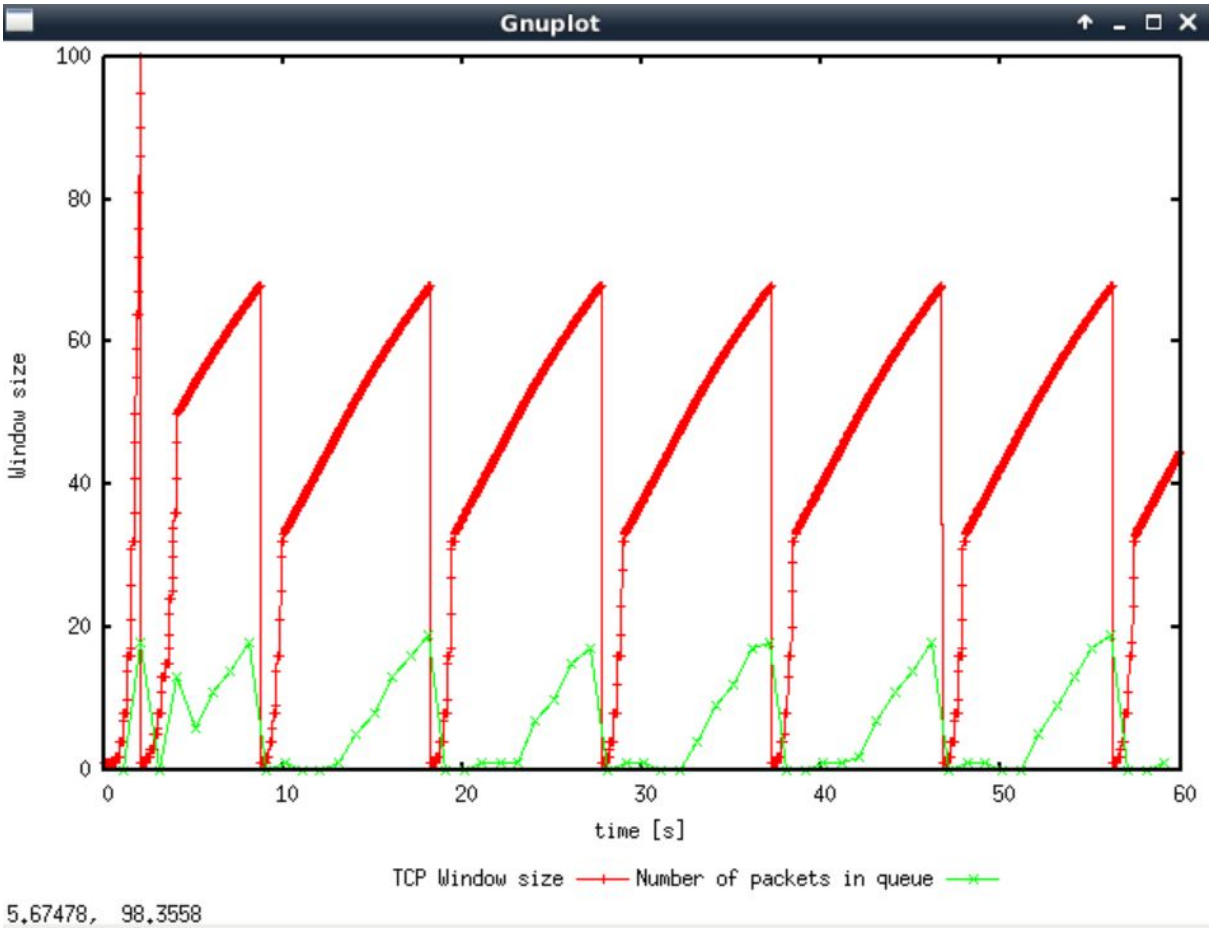
Lab5
z5165555
Bethia Sun

Exercise 1:

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). 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 maximum size of the congestion window that the TCP flow reaches is approximately 70 packets. TCP Tahoe sets its cwnd window to 1 when a timeout/triple duplicate ACK is detected and sets ssthresh to half the value of the cwnd value when a loss is detected. Then, TCP Tahoe restarts the slow start phase until it reaches ssthresh, at which point it increases the congestion window linearly.

