Performance Analysis of TCP Variants

Rouni Yin
Northeastern University, Boston, MA 02115
yin.ro@northestern.edu

Chihao Sun
Northeastern University, Boston, MA 02115
sun.chih@northestern.edu

1. Introduction

The Transmission Control Protocol (TCP) is one of the main protocols of the Internet protocol suite. TCP is connection-oriented, and a connection between client and server is established before data can be sent. It provides reliable, in-order and error-checked delivery of a stream of bytes. The final main aspect of TCP is congestion control. It uses a number of mechanisms to achieve high performance and congestion collapse. The mechanisms control the rate of data entering the network, keeping the data flow below a rate that would trigger a collapse . TCP uses a network congestion-avoidance algorithm that includes various aspects of an AIMD scheme, along with other schemes including slow start and congestion window, to achieve congestion avoidance. The TCP congestion-avoidance algorithm is the primary basis for congestion control in the Internet. There are several variations and versions of the algorithm.

In this paper, we are going to analyze the performance of these different TCP variants (Reno, NewReno, Tahoe, Vegas and SACK). We use the NS-2 network simulator to perform the comparison in the perspectives of performance under congestion, fairness in pairs and influence of queueing. The methodology details and conclusions of each experiment are discussed in the following sections.

2. Methodology

Network topology: To study the behaviour of the TCP variants, we set up a network topology as *Fig.1*, and set the bandwidth of each link to 10Mbps and delay 10ms by default.

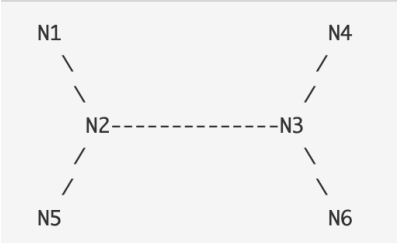

Fig.1 network topology

Tools: NS-2 network simulator: NS-2 is an object-oriented, discrete event driven network simulator. It implements network protocols such as TCP and UDP, and router queue management mechanisms such as Drop Tail, RED. In NS, an event scheduler keeps track of simulation time and fires all the events in the event queue scheduled for the current time by invoking appropriate network components. We can learn the behaviors of TCP variants from the interpretation of the traces generated in NS.

Metrics: MetricsThree parameters, throughput, latency and drop rate, are selected to evaluate the performance of variants of TCP in our experiments.

Throughput is the ratio of total data transferred to the time taken to transfer. It's measured in Mbits per second.

$$throughput = \frac{total\ packets\ received \times average\ packet\ size}{time}$$

Latency is how fast the data packets inside the pipe travel from client to server and back. Here we use average Round Trip Time (RTT) to indicate the overall latency.

$$latency = \frac{\sum(ACK\ TIme - Sent\ Time)}{Total\ Packets\ Number}$$

Drop Rate shows how many packets are lost during the transmission process.

$$drop\ rate = \frac{(total\ sent\ packets\ num - total\ received\ packets\ num) \times 100\%}{total\ sent\ packets\ number}$$

Experiment Configuration: In Experiment 1, using the topology in *Fig.1*, we performed tests that analyze the performance of different TCP variants in the presence of a Constant Bit Rate (CBR) flow.Add a CBR source at N2 and a sink at N3, then add a single TCP stream from N1 to a sink at N4. Analyze the throughput, packet drop rate, and latency of the TCP stream as a function of the bandwidth used by the CBR flow. Start the CBR flow at a rate of 1 Mbps and record the performance of the TCP flow. Then increase the CBR flow's rate by 2 Mbps and perform the test again. Keep changing the CBR's rate until it reaches the bottleneck capacity. The TCP stream's performance will change depending on the amount of contention with the CBR flow.

In Experiment 2, using the same topology in *Fig.1*,we analyzed the fairness between different TCP variants. It starts three flows: one CBR, and two TCP. Add a CBR

source at N2 and a sink at N3, then add two TCP streams from N1 to N4 and N5 to N6. As in Experiment 1, plot the average throughput, packet loss rate, and latency of each TCP flow as a function of the bandwidth used by the CBR flow. Start the CBR flow at a rate of 1 Mbps and record the performance of the TCP flow. Then increase the CBR flow's rate by 2 Mbps and perform the test again. Keep changing the CBR's rate until it reaches the bottleneck capacity. Repeat the experiments using the following of TCP variants:

- Reno/Reno
- NewReno/Reno
- Vegas/Vegas
- NewReno/Vegas

In Experiment 3, Use the same topology from experiment 1 and have one TCP flow (N1-N4) and one CBR/UDP (N5-N6) flow. First, start the TCP flow. 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. Analyze the performance of the TCP and CBR flow over time.

