Performance Analysis of TCP variants and Queueing Algorithms

Devendra Sharma
Telecommunication System Management
College of Engineering, Northeastern University
sharma.dev@husky.neu.edu

Rachit Puri
Computer Science
College of Engineering, Northeastern University
puri.rac@husky.neu.edu

Abstract

This paper analyses the performance of different TCP variants like Reno, Tahoe, New Reno, Vegas and Cubic under the influence of different load conditions by using simulations in ns2. We also analyze the two different queueing algorithms viz. Random early detection (RED) and Drop-tail and their influence on the performance on TCP Reno and TCP SACK. The results suggest that in terms of throughput performance and Bandwidth utilization TCP CUBIC outperforms all the other abovementioned TCP variants, whereas the performance of TCP Vegas is better than the others as the congestion in the network increases. Also, the simulations show that the RED queuing algorithm has a better performance when used with TCP SACK compared to Drop-Tail.

1. INTRODUCTION

The Transmission control protocol (TCP) is one of the most vital protocols of the transport layer in the Internet Protocol suite. The very characteristics of TCP such as reliability, error-free and ordered delivery of packets make it necessary for the application layer protocols like HTTP, FTP, SMTP and other to use it as the underlying protocol. The evolution of TCP since the inception of the internet has led to the development and use of different TCP variants Reno, Tahoe, Vegas, New Reno, CUBIC, SACK and others which were aimed to solve the problems arising in today's increasingly congested networks.

The analysis of different variants of TCP in this paper helps us to understand the basic concept and the purpose of each and every variant. TCP Tahoe is one of the simplest variant without any fast recovery. TCP Reno differs from Tahoe in terms of action taken after triple duplicate acknowledgments and its fast recovery technique. In TCP New Reno only a new data ACK does not take TCP out of fast recovery to congestion avoidance since it demands all the packets that are unacknowledged at the start of the fast recovery period are acknowledged. In TCP SACK the sender maintains the data of which packets where not received by the receiver and only resends those packets. TCP Vegas focusses more on packet delay rather packet loss. All these variants are different in their approach of dealing with congestion. The significance of the analysis of these variants lies in the fruitful application of these variants in different scenarios depending upon the requirement of the service and the degree of congestion in the network. It will be much better to have a prior information and knowledge of these TCP variants before using any of them randomly and reducing the throughput of our application.

2. METHODOLOGY

Network Simulator-2 (ns-2) was used to simulate the network topology which uses scripts written in object oriented Tcl and extension of Tcl scripting language. These scripts specify the network topology, characteristics of each node and link between them. The topology that was used is shown by Fig.2.1. In the scripts we had to assign bandwidth of each link, CBR rate and the Queueing Algorithm at the nodes. We also scripted the type of TCP variant source and its sink at their respective position in the topology. The TCP source and sink were positioned at node N0 and node N4 respectively. Similarly the CBR source and sink were kept at N2 and N3. The bandwidth of each link was set to 10Mbps, 10Mb queue limit and 10ms delay. For the experiments 1 and 2 the queueing algorithm was set to Drop-tail and for experiment 3 it was varied with RED in order to compare and test it with TCP SACK and Reno. Also for experiment 2 another TCP source were added at node N1 and N5 whereas for experiment 3 the CBR source and sink were repositioned to node N1 and N5.

The CBR rate was varied from 1Mbps to the bottleneck capacity of the link i.e. 10Mbps for all the three experiments. The start time and end time of each flow were specified differently in the Tcl files for all the three experiments After the Tcl files were executed using ns-2, the tracer files which were generated as the output of each experiment was parsed using a separate python script. The execution of Tcl file used command line arguments from the user to specify the TCP variant and the CBR rate. The python script read the tracer output files and calculated the Throughput, Number of dropped packets and Latency. The tracer files contains the values of number of dropped packets, the start and end time of the simulation, number of bytes received by the destination and number of acknowledgments received by the sender. These values were then used to plot graphs and showcase the behavior of different TCP variants. Plotting of graphs were done using Excel. NAM was used to verify the correctness of the topology and positioning of TCP and CBR flows.

