ASSIGNMENT 3

Name-Sibasish Rout

Entry No- 2020CS10386

TASK 1

Steps used: -

1) First, we have used ns-3.29/examples/tutorials/seventh.cc as a basic template.
2) Set the appropriate values of application data rate, channel data rate, channel delay , packet size and error rate in the appropriate functions of seventh.cc.
3) We configure the TCP congestion control protocol to be used by using the following line

   Config::SetDefault ("ns3::TcpL4Protocol::SocketType", StringValue (**protocol**));

   The value of protocol can be "ns3::TcpWestwood", "ns3::TcpVeno", "ns3::TcpVegas", "ns3::TcpNewReno" according to the question.

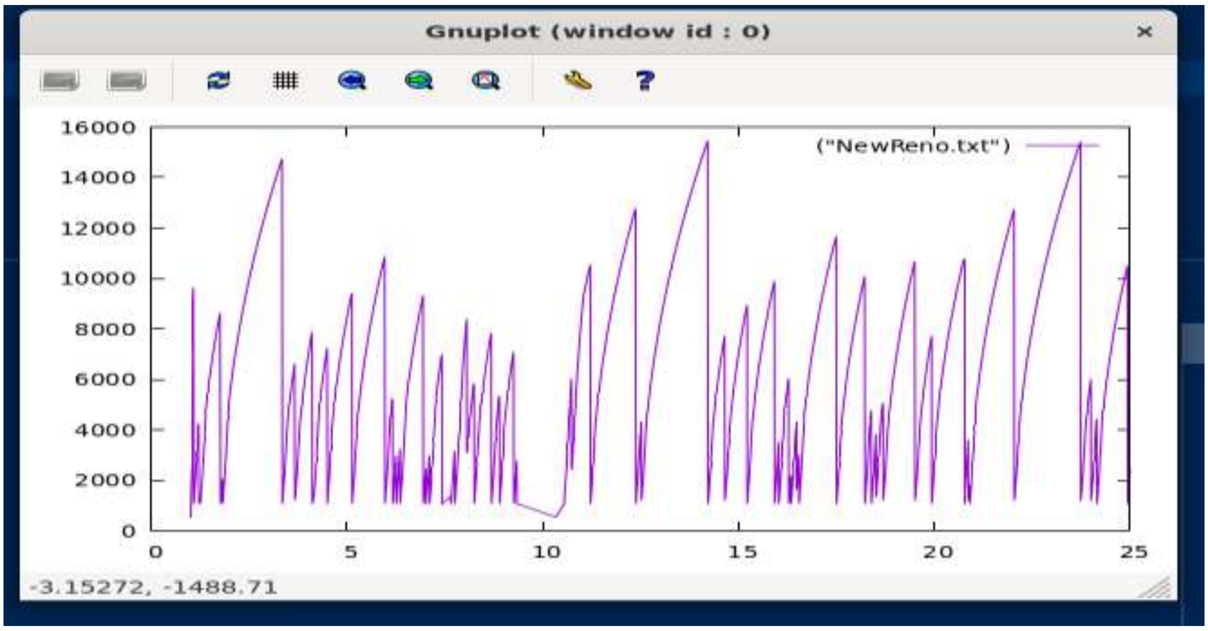
4) Declare two global variables which store the maximum value of congestion window and no of packets dropped until that point of time.
5) Whenever there is change in the congestion window size we check for change in the value of maximum congestion window size . This is implemented in the CwndChange function. Also whenever there is a change in the congestion window size we can store the pair consisting of time and value of cingestion window in a vector. We will later use these values to plot the graph. After execution is over we write these values to a .txt file.
6) Whenever there is a packet drop we increase the value of the number of packet drops
7) Finally we print these vales and plot the graph from the newly created .txt file using gnuplot

Question 1

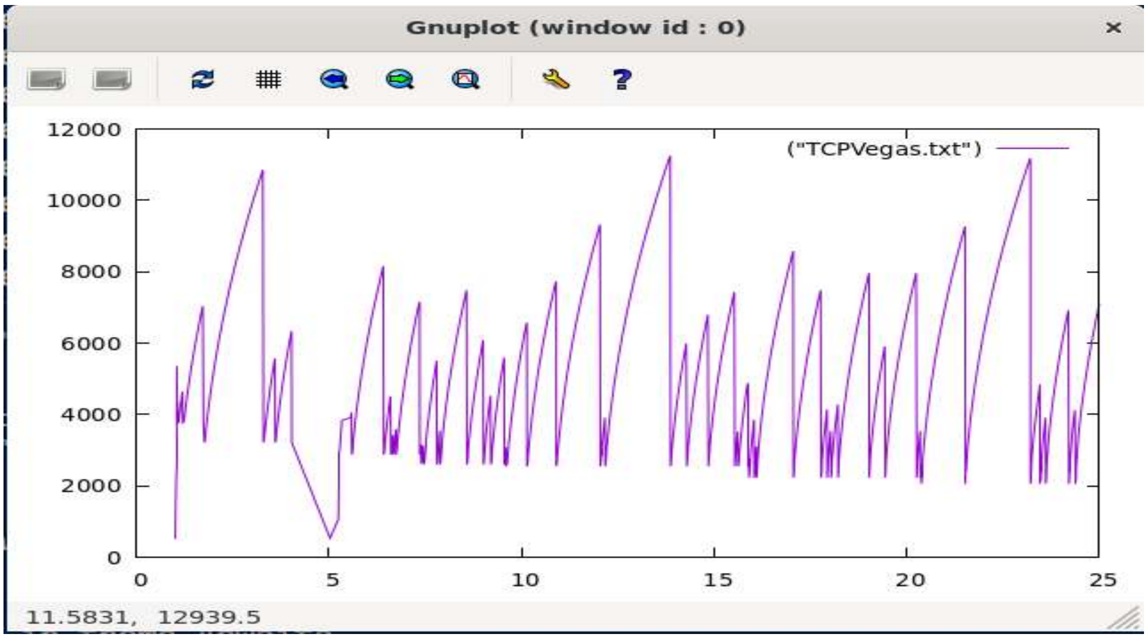
Protocol name	Max window size	No of packet drops
NewReno	15462	58
Vegas	11252	58
Veno	15462	59
Westwood	15471	60

