Performance Analysis of TCP Variants

Yunfan Tian
College of Engineering
Northeastern University, MA, 02115
Email: tian.yun@husky.neu.edu
NUID: 001273551

***Abstract*—This project contains 3 experiments, we use NS-2 to simulate congestion on a simple 6-node network topology with 4 TCP variants and analyze the performance in experiment 1. We also conduct experiment 2 to analyze the fairness between different TCP variants. Then we investigate the influence of queuing disciplines in experiment 3.**

I. Introduction

Network congestion is the reduced quality of service that occurs when a network node or link is carrying more data than it can handle. Different TCP variants provides different algorithms to control and avoid congestion. In this paper, we present performance and comparative analysis of TCP variants under increasing load and different queuing disciplines.

In experiment 1, we analyze the performance of four TCP variants (Tahoe, Reno, NewReno, Vegas) under increasing CBR load based on throughput, drop rate and latency.

In experiment 2, we analyze the fairness between different TCP variants (Reno/Reno, NewReno/Reno, Vegas/Vegas, NewReno/Vegas) based on throughput, drop rate, and latency.

In experiment 3, we compare the performance of different queuing algorithms (DropTail and RED) under constant CBR load based on throughput and latency.

II. Methodology

A. Network Topology

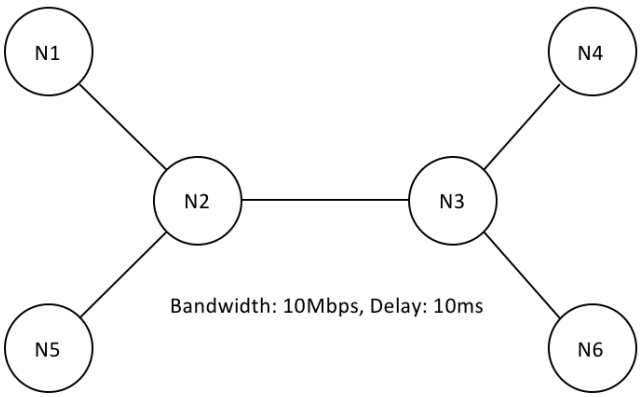

Fig. 1: Basic Network

Fig. 1 shows the basic network for our three experiments. There are 6 nodes and 5 links in network, the bandwidth and delay for each link are configured to **10Mbps** and **10ms**.

B. Tools

We use NS-2 to conduct all the experiments. NS-2 is a discrete-event computer network simulator, we use it to simulate congestion on network.

Then we use python to analyze the trace files generated by NS-2 and use gnuplot to plot the data.

C. Performance Evaluation

We consider throughput, drop rate and latency to evaluate and compare each TCP evaluation. From the trace logs generated by NS-2, we could get the behavior of each packet and then analyze the data flow in network.

Throughput is the amount of data moved successfully from one node to another in given period, measured in **Mbps**.

$$throughput = \frac{packets\ received \times packet\ size}{time}$$

Drop rate is the percentage of packets that travel across network fail to reach their destination.

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

latency is time interval between packets sent and received, measured in **s**.

$$latency = \frac{\sum packets\ received \times delay}{packets\ received}$$

D. Configurations

Experiment 1:

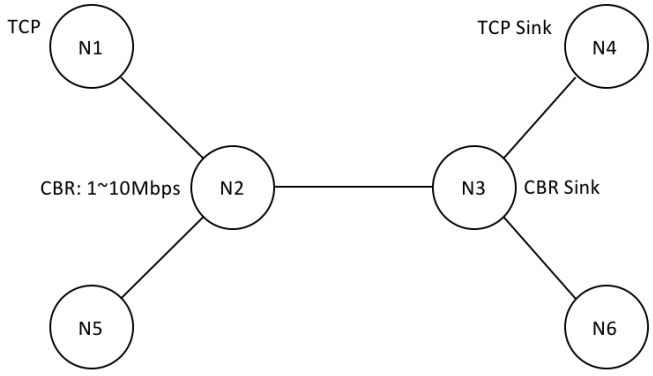

Fig. 2: Experiment 1

Fig. 2 shows the basic configuration in experiment 1. The queuing algorithm of each link is DropTail with a default

queue size. The test time is set to 10 seconds. The CBR (Constant Bit Rate) varies from **1Mbps** to **10Mbps**. We add CBP from N2 to N3, TCP from N1 to N4, then start N2 and N1 at time 0. Finally, we conducted experiments with TCP Tahoe, Reno, NewReno, Vegas and evaluated the performance.

