Performance Analysis of TCP Variants

Riteshkumar Gupta
Northeastern University, MA 02115
Email Id: gupta.rite@husky.neu.edu

Prajwal Patil
Northeastern University, MA 02115
Email Id: patil.pra@husky.neu.edu

***Abstract* - The widely used TCP protocol was developed to provide reliability in all end-to-end delivery of packets under varying degrees of congestion in the network. This paper presents an in-depth study of performance characteristics of different TCP variants viz. Tahoe, Reno, NewReno and Vegas. NS-2 simulation environment is used to carry out simulations on these TCP variants and analyze the results with respect to throughput, latency and packet drop to test if there is an overall efficient variant. The performance of Reno and SACK for different queuing algorithms viz. Droptail and Random Early Detection (RED) have also been studied.**

I. INTRODUCTION

The primary focus of this experiment is to study and analyse the behaviour of TCP under different conditions.
TCP is the most popular protocol and it provides in sequence deliverance of data and an unfailing data transmission among communicating nodes. One of the strengths of TCP is its high responsiveness towards network congestion. Conducting this experiment will allow us to understand the execution of various TCP variants under different parameters. While performing the experiment, we discovered that Vegas performs better than other variants and same variations in a system are fair to each other.
Few of the TCP variants are discussed below:

1. *TCP Tahoe*

TCP Tahoe was one of the simplest of the 4 variants and was introduced with 3 congestion control algorithms namely, Slow Start, Fast Retransmit and, Congestion Avoidance.

Parameters like congestion window and slow start threshold (ssthresh) were introduced for congestion control. Tahoe doesn't have a fast recovery and treats the 3 duplicate ACKs same as timeout at the congestion avoidance phase. When a timeout or a triple duplicate acknowledgment is received, it performs a fast retransmit, reduces congestion window to 1 and enters slow-start phase.

2. *TCP Reno*

Reno has the same congestion control algorithms as Tahoe with the inclusion of fast recovery. TCP Reno differs from TCP Tahoe at the stage of congestion avoidance. Whenever 3 duplicate ACKs are received, it will halve the congestion window, perform a fast retransmit, and enter fast recovery phase. If a timeout event occurs, it will enter slow-start, as in same as TCP Tahoe.

3. *TCP New Reno*

Reno performs poorly when there are multiple packet drops in the same window, and hence New Reno was introduced. Unlike Reno, New Reno does not exit the fast recovery phase unless and until it receives acknowledgements for all the packets that were present in the window while entering fast recovery phase.

4. *TCP Vegas*

Vegas is a modification of Reno. It builds on the fact that a proactive measure to encounter congestion is much more efficient than reactive ones. Vegas implements congestion avoidance rather than first detecting the congestion and then taking steps to decrease congestion on the channel. It calculates a base RTT (Round Trip Time) and compares it with RTT of packet with recently received ACK. It increases its sending window if the compared RTT is much smaller than the base RTT and if RTT is greater than the base RTT, it decreases its sending window.

5. *TCP SACK*

SACK is an extension of Reno. In SACK, selective ACKs are carried out instead of cumulative ACKs. When a TCP enters the fast recovery phase, SACK implements a parameter called "pipe" which represents the estimate of the number of unacknowledged packets in the network. The congestion window is reduced to half of the current window. The pipe is decremented by 1 for every acknowledgement received and is incremented by 1 for every retransmitted packet. If there are no unacknowledged packets in the network, a new packet is transmitted enabling multiple packet transmission in one RTT.

II. METHODOLOGY

NS-2 simulator has been used to conduct this experiment. It is a discreet event simulator targeted at networking research and provides valuable support for simulation of routing, multicast and IP protocols, such as TCP, UDP, and RTP over wired and wireless networks. The NS-2 simulation to achieve the performance between different TCP variants are performed over the following network topology as shown in Fig. 1.

Topology in Fig. 1 have nodes connected using full duplex links with a bandwidth of 10 Mbps and a delay of 10ms.