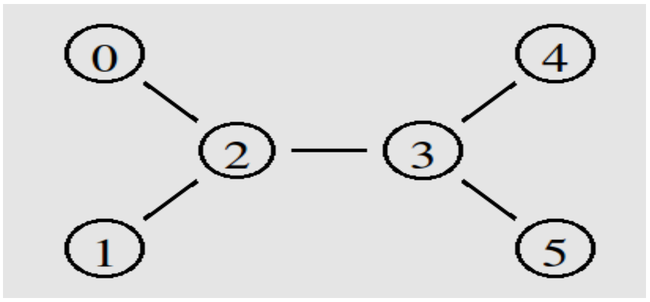

Figure 2.1 Network Topology

3. TCP Performance under congestion

In the first experiment a TCP source and sink were kept at nodes N0-N4 and a CBR source and sink were kept at N2-N3. The CBR rate was increased in steps of 1Mbps from 1Mbps to 10Mbps. By supplying a different TCP variant input from command line argument the above mentioned process of varying the CBR rate was carried for each TCP variant viz. Tahoe, Reno, New Reno, Vegas and CUBIC thereby analyzing these variants under different loading conditions of congestion. Throughput, Packet drop rate and Latency for each of these TCP streams was plotted against the increasing CBR rate.

After the analysis of individual throughput. Latency and packet drop rate of each TCP variant we came to the conclusion that there is no overall best TCP variant. Depending upon different circumstances we find a particular TCP variant is better than other for that particular condition.

3.1 Throughput of different TCP variants.

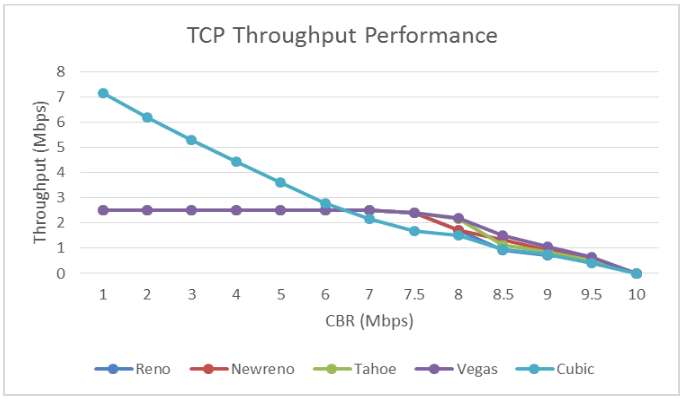

Figure 3.1 Throughput of different TCP variants

Figure 3.1 shows the throughput of Tahoe, Reno, New Reno, Vegas and CUBIC in the presence of a CBR flow at nodes N2-N3. When the CBR rate is less than 5 Mbps the all the TCP flows have a constant throughput. CUBIC has the highest throughput among them and Vegas has the lowest, but as the CBR rate increases after 5 Mbps, the throughput of all the flows decreases and becomes almost

Zero when the CBR rate is 10 Mbps. After analyzing the trace files we get to know that after CBR rate increases above 5 Mbps, Vegas continues to have a better performance than the other and has a lesser slope in throughput. The reason for CUBIC having a higher output when the congestion is less is the Cubic function that it uses to increment the window if there is no congestion and also it does not rely on the acknowledgment receipts but on the last congestion event.

The reason for TCP Vegas having a better throughput when the network is severely congested is its bandwidth detection mechanism which is based on RTT. Even in a congested network it will probe for and compete for the available bandwidth to maintain a certain amount of throughput.

3.2 Packet Drop Rate of different TCP variants

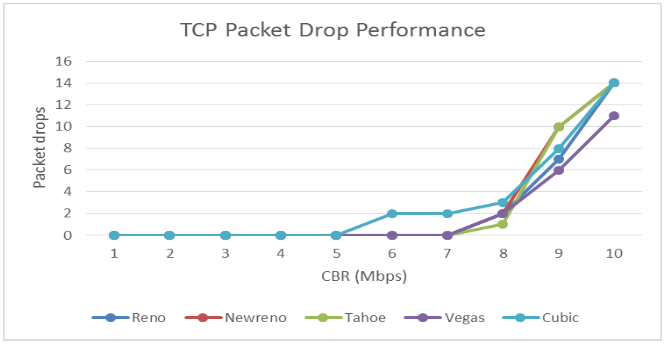

Figure 3.2 Packet drop performance of different TCP variants

Figure 3.2 shows the relationship between Packet drop rate and the CBR rate. Results show that below CBR rate of 5 Mbps there are no packet drops for all the variants. As the CBR rate increases above that value packet drop rate increases. CUBIC reacts quickest to the increasing CBR rate in terms of packet drops and demonstrates the highest number of drops while Vegas continues to have the least number of packet drops and the packets are dropped only above 7 Mbps of CBR rate. Tahoe, Reno and New Reno show similar behavior but are outperformed by Vegas. The reason for this might be the bandwidth estimation mechanism of TCP Vegas which is different from others. Vegas constantly monitors the RTT and tries to maintain at least α number of packets but not more than β in its queue. It ties to detect and utilize the extra bandwidth whenever it is possible without causing congestion in the network. The default setting of α and β are 1 and 3. TCP Vegas can be made more competitive for bandwidth by changing these values.

