Performance Analysis of TCP Variants

Raghuveer Ramesh
Northeastern University
Email: ramesh.r@husky.neu.edu

Soumya Ramesh
Northeastern University
Email: ramesh.so@husky.neu.edu

***Abstract*—In this paper, we analyze performance and fairness of different variants of the Transmission Control Protocol (TCP) such as Tahoe, Reno, New-Reno, Vegas and SACK under the influence of congestion and different queuing disciplines like DropTail and Random Early Drop (RED). We perform simulation based experiments using the NS-2 network simulator and study performance attributes such as goodput, end-to-end latency, throughput and packet drop rate. We found that TCP Vegas outperformed the other variants in terms of the above performance metrics under varying load congestions. In terms of fairness, TCP Reno and New-Reno perform fairly with each other and other variants. But, TCP Vegas portrays unfair bandwidth usage when paired with itself or other TCP variants. While comparing RED and DropTail queueing mechanisms, we found that RED is more fair in terms of bandwidth allocation. Results of the simulated experiments and a comparative analysis of these TCP variants are reported.**

I. Introduction

The Transmission Control Protocol is a quintessential protocol in TCP/IP networks. The evolution of congestion control mechanisms began in the late eighties starting with Tahoe TCP[1]. Since then, many variants have been proposed and adapted over the years to achieve better congestion control. This paper aims at analyzing some of these TCP variants which are Tahoe, Reno, New-Reno, Vegas and SACK under the influence of congestion and queuing algorithms such as DropTail and RED. We perform 3 experiments using the NS-2 network simulator to generate a comparative analysis of performance metrics such as Goodput, Throughput, End-to-end Latency, and Packet Drop Rate for each variant. The definitions for these metrics are defined in the methodology section.

Tahoe TCP employs cumulative acknowledgments and a timeout based mechanism to retransmit packets. The recent versions of Tahoe TCP use algorithms such as Slow-Start and Fast Retransmit. Slow-Start varies the size of the congestion window in an exponential manner until a threshold is reached. After this threshold is reached, the window size is linearly increased to perform congestion control. In Fast Retransmit, when multiple duplicate acknowledgments for a packet are received, it is inferred that packet retransmission is required. Once a packet loss has been detected, the window size is reset and the Slow-Start process is repeated.

Reno TCP is a variation of Tahoe TCP in that the Fast Retransmit algorithm is modified to the Fast Recovery algorithm. When a certain number of duplicate ACKs (usually 3) are received, the lost packet is retransmitted and the size of the congestion window is reduced by half. The performance of Reno TCP is known to deteriorate when there are more than one packet losses within a round trip time since at most one packet retransmission occurs within the same window.

New-Reno TCP is a modification to Reno TCP in its better capability to handle multiple packet losses within the same window of data. New-Reno TCP continues to stay in the Fast Recovery phase until all outstanding packets have been acknowledged since the start of Fast Recovery, whereas in Reno TCP, a similar situation would result in exiting the Fast Recovery phase.

SACK TCP is an extension of Reno TCP where SACK blocks are used to remember acknowledgments of non-contiguous set of packets that have been received. This mechanism allows to efficiently handle multiple packet losses that occur within the same window and perform retransmissions accordingly.[2]

Vegas TCP is again a modification to Reno TCP. It estimates the difference between the expected rate and observed rate to perform congestion control to determine the optimal sending rate by correspondingly adjusting the congestion window to maintain a small number of packets along the transmission buffer. The algorithm anticipates congestion prior to the occurrence of packet loss. [3]

The paper contains the following sections - Methodology contains definitions for each performance metric and subsections for each experiment along with their results. This is followed by sections for conclusion and references.

II. Methodology

We perform 3 experiments using the NS-2 network simulator and parse the trace files to obtain meaningful results which are visually represented using graphs. The network topology is setup as shown in Fig 1. It consists of 6 nodes N1 - N6 where the bandwidth of each link is set to 10Mbps and the delay in each link is set to 10ms. Using this topology, we perform tests that analyze the goodput, throughput, packet drop rate, and latency of different TCP variants in the presence of a Constant Bit Rate (CBR) flow and different queuing algorithms such as DropTail and RED. We calculate the performance metrics as follows -

1) Throughput: - Throughput is the number of data bits received at the destination per time interval. It is represented in Megabits per second (Mbps).