3. Experiment 1: TCP Performance Under Congestion

In this experiment, we are going to analyze the throughput, latency and drop rate of TCP variants (Tahoe, Reno, NewReno and Vegas) under the influence of various load conditions.

3.1 Throughput

We parsed the trace generated in the NS, and calculated the throughput. The simulation result is shown in *Fig. 2*.

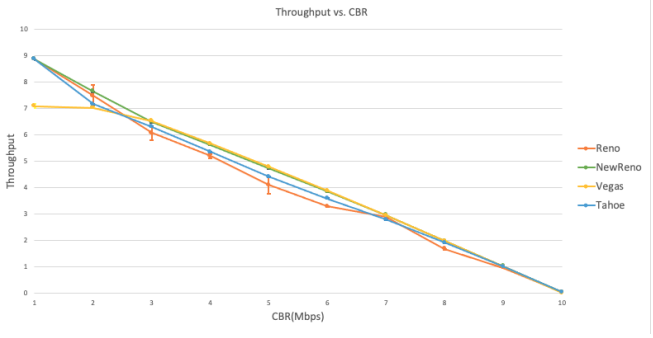

Fig. 2 throughput vs. CBR

As the figure shows, the throughput of each varians is similar with the rise of CBR; but the throughput for Vegas is a little bit lower than others at the beginning. It's because, during slow-start, Vegas has a lower rate of increasing window size due to its modified slow-start mechanism. Then, as the CBR flow increases, the throughputs for all the variants go down. Vegas has a slower rate of decrease. It may be the result of its congestion avoidance mechanism. TCP Vegas does not continually increase the congestion window during congestion avoidance. Instead, it tries to detect incipient congestion by comparing the measured throughput to its notion of expected throughput[1]. Among the variants, Vegas has a slightly better performance under congestion.

3.2 Latency

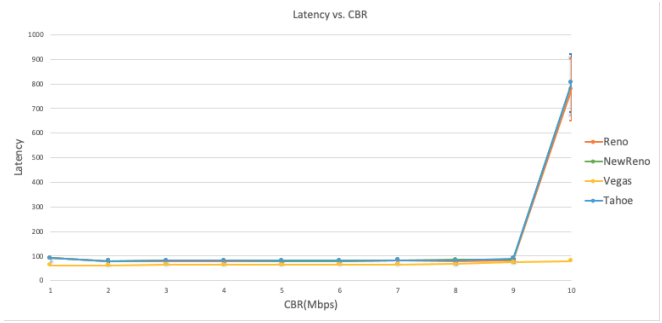

Fig.3 Latency vs. CBR

The average latency of each variant is calculated based on the actual RTT of the packets. We parsed the trace file to keep track of the send and ACK received time of a packet based on the sequence number. The average latency of different TCP variants is shown in *Fig. 3*.

We can see that Reno, NewReno and Tahoe have an almost the same and constant latency at about 80ms. Vegas's latency is the lowest average latency among four TCP varians. It may be the result of the way Vegas detects timeouts based on the RTT, and Vegas calculates RTT to avoid latency and packet drop.

3.3 Drop Rate

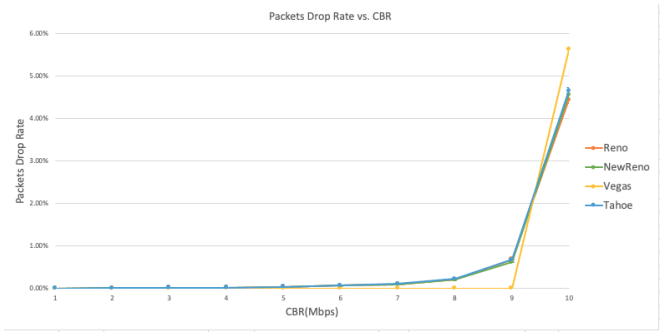

Fig.4 packets drop rate vs. CBR

We count the number of packets sent and acknowledged received to get the packets drop rate. The rates for different TCP variants against varying CBR bandwidth is shown in *Fig.4*.