Experiment 2:

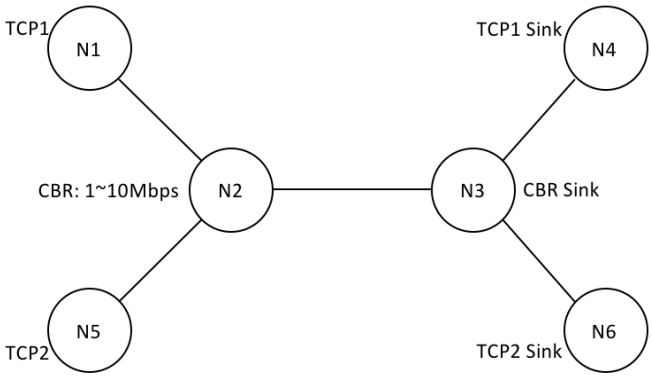

Fig. 3: Experiment 2

Fig. 3 shows the basic configuration in experiment 2. The only difference to experiment 1 is that we add another TCP stream from N5 to N6. Finally, conducted experiments with 4 pairs of TCP variants (Reno/Reno, NewReno/Reno, Vegas/Vegas, NewReno/Vegas) and present comparative analysis.

Experiment 3:

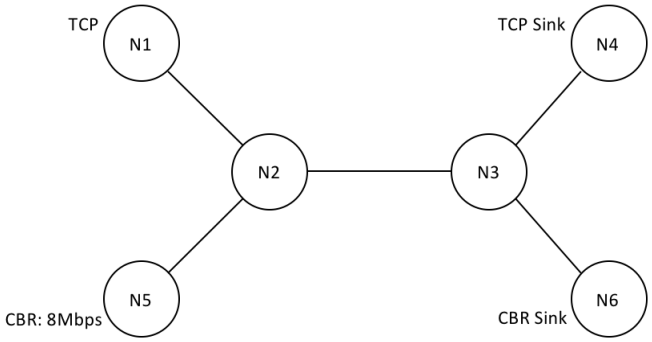

Fig. 4: Experiment 3

Fig. 4 shows the basic configuration in experiment 3. We add TCP from N1 to N4, CBR from N5 to N6. The link queuing algorithm is DropTail or Random Early Drop (RED), and the TCP variant is Reno or Sack. We set test time to 20s, queue limit to 10 and CBR to 8Mbps. Finally, we start CBR at **0s** and TCP at **4s**, and then monitor throughput and latency of TCP stream over 20 second period.

III. Experiment 1 Result Analysis

A. Throughput

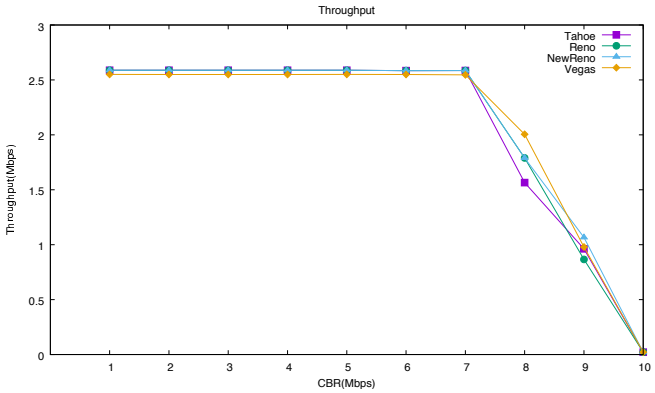

Fig. 5: Throughput

For CBR $\leq$ 7Mbps, the throughput of of all TCP variants stay stable around 2.6Mbps. Vega's throughput is a little less than the other three. After CBR exceed 5 Mbps, congestion raises, and all TCP variants begin to drop. However, Vega has better overall performance than the other three, Tahoe has least throughput. That's because Vegas conducts congestion detection at beginning, and when network become congested, the pre-detection help to keep a higher throughput.

