Performance Analysis of TCP Variants

Vinay Nambiar
Northeastern University, MA, 02115
Email: nambiar.v@husky.neu.edu
NUID: 001256057

Kalpesh Shardul
Northeastern University, MA, 02115
Email: shardul.k@husky.neu.edu
NUID: 001224004

Abstract: TCP is a reliable mode of transport protocol for end to end delivery of packets under congestion. In this paper, we have performed three different types of experiments to test the different variants of TCP in a simple network topology. In these experiments we have tested drop rate, throughput and latency of TCP Tahoe, Reno, NewReno and Vegas under congestion, tested the fairness between a pair of TCP variants and the influence of DropTail and RED queuing disciplines on TCP Reno and TCP SACK.

I. Introduction

The possibility of occurrence of congestion is increasing rapidly with growth in networks. Earlier the size of sending window of TCP was determined by the receiver window thereby providing only flow control and no means of congestion control. To adjust with the growing networks many TCP variants were researched and developed thereafter namely:

1. TCP Tahoe

TCP Tahoe is one of the simplest of TCP variants. It does not have Fast Recovery state. At congestion avoidance phase, it treats triple duplicate ACKs same as timeout and it will reduce the congestion window to one and enter slow start phase.

2. TCP Reno

TCP Reno differs from TCP Tahoe since on receiving three duplicate ACKs it halves the congestion window, performs fast retransmit and enters the fast recovery state. If a timeout occurs it behaves in the same way as Tahoe. TCP Reno is good for loss of single packet but its performance deteriorates when multiple packets are dropped.

3. TCP NewReno

TCP NewReno is a modified version of Reno where a new data ACK is not enough to take the TCP out of fast recovery to congestion avoidance state. Instead it requires all the outstanding packets at the start of the fast recovery period to be acknowledged.

4. TCP Vegas

TCP Vegas calculates congestion at the starting stage on increasing RTT values of the packets. Depending on the current RTT values it detects congestion and resizes the queue size itself.

5. TCP SACK

The difference between TCP SACK and TCP NewReno is that when multiple packets are dropped from one window of data, SACK sender maintains information which packets is missed at the receiver and retransmits only those packets selectively.

II. Methodology

NS-2 is an object-oriented, discrete event driven network simulator which is primarily useful for simulating local and wide area networks. It can implement various network protocols such as TCP, UPD, FTP, Telnet, queuing disciplines such as Drop Tail, RED and routing algorithms such as Dijkstra, etc. NS-2 generates a trace file which is used in studying and analyzing the different TCP variants. The following three parameters are used to study the TCP variants:

1) Packet Drop Rate: It is the number of packets lost with respect to the total number of packets sent.

2) Throughput: It is the ratio of total number of packets received at the destination in a given time frame.

3) Latency: It is the round trip time for a packet.

T-Test is calculated for testing the difference between the two samples when the variance of two normal distributions is not known. Also, standard deviation was calculated for TCP variants to decide the stability between two samples. The NS-2 simulation was performed over the following network topology:

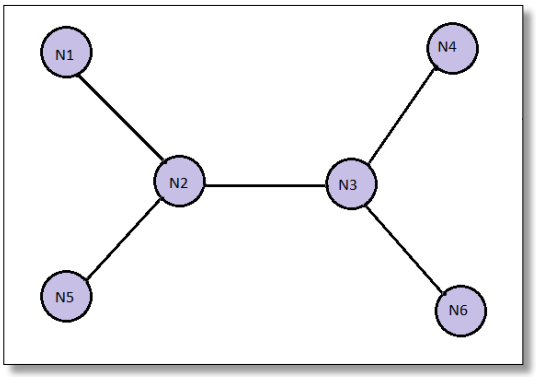

Fig1: Main network topology

The above topology has six nodes. Each node is connected by full duplex link with a bandwidth of 10 Mbps and a delay of 10ms.

III. Experiment 1: TCP Performance Under Congestion

In this experiment, packet drop rate, throughput and latency were calculated for all TCP variants (Tahoe, Reno, NewReno, Vegas and SACK). For this, a single TCP stream was added from source N1 to a sink at N4 and UDP based CBR flow was added at source N2 and a sink at N3. The CBR flow was linearly varied from 1 Mbps to a bottleneck capacity of 10 Mbps and the TCP flow performance was recorded. The performance of TCP stream was analyzed for packet drop rate, throughput and latency depending on varying CBR flow rate.

