Performance Analysis of TCP Variants

Mayank Narasimhan
College of Computer and Information Science
Northeastern University, Boston, MA
mayankn@ccs.neu.edu

Narendran Komaragounder Palanivelu
College of Computer and Information Science
Northeastern University, Boston, MA
naren90@ccs.neu.edu

Abstract — This paper analyses and compares different TCP variants (Tahoe, Reno, New-Reno, and Vegas) based on different congestion control and avoidance mechanisms, which have been proposed in these TCP/IP protocols. The paper also compares the performance of the TCP variants against different queuing management algorithms like Drop-Tail and Random Early Detection (RED), in a packet switched network.

Our study showed that TCP Vegas resulted in showing lesser latency and fewer packet drops when compared to other TCP variants. While comparing the queuing disciplines the results showed that TCP SACK with drop tail queuing, shared the bandwidth fairer than with RED.

Keywords—TCP, Reno, New-Reno, Vegas, Drop-Tail, RED

I. Introduction

TCP is the most widely used transport protocol for reliable connection, used by majority of the current Internet applications. TCP boasts of robust features like congestion detection, congestion avoidance and error checking, the extent and flavor of which varies with the multitude of network implementations and parameters that exist in the real world today. Hence a detailed study of the functioning of the various TCP variants is essential for any network researcher to help understand the network dynamics better.

In this paper we study the performance of the common TCP variants (Tahoe, Reno, New-Reno, Vegas) under the influence of varying load conditions and then analyze and illustrate how these variants respond to different types of network congestion when put against contending traffic. One of TCP's robust features is its reaction to congestion and ability to still maintain reliability of packet transmissions. All the above-mentioned TCP variants have their own implementations of these mechanisms and this paper is an effort to analyze and understand these mechanisms used the TCP variants.

Every forwarding device on a network (router) has a queue associated to it, which must implement some queuing discipline to manage the buffering of packets while waiting to be retransmitted. Drop-Tail is one of the simplest queuing algorithm in which the packets are simply dropped when the queue becomes full. RED effectively signals TCP's congestion control algorithm by dropping packets in the queue before congestion occurs based on the average queue length. In this paper, we study and compare the Drop-Tail and RED queuing disciplines.

The dumbbell network topology, with 6 nodes and 5 links, as shown in Figure1, was used to conduct the experiments on TCP variants (Tahoe, Reno, New-Reno and Vegas). The same topology was also used to analyze the working of queuing disciplines, Drop-Tail and RED over TCP variants, SACK and Reno.

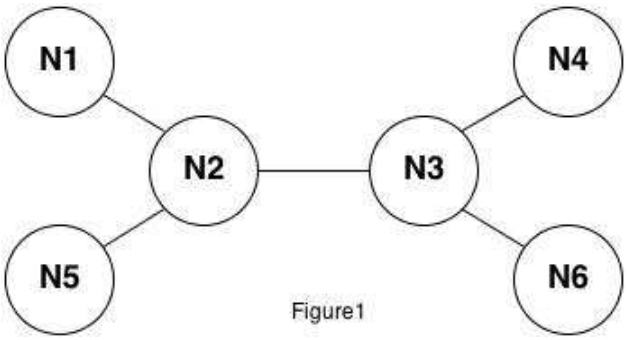


Figure1

II. Methodology

NS-2 is an open source event simulator used by the research community for research in networking. It has support for both wired and wireless networks and can simulate several network protocols such as TCP, UDP and Multicast routing etc. NS-2 is object-oriented simulator written in C++ with the TCL interpreter as the front end. The NS-2 simulation scripts for our study are written in OTcl. The trace files generated are then parsed and processed using AWK scripts for extraction the data from the columns of the trace file. The results obtained were plotted on graphs to analyze the outputs.

In the first experiment, the topology shown in Figure1 was used. The nodes in the topology were setup using 10Mbps links having a delay of 10ms with a Drop-Tail queue discipline. The node, N1 was attached to an FTP application over a TCP flow and the node, N5 was attached to a Constant Bit Rate (CBR) flow over a UDP connection. A TCP stream was started between the nodes N1 and N4 and a CBR flow was started between the nodes N2 and N3. The average throughput, latency and packet drop rate of TCP Tahoe, Reno, New-Reno and Vegas were obtained from this experiment and were plotted on graphs as a function of the CBR flow rate which varied from 1Mb to 9Mb. The simulation was run for 300 seconds to give TCP enough time to ramp up and stabilize.

