Performance Analysis of TCP Variants

Kezhi Xiong
Northeastern University
xiong.ke@northeastern.edu

Jiatian Wang
Northeastern University
wang.jiati@northeastern.edu

1 INTRODUCTION

In this paper we analyze the performance of some major TCP variants under different network scenarios and compare the results to gain a better understanding of their behaviours. Today's TCP includes various variants with different congestion control preferences. These variants are used by different parts of the internet (e.g. Windows uses Compound while Linux use Cubic). A better understanding on their behaviours under different network conditions can help us quickly locate bottleneck, troubleshoot the problem and build a fair and efficient internet environment.

Our experiment investigated 5 TCP variants (Thaoe, Reno, NewReno, Vegas, and SACK) and 2 queuing disciplines (DropTail and Random Early Drop or RED). We found that the RTT- or latency-base congestion control algorithm (Vegas) produces a better performance and fairness than others in a low congestion environment since it is more sensitive to the increasing number of packets in the network, thus can better predict the congestion and steady the flow. However, such an algorithm with additive increasing and additives decreasing (AIAD) congestion window makes it less aggressive than others with exponential increasing stage of congestion windows, especially in high congestion environment. We also found out that that among the first three variants, NewReno gets the best performance and is the most aggressive one in face of high congestion due to its mechanism to partial ACKs[1]. Lastly, we conclude that RED is better than Droptail because RED drops packets before the queue is filled up. It can also decrease the end to end latency because of that.

In section 2 we describe methodology to conduct our experiments and analyses. Section 3 describes the experiment for TCP performance under congestion. Section 4 describes the experiment for fairness between different TCP variants. Section 5 shows how different queuing disciplines influence TCP performance and fairness. Finally, Section 6 illustrates the lesson we learnt on TCP through these experiments and discusses how our results is related to the real-world deployed systems.

2 METHODOLOGY

As an overview, in order to find the relationship between the targets (performance, fairness, etc.) and corresponding network settings (e.g. CBR rate), we conduct our experiments by controlling variates and collecting packet's events or tracing data in the network. Then, we gather events of a certain type to further calculate the performance properties under the the a give network condition. Finally, we plot graphs on these properties in terms of the changing variate and analyze the statistical significance when necessary.

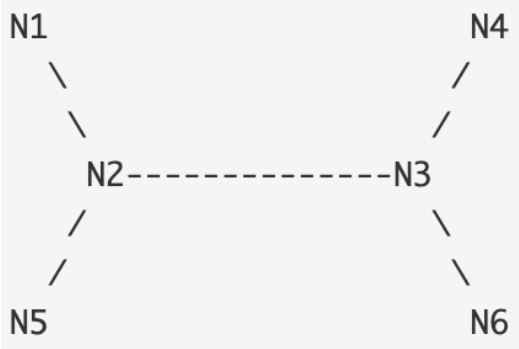

Figure 1. Network Topology

Figure 1. shows our network topology. Each link in the network is a duplex link with 10Mb throughput and 2ms latency. These links use DropTail as the default queuing algorithm which will be replaced only in experiment 3.

In this paper, we consider the performance in three aspects: throughput, drop rate, and latency. These aspects are all calculated from the tracing data between link N2 and N3, where all traffic meet and compete with each other. Throughput is calculated based on "r" or "received" events and the duration of the flow of a given type. Drop rate is calculated based on "d" or "drop" events and the total number of packets of a given type. Latency is slightly more complicated since we must first pair each "+" or "enqueue" event with corresponding "receive" event. To do that, we start by splitting the "enqueue" events and "receiving" events, then index these events on "pkg id". Finally, we join these two event groups on "pkg id" to calculate the latency for each packet and eventually, the average latency overall.

2.1 TCP Performance Under Congestion

In this experiment, we examine the average throughput, drop rate, and latency of Tahoe, Reno, NewReno, and Vegas under the increasing CBR/UDP flow from 1Mbps to 10Mbps with 0.1Mbps growth each time.

During the experiment, we first start the CBR flow from N2 to N3. Then, after 1 second, we start our TCP flow from N1 to N4. Finally, these two flow end after ten seconds. Since the CBR traffic is always constant and below the network bandwidth, we didn't vary the start time of the TCP flow. Also, due to our goals is to investigate TCP performance under various load conditions, not how it reacts to or compromise a new traffic, we didn't start TCP flow ahead of the CBR flow.

