Performance Analysisof TCP Variants

Zhuoli Liang
Northeastern University
360 Huntington Ave., Boston, MA 02115
liang.zhu@husky.neu.edu

Abstract

TCP provides a reliable data transmission mechanism on a unreliable transmission link layer by using acknowledgment(ACK) and retransmission mechanism. With the increasing of modern bandwidth of network and more congestion occasions, several new TCP protocols were proposed to handle problems such like congestion control and fairness. This paper uses ns-2 simulations to explore the performance between different TCP protocols, first we uses simulations to analysis the congestion performance of each TCP protocol and then analysis the fairness between TCP variants, and at last to see the link queue type's influence on TCP performance, we simulated a network link with queue type DropTail and RED.

1 Introduction

The Transmission Control Protocol(TCP) has been the dominant transport-layer protocol on the Internet for many years. TCP provides reliable byte stream delivery between end-host applications, using a suite of mechanism for connection management, flow control, and error control. A congestion control mechanism was also added to TCP in 1988.

There are two different paradigms for TCP congestion, one is **Congestion Control**, the other is **Congestion Avoidance.** TCP Tahoe, TCP Reno, TCP New Reno implement Congestion Control. Congestion control uses RTT and delay to detect network congestion and then take actions to recovery from network congestion before goes in to congestion collapse. TCP vegas implements Congestion Avoidance. Congestion Avoidance use RTT to infer a upcoming network congestion, and can slow down transmit rate before congestion happens.

Since TCP is self clocked, the Round Trip Time affects its transmission rate, with the increasing of modern bandwidth of network and the more and more long-distance transmitting, the original design of TCP was unable to provide acceptable performance in large and congested networks. So several TCP variants have been proposed since then to handle these problems. The objective of this paper is to analyze the performance of these different TCP variants.

2 TCP Performance Under Congestion

In data networking and queueing theory, network congestion occurs when a link or node is carry so much data that its quality of service deteriorates. Typical effects include queueing delay, packet loss. To avoid congestion collapse, TCP uses a multi-faceted congestion control strategy. For each connection, TCP maintains a congestion window, limiting the total number of unacknowledged packets that may be in transit end-to-end. This is somewhat analogous to TCP's sliding window used for flow control. TCP uses a mechanism called slow start to increase the congestion window after a connection is initialized and after a timeout.

In this section we are going to test four TCP protocol's performance under congestion, they are: TCP Tahoe, TCP Reno, TCP New Reno, and TCP Vegas. Tahoe is the oldest version of TCP, it uses three method to avoid congestion collapse: slow start, Additive-Increase Multiplicative-Decrease, and fast retransmit. TCP Reno is the version which is now widely implemented, it extends Tahoe's three mechanisms with one new mechanism called Fast Recovery which will set ssthresh to half instead of 1 when get duplicate ACKs. TCP New Reno is defined to improve retransmission during the fast recovery phase of TCP Reno. During fast recovery, for every duplicate ACK that is returned to TCP New Reno, a new unsent packet from the end of the congestion window is sent, to keep the transmit window full. For every ACK that makes partial progress in the sequence space, the sender assumes that the ACK points to a new hole, and the next packet beyond the ACKed sequence number is sent. TCP New Reno modified Fast Recovery algorithm this performs a higher throughput at high packet losses rate. The above mentioned three TCP protocols are so called congestion which uses timeouts and round-trip delays as a measurement to detect network congestion. TCP Vegas, instead, can infers there will have a congestion when the round-trip delay gets longer, and before congestion occur, it can take actions to avoid it.

Following is a simulation of these four TCP's performance under the influence of a Constant Bit Rate flow. First, we set up a network topology like figure 1.1. The bandwidth of each link is 10Mbps. Then add a CBR source at N2 and a sink at N3, add a single TCP stream from N1 to N4. We start CBR flow at time 0.01 second, and start TCP at time 0.2 second.

