Performance Analysis of TCP Variants

Chi Zhang
College of Computer & Information Science
Northeastern University
NUID: 001211610
Email: zhang.chi12@husky.neu.edu

Kaibin Yin
College of Computer & Information Science
Northeastern University
NUID: 001267001
Email: yin.k@husky.neu.edu

***Abstract*—This paper analyzes the performance of different TCP variants from the result of three experiments based on NS-2. The experiments are designed to inspect some key features of TCP like congestion control, fairness and queuing disciplines, and to record the performance aspects of throughput, latency and packets drop rate.**

Keywords—TCP, NS-2, Congestion Control, Packet Queuing

I. Introduction

Since TCP (Transmission Control Protocol) was invented in 1974, it has been one of the most important protocol in Internet.[1] But the original design of TCP performs not well enough to handle the large network congestion. Thus, several TCP variants like TCP Tahoe, Reno, New Reno, Vegas were proposed, implementing different algorithms to control and avoid congestion.

Based on the NS-2 network simulator, this paper performs three experiments including: TCP Performance Under Congestion, Fairness Between TCP Variants, Influence of Queuing, on 5 TCP variants: Tahoe, Reno, New Reno, Vegas and SACK. Then, the paper analyzes their performance from the experiment results, which involve three features: throughput, latency and packets drop rate.[2]

II. Methodology

The network topology as figure.1, is set up in the NS-2 script for experiment. Its structure is with 6 nodes and 5 duplex links, respectively N1-N2, N2-N5, N2-N3, N3-N4, N3-N6 connections. We use Python to trigger the running of NS-2 script and record its trace log, which will be then used for further analysis. By passing predesigned parameters like TCP variant agent, CBR's rate, queue type, start/end time and duration to the NS-2 scripts, we can easily customize different scenarios.

In the first experiment, all links bandwidth is set to 10MB. We set up a TCP flow from N1 to N4 and CBR (Constant Bit Rate) from N2 to N3. We start TCP and CBR at the same time, and set the duration of NS-2 simulator experiment to 30s. By increasing CBR rate from 1MB to 10MB, in an increment of 0.5, we analyze the performance of different TCP variants in the same experiment environment.

In the second experiment, we set up two TCP flow separately from N1 to N4 and N5 to N6 with different TCP variants and a CBR flow from N2 to N3. We start the two TCP flows at first, and start the CBR flow after 3s (from experiment1, we know the TCP flow typically needs 3s to be stable). Because two TCP flows may have contention when they are in the same link, we use its output result to analyze the performance of one TCP variants in the existence of another TCP flow.

In the third experiment, we use the same topology from experiment1. We set the TCP flow from N1 to N4 and start it at the very beginning. After 3 seconds, we put one CBR flow of 7.5 Mbps (which is the threshold to cause TCP flow congestion) from N5 to N6. After running for 30 seconds, we inspect the TCP performance over time. By combining two TCP variants: TCP Reno and SACK, with the queuing disciplines of DropTail and RED, we are to contrast the influence for TCP performance when using different queue algorithms.

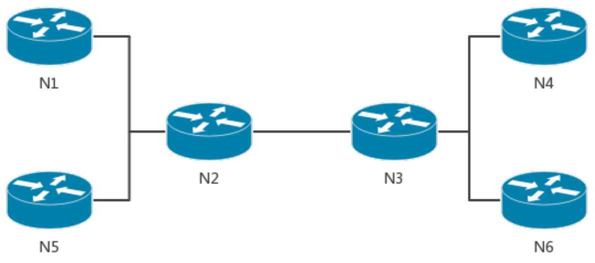

Figure 1: experiment network topology

After running the NS-2 scripts, we parse the trace files and calculate the Throughput, Latency and Packet Drop Rate based on the equations below. The throughput is the size sum of all TCP packets sent per second; the latency is typically the RTT (Round-Trip Time), which is the time difference when the TCP packet dequeued from sender and ACK received by the same sender; and the packet drop rate is the percentage of dropped packets in all TCP packets number.

