Performance Analysis of TCP Variants

Xiaokang Xin, Jiaxin Lyu

Northeastern University, Boston, MA

shogunx@ccs.neu.edu jiaxinlyu@gmail.com

1 Introduction

The TCP (Transmission Control Protocol) is the most widely used transport layer protocol in the world. It is aimed to provide reliable Internet service at transport layer in complex network conditions, different variants of TCP have emerged to provide a better performance with their different approaches to handle network congestion. Therefore it is important for us to have a good understanding of how each TCP variant performs congestion control and how they react in complex network conditions.

In this paper we present analysis of four TCP variants, Tahoe, Reno, New Reno and Vegas, from three major perspectives, which are performance under congestion, fairness between each other and performance under different queuing disciplines. All experiments and analyses are based on simulation results of NS2, which is a discrete event driven network simulator primarily useful for simulating local and wide area networks.

In section two we give the methodologies used in these experiments, in section three we present the experiment results and analysis for them. We found that all variants are able to provide congestion control to a certain level, however when congestion is really severe the performance of variants drops accordingly. Based on our experiment condition, New Reno gives a better overall throughput, while Vegas gives a smaller overall latency. Also New Reno can give better throughput when deployed with other variants since its congestion control approach is more aggressive. In last experiment we show that queuing discipline DropTail has a better influence than RED on TCP variants.

2 Methodologies

2.1 Topology and experiment parameter.

As mentioned earlier we use NS2 simulation to conduct our experiments, and the topology used by these experiments are shown in Fig. 1. We set six nodes connected by duplex link of 10Mbps bandwidth and 10ms delay as the figure shows.

For experiment one, we set TCP source at N0, sink at N3, and the data over this TCP flow is generated by a FTP application attached to N0. Meanwhile a UDP connection is set up from node N1 to node N2 with a CBR (Constant Bit Rate) flow. For each TCP variant we change the CBR rate from 0.5Mbps to 10.5Mbps increasing by a step of 0.5Mbps each time, and for each CBR rate we let simulation running for 30 seconds.

For experiment two we use the same setup like experiment one plus that we add another TCP flow from node N4 to N5.

For experiment three we set one TCP flow from N0 to N3, the other CBR flow from N4 to N5, and tested with queuing disciplines DropTail and RED on all the links.

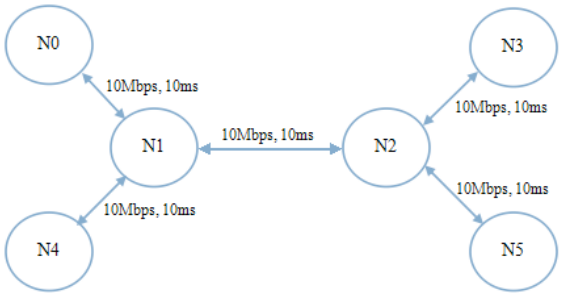

Fig. 2.1 Topology of the network.

2.2 Analyzing method.

One important reason we choose NS2 simulation for our experiments is that it can generate very detailed trace file which contains all necessary information we need to analyze network condition, such as packet sequence number, event time, source node, destination node, packet size and so on.

To analyze the trace file we wrote our own Ruby scripts to parse it, and we use Gnuplot to plot the graphs for further interpretation.

3 Experiment Results

3.1 TCP Performance Under Congestion

In this experiment we study the performance of four TCP variants under different levels of congestion. We calculate throughput, drop rate and latency (RTT) as a function of CBR rate, which increase from 0.5Mbps to

10.5Mbps with a step of 0.5Mbps.