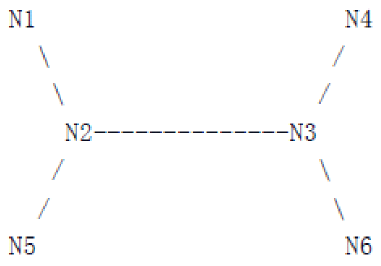


Figure 1.2 is the various TCP's cwnd at the influence of 9Mb CBR. Since we know for TCP

throughput = a * cwnd / RTT

So, figure 1.2 reflects the throughputs of TCP variants. Through this figure we see that upon received duplicate ACKs, tahoe set cwnd to 1, Reno and New Reno set cwnd to half, since vegas could infer the upcoming congestion, it decrease cwnd to avoid congestion. During 0 ~ 1 second, all the TCPs goes in slow start model, then AIMD.

Figure 1.1 (Network Topology)

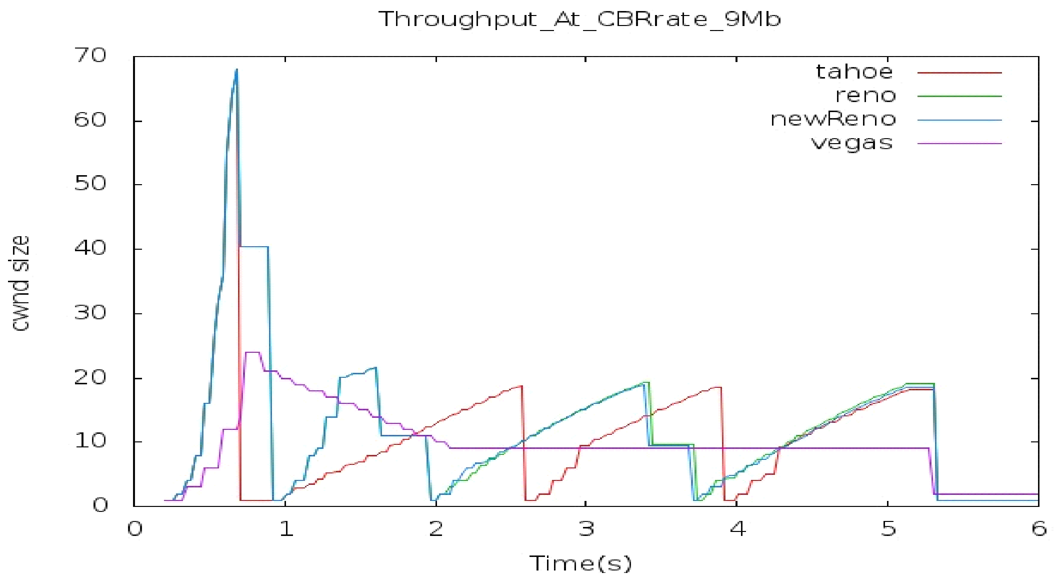


Figure 1.2 (Throughput at