Question 2

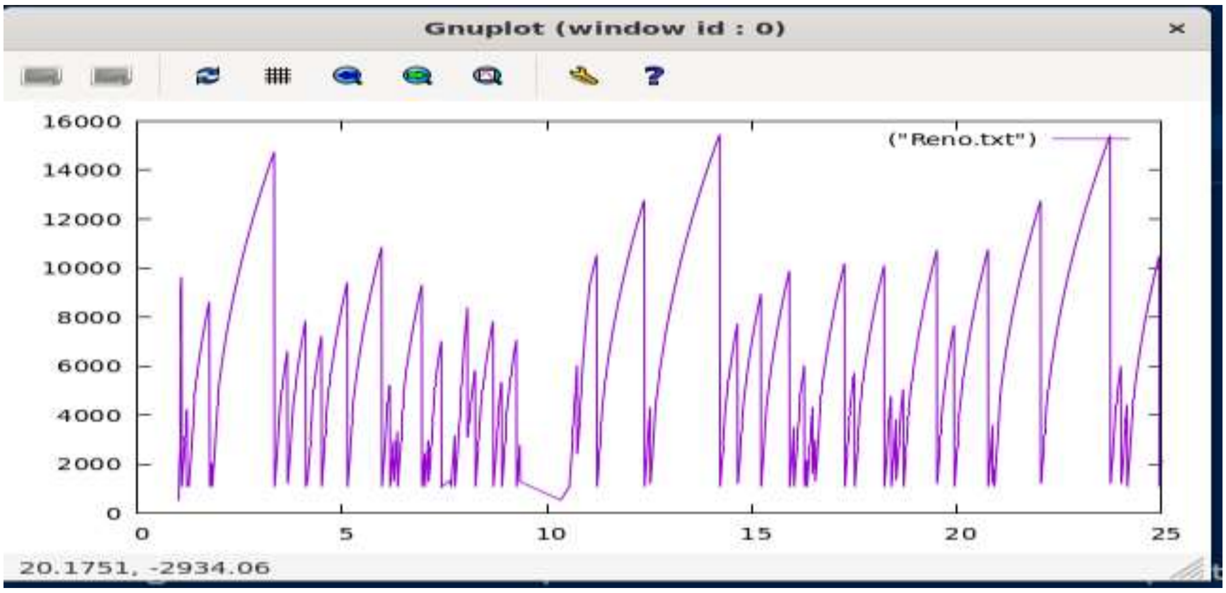
NewReno



Vegas

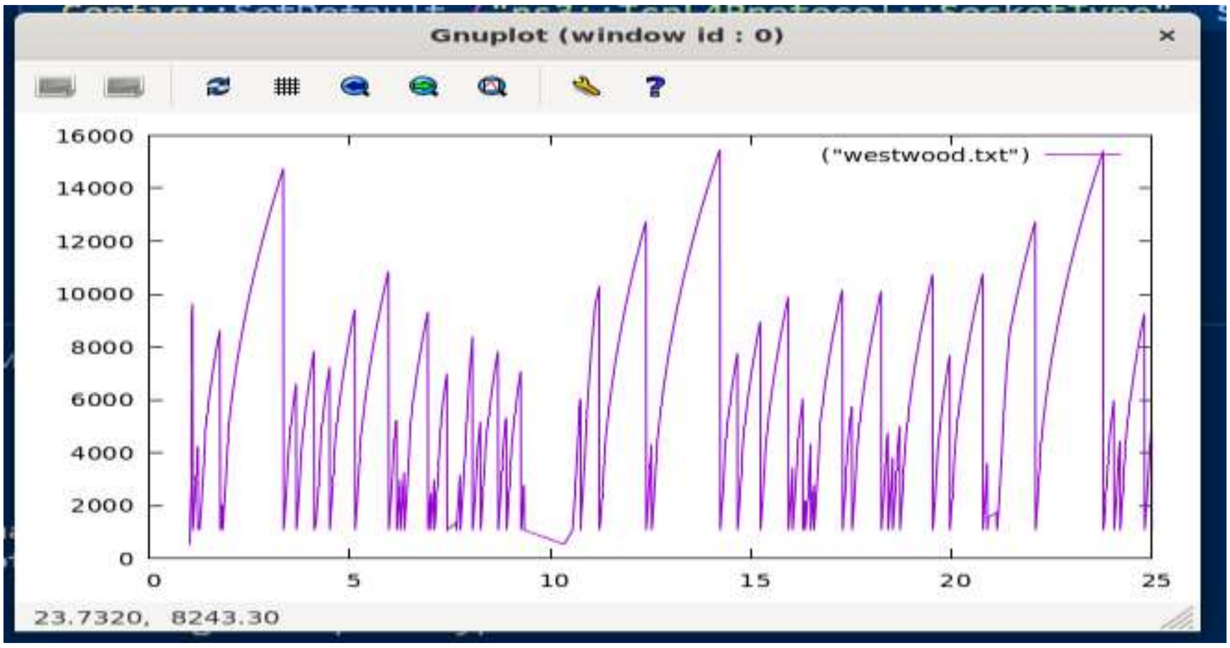


Veno



20.1751, -2934.06

Westwood



23.7320, 8243.30

For all the 4 graphs given above y axis denotes the size of congestion window and x axis denotes time in seconds.