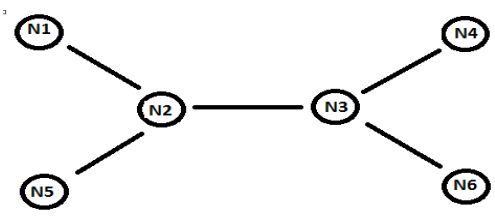

Fig. 1. Network Topology

This network topology is run in NS-2 under different conditions to generate trace files, which are parsed and studied to analyze the behaviour of the above mentioned TCP variants with respect to throughput, latency, drop rate, fairness and different queueing algorithms.

A. *Experiment 1*

In this experiment we analyze the performance of TCP variants under different load conditions. Only one TCP flow is set from N1 to N4, linearly varying the rate of CBR flow from 1 Mbps to 10 Mbps. This is performed for different variants: Tahoe, Reno, New Reno and Vegas.

1. *Throughput Analysis*:

a) T- test: It is the relative comparison of mean of two different sets of values and notifies the stability of a set of observations with respect to each other.

b) Mean: Mean is the average throughput of a specific variant. Higher the calculated mean, higher is its average throughput for entire range of CBR bandwidth.

2. Latency

Average latencies are calculated for each variant and compared. This helps us to understand the variant with least overall latency as well as for a specific bandwidth.

3. Packet Drop Rate

Number of dropped packets are calculated over the total number of packets. It is considered to find variant with the minimum drop rate.

B. *Experiment 2*

This experiment is performed to see if one TCP variant is fair to another TCP variant or not. One TCP flow is set from N1 to N4 and the other flow from N5 to N6. The rate of CBR flow is varied linearly from 1Mbps to 10 Mbps. The fairness of variants to one another is analyzed by plotting graphs for throughput, latency and drop rate for each flow with respect to CBR flow rate.

C. *Experiment 3*

In this experiment we study the effect of queuing algorithm like Drop Tail and Random Early Detection (RED) on the throughput and latency of TCP Reno and TCP SACK. The same topology as in experiment 1 is used to carry out the experiment.

EXPERIMENT 1: TCP PERFORMANCE UNDER CONGESTION

The network topology and the flow setup is shown in Fig. 2. The CBR is made to flow from N2 to N3 and is varied and the TCP flow is streamed from N1 to N4. The TCP Variants considered for the experiment are Tahoe, Reno, New Reno and Vegas.

The NS2 simulation is carried out for each TCP variant against the increasing CBR value. The CBR flow is raised from 1Mbps to 10Mbps

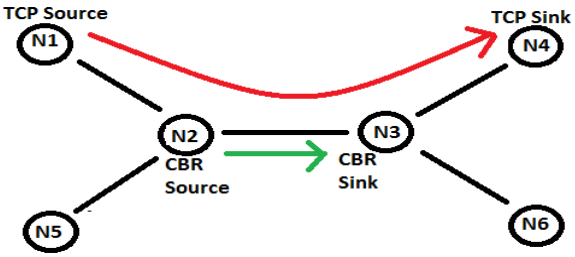

Fig.2: Experiment 1 Network Topology

The trace file is generated for every change in CBR value, and we computed the throughput, amount of latency, packet drop rate of the 4 TCP variant during the simulation. As CBR flow is increased, the performance of TCP decreases because TCP being a reliable protocol, will only send the next window of packets only when the acknowledgment for previous packets are received. Whereas CBR, that runs over UDP connection does not rely on any acknowledgments and the packets are continuously injected into the network creating network congestion. Hence, the throughput of any network decreases with the increase in CBR flow which is verified in experiment 1.

A) Throughput

The trace file is generated by NS-2 simulation and each line describes the network event. We parsed all the TCP events based on the flow ID and calculated the number of received bits during a particular time interval as the throughput for that time of simulation. We get the simulation result as shown in Fig. 3.