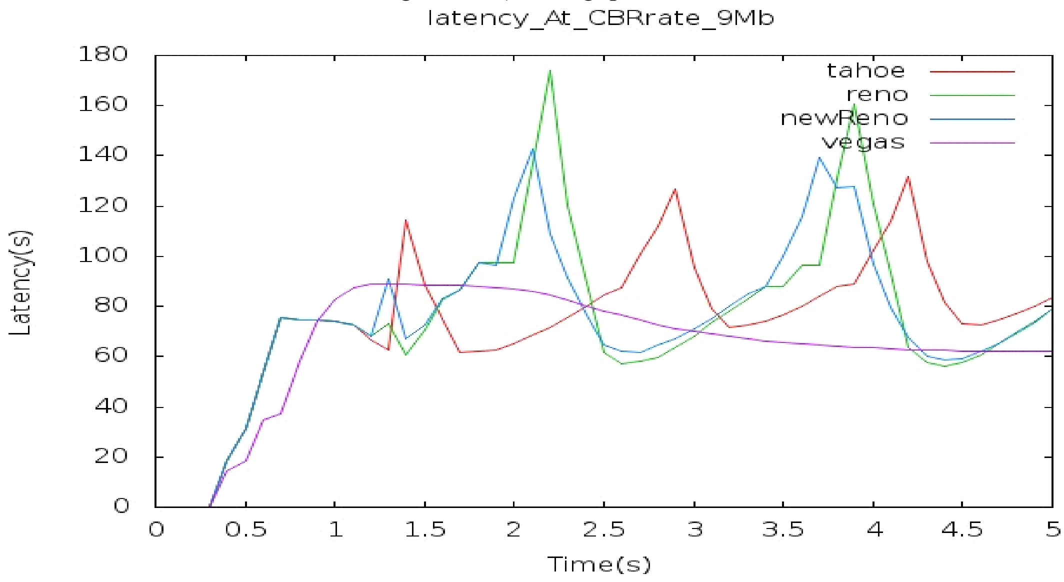


Figure 1.3 (Latency)

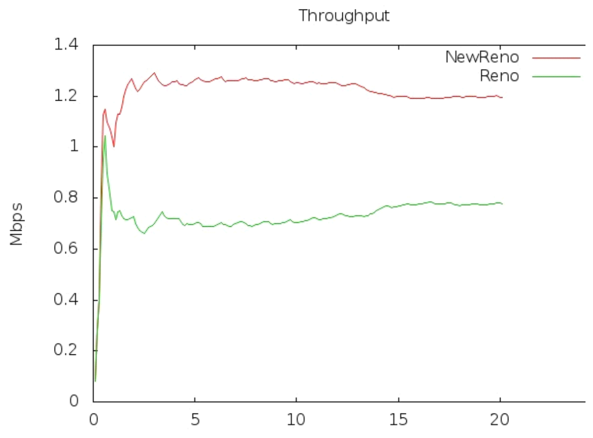
Figure 1.3 shows the TCPs latency performance under Constant Bit Rate flow. We use the following formula to calculate latency:

$$latency_t = (1-\partial) * latency_{t-1} + \partial RTT$$

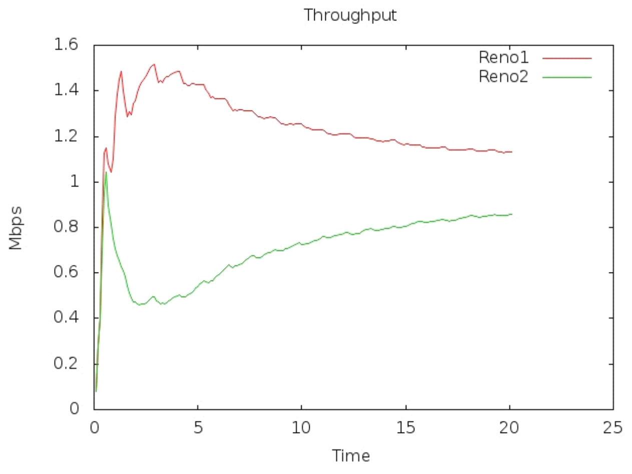
Where RTT is the latest received packet's round trip time. Through figure 1.3 we conclude that vegas has the lowest average latency among these four TCP protocols. Since vegas takes advantage of congestion avoidance instead congestion control, so vegas has the fewest drops.