B. Drop Rate

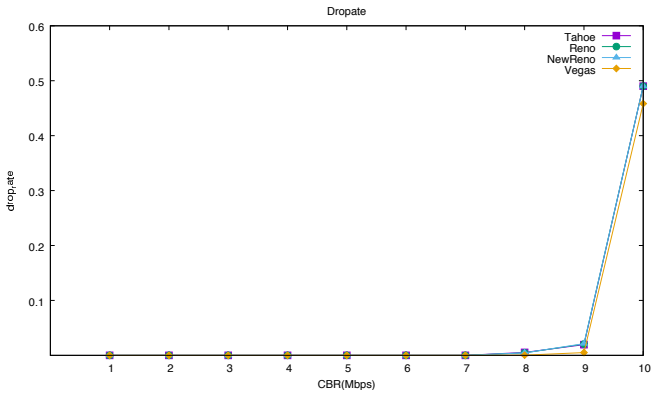

Fig. 6: Drop Rate

After the congestion begin, Vegas has best performance compare the other three. All TCP variants are sensitive to the congestion, but Vegas has 50% less drop rate compare to others when CBR is 9Mbps.

C. Latency

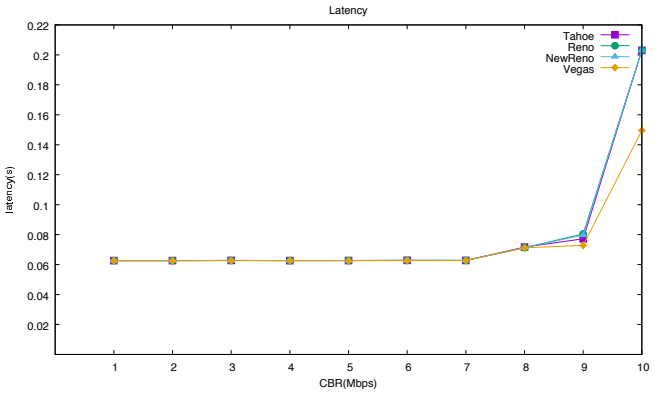

Fig. 7: Latency

All TCP variants have same latency before congestion, but Vegas still remains fairly low latency under congestion. As CBR increases to the bandwidth limit (10Mbps), Vegas keeps 0.15s latency, but other variants reach 0.2s.