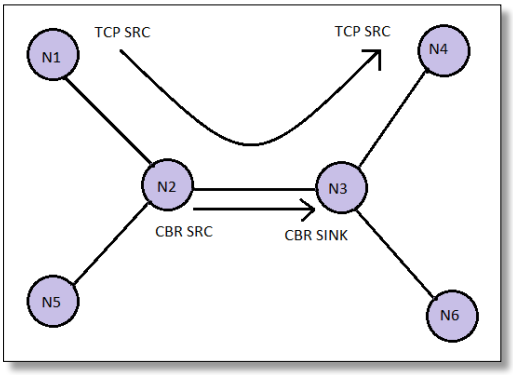

Fig 2: Experiment 1 Network Topology

The network topology used for experiment 1 is shown in figure 2. First, trace file is generated using NS-2 for every required CBR value and then graphs are plotted for packet drop, latency and throughput against CBR.

1) Throughput

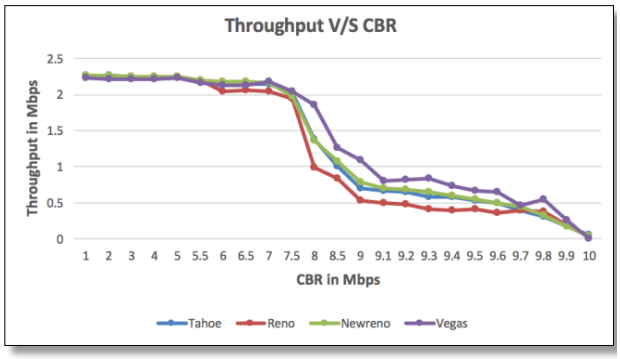

Fig. 3: Throughput V/S CBR

In the above graph, we can see that for all TCP variants, the throughput is decreasing with increase in CBR. For CBR which is less than 7.5, the throughput for all TCP variants is almost similar. While in Reno in terms of multiple packet loss it gives less throughput in comparison to Tahoe since it enters and exits the Fast Recovery state for each packet loss. Now, as the CBR starts increasing the average throughput for Vegas is better than others because of its feature to detect congestion at an early stage by the increasing RTT values. It than starts transmitting less data for congestion avoidance. Hence, due to Vegas's pre detection of congestion it gives better throughput than others.

Furthermore, we conducted T-Test of Vegas with respect to TCP Tahoe, Reno and NewReno and the results are as follows:

T- Test of Vegas over Tahoe = 0.407628913

T- Test of Vegas over Reno = 0.753559685

T- Test of Vegas over NewReno = 0.335477507

Since, the T-Test values are positive, we can conclude that Vegas is better in comparison to TCP Tahoe, Reno and NewReno.

2) Packet Drop Rate

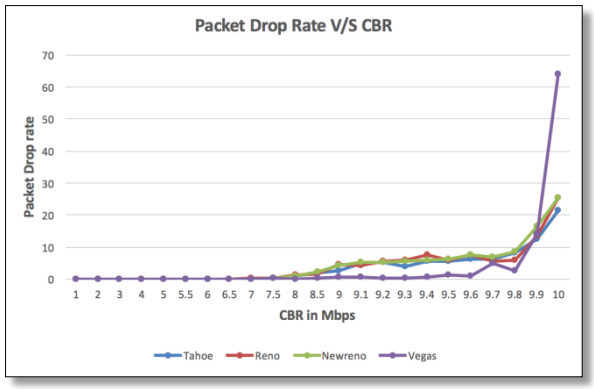

Fig. 4: Packet Drop Rate V/S CBR

The above graph is the CBR against the packet drop rate. Till a CBR value of 7.5, the drop rate for all TCP variants is very less. As, the CBR increase further, the drop rate for all TCP variants starts to increase, but, for Vegas it is the least. But, at a CBR value of around 9.8, the packet drop rate of Vegas suddenly shoots up to a high value. This is because at the initial stage the packet loss is very less and due to this the window size increase and hence, more packets are transmitted. So when the queue is filled, the arriving packets are dropped and the window size is reduced to half. Such changes are seen in another TCP variant as well and Tahoe, Reno and NewReno are behaving in almost similar manner. So in terms of packet drop rate, TCP Vegas has better result than

Tahoe, Reno and NewReno.

