Performance Analysis of TCP Variants

Kexi Long
School of Computer and Information Science,
Northeastern University, MA
long.k@husky.neu.edu
NUID 001729512

Jian Li
School of Computer and Information Science,
Northeastern University, MA
jli@ccs.neu.edu
NUID 001731714

I. Introduction

A big issue when applying Transmission Control Protocol (TCP) is congestion in networks. Several TCP variants, (such as TCP Tahoe, Reno, New Reno, Vegas and others), have been proposed with different mechanisms to control and avoid congestion. In this paper, we perform three experiments in NS-2 Simulator to analysis behavior of TCP variants under various load conditions and queuing algorithms.

In first experiment, we compare the performance of four TCP variants when a Constant Bit Rate flow is set in the networks. The second experiment is about comparing the allocation of bandwidth when four pairs of TCP variants are applied. The third experiment is about comparing the performance of two different queuing disciplines: Drop Tail and Random Early Drop. The methodology details and conclusions of each experiment are discussed in following sections.

II. Methodology

A. Topology

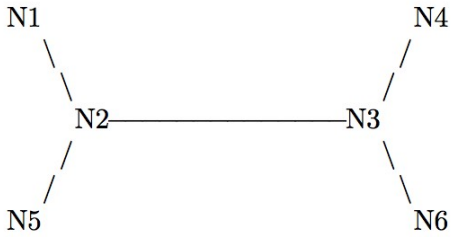

Figure 2.1 Topology

Networks topology for our three experiments is shown in the figure above. There are six Nodes (1-6), and the connectivity situation between them is clearly shown above. The bandwidth of each link in Firgure 2.1 is configured to **10Mbps** with a delay of **10ms**.

B. Simulator

All experiments are conducted in NS-2, an event driven network simulator, consisting several network components and event scheduler.

C. Evaluate Performance

Three parameters, drop rate, latency, throughput are taken into consideration when evaluating performance here. Drop rate is the percentage of dropped packets in total packets that have been sent. Latency in our experiments is measured as one-way, which is the time from the source sending a packet to the destination receiving it. Throughput is measured in Mbits per second (Mbit/s or Mbps), i.e. total data packets size per second.

The formulas for calculating them are as follows:

$$latency = \frac{\sum_{packets\ received} RTT}{totoal\ packets\ received}$$

$$throughput = \frac{total\ packets\ received * avg\ packet\ \ size}{time}$$

$$drop\ rate = \frac{dropped\ packets \times 100\%}{total\ packets\ sent}$$

D. Experiment Configurations

The queuing discipline of each link in experiment 1 and experiment 2 is DropTail with default queue size.

The window size of TCP streams in all the experiments are set as 20, which is the default size in NS2.

In experiment 1, we add a Constant Bit Rate (CBR) from N2 to N3, then add a TCP stream from N1 to a sink at N4; we start both traffic sources at 0; finally, we evaluated performance of Tahoe, Reno, New Reno and Vegas under congestion by varying the CBR rate from 1 to 10Mbps.

In experiment 2, we set a CBR from N2 to N3 and TCP stream from N1 to N4. The only difference is we add another TCP stream from N5 to N6. We conduct experiments with the fours TCP variants pairs - Reno/Reno, NewReno/Reno, Vegas/Vegas, NewReno/Vegas - with one flow from N1 to N4, and the other flow from N5 to N6. As in experiment 1, we plot the average throughput, packet loss rate and latency of each TCP flow as a function of the bandwidth used by the CBR rate.

In experiment 3, we use the same topology from experiment 1. We have one TCP flow from N1 to N4 and one CBR flow from N5 to N6. We compare the performance of two different queue algorithms - Drop Tail and RED - with two TCP variants Reno and SACK under congestion.

We first start the TCP stream, and then start the CBR after the TCP stream is steady.

From the previous experiments, we know congestion rises when CBR's bandwidth is 7Mbps and these two TCP variants need 3 seconds to stablize, so we start the CBR of 7Mbps at 3 and monitor the performance of TCP stream over a time period of 20 seconds.

