Performance Analysis of TCP Variants

Archit Mathur
Northeastern University, MA, 02115

Abstract—TCP is considered as a reliable source for end-to-end delivery of packets under congestion. In this paper, we have used NS-2 to simulate different TCP variants on a simple network topology and analyzes their performance based on throughput, packet drop rate, and latency. We compare Tahoe, Reno, NewReno and Vegas to analyze their performance under congestion and fairness between them. We have also studied the influence of queuing disciplines DropTail and Random Early Drop (REM) on TCP SACK and Reno.

Index Terms – DropTail, SACK, TCP NewReno, TCP Rene, TCP Vegas, TCP Tahoe

I. INTRODUCTION

The main purpose of this experiment is to study and analyze the behavior of TCP under various conditions. The traditional model of TCP could not be used in the developing and expanding Internet architecture due to which there were many variants which arose with it such as Tahoe, Reno, NewReno, Vegas, and SACK each of them finds its applications under given circumstances and have their advantages and disadvantages. Conducting this experiment will allow us to understand the execution of various TCP variants under different parameters. While conducting the experiment, we discovered that Vegas performs better than other variants and same variations in a system are fair to one another.

Some of these TCP Variants are discussed below:

1. TCP Tahoe

Tahoe is the simplest TCP variant. It does not have fast recovery state and during congestion avoidance phase, Tahoe treats a triple duplicate ACKs same as a timeout.

2. TCP Reno

Reno is different from TCP Tahoe at congestion avoidance stage. When a triple duplicate ACKs is received, the congestion window gets halved and performs a fast retransmit to enters fast recovery mode

3. TCP NewReno

NewReno was designed to improve TCP Reno's performance. In TCP NewReno, data ACK is not enough to take out TCP from fast recovery to congestion avoidance. Instead, it requires all the outstanding packets to get acknowledged.

4. TCP Vegas

Vegas calculates the RTT and compares it with RTT of received packets with the recently received acknowledgments. Depending on the current RTT value, Vegas detects a congestion and resizes its queue size itself

5. TCP SACK

SACK allows the receiver to acknowledge non-consecutive data, which only allow non-transmitted or the missing data to be retransmitted once again. TCP experience downfall in performance when multiple packets are lost from one window of data.

II. METHODOLOGY

To conduct this experiment we have used the NS-2 simulator. An NS-2 simulator is a tool, used while developing or analyzing the behavior of network protocols by extracting the packet information from the generated trace file. We have specifically used NS-2 because it has built in extensive support, reliable, open source and lastly universally accepted. The NS-2 simulation to achieve the performance between different TCP variants are performed over the following network topology.

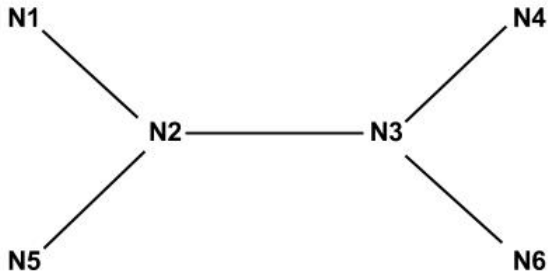


Fig1: Network Topology

N1, N2, N3, N4, N5 and N6 represent nodes over which we design our experiments. Each of nodes is connected by a full-duplex link which has a bandwidth of 10 Mbps with a delay of 10ms.

In this paper, we have conducted three different experiments to understand clearly the performance of TCP variants

including Tahoe, Reno, NewReno, Vegas and SACK. This network topology was executed under different condition in

NS-2 to generate trace files. These trace files were parsed to analyze the behavior of TCP variants under different conditions.

Experiment 1

This experiment inspects the Throughput, Latency and Packet drop rate of Tahoe, Reno, NewReno, and Vegas by varying the UDP based CBR (Constant Bit Rate) and analyzing their performance individually. A TCP flow is set from N1 to N4 and a linearly varying CBR flow from 1Mbps to 10Mbps was introduced between N2 to N3. TCP window size is 20 and maximum congestion windows size is 0. The CBR rate is varied, and the experiment was conducted under following conditions:

- Start and end the TCP and CBR flow simultaneously
- Start CBR flow after TCP stabilizes
- Decrease the CBR rate from 10Mbps to 1Mbps.