In the next experiment, the same topology as in Figure1 was used and two TCP streams were compared in contention with a CBR flow in between. One TCP stream was started from node N1 to N4 and another from N5 to N6. A CBR flow was then introduced between the nodes N2 and N3 to create congestion in the network. The experiment was conducted on the following pairs of TCP variants with one at each TCP stream: Reno (N1-N4) / Reno (N5-N6), New-Reno (N1-N4) / Reno (N5-N6), Vegas (N1-N4) / Vegas (N5-N6), New-Reno (N1-N4) / Vegas (N5-N6). The average throughput, latency and packet drop rate for these TCP combinations were obtained and plotted as a function of the CBR flow rate which varied from 1Mb to 9Mb. The simulation was run for 300 seconds.

The last experiment used the same topology as in Figure1, to compare the performance of the TCP variants, SACK and Reno when used with the queuing disciplines, Drop-Tail and

RED. A TCP stream (Reno or SACK) was started from node N1 to N4 and a CBR flow, with half of the bottleneck bandwidth capacity (5Mb), from N5 to N6. The TCP stream was configured to use either of the two queuing disciplines, Drop-Tail and RED. The average throughput and latency of the TCP stream was observed during bottleneck congestion and was plotted over discrete time intervals. The CBR flow was started 150 seconds after the TCP stream was started, to give enough time for TCP to ramp up and stabilize, and the entire simulation was run for 300 seconds.

III. TCP performance under congestion

In this experiment, the TCP stream was subjected to varying congestion from a CBR flow. The behavior of TCP was then studied under increasing load conditions and its performance was quantified in terms of the average throughput, latency and packet drop rate over the varying CBR flow rate.

A. Throughput

Throughput is the rate at which packets are received at a given network device or rate of successful packet delivery in any given amount of time.

The average throughput of the TCP variants, Tahoe, Reno, New-Reno and Vegas were compared under the congestion load created by the CBR flow rate, to study TCP's behavior during congestion, in terms of the throughput and bandwidth utilization. The graph in Figure2 depicts the average throughput of the TCP variants versus the CBR flow rate. Right from the beginning, Vegas has the best throughput, followed by New-Reno. When the CBR flow rate increases and creates a congestion bottleneck, Vegas continues to have a better throughput followed by New-Reno. Tahoe has the least throughput out of all the variants since it is forced to reset to slow start each time it drops packets during congestion. Hence, its congestion window is reset to 1 and then takes time to ramp up again to utilize the available bandwidth. New-Reno on the other hand, fast-retransmits after every duplicate ACK and doesn't go back to slow start during congestion and is hence able to utilize more of the bandwidth available to it resulting in a better throughput than Tahoe and Reno. Vegas proactively detects congestion at an incipient stage by using packet delay rather than packet loss as a signal for congestion and then accordingly determines the rate at which to send packets. This results in a better throughput and bandwidth utilization as compared to the other variants Tahoe, Reno and New-Reno.

B. Latency

Latency is a measure of the delay experienced by packets in a network to complete one round-trip from the source to destination and then back to the source.

The average latency of the TCP variants were compared over the increasing CBR flow to analyze its behavior during congestion in terms of round-trip time (RTT) of each packet. The graph in Figure3 depicts the average latency of TCP variants versus the CBR flow rate. Vegas showed a lower average latency than Tahoe, Reno and New-Reno since it does not solely depend on packet loss as a sign of congestion.

Unlike other TCP variants, Vegas does not have the problem of having to wait for enough duplicate ACKs to detect packet loss and it also implements a modified slow start algorithm, which prevents it from getting affected by the network congestion. When it receives a duplicate ACK, it checks if the current packet transmission time is greater than RTT estimate; if it is, then the packet is retransmitted immediately without having to wait for ACKs (3 for Reno) or Retransmission Timeout (RTO).