$$\Sigma(\mathit{Packet\text{-}Size}) * 8 / (T_{end} - T_{start}) = \mathit{Throughput} \qquad (1)$$

$$T_{N1\text{-}ACK\text{-}RECEIVE} - T_{N1\text{-}ENQUEUE} = \mathit{Latency} \qquad (2)$$

$$\Sigma(\mathit{Packet\text{-}Drop}) / \Sigma(\mathit{Packet}) = \mathit{Packet\ Drop\ Rate} \qquad (3)$$

III. TCP Performance Under Congestion

A. Introduction

The first experiment is to test and analyze TCP variants performance under congestion. As mentioned before in methodology section, we set up a network topology to create congestion on the link N2-N3 by increasing the CBR rate.

The purpose of this experiment is to analyze which TCP variant achieves the highest throughput, the lowest latency, and lowest of packet drops rate.

The result of experiment is shown in Figure 2-4, respectively record the throughput, latency and packet drop rate of 4 TCP variants, whose values are plotted for different CBR values starting from 1MB/s to 10MB/s in the increment of 0.1.

B. Result of Experiment

1. **Throughput.** From Figure 2, when CBR rate is relatively low comparing the bandwidth(10MB/s), all 4 TCP variants can get high and steady throughput, while Vegas performs relatively worse. After the CBR rate increases to 7.5MB/s, the congestion happens. The throughput of all TCP variants drops sharply from 2.5. The dip is caused by the reduction of the congestion window.

2. **Latency.** From Figure 3, the latency is low at 0.06 while the CBR rate is low. After the congestion happens when CBR rate is 7.5, the latency starts increasing. It is not surprising that Vegas performs better than any other TCP variants, for it uses RTT values to control congestion, while others wait for the first drop packet. Especially in the condition when CBR rate is lower than 9, Vegas keeps relatively low latency than other variants, which means it could have better performance in most cases except when the congestion is really serious (CBR > 9MB/s).

3. **Packet Drop Rate.** From Figure 4, we can see 4 TCP variants perform almost the same, but Vegas has relatively lower drop rate than others before serious congestion happened (CBR > 9.5MB/s), because of its delay-based congestion avoidance strategy.

