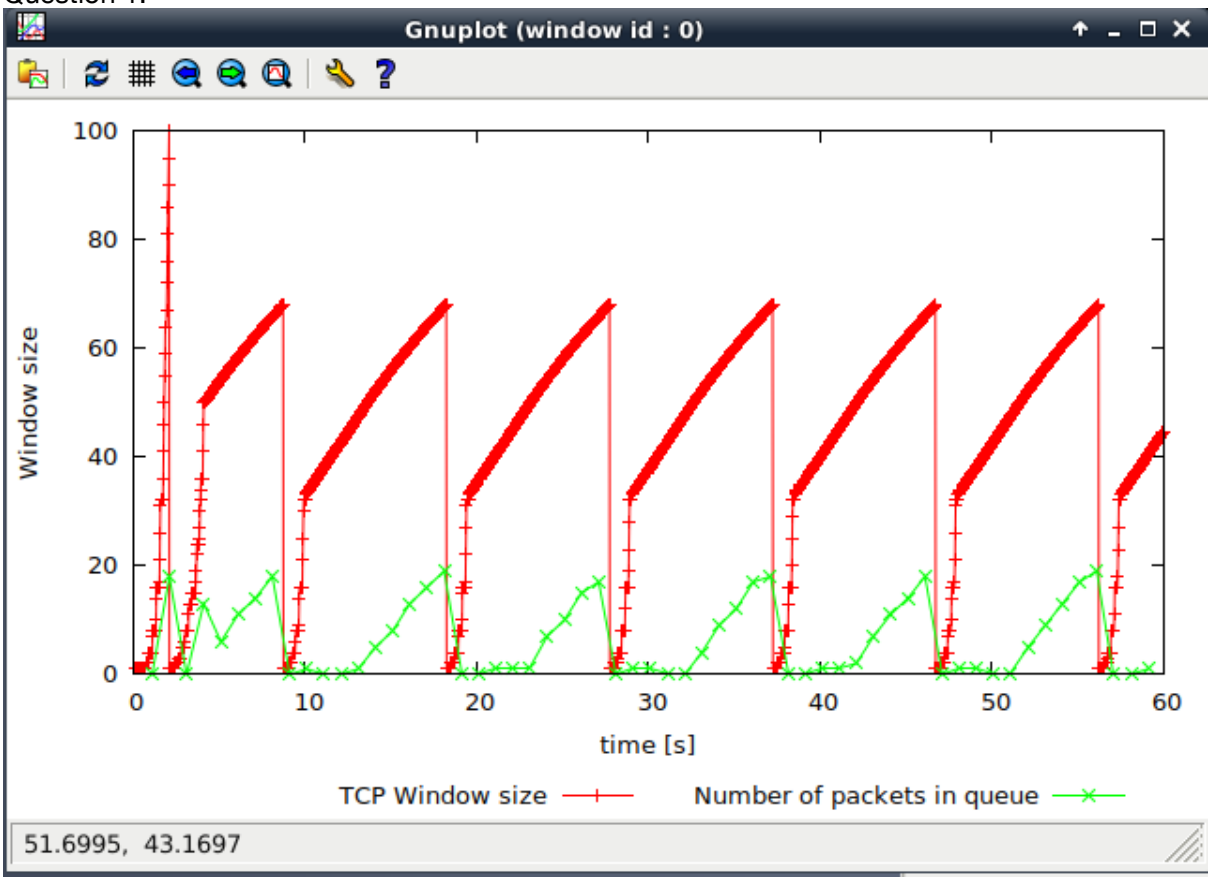
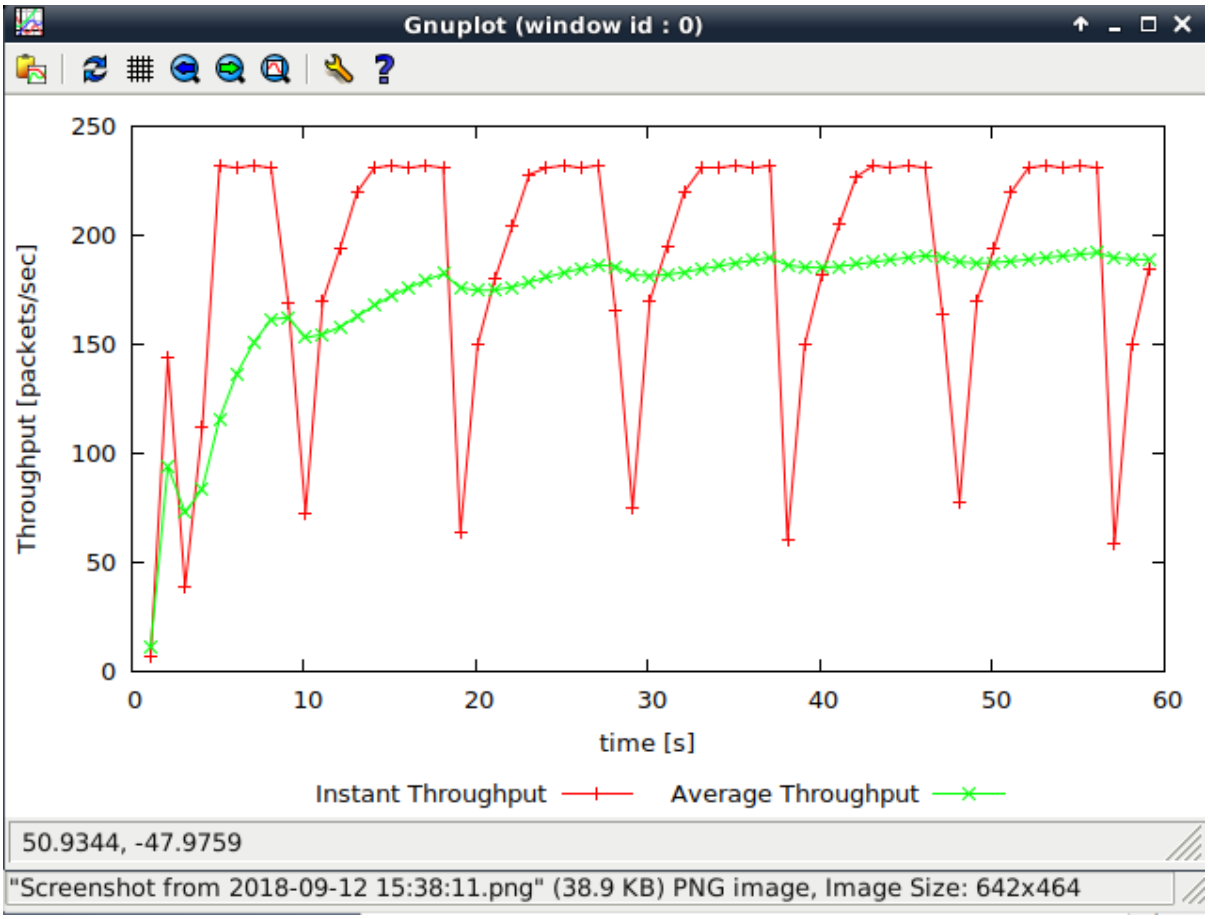
Exercise 1: Understanding TCP Congestion Control using ns-2