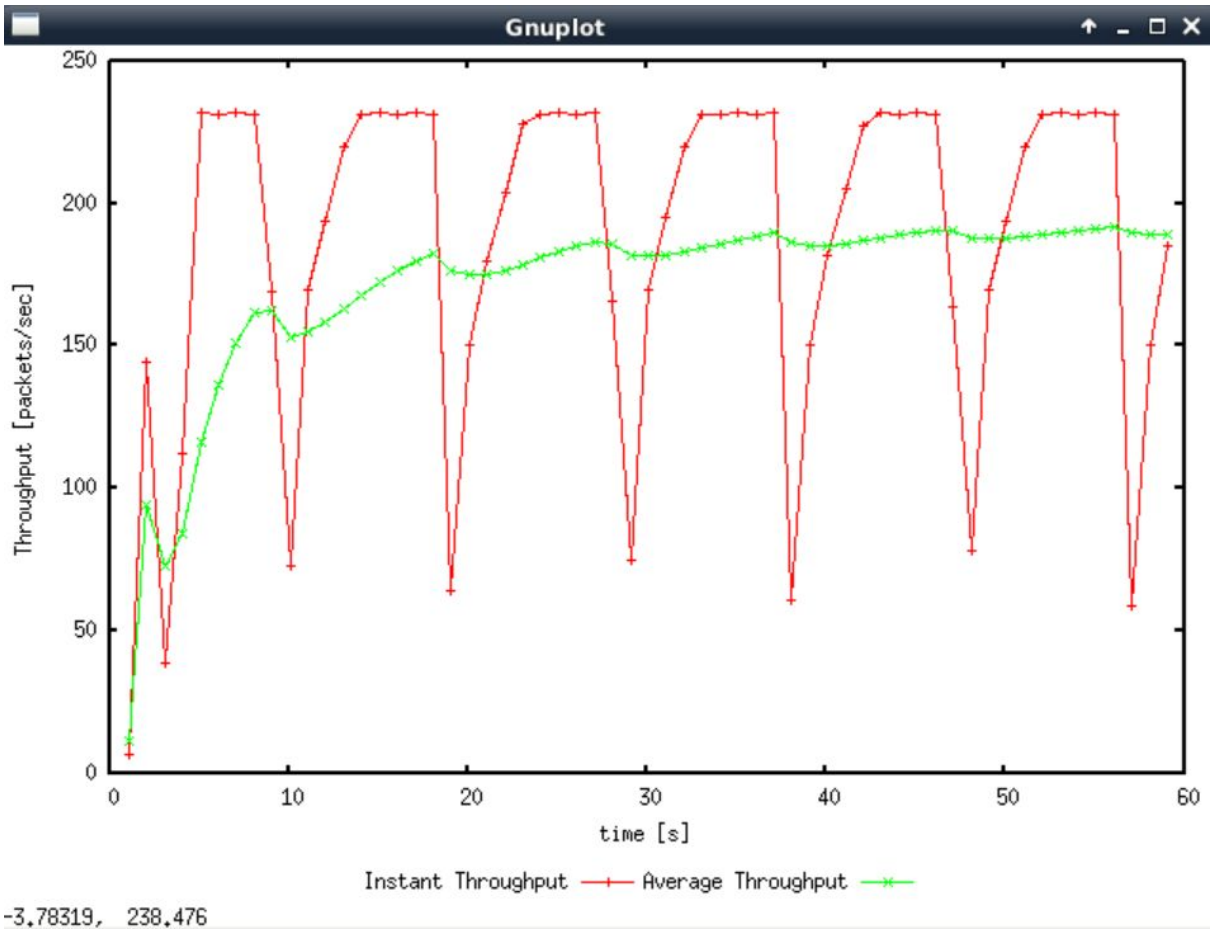
These actions are done so that TCP Tahoe can ‘probe’ to determine the available bandwidth.