2) Goodput: - Goodput is the number of useful data bits received at the destination per time interval. Useful data bits excludes ACKs and data bits received out of sequence. It is represented in Megabits per second (Mbps).

3) Packet Drop Rate: - Packet Drop rate is the (number of data packets sent - number of data packets received) / number of data packets sent.

4) End-to-end Latency: - End-to-end Latency for a packet is the time taken for a data packet to reach the destination after being enqueued at the source. It is represented in seconds.

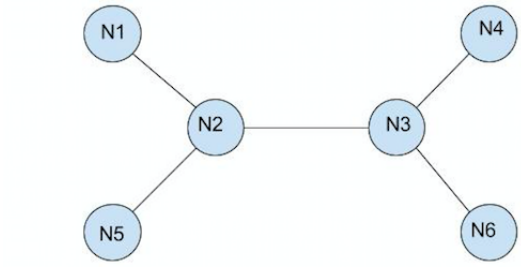

Fig. 1. Network topology

The experiments are described as follows:

A. Experiment 1 - TCP Performance under congestion

In this this experiment, the above mentioned network topology was used. A CBR source was added at Node-2 and a sink at Node-3 . A TCP stream source was added at Node-1 with a sink at Node-4. The simulation was run with different CBR flow rates starting at 0.1 Mbps to 10Mbps in increments of 0.1 Mbps.

We experimented by varying other factors that affect congestion such as the maximum congestion window size for a range of values from 20(default) to 100 while maintaining of a constant CBR of 2 Mbps. We found that the throughput kept increasing until a congestion window size of 75 and after that, throughput started to reduce. Hence we chose the maximum window size of 75 as the optimal maximum window size for the remainder of the experiment.

We also varied the start and end times for the TCP and CBR flows. We ran simulations for TCP and CBR flows starting at the same time and another set of simulations with TCP starting after CBR and CBR starting after TCP and averaged these values. Each of these experiments was done for TCP versions of Tahoe, Reno, New-Reno and Vegas.

B. Experiment 2 - Fairness between TCP variants

For this experiment , we start two TCP flows along with a CBR flow and analyze how each variant behaves with the other. We added a CBR source at N2 and a sink at N3 and added two TCP streams from N1 to N4 and N5 to N6, respectively. CBR rate is varied from 0.1 to 10 with increments of 0.1 Mbps. Each of these are further varied with different start times and the parameters averaged. To compare fairness between a pair of TCP variants, we compare the Goodput and latency of each variant and then determine if it is fair or unfair.

C. Experiment 3 - Influence of Queuing

In this experiment, we look at how DropTail and RED queuing techniques affect the performance of TCP SACK and TCP Reno. For this experiment TCP was started at the 1st second and after 9 seconds, CBR flow was started and both were stopped at the 25th second. CBR flow was fixed at 7Mbps and the Queue limit across the link Node-2 to Node-3 was set to 10.