3.3 Average Latency of Different TCP variants.

Figure 3.3 shows the relationship between average latency of TCP variants versus the CBR flow rate. Almost all the variants except Vegas have the similar average latency. Vegas has a higher average latency than others. CUBIC’s latency exceeds than Vegas the CBR rate approaches the bottleneck link capacity i.e. above 8 Mbps. Reno and New Reno exhibit the least average latency when the trace file output were closely analyzed.

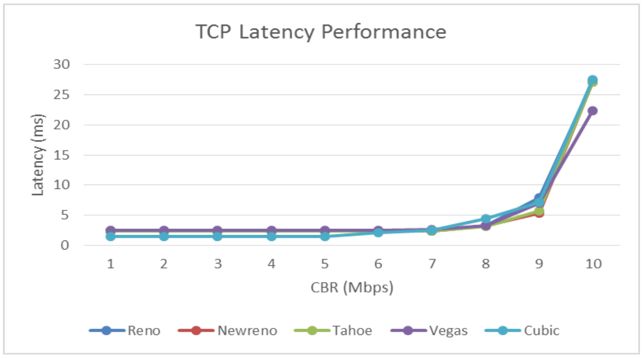

Figure 3.3 Average Latencies of different TCP variants

4. Fairness between TCP variants

In the second experiment we analyze the fairness between two TCP variants. Two TCP flows are set up at nodes N0-N4 and N1-N5, also a CBR flow is set up at N2-N3. The pairs of TCP variants were set up in following combinations.

(a) Reno/Reno
(b) New Reno/Reno
(c) Vegas/Vegas
(d) New Reno/Vegas

We then analyze the throughput, average latency and packet drop rate of all the TCP flow pairs against the increasing CBR rate.