Graph of output

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)

The average throughput of TCP in packets per second is 175 packets/second. To calculate the average throughput of TCP in bps, we consider how much data is in a packet. The data in a packet is 500 Bytes (payload) + 2*20 Bytes (2 headers) = 540 Bytes. So, the average throughput of TCP in bps is 175*540 Bytes/s = 94500 Bytes/s = 756000 Bits/s = 0.756 Mbps



Graph of output

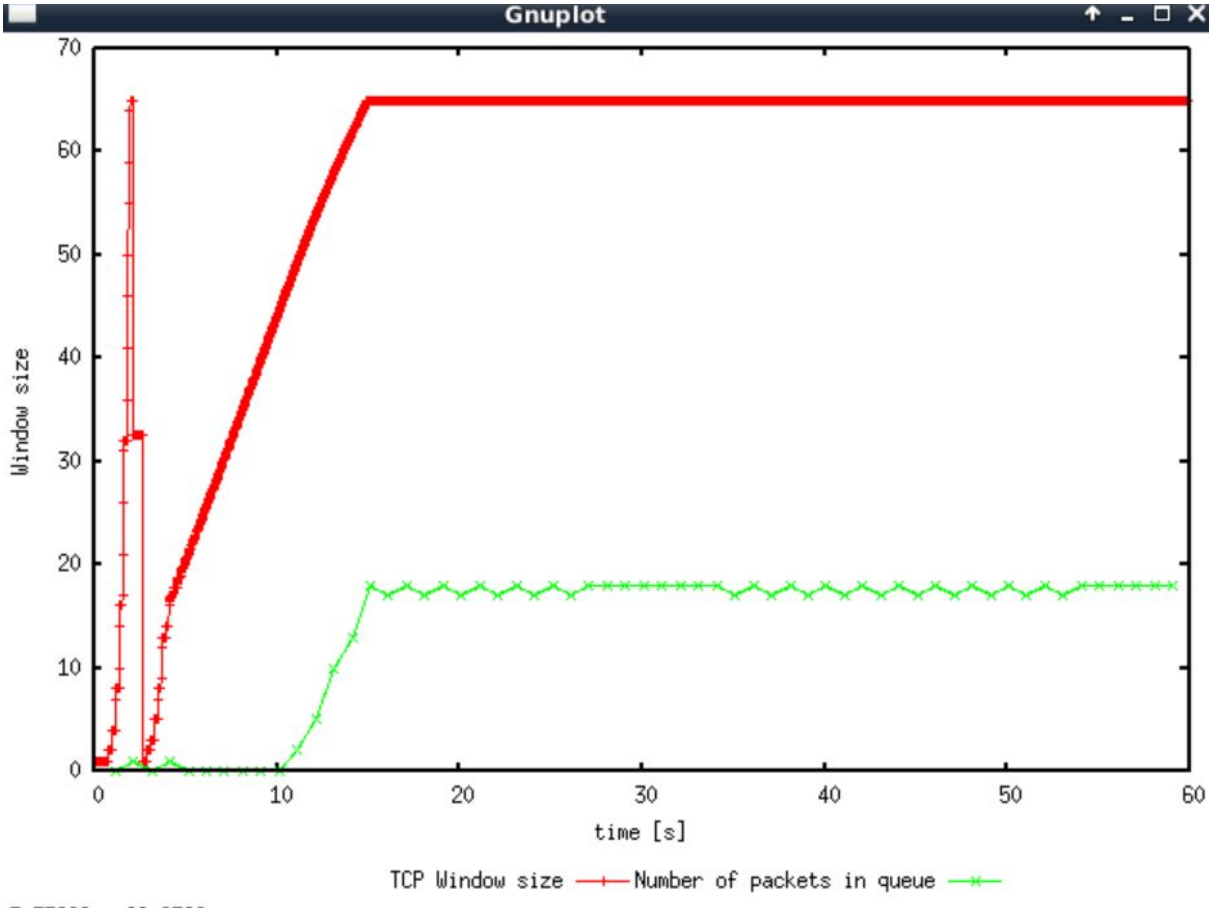
Question 3: Rerun the above script, each time with different values for the max congestion window size but the same RTT (i.e. 100ms). How does TCP respond to the variation of this parameter? Find the value of the maximum congestion window at which TCP stops oscillating (i.e., does not move up and down 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 maximum congestion window at which TCP stops oscillating to reach a stable behaviour is when cwnd is approximately 65 packets (see first graph below). When we increase the congestion window size past this value, we get ‘tooth’ behaviour where TCP repeatedly