2.2 Fairness Between TCP Variants

This experiment evaluates the fairness between 4 TCP variants pairs: Reno/Reno, NewReno/Reno, Vegas/Vegas, and NewReno/Vegas. The fairness of two TCP variants, like in the last experiment, is also concluded from their average throughput, drop rate, and latency. In the topology shown by Figure 1., the first TCP variant in each pair flows from N1 to N4 and the second one flows from N5 to N3. To simulate the real-world situation in terms of congestion, we also set up a CBR traffic from N2 to N3 which increases from 1Mbps to 10Mbps at the step of 0.1Mbps each time.

Unlike experiment 1, now we are investigating the fairness or how each TCP competes or compromises another. Thus, for each CBR from 1Mbps to 10Mbps, we overlapped the time window of the two TCP variants. To illustrate, we use T(t_1, t_2) to denote the start time of the two TCP variants in second. We run our experiment in T(0,0), T(0,2), T(2,0), T(0,5), T(5,0). For each run, the CBR starts at the beginning and all three flows end at the 10th second.

2.3 Influence of Queuing

In this part, we analyze how different active queuing managements (AQM) influence the performance of different TCP variants. The two strategies we used in this experiment are Droptail and Random Early Drop (RED).

With topology shown in Figure 1., we set one TCP flow from N1 to N4 and another one from N2 to N3. In our experiment, the TCP flow starts first, then we set the cbr flow to start at 1, 3, 5 second respectively in 3 different tests. Finally all flows stop at 10 second. Unlike the previous experiments, the CBR flow is set to 8Mbps by default. Besides all above settings, we also vary TCP variants (Reno and SACK) with the two different AQM (DropTail and RED), making 4 pairs to test (12 tests in total considering the 3 different CBR flows). During the experiment, we collect each packet's event to eventually calculate the overtime throughput, drop rate, and latency of the TCP and CBR flows.

3 TCP PERFORMANCE UNDER CONGESTION

Figure 2 shows the relation between different performance properties and the CBR rate on the link between N2 and N3. The sub-figures show a synchronous trend: after CBR reaches 5Mbps, the performance decreases dramatically and different variant behaves in different ways. The turning point appears around 5Mbps due to the congestion for the 10Mbps bandwidth of the link. Notice that Vegas's turning point in drop rate and latency comes much later or its throughput decrease is smooth overall CBR rate. This difference is because of Vegas's unique delay-based congestion detection over the ones based on dropped packets[2]. Its advantage in latency can be tell from the latency graph. Besides, Vegas is also efficient in congestion control in terms of the drop rate, its drop rate can be ignored until CBR reaches 9MB. After reaching the bottlenect, Reno perform worse on throughput. Comparing other drop-based variants, its handling to partial ACKs keeps it in "Fast Recovery" stage very shortly: once it receives a new ACK, it leaves the "Fast Recovery" period. This mechanism makes it switch back and forth between "Fast Recovery"

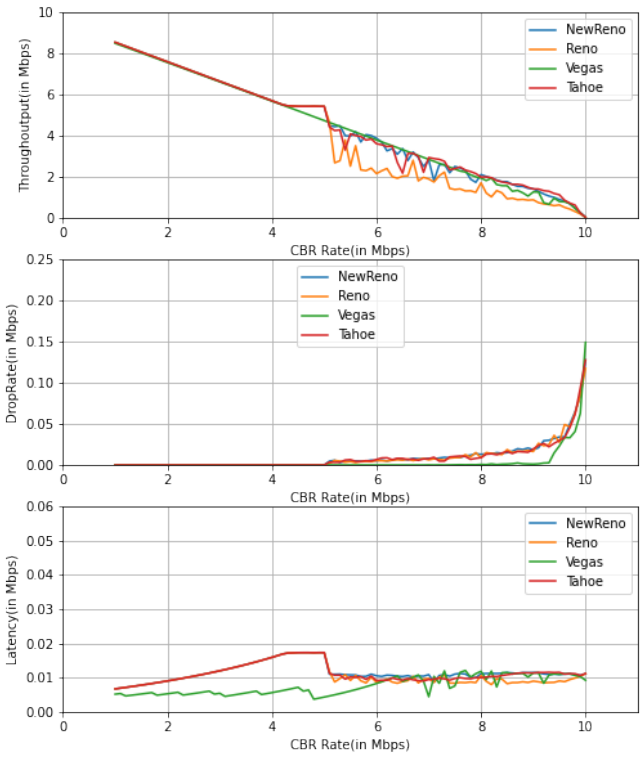

Figure 2. Performance of TCP Variants Under Congestion

and "Fast Re-Transmission" stages and doesn't make much progress on the congestion window size.

