Performance Analysis of TCP Variants

Zetao Wang, Yidong Wang
Northeastern University
360 Huntington Ave., Boston, Massachusetts 02115
wang.zet@husky.neu.edu, wang.yid@husky.neu.edu

1 Introduction

The original design of the Transmission Control Protocol (TCP) worked reliably, but was unable to provide acceptable performance in large and congested networks. So several TCP variants were proposed in order to improve the performance of Transmission Control Protocol under congested networks. For example, a TCP variant named Reno improves the performance of TCP by fast retransmit and fast recovery. Each TCP variant have their mechanisms to control and avoid congestion.

We do a research on the contrast the performances of different Transmission Control Protocol under three experiments:

(1) In the first experiment, we set different CBR flow rate and test TCP variants' performance.
(2) In the second experiment, we test the allocation of a bandwidth while running 2 different TCP variants on it.
(3) In the third experiment, we test the influence of the queuing discipline used by nodes on the overall throughput of flows.

In this paper, the methodology of the three experiments would be proposed in section2. We will talk about the above three experiments though section 3, 4 and 5. In section 6, we will make a summary of our work.

2 Methodology

2.1Conduction of Experiments

2.1.1conduction of the first experiment

By setting CBR rate from 1 to 10Mbps, we will compute throughput, latency and the drop rate of Reno, New Reno, Tahoe and Vegas in order to understand which TCP variant performs better.

2.1.2conduction of the second experiment

We compare different combination of TCP variants and let them compete with each other to see whether different combinations of variants fair to each other

2.1.3conduction of the third experiment

By setting queue type, we will compute throughput, latency, lost packets numbers to test the performance of the TCP variants and CBR flow over time.

2.1.4formula

Throughput = TransferSize / TransferTime
TransferTime = TCPSEndtime - TCPStarttime

PacketLosses = PacketSend - PacketReceived
PacketLossesRatio = PacketLosses / PacketReceived

Latency = TotalLatency / PacketReceived
TotalLatency = Sum (PacketReceivedTime - PacketSentTime)

2.2 Analysis Tool

These experiments were conducted using NS-2, an event driven network simulator, which consist of event scheduler and network component.

We use python to filter data in .tr files compute performance parameters needed for measure, using formulas above.

And in order to get a precise result, we also use bash to run .tcl and .py files several time. So we can get a result with less oscillation.

3 TCP Performance Under Congestion

3.1 Topology of Experiment 1

Experiment 1 is set up in the topology in Figure 3-1. A constant CBR source is added at N2 and sink at N3. A single TCP stream is added from N1 to N4. And the TCP stream varies from Tahoe, Reno, New Reno and Vegas

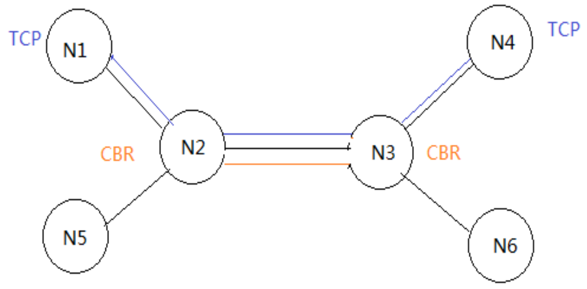


Figure 3.1.1 topology of experiment 1

In the experiment, the CBR flow rate was set varying from 1Mbps to 10Mbps, to change the congestion condition in the network. Under each constant CBR flow rate, the throughput, latency and drop rate of each TCP flow would be recorded. By analyzing the experiment result, we could make a comparison of how the four TCP variant react to the presence of congestion.