Question 3

From the graph we can draw the following observation

TCP NewReno functions very similar to TCP Reno . Whenever there is a timeout the congestion window size is set to 1MSS and whenever 3 duplicate acknowledgements are received it reduces the congestion window size by half. Additionally it can detect multiple packet as NewReno can distinguish between partial and full acknowledgement. Thus the congestion window size is relatively high. It has slow start , fast retransmit and congestion avoidance phases similar to TCP Reno. These trends are easily observed in the graph.

TCP Vegas functions on the basis of delay based approach to control congestion. Thus it increases the congestion window if RTT is almost same as uncongested RTT and decreases the congestion window If the RTT is very much larger than uncongested RTT. Due to this proactive measure the congestion window size rarely falls to 1MSS as observed in the graph and also the maximum congestion window size is also lower than the other three protocols.

TCP Veno is TCP enhancement for wireless networks and uses network congestion level data to check whether packet loss is due to congestion or due to bit errors . It modifies the AIMD protocol of TCP Reno by adjusting the slow start threshold according to the perceived network congestion. As in the wred topology we have used the chance of bit errors is very less the graph is similar to that of TCP NewReno.

TCO Westwood is a sender side only modification of TCP NewReno. Here it looks at the rate of received Acknowledgement to decipher the congestion in the network and adjust the threshold and congestion window parameters . It is slightly more efficient than TCP NewReno and is more suitable for cases having large number of nodes. As the number of nodes here is 2 , the graphs are similar.

Question 4

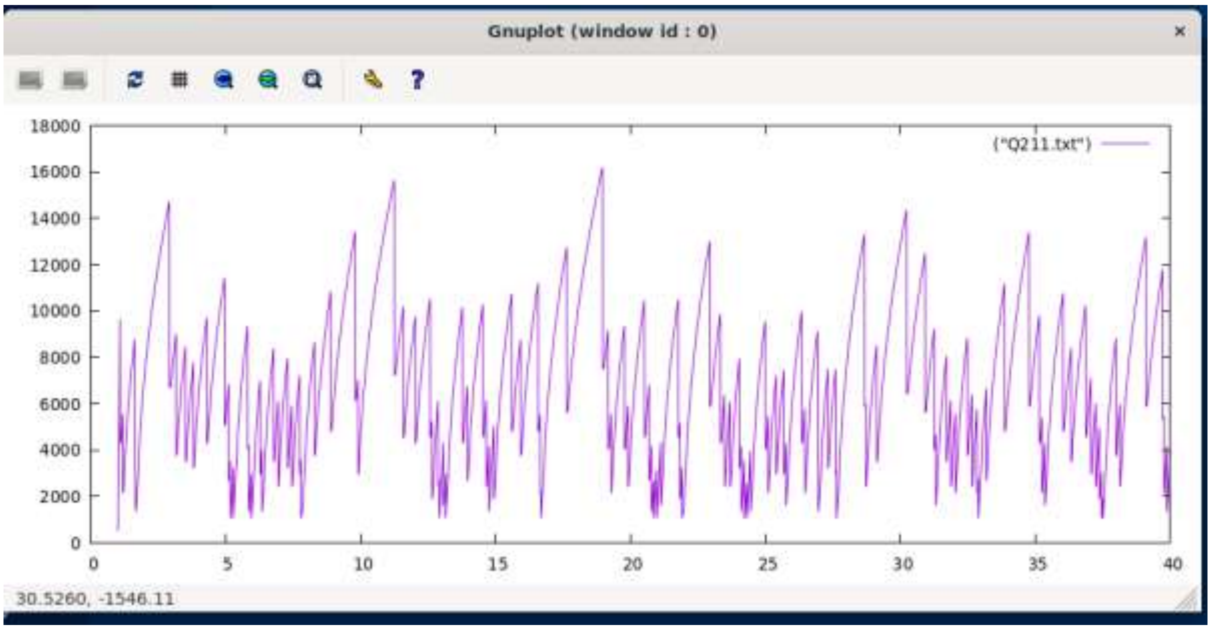
TCP BBR was developed for google cloud. The problems with the protocols seen above (like NewReno , Veno, Westwood) is that they overreact to congestion by drastically halving the congestion window size even though the congestion might occur due to transient burst of traffic. BBR instead use recent measurements of the networks delivery rate and RTT to control the rate of sending data and maximum amount of data to allow in the network at any time. This protocol and TCP Vegas are very close but BBR sends the data at 1.25 times the current data rate and check if the bandwidth has increased making it much faster (exponentially faster) than TCP Vegas. This enables the user to use the optimal part of bottleneck bandwidth efficiently.

