Performance Analysis of TCP Variants

Yang Huang, Gang Liu

Northeastern University

Feburary 2014

1 Introduction

Research on the performances of different Transmission Control Protocol (TCP) variants raises great interest in nowadays, since it is important to understand how TCP variants will affect today's large, congested network. Our experiments focus on the following three circumstances:

1. TCP Performance Under Congestion

   By establishing a TCP connection over a Constant Bit Rate (CBR) Flow and making the network towards congestion, we can tell the difference in throughput, latency and drop rate.

2. Performance of Fairness Between TCP Variants

   By sharing a common route with two TCP flows, we can analyze the fairness of different TCP variants via the comparison of throughput, latency and drop rate.

3. Influence of Different Queuing Policies

   By letting a CBR flow impacts on a steady TCP flow under two kinds of queuing policies, we can compare the different effects over throughput and latency between the two queuing policies.

We found out that TCP Vegas is better variant when comparing with individual performance because of its congestion avoidance algorithm. DropTail queuing policy is better than Random Early Detection policy in throughput performance while Random Early Detection policy is better in latency.

2 Methodology

2.1 Analysis Tool

Our experiments are based on Network Simulator (NS-2) which is a discrete event simulator targeted at networking research. NS-2 provides substantial support for simulation of TCP over wired network, which is set in this paper.

We also choose Python, a scripting language with easy I/O interaction and parsing, to filter out the data from the trace files NS-2 dumped and to measure the corresponding performance parameters.

2.2 Topology

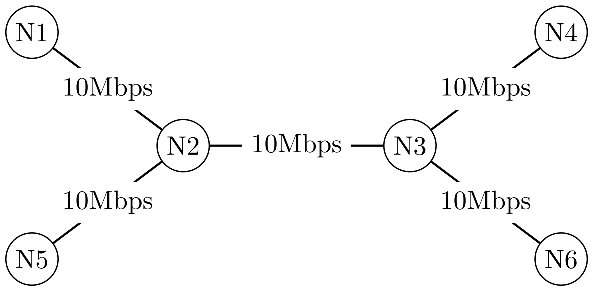


Figure 2.2 Simulation Topology

Figure 2.2 shows the topology we use in the experiments. Each edge represents a duplex link with a bandwidth of 10Mbps.

2.3 Experiment Parameters

Section 3 discusses TCP performance under congestion by establishing a TCP stream from N1 to N4 and a CBR flow from N2 to N3. The TCP stream has a window with 200 bytes, 1500 bytes packet size, while the CBR flow uses a 500 bytes packet size. We will increase the CBR flow rate from 1Mbps to 10Mbps, each rate conducts one run.

Section 4 is the experiment of fairness between TCP variants with using the same topology settings. But this time we add one more TCP flow from N5 to N6. The experiment is also run by changing the CBR flow rate from 1Mbps to 10Mbps, except the TCP variants is using the following pairs (One for N1-N4, the other for N5-N6): Reno/Reno, NewReno/Reno, Vegas/Vegas and NewReno/Vegas.

Section 5 shows the influence of TCP performance by applying different queuing policies to the links. By setting one TCP flow from (N1-N4) and one CRB flow (N5-N6), we will measure the comparison over time span instead of CBR flow rates, and only use with TCP Reno and SACK variants.

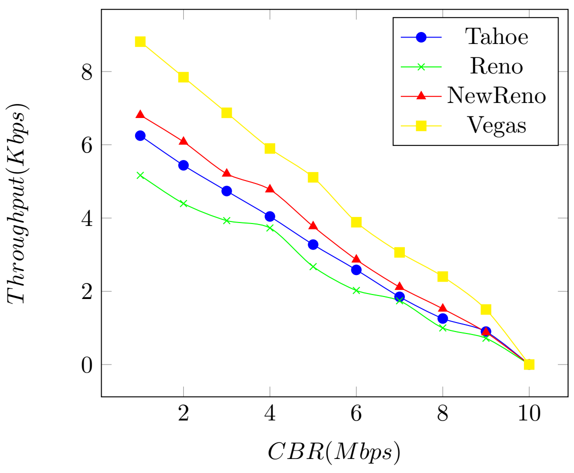
3 Performance Under Congestion

Congested network will result in packet drops and high latency. However, with different TCP variants, the TCP stream will react differently to the congestion situation. In this section, we will compare the average throughput, latency and drop rate among TCP Tahoe, Reno, New Reno and Vegas.

