PERFORMANCE ANALYSIS OF TCP VARIANTS

Anuj Tyagi
Email: tyagi.an@husky.neu.edu

Abstract:

TCP is an indispensable protocol of the Internet protocol suite. It provides reliable, connection oriented, ordered, and error free stream of packets between applications over a network. In this paper we explore the performance of different TCP variants Tahoe, Reno, New Reno, and Vegas under congestion introduced in the network. We use NS-2 to carry out our simulations for these variants, and analyze the results with respect to throughput, latency, and packet drop rate in order to gauge the best variant of them all. We also study the performance of Reno, and SACK under the Droptail, and RED queuing algorithm.

1. INTRODUCTION

Network congestion has been a problem since the existence of the internet. The initial TCP protocol developed did not account for congestion within the networks, but only allowed flow control. This led to the research and development of many variants of TCP each one being advantageous than the previous one. A brief description of the TCP variants used in the experiments is provided below.

1.1 TCP Tahoe

TCP TAHOE is the most basic of the TCP protocols, and has 2 congestion control methods viz slow start, and congestion avoidance. It maintains a congestion window to limit the total number of unacknowledged packets, and a slow start threshold to increase the congestion window. If a packet is not received before the RTT period expires, then TCP assumes that the packet is lost due to congestion in the network, and starts again with the lowest possible slow start threshold value.

1.2 TCP Reno

RENO on the other hand, makes use of Tahoe's slow start and congestion avoidance along with it's own method called, fast recovery. Here, if three duplicate acknowledgements are received for a packet that is sent, then Reno will retransmit that particular packet again, and will then enter slow start phase at a value half that of the congestion window.

1.3 TCP New Reno

NEW RENO was introduced as Reno was not able to handle multiple packet drops within the same window. It improves the retransmission during the fast recovery phase. New Reno does not come out of fast recovery phase until it receives ACKs for all the packets that were present whilst entering the fast recovery phase.

1.4 TCP Vegas

VEGAS measures the RTT for every packet that is sent and compares it with the RTT of the received packet. The sending window is decreased if the obtained RTT is smaller than the initial RTT, and increased if the obtained RTT is greater than the initial RTT.

1.5 SACK

SACK is another TCP variant wherein the receiver explicitly mentions the packets that are acknowledged. When this TCP enters fast recovery phase it uses a parameter called pipe which denotes the number of packets that are still unacknowledged. New packets are transmitted if there are no unacknowledged packets, thus multiple packets can be retransmitted in SACK in just 1 RTT.

2. METHODOLOGY

2.1 Network Topology

The network topology is depicted in Figure 2.1. The network consists of 6 nodes numbered N1 to N6. The bandwidth of each link is 10Mbps, and the delay of each link is 10ms.

For experiment 1, the TCP flow is set with source at node N1 and sink at N4, whereas the CBR flow is set with source at N2 and sink at N3.

For experiment 2, the TCP flows are set from source at node N1 and sink at N4, and source at N5 to sink at N6, whereas the CBR flow is set with source at N2 and sink at N3.

For experiment 3, the TCP flow is set with source at node N1 and sink at N4, while the CBR flow is set from source at N5 to sink at N6.

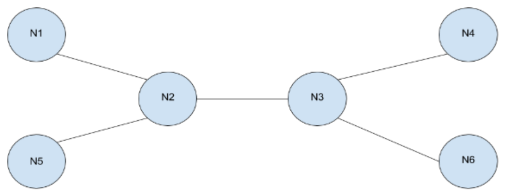

Figure 1

Using the above topology, trace files are generated which are then parsed in order to analyze the behavior of each TCP variant with respect to the throughput, latency, packet drop rate, fairness the TCP variants to one another, and 2 different queuing algorithms.

Also note that for all the graphs plotted for experiment 1 and 2 we have started TCP flows first and then the CBR with a queue size of 10. For experiment 3 we have varied the queue size from 5 to 25, and have plotted the graphs for queue size of 10.