A) Throughput is analyzed by computing the following parameters:

A T-Test computes a statistical examination between two population means. A T-Test examines whether samples are different or if the variances of two normal distributions are unknown.

The ”mean” is the ”average” of all the values obtained by the total number of observations, where we add up all the numbers and then divide by the number of numbers.

The variance and the closely-related standard deviation are measures of how spread out a distribution is. The variance can be computed as the average squared deviation of each number from its mean.

B) Packet Drop Rate: Packet drop rate is the difference between a number of packets sent and the number of packets acknowledged over a period. A number of dropped are calculated to find the variant with least packet drop rate.

C) Latency: Latency is a time interval between the stimulation and response. It was calculated to for each variant to understand the variant with least and highest overall latency.

Experiment 2

This experiment compares the fairness between TCP variants in the same network with congestion by inspecting the through-put, Latency and Packet drop rate on the following TCP variants pairs: Reno/Reno, NewReno/Reno, Vegas/Vegas and NewReno/Vegas.

One TCP flow is set from N1 to N4 and another from N5 to N6. The rate of CBR flow is varied from 1Mbps to 10Mbps. The fairness between different variants is analyzed by plotting the graphs for each flow on CBR flow rate.

Experiment 3

In this experiment, we will verify the influence of the queuing disciplines, DropTail and Random Early Drop (RED) on Throughput and Latency of TCP Reno and SACK. The topology used is same as Figure 1.

III. EXPERIMENT 1: TCP PERFORMANCE UNDER CONNGESTION

The network topology and flow setup are shown in Figure 2. The CBR flows from N2 to N3 is the only varying factor. The TCP flow from N1 to N4 will be one of Tahoe, Reno, NewReno and Vegas.

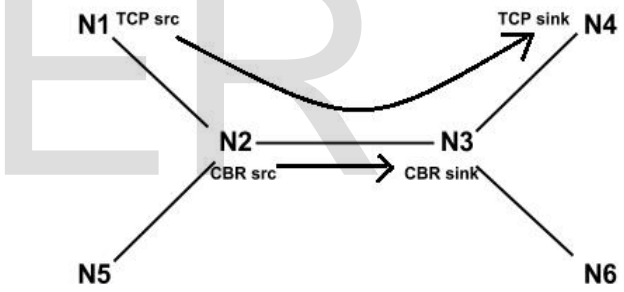


Fig2: Experiment 1 Network Topology

The simulation will be carried out for each TCP variant against the increasing CBR. The CBR flow increases from 1Mbps to 10Mbps gradually by 0.1Mbps. We get a trace file for every change in CBR value, and we compute the amount of latency, packet drop, and throughput. With the increase in CBR flow, the performance of TCP decreases because TCP is a reliable protocol and it will only send the next window of packets only if the acknowledgment for previous packets are received. Whereas CBR does not rely on any acknowledgments and it keeps on sending the packets to the network and creates network congestion. Hence, the throughput of any network should decrease with an increase in CBR flow. This can be verified in experiment 1.

A) Throughput

The trace files generated by NS-2 describes network event in each line, we parse all the TCP events based on the flow ID

and calculate the packet size field as the throughput for that time of simulation. We get the simulation result as shown in Figure 3.