III. Experiment 1 Result Analysis

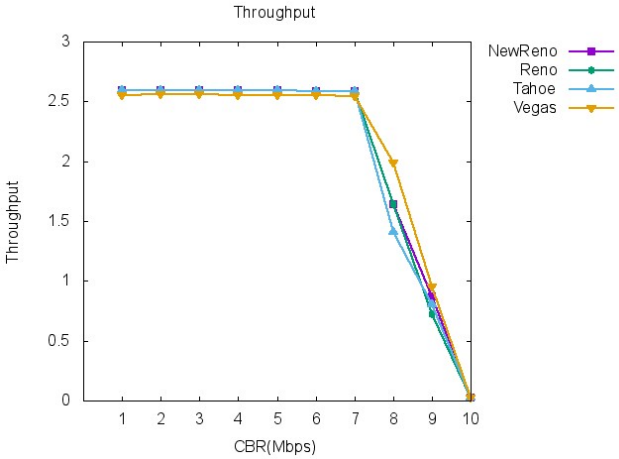

Figure3.1.1 Throughput

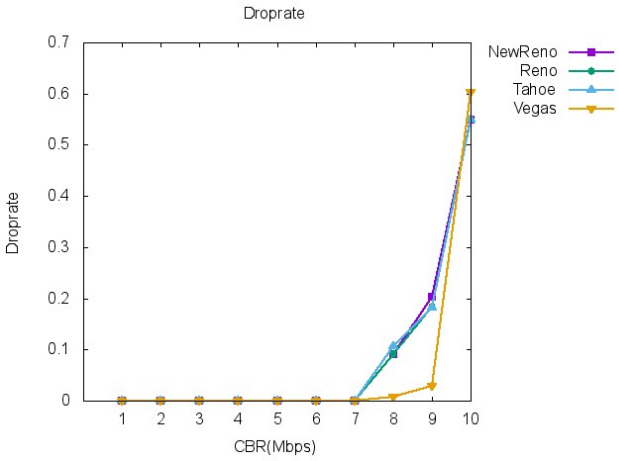

Figure3.1.2 Drop rate

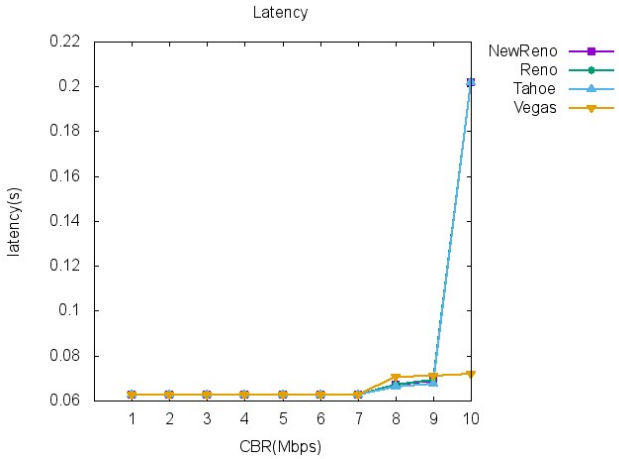

Figure3.1.3 Latency

A. *Throughput*

As shown in figure 3.1.1, the throughputs of all variants are around 2.5Mbps before the bandwidth of CBR reaches 7Mbps. And we can conclude that Tahoe and NewReno have higher throughput without congestion. When the bandwidth of CBR is 7Mbps, congestion rises and the throughput knee point is around here. As we can see, all the throughputs of the TCP variants drop drastically. But Vegas performs better than other variants under congestion.

B. *Drop rate*

The drop rate of Tahoe is very sensitive to congestion, as it increases earlier than other variants as shown in figure 3.1.2. While Vegas maintains the lowest drop rate under heavy congestion. NewReno and Reno behave almost the same and their drop rates are between Vegas and Tahoe.

C. *Latency*

All the variants have the same RTT without congestion. But Vegas's latency remains fairly low while other variants increase drastically under congestion.