TASK 2

The code is very much similar to code for Part 1 . Here we have only set the vale of protocol used to be TCP NewReno and changed the channel data rate and application data rate values

a) Application data rate=5Mbps

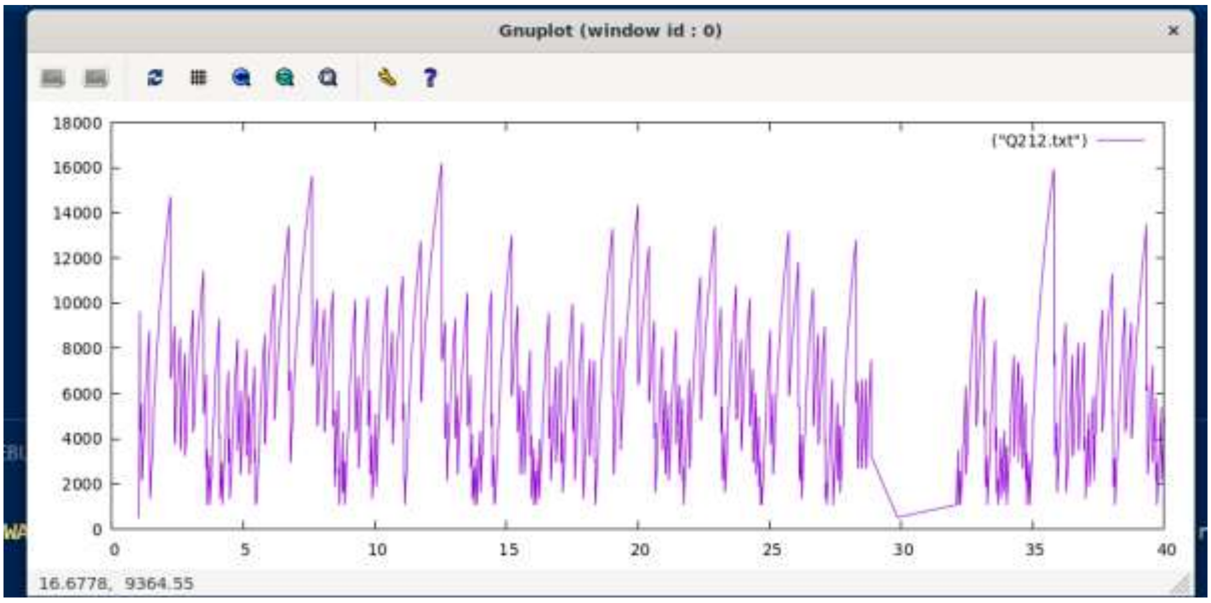
Channel Data Rate=3Mbps



max window size: 16206

no. of packets dropped: 131

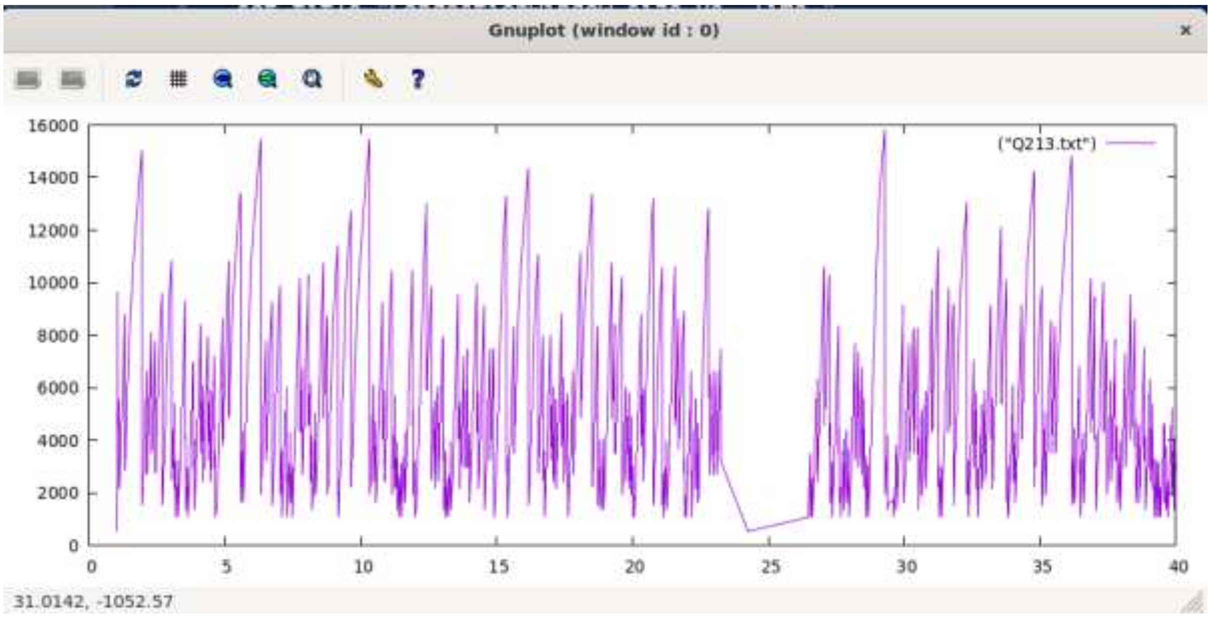