3.1 Average Throughput

Average Throughput is calculated by: the total packets count received by N4 over the total running time. Here is the result of average throughput comparing the four variants.

Figure 3.1. Throughput of TCP Variants



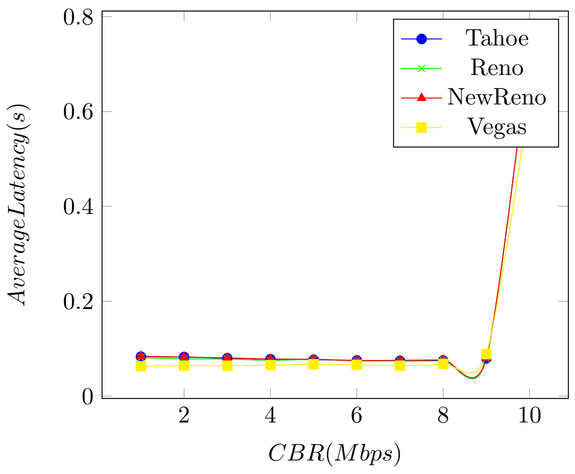
Vegas keeps a higher throughput than all the other variants and can keep to reach the maximum bandwidth due to its congestion avoidance algorithm. TCP Tahoe, Reno and New Reno have to waste some bandwidth because of their lazy congestion detection.

New Reno has a higher throughput among the three because it has an aggressive congestion detection algorithm with one duplicate ACK packet, while Reno is the worst because it will retransmit after three duplicate ACK packets. Tahoe will just retransmit after detecting a packet dropped.

3.2 Average Latency

We analyze the average latency based on Round Trip Time (RTT), which is the time span between sending a packet and receiving the acknowledgement of that packet.

Figure 3.2. Average Latency of TCP Variants



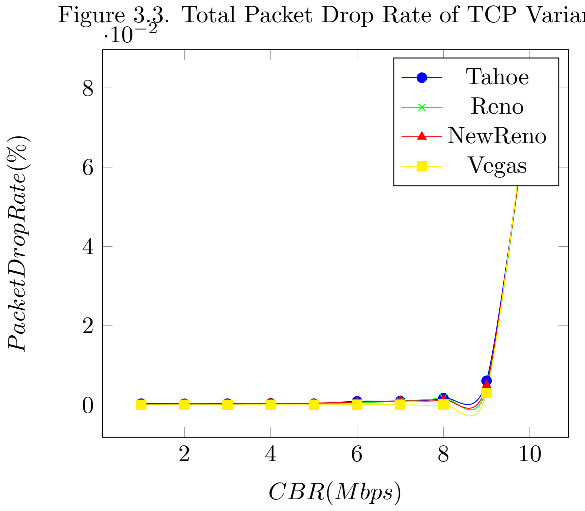
Vegas outperform other variants in this case, it has the lowest latency. TCP Vegas detects congestion in advance based on the RTT instead of an actual packet drop, therefore Vegas will send fewer packets to the network, causing less average latency.

For TCP Tahoe, Reno and New Reno, the average latency is quite similar under 8Mpbs since they will try send as much packets as possible to the network until they detect congestion.

3.3 Packet Drop Rate

With the lower drop rate, we can tell which variant can maximize the resources consumption in the network better. The drop rate is calculated by: the total number of dropped packets over the total number of sent packets.

Figure 3.3. Total Packet Drop Rate of TCP Variants



With all the rates we experimented, TCP Vegas remains a relative low drop rate. Since Vegas has a congestion avoidance algorithm based on the RTT, it will reduce the sending rate when RTT is high. But for TCP Tahoe, Reno and New Reno, they all have some packet drops because of their congestion detection is slower.

3.4 Summary

In overall, TCP Vegas reacts a relatively better performance than other variants due to its congestion avoidance algorithm.