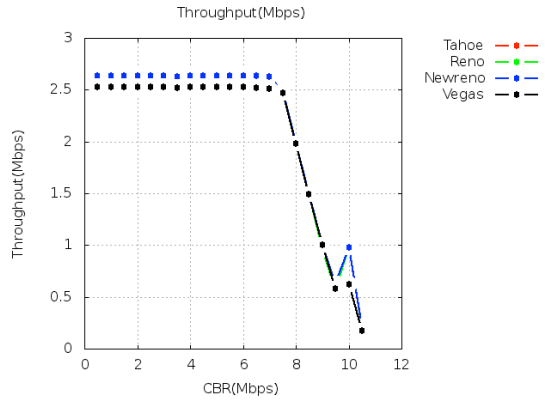

Fig. 3.1 Throughput of different TCP variants.

Fig. 3.1 shows that when rate of CBR flow increases the throughput of TCP variants changes accordingly. When CBR is below 7Mbps, it has little impact on TCP flow, as the rete goes larger it will compete bandwidth and gives pressure to TCP flow. As we can see when CBR rate reaches bottleneck capacity 10Mbps, the throughput of all variants get close to zero meaning the congestion is really bad. Also by inspecting raw data of the graph we noticed that when there is no congestion, Tahoe, Reno and New Reno have almost same throughput greater than 2.6Mbps while Vegas has smallest throughput around 2.5Mbps.

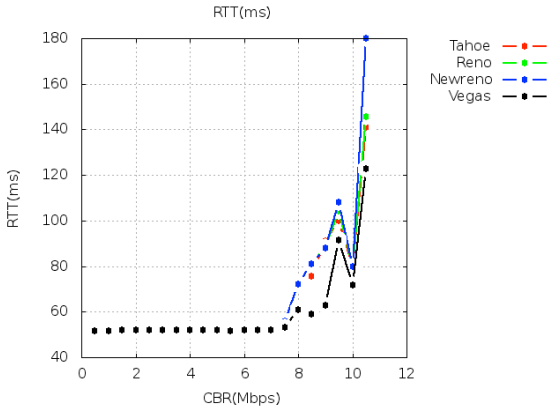

Fig . 3.2 Round trip time of different variants.

Fig. 3.2 shows the relationship between RTT and different rate of CBR flow. When there is no congestion, all variants have RTT around 50ms, as CBR becomes larger than 7Mbps, RTT of all variants increase accordingly, New Reno grows most fast while Vegas grows most slowly, this can be explained by the fact that Vegas use delay based congestion control algorithm so that it can detect congestion based on the increase of RTT, thus avoid congestion more early than other variants. Also since New Reno retransmits the packets when every duplicate ACK is received, so it is more sensitive to congestion and RTT increases drastically.

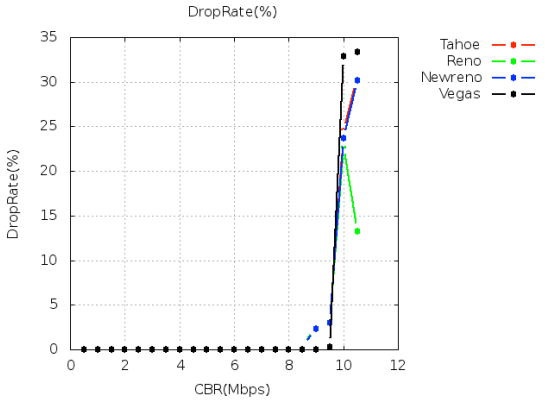

Fig. 3.3 Drop rate of different variants.

Fig 3.3 shows the relationship between drop rate and rate of CBR flow. When there is no congestion, all TCP variants have fairly good performance on drop rate of almost zero. Similar to Fig. 3.2, when CBR rate increases, congestion in the links become worse, drop rate of New Reno increases first, when CBR flow takes up almost the whole bandwidth, all variants have a increasingly large drop rate. An interesting phenomenon in Fig. 3.3 is that when CBR rate is 10.5Mbps, the drop rate of Reno decreases almost half from the value when CBR is 10Mbps, we think the fast recovery feature of Reno is the main reason for this change.