3) Latency

The latency for all TCP variants Tahoe, Reno, NewReno and Vegas are almost equal. Thereafter, the latency keeps on increasing for Tahoe, Reno and NewReno. In case of Vegas, as it detects congestion at an early stage, it will queue the packets when the received RTT is greater than base RTT. Due to this, Vegas have very low latency. Hence, in this experiment as well, Vegas have the best performance amongst all other TCP variants.

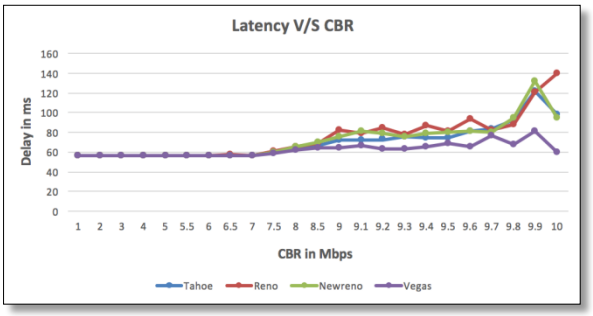

Fig. 5: Latency V/S CBR

Vegas is the better TCP variant which is concluded by performing experiment on the topology shown above. But, it cannot be declared as the best TCP variant as there can be other topologies upon which this experiment has to be performed again.

IV. Experiment 2: Fairness Between TCP Variants

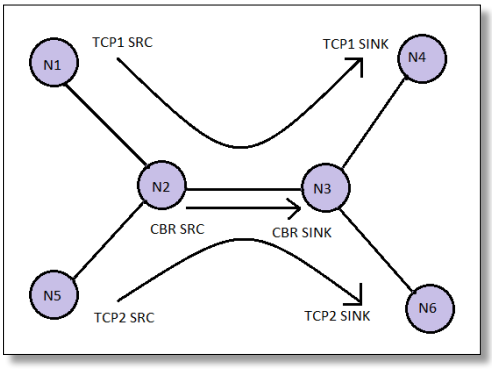

Fig 6: Experiment 2 Network Topology

The above is the topology for experiment 2. This topology has one TCP flow from node N1 (source) to node N4 (sink) and other TCP flow from node N5 (source) to node N6 (sink). The CBR flow is from node N2 (source) to node N3 (sink). In this experiment, the graphs for packet drop rate, throughput and latency are calculated for various pair of TCP variants, which are follows:

1) Reno/Reno

For this case, TCP1 and TCP2 both are assigned as TCP Reno and graphs for both TCP streams are plotted for packet drop rate, throughput and latency in presence of varying CBR.