3 Fairness Between TCP Variants

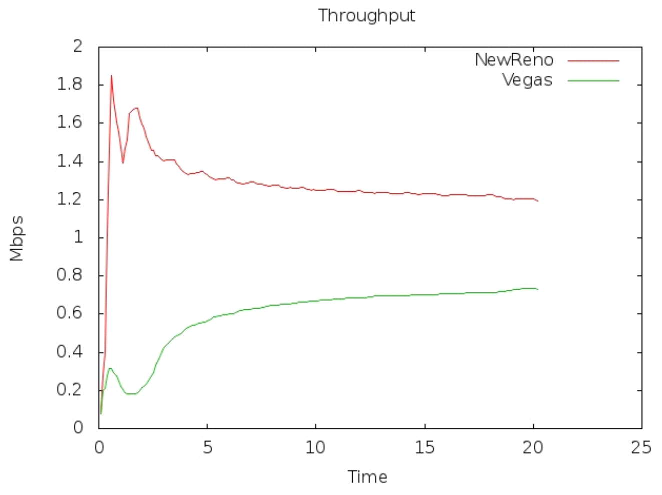
In experiment 2, we conduct experiments to analyze the fairness between different TCP variants. Out on the Internet, there are many different operating systems, each of which may use a different variant of TCP. Ideally, all of these variants should be fair to one another, i.e no particular variant gets more bandwidth that the others. To test whether these assumption exists in real network, we conduct the following experiment. For these experiments, we start three flows: one CBR and two TCP. Add a CBR source at N2 and a sink at N3, the add two TCP streams from N1 to N4 and N5 to N6, respectively.



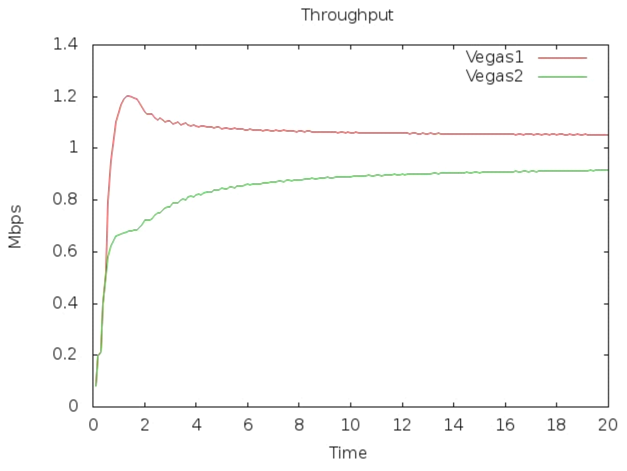
A (NewReno Reno)



B (Reno Reno)



C (New Reno Vegas)



D (Vegas Vegas)

Figure 2.1 (Throughput with two TCP flow)

Based on figure 2.1, we know that there has no fair for combinations TCP flows. For figure 2.1 A the reason NewReno has a higher throughput is because New Reno has the ability to transmit more packets than Reno during Fast Recovery. For 2.1 B the reason that even two same TCP protocol scan not have fair bandwidth is because that upon received duplicated ACK, one Reno decrease its transmitting rate to half, thus release more available bandwidth to another flow. Figure 2.1 C is more clearly that vegas uses Congestion Avoidance so that its throughput is always around throughput "Knee" which is fewer than New Reno.

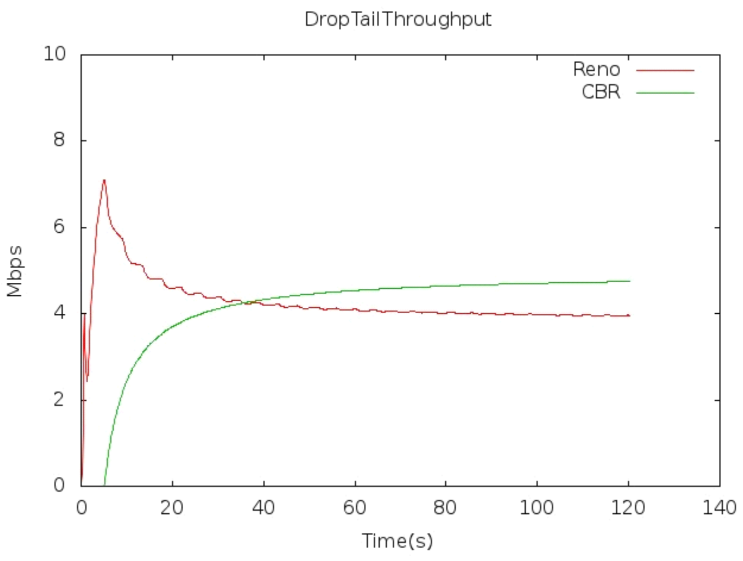
4 Influence of Queuing

Queuing disciplines like DropTail and Random Early Drop are algorithms that control how packets in a queue are treated. In this experiment, instead of varying the rate of the CBR flow, we are going to study the influence of the queuing discipline used by nodes on the overall throughput of floss.