IV. Experiment 2 Result Analysis

1) *NewReno vs Reno*

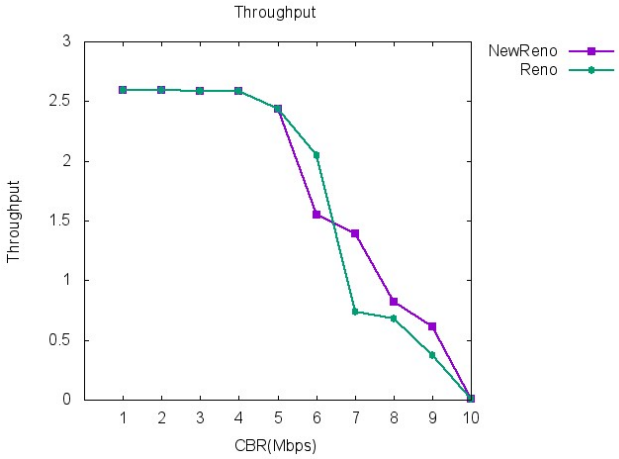

Figure4.1.1 Throughput

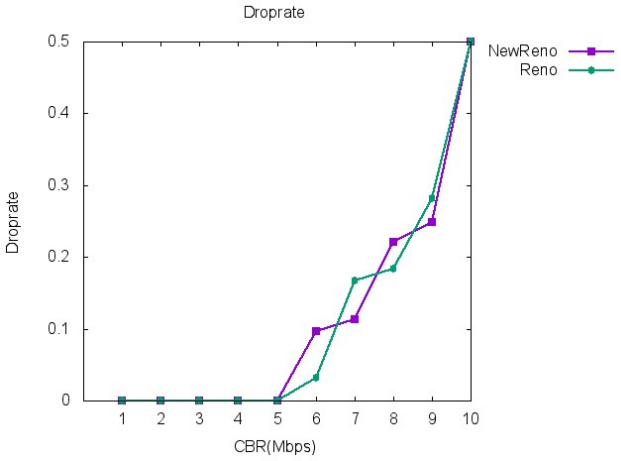

Figure4.1.2 Drop rate

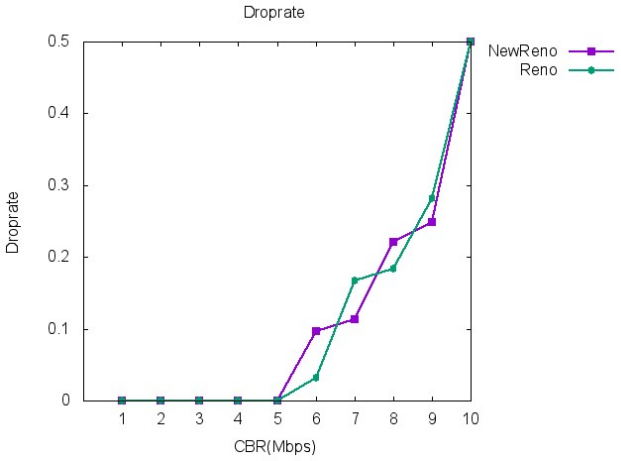

Figure4.1.3 Latency

A. Throughput

As shown in figure 4.1.1, we can conclude that NewReno and Reno can both get a fair throughput under congestion when competing to each other.

B. Drop rate

As shown in figure 4.1.2, the drop rate of these 2 variants oscillate on and forth as the congestion becomes heavier. Thus maintain fair to each other.

C. Latency

As shown in figure 4.1.3, the latency of these 2 variants oscillate on and forth as the congestion becomes heavier. Thus they are fair to each other.