The reason that the drop rate and latency of Vegas better than others is also its delay-based congestion detection. When reaching the bottleneck of the network, Vegas can detect congestion and reduce the amount of data to transmit earlier due to the increasing RTT, the result of increasing number of packets on the link. While other variants will not until the first dropped packet, at which point congestion already formed.

By running t-test on all of our data, if comparing one variant with all others, Reno has a statistically lower throughput (4.0 $\leq$ CBR $\leq$ 8.1) than all other three variants. Vegas has a statistically lower drop rate (5.5 $\leq$ CBR $\leq$ 10) and latency (1.2 $\leq$ CBR $\leq$ 10). Thus, we consider Vegas as the overall best due to its smoothly decreasing throughput, lower latency and drop rate in a congestion environment.

4 FAIRNESS BETWEEN TCP VARIANTS

Figure 3-6 show the performance between the pair of variants.

Consider Figure 3 and 4, which both show competing between the two same TCP variants. The trend shows that same TCP variants share the bandwidth fairly, although one maybe more competing than other for one small period. This fairness is obvious since when two variants are using the same congestion control algorithm, their interaction should be symmetric. Besides, our statistical result from t-test also proves the fairness (p-value is always greater than 0.05).

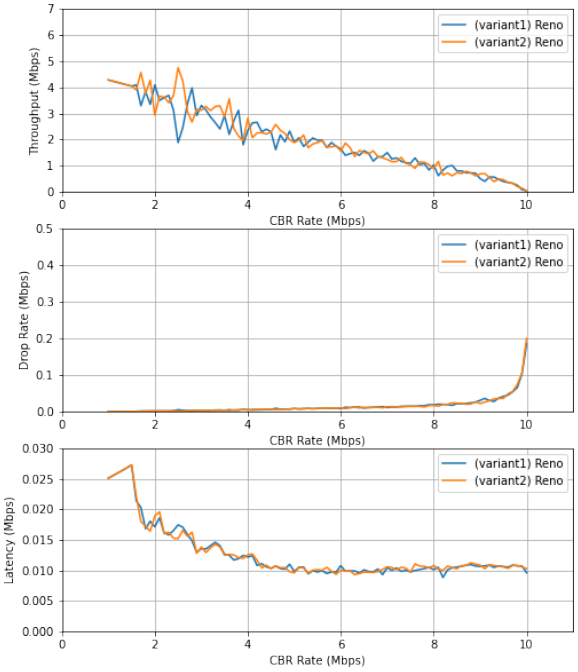

Figure 3. Performance Between Reno and Reno

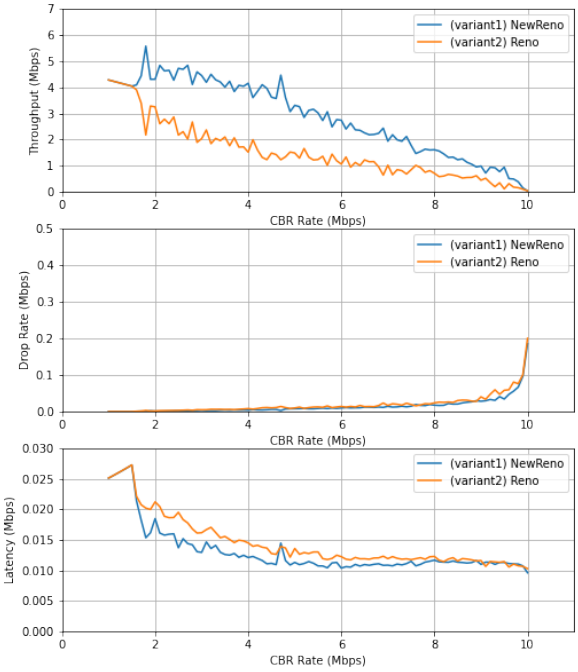

Figure 5. Performance Between NewReno and Reno

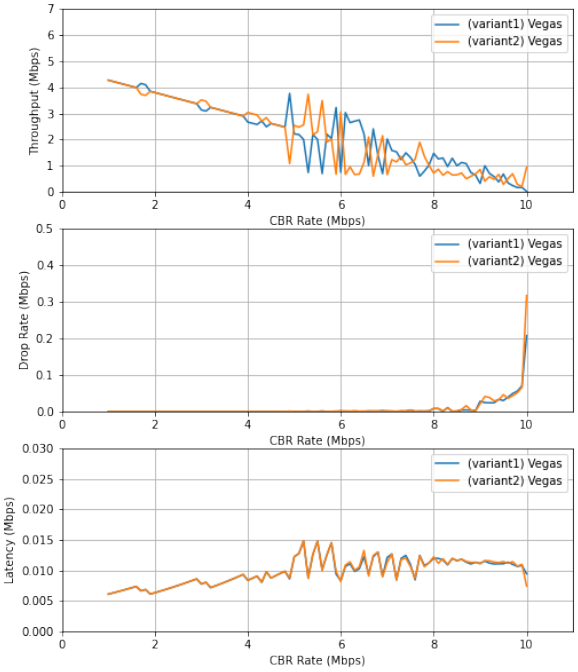

Figure 4. Performance Between Vegas and Vegas

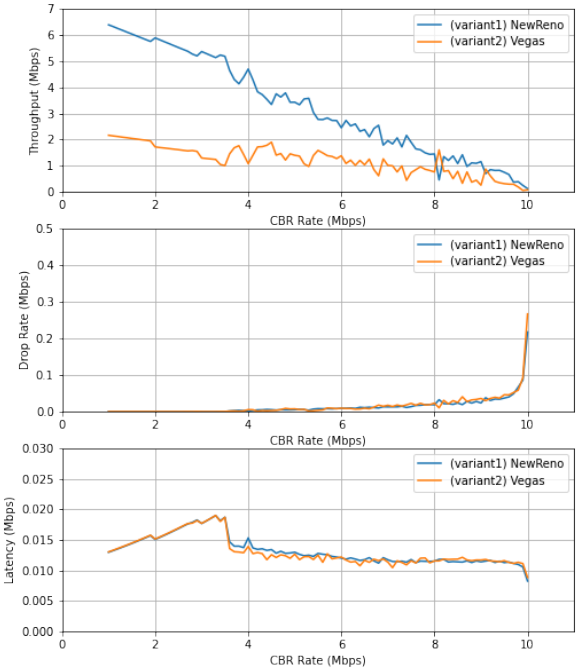

Figure 6. Performance Between NewReno and Vegas

Figure 5 shows the competing between NewReno and Reno. It shows that NewReno is more aggressive than Reno in a congestion environment in terms of Throughput which is shown by our statistical data for CBR $\geq$ 1.2Mbps. The reason behind is similar to the discussion in Section 3, NewReno only exit "Fast Recovery" stage when all lost data are acknowledged which suffers from multiple packets lose in a high