Performance Analysis of TCP Variants

Abhishek Sawarkar
Northeastern University, MA – 02115

Himanshu Saraswat
PES MCOE,Pune-411005

Abstract
The widely used TCP protocol was developed to provide reliable end-to-end delivery of packets under varying degrees of congestion in the network. This paper presents with in-depth study of performance characteristics of different TCP variants viz. Tahoe, Reno, New Reno and Vegas under congestion. We carry out simulations in NS-2 environment on these TCP variants and analyze the results with respect to throughput, packet drop rate and latency to determine if there is an overall efficient TCP variant. We also study the performance of Reno and SACK for different queuing algorithms viz. Droptail and RED.

Keywords
throughput; latency; drop rate; NS-2;

1. Introduction

The possibility of congestion occurance is growing rapidly with growth of networks. Initially, the size of sending window of TCP was determined by the available buffer size (advertised window) at the receiver. But this resulted in providing with only flow control and no congestion control. To cope with the growing congestion in networks, many TCP variants were researched and developed thereafter.

Few of these TCP variants are discussed below:

A. TCP TAHOE

TCP Tahoe was introduced with 3 congestion control algorithms, namely:

a) Slow Start
b) Congestion Avoidance
c) Fast Retransmit

New parameters for congestion control were introduced like congestion window (cwnd) and slow start threshold window (ssthresh). The size of congestion window varies as per the reception of acknowledgements for the sent packets. Failure to receive an acknowledgement before the expiry of the defined RTT period is interpreted as packet loss due to congestion by TCP. The next for TCP is to implement fast retransmit where-in if the sender receives three duplicate acknowledgements for the same packet, then it retransmits the packet without waiting for the timeout. Entering fast retransmit, it sets the ssthresh to half of the current cwnd and decrements the cwnd to 1. This leads to TCP entering the slow start stage and the cwnd increments exponentially for every ACK it receives for sent packet until the ssthresh is reached. Thereafter, TCP enters congestion avoidance where cwnd increments linearly for every received ACK. The window size is taken as minimum of (congestion wnd, advertised wnd) .

B. RENO

Reno has same congestion control algorithms as Tahoe with an addition of fast recovery. In fast recovery, ssthresh and new cwnd is set to half of the current cwnd instead of setting the cwnd to 1. Thus Reno skips slow start and directly enters congestion avoidance.

C. NEW RENO

New Reno was introduced as Reno does not perform well when there are multiple packet drops in the same window. Unlike Reno, New Reno does not come out of the fast recovery unless and until it receives ACKs for all the packets that were present in the window while entering fast recovery.

D. VEGAS

Vegas implements congestion avoidance rather than first detecting the congestion and then taking steps to decrease congestion on the channel. Vegas basically calculates a base RTT and compares it with RTT of packet with recently received ACK. If compared RTT is much smaller than base RTT it increases its sending window and if RTT is greater than the base RTT, it decreases its sending window.

E. SACK

SACK is an extension of Reno. In SACK, selective ACKs are done rather than cumulative ACKs. Each ACK has a section which contains the sequence number of packets that have been acknowledged. When TCP enters fast recovery,

SACK implements a parameter named “pipe” which represents the estimate of the number of unacknowledged packets in network. The congestion window is reduced to half of the current window. For every ACK received pipe is decremented by 1 and for every packet retransmitted pipe is incremented by 1. If there are no un-ACKed packets in the network a new packet is transmitted. Thus, using SACK multiple packets can be retransmitted in just one RTT.

