window size = 66
- Average throughout = 220 packets/sec
- Link capacity = 1Mbps = 8million bps
    - Can handle packets of size = 36363 bits ~ 4545 bytes
    - Link capacity is most likely still underutilised. The bottle neck is the queue.

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?

TCP Tahoe	TCP Reno
	
	

TCP Reno (Q1 & 2)

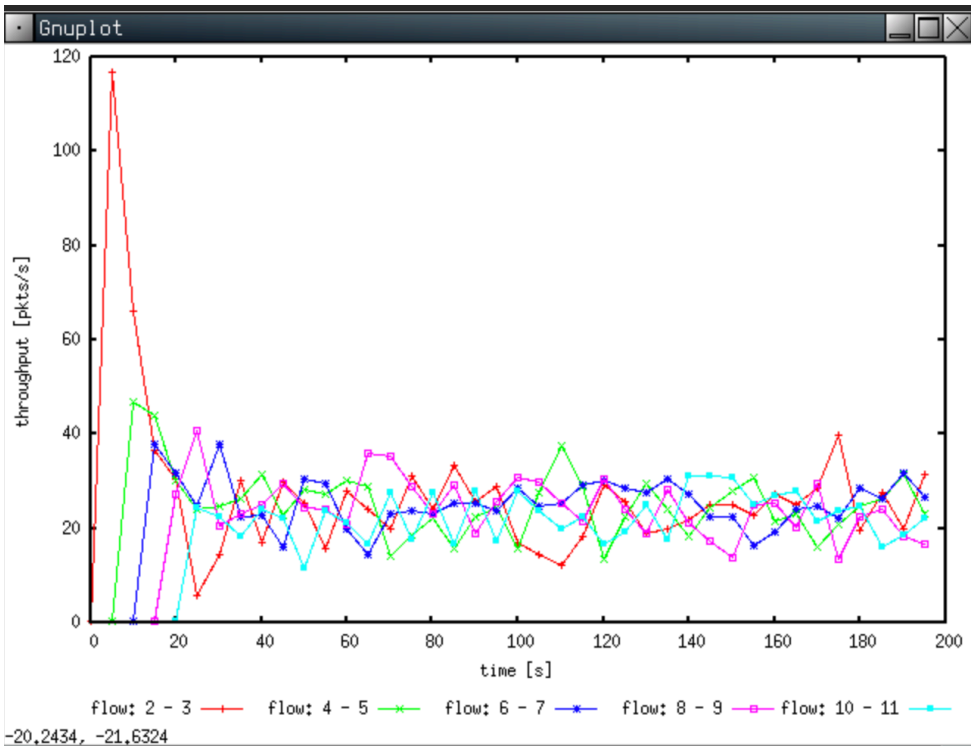
- TCP Reno goes into fast recovery right after getting the three duplicate ACKs, as we see in the chart above. After fast recovery we see it transition to congestion avoidance. TCP reno doesn't seem to keep restarting back with slowstart and the throughout looks to be more consistent, compared with TCP Tahoe.
- Q1 – TCP Reno
    - Max congestion window size = 100
    - Packet loss occurs when the window size reaches this size.

    - This is because the simulated network's queue size is only 20.
    - Once the threshold has been reached and the network is congested, the sender resets back to the fast recovery (instead of back to slow start like TCP Tahoe) phase and then congestion avoidance.
- Q2 – TCP Reno
    - Header + Packet = 20 + 20 + 500 = 540 bytes = (540 x 8) bits
    - Throughput rate is approximately 200 packets/sec
    - Average Throughput = 4320 bits x 200 = 864000 bps

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.

- Initially each flow didn't seem fair, but as time went on, it slowly started converging to become fairer.
- The throughput = 120 pkts/sec, there are 5 flows, thus each flow should be around 24 pkts/sec. By observing the graph, although not perfect, each flow does seem to be hovering between 20 – 40 pkts/sec.