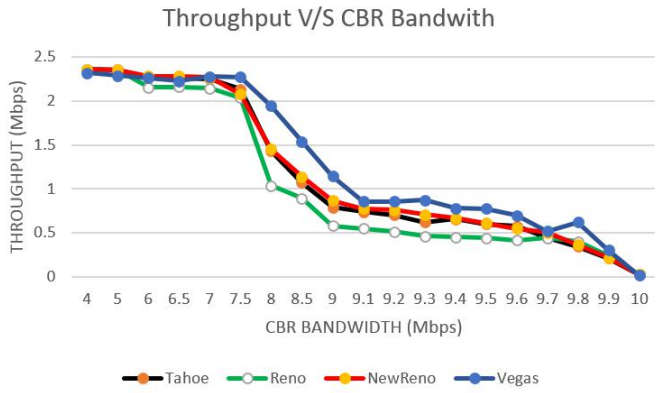


Fig3: Throughput V/S Bandwidth

For CBR ≤ 7.5Mbps, the throughput for all TCP variants is almost similar. However, once the CBR flow increases and the network start becoming congested, Vegas performs better average throughput as compared to others three, while TCP Reno has the least throughput. Vegas detects congestion at an early stage based on increasing RTT values of the packets. Hence, it transmits fewer data when the network is not congested since it conducts more congestion detection. However, as the network becomes congested, Vegas pre-detection helps a lot towards its throughput.

B) Packet Drop Rate

We can filter out all the TCP packets drop events from the trace file. The unique 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 Figure 4.

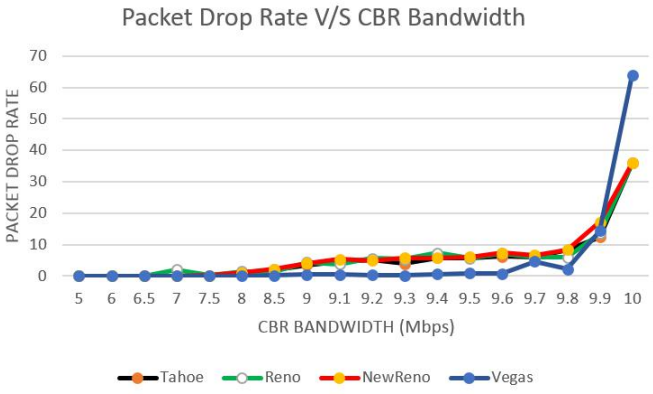


Fig4: Packet Drop Rate V/S Bandwidth

Initially, when the CBR ≤ 8Mbps, the packet drop is very less, but as CBR flow gradually increases, Vegas drops fewer packets while the other variants have higher drop rate. When CBR reaches to bandwidth limitation, Vegas has a sudden increase in drop rate compared with the others because as the packet drop rate remains same during the initial stages, the window increases, and doubles the input from the previous transfer. However, once the extended queue is filled, the arrived packets are dropped, and the window reduces drastically to half its current value. The concept holds good for the other three as well, but they almost show the same characteristics differing only towards the end.

C) Latency

The average latency for each variant was roughly calculated based on the actual average RTT of packets. We maintained a table to map each TCP packet with their sequence number and the send and ACK received time.

The average latencies of the TCP variants are depicted in Figure 5

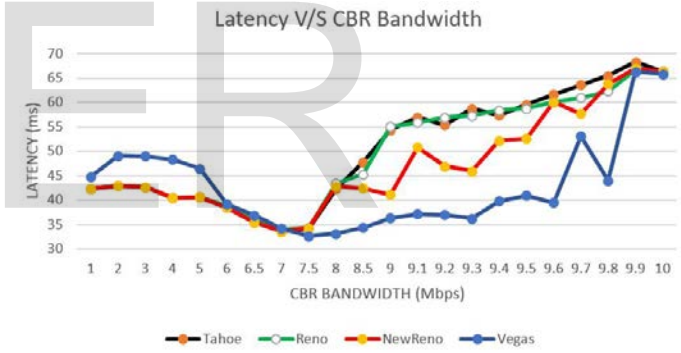


Fig5: Latency V/S Bandwidth

Till CBR ≤ 7Mbps; Tahoe, Reno, and New Reno have same latency, but after that Tahoe has a blast when CBR is reached towards the bandwidth limitation because Tahoe uses Go Back N and as the packet drop occurs retransmission rate will increase which results in high queuing delays. Whereas Vegas detects congestion at the initial stage, and it will queue the packets when the received RTT is greater than the base RTT. Hence, Vegas has the least latency. However, just before CBR becomes dominant, all the variants have similar performance.

