Performance Analysis of TCP Variants

Zhongxi Wang, Shiyu Wang

Professor: David Choffnes

Feb. 25th, 2017

Introduction

In this paper, we present the result of our study on the performances of different TCP variants. The NS-2 network simulator is used as the main tool for this study. We compared the performances of TCP Tahoe, Reno, New Reno and Vegas under different situations. In experiment 1, we analyze the performance of each one under different congestions. In experiment 2, we simulate two TCP flows to analyze how well they share bandwidth and their fairness to each other. In experiment 3, how queueing mechanism used in buffer impacts the TCP performance is studied. At the end of the paper, the key results from these experiments are highlighted.

Methodology

We conducted the simulation in NS-2. NS-2 is an object oriented network simulator which can simulate TCP, routing and multicast protocols over LANs and WANs. In terms of TCP's performances, following properties are studied:

1. Throughput, the amount of successfully transferred data per second. The unit is in MB/s
2. Average RTT, the average round-trip delay time of packets. The unit is in second.
3. Drop rate, the percentage of package dropped. The unit is in percent.

The simulation of network topology, flow starting time, buffer size and queuing mechanism are set up by TCL scripts. The trace results given by NS-2 are exported to standard output and parsed by a Java program which also calculates throughput, average RTT and drop rate. We also conducted T-test for our results in order to check the significance of their statistical difference.

Experiment 1

In this experiment, we use a network topology as shown below. One TCP flow is sent from N1 to N4, and one CBR flow is sent from N2 to N3. The bandwidth of the links of N1-N2, N2-N3 and N3-N4 are all 10 Mbps. The experiment starts with CBR flow rate at 1 Mbps, it is increased in 1 Mbps every iteration until bottleneck capacity 10 Mbps is reached. This experiment is conducted with following TCP variants: Tahoe, Reno, NewReno and Vegas. For each TCP variant, we run the experiment 10 times, in order to

calculate T-value. The results of this experiment are the average results of 10 times. Fig 1.2-1.4 show the results in the form of throughput, drop rate and RTT.
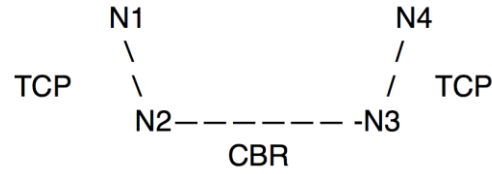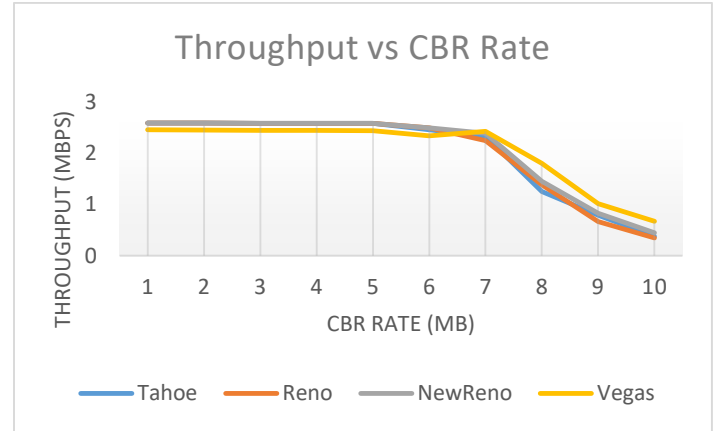