For CBR $\leq$ 7.5Mbps, the throughput for all TCP variants is almost similar. However, when the CBR flow increases and the network start to congest, Vegas performs better and have the highest average throughput as compared to other three, while TCP Reno has the lowest throughput. This is also confirmed with the T-Test mean value of Vegas that equals 5.36 which is greater compared to other variants.

Hence, it transmits less data when the network is congested by conducting congestion detection. Thus, the good throughput of Vegas, as the network becomes congested, is the result of its pre-detection of the network congestion.

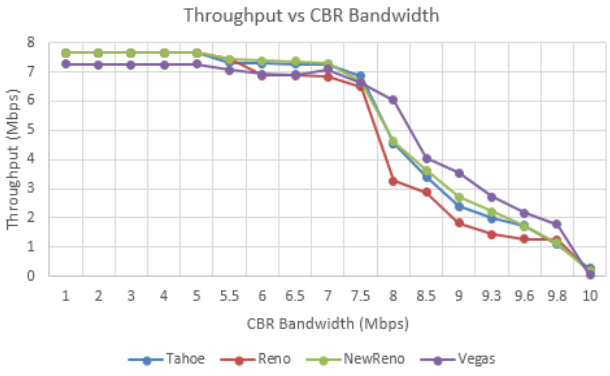

Fig.3: Throughput vs. CBR Bandwidth

B) Latency

The average latency for each TCP variant was computed based on the actual average of RTT of packets. After the NS2 simulation and generation of the trace file, we parsed each event of packet traversal when it leaves the source and reaches the destination and returns back. We mapped each TCP packet with their sequence number and with the Send and ACK time. Average latencies of the TCP variants are depicted in Fig. 4

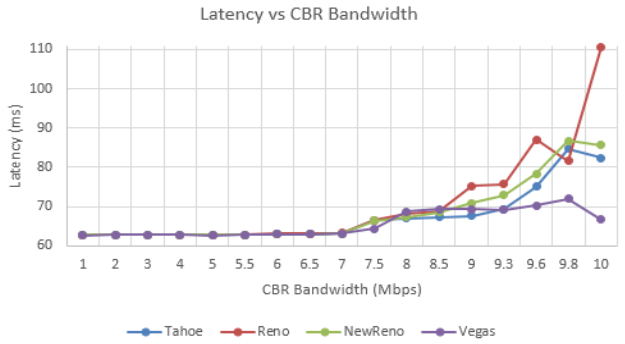

Fig.4: Latency vs. CBR Bandwidth

Tahoe, Reno, NewReno and Vegas all exhibit the same latency till CBR ≤ 7Mbps, but thereafter the average latency for each TCP variant varies. As seen from the Fig. 4, Vegas detects congestion at the initial stage, and it queues the packets when the RTT received is greater than the base RTT. Hence, Vegas has the lowest average latency. The other three variants have comparatively higher latency than TCP Vegas. The other variants continue to increase their window size until a packet loss is detected after which they enter into congestion avoidance mechanism. They react to the congestion only after the congestion has started and packet drops tend to increase which thereby causes high latency in packet transmission.

C) Packet Drop Rate

We filtered all the dropped packets during transmission by checking the dropped events from the trace file. The unique sequence ID field can count the total number of TCP packets for all TCP flow events. The packets drop rate under varying CBR traffic is shown in Fig. 5.

Initially, when the CBR ≤ 7.5Mbps, the packet drop rate is very less, but as CBR flow increases, Vegas drops fewer packets while the other variants have more packet drop rate.

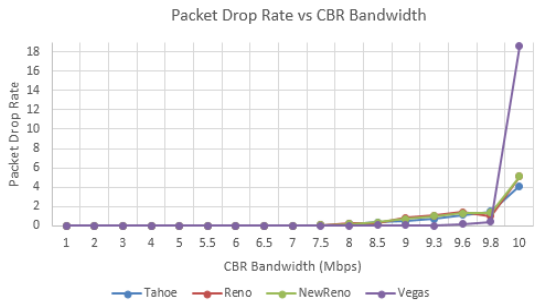

Fig.5: Packet Drop Rate V/S Bandwidth

When CBR reaches to bandwidth limitation i.e. CBR =10Mbps, Vegas has a sudden increase in drop rate compared to others because when the packet drop rate remained constant during the initial stages, the window increased, and doubled the input from the previous transfer. However, once the queue got filled, the arrived packets that are large in number are dropped, and the window reduces drastically to half of its current value.