Based on our analysis Vegas performed better than other variants. Hence, it is the best among the other three variants regarding performance within this experiment. However, it cannot be declared as the best among all because experiment 1 only considers a simple network topology. It might be

important to set up a greater and more entangled topology and perform the experiment again to compare the results.

IV. EXPERIMENT 2: FAIRNESS BETWEEN TCP VARIANTS

The network topology and flow setup of experiment 2 are as Figure 6

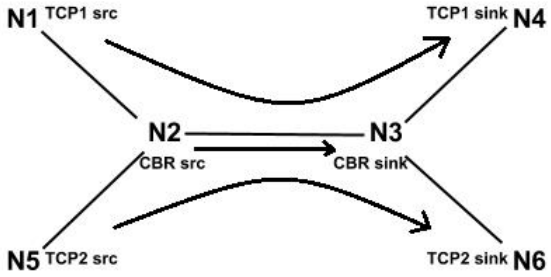


Fig6: Experiment 2 Network Topology

In this experiment, we have used similar topology as experiment number 1. Also, we run one more variant from N5 to N6 and compute the graphs for Throughput, Packet drop rate and Latency for the following pairs of TCP variant.

A. Reno/Reno

According to the network topology in Figure 6, both TCP1 and TCP2 link are assigned as Reno. Figure 7, Figure 8 and Figure 9 show a comparison of throughput; packets drop rate and latency between the two Reno agents in the same network.

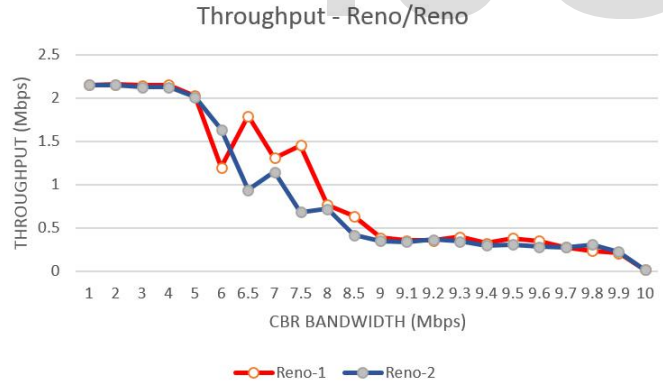
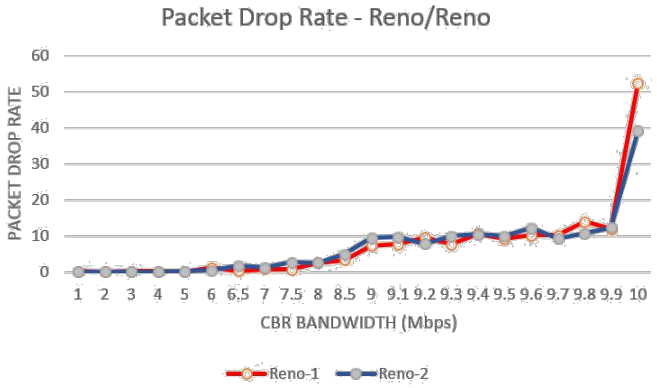