2.2 Experiment 1

For this experiment we analyze the performance of the TCP variants under different load conditions ie in the presence of a Constant Bit Rate(CBR) flow. The variants used here are TCP Tahoe, Reno, New Reno, and Vegas. To introduce randomness in the network we have different combinations of TCP variants and the CBR flow:

1. Start CBR and TCP flow at the same time
2. Start CBR first and then TCP
3. Start TCP first and then CBR
4. End CBR first and then TCP
5. End TCP first and then CBR
6. Vary the start and end times of TCP and CBR

Using the above combinations, we analyze the throughput, the latency, and the packet drop rate.

2.3 Experiment 2

For this experiment we analyze the fairness between the TCP variants. We set two TCP flows here, and one CBR flow. CBR flow is varied from 1Mbps to 10Mbps. A combination similar to experiment 1 is done by using the second TCP flow instead of the CBR.

By obtaining the throughput, latency, and the packet drop rate, we analyze the fairness of the TCP variants to each other.

We make use of the following scenarios to find out the fairness between the TCP variants. The two TCP flows are used in different combinations with CBR are given below:

1. New Reno and Reno
2. New Reno and Vegas
3. Reno and Reno
4. Vegas and Vegas

2.4 Experiment 3

For this experiment, we will analyze the effects of two queuing algorithms viz. DropTail, and Random Early Drop(RED) on the TCP Reno, and SACK. The variation of the throughput and the latency will help us determine the better variant.

3. ANALYSIS

This section consists of graphs of the throughput, latency, and packet drop rate for experiments 1 and 2, and the throughput for experiment 3. To analyze the trace files, we use awk scripts to parse the trace files, and use Microsoft Excel or Xgraph to plot the graphs.

3.1 Throughput

The throughput is defined as the rate at which packets are delivered over a link for a given period of time. The time difference between the first TCP packet sent into the pipe and the last TCP packet received from the pipe is calculated. The total number of bits accumulated at the destination node is then divided by this time difference to obtain the throughput.

3.2 Drop Rate

The drop rate is defined as the ratio of number of packets that are dropped to the total number of packets that are sent. In order to calculate the drop-rate we can use the event d which gives the number of packets that are dropped, or we can obtain the difference between the number of packets sent and the number of packets received to that of the total number of packets sent.

3.3 Latency

In order to calculate the latency, we obtain the Round Trip Time (RTT). We capture the time a packet was initially sent, and obtain the ACK that is generated for the packet. The difference between the initial time, and the time the ACK was received gives us the RTT of the packet. However, for experiments 2 and 3, we also calculate the end-to-end latency.

3.4 TCP Performance Under Congestion

As per experiment 1 the throughput, latency, and the drop rate are calculated by varying the the CBR flow from 1Mbps to 10Mbps with a step size of 0.5Mbps. The network topology used is shown in Figure 1. The measurements for the experiment are made for different scenarios as mentioned in Section 2.2 while maintaining the same packet size, and queuing algorithm. The graphs for throughput, latency, and drop rate are plotted against the CBR flow, and are shown in the figures 2, 3, and 4.

This experiment answers the questions; Which TCP variant has the highest average throughput, the lowest average latency, and the fewest drops.