III. Results

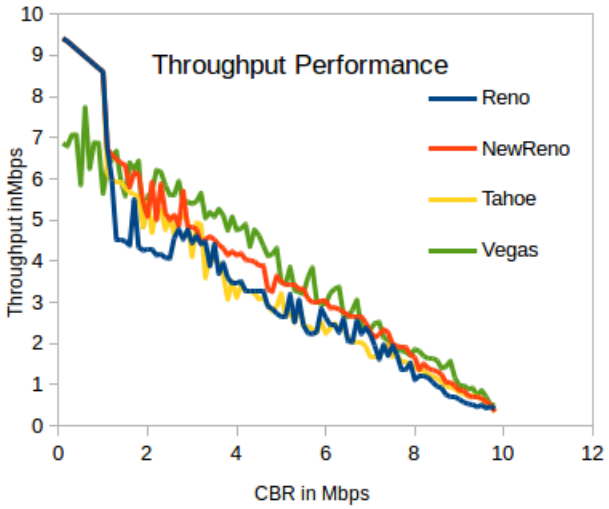

Fig. 2. Throughput for TCP Tahoe, Reno, New-Reno and Vegas

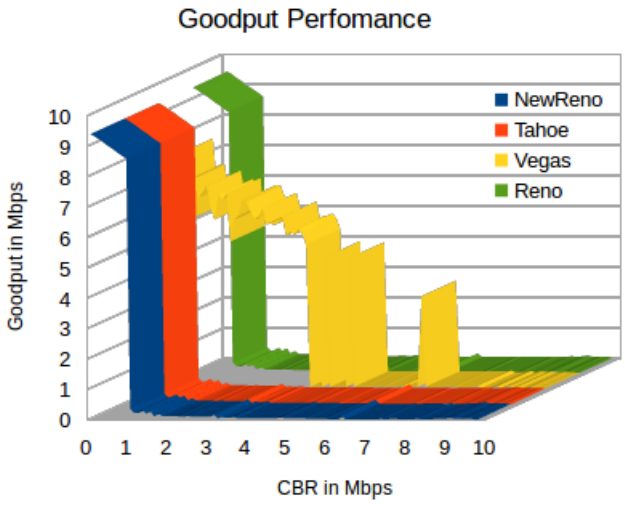

Fig. 3. Goodput for TCP Tahoe, Reno, New-Reno and Vegas

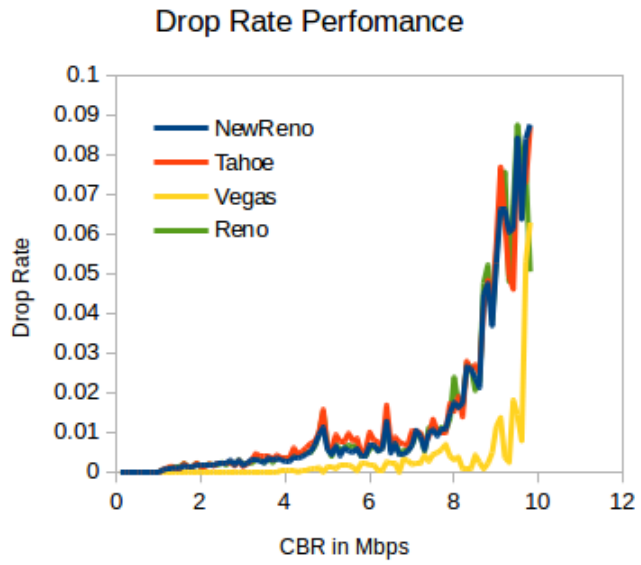

Fig. 4. Drop Rate for TCP Tahoe, Reno, New-Reno and Vegas

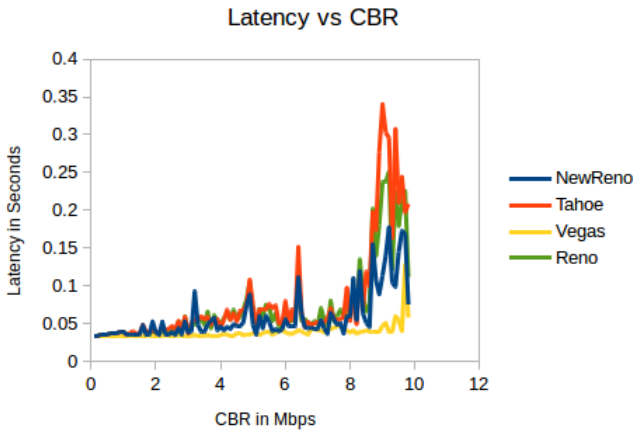

Fig. 5. Latency for TCP Tahoe, Reno, New-Reno and Vegas

A. Results for Experiment 1

Based on this experiment, we can answer the following questions-

1) Which TCP variant(s) are able to get higher average throughput?
2) Which has the lowest average latency?
3) Which has the fewest drops?
4) Is there an overall ”best” TCP variant in this experiment, or does the ”best” variant vary depending on other circumstances?