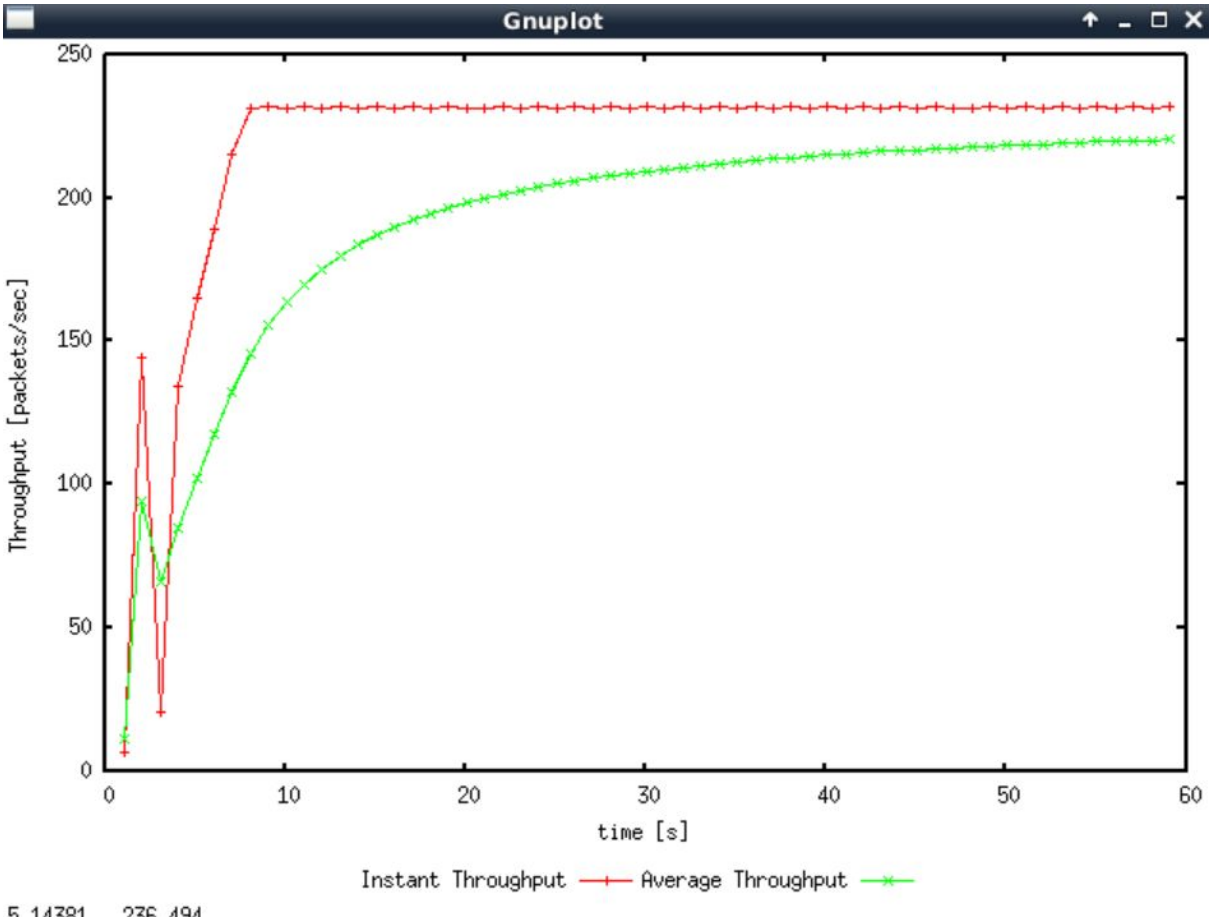
increases and decreases its cwnd window while probing for available bandwidth. This is because some form of loss will always occur when cwnd exceeds the maximum congestion window size, prompting TCP Tahoe to reset cwnd to 1, at which point it increases cwnd at first exponentially, then linearly.