4 Fairness Between TCP Variants

The four combinations, Reno/Reno, New Reno/Reno, Vegas/Vegas and New Reno/Vegas will be implemented in this experiment. We add another TCP flow from N5 to N6 to compete with the one on N1 and N4.

4.1 Reno vs. Reno

Figure 4.1.1, 4.1.2 and 4.1.3 shows the situation of two TCP flows running in Reno variant competing for the resource in one single channel.

Figure 4.1.1 Throughput with competition

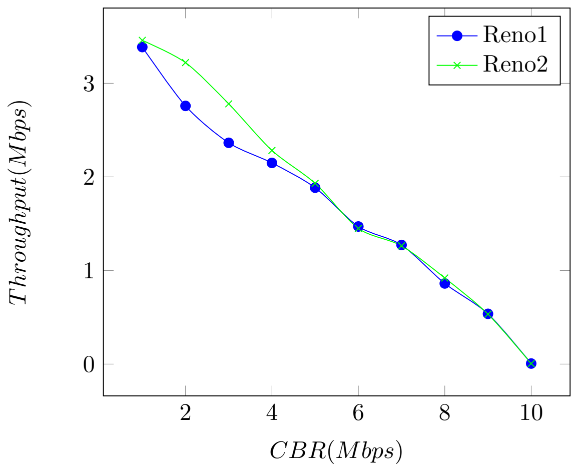


Figure 4.1.2 Latency with competition

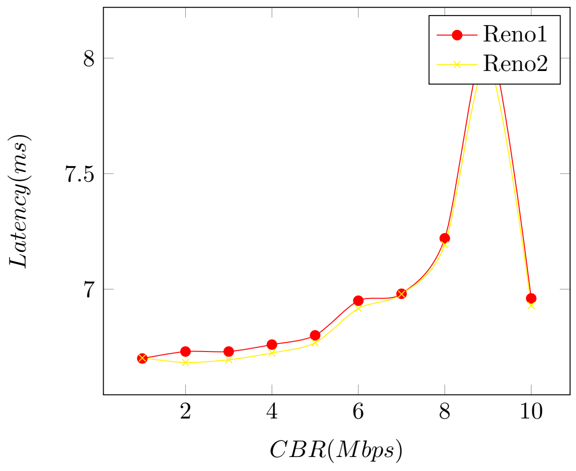
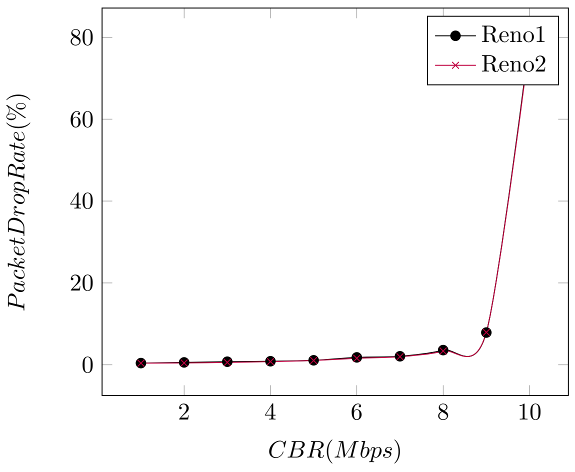


Figure 4.1.3 Packet Drop Rate with competition



From all the figures we can see their performance varies nearly none. Overall they are overlapping with each other. Therefore in this case, both TCP Reno act fairly and share the network bandwidth equally.

4.2 New Reno vs. Reno

For TCP New Reno per se, it is a newer version of TCP Reno. Both of them use “Slow Start”, but New Reno will retransmit after one duplicate ACK while Reno will do after three. After detecting potential dropped packet, New Reno will reduce its window size to ssthresh which is deemed as ”Fast Recovery” in order to avoid ”Slow Start”.

Figure 4.2.1, 4.2.2, 4.2.3 shows the situation of one New Reno variant competing with one Reno variant over one single channel.