IV. Experiment 2 Result Analysis

A. Reno vs Reno

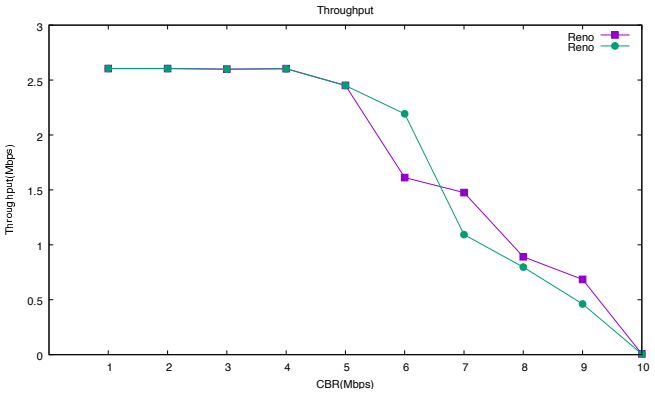

Fig. 8: Throughput

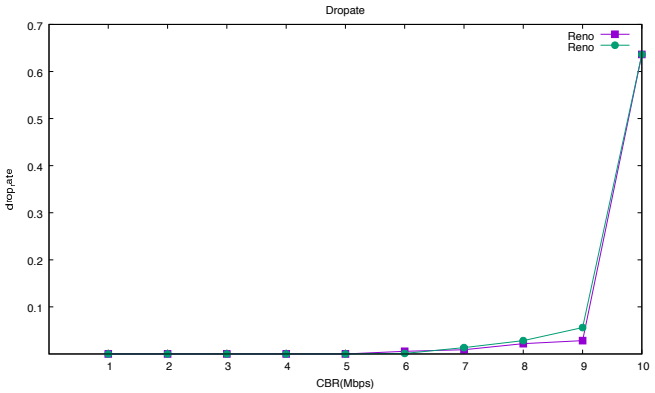

Fig. 9: Drop rate

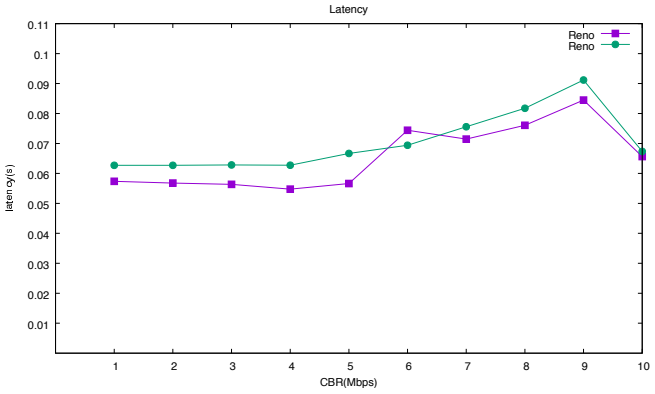

Fig. 10: latency

Fig. 8, Fig, 9 and Fig. 10 show the comparison of throughput, drop rate and latency between Reno and Reno. In Fig. 8 and Fig. 9, *throughput* and *Drop rate* value oscillated back and forth but keep close in the same track as CBR increased from 0 to 10Mbps. The oscillation is acceptable. In Fig. 10, One TCP Reno agent has a little lower latency than the other. Thus two TCP Reno maintain fair for throughput and drop rate.