We can tell from the figure that when the CBR is less than 3Mbps, all of the variants don't drop any packets; because at that time, the congestion hasn't formed yet. As CBR increases, there is an increase in the rate for all the variants. Reno, taking advantage of fast retransmit and fast

recovery[2], maintains the lowest packets drop rate, followed by Vegas. As for Vegas, it calculates RTT to avoid latency and packet drop mentioned above.

3.4 Summary

In this experiment, we can never deny that each variant has its own advantages in specific aspects. When there's no congestion in the network, Reno, NewReno and Tahoe have better performance than Vegas, because they have higher average throughput, making them more efficient. When it comes to congestion, Vegas might be a better choice for the lowest latency and relatively low packets drop rate. We find that Vegas uses a different congestion avoidance mechanism and modified slow-start based on RTT, which makes it perform differently.

4. Experiment 2: Fairness Between TCP Variants

In this experiment, we conduct experiments to analyze the fairness between different TCP variants. There are four pairs of TCP variants we need to check. The topology and experimental setting is the same as Experiment 1, in addition we add another TCP stream from N5 to N6.

We conducted 10+ experiments and calculated the average throughput, packet loss rate and latency of paris of TCP variants. And in order to run experiments that generalize to TCP behavior outside of the simulated environment, we set two random numbers as the relative time when TCP1 and TCP2 are started. Thus, TCP1 will start at first and after a random seconds, the TCP2 will start. The results of simulation are as follows:

4.1 Reno/Reno

Both two TCP streams are set as Reno. According to *Fig.5*, *Fig.6* and *Fig.7*, we can find that throughput, packet drop rate and latency of two Reno are very close to each other until CBR rate reaches the bottleneck capacity. Thus, the pairs of Renos are fair to each other.

4.2 NewReno/Reno

In this part, we set TCP1 as NewReno and set TCP2 as Reno. The result of simulation are as follows:

In *Fig.8*, we can see the throughput of two different TCP variants. Obviously, the average throughput of TCP NewReno is higher than TCP Reno. And in *Fig.9* and *Fig.10*, we can find that the TCP NewReno has a lower packet drop rate and latency when CBR rate reaches the bottleneck capacity. This is because TCP Reno cannot distinguish between full ACK and partial ACK while TCP NewReno can.

Thus, the pairs of New Reno and Reno are unfair to each other.