Fig. 1.1 Topology
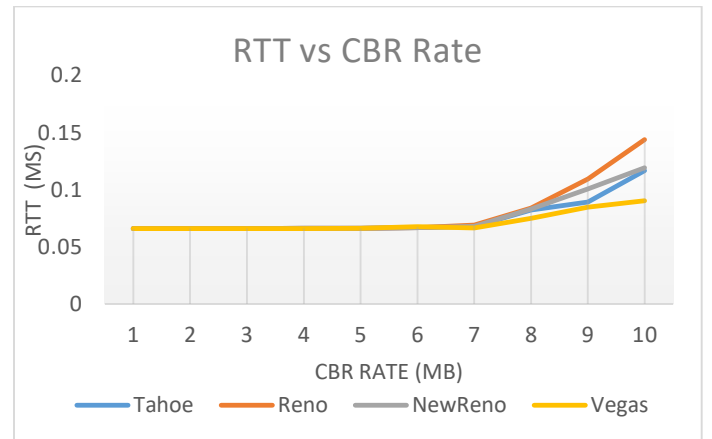

Fig. 1.2 Throughput of all TCP variants vs. CBR rate

Fig. 1.3 RTT of all TCP variants vs. CBR rate

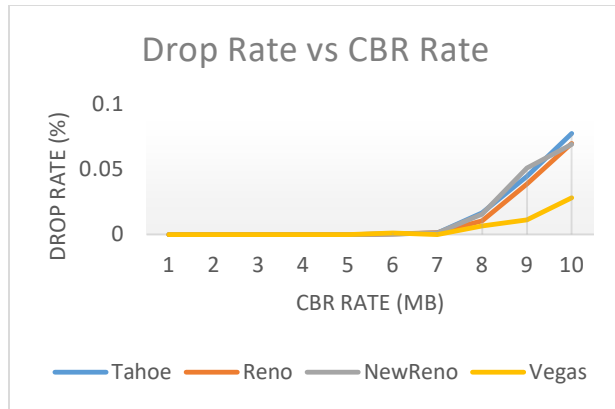Fig. 1.4. DropRate of all TCP variant vs CBR rate

T-Value	Tahoe	Reno	NewReno	Vegas
Tahoe	0	8.4079	5.8644	5.9734
Reno	8.4079	0	16.747	5.9188
NewReno	5.8644	16.747	0	21.955
Vegas	5.9734	5.9188	21.955	0

Table 1.1. T-Value for throughput

Table 1.1 shows the T-value of the experiment, which is produced by running the experiment than 10 times. With p-value at 0.001, and degree of freedom at 18, if T-value is larger than 3.92, then the null hypotheses that the two TCPs have the same throughput can be rejected. As the value in the above table shows, we can safely conclude all the TCPs in our experiment have significantly different throughput. In the following paragraphs of this section, we will discuss the causes of these differences.

Tahoe, from the experiment, it shows Tahoe has a relatively low throughput and high drop rate, especially when the CBR rate is high. This is due to the fact that Tahoe takes a full retransmission time out to detect a packet loss. This causes many packets to be transmitted in vain.

Reno, unlike Tahoe, Reno enters fast retransmit once it receives three duplicate ACKs. As a result, it would not waste many packets which would be discarded by the receiver. Therefore, we can see it has a much better drop rate than Tahoe. However, due to the fast retransmission of Reno, it can send packets at a fast rate even when the network is pretty congested, which can cause many packets to be stranded in buffer for a long time. Therefore, it has a longer RTT than other TCPs.

NewReno, pretty similar to Reno, NewReno has one major difference which is it does not exit fast retransmission until all the data in the pipeline has been acknowledged. Due to this difference, NewReno is generally more aggressive than Reno. As a result, we can see that it has a high RTT and high drop rate. However, when the congestion is low, its throughput is the best among all the TCPs.

Vegas, it uses a much more accurate mechanism to estimate RTT and decide the retransmission timeout. We can see this has an enormous benefit when congestion is high. Among all the TCPs, Vegas has the best performance in terms of drop rate and RTT. However, Vegas uses additive increases in the congestion window, which makes it less aggressive than the other TCPs. This is reflected in the relatively lower throughput of it when the congestion is low. Whereas, when the congestion is high, Vegas' better RTT estimation mechanism makes it outperformed all the other TCP in throughput.

Experiment 2

This experiment mainly focused on fairness between different TCP variants. As we expected, all of these variants should be fair to one another, so different TCPs share equal bandwidth. However, we knew fairness is one of the problems of TCP variants in reality.