B. NewReno vs Reno

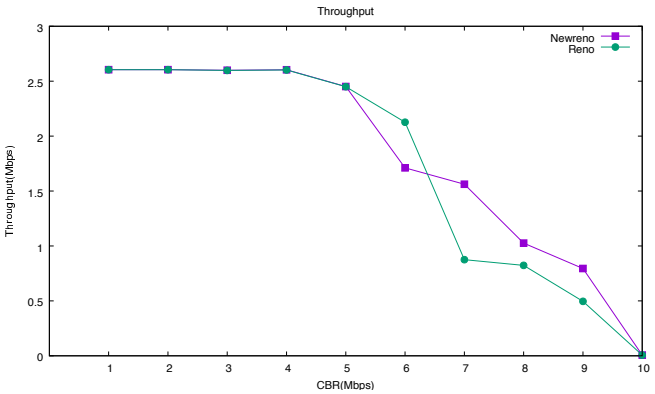

Fig. 11: Throughput

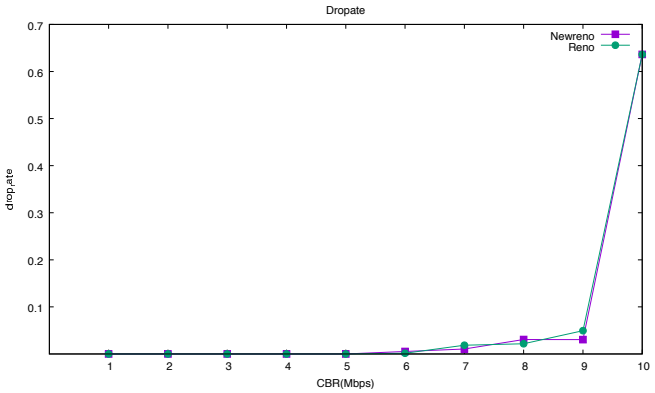

Fig. 12: Drop rate

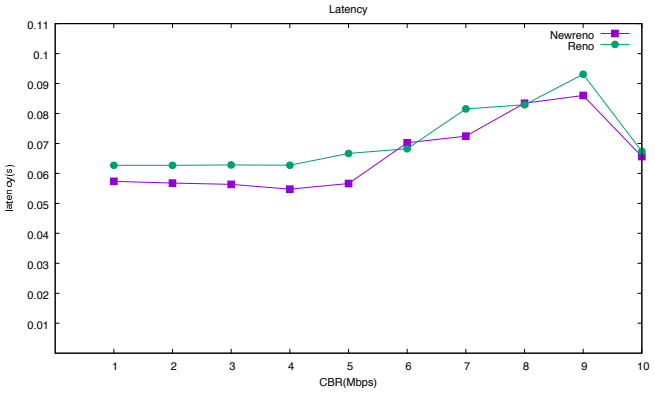

Fig. 13: latency

Fig. 11, Fig, 12 and Fig. 13 show the comparison of throughput, drop rate and latency between NewReno and Reno. In these three figures, the value oscillated back and forth but keep close in the same track as CBR increased from 0 to 10Mbps. The oscillation is acceptable. Thus, NewReno and Reno maintain fair for all three features.