Manuscript received April 5, 2016
Manuscript revised April 20, 2016

2. Methodology

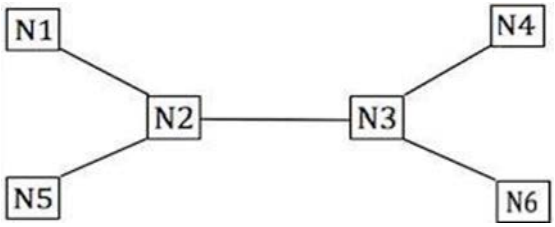

Figure 1. Network topology

Topology in Fig. 1 have nodes connected using full-duplex links with a bandwidth of 10 Mbps. An Unresponsive UDP flow i.e CBR is implemented with its source at node 2 and sink at node 3.
Under various conditions, this topology is run in NS-2 to generate trace files. These trace files are parsed and studied to analyze the behaviour of above mentioned TCP variants with respect to delay, throughput, packet drop rate, fairness and different queueing algorithms.

A. Experiment 1
In this experiment we analyze the performance of TCP variants under varying load conditions. Only one TCP flow is set from node 1 to node 4, linearly varying the rate of CBR flow from 1 Mbps to 10 Mbps. This is performed for different variants: Tahoe, Reno, New Reno and Vegas under following test conditions to create randomness:

a) Start and end CBR and TCP flows at once
b) Vary start and end times of both flows
c) Start CBR flow once TCP is stable
d) Start CBR flow when TCP in slow start

1. Throughput is analyzed over these paramters:
   a) T- test: It is relative comparison of mean and variance of two different sets of values. It tells us about how stable a set of observations are with respect to the other.
   b) Variance: It describes the stability in its own set of observations. Thus it helps to determine the variant with most stable throughput, the one with the least variance.
   c) Mean: Mean was calculated to describe the average throughput of a specific variant. Higher the calculated mean, Higher is its average throughput for entire range of CBR bandwidth.

2. Latency
Average latencies are calculated for each variant and compared. This helps us to understand the variant with least overall latency as well as for a specific bandwidth.
3. Packet Drop Rate
Number of dropped packets are calculated over the total number of packets. It is considered to find variant with minimum drop rate.

B. Experiment 2
This experiment is performed to see if one TCP variant is fair to another TCP variant or not. One TCP flow is set from node 1 to node 4 and the other flow from node 5 to node 6. The rate of CBR flow is varied linearly from 1Mbps to 10 Mbps.
The fairness of variants to one another is analyzed by plotting graphs for throughput, latency and drop rate for each flow with respect to CBR flow rate.

C. Experiment 3
In this experiment we will study the effect of queuing algorithm like DropTail and Random Early Detection on the throughput and delay of TCP Reno and TCP SACK. The same topology as in experiment 1 is used to carry out the experiment.

3. Analysis

A. Experiment 1
Graphs of throughput, latency and drop rate are plotted over rate of CBR flow in order to analyze the performance of each TCP variant.
1. Throughput
T-test was done for all possibe combinations of variants. T-values were caluculated to determine the stability of one variant over another.
T-values were calcultaed using the formula:

$$t = \frac{Mean_1 - Mean_2}{\sqrt{\frac{Var_1}{N_1} + \frac{Var_2}{N_2}}}$$

T-values for all conditions and every combination were studied and hence we concluded that T-values of Vegas are almost always greater than zero when compared with any other variant.
Thus, Vegas gives more stable throughput over any other variant under same conditions.
T-test values for Vegas uner test condition:
1. CBR flow starts TCP gets stable

Vegas over Tahoe:	0.350
Vegas over Reno:	0.821
Vegas over New Reno:	0.2496

Formula used for variance:

$$Variance = \frac{\sum_{n=1}^{N} (x_i - x)^2}{N}$$

Standard Deviation = (Variance) 0.5
Vegas also gave the best mean and the least value of variance and standard deviation for almost every experiment. Thus concluding that Vegas has the least deviations of throughtput over the entire experiment run.