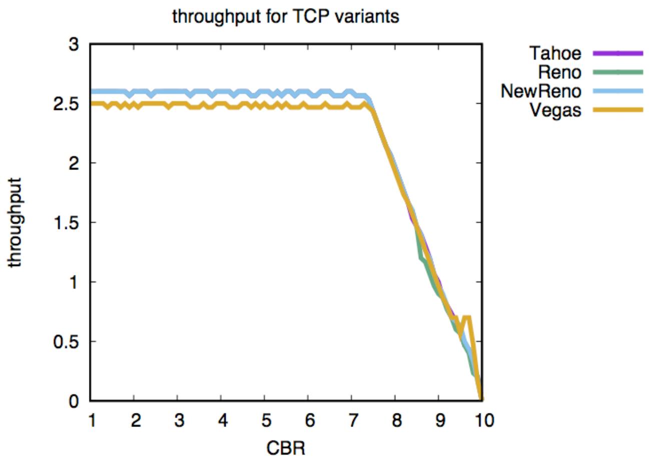

Figure 2: Throughput of TCP variants

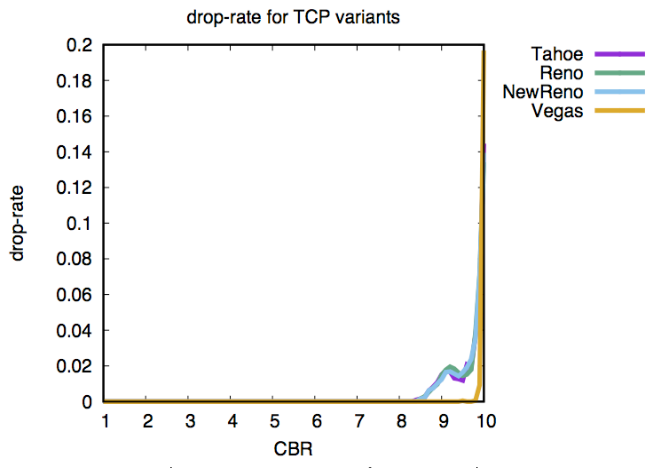

Figure 3: Latency of TCP Variants

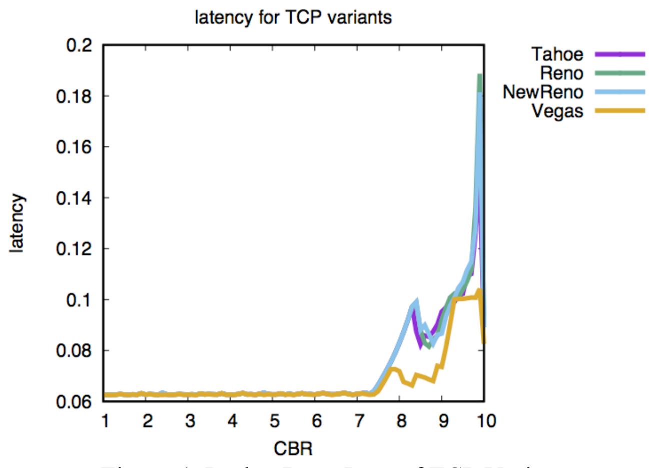

Figure 4: Packet Drop Rate of TCP Variants

C. Conclusion

From the result of experiment, we can come to the conclusion that Vegas performs better in latency and packet drop rate than other 3 TCP variants when congestion happens. But when the CBR rate is relatively low and no existence of congestion, Vegas has slightly lower throughput than other 3 TCP variant. So we can consider Vegas as the overall “best” TCP variant in this experiment.

IV. Fairness Between TCP Variants

A. Introduction

In the second experiment, we aim 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 these variants should be fair when sharing resources like bandwidth with each other. But when it comes to be under the complex network environments, this fairness is still in question.

We use the same topology as experiment 1 and set two TCP flow described in the methodology section. Since both TCP flows will pass through N2-N3, we can analyze and compare the performance of both TCP flows to conclude the fairness between those two TCP variants. We compare 4 combinations of TCP variants: Reno/Reno, NewReno/Reno, Vegas/Vegas, NewReno/Vegas.

B. *Result of Experiment*

1. **Reno/Reno**. When the CBR rate increases, congestion happens, therefore two Reno flows take turn to "occupy" the link to transport packet, as one has high throughput while another has low throughput for a time, like Figure 5. So the throughput of Reno/Reno can be considered fair. The latency and packet drop rate of tow TCP flow is similar from Figure 6, 7, which is also fair for each other.
2. **NewReno/Reno**. From Figure 8, it is obvious that NewReno has higher throughput than Reno when the CBR rate increases to 7MB/s, therefore this combination is not a fair one. The biggest gap between two flow reach 1MB/s. The NewReno and Reno have similar latency, while NewReno has higher drop rate, from Figure 9, 10. The unfairness is due to the more aggressive retransmission of NewReno, which allows NewReno maintained high throughput.[3]
3. **Vegas/Vegas.** Two Vegas flow has similar performances like Reno/Reno. But the oscillation happens that two TCP flow take turn to perform high throughput, from Figure 11. The latency is similar but one Vegas flow has higher packet drop rate than another one, from Figure 12, 13. Therefore, we can say Vegas/Vegas is fair to some extent.
4. **NewReno/Vegas.** From Figure 14, NewReno also suppresses the throughput of Vegas, keeping higher throughput about 0.5-0.8MB/s than Vegas. But Vegas has relatively lower latency and packet drop rate than NewReno from Figure 15, 16. From 1st experiment, we can see Vegas has lower average throughput than NewReno, therefore it could be one of the reason of the unfairness. Further, NewReno aggressive retransmission is also account for it.