Question 1:



The maximum size of the congestion window that the TCP flow reaches is 100; The maximum size of the queue is only 20 packets.
Some of packets will drop because the packet congestion > the size of the queue.
When the congestion window reaches this value it starts slow-start mechanism and drop to 1 to restart and threshold to 1/2 the size of the window(50). And when it increases rapidly to the threshold and then transitions to congestion avoidance phase begin.
It will back to the slow start phase and repeat the phase from packet congestion to drop of packets.

Question 2:



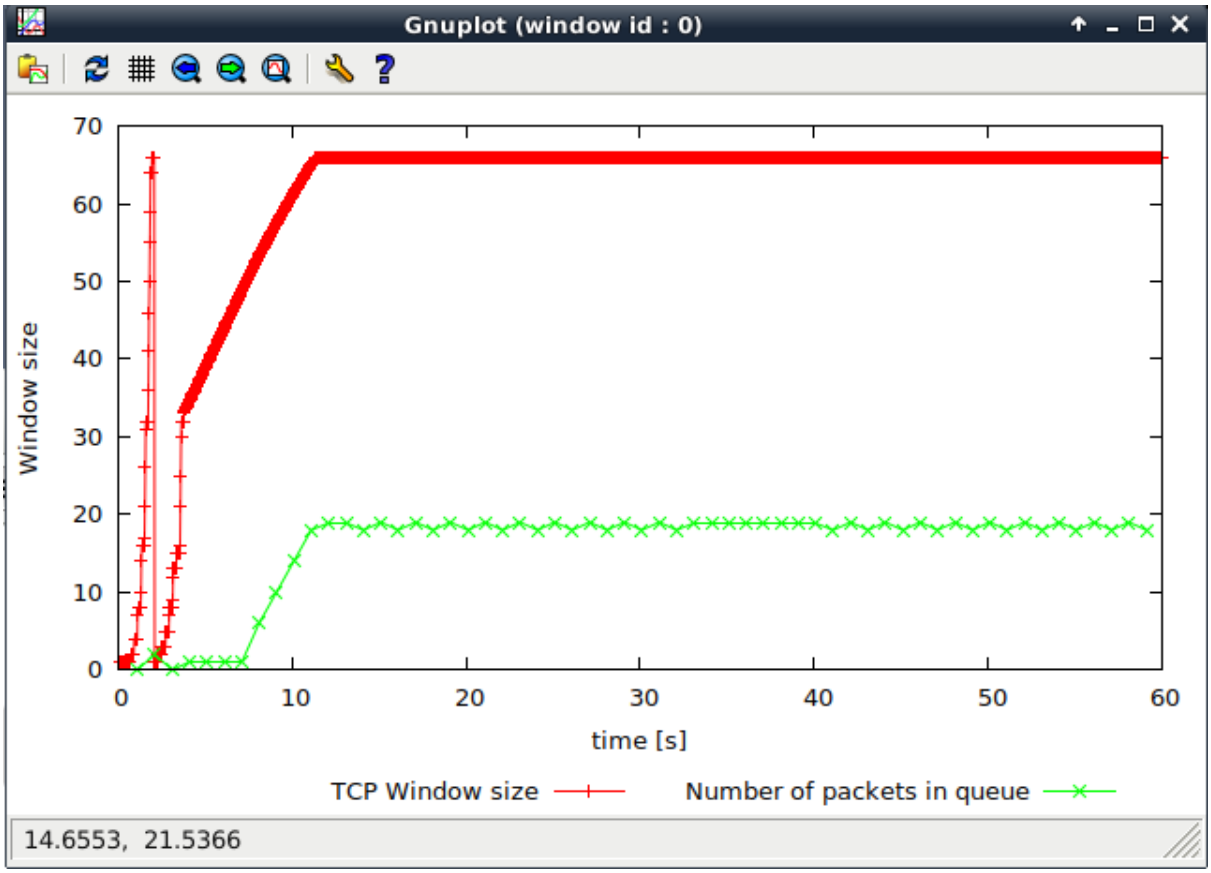
The average throughput of TCP in this case:

IP headers + TCP headers = 20 + 20 = 40 Bytes.
The payload = 500 Bytes.
Average throughput of TCP packets = 190 packets.

So the throughput is : (500 + 40) bytes * (8 bits per bytes) * 190 packets = **820800 bps**