2) NewReno vs Vegas

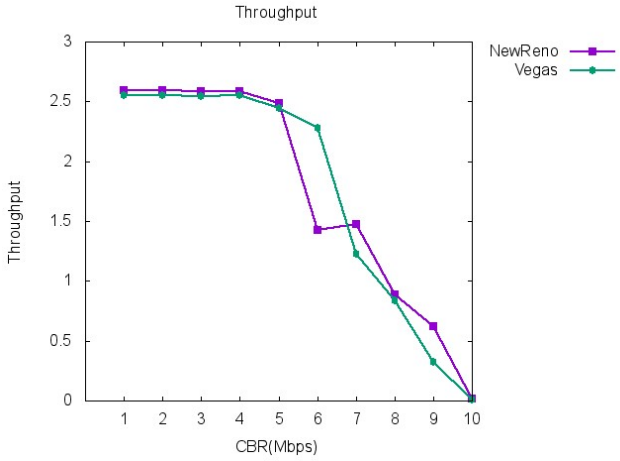

Figure4.2.1 Throughput

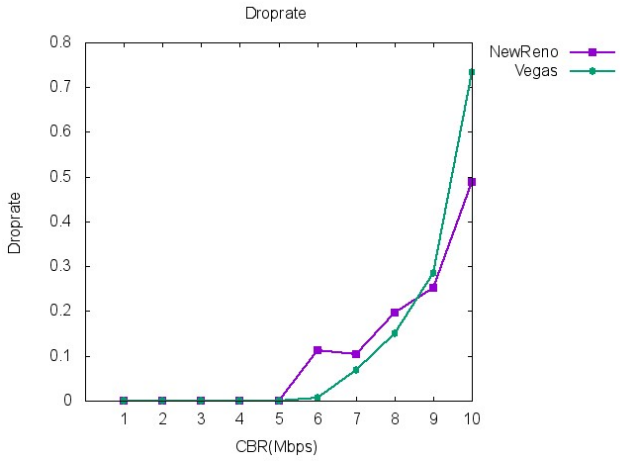

Figure4.2.2 Drop rate

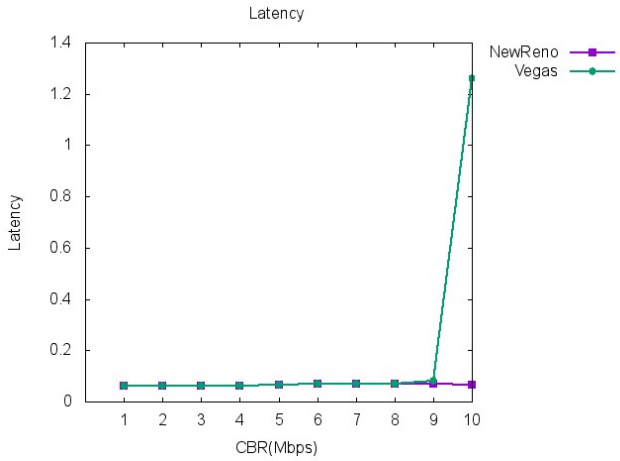

Figure4.2.3 Latency

A. Throughput

As shown in figure 4.2.1, NewReno and Vegas performs fairly under congestion, even though the change of their throughput sometimes is drastic.

B. Drop rate

Vegas reacts to congestion earlier as it drops more packets as shown in figure 4.2.2. But then the drop rate of these 2 variants are almost the same.

C. Latency

In terms of latency, Vegas increases drastically when CBR is over 9Mbps, while NewReno maintains unchanged. Thus it is unfair.

3) Reno vs Reno

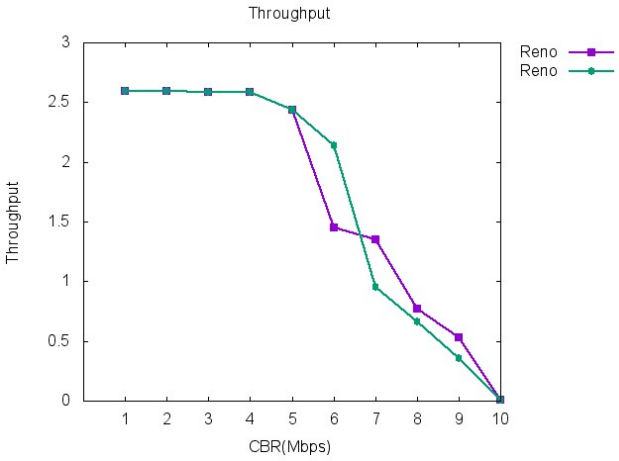

Figure4.3.1 Throughput

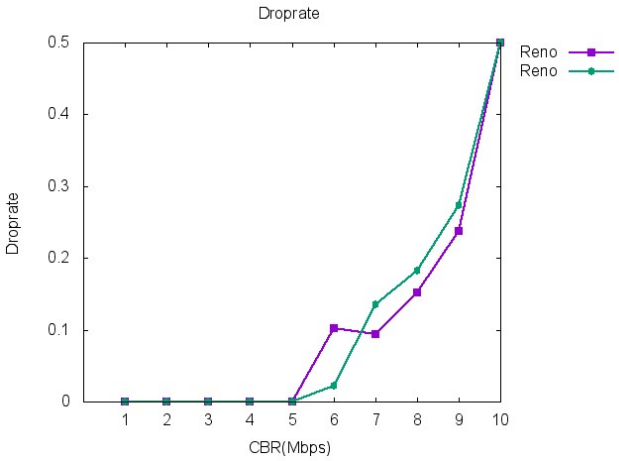

Figure4.3.2 Drop rate

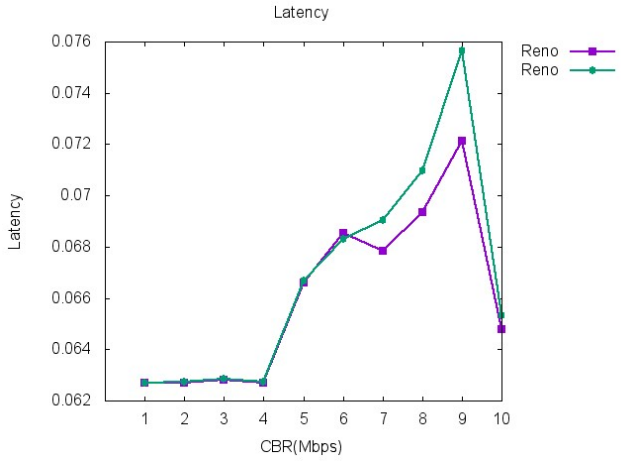

Figure4.3.3 Latency

A. Throughput

As we can see in figure 4.3.1, two Reno get fair throughput when sending packets on the same congested network.