If the TCP congestion window is less than this maximum cwnd value, no losses due to congestion occur so the graph appears linear after the slow start phase.

The average throughput in packets/s is approximately 220 packets/second (see second graph below). In Bytes per second, this is equivalent to 220*(540) Bytes/s = 118800 Bytes/s = 950400 Bits/s = 0.9504Mbps. This is 95% of the link capacity (1Mbps)



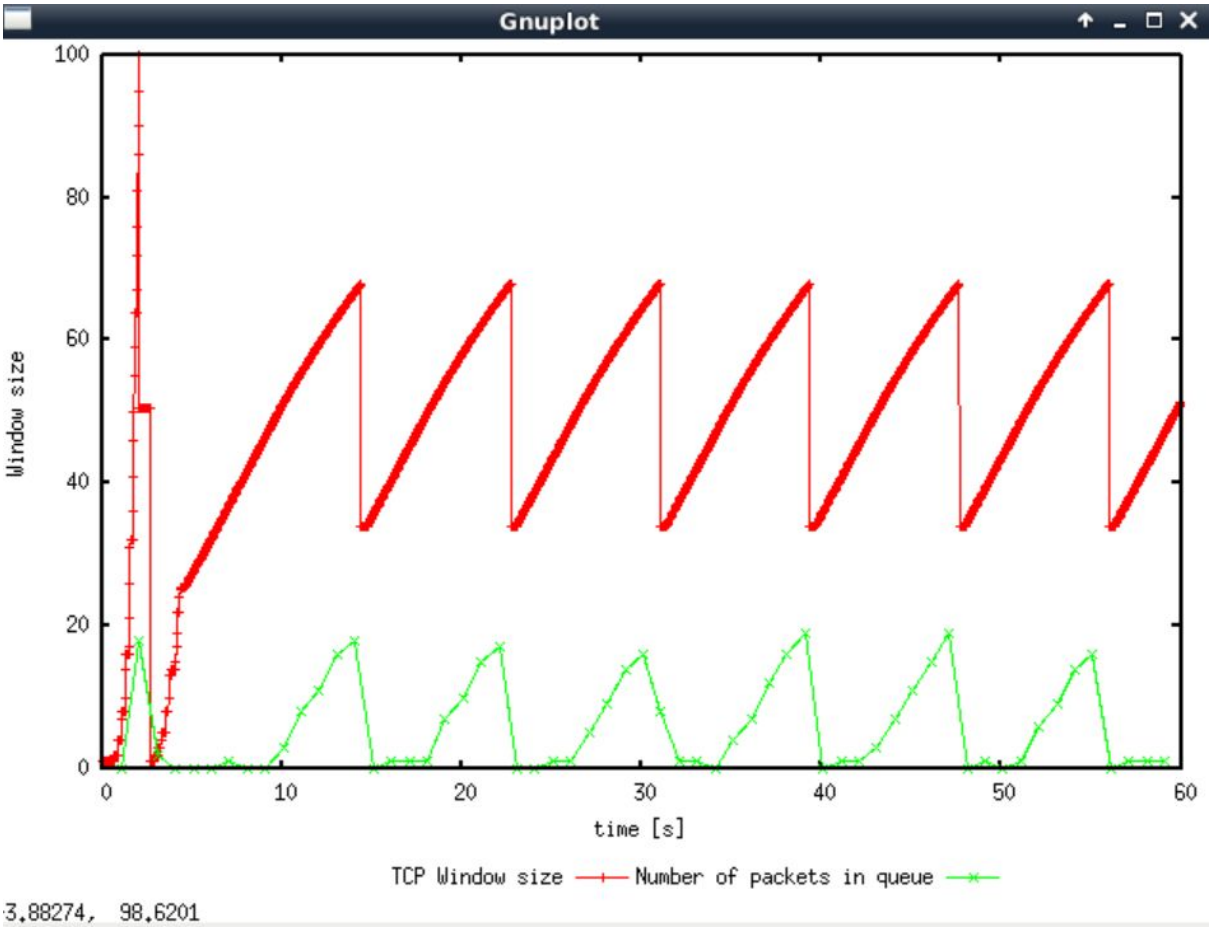
Graph of output showing the script's behaviour with max initial window size set to 65