3.2 Average Throughput

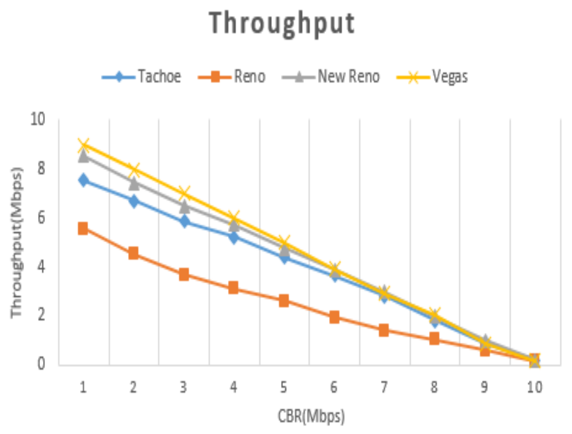


Figure 3.2.1 Throughputs

We calculate Average Throughput by dividing total packets received at N4 by the total running time. Figure 3.2.1 is the result of average throughput (CBR change from 1~10 Mbps) contrast between Tahoe, Reno, New Reno and Vegas.

And we could tell from figure 3.2.1, with the CBR flow rate increasing from 1~10Mbps, the throughput of all four kinds of TCP flows would decrease. Because the increasing CBR flow rate aggravated the congestion in the network, TCP flows under all the four TCP variants' throughput decreased in this situation.

But, under different TCP variants, the throughputs of different TCP are different: Vegas get the highest average throughput. New Reno gets the second highest average throughput and then Tahoe keeps the third highest average throughput. Reno has the lowest average throughput. Because the TCP Vegas uses a congestion avoidance algorithm, which emphasizes packet delay rather than packet loss, it controls its congestion window by detecting the RTT. So the size of congestion window is more adaptable to the latest network status, that's why TCP Vegas could have higher throughput in the congestion situation than others. New Reno is modified from Reno. Its fast retransmit performs much better than Reno: Each duplicate ACK triggers a retransmission, but in Reno, three duplicate ACK triggers a retransmission. So New Reno performed better than Reno. Tahoe will just retransmit after detecting a packet dropped.

3.3 Average Latency

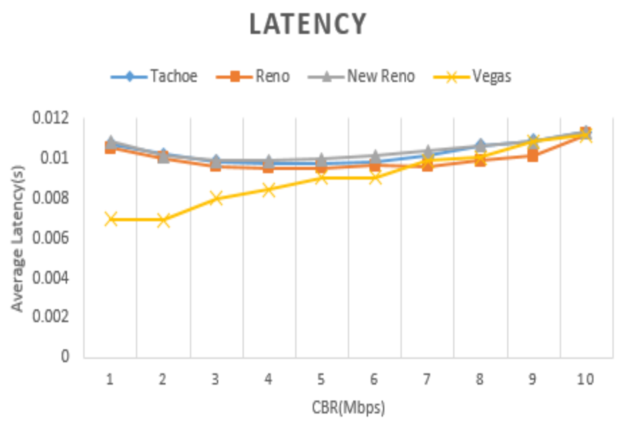


Figure 3.2.2 Latency

Vegas has lowest latency of TCP Flows than other variants in this case, 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 when the network is under congestion, causing less average latency. The TCP flows of Tahoe, Reno and New Reno have similar latency. Because Vegas have efficient congestion window control algorithm, TCP Vegas could detect and avoid the congestion more easily. So Latency of Vegas variant is shorter than Reno, New Reno and Tahoe.

3.4 Drop Rate of TCP Flows

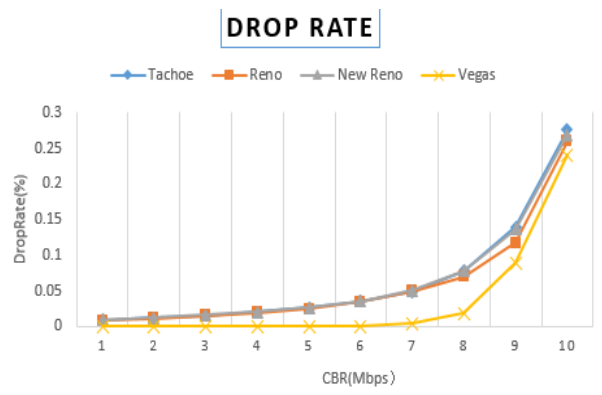


Figure 3.4.1 Drop Rate

We can see from the figure 3.4 that the higher the CBR Flow is, the higher the drop rate is. It is obvious that high CBR flows means more congestion the network could have. So the drop rate increases when the network gets more congestive.