C. Packet Drop Rate

Packet drop rate is the rate at which packets are lost or discarded from the network when a device is overloaded and is unable to accept incoming data.

The average packet drop rate of the TCP variants, Tahoe, Reno, New-Reno and Vegas were compared under the congestion load created by the CBR flow rate, to study TCP's behavior during congestion, in terms of the packet drop rate. The graph in Figure4 depicts the average packet drop rate of the TCP variants versus the CBR flow rate.

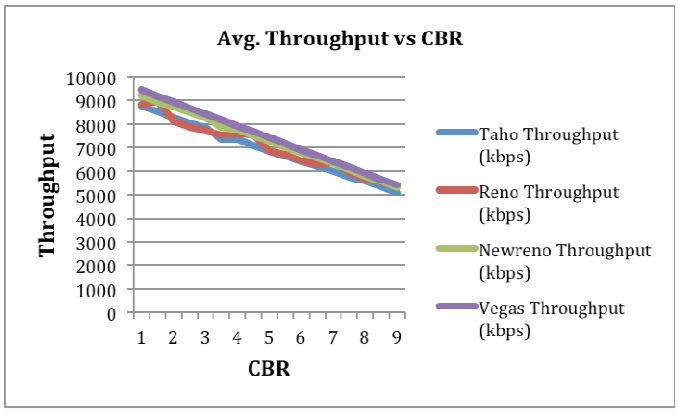


Figure2

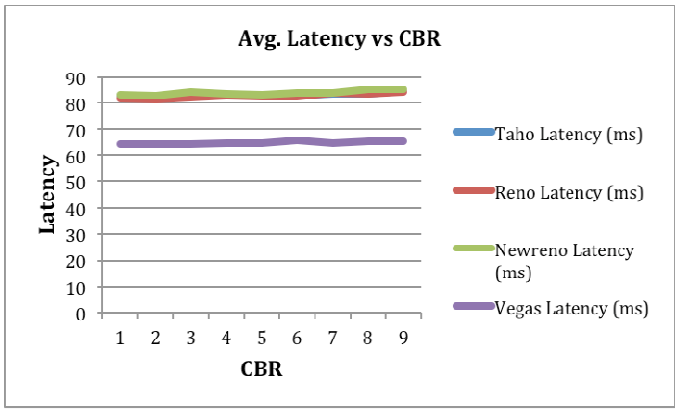


Figure3

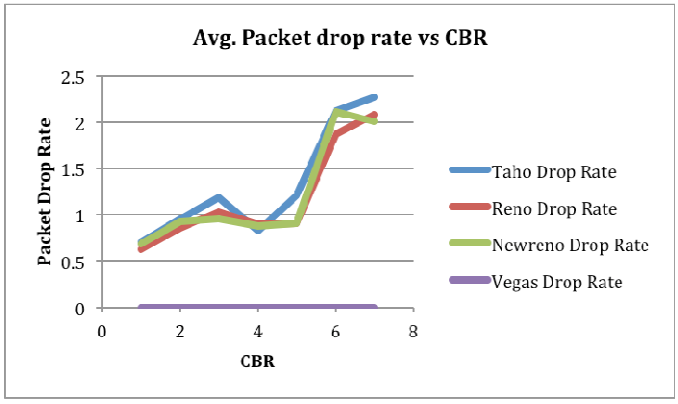


Figure4

IV. Fairness between TCP variants

In this experiment, two TCP streams were subjected to increasing congestion from a CBR flow. The behavior of the two TCP variants was then studied under increasing load conditions and a measure of their fairness was quantified in terms of the average throughput, latency and packet drop rate over the varying CBR flow rate.

Fairness between TCP variants is a measure of network utilization such that no larger share of the network is used by a TCP flow than another comparable TCP flow.

A. Reno-Reno

When the experiment was conducted with two TCP Reno flows at both ends of the network topology, we observed that the average throughput, latency and packet drop rate were more or less similar for both flows even as the CBR was increased to create congestion. This shows that TCP Reno is fair when coupled with another TCP Reno flow in the network as seen in Figure5.