4.1 Throughput Fairness

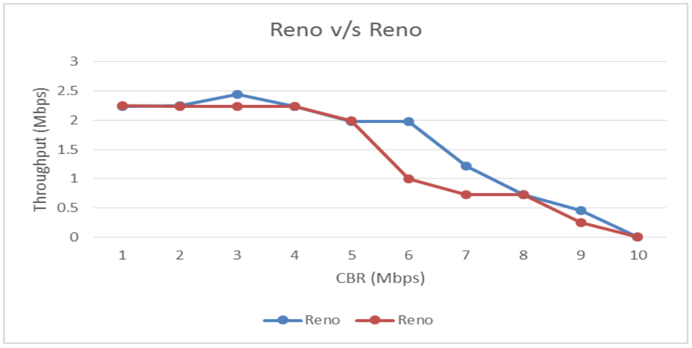

Figure 4.1.1 Throughputs of Reno and Reno

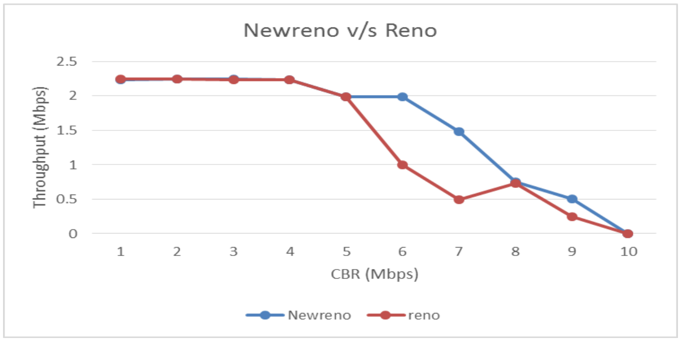

Figure 4.1.2 Throughput of New Reno and Reno

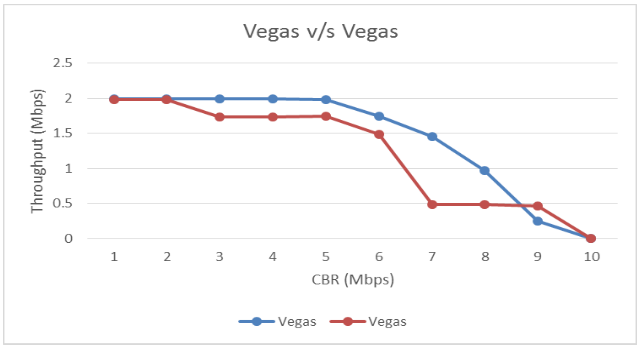

Figure 4.1.3 Throughputs of Vegas and Vegas

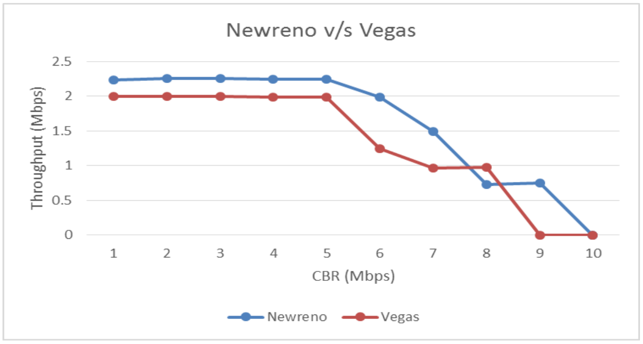

Figure 4.1.4 Throughputs of New Reno and Vegas

Figure 4.1.1-4.1.4 show the results of different pairs of TCP variants against the increasing CBR rate. After the analysis of these plots we understand that as the CBR rate increases, throughput of the two TCP links decreases and both the streams start competing with each other for maximum bandwidth. In terms of throughput fairness except Vegas-Vegas pair, the other three combination provide enough fairness to each other's flows. In Vegas-Vegas, one flow gets a higher throughput while the other flow gets a very low throughput in spite of both streams starting at same time. The reason for this behavior might be the bandwidth detection mechanism used by Vegas in which one random flow sees a higher throughput and the other doesn't see it. This continues till the available bandwidth becomes zero for both the flows.

4.2 Latency Fairness

Figures 4.2.1-4.2.2 show the relationship between average latencies of the TCP flow pairs versus the increasing CBR rate. The latencies increase drastically to 300 milliseconds from 5-10 milliseconds only when the CBR rate approaches the bottleneck link capacity, otherwise the latencies of both the flows are significantly small. The results for Reno-Reno and Vegas-Vegas pair are also similar to NewReno-Reno and NewReno-Vegas pair