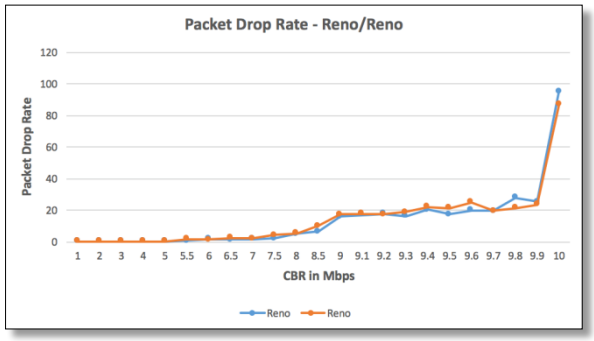

Fig 7: Packet Drop Rate - Reno/Reno

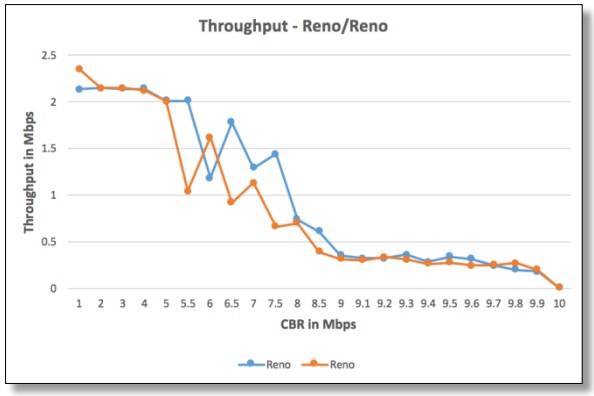

Fig 8: Throughput - Reno/Reno

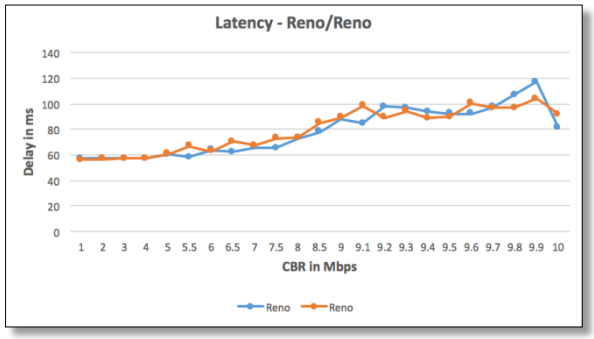

Fig 9: Latency - Reno/Reno

The behavior of Reno/Reno in all the three cases is almost similar. In throughput, Reno/Reno show some spikes but the two lines are similar to each other. This

is because in a constant bandwidth connection, when one variant has more throughput then the other suppress due to bandwidth capacity. And, hence, the two TCP variants alternatively use the given bandwidth. **Thus, the two TCP Reno are behaving fairly with each other.**

2) NewReno/Reno

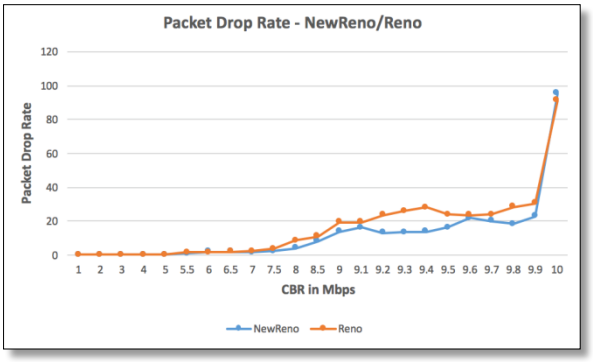

Fig 10: Packet Drop Rate - NewReno/Reno

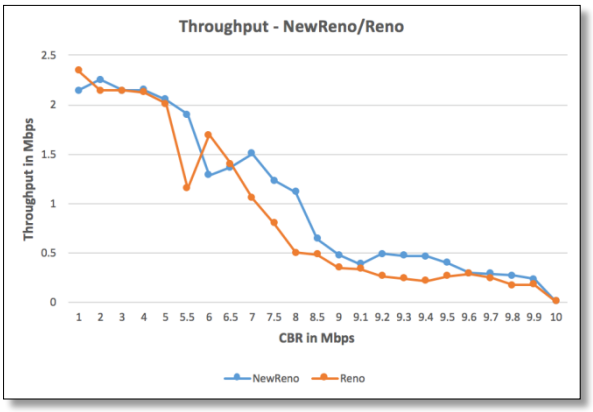

Fig 11: Throughput - NewReno/Reno

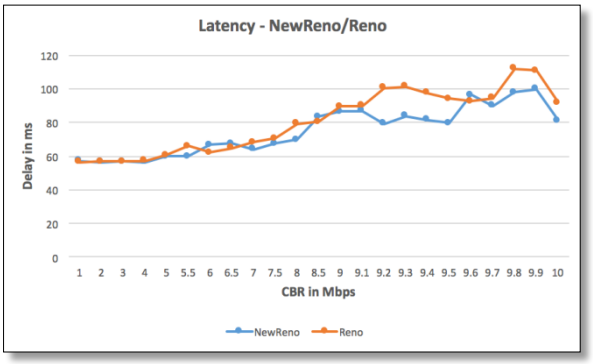

Fig 12: Latency - NewReno/Reno

In this case, TCP1 is assigned NewReno and TCP2 is assigned as Reno and graphs for packet drop rate, throughput and latency are plotted for varying CBR values.