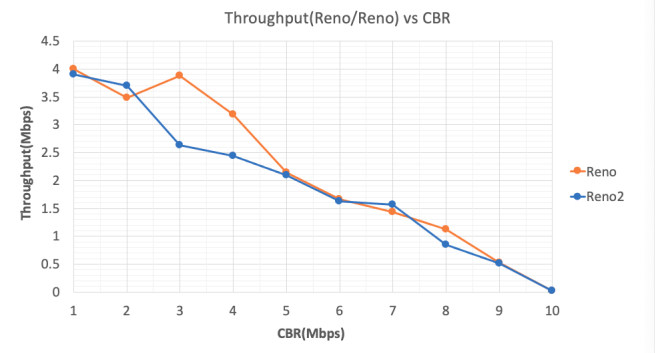

Fig.5 Throughput(Reno/Reno) vs. CBR

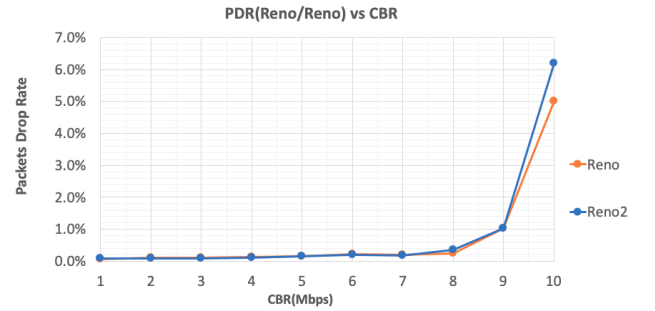

Fig.6 Packets Drop Rate(Reno/Reno) vs. CBR

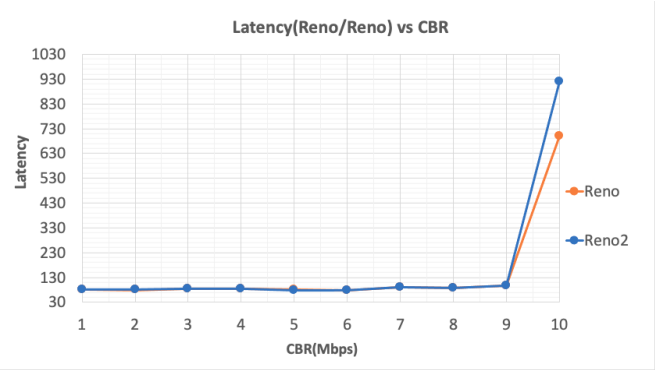

Fig.7 Latency(Reno/Reno) vs. CBR

4.3 Vegas/Vegas

This part, we set both two TCP as Vegas. The *Fig.11*, *Fig.12* and *Fig.13* show the Throughput, Packet Drop Rate and Latency respectively. Obviously, the result of simulation is similar to Part A. Thus, the pairs of Vegas and Vegas are fair to each other.

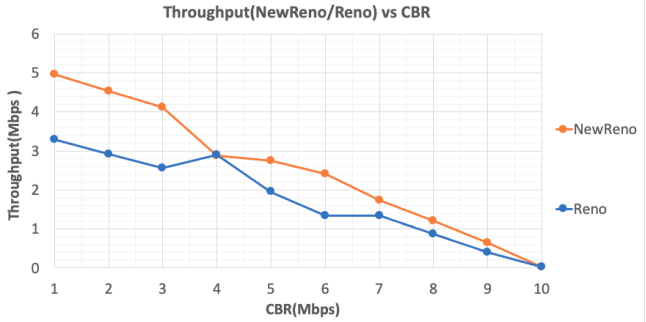

Fig.8 Throughput(NewReno/Reno) vs. CBR

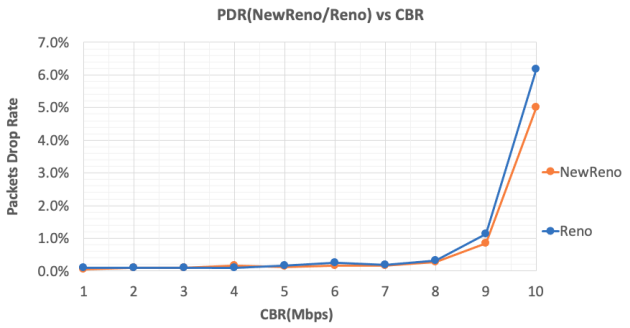

Fig.9 Packets Drop Rate(NewReno/Reno) vs. CBR

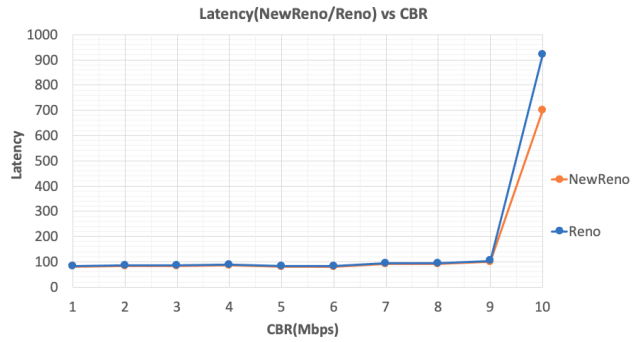

Fig.10 Latency(NewReno/Reno) vs. CBR Bandwidth

4.4 NewReno/Vegas

In this part, we set TCP1 as NewReno and set TCP2 as Vegas. The result of simulation are as follows:

According to *Fig.14*, we can find that the NewReno has higher average throughput. *Fig.15* and *Fig.16* show that NewReno and Vegas have the similar packet drop rate and latency until the CBR rate reaches the bottleneck capacity.

We think this pair of TCP variants are unfair to each other because Vegas emphasizes packet delay rather than packet loss.