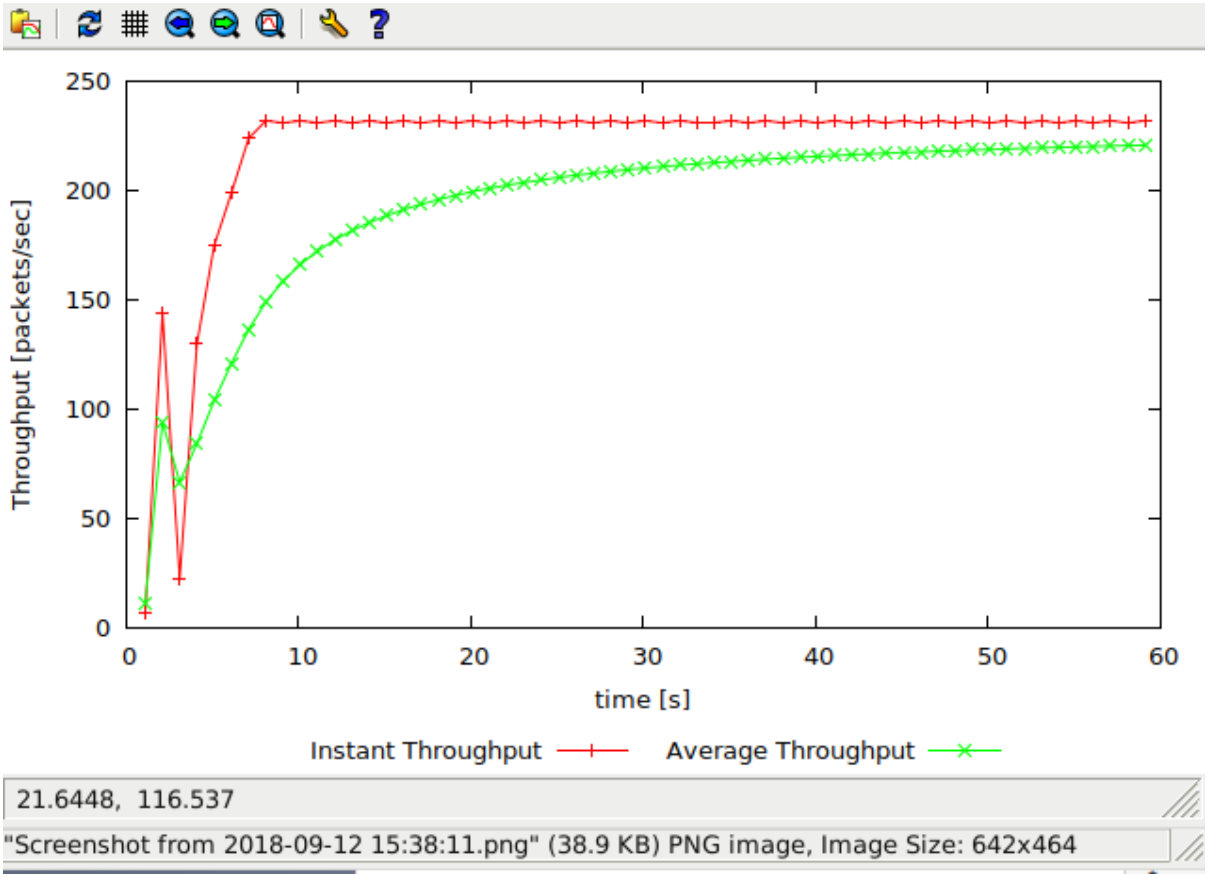
If neglect any headers: : (500) bytes * (8 bits per bytes) * 190 packets = **760000 bps**

Question 3:



When the max congestion window size is less than 66, oscillations stop when the first slow start happen, after that there will be not packets drop and keep transfer stable, and the queue never gets full. In this case, packets are not dropped. But when it goes over 66 window, some packets may drop, and congestion voidance mechanism start.

The value of the maximum congestion window is 66;



At this point:

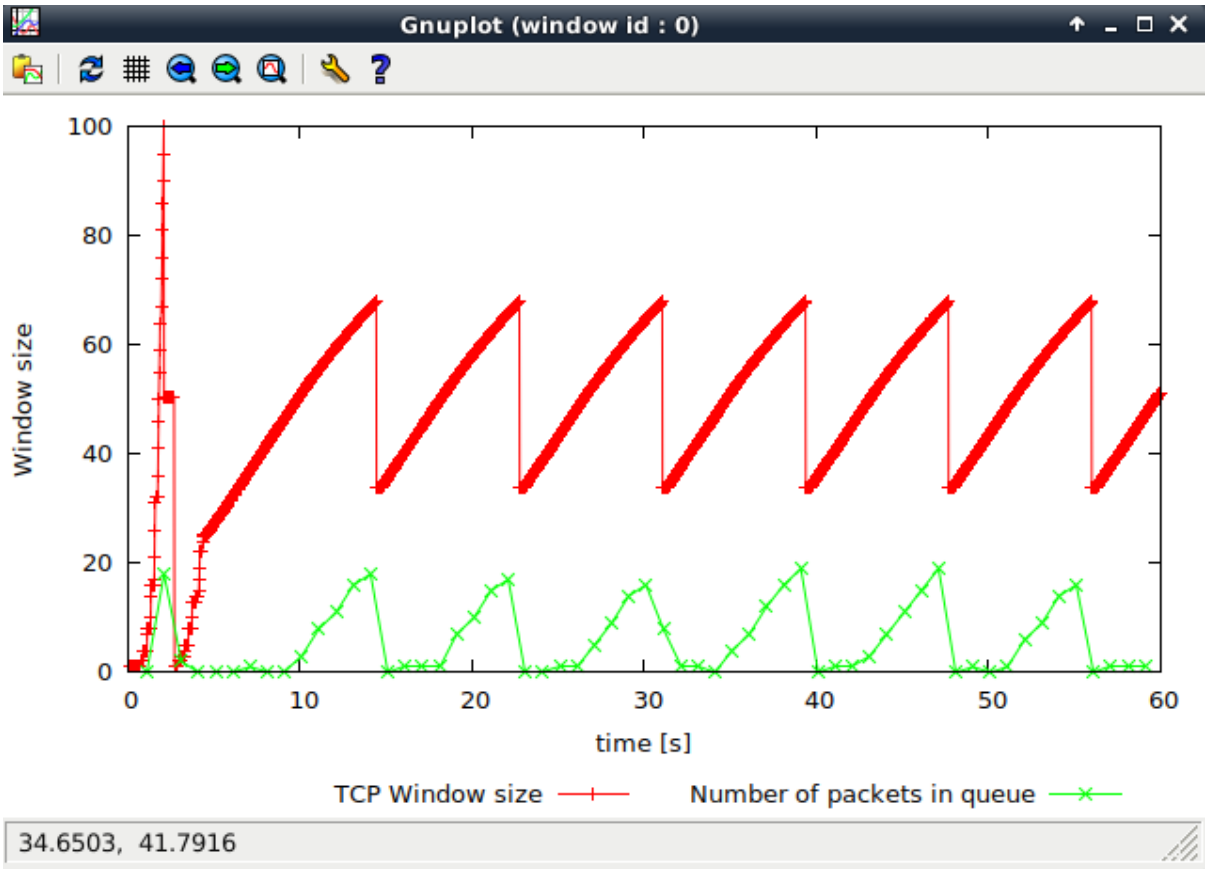
IP headers + TCP headers = 20 + 20 = 40 Bytes.
The payload = 500 Bytes.
Average throughput of TCP packets = 225 packets.