From the throughput graph we can say that, TCP Reno has less throughput than TCP NewReno, when the CBR value is above 6.5. After reaching CBR value of 6.5, TCP Reno losses more packets and the latency of it increases. The reason for such behavior of TCP Reno is that whenever there is a single packet loss it enters Fast Recovery phase and exits. While in NewReno whenever there are multiple packet losses, TCP NewReno exits Fast Recovery phase only after receiving acknowledgement from all the previous packets. **So, in this case, we concluded that TCP NewReno outperforms TCP Reno and is not fair to it.**

3) Vegas/Vegas

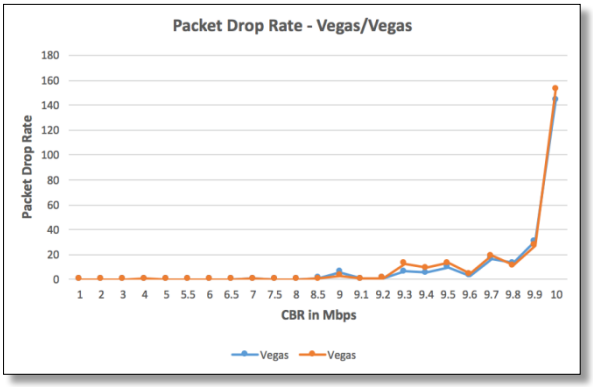

Fig 13: Packet Drop Rate - Vegas/Vegas

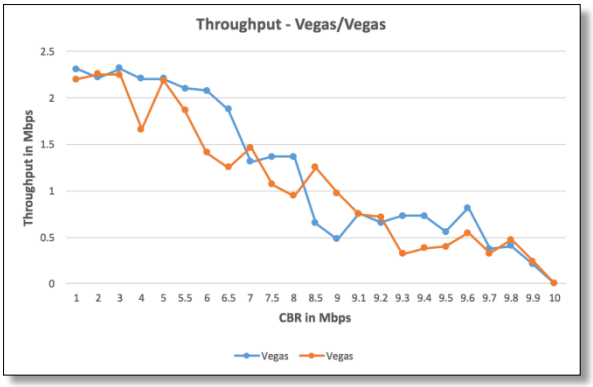

Fig 14: Throughput - Vegas/Vegas

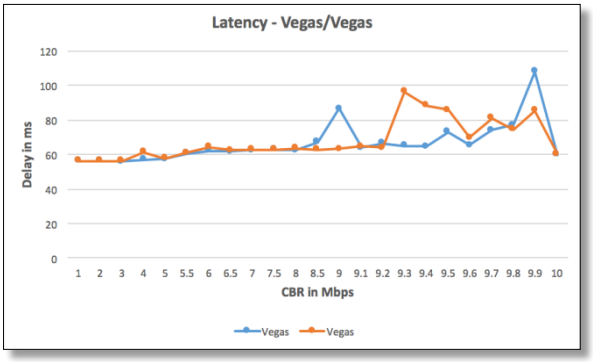

Fig 15: Latency - Vegas/Vegas

In this case, TCP1 and TCP2 both are assigned as TCP Vegas and graphs are plotted for packet drop rate, throughput and latency under varying CBR.

From this graphs, we concluded that both Vegas are fair to each other. When one Vegas detects occurrence of collision, then it immediately backs-off and reduces the number of packets being sent and the other Vegas utilizes the bandwidth. This behavior is examined from the throughput graph by rising and falling trends. But, overall throughput of them is the same.