B. New-Reno-Reno

When the experiment was conducted with New-Reno at one end and Reno at the other end of the network topology, we observed that the average throughput of the New-Reno stream was more than that of the Reno stream. This behavior of New-Reno can be associated to its feature of Fast-Retransmit for each and every duplicate ACK instead of waiting for three duplicate ACKs, which is the case in Reno. New-Reno does not exit fast-recovery until all the data, which was outstanding at the time it entered fast-recovery, is acknowledged. Hence it overcomes the problem faced by Reno of reducing the congestion window multiple times. This is evident from the graph in Figure6.

Since the throughput of the TCP flow using New-Reno is higher, the packets in that flow have a lesser RTT. This is evident from the graph in Figure6 which shows New-Reno to have a lower average latency than Reno.

New-Reno overcomes the problem faced by Reno, which suffers where there are multiple packet drops. This is due to the modified fast-recovery phase in New-Reno, which allows multiple retransmissions. Hence it has a lower average packet drop rate than Reno as shown in the graph in Figure6. This shows that New-Reno is unfair when coupled with a Reno flow in the network.

C. Vegas-Vegas

When the experiment was conducted with two TCP Vegas flows at both ends of the network topology, we observed that the average throughput, latency and packet drop rate were more or less similar for both flows even as the CBR was increased to create congestion. This shows that TCP Vegas is fair when coupled with another TCP Vegas flow in the network as seen in Figure7.

D. New-Reno-Vegas

When the experiment was conducted with New-Reno stream at one end and Vegas at the other end of the network topology, we observed that the average throughput of the New-Reno stream was more than that of the Vegas stream. This behavior is due to the fundamental difference between congestion detection mechanisms of both these TCP variants. New-Reno is packet-loss based whereas Vegas is packet-delay based. New-Reno continuously increases its congestion window when it detects packet loss. Vegas on the other hand increases or decreases its congestion window based on the observed RTT estimate of the transmitted packets. This aggressive increase of window size by Reno, affects the RTT estimates of the other TCP stream, Vegas. Therefore Vegas continuously tries to decrease its window size assuming congestion in the network.

Individually, Vegas achieves a higher throughput than New-Reno. However when New-Reno shares the bottleneck link with Vegas, the performance of Vegas is degraded which is not originally expected as seen in Figure8.

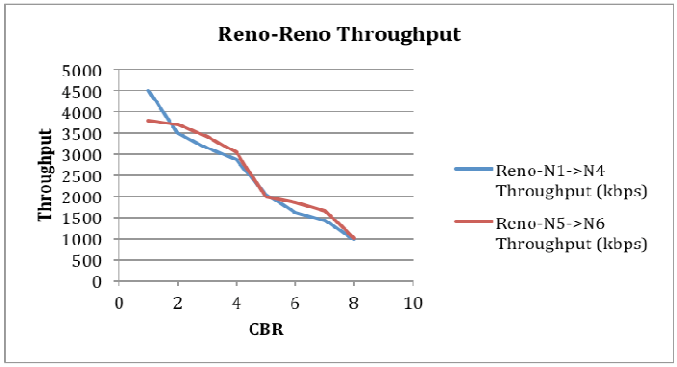
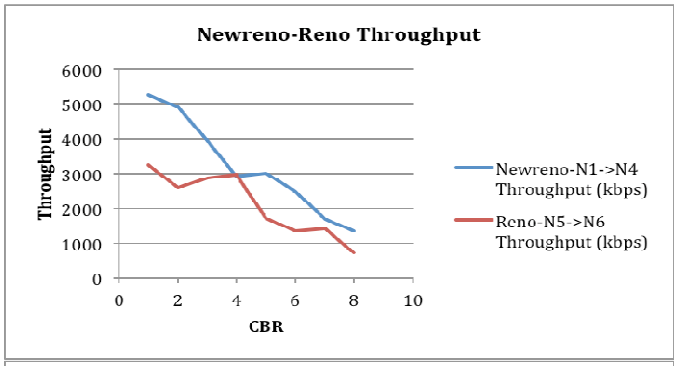
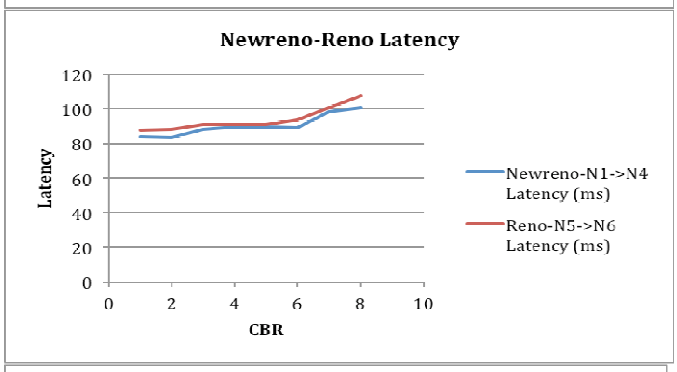