Thus, based on our analysis, Vegas performed better than other three variants. Nevertheless, it is important to set up a bigger and more complex topology and perform the experiment again and to compare the results to determine the best variant.

EXPERIMENT 2: FAIRNESS BETWEEN TCP VARIANTS

The network topology and flow setup of experiment 2 is as Fig. 6.

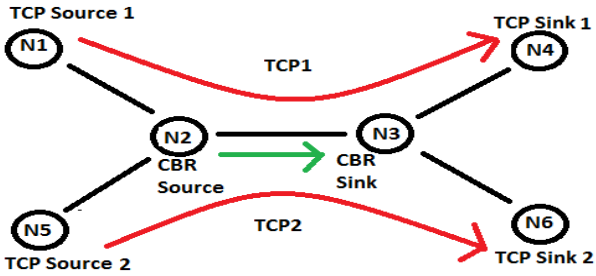

Fig.6: Experiment 2 Network Topology

For this experiment we are running one TCP variant from N1 to N4, second TCP variant from N5 to N6 and CBR flow from N2 to N3. In this experiment, we will be determining the fairness between different TCP variants. For determining fairness, we are computing and plotting the graphs for Throughput, Packet drop rate and Latency for the following pairs of TCP variant.

A. Reno/Reno

As per the network topology in Fig.6, both TCP1 and TCP2 links are assigned as Reno. Fig.7, Fig.8 and Fig.9 show a comparison of throughput, latency, and packet drop rate between the two Reno agents in the same network.