C. *Vegas vs Vegas*

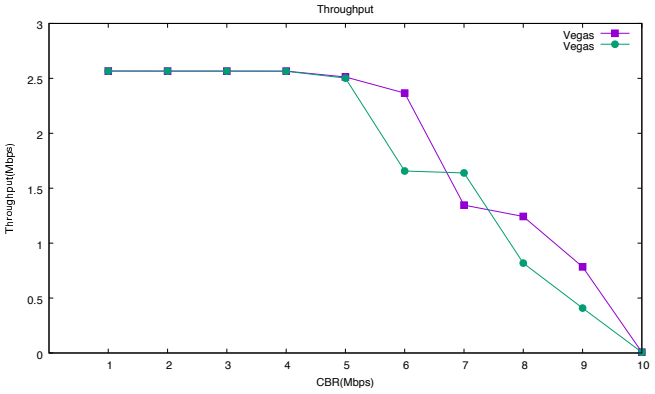

Fig. 14: Throughput

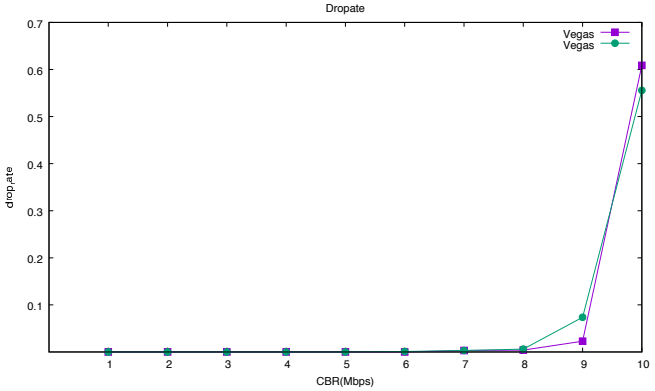

Fig. 15: Drop rate

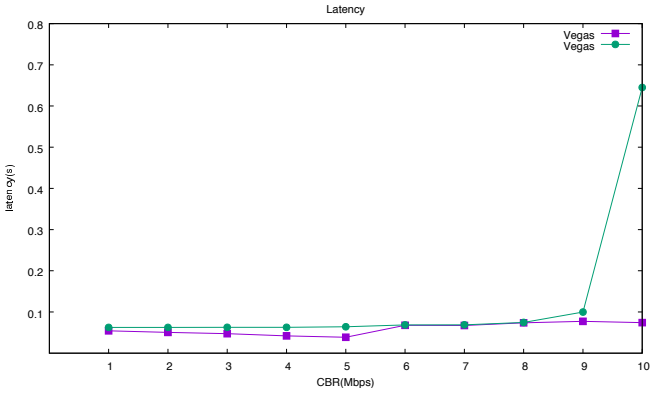

Fig. 16: latency

Fig. 14, Fig, 15 and Fig. 16 show the comparison of throughput, drop rate and latency between Vegas and Vegas. In Fig. 14 and Fig, 15, the throughput and drop rate value oscillated back and forth but keep close in the same track as CBR increased from 0 to 10Mbps. The oscillation is acceptable. However, in Fig. 16, one Vegas agent's latency differed much from another. When CBR reached 10Mbps, one Vegas agent has 0.65s latency, but another still maintain 0.08s. Thus two Vegas agents didn't keep fair. That's because Vegas provide congestion detection mechanism.

D. *NewReno vs Vegas*

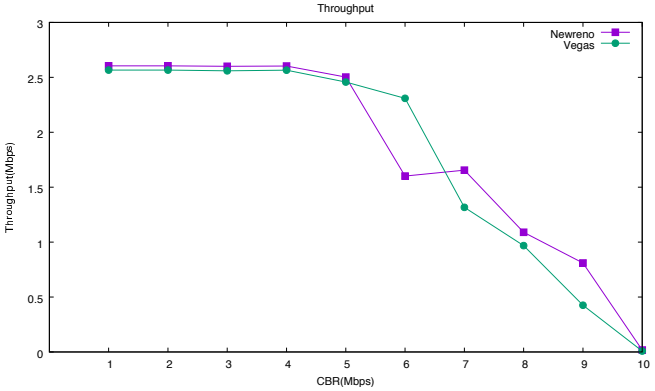

Fig. 17: Throughput

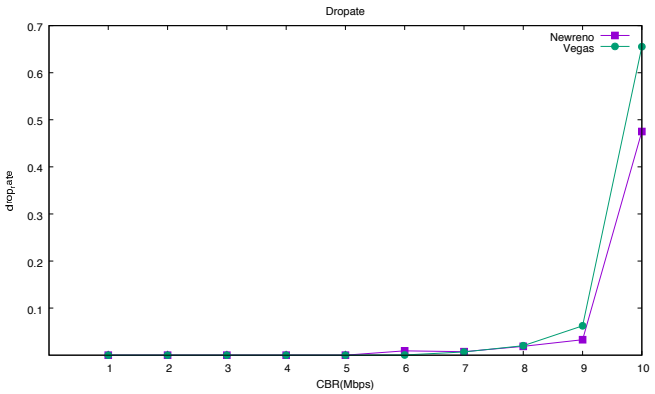

Fig. 18: Drop rate

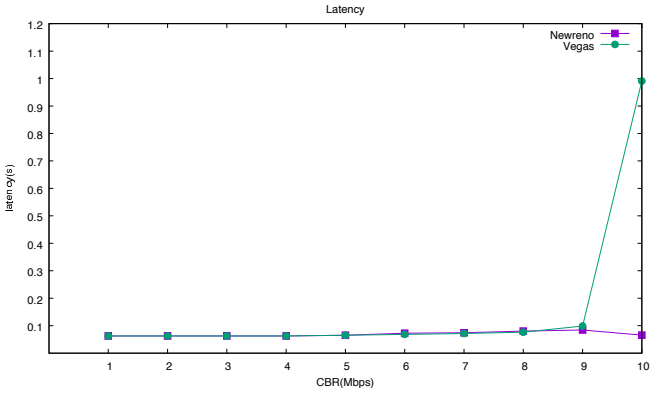

Fig. 19: latency

Fig.17, Fig, 18 and Fig. 19 show the comparison of throughput, drop rate and latency between NewReno and Vegas. In Fig. 17, the throughput value oscillated back and forth but keep close in the same track as CBR increased from 0 to 10Mbps. The oscillation is acceptable.

However, in Fig. 18 and Fig. 19, NewReno agent's drop rate and latency differed much from Vegas. NewReno has a lower drop rate under congestion. When CBR reached 10Mbps, Vegas has 1s latency, but NewReno maintain 0.08s. Thus NewReno and Vegas agents didn't keep fair. That's because when NewReno is dominate, Vegas supposed the network is congested and reduced its sending rate.