That means Vegas will reduces its sending rate before NewReno.

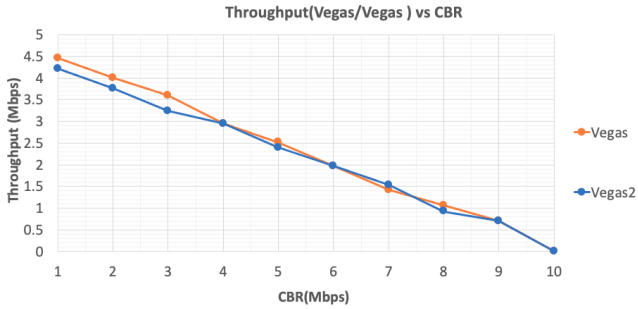

Fig.11 Throughput(Vegas/Vegas) vs. CBR

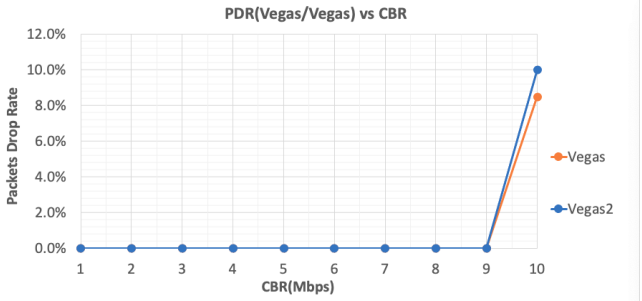

Fig.12 Packets Drop Rate(Vegas/Vegas) vs. CBR

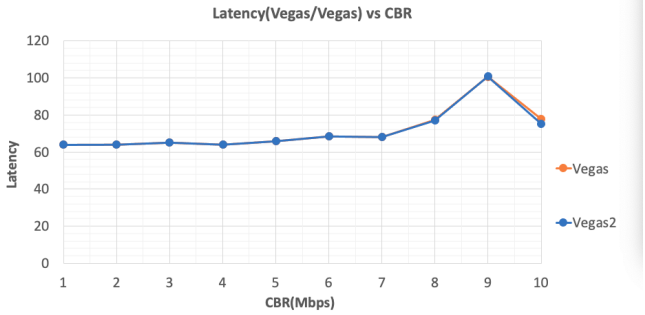

Fig.13 Latency(Vegas/Vegas) vs. CBR

5. Experiment 3: Influence of Queuing

In this experiment, we are going to analyze the influence in throughput and latency of queueing disciplines on the overall throughput of flows. We performed the tests for DropTail and Random Early Drop (RED) with 2 TCP variants, Reno and SACK.