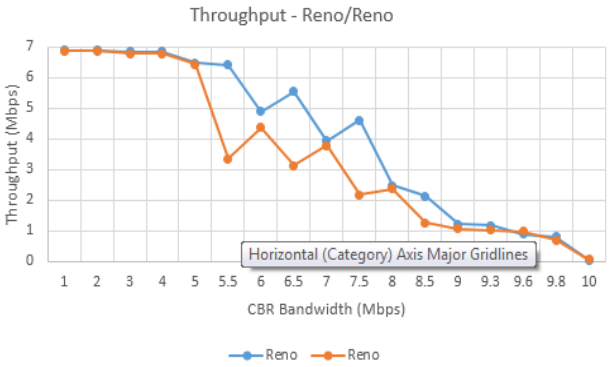

Fig.7: Throughput V/S Bandwidth

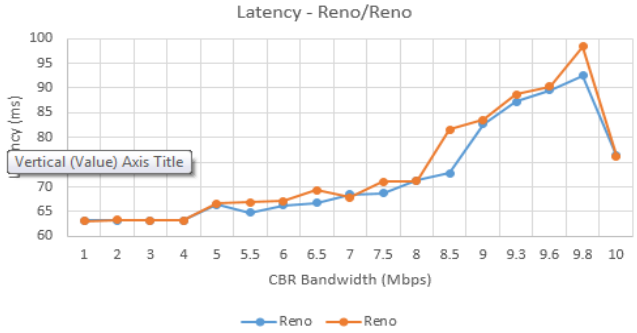

Fig.8: Latency V/S Bandwidth

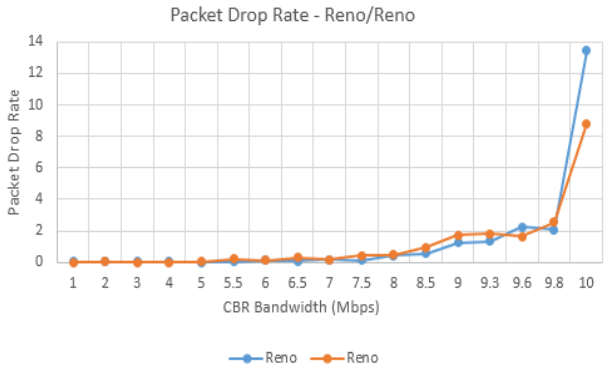

Fig.9: Packet Drop Rate V/S Bandwidth

Despite the fact that there are flutters in the throughput, the two Reno lines change with similar tracks and are close most of the time. Same goes with latency and packet drop rate. This wavering of the throughput is justified, since in a connection with a constant bandwidth when one TCP variant at a point has more throughput, the other is suppressed due to the bandwidth capacity, and vice versa. Hence, the two TCP agents are alternatively utilizing the bandwidth. Thus, the two Reno TCP variants are fair to be on the same system.

B. NewReno/Reno

Now, assigning New Reno to TCP1 and Reno to TCP2 link. Fig.10, Fig.11 and Fig.12 show a comparison of throughput, latency, and packets drop rate between the New Reno and Reno agents in the same network topology.

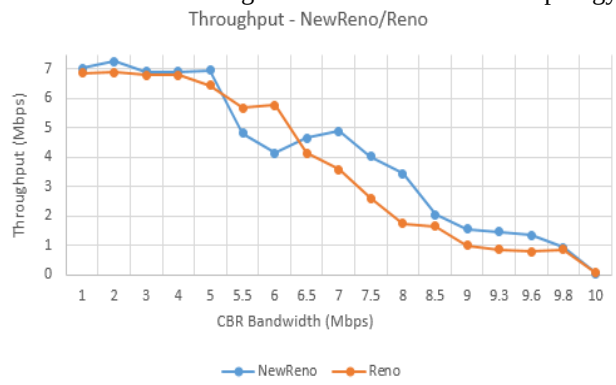

Fig.10: Throughput V/S Bandwidth

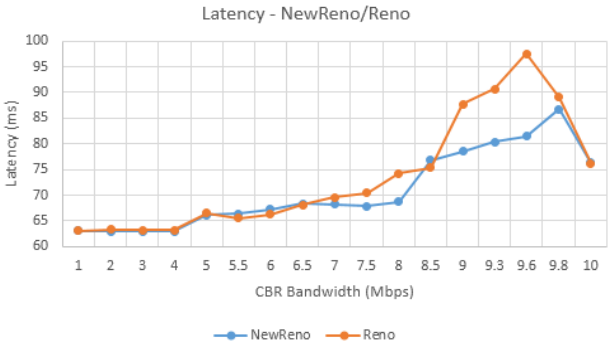

Fig.11: Latency V/S Bandwidth

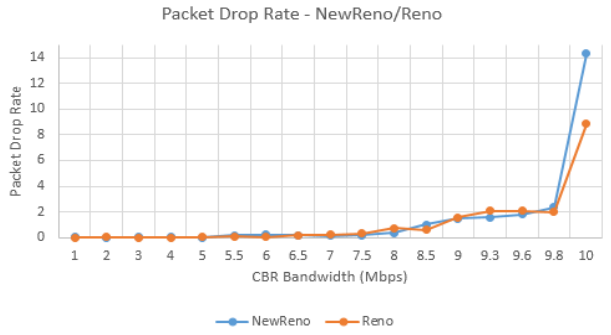

Fig.12: Packet Drop Rate V/S Bandwidth

New Reno has higher throughput than Reno, Reno drops more packets and has a higher increment in latency when CBR comes closer to bandwidth limitation as Reno handles a single packet loss scenario and exits and enters the Fast Recovery stage per loss. Whereas, New Reno handles multiple packet drops but it does not exit the fast recovery stage until all previous packets are acknowledged. Therefore, New Reno is not fair to Reno. Moreover, it does not overcome Reno's performance completely.

C. Vegas/Vegas

We now assign TCP Vegas to both TCP1 and TCP2 link. Fig.13, Fig.14 and Fig.15 show a comparison of throughput, latency, and packets drop rate between the two Vegas agents in the same network topology.