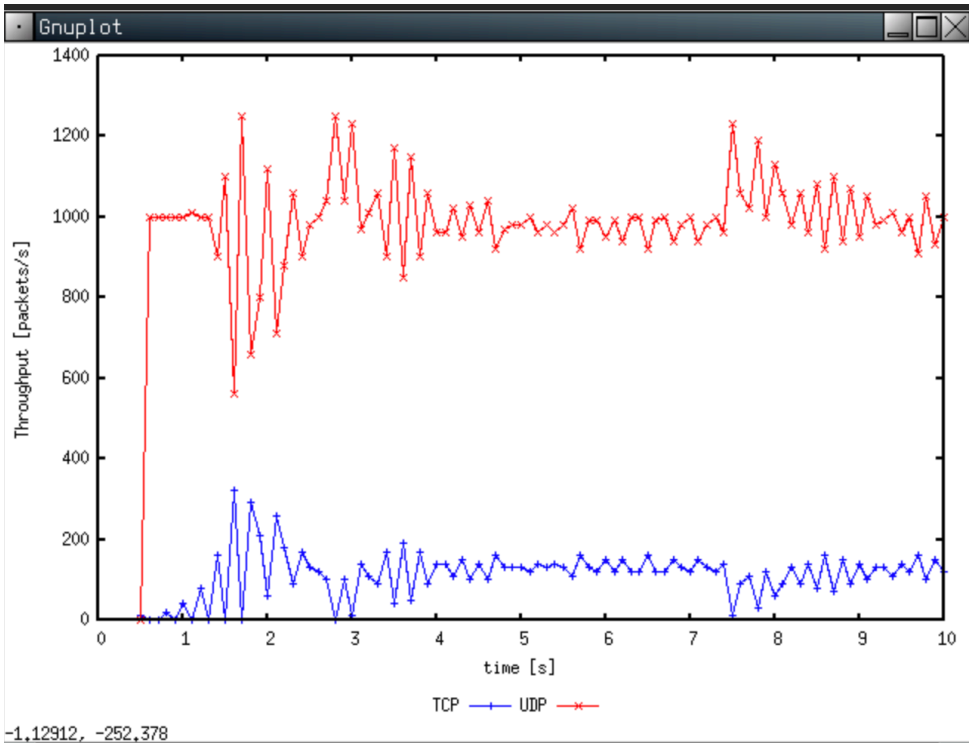
Question 2. What happens to the throughput of the pre-existing TCP flows when a new flow is created? Explain the mechanisms of TCP which contribute to this behaviour. Argue about whether you consider this behaviour to be fair or unfair.

- Initially when a new flow is created, it would seem unfair, as seen in Q1. However, TCP's congestion control mechanisms would slowly start redistributing throughput allocation via AIMD, thus ultimately making it fair. Although it's not initially fair, it becomes pretty fair after some time.

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 ?

- I'm expecting UDP to be greedier compared with TCP, since UDP doesn't care about congestion control, congestion avoidance, flow control etc.
- After running the simulation, we get the following graph below. The red line represents UDP and blue line represents TCP.



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.

- UDP doesn't have any form of congestion control (by default, programmers can implement their own if they'd like). This is why we see UDP having a higher throughput than TCP.
- Since TCP implements congestion control, we can see that in order to prevent congestion, the TCP connection lowers its throughput.
- In a congested network, regardless of the transport protocol used (TCP or UDP), if the network is congested, packets start dropping. This is why we see UDP is being restricted to a certain level, because every time it tries to reach 100% throughput, packets start dropping and it goes back down etc. Also, perhaps on the network layer, the router might be enforcing some QoS mechanism, preventing UDP from hogging the link.

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 of using UDP over TCP for file transfer	Disadvantages of using UDP over TCP for file transfer
• No need to establish a connection every time (useful for pausing and resuming transfers) • Lightweight compared with TCP. Header is smaller. • Since UDP doesn't have many in-built reliability mechanisms, we can reduce the need for additional processing.	• Provides no form of reliability (except checksum) • Can clog up the network (assuming the sender cares) • No way of knowing if the receiver correctly received the packet (application developers need to implement this if they want).

- If everybody started using UDP, instead of TCP, at the highest throughput, the network would become so congested where it would no longer be usable.