C. *Conclusion*

From the result of this experiment, two same TCP variants is fair to some extent, NewReno suppresses the throughput of Reno and Vegas, because of its aggressive retransmission. Especially in the combination of NewReno and Vegas, NewReno increases its congestion window until packet drops, while Vegas uses RTT to avoid congestion, restricting its congestion window. Therefore, the result shows apparent unfairness between them.

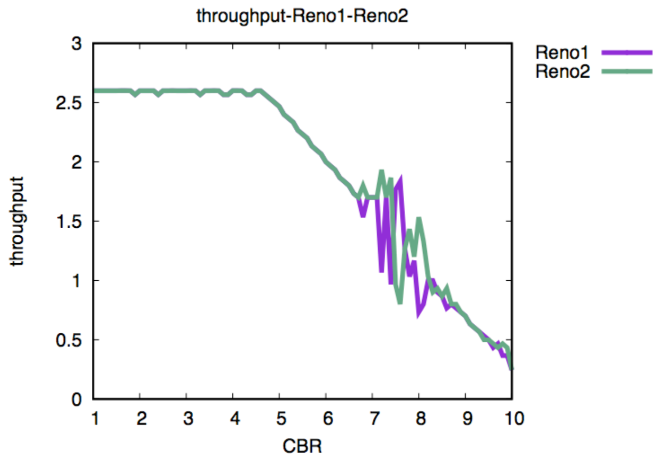

Figure 5: Throughput - Reno/Reno

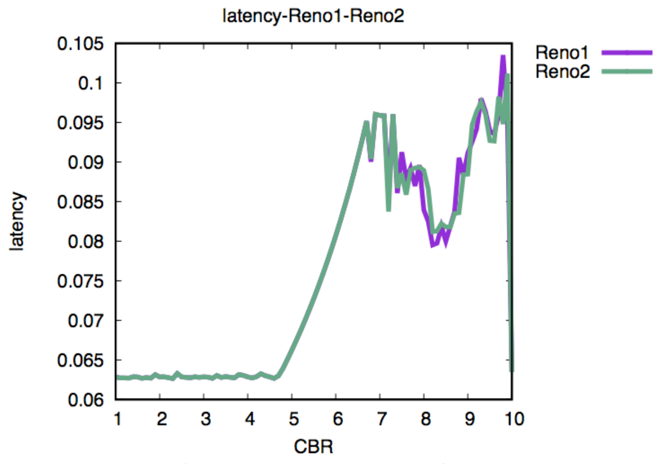

Figure 6: Latency – Reno/Reno

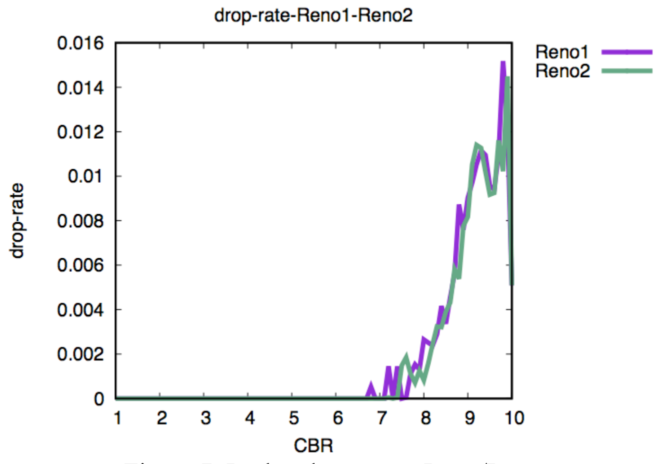

Figure 7: Packet drop rate – Reno/Reno

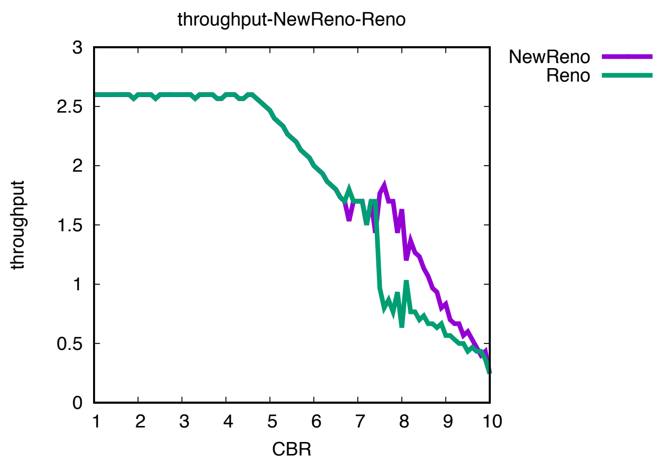

Figure 8: Throughput – NewReno/Reno

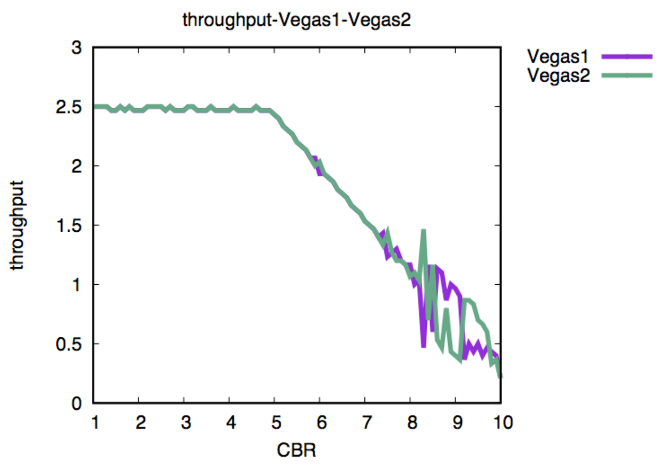

Figure 11: Throughput – Vegas/Vegas

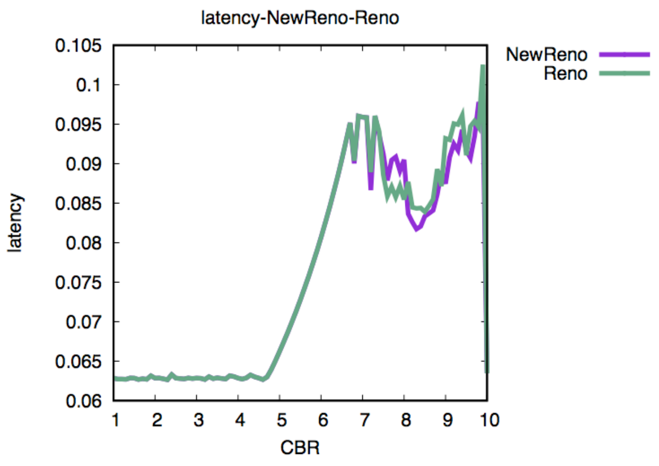