B. Drop rate

Two Reno also get fair drop rate when operate on the same congested network with little oscillation.

C. Latency

The latency of two Reno running on the same network have the same trend towards congestions, but one of them might get a slightly lower latency than the other.

4) Vegas vs Vegas

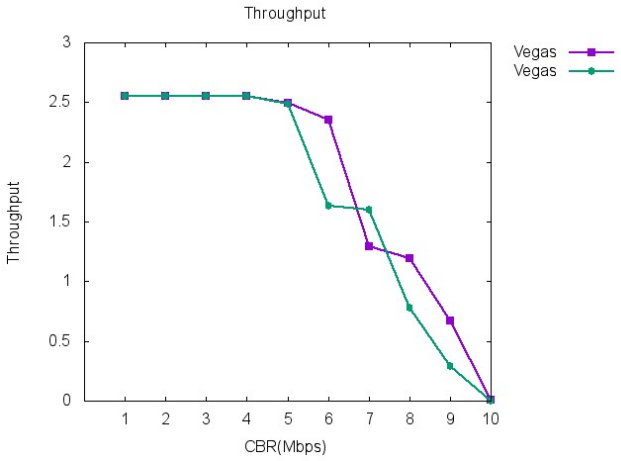

Figure4.4.1 Throughput

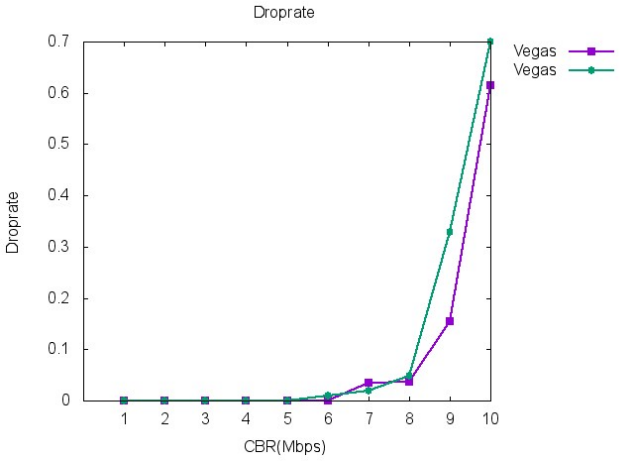

Figure4.4.2 Drop rate

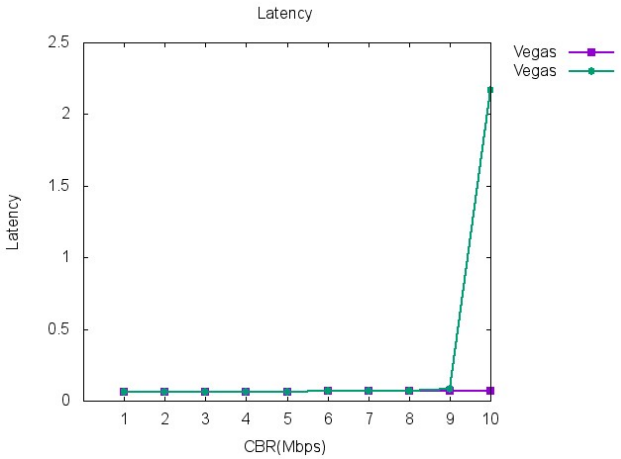

Figure4.4.3 Latency

A. Throughput

As we can see in figure 4.4.1, two Vegas get fair throughput when competing with each other the same congested network.

B. Drop rate

As shown in figure 4.4.2, two Vegas on the same network have the same trend towards congestion. But one of them get lower drop rate than the other.

C. Latency

As shown in figure 4.4.3, one of the two Vegas TCP stream gets a much higher latency when CBR's bandwidth reaches 9Mbps. But as we increase the bandwidth of CBR, the latency of these two Vegas start to interchange with each other as shown in figure 4.4.4.