When the CBR from 1Mbps to 7 Mbps, we can see that Vegas's drop rate is much lower than the other three kinds of TCP variants. Reno, New Reno and Tahoe have similar drop rates. And when CBR flow from 7 Mbps to 9Mbps, all the drop rates grows fast but Vegas still has the lowest drop rate, and other three variants still have similar drop rates. When CBR flow grows from 9 to 10Mbps, the drop rates under all of the variants grow rapidly and all of the variants drop rate become similar. Because Tahoe, Reno and New Reno are using Additive Increase Multiplicative Decrease algorithm to control the congestion window and they use detecting dropped packets to change their congestion window. On the other hand, TCP Vegas use detecting RTT rather than dropped packets to control its congestion window. It can detect the congestion earlier than Tahoe, Reno and New Reno, and then take action to avoid the congestion. That's why its drop rate is the smallest.

3.5 Summary

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

4 Fairness Between TCP Variants

4.1 topology of experiment 2

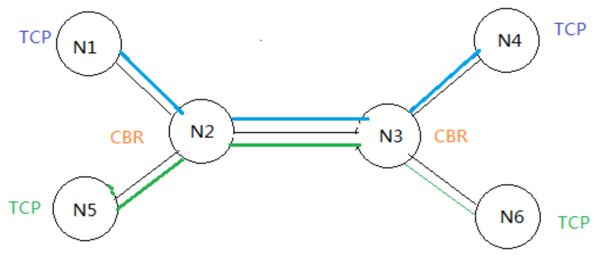


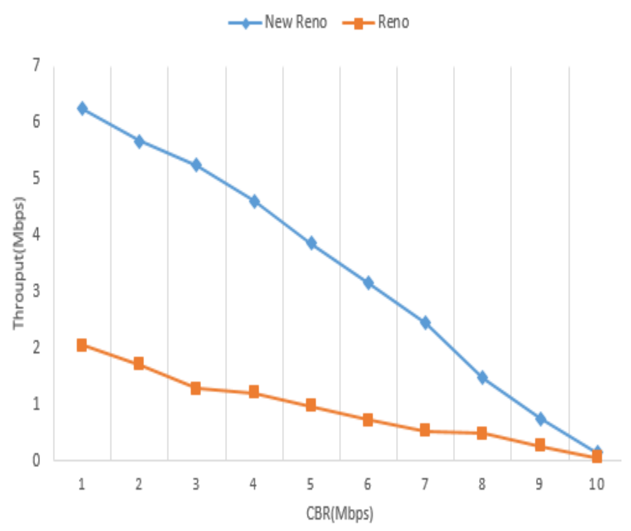
Figure 4.1.1 topology of experiment 2

We compare the four combinations of variants: Reno/Reno, New Reno/Reno, Vegas/Vegas and New Reno/Vegas .We add a TCP flow from N1 to N4 and add another TCP flow from N5 to N6 to compete the one on N1 and N4.

4.2 Reno vs. Reno

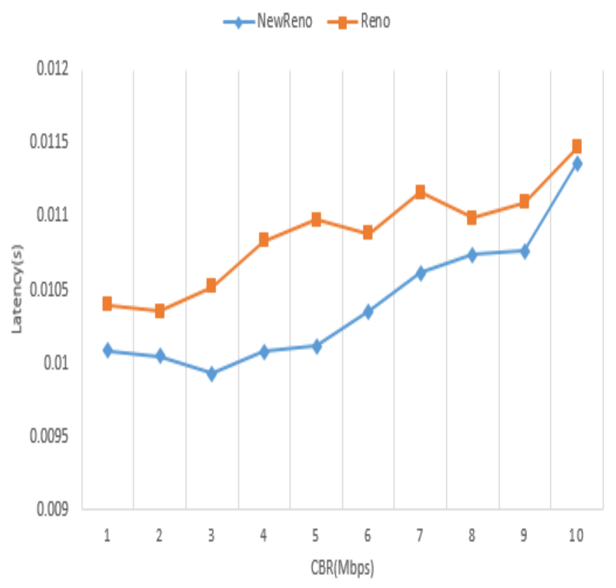
From this comparison, we can see that their performances are almost the same in throughput, latency and drop rate. Overall they are overlapping with each other. Therefore in this case, both TCP Reno act fairly and share the network bandwidth equally. Because they are the same variants, they have the same functionalities: they share the network bandwidth equally between each other. Reno and Reno flows are fair to each other.