From this experiment we learned that congestion is a big factor that impacts the performance of TCP, although all TCP variants handle congestion fairly well, when congestion is severe throughput of TCP drops quickly while RTT and drop rate increase quickly as well. We observed that New Reno has a better overall throughput, Vegas has relatively smaller overall latency and drop rate, also we think overall "best" TCP varies depending on the statistics we pick and on the various experiment conditions such as parameters and topology.

3.2 Fairness Between TCP Variants

In this experiment we compare the fairness between two TCP variants when deployed in one network together. The purpose of this experiment is that, in real world scenario it is very common to have different devices running on top of different TCP protocols, so it is important for these variants share bandwidth fairly to each other.

The topology is same with the one used in experiment one and shown in Fig. 2.1. We conducted four groups of comparison, each time place one TCP variant at node N0, the other at node N4, and their sink nodes at N3 and N5 respectively.

3.2.1 Reno/Reno

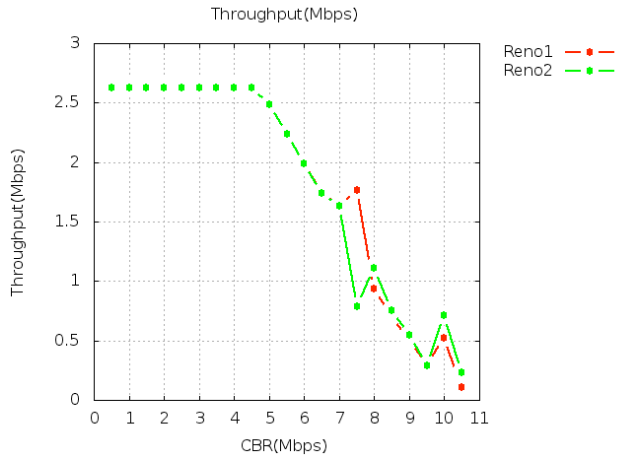

Fig. 3.4 Throughput of two Reno TCP.

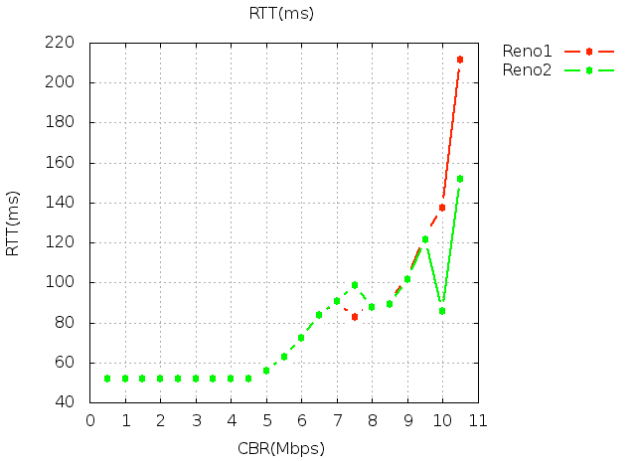

Fig. 3.5 RTT of two Reno TCP.

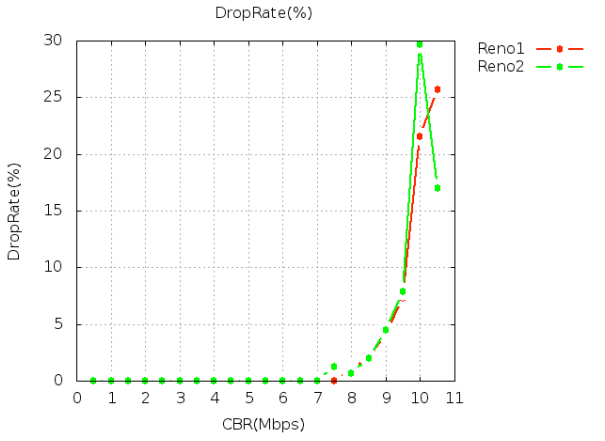

Fig. 3.6 Drop rate of two Reno TCP.