Fig7: Throughput V/S Bandwidth



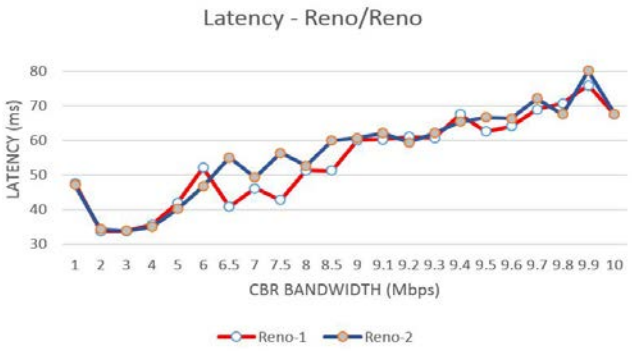


Fig8: Packet Drop Rate V/S Bandwidth
Fig9: Latency V/S Bandwidth

Despite the fact that there is wavering in the throughput, the two Reno lines change with similar tracks and keep close, with increase in CBR. The wavering is justified, since in a connection with constant bandwidth, at a point when one TCP variant has more throughput, the other one suppress due to the bandwidth capacity, and vice versa. Hence, the two TCP agents can alternatively utilize the bandwidth. Thus it is fair for the two Reno TCP variants to be on the same system.

B. NewReno/Reno

Now, we assign NewReno to TCP1 and Reno to TCP2 link.

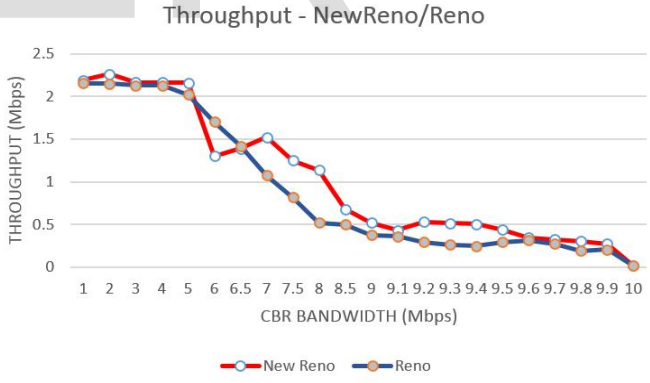


Figure 10, Figure 11 and Figure 12 show a comparison of throughput; packets drop rate and latency between the NewReno and Reno agents in the same network topology.

Fig10: Throughput V/S Bandwidth
Fig11: Packet Drop Rate V/S Bandwidth

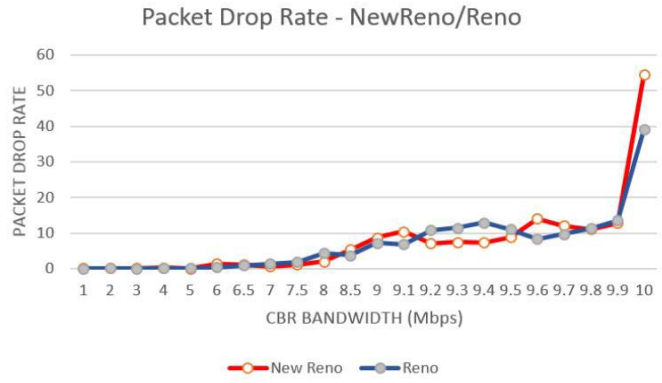


Fig12: Latency V/S Bandwidth

TCP Reno had lower throughput than NewReno when CBR ≥ 6.5Mbps, Reno drops more packets and has a higher increment in latency when CBR comes to close bandwidth limitation because it handles a single packet loss scenario and exits and enters the Fast Recovery stage per loss. Whereas NewReno handle multiple packet drops but it does not come out of fast recovery unless all previous packets are acknowledged. Therefore, NewReno is not fair to TCP Reno; also, it does not overcome Reno's performance completely.

C. Vegas/Vegas

On Figure 5, we now assign TCP Vegas to both TCP1 and TCP2 link. Figure 13, Figure 14 and Figure 15 show a comparison of throughput; packets drop rate and latency between the two Vegas agents on same network topology.