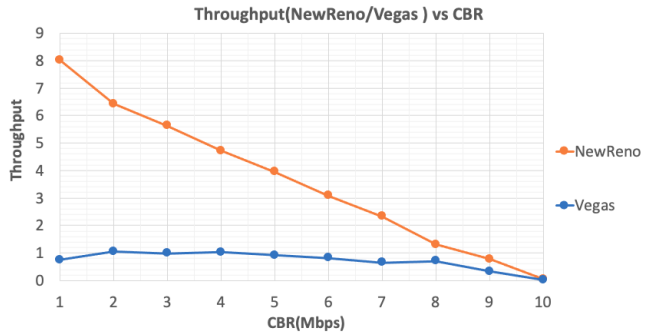

Fig.14 Throughput(NewReno/Vegas) vs. CBR

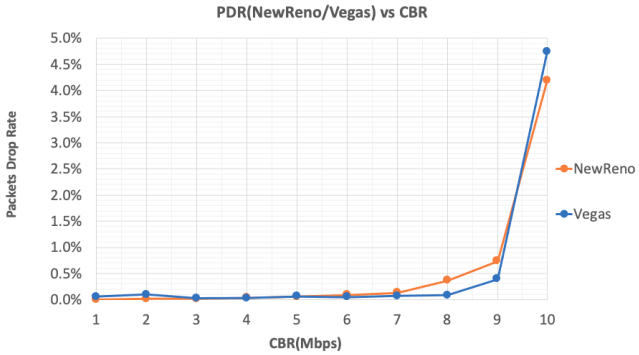

Fig.15 Packets Drop Rate(NewReno/Vegas) vs. CBR

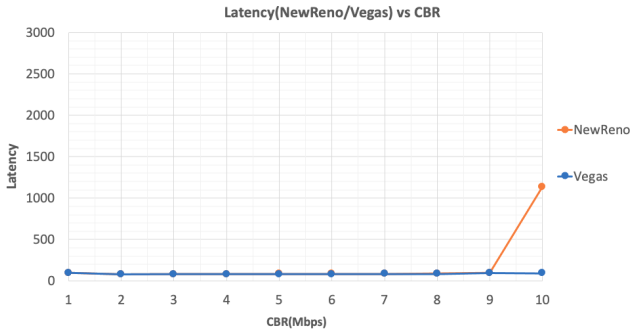

Fig.16 Latency(NewReno/Vegas) vs. CBR

5. 1 Throughput

Before CBR started, DropTail had a better throughput than RED for both Reno and SACK. Plus, Reno and SACK have the similar throughput in DropTail; SACK with RED has a better average throughput than Reno with RED.

When CBR was added, the throughputs dropped and tried to recover to the constant in a period time. We can tell from the graph that Reno with RED has a worst performance in throughput, and SACK with RED and DropTail are working better than Reno. So queuing disciplines are unfair.

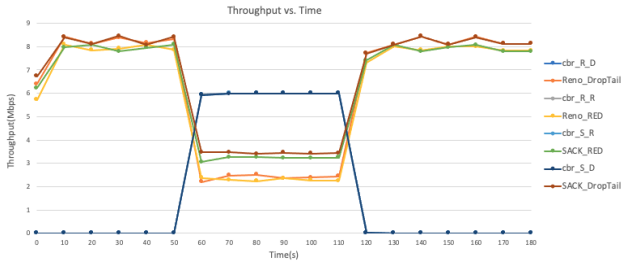

Fig. 17 Throughput vs. Time

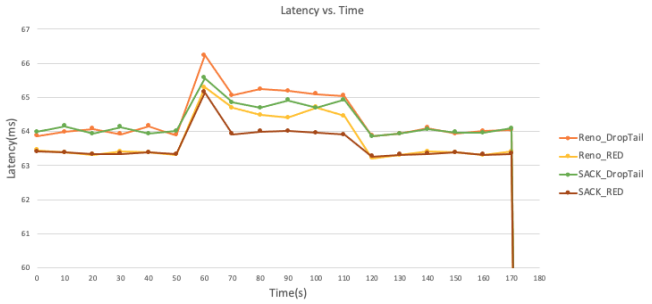

Fig.18 Latency vs. Time

5.2 Latency

Before CBR started, the latency of RED are lower than DropTail. When CBR was added, all the latency rised before recovery. For both SACK and Reno, RED has lower latency than DropTail. And for both RED and DropTail, SACK has lower latency than Reno because of its selective acknowledgement mechanism[3].

According to experimental data,the TCP flow can be affected by the creation of CBR, resulting in lower throughput and higher latency; but it tries to recover. Considering throughput and latency, RED is a good idea while dealing with SACK.

6. Conclusion

In this paper, we analyzed the performance of these different TCP variants in performance under congestion, fairness in pairs and influence of queuing.

According to the experiments, we can draw a conclusion. Under congestion, Vegas and Reno have a relatively better performance. In the experiment of Fairness Between TCP Variants, we can find that the same TCP variants are usually fair to each other, but there will always be some conflict between different TCP variants because of different algorithms and strategies . As for queuing discipline, RED works better then DropTail with both SACK and Reno.

However, it's hard to select a best variant solely according to the experiment results discussed above. Each

TCP variant has its pros and cons. In the future, experiments can be conducted in more complex networks, and more variants can be taken into account for a concrete analysis.

References

[1] U. Hengartner, J. Bolliger and T. Gross, "TCP Vegas revisited," Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 2000, pp. 1546-1555 vol.3, doi: 10.1109/INFCOM.2000.832553.

[2] Khan, Fahad. "A Comparative Analysis of TCP Tahoe, Reno, NewReno, SACK and Vegas."

[3]Fall, K., and Floyd, S., “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP.” Computer Communication Review, V. 26 N. 3, July 1996, pp. 5-21.