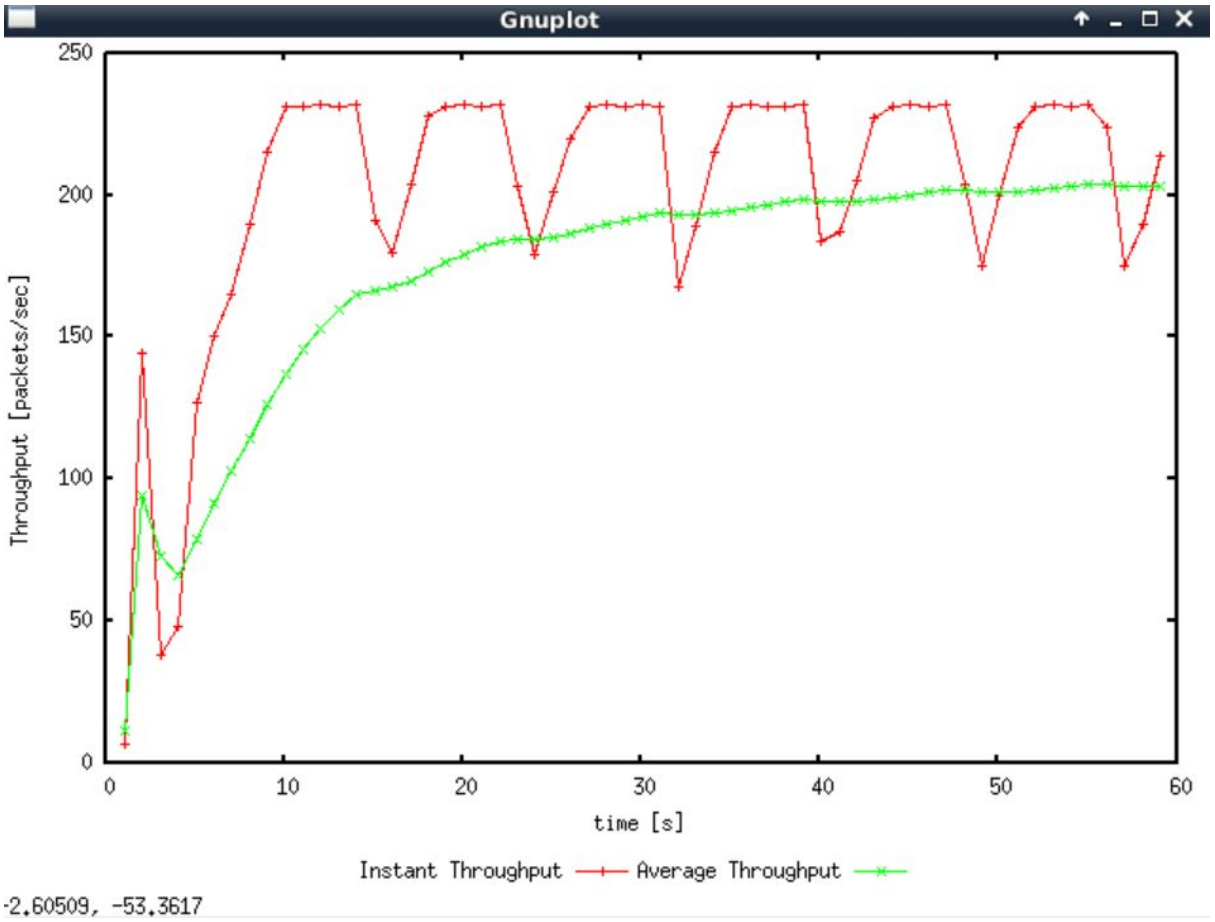


Question 4: Repeat the steps outlined in Question 1 and 2 (NOT Question 3) but for TCP Reno. Compare the graphs for the two implementations and explain the differences. (Hint: compare the number of times the congestion window goes back to zero in each case). How does the average throughput differ in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