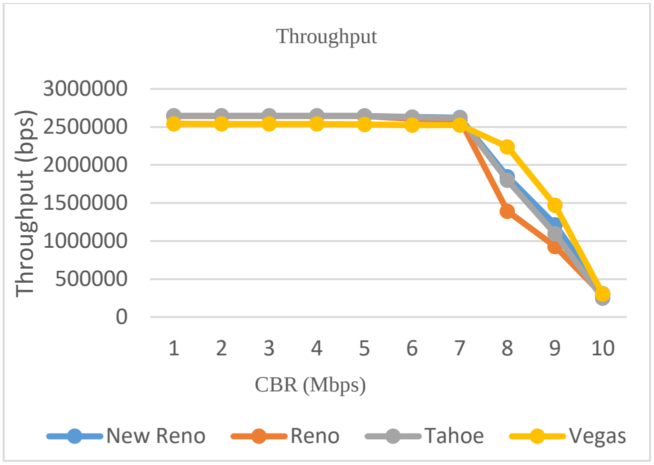

Figure 2 Average Throughput of TCP Variants vs CBR

From Figure 2 we can see that the average throughput is around 2.5Mbps before introducing the CBR flow, and the TCP Variant Vegas has a higher average throughput than the others. This is supported by observing the graphs for the other scenarios as well. The reason is because Vegas is able to detect congestion at an earlier stage using its delay based congestion avoidance as opposed to the ACK based method used by the other variants. On the other hand, Reno shows the lowest average throughput because its algorithm is not designed to handle multiple packet loss.
Furthermore, the T-Test of Vegas with respect to the other variants resulted in values greater than 0 which confirms the better performance of Vegas.

T- Test is calculated by $\frac{\boldsymbol{Mean(Vegas) - Mean(Variant)}}{\sqrt{\frac{\boldsymbol{Variance(Vegas)}}{\boldsymbol{Number\ of\ Obs}} + \frac{\boldsymbol{Variance(Variant)}}{\boldsymbol{Number\ of\ Obs}}}}$

T- Test of Vegas over Tahoe = 0.0162
T- Test of Vegas over Reno = 0.1443
T- Test of Vegas over New Reno = 0.04909

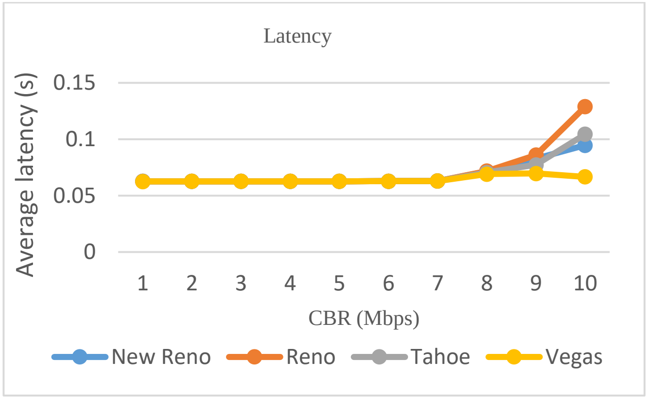

Figure 3 Average Latency of TCP Variants vs CBR

Figure 3 shows the average latency for all the TCP variants over CBR flow. It can be seen that Vegas has the lowest latency of all the variants. When CBR is around 7.5Mbps, notice the increasing latency for all the variants, till 10 Mbps, except for Vegas which, thanks to its latency measurement using RTT is able to detect congestion, and thus have low latency till the end of the simulation.

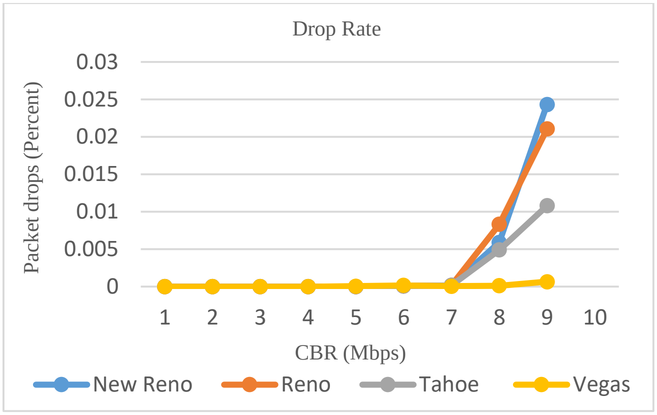

Figure 4 Average Packet Drop rate of TCP Variants vs CBR

From Figure 4, we observe the packet drop rate for all the variants against CBR flow that varies from 1Mbps to 10Mbps. Vegas has almost no packet drops even when the CBR rate is around 7.5Mbps. At this point all the other variants experience packet drops and enter the congestion avoidance phase.