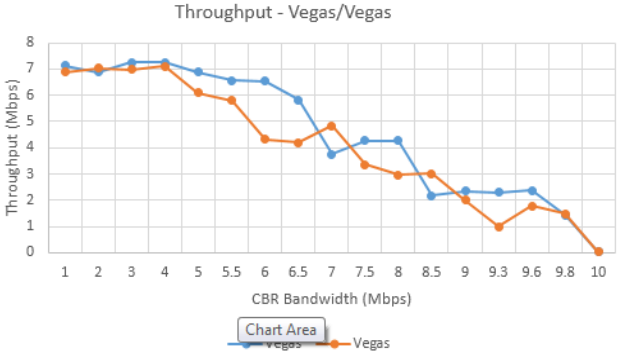

Fig.13: Throughput V/S Bandwidth

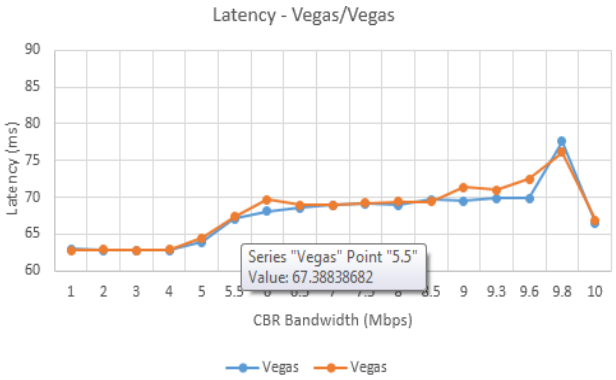

Fig.14: Latency V/S Bandwidth

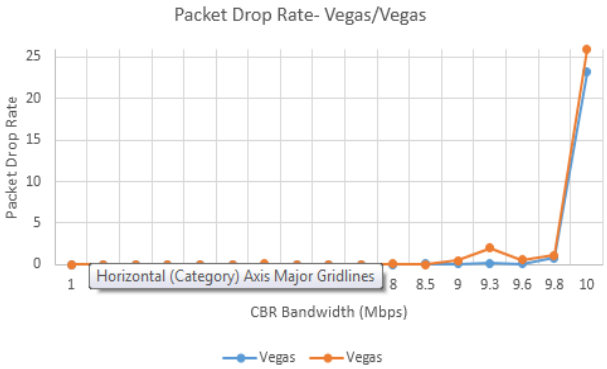

Fig.15: Packet Drop Rate V/S Bandwidth

As similar to the case of Reno/Reno, Vegas/Vegas show a similar behaviour. On comparing the throughput, latency and drop rate graphs, when one Vegas at a point has more value, the other is suppressed due to the bandwidth capacity, and vice versa. Thus, the two Vegas TCP variants are fair to be on the same system.

D. New Reno/Vegas

Now, we assign New Reno to TCP1 and Vegas to TCP2 link. Fig.16, Fig.17 and Fig.18 show the comparison of throughput, latency, and packets drop rate between the New Reno and Vegas agents in the same network.