For this experiment, three flows are set in the network. In Fig. 2.1, one CBR is added at N2 and sink at N3. Then, two TCPs are from N1 to N4 and N5 to N6, respectively. The bandwidth of each link is set at 10 Mbps. The performance of these TCP flows was measured by changing the CBR rate until it reached the bottleneck capacity.
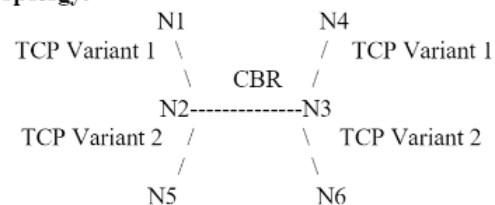

Fig. 2.1 Topology

In this experiment, we tested and measured average data for four pairs TCP variants, including:

    a.   Reno vs. Reno
    b.   NewReno vs. Reno

c. Vegas vs. Vegas
d. NewReno vs. Vegas

As in Experiment 1, we plot the average throughput, packet loss rate, and RTT of each TCP flow as a function of the bandwidth used by the CBR flow.
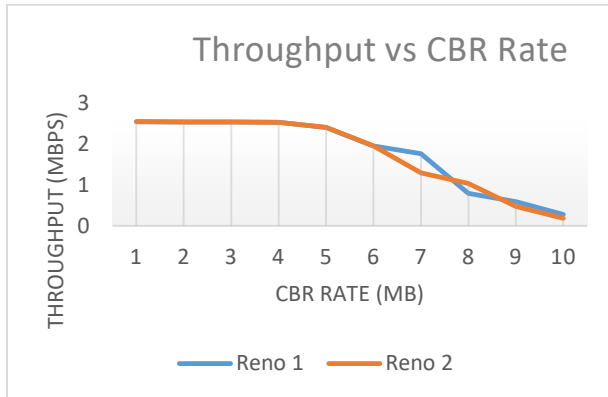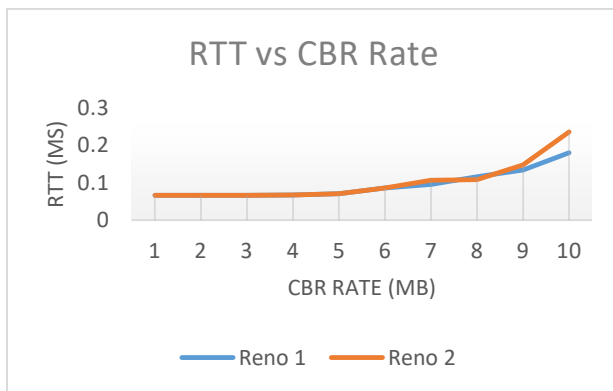
Reno vs. Reno

Throughput vs CBR Rate

Fig. 2.2 Throughput of Reno vs. Reno

RTT vs CBR Rate

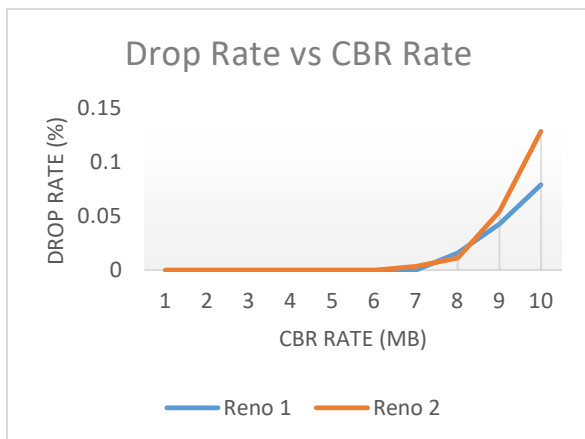Fig. 2.3 RTT of Reno vs. Reno

Drop Rate vs CBR Rate

Fig. 2.4 Packet Drop Rate of Reno vs. Reno

From the figures above, it shows the throughput of two TCP Reno are similar. As the CBR rate increase, the throughput goes done together with slightly different because of the bottleneck capacity and buffer limitation. This shows they share the bandwidth equally. Although the drop rate and latency figures show that Reno 2 has higher value, but it's actually less than 0.05ms and 0.05%. On the other hand, the T-test was performed on the data of throughput of two TCP variants:

T-Value	Reno 1	Reno 2
Reno 1	0	34.345
Reno 2	34.345	0

Table 2.1 T-Value of Reno vs. Reno

According to the Table – 1 with P less than 0.01 and T-value is big enough to against null hypothesis. Therefore, we can conclude the data is significant that Reno and Reno are fair to each other.

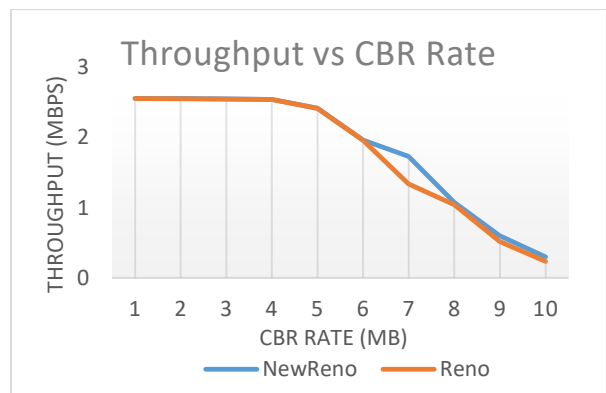## New Reno vs. Reno

Throughput vs CBR Rate

Fig. 2.5 Throughput of New Reno vs. Reno

New Reno has higher throughput than Reno when they are sharing the bandwidth. The Fig. 2.5 shows that this combination is unfair. As the CBR rate increase, it is obviously that New Reno always has higher throughput. Again, the T-test was performed against the null hypothesis with big T-value for P less than 0.01.

T-Value	New Reno	Reno
New Reno	0	16.747
Reno	16.747	0

Table – 3 T-Value of New Reno vs. Reno

RTT vs CBR Rate

Fig. 2.6 RTT of NewReno vs. Reno

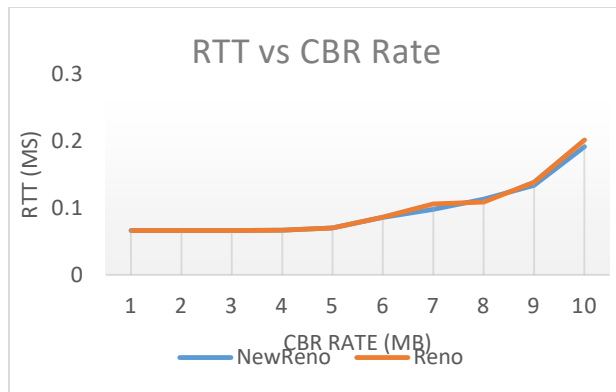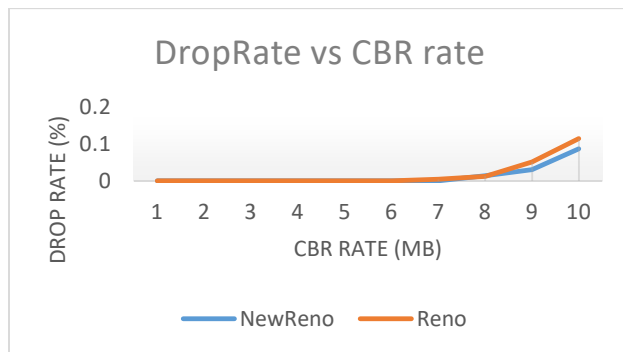## DropRate vs CBR rate

Fig. 2.7 Packet Drop Rate of NewReno vs. Reno

In another aspect, Reno has higher Drop rate and RTT in the Fig. 2.6 and Fig. 2.7. It is because New Reno has the advantage of fast retransmit which can detect multiple packet losses and does not need to wait for retransmission of packet as in the case of Reno. This gives New Reno priority in bandwidth share.