Channel Data Rate=5Mbps



max window size: 16206

no. of packets dropped: 194

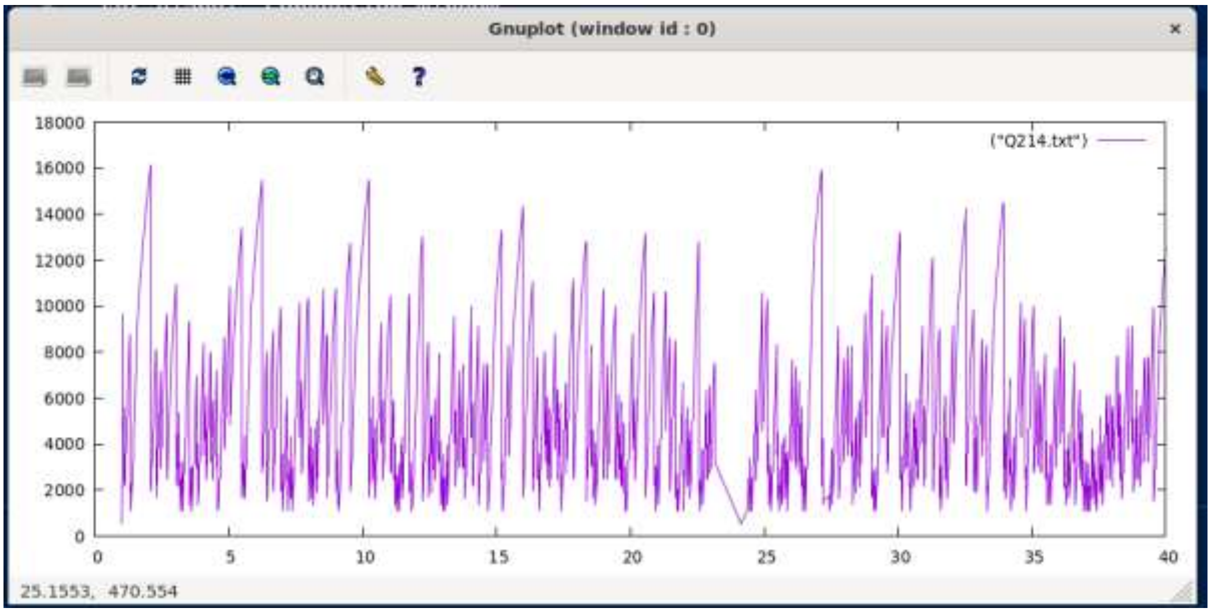
Channel Data Rate=10Mbps



max window size: 15849

no. of packets dropped: 246

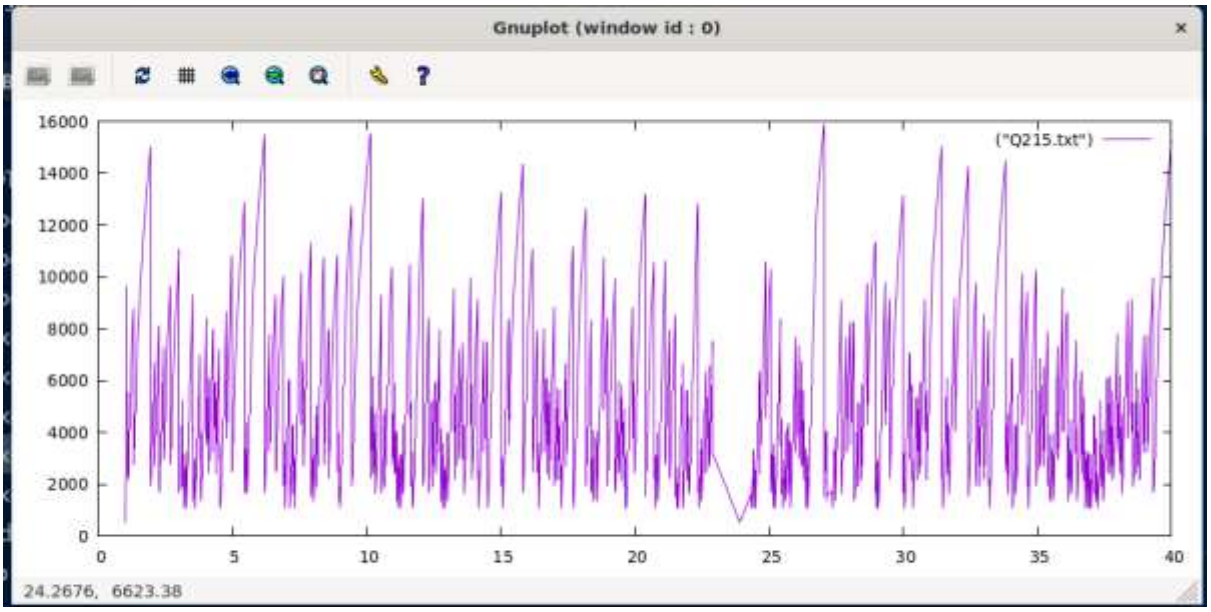
Channel Data Rate=15Mbps



max window size: 16162

no. of packets dropped: 260

Channel Data Rate=30Mbps

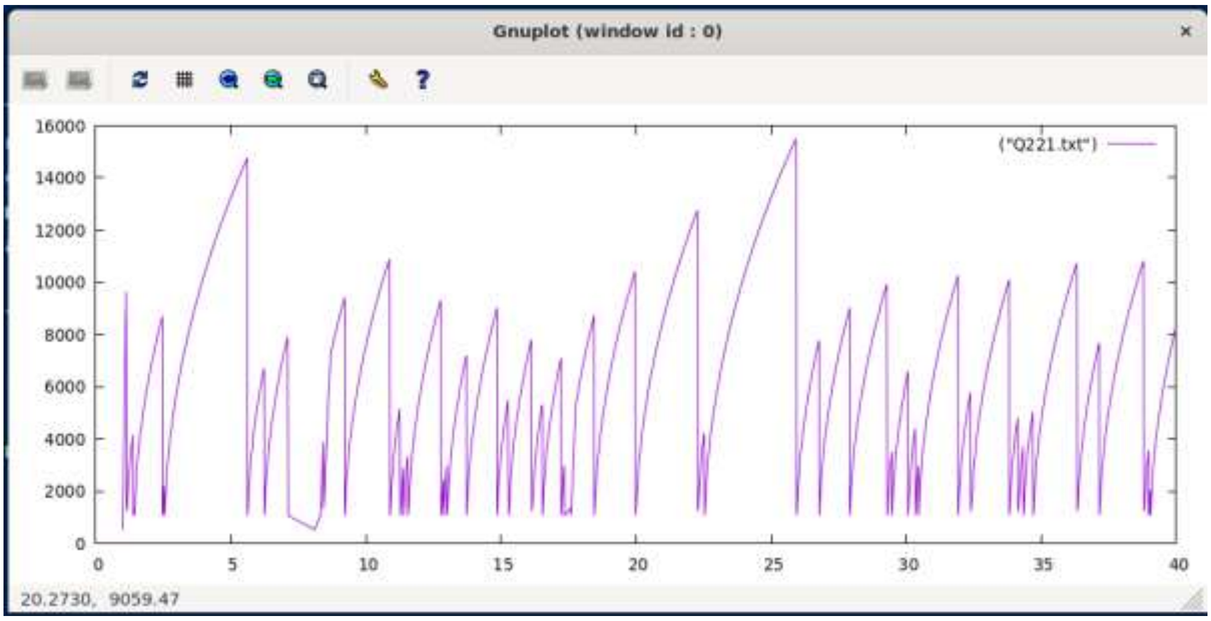