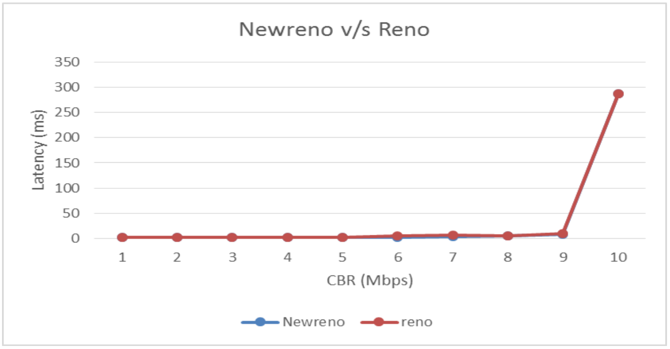

Figure 4.2.1 Average latencies of New Reno and Reno

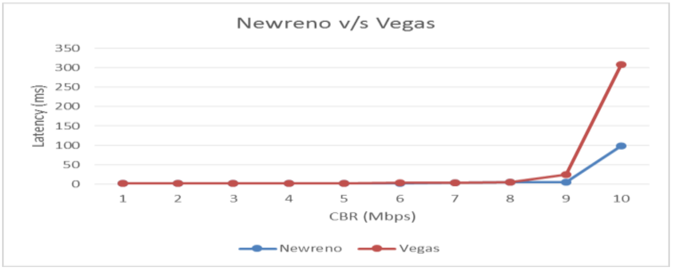

Figure 4.2.2 Average latencies of New Reno and Vegas

4.3 Packet Drop Rate Fairness

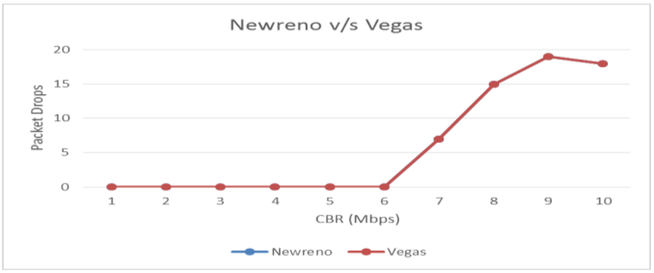

Figure 4.3 Packet Drop Rate of New Reno and Vegas

Figure 4.3 shows the relationship of Packet drop rates of New Reno and Vegas pair against the increasing CBR rate. As the CBR rate increases, the congestion in the network increases causing the increase in the packet drops for both flows. As the CBR rate approaches 10Mbps, both the TCP flows undergo a slight decrease in the number of packet drops as they sense congestion and reduce their window size. Both the flows had undergone equal number of packet drops in this experiment. The other three pairs also show the same result in which both the flows have equal number of packet drops. Hence we can deduct from this analysis that all the TCP variant pairs in this experiment show significant amount of fairness in terms of packet loss.

5. Influence of Queueing

In the third experiment, we compare two different queueing disciplines i.e. Random early detection (RED) and Drop-Tail with two different TCP variants. Reno and TCP SACK flows are varied at nodes N0-N4 and CBR flow is set up at N1-N5. Initially we allow the TCP flow to stabilize for 4 seconds and introduce the CBR flow of constant bandwidth after TCP flow stabilizes. We then plot the throughput of each combination i.e. Reno with RED, Reno with Drop-Tail, SACK with RED and SACK with Drop-Tail against time.