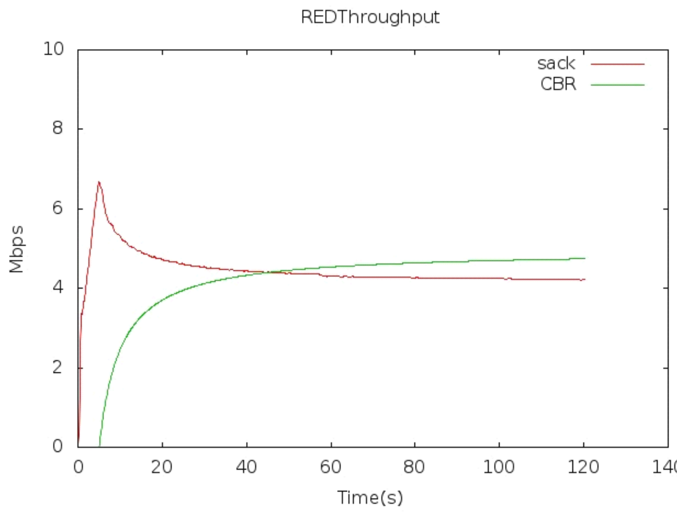
Use the same topology from figure 1.1 and have one TCP flow (N1-N4) and one CBR/UDP (N5-N6) flow. First, tart the TCP flow, we set CBR rate at 5Mbps. Once the TCP flow is steady, start the CBR source and analyze how the TCP and CBR flows change under the following queuing algorithms: DropTail and RED. Perform the experiments with TCP Reno and SACK.

Figure 3.1 shows the average throughput of each flow. From it, we see that not all queuing discipline provide fair bandwidth to each flow. Figure 3.1 A and experiment's data show Drop Tail and RED queuing discipline could not provide fair bandwidth for TCP Reno. Figure 3.1 B and C show Drop Tail tends to provide fair bandwidth for TCP sack but RED queuing failed to provide fair bandwidth.

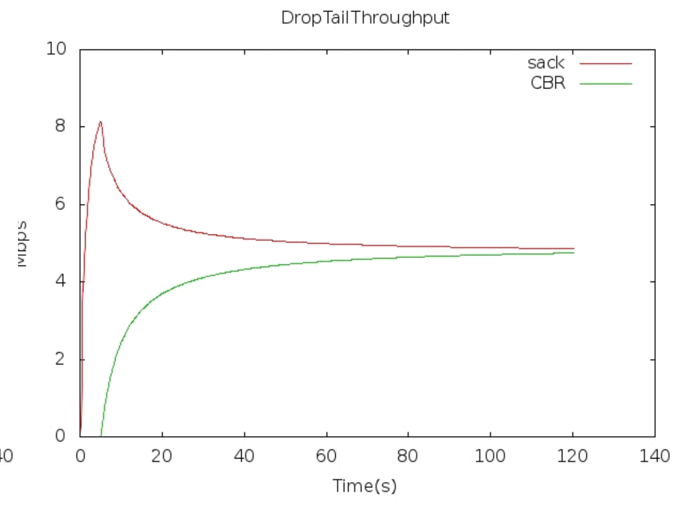
When a CBR flow start transmitting at constant rate 5Mbps, the TCP throughput decreases to around 5Mbps..