max window size: 15867

no. of packets dropped: 259

b) Channel data Rate 4Mbps

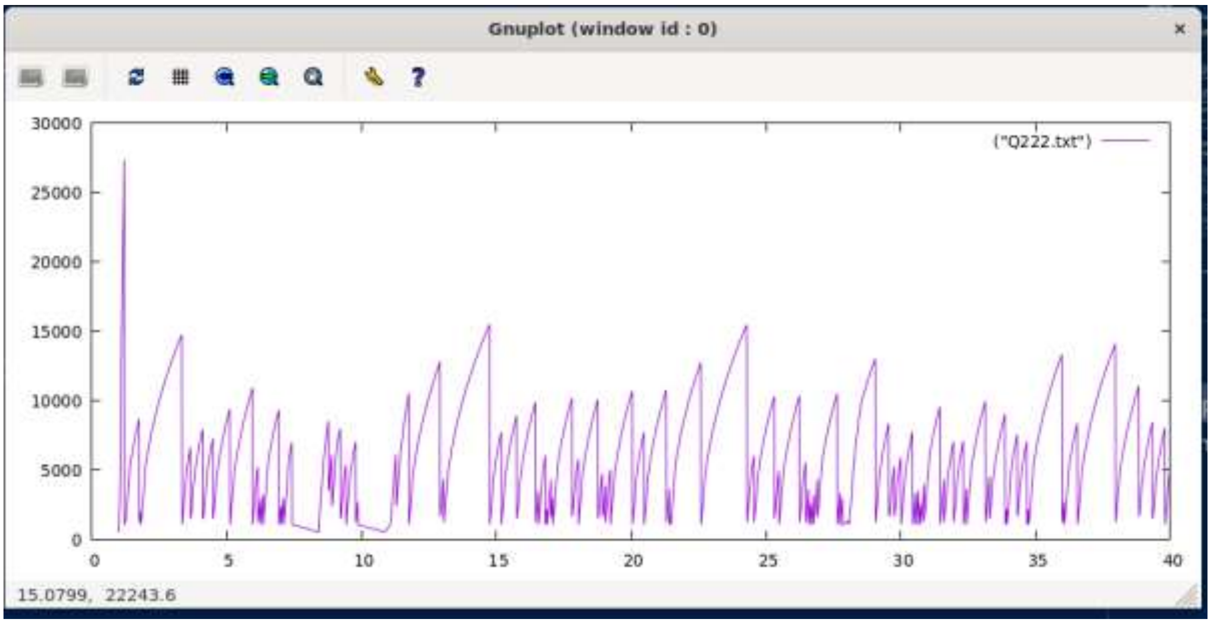
Application data rate 1Mbps



max window size: 15507

no. of packets dropped: 51

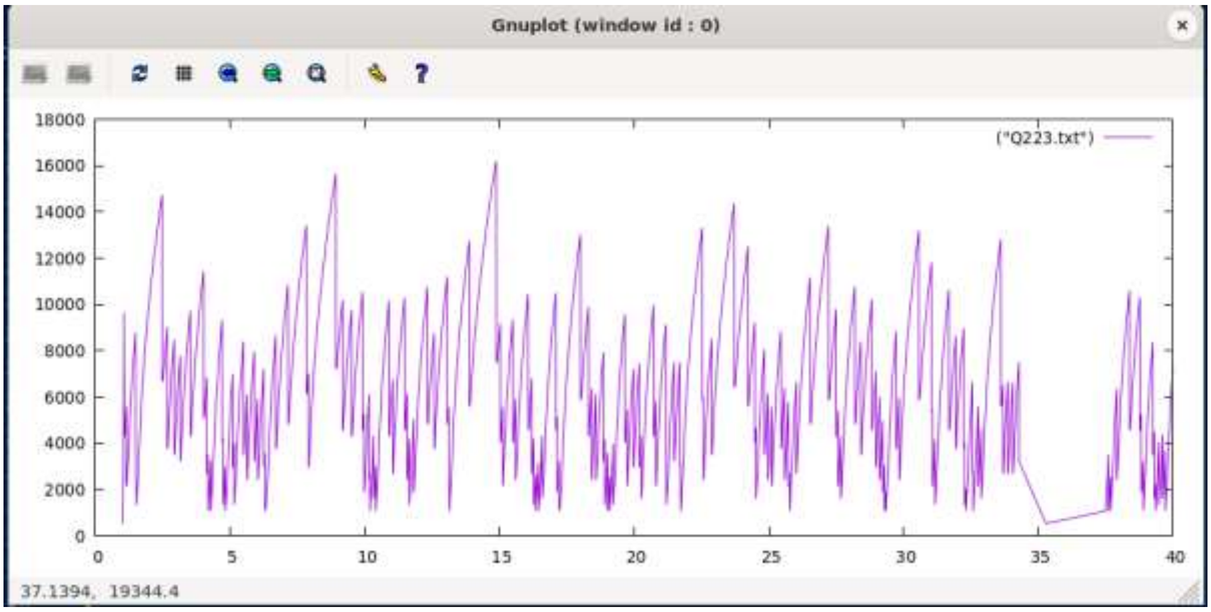
Application data rate 2Mbps



max window size: 27336

no. of packets dropped: 100

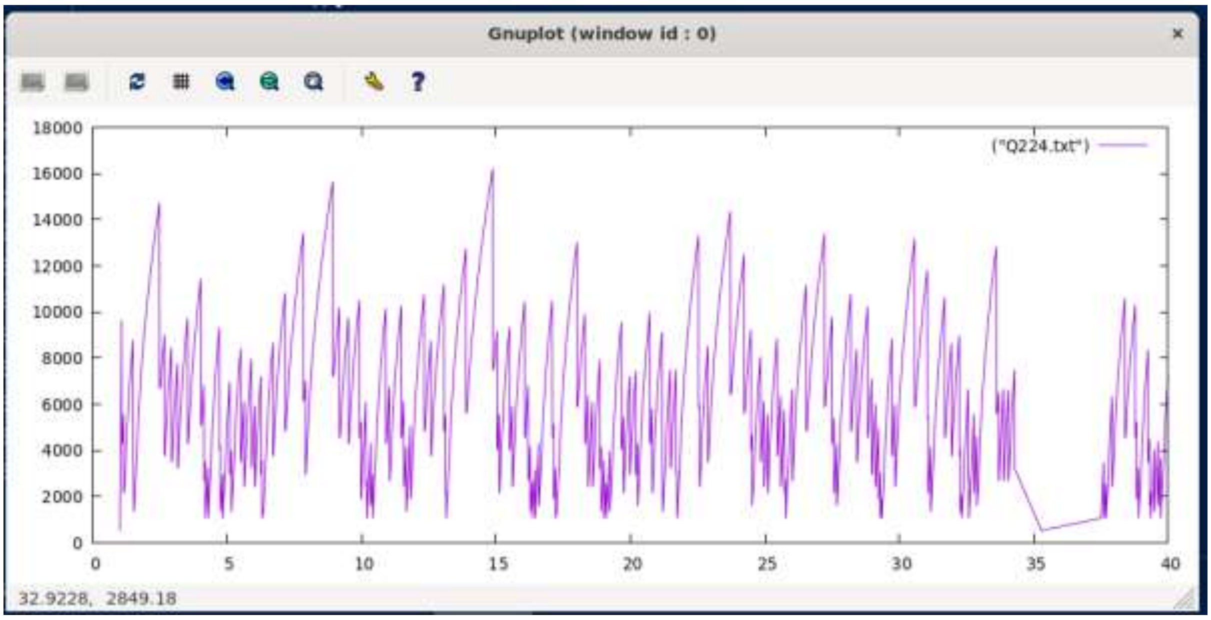
Application data rate 4Mbps