From Fig. 2, we see that as congestion continues to increase, TCP Vegas achieves the best throughput. From Fig. 3, when we consider goodput as a distinguishing parameter, it is evident that Vegas outperforms all the other variations once congestion is introduced in the network. From Fig.4 and Fig.5, we observe that TCP Vegas also has the least end-to-end latency and least packet drop rates. Hence, in this experiment we can say that TCP Vegas is the overall ”best” TCP variant.With no congestion in the network, we notice that that only TCP Reno has higher throughput compared to the other variants. However once congestion is introduced, we see that the throughput begins to reduce and forms a zig-zag pattern. This pattern is due the congestion window that keeps changing .

From Fig.4, New-Reno is more suited to handle multiple packet drops and this is evident from the graph, especially when congestion increases. New-Reno performs better in terms of higher throughput, lower latency and lower drop rates in comparison with Reno.

B. Results for Experiment 2

Based on this experiment, we can answer the following question -

- Are the different combinations of variants fair to each other?
- Are there combinations that are unfair, and if so, why is the combination unfair?

From Fig.6, when TCP Reno is paired with another TCP Reno flow, we observe similar bandwidth consumption and latency which leads to the conclusion that Reno is fair when paired with itself.

From Fig.12 and Fig. 13, when TCP New-Reno is paired with Reno, TCP New-Reno uses up more bandwidth and has lesser latency. This is due to New-Renos ability to deal with multiple packet drops. TCP New-Reno is unfair to TCP Reno.

From Fig.10, when TCP Vegas is paired with New-Reno we observe that Vegas performs much better than New-Reno due to Vegas accurate clock timing techniques to determine and monitor the RTT for packets. When Vegas is paired with itself, one variation of Vegas shows better throughput and lower latency compared to the other variant which leads to the conclusion that Vegas is unfair to itself as well as TCP New-Reno.

From Fig.13 and Fig.10, the combinations of Reno/New-Reno and New-Reno/Vegas are unfair. With Reno/New-Reno, we know that New-Reno is an improvement over Reno in that it is designed to perform better in multiple packet loss scenarios which is validated by our experiments. Similarly, with New-Reno/Vegas, Vegas uses accurate clock timers to calculate the RTT to further improve packet loss by using an improvised congestion avoidance mechanism, and an improvised Slow-Start mechanism.

C. Results for Experiment 3

Based on this experiment, we answer the following questions -

1) Does each queuing discipline provide fair bandwidth to each flow?