Fig. 3.4, 3.5, 3.6 shows the throughput, RTT and drop rate of two Reno TCP respectively. We can tell from these figures that when available bandwidth becomes smaller as CBR increases, two Reno TCP began to compete for bandwidth, especially when CBR is 7.5Mbps we can see that throughput of Reno1 gets over that of Reno2, but as CBR continues to increase, two TCP flows act similar to each other. Same behavior like can be seen in RTT and drop rate figure. So we conclude that two Reno TCP will be fairly to each other in the same network.

3.2.2 New Reno/Reno

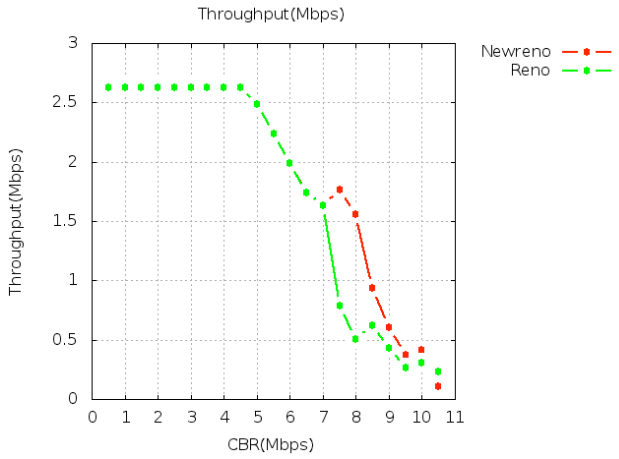

Fig. 3.7 Throughput of New Reno TCP and Reno TCP

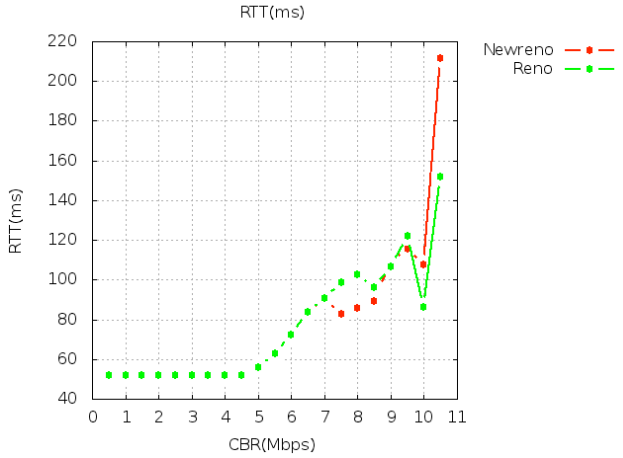

Fig. 3.8 RTT of New Reno and Reno

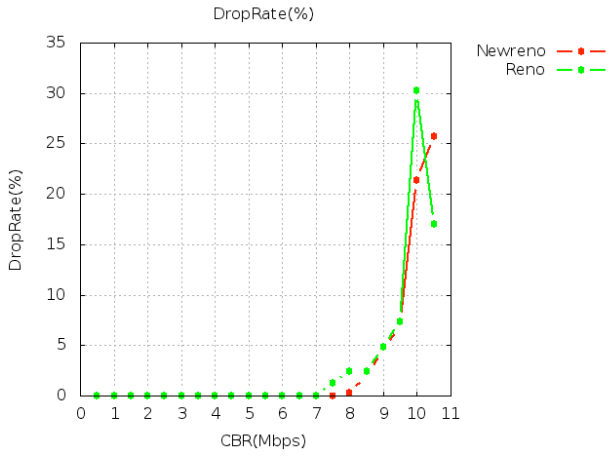

Fig. 3.9 Drop rate of New Reno and Reno

Fig. 3.7, 3.8, and 3.9 shows the throughput, RTT and drop rate of New Reno TCP and Reno TCP of same network respectively. As the figure shows that, when CBR is small, there is no congestion in the network so each TCP flow has maximized throughput and lowest RTT and drop rate. As CBR increases, two variants start to compete for bandwidth, we can see clearly that when CBR is 8Mbps, the throughput of New Reno is larger than that of Reno, RTT and drop rate of New Reno is also smaller than those of Reno TCP. So we conclude that when New Reno TCP and Reno TCP is deployed in one network together, New Reno will have a better over all performance, we think the congestion control approach New Reno applied is the reason for this. Because New Reno performs retransmission when each duplicate ACK is received, while Reno retransmits after three duplicate ACKs are received, so New Reno is more likely to outperform Reno in the same condition.