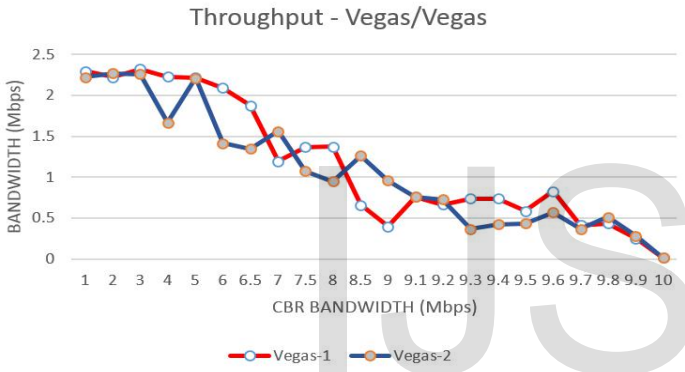
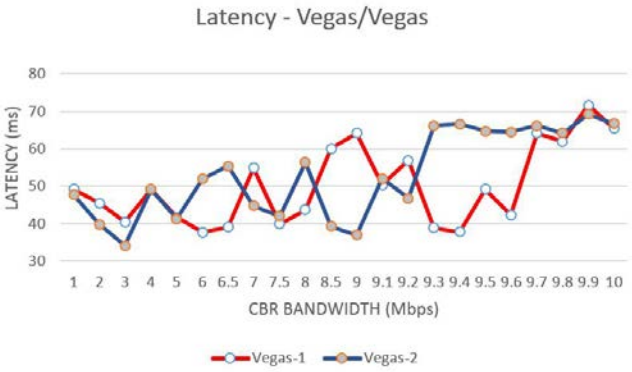


Fig13: Bandwidth V/S Bandwidth



As we have as of now specified in Reno/Reno subsection, the oscillations in throughput are acceptable. Thus, it is fair for two Vegas TCP running on the same system.

D. NewReno/Vegas

On Figure 5, we assign NewReno to TCP1 and Vegas to TCP2 link. Figure 16, Figure 17 and Figure 18 show the comparison of throughput; packets drop rate and latency between the NewReno and Vegas agents in the same network.

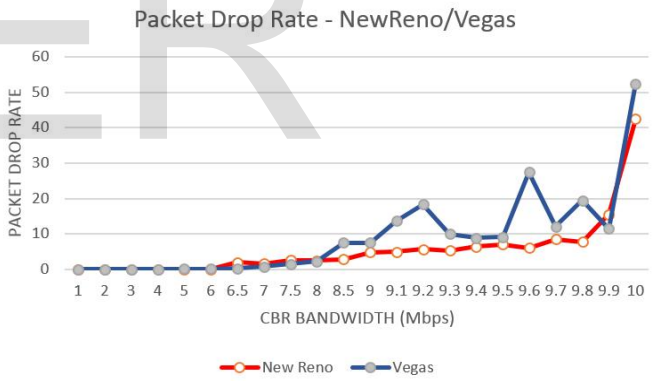
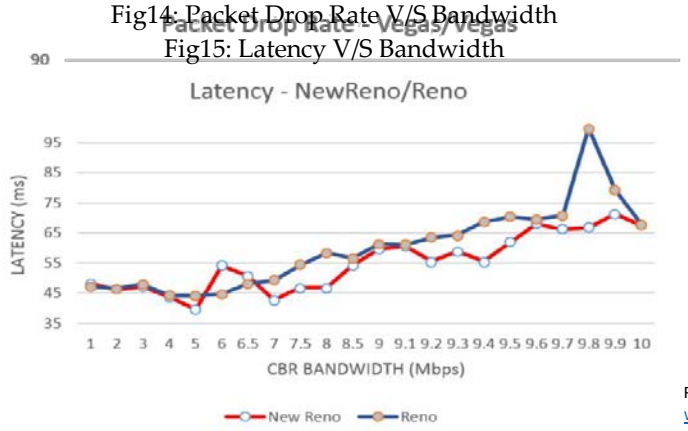