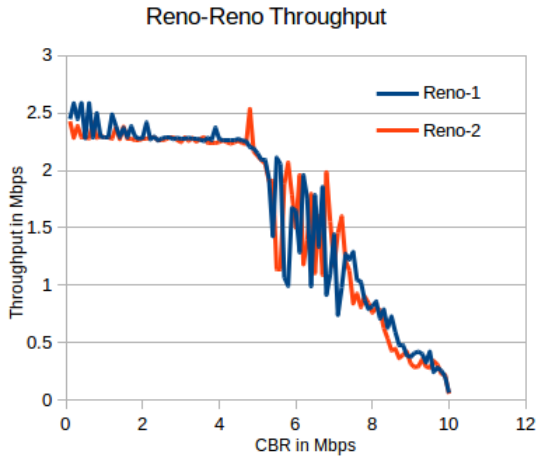

Fig. 6. Throughput for TCP Reno Vs Reno

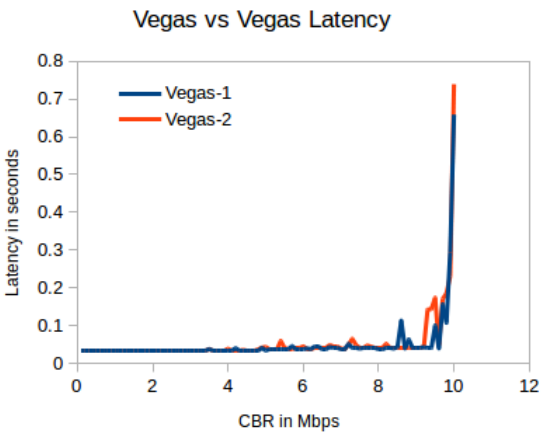

Fig. 9. Latency for TCP Vegas Vs Vegas

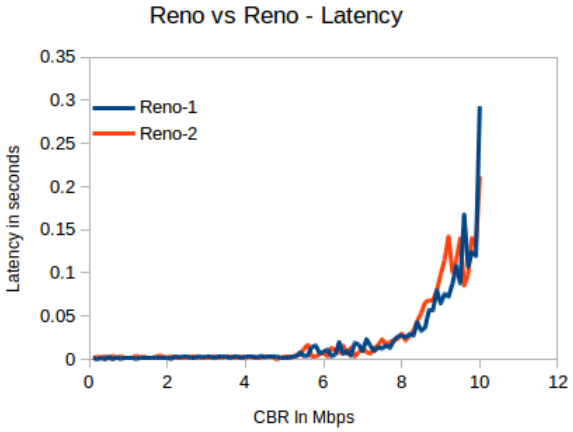

Fig. 7. Latency for TCP Reno Vs Reno.