4) NewReno/Vegas

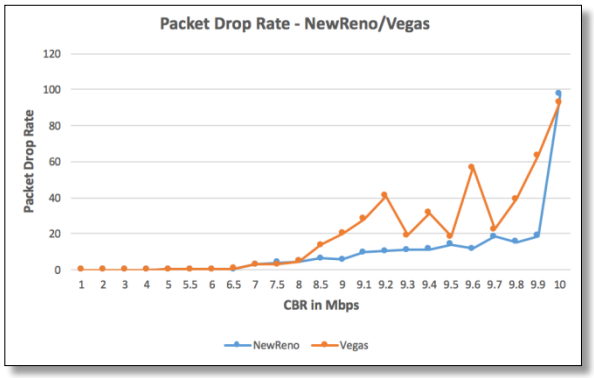

Fig 16: Packet Drop Rate - NewReno/Vegas

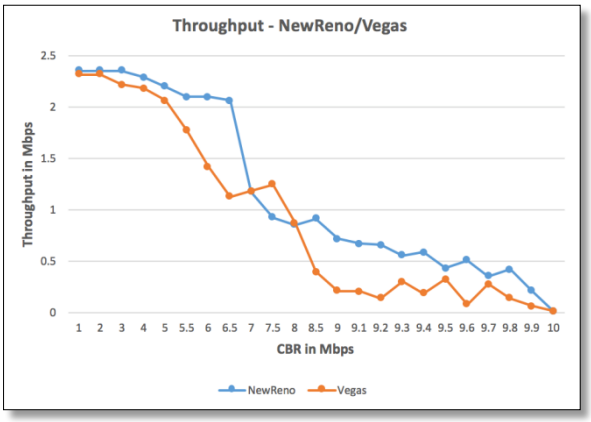

Fig 17: Throughput - NewReno /Vegas

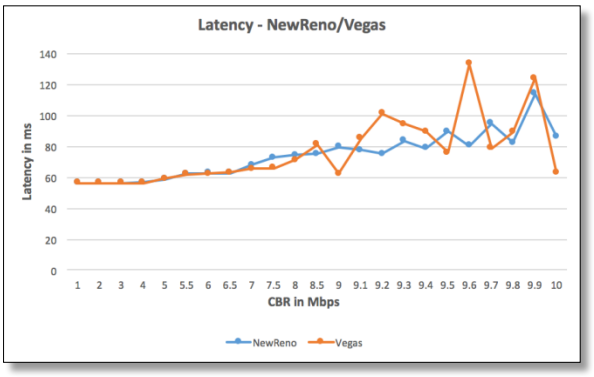

Fig 18: Latency - NewReno /Vegas

The throughput of NewReno becomes more than Vegas after CBR value of 8.5. Also, packet drop rate of NewReno remains less than Vegas. Also, Vegas latency suddenly increases after CBR value of 8.5. **This shows that NewReno is not fair to Vegas.** The reason for this is that when New Reno's traffic increases, TCP Vegas detects congestion and reduces the number of packets sent into the network. But, as CBR increases the number of packet drop increases and the throughput of NewReno decreases while that of Vegas increases due to congestion detection mechanism.

V. Experiment 3: Influence of Queuing

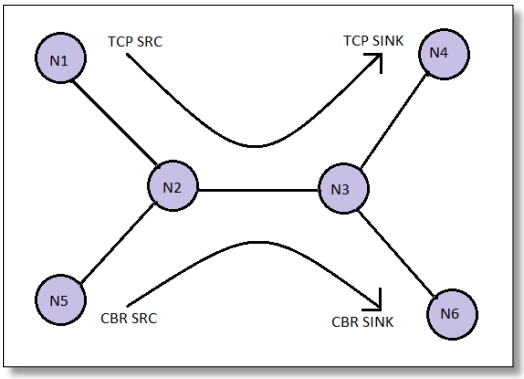

Fig 19: Experiment 3 Network Topology

The network topology for this experiment has TCP flow from node N1 to node N4 and CBR flow from node N5 to node N6. In this experiment, TCP flow is started first and once the TCP flow is stabilized, we have started the CBR flow after 5 seconds. TCP window size is 120 and maxcwnd is 150. The CBR stream rate is kept at 7 Mbps. In this experiment, we analyzed changes in TCP and CBR flow under two queuing algorithms: DropTail and RED. This experiment is performed on TCP Reno and SACK and hence experiments are performed on following pairs: Reno with DropTail, Reno with RED, SACK with DropTail, and SACK with RED.

We have performed the experiment for 30 seconds and have started the TCP flow at the start and then once the TCP flow is stabilized we started the UDP flow on 5 second and stopped at 25 second. RED queuing protocol functionality is different in comparison with DropTail as in RED protocol packets are dropped in a statistical manner of probability while in DropTail packets are dropped independently whenever the queue gets full.