Figure 9: Latency – NewReno/Reno

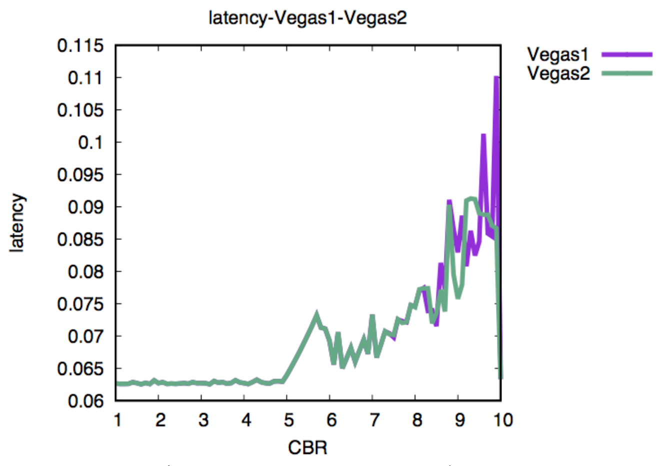

Figure 12: Latency – Vegas/Vegas

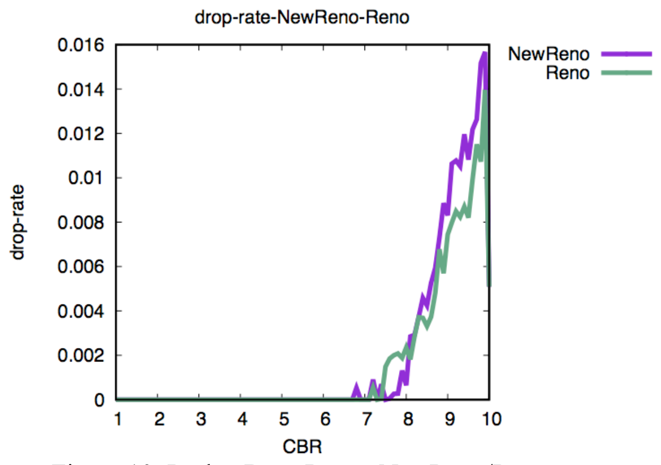

Figure 10: Packet Drop Rate – NewReno/Reno

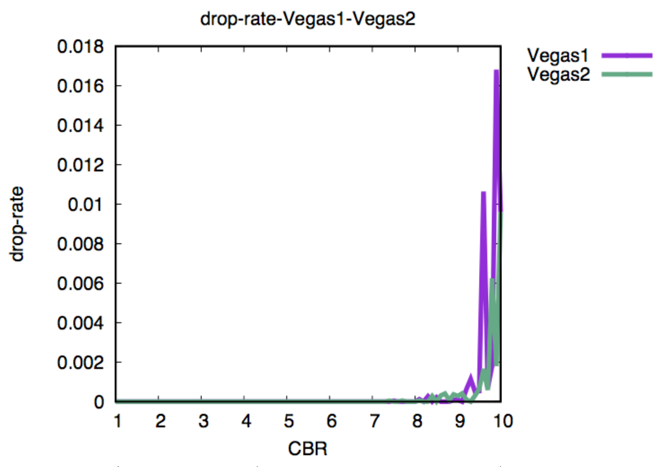

Figure 13: Packet Drop Rate – Vegas/Vegas

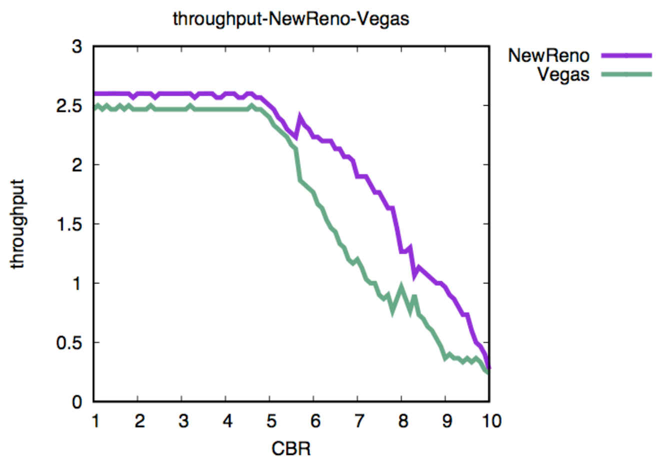

Figure 14: Throughput – NewReno/Vegas

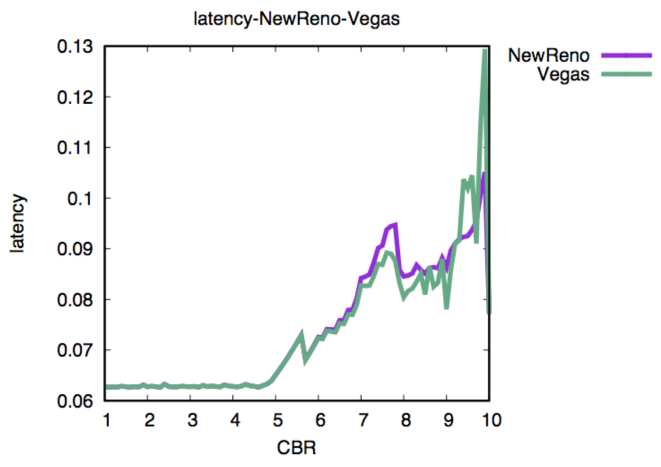

Figure 15: Latency – NewReno/Vegas

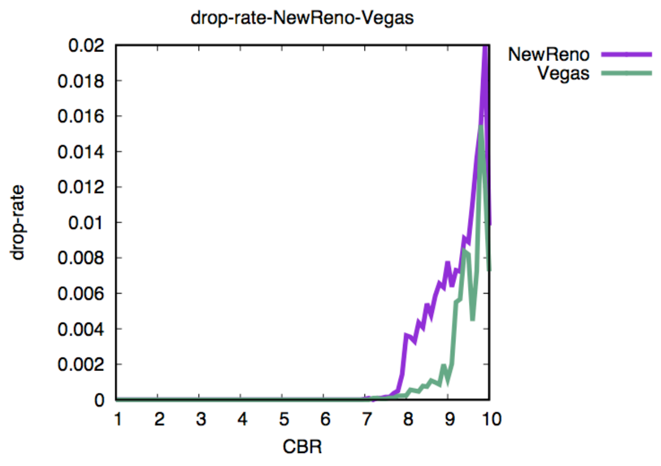

Figure 16: Packet Drop Rate – NewReno/Vegas

V. Influence of Queuing

A. Introduction

In the third experiment, we study the influence of the queuing discipline used by nodes on the overall throughput of flows. We compare the influence queuing mechanisms by comparing throughput and latency for TCP Reno and TCP respectively using DropTail queuing mechanism and RED (Random Early Detection) queuing mechanism.