3.2.3 Vegas/Vegas

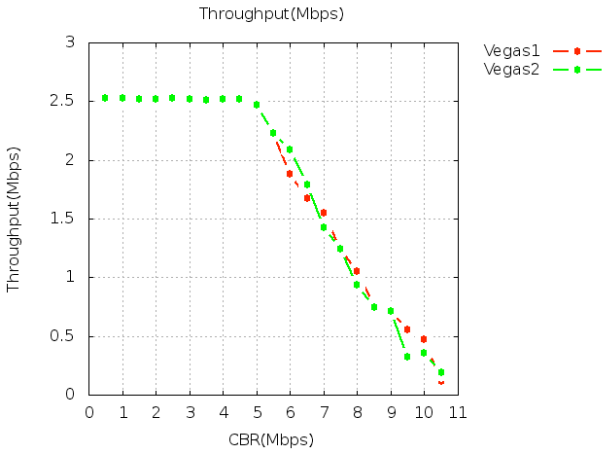

Fig. 3.10 Throughput of two Vegas TCP

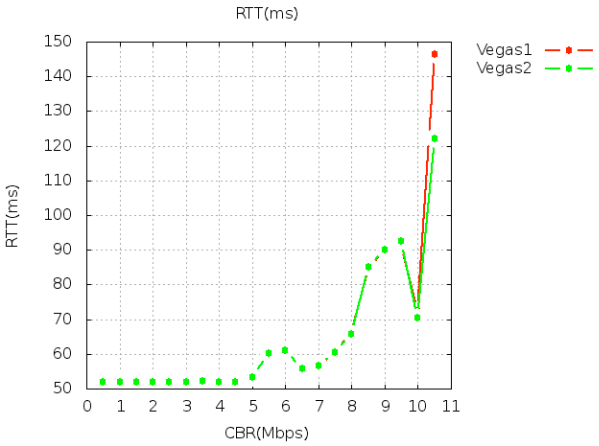

Fig. 3.11 RTT of two Vegas TCP

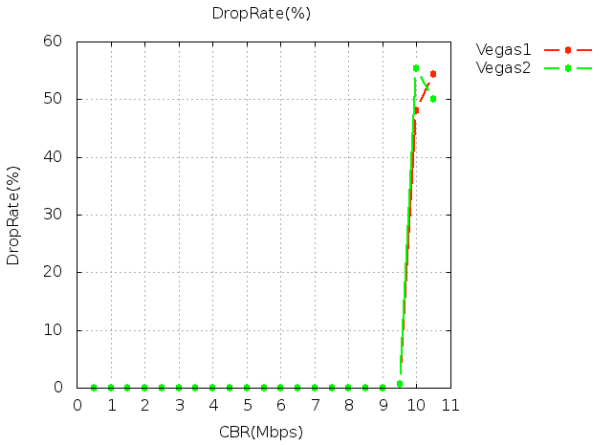

Fig. 3.12 Drop rate of two Vegas TCP

Fig. 3.10, 3.11, and 3.12 shows the throughput, RTT and drop rate of two Vegas TCP in the same network respectively. We can see from this group of comparison that two Vegas act similarly when network condition becomes worse, so we consider that two Vegas TCP in the same network will be fair to each other and have similar performance.