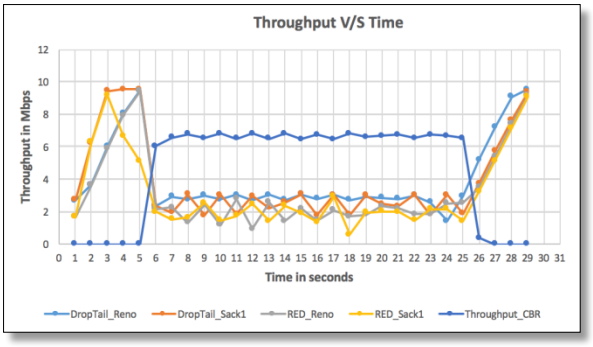

Fig 20: Throughput V/S Time

As seen from figure 20, the throughput of SACK with DropTail and Reno with DropTail is giving better output in comparison with the other two TCP variants with RED queuing discipline and the reason behind this output is the difference in the functionality of both the queuing methods since RED drops packet in a statistical algorithmic manner**. This shows that RED queuing discipline is fair in comparison to DropTail to a particular flow of network.**

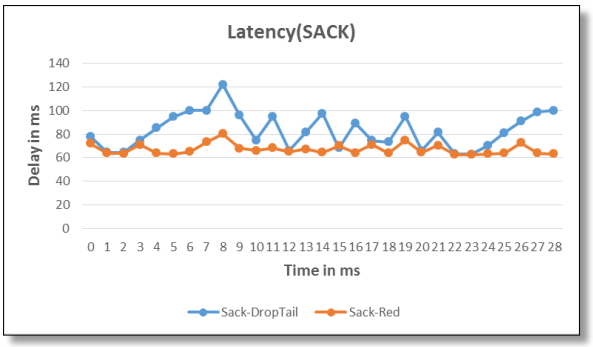

Fig 21: Latency (SACK-DropTail/RED)

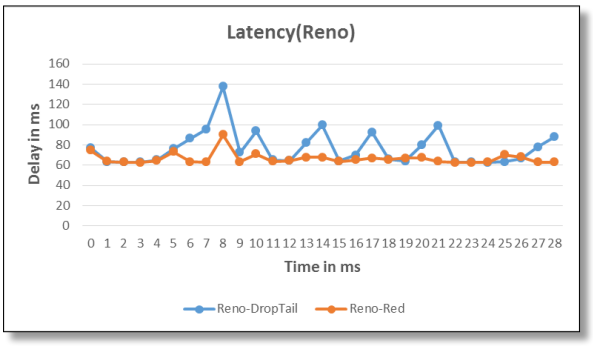

Fig 22: Latency (Reno-DropTail/RED)

As seen in figure 21 and 22 the TCP variants SACK and Reno with RED queuing discipline gives less latency in comparison to TCP (Reno and SACK) with DropTail since RED defines a particular limit upto which bursty traffic can be allowed in the queue while in DropTail there is no such particular limit and drop can take place independently. **Thus, RED queuing method gives less latency as compared to DropTail mechanism.**

The CBR flow does not takes congestion into account as it just bursts the packet irrespective of the congestion present in the network. But, in the **presence of UDP** flow TCP reduces the window size and the thus we can see reduction in the throughput in the presence of CBR between 5-25 sec.

RED queuing method is fair with different variants of TCP. Also, SACK selectively acknowledges packets which were lost thereby giving better throughput and also providing less latency. **Thus, RED is a good idea while dealing with TCP SACK.**

VI. Conclusion

In this paper we have explored the use of NS-2 to study and compare different TCP variants by analyzing the throughput, latency, and packet drop rate within a congested network. We also have checked the fairness between a pair of TCP variants flowing through the network and also the influence of queuing. In real life the network will be more complex and we might get varying results since the network might behave in a different manner than expected. To jump on to a final conclusion we need to do further research with varying network topologies. From the above conducted experiments, we can conclude:

1) TCP Vegas performs better in comparison to TCP Tahoe, Reno, NewReno and Vegas.

2) Same variants of TCP are usually fair to each other while different variants of TCP suppress each other's performance when put through the same kind of network.

3) TCP variants perform better in terms of throughput when DropTail queuing algorithm is used while RED queuing algorithm gives least latency with the same TCP variants.

References

[1] NS by Example by *Jae Chung and Mark Clay pool*, http://nile.wpi.edu/NS/

[2] Simulation-based Comparisons of Tahoe, Reno, and SACK TCP by *Kevin Fall and sally Floyd*.

[3] Analytic models of TCP Performance, *Debessay Fesehaye Kasa*

[4] Wikipedia, TCP congestion-control, https://en.wikipedia.org/wiki/TCP_congestion_control

[5] The-Test, Research Methods Knowledge Base, https://www.socialresearchmethods.net/kb/stat_t.php