4.3 New Reno vs. Reno



4.3.1 Throughput with competition

From figure 4.3, we can see that New Reno's throughput is larger than Reno until the CBR become 10Mbps. Both of them begin as a slow start, and have fast retransmit and fast recovery, but New Reno will retransmit after one duplicate ACK while Reno will retransmit after received three duplicate ACK, and New Reno will reduce its window size to ssthresh in order to avoid slow start from one byte. So we can see New Reno recover much fast than Reno. And the throughput is much bigger than Reno. And because of the New Reno retransmit faster than Reno, so at first the Latency is much lower than Reno, but when the CBR arrive a threshold, the Latency of New Reno and Reno become almost the same.

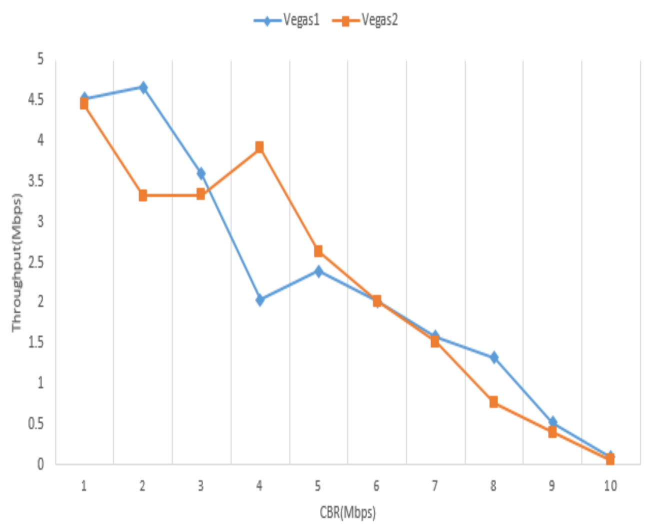


4.3.2 Latency with competition

And when we compare the Drop Rate of these two variants, they are almost the same.

Generally, when TCP New Reno is competing with TCP Reno, Reno and New Reno are not fair to each other. New Reno will gain an advantage of better performance.

4.4 Vegas vs. Vegas



4.4.1 Throughput

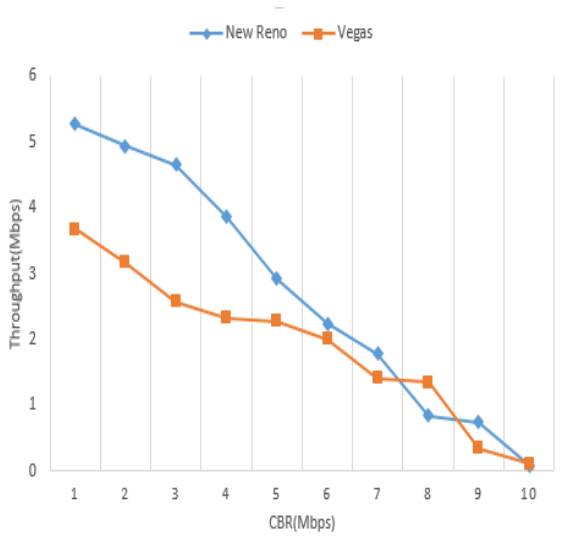
TCP Vegas detects congestion at an incipient stage based on increasing Round-Trip Time (RTT) values of the packets in the connection.