Figure 4.2.1 Throughput with competition

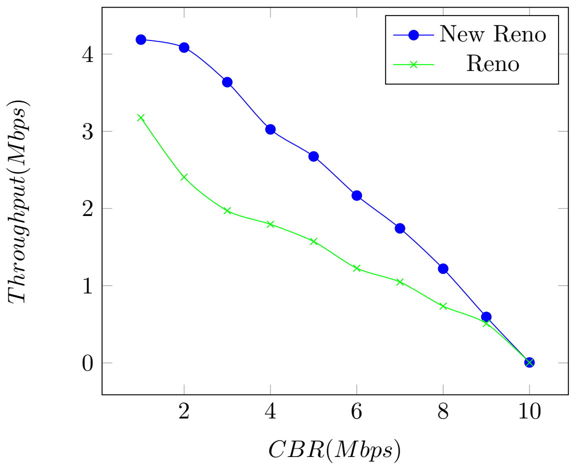


Figure 4.2.2 Latency with competition

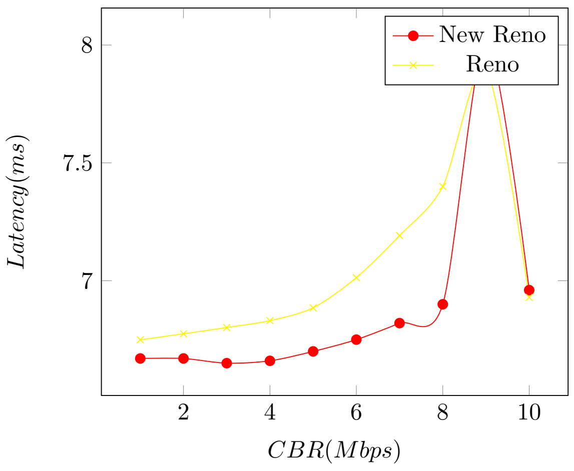
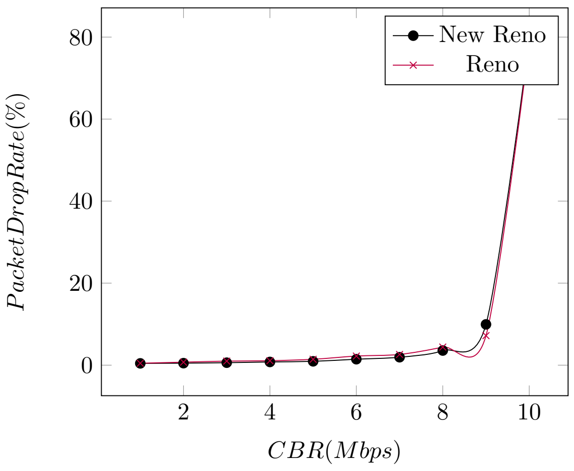


Figure 4.2.3 Packet Drop Rate with competition



From Figure 4.2.1, TCP New Reno has a much better throughput than TCP Reno and in Figure 4.2.2, New Reno has lower latency over TCP Reno. While both of the packet drop rates in Figure 4.2.3 remain almost the same, because both of them detect potential dropped packets in advance. But overall, New Reno will do quicker "Slow Down" and quicker "Speed Up" than Reno, therefore New Reno has a higher throughput than Reno, as well as a lower latency.

Generally, when TCP New Reno is competing with TCP Reno, New Reno will gain an advantage of better performance.

4.3 Vegas vs. Vegas

TCP Vegas generally will decrease its sending rate when it detects an increasing RTT when compared with the other variants.

Figure 4.3.1, 4.3.2, 4.3.3 shows the situation of two TCP flows running in Vegas variant competing for the resource in one single channel.