In conclusion from the observations made from the graphs, it is clearly visible that TCP Vegas is a better variant when compared to New Reno, Reno, and Tahoe as it out performs these variants in average throughput, latency, and packet drop rate.

3.5 Fairness between TCP Variants

As per experiment 2, the average throughput, and the average latency are calculated by varying the CBR flow from 1Mbps to 10 Mbps with a step size of 0.5Mbps. The network topology used is that of Figure 1. The measurements are made for different combinations of both TCP flows, and CBR flow. However, the best observations are obtained when starting both the TCP flows together, and the CBR flow after a few seconds. Figure 5, 7, 9, and 11 shows graphs for the throughput, and figures 6, 8, 10, and 12 show the latencies in order to determine fairness between the two TCP variants in the order of New Reno vs Reno, New Reno vs Vegas, Reno vs Reno, and Vegas vs Vegas. **The observations from this experiment answers if variants are fair or unfair to each other, and what causes the fairness.**

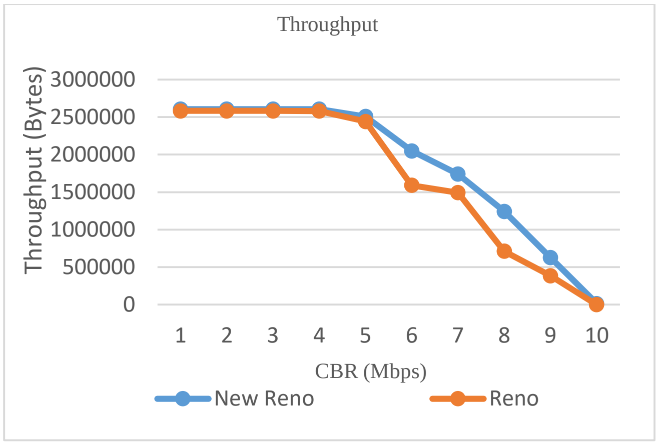

Figure 5

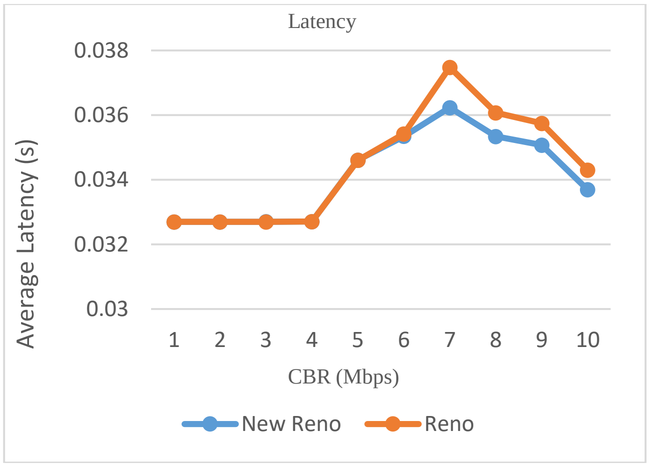

Figure 6

Figure 5 and 6 shows the throughput and latency comparison between New Reno and Reno. The average throughput of each of the TCP variants is 2.5Mbps, and when CBR flow is 5Mbps, it amounts to a total of 10Mbps. Thus, it is observed that the throughput of New Reno is much better when CBR crosses 5Mbps, as New Reno starts to use more bandwidth and competes with Reno. This is due to its improved retransmission during the fast recovery phase as opposed to Reno's waiting time for a triple duplicate ACK, and also because Reno's algorithm cannot handle multiple packet loss well.