The congestion window goes back to zero 7 times with TCP Tahoe. In comparison, with TCP Reno, the congestion window goes back to zero only once (see graph below). This is because with TCP Reno, the congestion window resets to 1 on timeouts and not triple duplicate ACKs in which case the value of cwnd is halved. In contrast, TCP Tahoe's congestion window is reset to 1 on both triple duplicate ACKs and timeouts.

The average throughput for TCP Reno is approximately 200 packets per second (see second graph below). This is equivalent to 200*(540) Bytes/s = 108000 Bytes/s = 864000 bits/s = 0.864Mbps. This is 1.14 times the throughput of TCP Tahoe (which is 0.756 Mbps). An intuitive explanation for this is that the cwnd value does not reset back to 1 every time a triple duplicate ACK is received, so TCP Reno can send more packets when a triple duplicate ACK is received, whereas TCP Tahoe has to restart by sending 1 packet.


Gnuplot
Window size
time [s]
TCP Window size
Number of packets in queue
3.88274, 98.6201



Exercise 3:

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

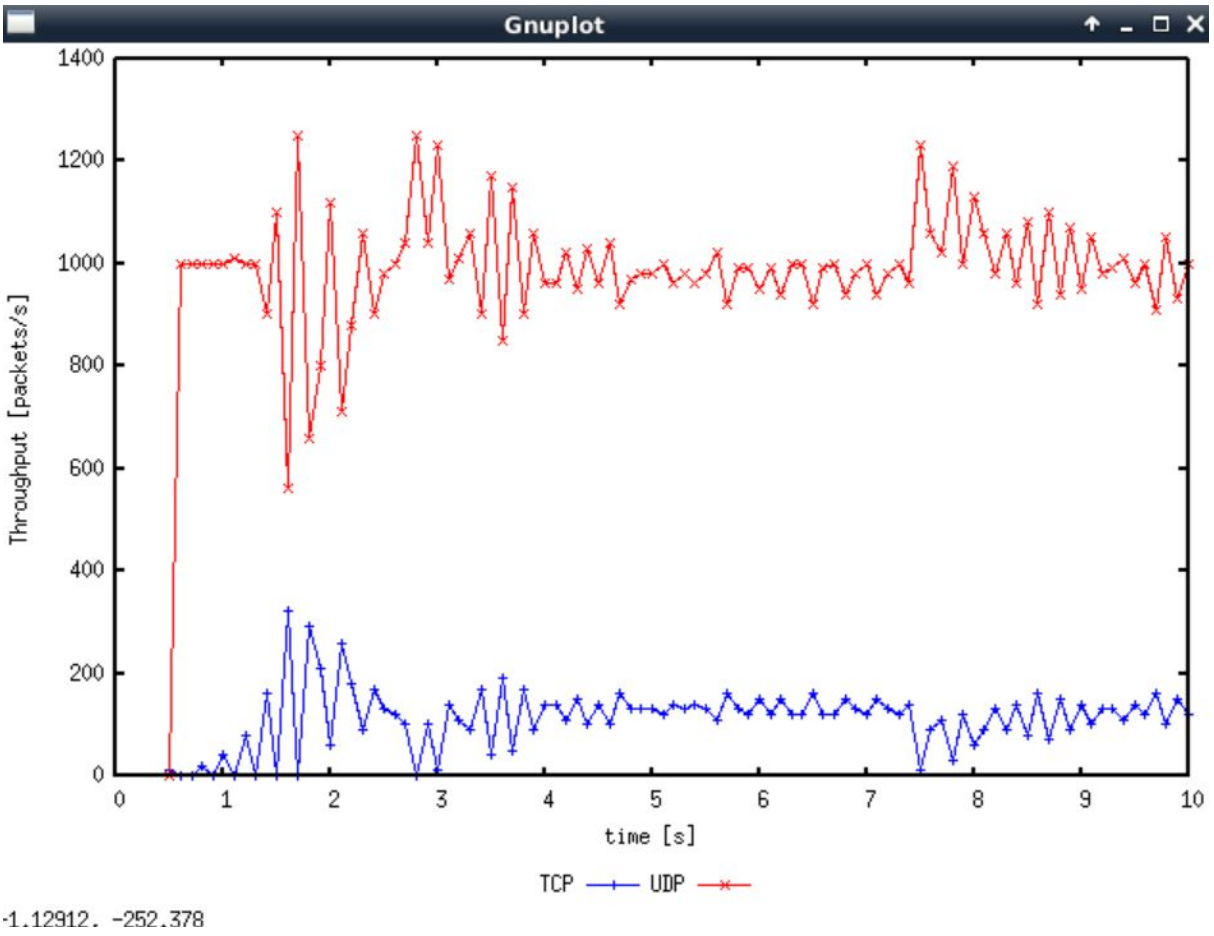
UDP (in red) will blast away a continuous packet stream from the sender to the receiver. UDP will blast away as close as it can to its maximum rate (4Mbps) as it has no congestion and flow control mechanism. Initially, UDP throughput will be high while the network is not congested as the receiver will be able to receive many of these UDP packets. TCP will start slowly sending packets during its slow start period, exponentially increasing the value of cwnd until it detects a timeout/triple duplicate ACK. Once TCP's value of cwnd exceeds/becomes close to the amount of available bandwidth that is left in the network, the network will start becoming congested, and a lot of UDP packets will be lost by the receiver and an increasing amount of TCP packets will be lost by the receiver. Once TCP's value of cwnd starts to stabilise, both the the TCP and UDP flow will stabilise accordingly.