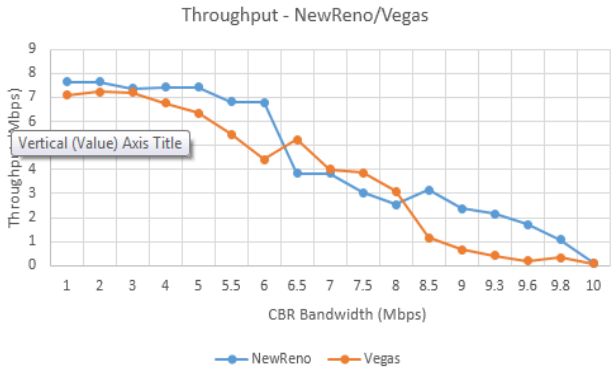

Fig.16: Throughput V/S Bandwidth

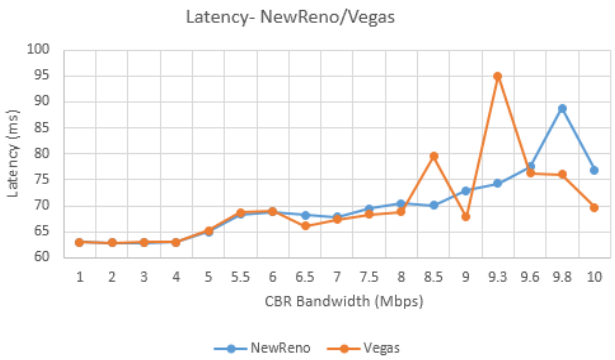

Fig.17: Latency V/S Bandwidth

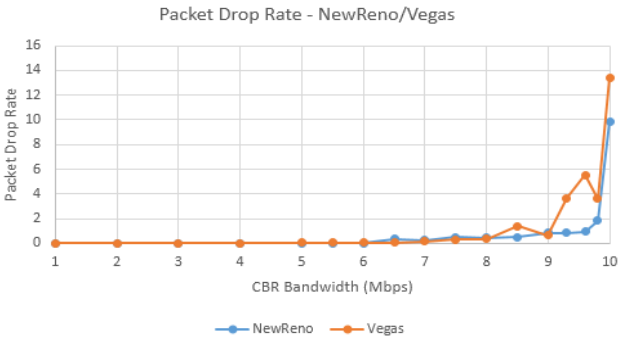

Fig.18: Packet Drop Rate V/S Bandwidth

When CBR flow increases, New Reno has higher throughput as compared to Vegas and this division becomes larger gradually. Also, it runs more steadily and drops less number of packets than Vegas. For latency, they remain close for a particular period, but when CBR reaches 8Mbps, the latency of Vegas increases gradually till the bandwidth limitation. So, it is quite clear that New Reno is unfair to Vegas. This is because Vegas recognizes congestion at an early stage and it decreases its rate of sending packets which allow it to provide more bandwidth to New Reno and hence, New Reno becomes dominant.

Thus, from the above experiment, we infer that when same TCP variants are assigned to a network, then they are fair to one another, while a combination of different TCP variants can't be entirely reasonable to each other. Exemplifying this with the early congestion detection of Vegas makes it dominated by New Reno, and once New Reno becomes dominant, Vegas always considers the network to be congested and diminishes its sending rate, and therefore never gets predominant back again.

EXPERIMENT 3: INFLUENCE OF QUEUING

The network topology and flow setup of experiment 3 is as Fig. 19.

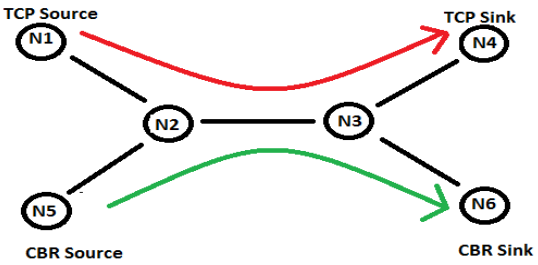

Fig.19: Experiment 3 Network Topology

At first, for this experiment, we start the TCP flow and allowed it to reach to its stable state. As soon as it becomes stable we start CBR flow i.e. at 3 seconds. We stopped both, TCP and CBR flow at 10 seconds. The bandwidth of every link is kept to 10Mbps with a delay of 10ms, TCP window size is set to 80, TCP congestion window cwnd is kept as 110, and the CBR stream rate is set to 7Mbps. We recreate four sets of TCP variations and queue types: Reno with RED, Reno with DropTail, SACK with RED, and SACK with DropTail. The assessment is done on the throughput and average latency of TCP stream. Based on the network topology from Fig. 19, we assigned one of the TCP variants (Reno or SACK) between N1 and N4 and CBR flow between N5 and N6.