The algorithm depends heavily on accurate calculation of the Base RTT value. If it is too small then throughput of the connection will be less than the bandwidth available while if the value is too large then it will overrun the connection.

When one Vegas increases the sending rate, the other may detect an increasing RTT and thus decrease the sending rate. That's why the throughputs are changing. When the CBR flow rate is bigger than 7Mbps, the two cannot share the bandwidth quite well. And packet loss rate and latency of the two TCP Vegas flows are similar.

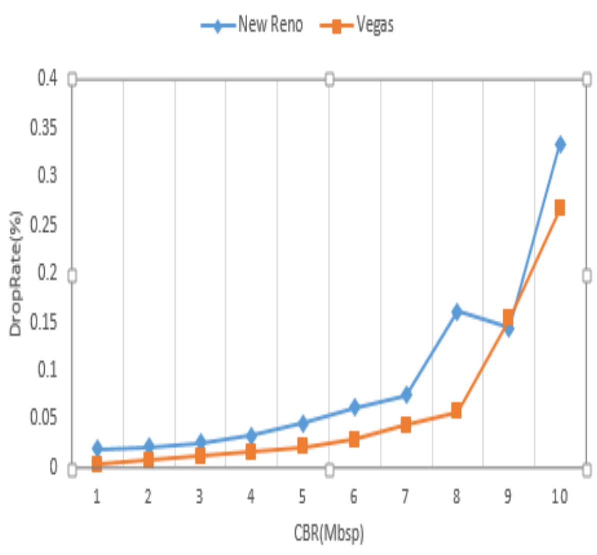
Overall, both Vegas compete fairly in the experiment. But due to the congestion avoidance algorithm, both variants are oscillating in the result.

4.5 New Reno vs. Vegas



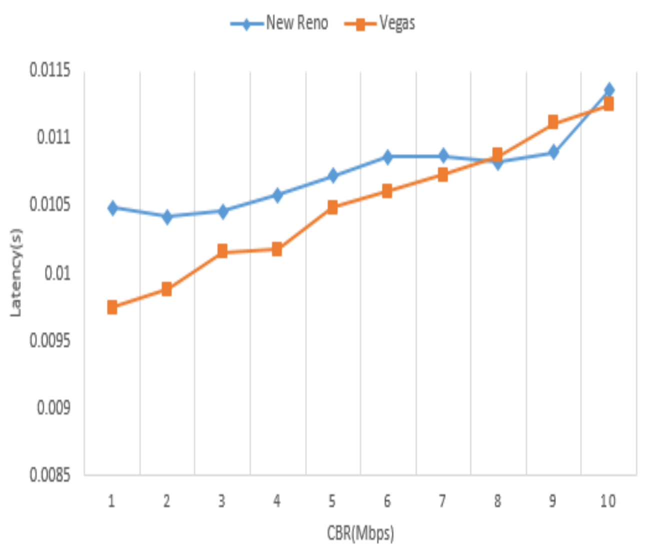
4.5.1 Throughput

We can see from figure 4.5.1 that New Reno gains a better throughput. Because New Reno will keep sending packets until a RTO happens, while the Vegas will detect the congestion by RTT. At first New Reno throughput is huge, so the RTT will be long, so the throughput of Vegas will be low. And when CBR grows from 1 to 10 Mbps, the network will be more congestive, so the throughput of New Reno will be deceased.



4.5.2 Drop Rate

From 4.5.2, the drop rate under the New Reno and Vegas are similar when the CBR flow is not huge. When the CBR flow grows to 9Mbps, drop rate under New Reno increased rapidly but drop rate under Vegas is still stable. Because the Vegas can detect the congestion situation of the network, it can control this throughput in order to decrease the drop rate.



4.5.3 Latency

In conclusion when Vegas is inter-operated with other versions like New Reno. In this case, performance of Vegas degrades because Vegas reduces its sending rate before New Reno as it detects congestion early and hence gives greater bandwidth to co-existing TCP Reno flows.