congestion environment. Accordingly, this also reduces NewReno's latency since it would encounter less timeout and re-transmission. In fact, NewReno's latency is statistically lower over all CBR rate. Lastly, the drop rate is less obvious from the Figure. However, NewReno's drop rate is statistically lower after CBR reaches 1.7Mbps.

Figure 6 compares NewReno and Vegas. Like in the last comparison, NewReno is the more aggressive one, especially in a relative low congestion environment. In a congestion environment, network tends to have more timeout than package drop. At this time, Vegas still uses AIAD while NewReno fallback to slow start and increases its congestion window exponentially, enabling it to take more advantage of the bandwidth. Statistically, NewReno has higher throughput (over all CBR rate). However, when it comes to the latency which is the only congestion evaluation metrics for Vegas, although it is not statistically lower overall CBR rate, it performs better in some smaller ranges: from 3.3 to 4.3 and from 7.1 to 8.5.

As a summary, this section analyzes fairness between TCP variants. We found that between the same TCP variants, they share the network fairly. However, there is no fairness at all between different TCP variants, especially when it comes to NewReno, a very aggressive variant.

5 INFULENCE OF QUEUING

Droptail queuing mechanism is most widely applied active management (AQM) algorithm used in routers. The algorithm set a buffer to receive packets. When congestion happens, new arrived packets will be stored in the buffer, if the buffer is not empty. The upcoming packets will be discarded evenly. And Tcp sender will detect their packets are not delivered and will go throw a fast recovery progress. So it can not distribute buffer space fairly. Flows with higher rate will take more buffer size. Random Early Detection (RED) detects the congestion in advance and drops packets randomly before the queue is completely filled up. Although it randomly discards packets but it share the same drawbacks as Droptail. Flows with higher rate will take more space in queue and more likely stay in the queue. RED can not provide fair bandwidth either.