Figure 4.3.1 Throughput with competition

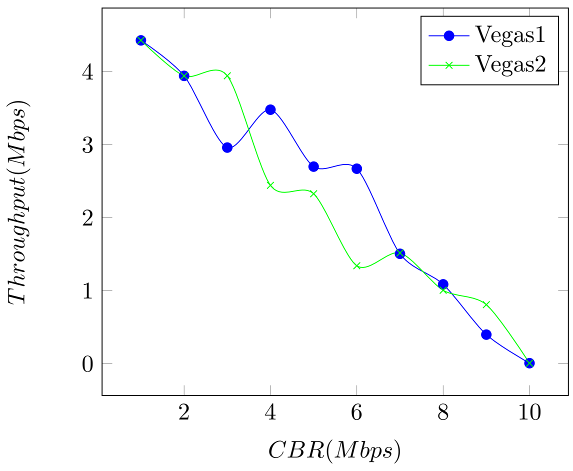


Figure 4.3.2 Latency with competition

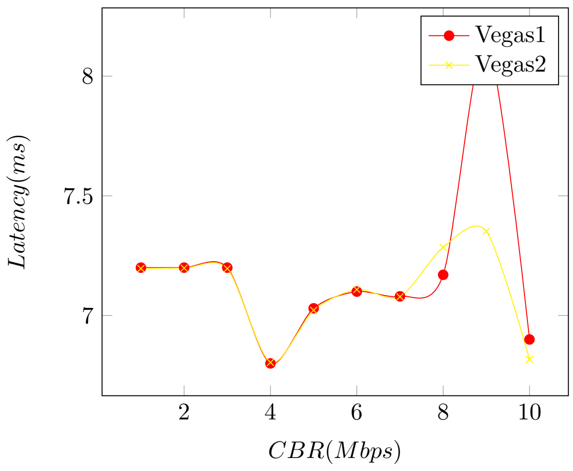
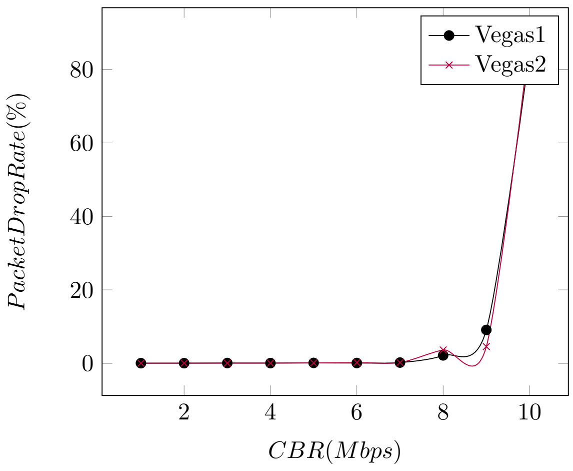


Figure 4.3.3 Packet Drop Rate with competition



When one Vegas increases the sending rate, the other may detect an increasing RTT and thus decrease the sending rate. That's why Figure 4.3.1 is oscillating. When the CBR flow rate is bigger than 7Mbps, the two cannot share the bandwidth quite well.

Overall, both Vegas compete fairly in the experiment. But due to the congestion avoidance algorithm, both variants are oscillating in the result.

4.4 New Reno vs. Vegas

As mentioned in the previous sub-section, Vegas has a better congestion avoidance algorithm than New Reno.

Figure 4.4.1, 4.4.2, 4.4.3 shows the situation of one New Reno variant competing with one Vegas variant over one single channel.