5 Influence of Queuing

5.1 topology of experiment 3

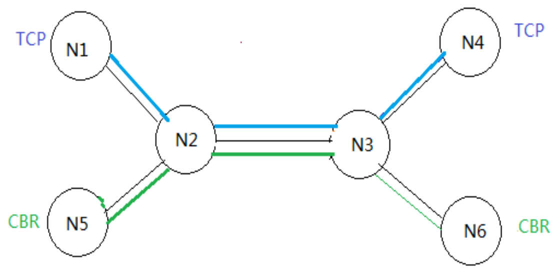
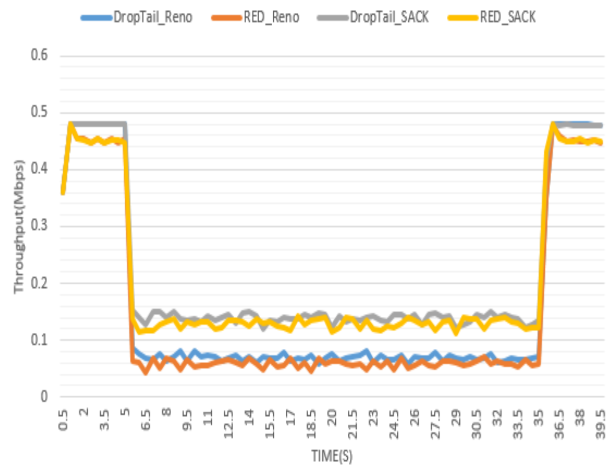


Figure 4.1.1 topology of experiment 3

5.2 Throughput

In this experiment, we adding a TCP flow from N1 to N4, and a CBR/UDP flow from N5 to N6. When the TCP flow stabilized we start CBR flow. Performing the experiments with TCP Reno and SACK under different queue algorithms to see the performance of TCP and CBR flow over time. We let TCP flow starts at 0s, and CBR flow starts at 5s. Give enough time to make TCP flow stable. And then shut down CBR flow at 35s.



5.2.1 Average throughputs of TCP

Theoretically, Drop Tail is a strategy that drops packet when buffer is full, and RED drop packets randomly intending for congestion avoidance. It can drop packets by estimating the congestion of the network.

It used average queue length as the parameter to calculate the discard possibility in order to determine the possibility of dropping certain packet, and smooth its reaction toward congestion.

As shown in figure 5.2.1, when CBR flow start, it grabs bandwidth from TCP flow, and the TCP flow's throughput decrease markedly, CBR flow has no reaction to congestion announcement. That means both of the algorithms cannot provide fair bandwidth to difference flows, such as TCP flows and CBR flows. Both of the queue algorithms treat different flows without difference, so it cannot protect TCP flows from misbehaving flows as the CBR flow in this experiment. And after 35s, when the CBR flow shut down, TCP flow increases again, and return to a normal level.

When compare to the latency, the end-to-end latency for both TCP flows is shorter when using RED as their queuing strategy than that of Drop Tail, as we can see from 5.2.1 the combination of SACK and Drop Tail provides the highest throughput than the other combinations.

6 Conclusion

By analyzing the results stated above, we can come to following conclusion: TCP Vegas has a better performance than the other TCP Variants under the congestion situation: it has high average throughput, low drop rate and low latency than others.

But when there are other TCP Variants compete each other like in experiment 2,the TCP Vegas performed not very good. And different TCP variants don't fair to each other. Because nowadays network has all kinds of TCP variants so Vegas could not have such good performance in the real networking.

Additionally, both RED and Drop Tail do not have a good protection to deal with the misbehaving flows, such like CBR. And Drop Tail performs better than RED in both SACK and Reno. And TCP SACK combined with Drop Tail preform the best, so we should better use the queuing of Drop Tail in the network that most flows are TCP flows.

The contributions of this paper are as follows. Firstly, we analyzed the performance of different TCP variants. And then, in compete with each other, we find whether different variants fair to each other. Thirdly, the performance of different queuing method is analyzed. Further work includes discussing the real world networking.