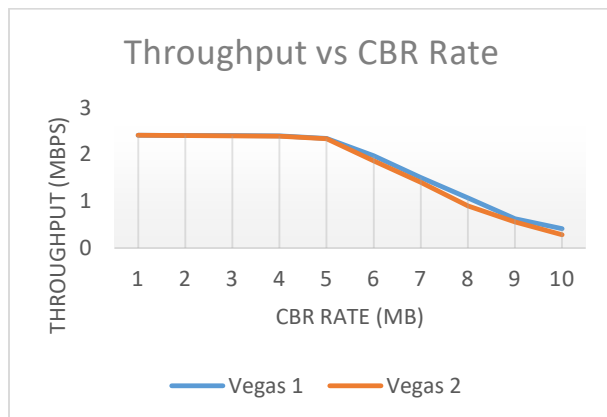**Vegas vs. Vegas**

Throughput vs CBR Rate

Fig. 2.8 Throughput of New Reno vs. Reno

From the figures, although two variants have similar latency and drop rate, Vegas 1 has a little bit higher throughput in the bandwidth share. The reason

is because Vegas is different from other TCP variants that it applies delay-based congestion avoidance. It considers queue building rather than loss of segment for congestion detection to adjust the sending rate. Hence, when it notices the rate is lower compared to the expected rate, it increases its rate of transmissions to utilize the bandwidth. Conversely, transmission rate becomes lower when rate is very to the expected rate, which helps to avoid over saturation of the bandwidth of the network. This is the reason that both of them have amazingly low Drop rate. In this case, one Vegas increased the sending rate. And the other one detected it and reduce the transmission rate correspondingly. As shown in Fig. 2.8, the difference value between two Vegas are changing over time. As soon as they get close, one will start trying to increase rate and the other acts oppositely. However, it won't influence the RTT and Drop rate. Therefore, we conclude two Vegas variants are fair in bandwidth share, only slightly difference in throughput because of congestion avoidance algorithm. T-test is provided as well against null hypothesis. Again, the large T-value with P less than 0.001 can well prove the value is significant.

T-Value	Vegas 1	Vegas 2
Vegas 1	0	16.15
Vegas 2	16.15	0

Table – 5 T-Value of Vegas vs. Vegas

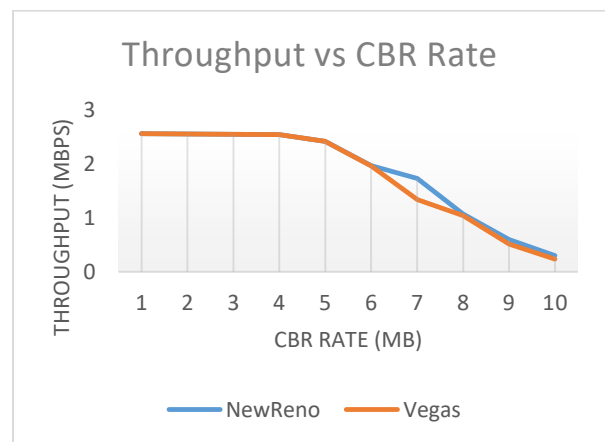**NewReno vs. Vegas**

Throughput vs CBR Rate

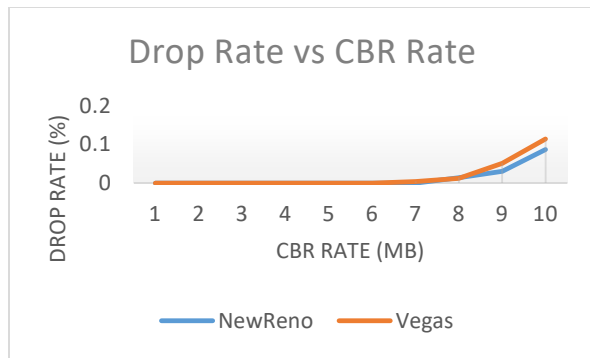Fig 2.11 Throughput of New Reno vs. Vegas