3.2.3 New Reno/Vegas

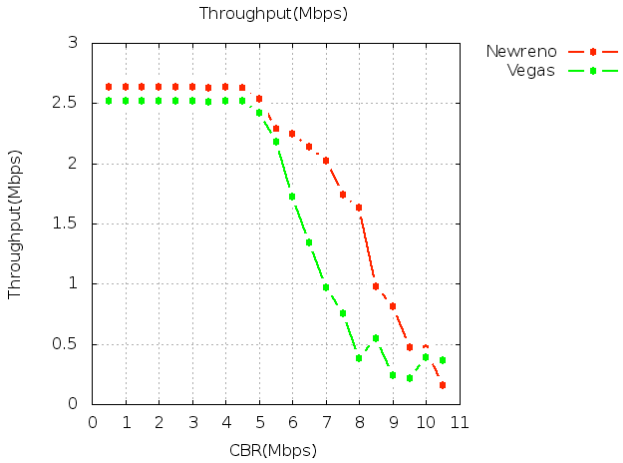

Fig. 3.13 Throughput of New Reno and Vegas.

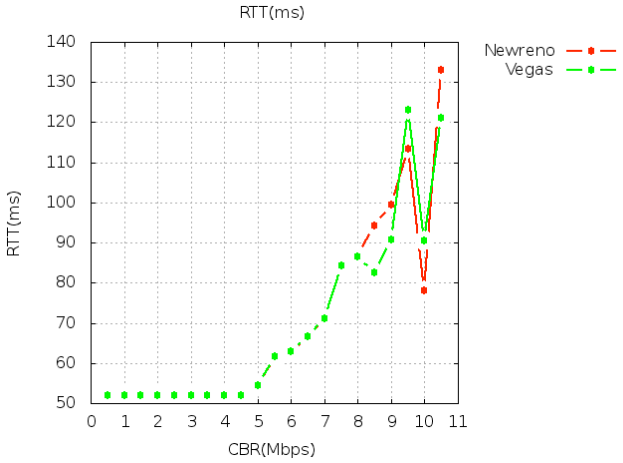

Fig. 3.14 RTT of New Reno and Vegas.

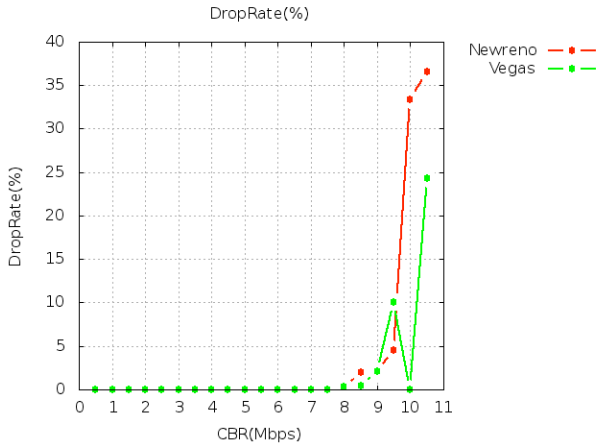

Fig. 3.15 Drop rate of New Reno and Vegas.

Fig. 3.13, 3.14 and 3.15 shows throughput, RTT and drop rate of New Reno TCP and Vegas TCP respectively. We can see from these figures that New Reno will have larger throughput under congestion, the RTT figure shows that two TCP take turns to outperform the other, while when comes to drop rate New Reno has relatively greater value. This can be explained by the fact that Vegas used delay based congestion control algorithm, so that it can detect congestion earlier than New Reno and ensure low drop rate. Therefore we conclude that when deployed in the same network New Reno and Vegas is not fair to each other, New Reno has better throughput while Vegas has lower drop rate.

3.3 Influence of Queuing

In this experiment we study the influence of two queuing disciplines, DropTail and RED (Random Early Drop) on the performance of TCP flow and CBR flow. We use the same topology as shown in Fig. 2.1, we set the TCP source at node N0, sink at node N3 and CBR source at node N4 and receiver at node N5. We start the TCP flow first, when it is steady then we start the CBR flow. To let CBR flow cause pressure to TCP flow we set CBR rate as 8Mbps. To pick a right time point of steady state, we conduct the experiment one again and plot the throughput of TCP variants overtime.