Firgure 8 - 11 shows the throughput, drop rate, latency of the Tcp flow and Udp flow when different TCP variants and AQM is applied. According the Figure 8-11, cbr flow starts at 3.0 sec, throughput of tcp flow dropped after that. We set the flow rate of cbr as 8Mbps. After a second, the throughput of cbr flow is stable at 7.5Mbps. The bandwidth is 10Mbps, by the result of experiment1, when cbr flow rate is relatively low, the ftp flow throughput is around 2.5Mpbs. That means these two flows share the bandwidth but not fairly. Flow with higher traffic volume take more queue space in every test based on result of Figure 8 -11.

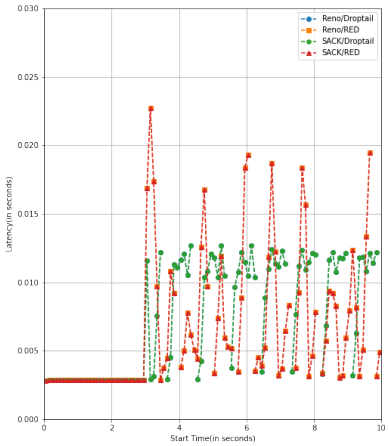

Figure 7. Latency

The end to end latency for Droptail algorithm is higher than RED, Because RED can discard packets before the queue is filled up so it can avoid discard all packets from a Tcp sender. For droptail, if all packets are discarded , the tcp sender will go through a fast-recovery procedure. 'Which will increase the overall latency. According to the figure 7, we can get the conclusion.

Figure 8-11 shows that after 3.0 sec which is the time that cbr flow starts. The throughput of Tcp drops rapidly, and go through several fast recovery procedure in almost evert test.

The congestion control of SACK is an extension of Reno's congestion control. Reno has a fast recovery procedure and can avoid slow start which increase the sending rate. We can see that in figure 9. It has a quiet stable flow when cbr flow is started. Based on Reno, SACK maintained a variable called pipe which limit the packets to be sent.

For SACK with RED, we noticed that the tcp drop rate is relatively high than the other three which means many tcp packets are dropped. So we think RED is not good on SACK. Instead, When TCP/Reno is applied with red, throughput of tcpnever reach 0Mbps and is quiet stable around 2.5Mbps. Which makes us think TCP/Reno with RED is the better choice based on the simualation.

However,We also run t-test on pairs of different queuing Strategies with different TCP variants. The p-value is always greater than 0.05. Based on the statistical result, the change of TCP variants make more impact on the change of Queuing Strategy. So to improve overall performance. We have to figure out the best combination of Queuing Strategy and TCP variant.

In summary, Droptail and RED can not distribute bandwidth fairly and RED can decrease the end to end latency. When a CBR rate(8Mbps) starts, tcp flow rate dropped. And in this experiment, RED is not dealing very well with SACK because the drop rate of tcp flow is very high. And TCP/Reno with RED perform better than the other three.