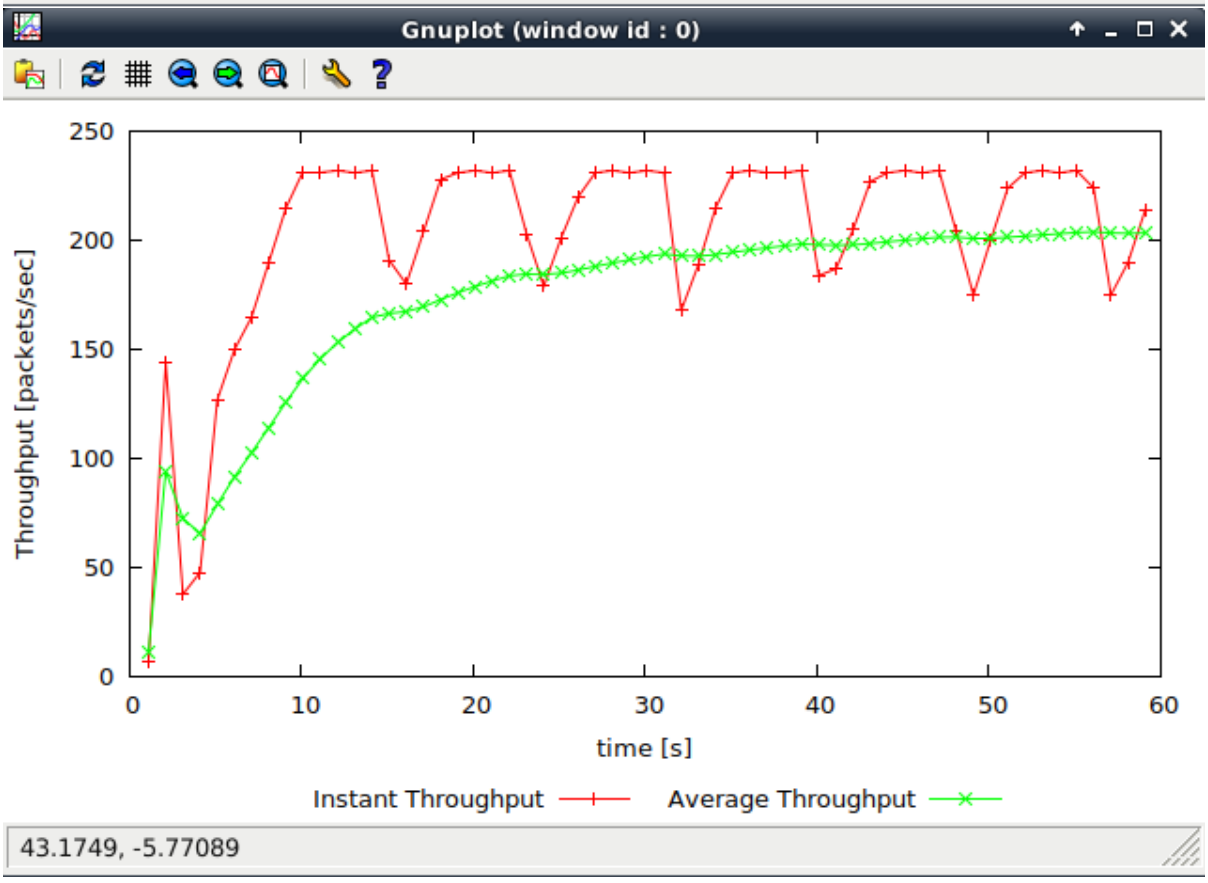
So the throughput is: (500 + 40) bytes * (8 bits per bytes) * 225 packets = **972000 bps**

Link capacity: 1Mbps = **1000000** bps

The actual average throughput is slightly less than the 1Mbps link capacity.

Question 4:


Gnuplot (window id : 0)
Window size
time [s]
TCP Window size
Number of packets in queue
34.6503, 41.7916


Gnuplot (window id : 0)
Throughput [packets/sec]
time [s]
Instant Throughput
Average Throughput
43.1749, -5.77089

In TCP Tahoe:
When a loss occurs, half of the current CWND is saved as ssthresh and slow start begins again from its initial CWND. When it reaches the ssthresh, TCP changes to congestion avoidance algorithm which is a a linear increase of the CWND.

In TCP Reno:
When a loss occurs, half of the current CWND is saved as ssthresh and as new CWND, it skip slow start and go directly to the congestion avoidance algorithm (fast recovery) and repeats this pattern.

In TCP Tahoe there are several times the congestion window go back to zero as long as a loss occurs.
However, in TCP Reno there is only one time the congestion window go back to zero when the loss occurs for the first time and half of the current CWND is saved as ssthresh and as new CWND. Since then it start fast recovery and never back to zero.

Average throughput:

In TCP Tahoe, the average throughout is = **820800 bps**
In TCP Reno, the average throughout is:

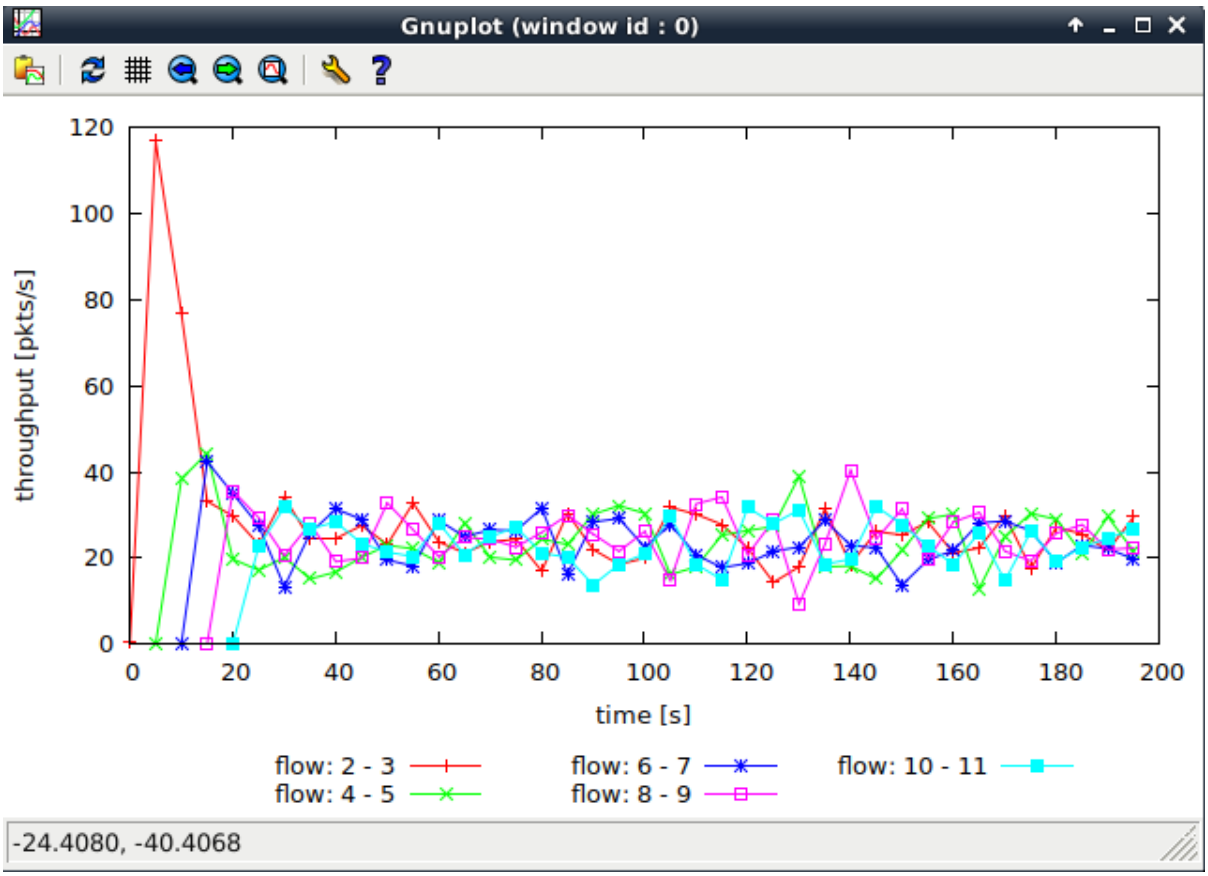
IP headers + TCP headers = 20 + 20 = 40 Bytes.
The payload = 500 Bytes.
Average throughput of TCP packets = 203 packets.

The throughput is : (500 + 40) bytes * (8 bits per bytes) * 203 packets = **876960 bps**

The TCP Reno has higher average throughput than TCP Tahoe.

Exercise 2: Flow Fairness with TCP

Question 1:



I think each flow gets an equal share of the capacity of the common link.