Figure5





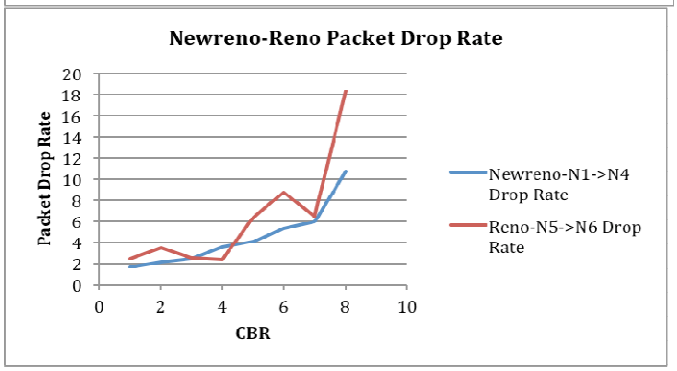


Figure6

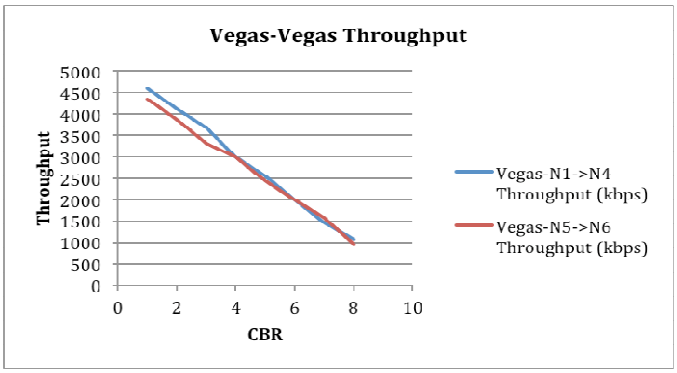


Figure7

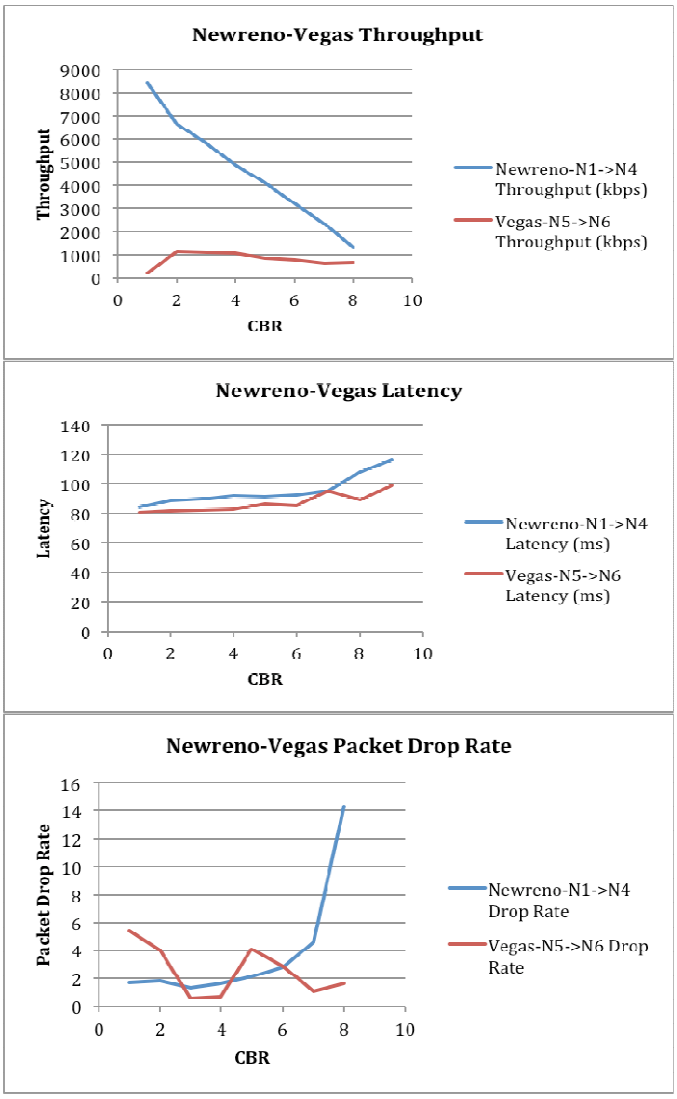


Figure8

V. Influence of queuing disciplines on TCP Reno and TCP SACK flows

The queuing mechanisms explained in this experiment are Drop-Tail and RED. Drop-Tail can be classified as a queue management algorithm since it is basically a simple FIFO mechanism where packets are thrown away when the queue length exceeds the buffer capacity. RED is an effective queuing mechanism, which is based on the average queue length. If the average is below some minimum threshold value, packets are queued up. If the average is above the maximum threshold, packets are discarded. If the average is between the thresholds, further calculations are performed to determine the probability of discarding the packet and further thresholding is applied to either drop or mark the packet.

First, TCP Reno was used with Drop-Tail queuing discipline against a constant CBR flow of half the bottleneck capacity. The results showed that Drop-Tail queuing was not fair to the TCP flow and ended up letting the CBR flow have more of the bandwidth than the TCP flow.

Next, TCP Reno was used with RED queuing disciplines at every node in the network topology against a constant CBR flow of half the bottleneck capacity. The results showed that RED queuing mechanism also failed to provide fairness in bandwidth utilization to the TCP flow.

TCP SACK was then used with Drop-Tail and RED queuing disciplines, as performed above against a constant CBR flow of half the bottleneck capacity. The results showed that the Drop-Tail achieved better fairness over RED and performed better in terms of bandwidth and packet loss. SACK showed a better performance with Drop-Tail queues than with RED because SACK creates higher packet loss rates in a Red environment than in a Drop-Tail environment, negating the low-delay benefit that RED is meant to achieve in the first place.

Since bandwidth utilization is not fair between the TCP flow and the CBR flow, there was no significant improvement in the latency as well as end-to-end delay is inversely proportional to bandwidth.