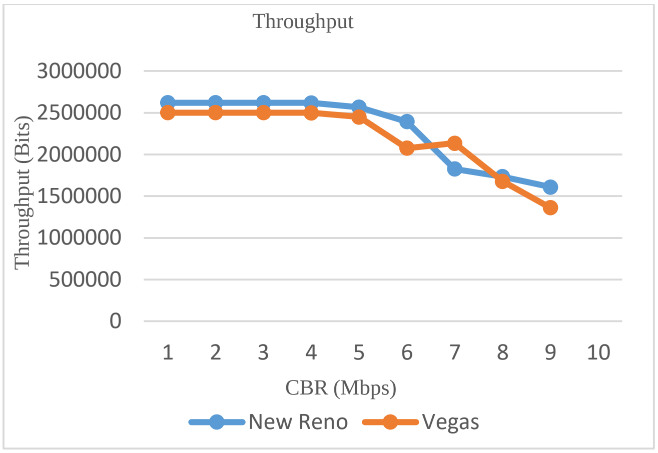

Figure 7

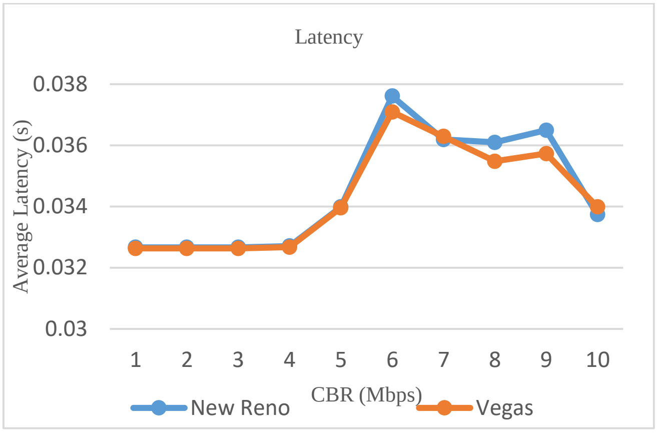

Figure 8

From Figures 7, and 8 we can see that New Reno is better than Vegas, as Vegas's congestion detection algorithm detects the congestion prior to New Reno and starts to decrease its throughput thus giving the entire bandwidth to New Reno. It can be seen here that there is unfairness among different TCP variations.