max window size: 16206

no. of packets dropped: 162

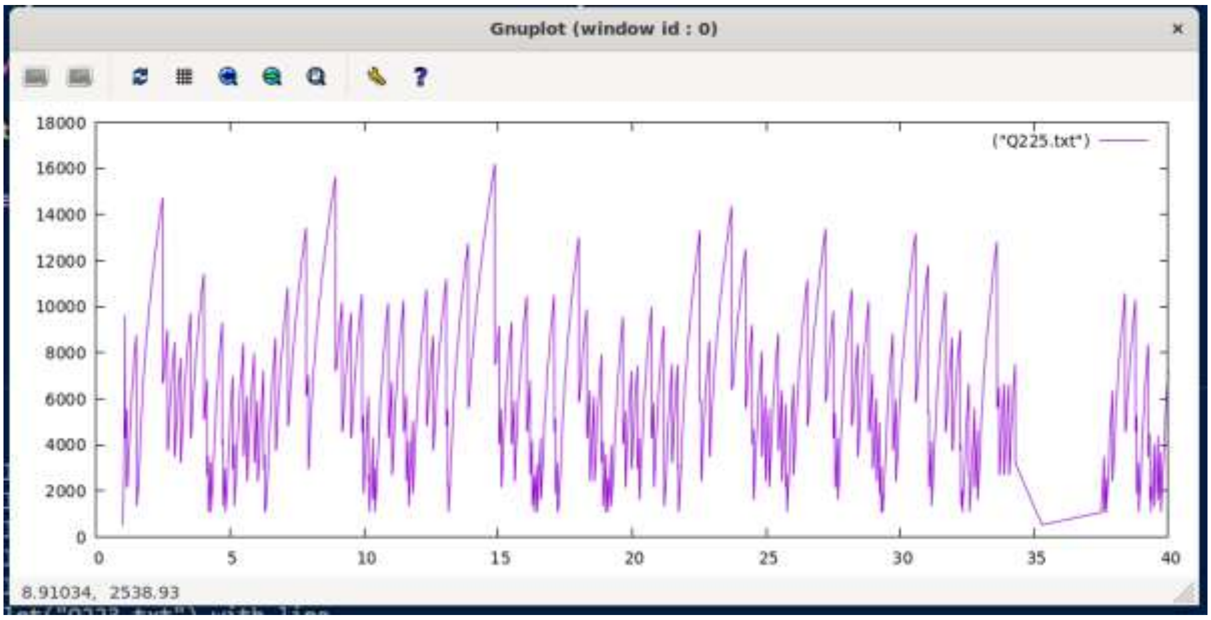
Application data rate 8Mbps



max window size: 16206

no. of packets dropped: 162

Application data rate 12Mbps



max window size: 16206

no. of packets dropped: 162

For all the 10 graphs given above y axis denotes the size of congestion window and x axis denotes time in seconds.