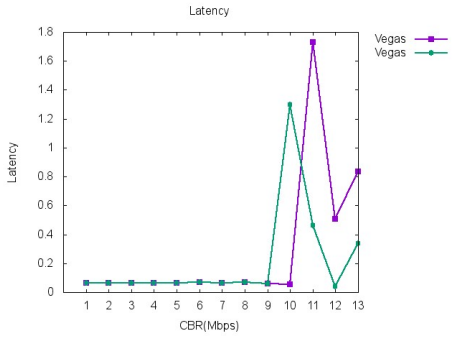

Figure4.4.4 Latency with CBR over 13Mbps

V. EXPERIMENT 3 RESULT ANALYSIS

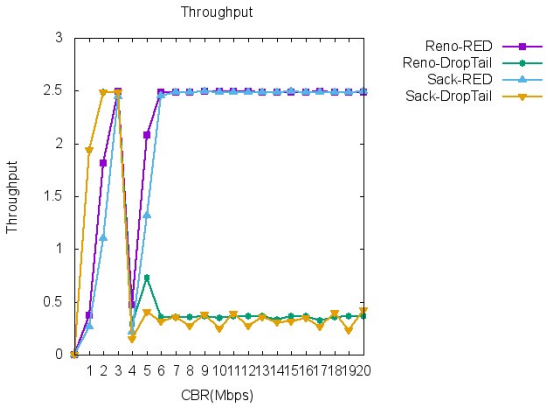

Figure5.1.1 Throughput

Before the CBR stream is added over the link, SACK has a better throughput than Reno, and when the CBR stream is added at 3.0 second, all the four combinations' throughput drop. Noticeably, not all the queuing discipline provides a fair bandwidth, Reno with Droptail get a really low throughput, and SACK with Droptail's throughput keep oscillating. In contrast, RED has a much better recovery from the congestion, and able to get a much better throughput. We conclude that the Droptail algorithm cannot provide a fair throughput when it encounters congestion; in contrast, RED recovers much faster and being able to utilize the available bandwidth, thus provides a better throughput.

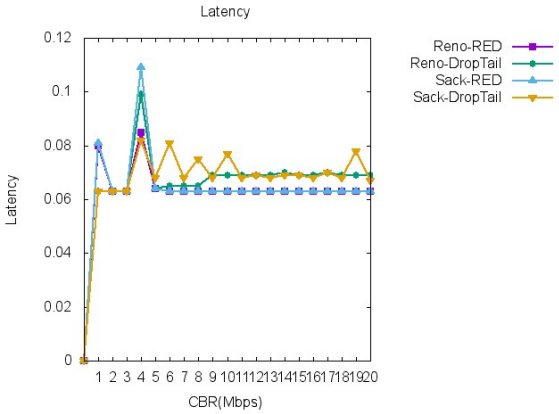

Figure5.1.2 Latency

To calculate the latency, we split the time into small time intervals, and calculate the average RTT of each interval. As shown in figure 5.1.2, when there is a surge CBR stream over the link, all the four combinations' latency rises. Generally RED has lower latency than Droptail, and SACK has lower latency than Reno because of its ACK mechanism. What is worth noting is that the Reno Droptail combinations always have a high latency even it has the lowest throughput shown in Figure 5.1.2.

VI. CONCLUSION

In these experiments, we analyzed the performance of 4 TCP variants and of different queuing strategies under congested network. As shown in Section 3, Vegas has a better throughput and drop rate than the Reno, NewReno and Tahoe, when the

network became really congested, only Vegas kept a small throughput in sacrifice of RTT. The “best” variant is the most suitable one, and Vegas have a better throughput than the other when the link is really congested. In Section4, the results show that different TCP variants when being deployed in the same network, not all of them get a fairly performance. Especially when two Vegas are deployed over the same link, one Vegas stream is likely to get a lower latency than the other one, when the network is congested. In the last experiment, we can find that queuing discipline plays a big role on the throughput and delay. RED shows perform better when the network is congested. And SACK RED outperforms all the other combinations, which suggests that SACK is better used with RED.

REFERENCES

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

[2] J.Chung and M.Claypool, NS by example

[3] Esha Kumar, Shivanka Chugh, S.P Ghrer, Performance Analysis of RED With Different TCP Congestion Control Mechanism

[4] A.Dana and A.Malekloo, Performance Comparison between active and passive queue management, *IJCSI Internaiton Journal of Computer Science Issues, vol. 7 , May 2010*