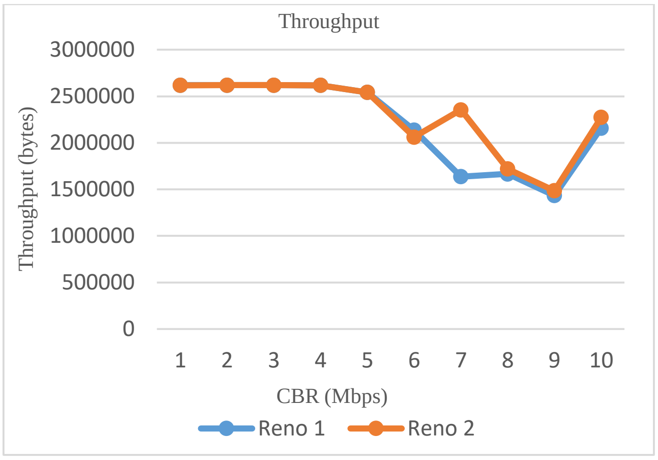

Figure 9

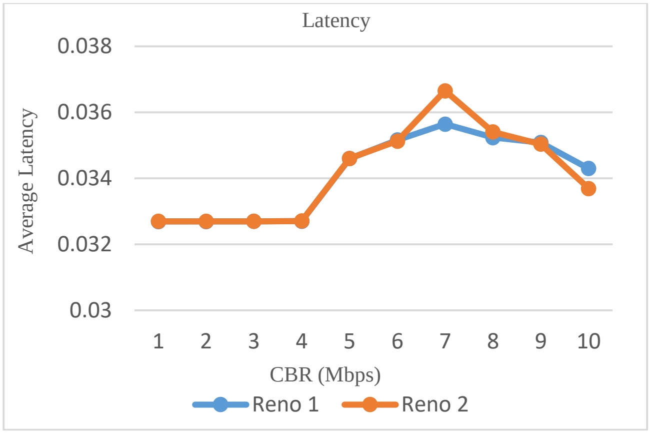

Figure 10

From figures 9, and 10 we can observe that throughput of both Reno 1 and Reno 2 are almost similar, thus share the bandwidth equally. This is because both of them use the same algorithm thus handling congestion in the same way. When bandwidth has reached its maximum capacity both start to reduce the number of packets. This shows that the same variants are being fair to each other.

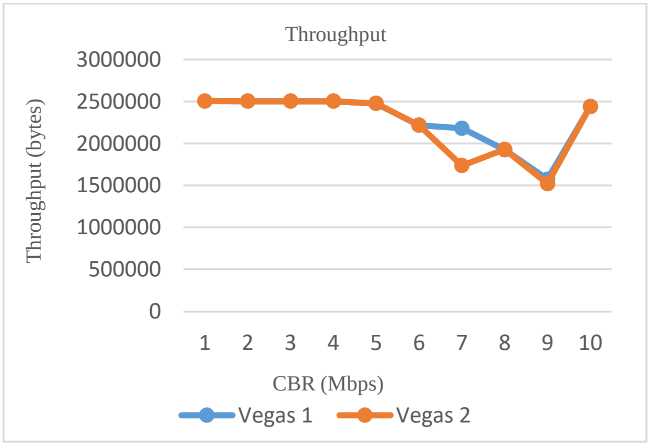

Figure 11

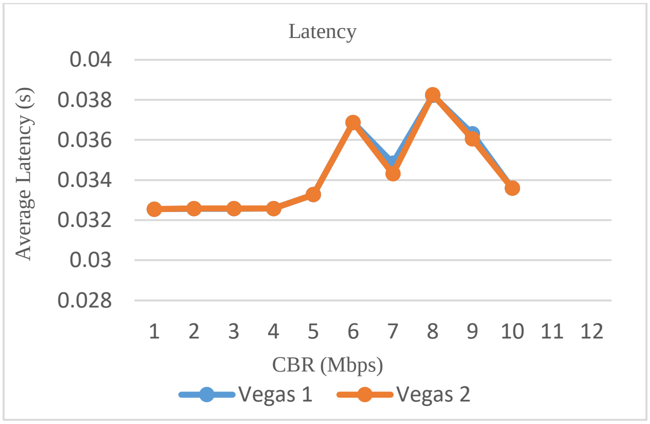

Figure 12

Figures 11, and 12 demonstrate the fairness between two TCP Vegas flows. This is similar to the observation from Reno vs Reno.

3.6 Influence of Queuing

As per experiment 3 the throughput is calculated for Reno and SACK TCP flows under the effect of 2 different queuing algorithms viz Droptail, and Random Early Detection (RED) and graphs are plotted by varying the time. Here, we try to find if: the different queuing algorithm affect the fairness in the bandwidth available to each of the TCP variants, RED is a good algorithm for dealing with SACK, and the end-to-end latency differ between the two queuing algorithms.