A (DropTail Throughput)

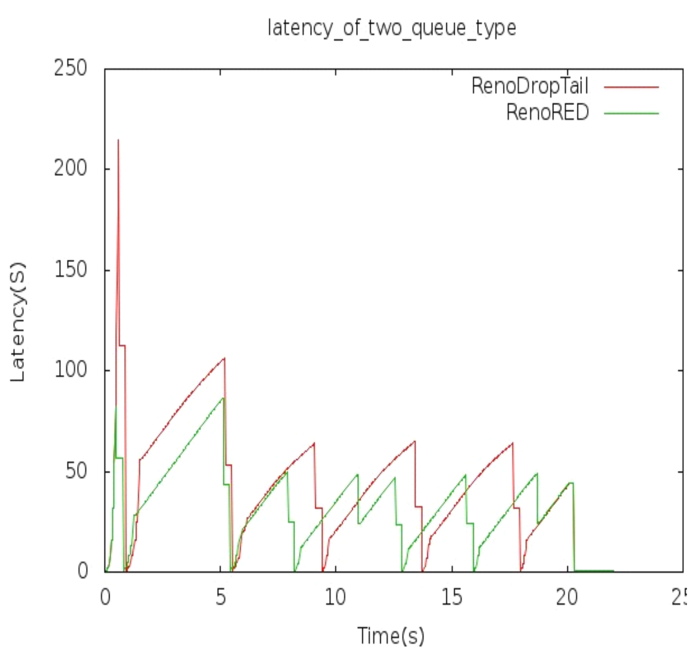


B (RED Throughput)



C (DropTail Throughput)

Figure 3.1



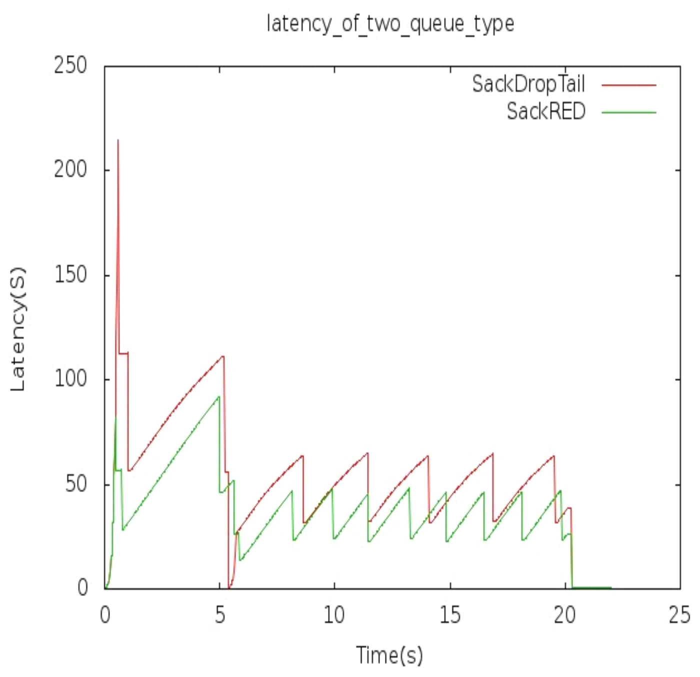


Figure 3.2

Figure 3.2 records the latency between Drop Tail queuing type and RED queuing type, it shows that RED tends to provide fewer latency than Drop Tail for TCP Sack and TCP Reno.

5 Conclusions

In this paper we have explored the performance between TCP Tahoe, TCP Reno, TCP New Reno and TCP Vegas. Through experiment 1 we know that vegas has lowest Round Trip Time and Latency but New Reno has the highest throughput. Throughput experiment 2 we see that in real network event the flows in same TCP protocol can not guarantee bandwidth fair. From experiment 3 we get to know not only TCP protocol but also link queue type affect network performance. One thing that caught us attention is that Figure 1.2 shows that even Reno and New Reno does set cwnd to 1 periodically. More experiments need to be done to find out the reason.