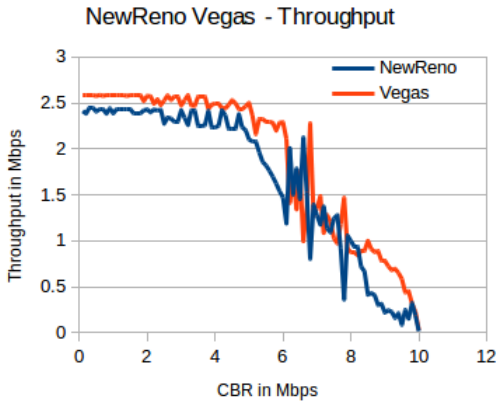

Fig. 10. Throughput for TCP New-Reno Vs Vegas

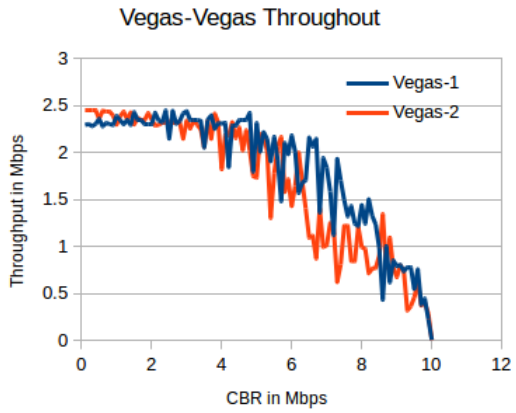

Fig. 8. Throughput for TCP Vegas Vs Vegas

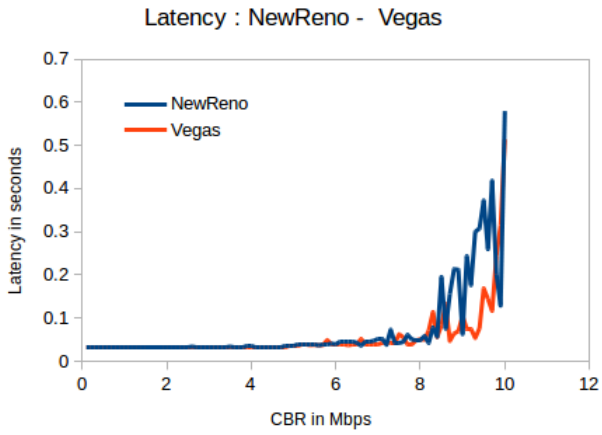

Fig. 11. Latency for TCP New-Reno Vs Vegas

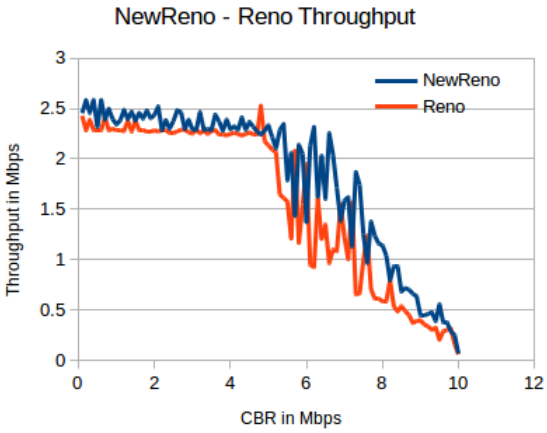

Fig. 12. Throughput for TCP New-Reno Vs Reno

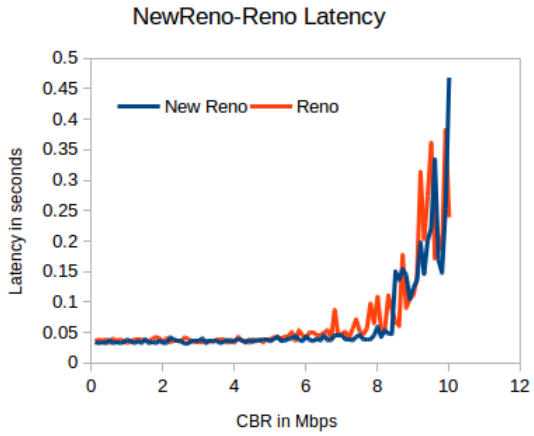

Fig. 13. Latency for TCP New-Reno Vs Reno

2) How does the end-to-end latency for the flows differ between DropTail and RED?
3) How does the TCP flow react to the creation of the CBR flow?
4) Is RED a good idea while dealing with SACK?

Congestion control algorithms such as RED and DropTail are implemented at the gateway layer. DropTail is a queuing algorithm that starts dropping incoming packets, once the queue size has reached its limit and will accept packets once there is room to accommodate packets in its buffer. RED monitors average queue size and works out probabilities to decide whether or not to drop incoming packets. From Fig. 16 and Fig. 17, we can see that when there is no congestion, both RED and DropTail provide fair bandwidth. However, as soon as congestion is introduced, RED seems to have lesser and therefore more fair bandwidth usage. This is due to REDs congestion control mechanism that starts dropping packets as soon as it realizes the Queue is about to fill up thereby that makes it fair to other competing flows . The same cannot be said about DropTail.

By looking at the graph in Fig. 15, we can conclude that the average end-to-end latency for RED is lesser compared to DropTail. This is because RED keeps an eye on the queue size using probability calculations to keep a check on traffic. While comparing TCP and CBR, we observe that TCP flow reaches a stable rate and remains stable until CBR is introduced. In Fig. 14 it can be seen that TCPs throughput dips in congestion and starts increasing again to reach its stable state.

From Fig. 17 we can see that the throughput of SACK reduces when used with RED as compared to throughput of SACK with DropTail from Fig. 16 . Therefore we can conclude that SACK with RED is not an ideal combination.