It can be observed from Fig. 20 that throughput of Reno and SACK with RED is bigger when compared with the other two. This is because, in droptail queuing, network component buffers as many packets as it can, and drops the ones it cannot buffer. The network is congested if buffers are constantly full and hence network become under-utilized. RED monitors the average queue size and drops packets based on statistical probabilities depending on the bandwidth availability of the flow. Hence, RED ensures fair utilization of the available bandwidth and provides better throughput.

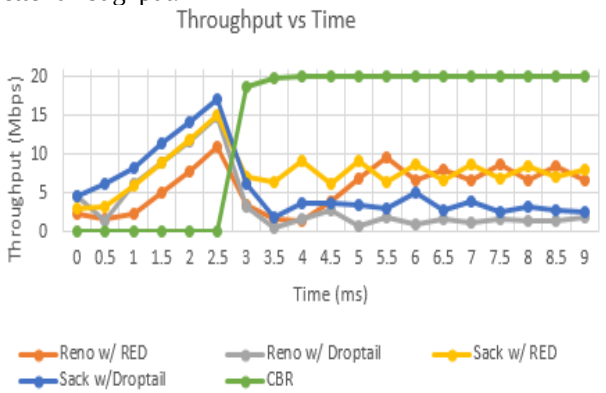

Fig20: Throughput V/S Time

B. Average Latency

From Fig. 21 we can infer that Reno and SACK have the lowest latency with RED queuing discipline. This can also be verified by the T- Test that is performed on them. T-Test values for Reno and Sack both with Droptail have higher value, that means higher latency and poor performance. Thus, it is depicted that average end-to-end latency is less for RED as compared to Droptail. This is because Droptail continuously fills up the queue in an unbiased manner and when CBR is introduced network gets congested, RTT to traverse a packet increases thereby increasing latency which is properly handled by RED

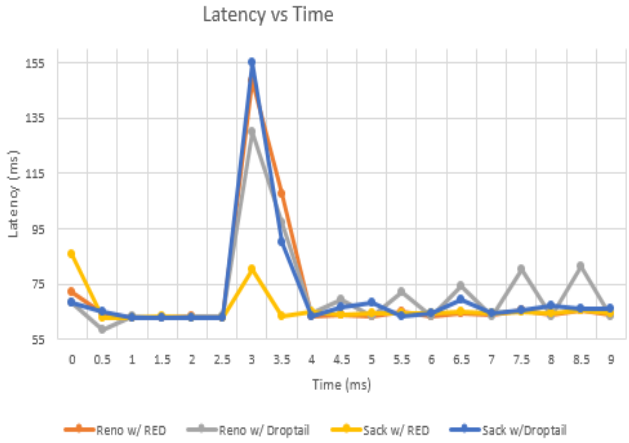

Fig21: Latency V/S Time

Referring the above graph and the T-test result, we can conclude that RED works well in accordance with TCP SACK and Reno, providing better throughput, good utilization of bandwidth and less latency.

Conclusion

This paper has researched the use of NS-2 Simulator to study and compare different TCP variants to scale the performance, fairness, and influence of queueing discipline by analyzing the throughput, latency, and packet drop rate within a congested network. These experiments were performed with a simple network topology and the results are based on it. In order to jump on a final conclusion, further research is needed with various different topology, varying with respect to complexity and number of nodes. From the above conducted experiments and analysis of the performance, we can conclude that:

1) TCP Vegas is better than any other TCP variant for sending data due to its better throughput, average latency, and fewer packet drops.

2) Networks with same TCP variants are found to be fair to each other while different TCP variants in the same network suppress each other's performance.

3) TCP SACK with RED was found to have better throughput, higher stability and least latency.

References

1] TCP Tahoe, Reno, NewReno, and SACK–a Brief Comparison http://www.roman10.net/2011/11/10/tcp-tahoe-reno-newreno-and-sacka-brief-comparison/

2] Fair comparisons of different TCP variants for future deployment of Networks http://www.academia.edu/183250/Fair_comparisons_of_different_TCP_variants_for_future_deployment_of_Networks

3] Textbook: Computer Networking, A Top-Down Approach by James F Kurose and Keith W. Ross.