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 achieves higher throughput compared to TCP because it does not need to wait for ACKs so a continuous packet stream is permitted. This means that in the initial stages, UDP can use most of the link's available bandwidth to transmit data at its maximum rate (4Mbps).

While TCP is in the very early stages of its slow start phase, it will not use much of the link's available bandwidth (as the cwnd value is relatively small in the very early stages of slow start and this limits the amount of packets TCP can send per RTT), so congestion in the network due to TCP and UDP both transmitting data will be low. So, when TCP throughput is low (during the early stages of slow start and whenever TCP halves its cwnd value), there is less competition over resources in the network meaning the receiver is able to receive most of the UDP packets successfully, leading to high UDP throughput. Once TCP increases its cwnd value as it attempts to probe for available bandwidth, however, it sends more packets, increasing competition in the network over available bandwidth and congesting the network, which leads to more UDP packets being dropped by the receiver. This reduces UDP throughput and increases TCP throughput. Once TCP's cwnd value has stabilised, both the TCP and UDP throughput have stabilised due to the mechanism described above.

This mechanism is evident in the graph below, where the peaks in TCP throughput correspond to the troughs of UDP throughput and vice versa.



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 UDP:

- UDP takes advantage of any available bandwidth in the network when TCP backs off

- Absence of acknowledgement field in UDP makes it faster than TCP as it does not need to wait for ACKs to be received
- Small header size reduces processing time at the routers, this frees up available processing time to be used elsewhere

Disadvantages:

- No congestion control - so a large number of users using UDP to transmit data can congest the network
- Unreliable - packets may not be successfully sent/received by the receiver, meaning the file will be corrupted

UDP has no congestion control. So, if everybody started using UDP for a file transfer, all these users would congest the network, meaning there would so much packet overflow at routers that very few UDP packets would successfully get from source to destination. Barely any of the users would be able to successfully download the file properly due to the high loss rate between UDP sender and receiver, and the congestion in the network may potentially lead to congestion collapse.