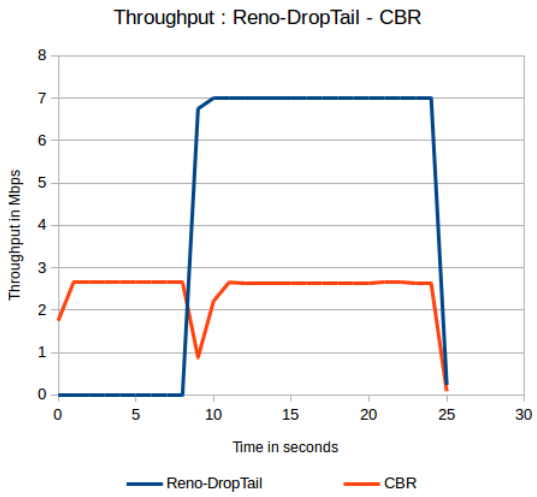

Fig. 14. Throughput comparison for TCP Reno and CBR with DropTail

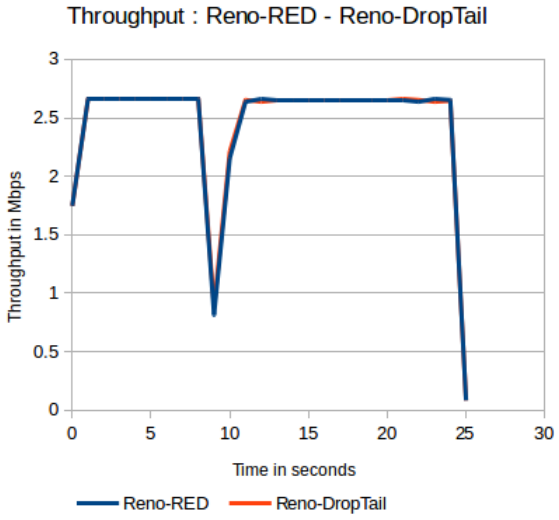

Fig. 15. Throughput comparison for TCP Reno using RED and DropTail

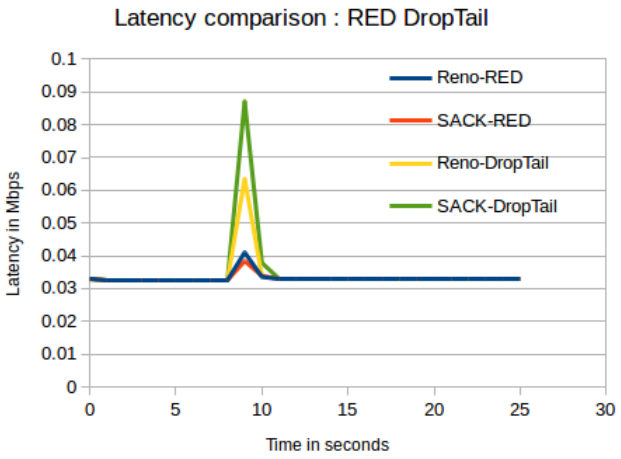

Fig. 16. Latency of TCP SACK and Reno using RED and DropTail

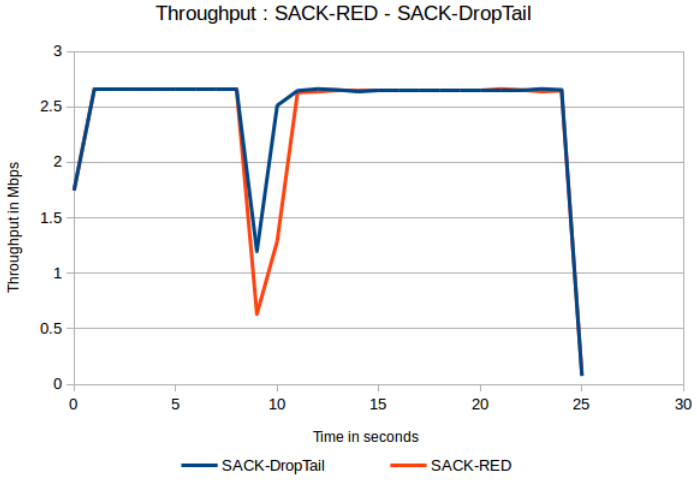

Fig. 17. Throughput comparison for TCP SACK using RED and DropTail

IV. Conclusion

This paper reports the behavior of TCP variants under the influence of congestion and different queuing mechanisms through simulated experiments. Experiment 1 analyzes the behavior of TCP variants under the influence of congestion. From the results of experiment 1, we can conclude that when there is constant congestion in the network, amongst different TCP variants, TCP Vegas performs better than TCP Tahoe, Reno and New-Reno in terms of throughput and latency . The difference is all the more evident when goodput is considered. Experiment 2 analyzes the fair utilization of resources amongst the TCP variants. From the results of Experiment 2, can conclude that TCP Reno seems fair when paired with itself . When TCP New-Reno was paired with TCP Reno , TCP New-Reno seems to use up more bandwidth than Reno on average and is unfair on some level. TCP Vegas on the other hand seems significantly unfair when paired with itself and TCP New-Reno. Experiment 3 analyzes the influence of queuing mechanisms such as DropTail and RED on TCP SACK and Reno. From experiment 3, we can conclude that RED and Droptail have similar throughputs while using Reno, but SACK performs better with DropTail than with RED and also the end-to-end latency for Drop tail is lower than RED. When CBR flow is introduced, TCPs throughput and latency suffers but recovers fairly quickly.

References

[1] V. Jacobson , Congestion avoidance and control in Proceedings of ACM SIGCOMM '88 (Stanford CA, August 1988), 314-329.

[2] F. Kevin and F. Sally , Simulation-based Comparisons of Tahoe, Reno, and SACK TCP in ACM SIGCOMM Computer Communication Review July 1996 , 26-3.

[3] L. Steven and P. Larry and W. Limin , Understanding TCP Vegas: Theory and PracticeP 2000.