The most common implementation in router queues on the internet is the DropTail queuing mechanism. It works by queuing up packets until a certain amount and then drops all the packets that come after this point.[4]

RED is more fair than tail drop, in the sense that it does not possess a bias against busiest traffic that uses only a small portion of the bandwidth. The more a host transmits, the more likely it is that its packets are dropped as the probability of a host's packet being dropped is proportional to the amount of data it has in a queue. Early detection helps avoid TCP global synchronization.[5]

B. Result of Experiment

1. **Reno/DropTail -- Reno/RED**
   From Figure 16, Reno with RED has the lowest throughput among 4 combinations. Because RED drops packets randomly before the queue is full, then Reno detects packet drop and enter fast recover, adjusting its congestion window size to a half of current size. The latency of Reno with DropTail is steady the same as SACK with DropTail, from Figure 17. The Latency of Reno with RED increases and decreases rapidly when the congestion happens. But in average, both throughput and latency of Reno/DropTail is higher than Reno/RED.
2. **SACK/DropTail - SACK/RED**
   From Figure 16, SACK with DropTail performs better in throughput than SACK with RED. But SACK with DropTail has higher average latency than SACK with RED, from Figure 17.

C. Conclusion

From the result of experiment, we can come to the conclusion that DropTail with SACK or RENO has good performance on throughput. Queuing mechanism does not provide fairness to the flows, while RED is relatively fair than DropTail. SACK with DropTail has higher throughput than SACK with RED, SACK with RED still provide more reliable transmission.