Fig 2.12 Packet Drop Rate of New Reno vs. Vegs

Comparing between New Reno and Vegas, the former is more aggressive. No matter about the start time, New Reno takes most of the bandwidth as the CBR rate gets closer to bandwidth. Because of the congestion avoidance algorithm talked in the last section, Vegas calculate have calculated the bandwidth while New Reno is still in the slow-start and has no idea about the bandwidth. During this period, Vegas reduces its transmission rate after every RTT, meanwhile, New Reno takes the majority of the bandwidth. Therefore, New Reno has a better throughput in Fig. 2.11. Therefore, in this pair, TCP New Reno is unfair to Vegas. T-test is provided below to against null hypothesis.

T-Value	New Reno	Vegas 2
New Reno	0	17.964
Vegas	17.964	0

Table – 7 T-Value of New Reno vs. Vegas

Experiment 3

In the last experiment we compare the influence of queuing displaces of two algorithms, Drop Tail and Random Early Drop (RED). As shown in Fig. 3.1 below, One TCP is set up from N1 to N4 and started at the start. After the TCP flows is steady in one to two seconds, the CBR flow will start running from N5 to N6.

Topology:

```
            N1              N4
TCP Variant 1  \          /  TCP Variant 1
                \        /
              N2-------------N3
   CBR       /          \      CBR
            /            \
           N5             N6
```
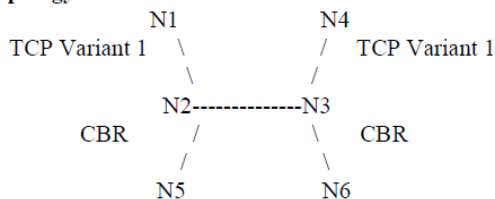
Fig. 3.1 Topology

During the forty times experiment with various starting time of CBR, average throughput, packet drop rate and latency are recorded every 0.5 seconds and analyzed with different queue displaces algorithms.

Does each queuing discipline provide fair bandwidth to each flow?

For this experiment, it is obvious the queuing discipline doesn't provide fair bandwidth to each flow, because the CBR flow does not care about the packet drops. However, the TCP flow is affected.

How does the end-to-end latency for the flows differ between DropTail and RED?
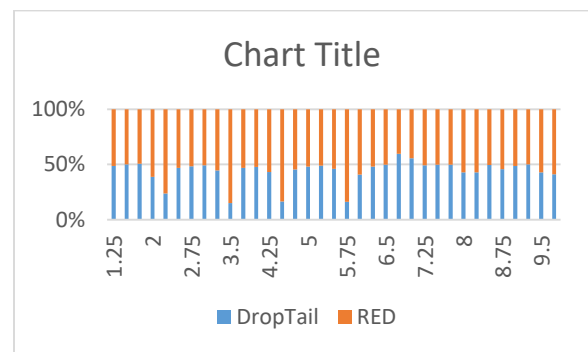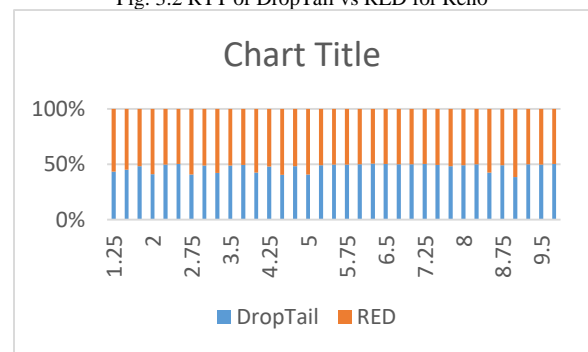

Fig. 3.2 RTT of DropTail vs RED for Reno

Fig. 3.3 RTT of DropTail vs RED for Sack

From the Fig. 3.3 and Fig. 3.4, it shows no matter for Sack or Reno TCP, RED algorithms take more time during transmission. The reason is RED monitors the queue size and drops packets based on the queue capacity. Hence, there will be a small delay as the packet transmitted in the flow. By contrast, DropTail will increase the buffer size over time.