Figure 4.4.1 Throughput with competition

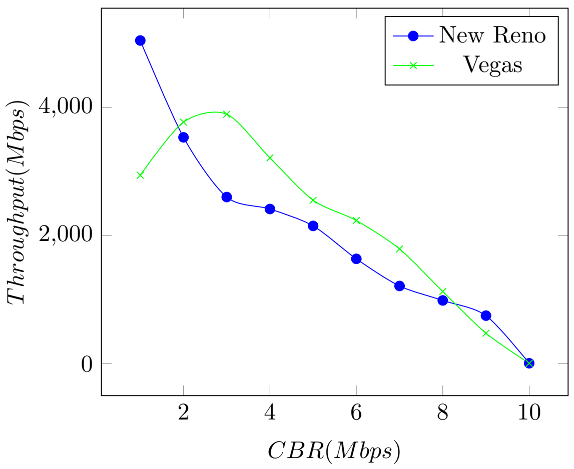


Figure 4.4.2 Latency with competition

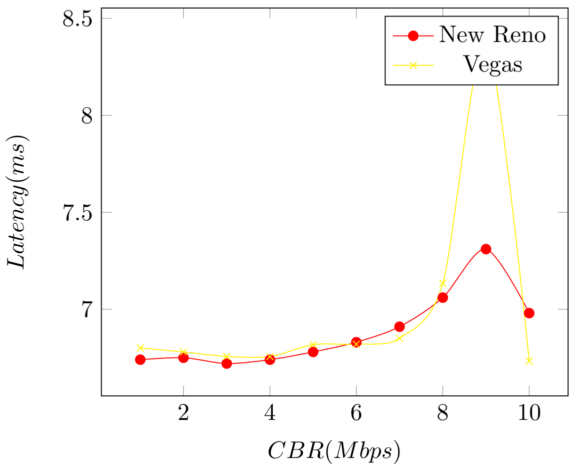
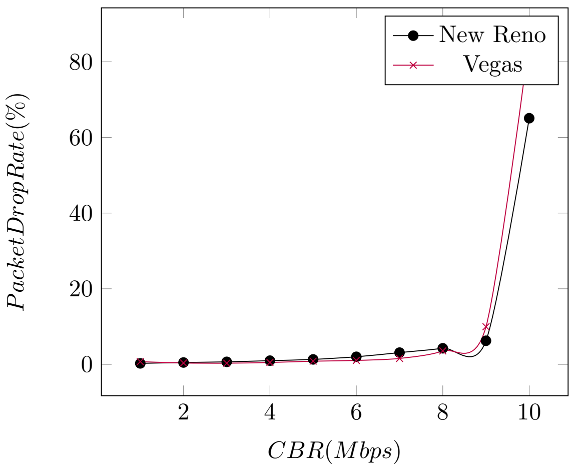


Figure 4.4.3 Packet Drop Rate with competition



In Figure 4.4.1, New Reno gains a better throughput in the beginning which is because New Reno will first try to keep sending packets until a RTO happens, while Vegas detects the RTT is increasing, it will slow down a little bit. But when New Reno starts to restore to "Slow Start", Vegas then increase its packets sending rate due to the decreasing RTT. But in the end New Reno just dominate again because Vegas is relatively conservative. However, TCP New Reno and TCP Vegas have similar packet drop rate and round trip latency as we can tell from Figure 4.4.1 and 4.4.2.

In conclusion, New Reno dominates Vegas when it comes to throughput competition because Vegas detects potential congestion and will act conservatively. So in this case, New Reno is taking advantage of Vegas.

5 Influence of Queuing

In this section, we will implement the experiment by varying queuing policies: DropTail and Random Early Detection (RED) and to see the fairness and difference in average end-to-end latency for both flows.

DropTail simply drop the last packet of the buffer when the queue is full, while RED, also known as Random Early Drop, maintains an estimated average queue length by randomly dropping packets.

5.1 Average Throughput

Figure 5.1.1 Throughput in TCP Reno

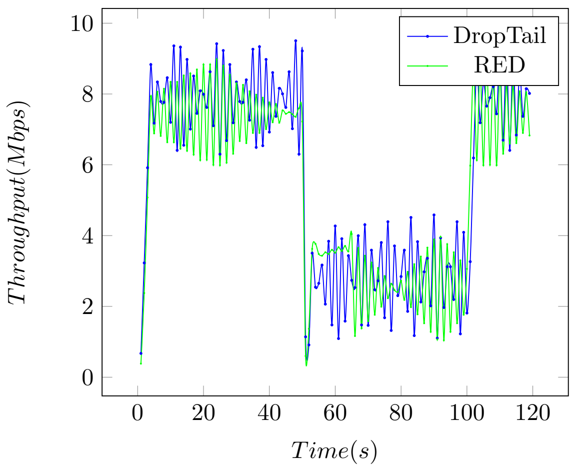
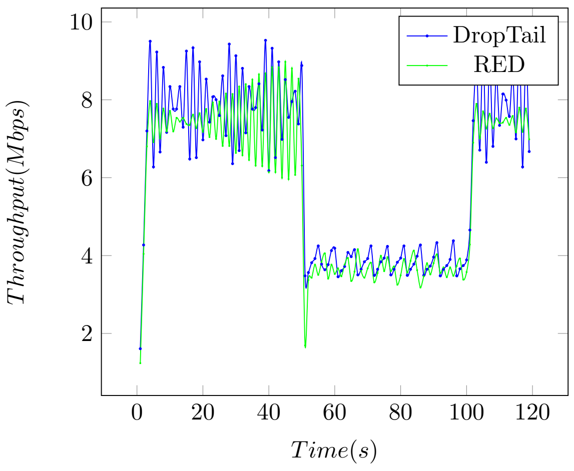