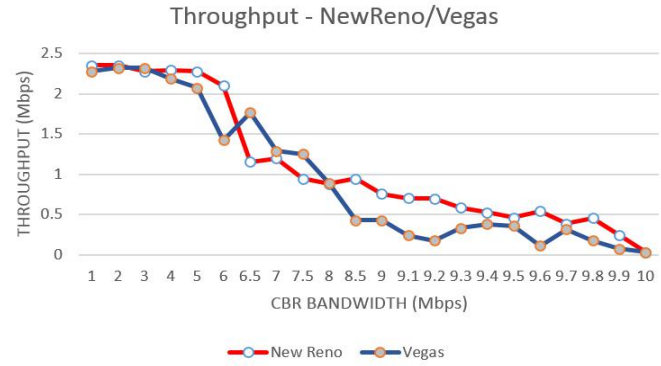
Fig16: Throughput V/S Bandwidth

Fig17: Packet Drop Rate V/S Bandwidth

Fig14: Packet Drop Rate V/S Bandwidth

Fig15: Latency V/S Bandwidth





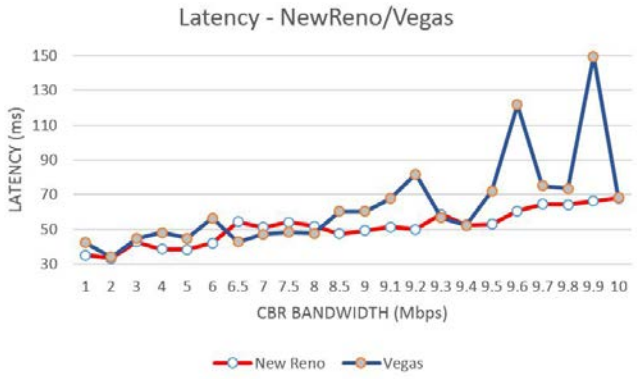


Fig18: Latency V/S Bandwidth

With an increase in CBR flow, the throughput of NewReno becomes higher than the throughput of Vegas TCP and this division becomes larger gradually. Also, it runs more steadily and loses less number of packets than Vegas. For latency, they remain close for a particular period, but when CBR reaches 8Mbps, it suddenly goes to a higher latency than NewReno. So we confirm that NewReno is unfair to Vegas. This is because Vegas recognizes congestion at an early stage and it reduces its sending rate which allows it to provide more bandwidth to NewReno and hence, NewReno becomes dominant.

From the above experiment, we can infer that same TCP variation is typically reasonable/fair to one another on through-put, packet drop rate, and latency, while a combination of different TCP variations can't be entirely reasonable to each other. Since these differences in variant pairs have execution points, transmission rate, congestion avoidance window or re-transmission strategy. This distinction may make one variation get suppressed by another. For instance, the early congestion detection highlight of Vegas makes it suppressed by NewReno, and once NewReno gets to be dominant, Vegas will always consider the network to be congested, and it will diminish its sending rate, thus never gets predominant back again.

V. EXPERIMENT 3: INFLUENCE OF QUEUING

The network topology and flow setup of experiment 3 are as Figure 19

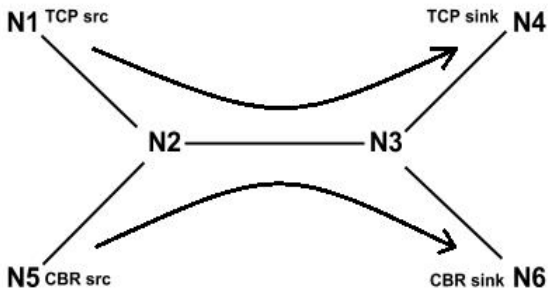


Fig19: Experiment 3 Network Topology

The total duration of this experiment is 30 seconds. We first start the TCP flow and start CBR flow after 11 seconds when TCP reaches its steady state. We stop CBR flow at 25th second and TCP flow at 30 seconds. The bandwidth of every link is 10Mbps with a delay of 10ms, TCP window size is 120, TCP maxcwnd is 150, and the CBR stream rate is 1Mbps. We recreate four sets of TCP variations and queue types: Reno with DropTail, Reno with RED, SACK with DropTail, and SACK with RED. The assessment procedure depends on the average bandwidth and average latency of TCP stream. Concerning the network topology from Figure 19 we assign one of the TCP variants (Reno or SACK) between N1 and N4 and CBR between N5 and N6.