V. Experiment 3 Result Analysis

A. Throughput

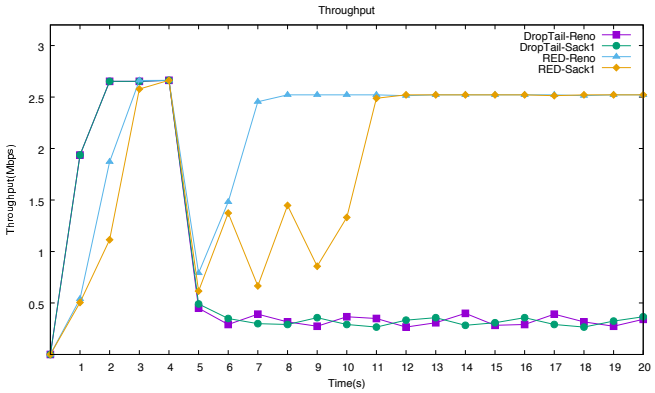

Fig. 20: Throughput

Before CBR added at 4s, DropTail has better performance than RED and achieves stability faster. When CBR added, all 4 TCP throughput drop dramatically. DropTail finally oscillated around 0.35Mbps since 5s. RED-Reno increased to 2.5Mbps since 7s and remain stable. RED-Sack oscillated and increased to 2.5Mbps since 11s and remain stable.

RED queuing algorithm generally has a better performance than DropTail under congestion. Each queuing discipline didn't provide fair bandwidth to each flow. RED recovers faster from throughput drop and has a better utilization of bandwidth.

B. Latency

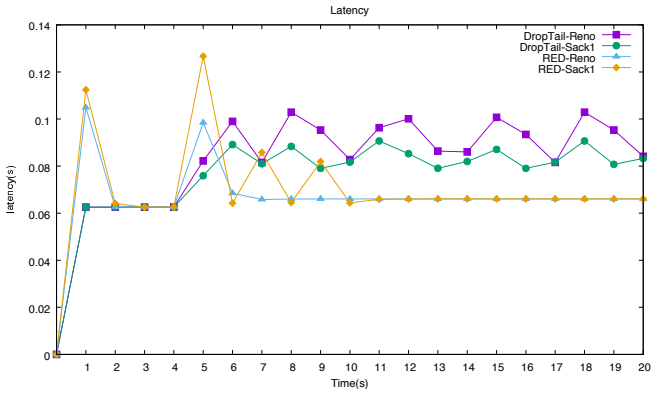

Fig. 21: Latency

We calculated average latency in each second and compare.

Before CBR added at 4s, DropTail achieves stability faster and both queuing algorithm have same 0.06s latency. When CBR added, all 4 TCP latency increase dramatically. DropTail finally oscillated around 0.09s since 6s. Sack has lower latency than Reno. RED-Reno drops to 0.06s since 7s and remain stable. RED-Sack oscillated and drops to 0.06s since 11s and remain stable.

RED queuing algorithm generally has a better performance of latency than DropTail under congestion. RED is a good idea

with SACK because it has low latency with high throughput. The only problem is RED-Sack takes more time to achieve stability.

VI. Conclusion

In this paper, we simulate different TCP variants and queuing disciplines under congestion with NS-2 and analyze the performance of them based on throughput, packets drop rate and latency. From these 3 experiments, we can conclude that:

1) For single TCP agent, Vegas has better performance than Tahoe, Reno and NewReno in throughput, drop rate and latency.
2) For two TCP agents in same network, they may perform unfair to each other. Reno is fair to Reno; NewReno is fair to Reno. Vegas is not fair to Vegas in latency; NewReno is not fair to Vegas in drop rate and latency.
3) RED is a good idea with SACK because it provides low latency with high throughput. The only problem is RED-Sack takes more time to achieve stability.

References

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

[2] Wikipedia, *Network congestion*, https://en.wikipedia.org/wiki/Network_congestion