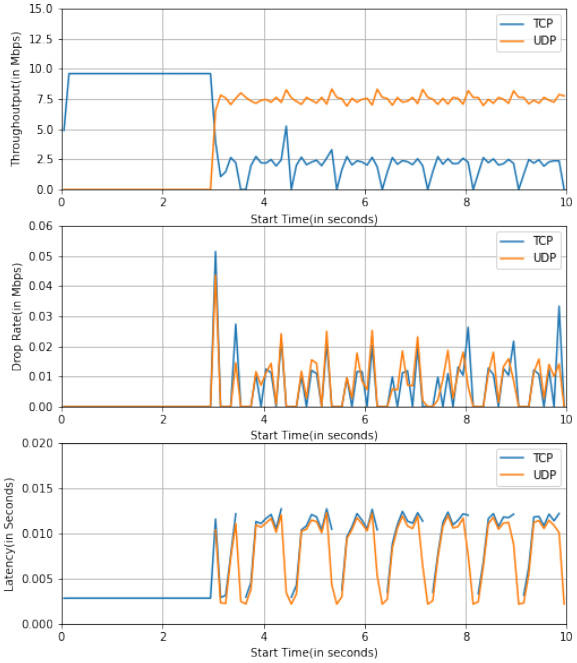

Figure 8. Reno with Droptail

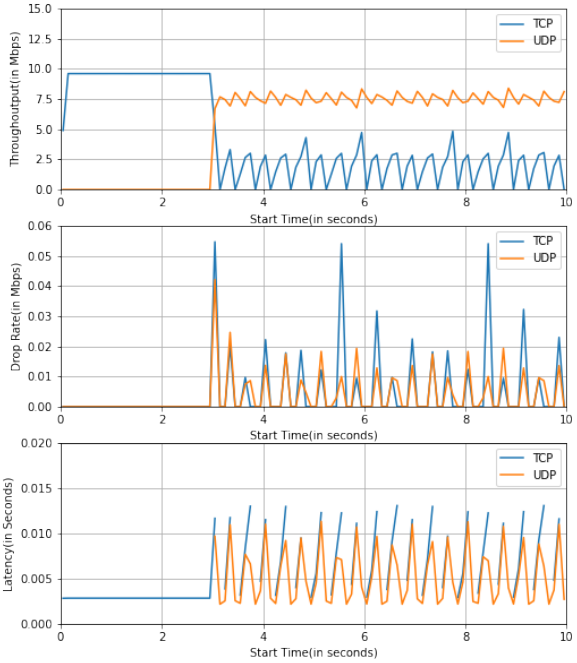

Figure 10. SACK with Droptail

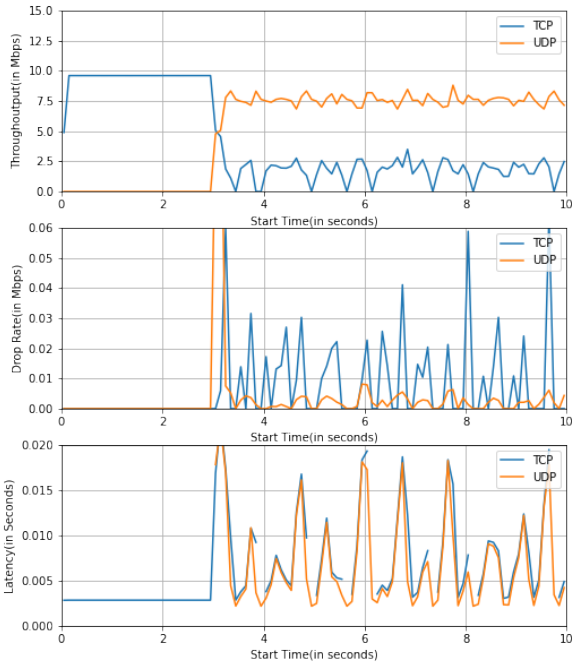

Figure 9. Reno with RED

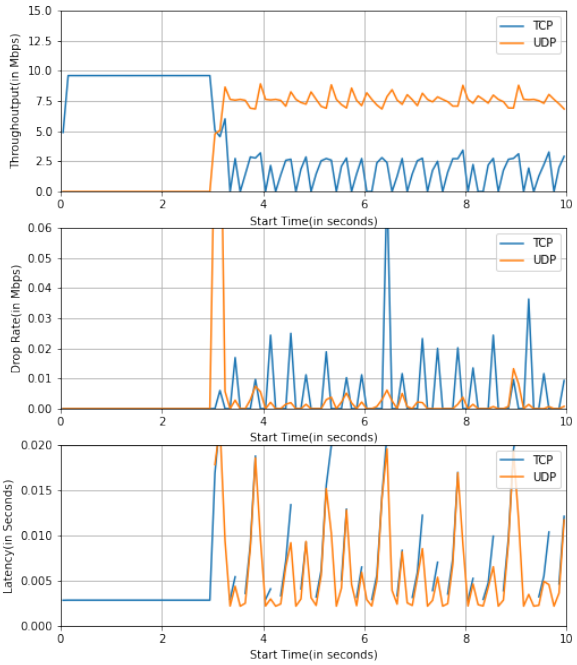

Figure 11. SACK with RED

CONCLUSION

In this paper we have explored the performance and fairness of different TCP variants and queuing disciplines. Among all TCP variants, NewReno gets the best performance but is also most aggressive. The only latency-based variant, Vegas, is most efficient in congestion control and most fair to others. Lastly, besides these variants, queuing algorithms also have an impact on TCP performance. Particularly, RED has a better performance than DropTail in a high volume network.

We believe that for a better internet environment, more varied algorithms to handle congestion should be discovered, more complex and estimation based mechanism on various network factors should be presented.

REFERENCES

[1] Kevin Fall and Sally Floyd. 1996. Simulation-Based Comparisons of Tahoe, Reno and SACK TCP. 26, 3 (jul 1996), 5–21. DOI: http://dx.doi.org/10.1145/235160.235162

[2] Andrew S. Tanenbaum and David Wetherall. 2011. *Computer Networks* (5 ed.). Prentice Hall, Boston. https://www.safaribooksonline.com/library/view/computer-networks-fifth/9780133485936/