Q1) How does application data rate affect congestion window size?
When the application rate is less than the channel data rate there is minimal chance of packet drops and hence all the data send by the application to the channel gets transferred fast so that the network is in congestion avoidance phase for large duration. Here number of packet drops is less and hence average value of congestion window in general is higher. This is indicated by the less number of spikes in congestion window vs time graph.
When the application data rate becomes equal to the channel data rate packet drops start to happen and spikes become closer. When application data rate is greater than channel data rate the number of packet drops is high due to fixed size of queues and hence there is a large number of spikes in congestion window size vs time graph. Thus, in general the congestion window size is lower.
Further increase in application data rate does not have any significant effect on congestion window size.

Q2) How does channel data rate affect congestion window size?
When the channel data rate is greater than the application data rate there is queueing at the receiver side and packet drops happen, but the congestion window size is significantly higher and the number of packet drops are less. Thus, the number of spikes in the graph is also less.

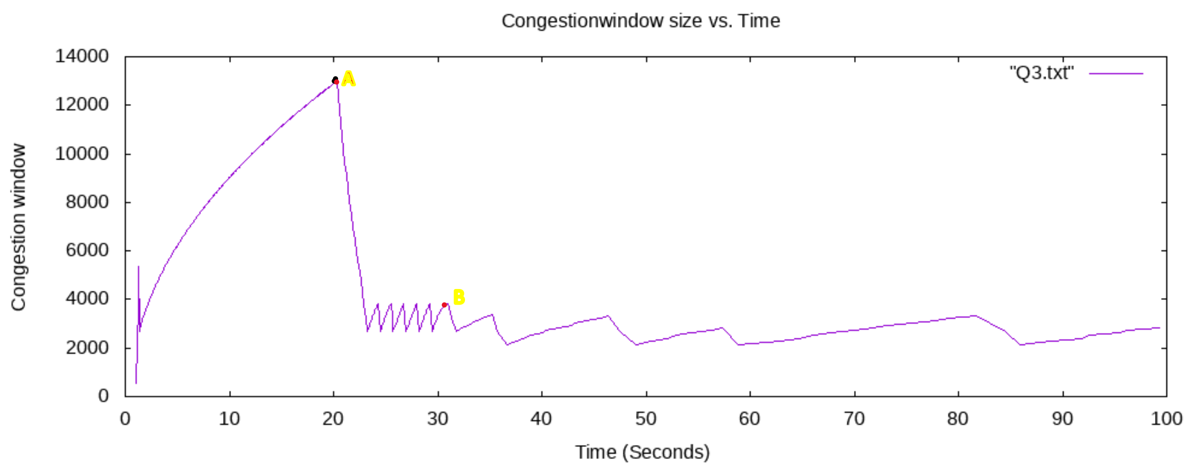
When channel data rate is much lower than application data rate there is a large number of packets drops at sender side as the application keeps on sending data, but the channel is unable to deliver the data. This results in packet drops at even low congestion window values. Thus, the congestion window size is comparatively lower and number of packet drops is also high. When the application data rate increases further now, the change in congestion window observed is not much.

Q3) What relation do you observe between channel data rate and application data rate?

Channel data rate and application data rate have opposing impacts on each other. Queueing happens at receiver side if channel data rate is lower than application data rate and queueing happens at the sender side if channel data rate is higher than application data rate. In both cases packet drop happens. When channel data rate is larger than application data rate the congestion window size is larger than the case when it is the other way round.

TASK 3

Question 1



Here I have used codes from seventh.cc as template as parts of first.cc, third.cc to write the code for this task. I have utilised examples of TCP and UDP to create the 5 nodes and connect them according to the topology given in the question.

Question 2

Before point A (at time 20 seconds) there was no congestion in the channel and application data rate is less than the channel data rate and hence the congestion window size increases as per the TCP Vegas protocol. At time 20s when the connection channel between node 2 and node 3 is full and hence the congestion window size decreases sharply.

At time 30s(at point B) the application data rate becomes more than the channel data rate and hence congestion window increases very slowly and cautiously so as to avoid packet drops and hence the time interval between the peaks increases. This continues happening till 100s.

Point when UDP channel established - A

Point when UDP channel rate increases- B

Question 3

The pcap files are included in the submission zip folder.

Files submitted :-

1)Question1.cc – code used for generating graphs in Task 1

2)Question2.cc -code used for generating graphs in Task 2

3)Question3.cc- code used for generating graph in Task 3

4)plotns3.plt – gnuplot script to plot data from .txt file

5)plotns3,sh - Bash file to plot data using gnuplot

6)Question3-0-0.pcap , Question3-0-1.pcap, Question3-1-0.pcap, Question3-1-1.pcap, Question3-2-0.pcap, Question3-2-1.pcap, Question3-3-0.pcap- These are 8 pcap files for each side of the 4 interfaces between the nodes in Part 3