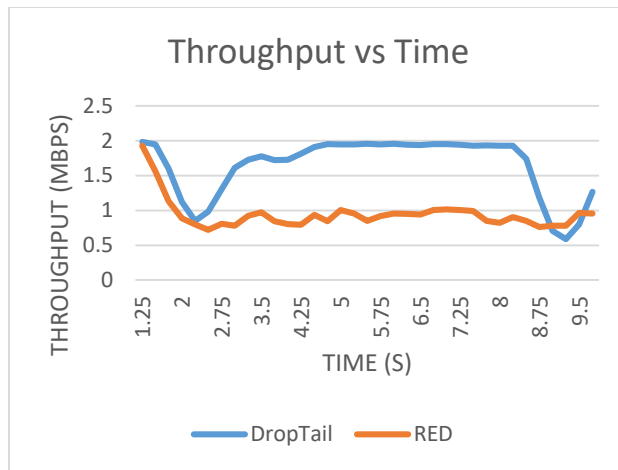**How does the TCP flow react to the creation of the CBR flow?**

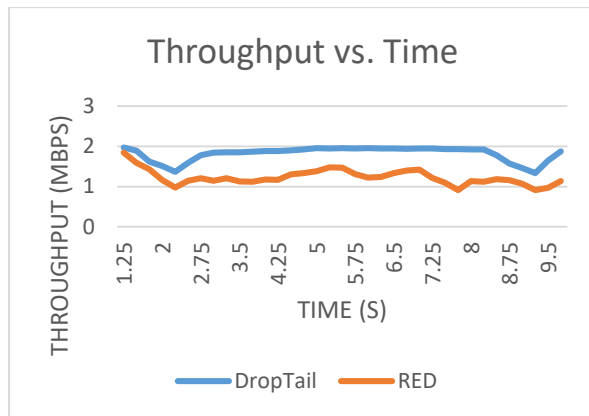Fig. 3.4 Throughput of DropTail vs RED for Reno

Fig 3.5 Throughput of DropTail vs RED for Sack

According to the Fig. 3.4 and Fig. 3.5, the throughput of TCP decrease dramatically when CBR flow is added into the network. As starting of CBR flow, the packets start dropping because of the limited queue size. However, after a short time, DropTail gives a better throughput as started, then decrease again for both Sack and Reno. Oppositely, the throughput using RED algorithm keeps the low rate. The is because DropTail only drops packets when the buffer is full, so it can keep the high throughput when there is still space in the buffer. However, RED drops packets randomly when it detects congestion based on its calculation. And the probability of dropping packets is proportional to the queue length.

Is RED a good idea while dealing with SACK?

As Fig. 3.5 shows, RED algorithm with Sack shows better than Red with Reno in Fig – 16, although it still has lower throughput than using DropTail algorithm. The reason is RED requires random and more resending than DropTail while transmission. Because Sack has a different sink using selective ACKs, which includes received, out-of-order sequence numbers in TCP header to tell sender about holes in the sequence. Therefore, it is much easier for the sender to resend packets than using Reno.

References

R. Braden, V. Jacobson and L. Zhang, "TCP extensions for highspeed

TCP Vegas: New Techniques for Congestion Detection and Avoidance L. Brakmo, S. O'Malley and L. Peterson Proceedings of the SIGCOMM '94 Symposium, August 1994, pg. 24-35.

Fall, Kevin, and Sally Floyd. "Simulation-based comparisons of Tahoe, Reno and SACK TCP." *ACM SIGCOMM Computer Communication Review* 26.3 (1996): 5-21. Web.

"RED (Random Early Detection) Queue Management." *RED (Random Early Detection) Queue Management*. N.p., n.d. Web. 28 Feb. 2017.