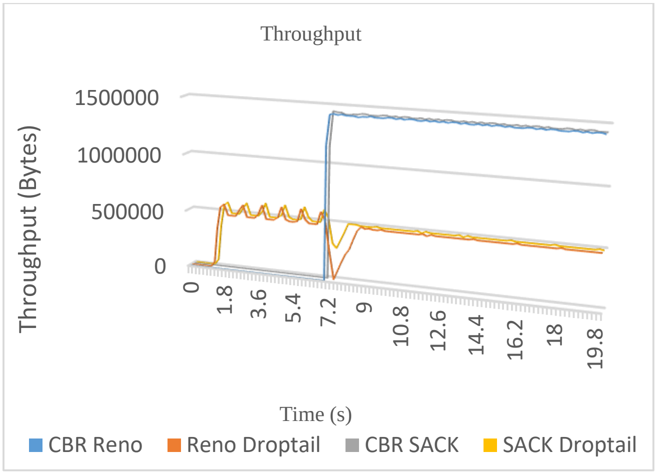

Figure 13

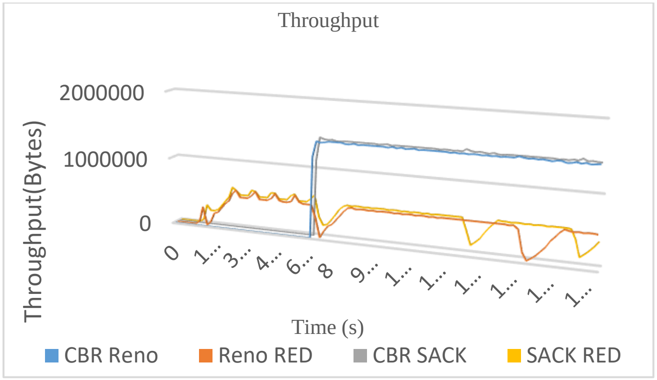

Figure 14

Average Latency (Reno Droptail)= 0.03296306 s
Average Latency (Reno RED) = 0.03092672 s
Average Latency (SACK Droptail)= 0.03290896 s
Average Latency (SACK RED) = 0.03130354 s

From Figure 13, and 14 we can observe that both the queuing algorithms the throughput of Reno and SACK are reduced substantially in the presence of CBR flow, but the bandwidth is utilized properly by both the variants before the CBR flow, with SACK having a slightly higher throughput. Due to the CBR flow both

the algorithms start to drop packets, and the variants detect this and reduce their throughput accordingly, but Reno faces higher drops when compared to SACK for both the queuing algorithms.
Comparing RED, and Drop Tail, RED avoids TCP global synchronization thus ensuring that packets are dropped equally from both Reno, and SACK unlike Drop Tail which drops packets as soon as the queue is filled. This leads to reduced throughput for both Reno and SACK while using Drop Tail. For RED the drops are based on statistical analysis, and on the amount of bandwidth a flow utilizes. This shows that RED is more fair than Drop Tail.

The average latency for SACK and Reno when using RED is lower, and the average throughput for SACK while using RED is higher than that of SACK when using Drop Tail, because RED avoids TCP Global Synchronization, thus we can say that RED is a good algorithm for dealing with SACK.

4. CONCLUSION

We can conclude that the characteristics of different TCP variants differ for Throughput, Latency and Drop rate based on the amount of congestion within the network.

From Experiment 1, the data provided from the graphs and the statistical analysis parameters viz T-test, and variance, and standard deviation it can be concluded that the performance of Vegas is much better than that of the other TCP variants in terms of Throughput, Latency, and Packet drop rate.

From the observations in Experiment 2, we can conclude that two same TCP variants are fair to each other when they are started together, whereas if the variants are not the same, then they behave unfairly towards each other when they are started together.

From the observations in Experiment 3, we can conclude that RED queuing performs better when compared to Drop Tail queuing as it treats the variants fairly, and avoids TCP Global Synchronization phenomenon.

The performance analysis of TCP variants still remains unexplored and with the widespread use of the internet, the number of variants is also increasing continuously, thus more research and analysis must be done on this area in order to find out the best TCP variant that could be developed in the future.

5. REFERENCES

- Simulation-based Comparisons of Tahoe, Reno, and SACK TCP by Kevin Fall and Sally Floyd
- https://en.wikipedia.org/wiki/Random_early_detection
- Congestion Avoidance and Control” by V. Jacobson
- http://www.isi.edu/nsnam/ns/tutorial/index.html
- https://en.wikipedia.org/wiki/TCP_congestion-avoidance_algorithm