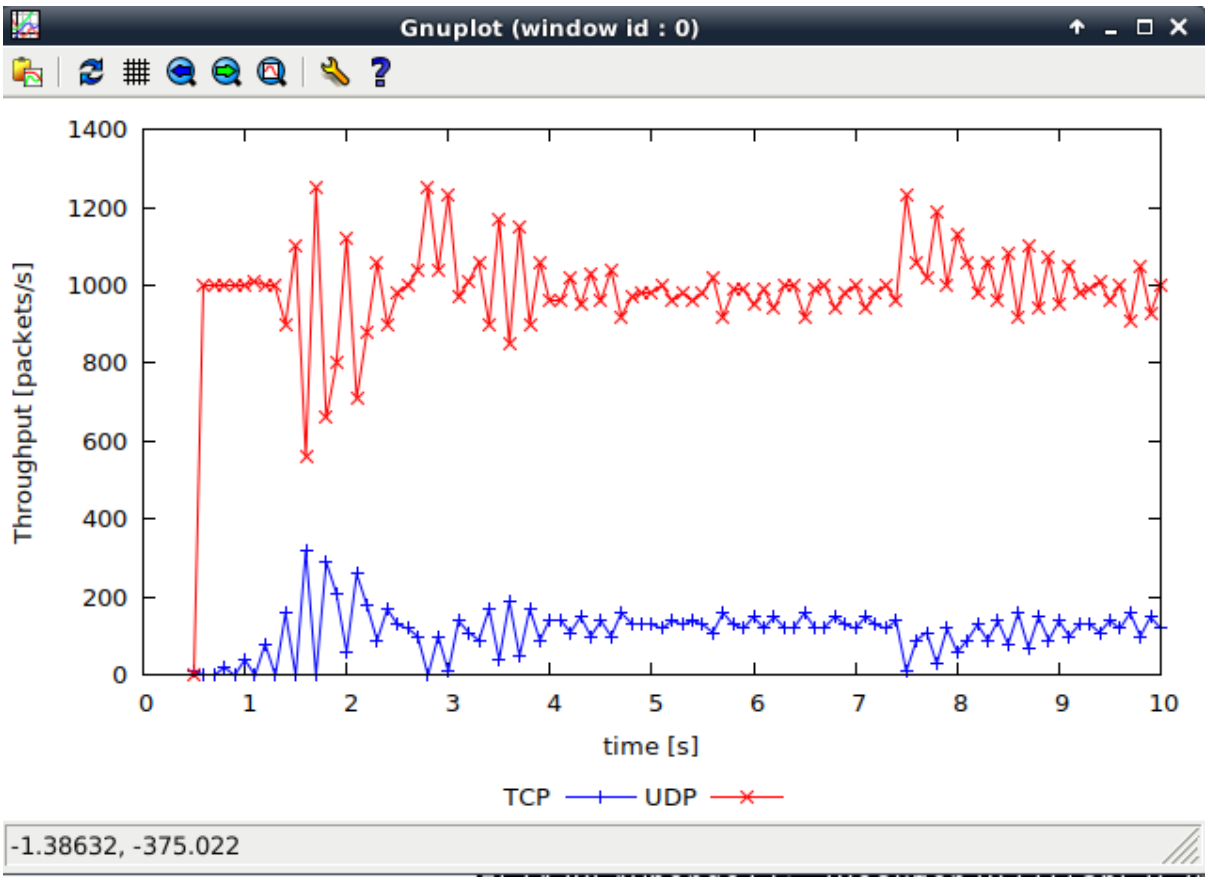
Although, the first of few flows take more capacity at the beginning as others not connect yet. When all flows get connected, they get stable and fairly equal share with some fluctuations in throughput. It because when multiple flows share a bottleneck link, the AIMD strategy start to control flows window size and share the same manner.

Question 2.

Throughput for the first few flows is much higher and then it decreases significantly after other flows get connected. This is because the flows creates congestion on the link and then the congestion control mechanisms start. And then they all get stable and fairly equal share with some fluctuations in throughput as I mention above. I think it is fair to adjust when new connections come in and it allows all flows to get equal service.

Exercise 3: TCP competing with UDP

Question 1:



The red colour is UDP, the blue colour is TCP.

I expect the UDP throughput is higher than the TCP throughput since UDP does not have any congestion control like TCP, it means UDP won't decrease its sending rate and it transfers faster.

Question 2:

UDP is faster than TCP because UDP does not have acknowledge packet (ACK) instead of TCP acknowledges a set of packets, calculated by using the TCP window size and round-trip time (RTT). In addition, there is no congestion control feature in UDP while TCP has congestion control. UDP transmits packets at a constant rate and it neglects the packet loss.
So, UDP can starve TCP flows in same bottleneck links.

Question 3:

File transfer:

Using UDP for file transfer::
Advantages: Faster, and no restriction to keep transmitting.
Disadvantages: Unreliable because it has no ACK packet and protocol like TCP to guarantee the reliable data transfer so it may loss data during the transmission.

If everyone using UDP instead of TCP, the internet would be congestion collapse because there is not any congestion control to control their sending rate when congestion happen. Lots of data flow into internet and internet would be congestion collapse.