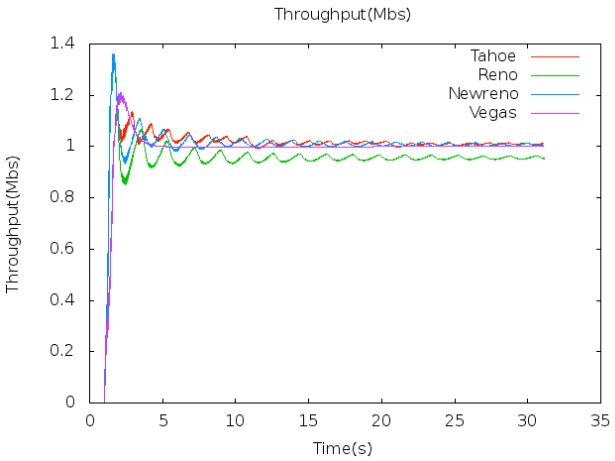

Fig. 3.16 TCP flow steady state test.

Fig. 3.16 shows that when CBR flow is 9Mbps, the throughput of four TCP variants become steady after about 15 seconds and oscillating around 1Mbps. So we start the CBR flow at 15 seconds in the following experiments.

3.3.1 Reno

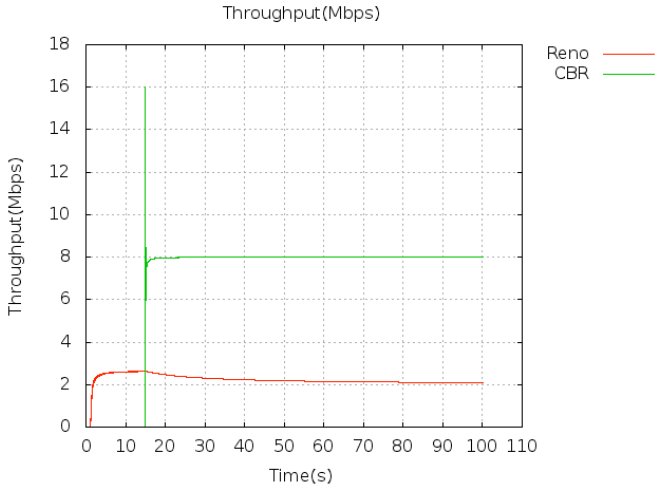

Fig. 3.17 Throughput of Reno and CBR with DropTail.

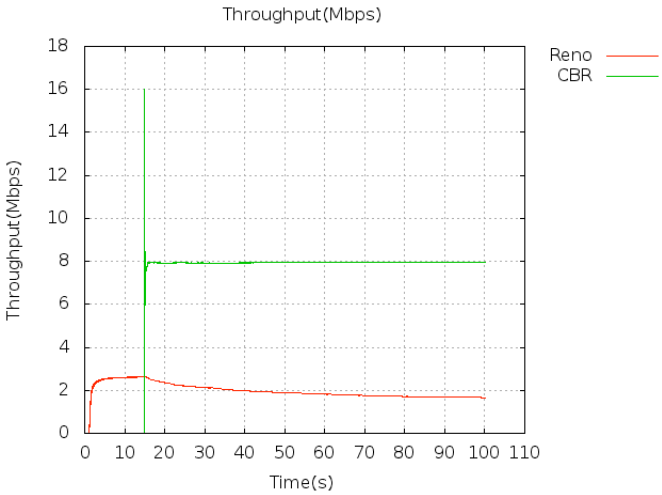

Fig 3.18 Throughput of Reno and CBR with RED.

Figure 3.17 and 3.18 shows the throughput of Reno and CBR with DropTail and RED queuing discipline respectively. We can see that two queuing disciplines have different throughput on each flow, though not quite obvious. TCP flow will become steady after 15 seconds and throughput is around 2.6Mbps, when the CBR flow is open it will cause TCP flow throughput to drop. Also we can see that with RED the TCP flow will recover slower than with DropTail and has smaller throughput. CBR flow also has slightly smaller throughput during 20 seconds to 40 seconds.