Figure 5.1.2 Throughput in TCP Sack



Because RED is randomly dropping packets, the throughputs in either TCP Reno or TCP Sack of RED are relatively lower than the ones in DropTail. But with TCP Sack implementing in RED policy, the throughput is quite similar to the one in DropTail.

When CBR flow comes in, TCP flow's throughputs drop at half in both situations in TCP Reno and TCP Sack. But Sack is Seletive ACK, it remains more steady throughput than in TCP Reno (Figure 5.1.1 and 5.1.2).

5.2 Average Latency

Figure 5.2.1 Latency in TCP Reno

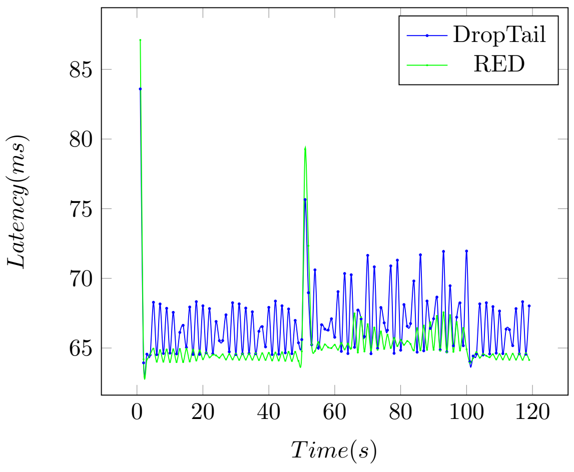
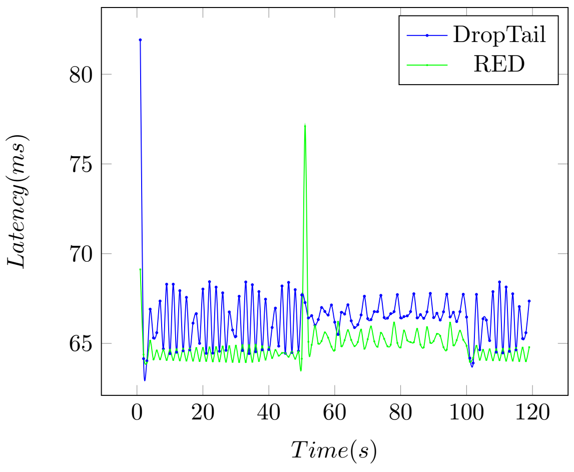


Figure 5.2.2 Latency in TCP Sack



However when it comes to latency, we can see that RED will keep fewer packets in the network than DropTail. With this feature, the congestion situation is better in RED than in DropTail. Both average latency performance in RED in TCP Reno and TCP Sack are lower.

When CBR flow comes in, the overall latency during that time goes higher. TCP Sack has more steady situation than TCP Reno (Figure 5.2.1 and 5.2.2).

In conclusion, DropTail provides fairly throughput either in TCP Reno or TCP Sack, but RED provides TCP Sack a better throughput performance than TCP Reno. RED has lower latency problems than DropTail.

6 Conclusion

In this paper, we have explored the performance of different TCP variants, the fairness between different TCP variants and the influence of different queuing policies. Among the four variants, TCP Vegas seems to be the better one in performance comparison. But for competition, New Reno and Vegas are the top two variants. New Reno is aggressive in throughput, but gained higher latency, while Vegas is conservative in throughput, but gained lower latency. Which should be deployed in our real-world situation really should depends on the attitudes of higher network throughputs or lower network latency.

For in the future study, we need to experiment more on New Reno and Vegas performance, not only because the algorithms are different, but also the network in the paper is based on Ethernet. If in other networks or internetworks, their performance will surely differ from the one we conducted.