A. Average Bandwidth

It can be surmised from Figure 20 that throughput of Reno RED and SACK RED is smaller when compared with the other two because, in RED, packets are dropped by the statistical algorithm. Whereas in DropTail packets are dropped independently when the queue gets full irrespective of flow type.

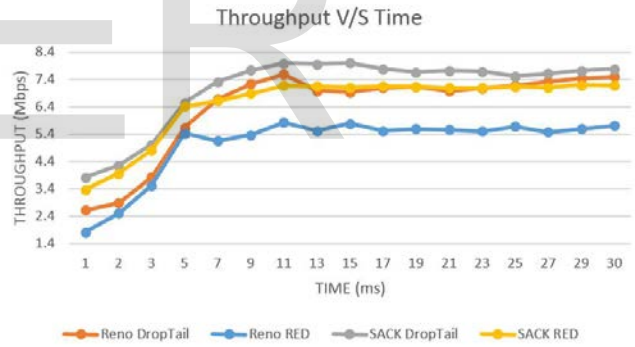


Fig20: Throughput V/S Time

After CBR flow begins at 11 seconds, the DropTail TCP flow will still have higher bandwidth than the RED. Hence, for TCP Reno and SACK, DropTail queue has better bandwidth performance as compared to RED.

B. Average Latency

From Figure 21 we can conclude that Reno and SACK have the lowest latency with DropTail and RED queue respectively. However, when Reno and SACK both are with DropTail, their performance is almost the same because RED is fairer than DropTail. In RED, a host's packet being dropped is a result of the amount of data present in the queue. Hence, irrespective of CBR flow, the packet drop rate will not change.

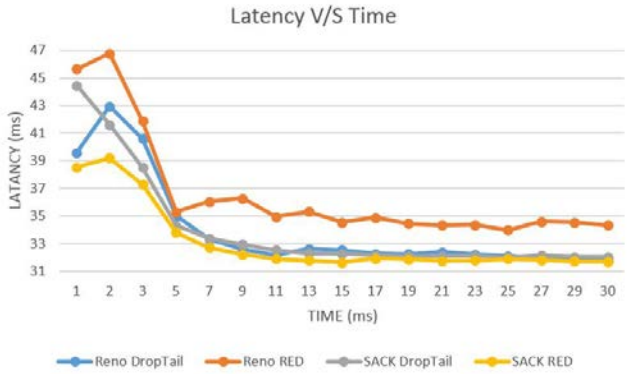


Fig21: Latency V/S Time

Referring the above graph, we can presume that RED works well by TCP SACK. Concerning latency, SACK with RED has better performance than SACK with DropTail, which is because RED drops packets unexpectedly while SACK retransmits packets selectively. In spite of the fact that SACK with DropTail takes higher bandwidth than SACK with RED and SACK with RED still give more stable and robust transmission particularly in a congested network.

VI. CONCLUSION

This paper has 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 to scale the performance, fairness, and influence of queueing. The results from these experiments cannot be taken as granted because these experiments were performed with a simple network topology and to jump to a conclusion we need to do further research with different network topologies. From the above-conducted experiments, we can conclude:

1) TCP Vegas performs relatively better than Tahoe, Reno and NewReno TCP.
2) Networks with same TCP variants are usually 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 higher stability and least latency.

REFERENCES

[1] Wikipedia, TCP congestion-avoidance algorithm, https://en.wikipedia.org/wiki/TCP congestion-avoidance algorithm

[2] The Doctoral Journey, One Sample, and Paired Sample t-Test Tutorial, https://www.youtube.com/watch?v=TCqVAtlFhN8

[3] Optimization of Communication Systems Lecture 6: Internet TCP Congestion Control, Professor M. Chiang Electrical Engineering Department, Princeton University.

[4] Jae Chung & Mark Claypool, NS by Example

[5] Evaluation Of Different TCP Congestion Control algorithm using NS-2, Hui (Hilary) Zhang & Zhengbing Bian