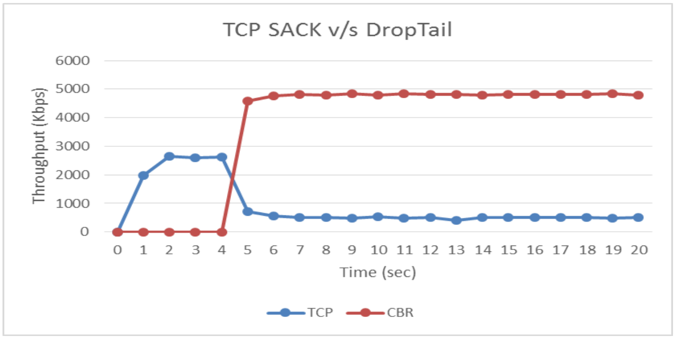

Figure 5.1 Throughput of TCP SACK with Drop-Tail

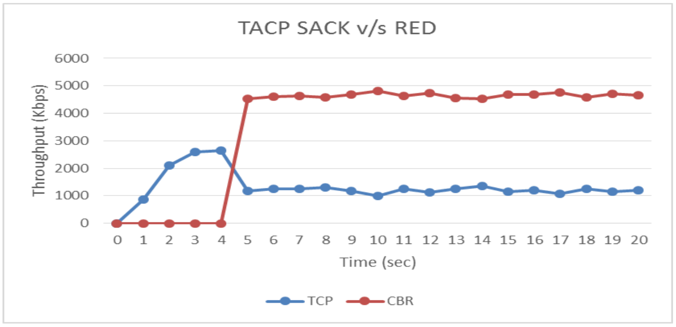

Figure 5.2 Throughput of TCP SACK with RED

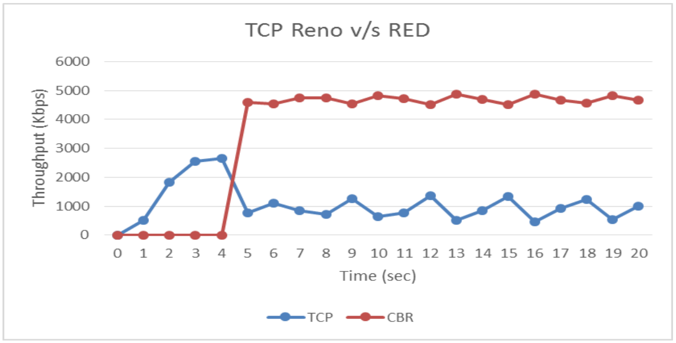

Figure 5.3 Throughput of TCP Reno with RED

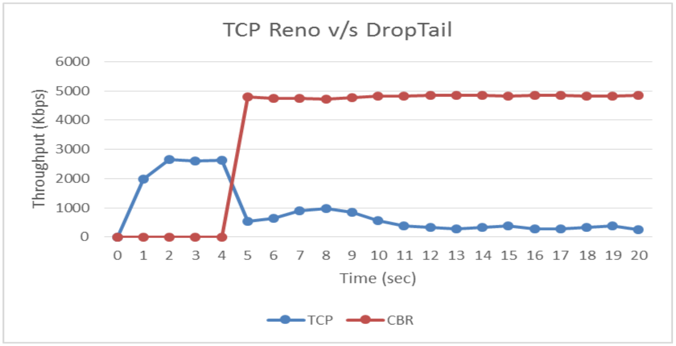

Figure 5.4 Throughput of TCP Reno with Drop-Tail

From figure5.1-5.4, we can conclude that before the CBR flow starts SACK with both RED and Drop-Tail has a higher throughput than Reno with RED and Drop-Tail. As the network congestion increase after the introduction of CBR flow SACK with RED has the highest throughput performance than all the other flows and Reno with Drop-Tail has the least throughput among all the other combination. Hence we understand that each queueing disciplines do not provide fair bandwidth to all the flows, RED outperforms Drop-Tail in this scenario. The reason for this behavior might be the intrinsic drawbacks of Drop-Tail algorithm which does not differentiates the traffic from different flows and does not distribute buffer space fairly whereas RED's algorithm avoids bias against bursty traffic and has an upper bound on average queue size even when the Transport layer protocols do not decide to co-operate**.**

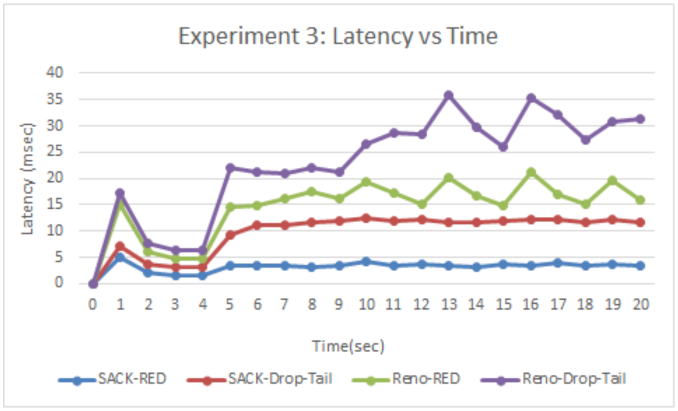

Figure 5.5 Latencies of TCP SACK and Reno with RED and Drop-Tail

From figure 5.5, we can conclude that after the CBR stream is introduced the latency experienced by all the flows increases. SACK performs better than Reno when tried with both either RED or Drop-Tail because of its particular acknowledgment mechanism. RED proves to a better queueing algorithm than Drop-Tail due to its nature of providing connection avoidance by controlling the average queue size. SACK with RED has the lowest latency whereas Reno with Drop-Tail has the highest latency.

By looking the results of throughput and latency for the four different combinations, we can conclude that SACK is best suited to use with RED and the use of Reno with Drop-Tail should be avoided.

6. Conclusion

In the above three experiments we analyzed the performance of TCP variants like Reno, Tahoe, New Reno, Vegas and CUBIC in terms of throughput, average latencies and packet drop rate under different loading conditions provided by CBR flow. We have also demonstrated the effect of deploying different queuing disciplines like Random early detection (RED) and Drop-Tail when they are used with TCP variants like Reno and SACK.

From section 3, we concluded that TCP Vegas performs better than other Tahoe, Reno and New Reno in terms of bandwidth utilization and packet drop rate when the network becomes heavily congested but TCP CUBIC has higher throughput when the congestion is negligible.

From section 4, we concluded that when two TCP flows are deployed in the same network, in most cases both of them allow a fair performance to each other except when two Vegas flows are deployed, one Vegas flow gets a higher throughput while the other gets a lower throughput when the network gets congested.

From section 5, we concluded different queueing algorithms strongly impact the throughput and latency of TCP flows. Drop-Tail queueing algorithm shows similar performance when used with both TCP Reno and SACK. It is important to note that when SACK is used with RED its performance is better than other three combinations and Reno when used with Drop-Tail performs the worst. Also, RED performs better than Drop-Tail when the network congestion increases.

7. References

[1] "A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas"Berkeley

[2] "Simulation-based Comparisons of Tahoe, Reno, and SACK TCP". Kevin Fall and Sally Floyd

[3] Jae Chung ,Mark claypool "Analysis of REDFamily Active Queue Management Over a WideRange of TCP Loads