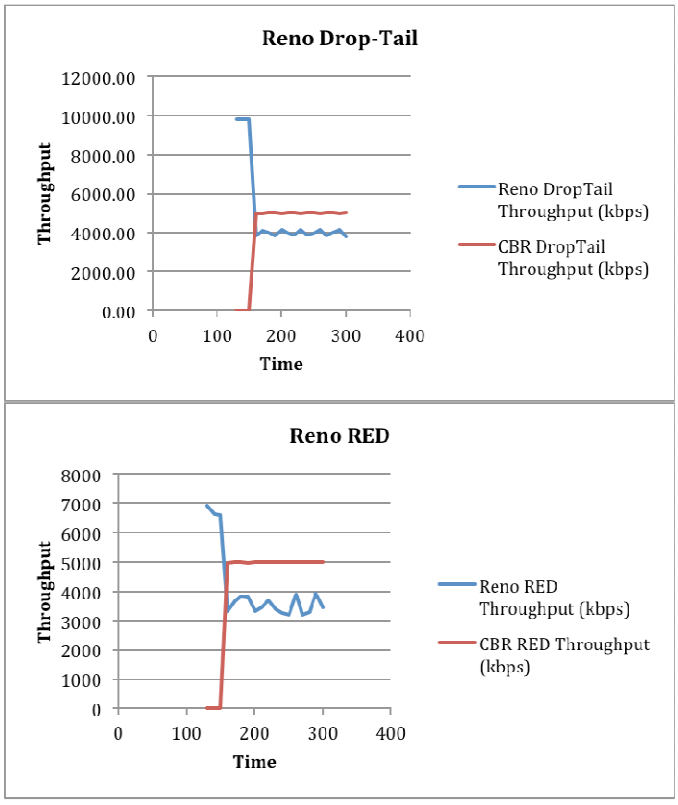


Figure9

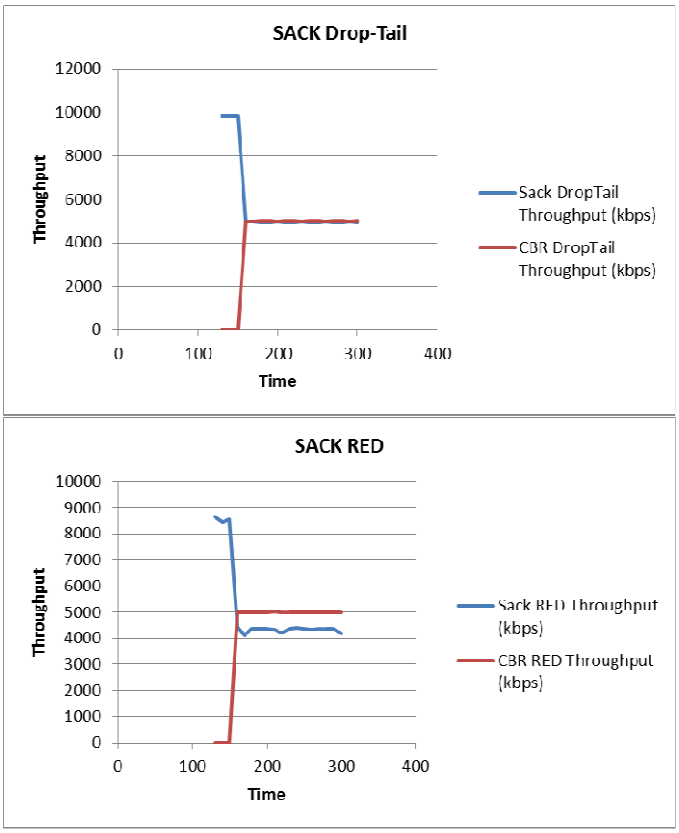


Figure10

VI. Conclusion

In the first experiment, TCP Vegas displayed a better throughput, lower latency and packet loss when compared to other TCP variants, even when the network congestion was increased by a CBR flow.

In the second experiment, we found that TCP New-Reno when combined with another flow in the network topology is unfair because of its aggressive increase of window size during fast-retransmit.

In the third experiment, we found that Drop-Tail queuing discipline performs better than RED when used with TCP Reno. TCP SACK though showed a better performance with Drop-Tail queues than with RED because SACK creates higher packet loss rates in a Red environment than in a Drop-Tail environment, negating the low-delay benefit that RED is meant to achieve in the first place.

So our experiments have revealed that designing a network topology is a complicated job as combining various types of TCP variants in the same network presents several challenges that we saw above. TCP Vegas though theoretically sounds like the best variant amongst all, cannot be deployed in a real world scenario where there could be multiple types of network protocols operating on the same topology.

We feel that, one of the possible solutions is to modify and improve the congestion control algorithm of TCP Vegas to make it equally competitive with the other TCP variants and to make it deployable on a large scale in the real world.

VII. References

[1] "The New-Reno Modification to TCP's Fast Recovery Algorithm" , S. Floyd, T. Henderson, A. Gurtov, IETF RFC 3782, April 2004

[2] S.Floyd, T.Henderson "The New- Reno Modification to TCP's Fast Recovery Algorithm" RFC 2582, Apr 1999.

[3] O. Ait-Hellal, E.Altman "Analysis of TCP-Reno and TCP-Vegas".

[4] K.Fall, S.Floyd "Simulation Based Comparison of Tahoe, Reno and SACK TCP" .

[5] Go Hasegwa Kenji, Kuruta and Masayuki Murata., "Analysis and Improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the internet".

[6] V.Jacobson. "Congestion Avoidance and Control". SIGCOMM , Communication Architecture and protocols.

[7] Jae Chung , Mark Claypool "Analysis of RED Family Active Queue Management Over a Wide-
Range of TCP Loads"