3.3.2 SACK

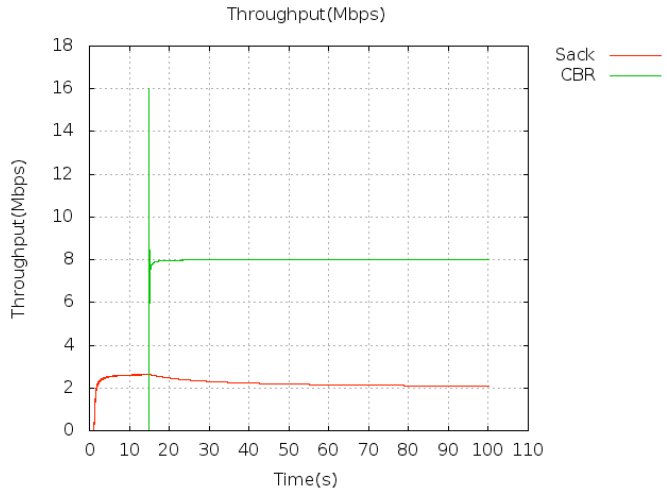

Fig. 3.19 Throughput of SACK and CBR with DropTail.

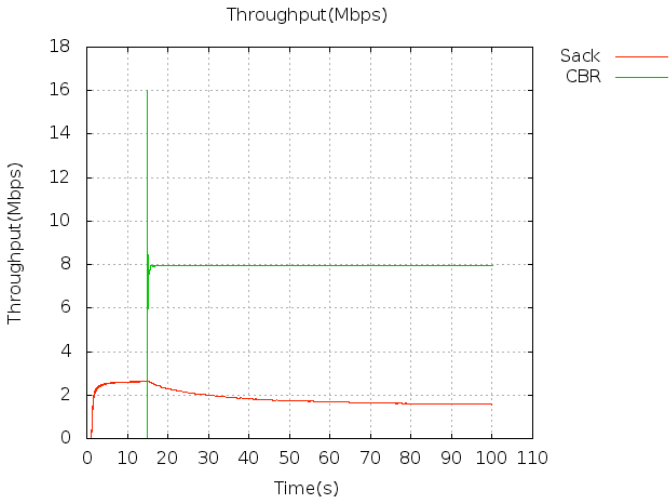

Fig. 3.20 Throughput of SACK and CBR with RED.

Fig. 3.19 and 3.20 shows the throughput of SACK and CBR with queuing discipline DropTail and RED respectively. Similar to Fig. 3.17 and 3.18, the presence of CBR flow will cause TCP flow to drop, and with RED the TCP flow need longer time to recover and has a smaller throughput. But different from results of Reno, the CBR flow throughput does not have slightly drop during 20 seconds to 40 seconds, so we conclude that SACK can recover quicker from randomly drop queuing discipline so it has small impact on CBR flow, therefore it can deal with RED better. Also based on the throughput changes between DropTail and RED we conclude that RED will have larger latency.

4 Conclusion

In our experiments, we study the performance of TCP variants under different network conditions.

In first experiment we showed that as the rate of CBR flow increases, throughput of TCP flow will drop and latency and drop rate of TCP flow will increase accordingly, also New Reno will have a better overall throughput and Vegas have a smaller overall RTT and drop rate. We conclude the overall "best" variant depends on the statistics we investigate and experiment conditions.

In the second experiment we compared four groups of variants, each comparison we investigate the throughput, RTT and drop rate of two TCP. And we find that different TCP don't act fairly to each other when deployed in the same network, such as New Reno will have higher throughput than Vegas and Vegas has lower RTT and drop rate.

In the last experiment we study the influence of queuing discipline DropTail and RED on TCP flow and CBR flow. We showed that DropTail can provide slightly higher throughput than RED thus the latency is accordingly smaller. By comparing the throughput of CBR flow we conclude that SACK can deal with RED better than Reno.