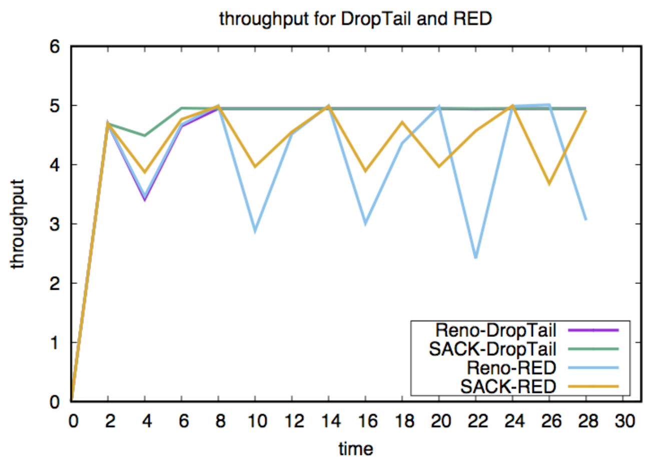

Figure 17: throughput of Reno-DropTail, Reno-RED, SACK-DropTail, SACK-RED

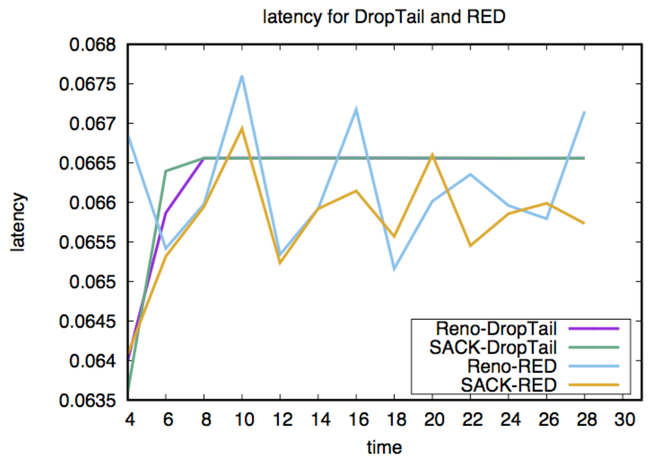

Figure 18: latency of Reno-DropTail, Reno-RED, SACK-DropTail, SACK-RED

VI. CONCLUSION

In these experiments, we analyze the performance of 4 TCP variants and 2 queuing disciplines under normal and congested networks. In section 3, we discover that Vegas has the best performance in latency and packet drop rate when there is congestion, while has a slightly lower throughput when the link is free. In section 4, we conclude that not all TCP variants share the bandwidth evenly when they are on the same link. Especially when NewReno is deployed with other TCP variants, it tends to suppress the other's throughput because of its aggressive retransmission. In the final experiment, we know TCP Reno and SACK can achieve better performance in throughput with the DropTail queuing algorithm, while can get lower latency in combination with RED.

REFERENCES

[1] Cerf, Vinton G., and Robert E. Icahn. "A protocol for packet network intercommunication." ACM SIGCOMM Computer Communication Review 35.2 (2005): 71-82.

[2] Fall, Kevin, and Sally Floyd. "Simulation-based comparisons of Tahoe, Reno and SACK TCP." ACM SIGCOMM Computer Communication Review 26.3 (1996): 5-21.

[3] I Floyd, Sally, Andrei Gurtov, and Tom Henderson. "The NewReno modification to TCP's fast recovery algorithm." (2004).

[4] Comer, Douglas E. (2005-07-10). Internetworking with TCP/IP (5 ed.). Prentice Hall. ISBN 978-0131876712

[5] Floyd, Sally, and Van Jacobson. "Random early detection gateways for congestion avoidance." *IEEE/ACM Transactions on networking* 1.4 (1993): 397-413.