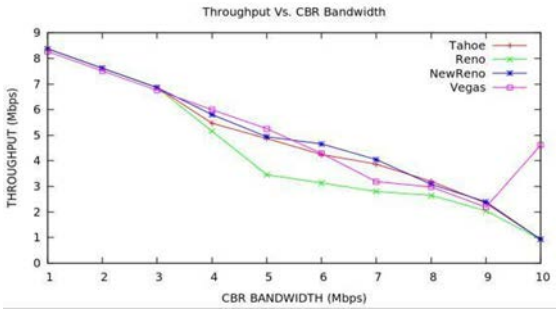

Figure 2. Throughput of variants over CBR rate

As seen from the Figure TCP Vegas initially starts with smaller throughput value than other variants since at first it calculates the base RTT which helps it figure out how much bandwidth is available to it. TCP Tahoe, Reno and New Reno are in slow start.

As bandwidth of CBR increases, the throughput of Tahoe decreases since when packet drop occurs, TCP enters slow start stage and reduces the congestion window to 1. Its congestion window will not grow unless it receives ACKs for sent packets and hence remain in slow start for longer duration of time.

The overall throughput of Reno decreases because it enters and exists the fast recovery mode for each packet in case of multiple drops in one window.

The New Reno throughput decreases a little but shows better performance under high congestion since it stays in the fast recovery stage unless and until it receives ACKs for all the packets that were present when New Reno entered the slow start stage.

Thus, we conclude that Vegas gives the best average throughput giving the better mean, lowest variance and relevant T-values over any variant.

2. Latency

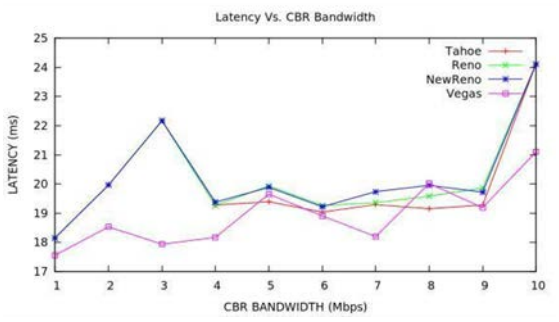

Figure 3. Latency of variants over CBR rate

As Vegas detects congestion[4], it will queue very small number of packets when the estimated RTT of recently received packet is greater than the base RTT. Thus, Vegas has the least latency.

In Tahoe, packet loss is detected only when the retransmission timer expires. Tahoe uses GO-BACK_N windowing technique. Thus if a packet drop occurs, then the number of packets retransmitted during congestion increases leading to larger queuing delays.

New Reno takes one RTT to detect each loss if multiple packets are lost in on window. This leads to increase in queuing delays and thus high latency.

The cwnd of Reno is decremented multiple times for multiple losses in the same window[1]. Since the window size becomes small, it slides over the queue very slowly unless it finally comes out of fast recovery leading to increased queuing delays.

3. Drop Rate

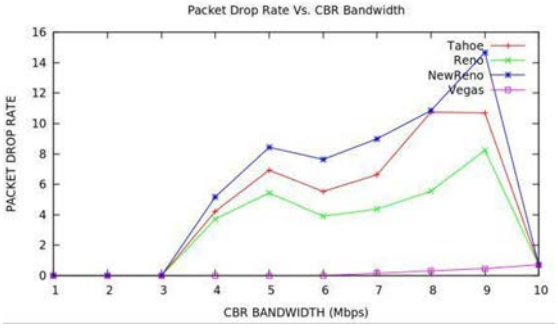

Figure 4. Drop rate of variants over CBR rate

Vegas detects congestion in its initial stage by calculating the base RTT and comparing it with RTT of every packet it sends. Hence, Vegas has the least drop rate as compared to all other variants.

Analyzing all parameters in the experiment, we conclude that Vegas has the highest average throughput, lowest average latency and minimum drop rate for almost every test condition. Though for low CBR rate and bottleneck CBR rate, the throughput of all variants becomes same but still the relative latency and number of packets drop is still very small for TCP Vegas.

B. Experiment 2

The fairness of variants to one another is analyzed by plotting graphs for throughput, latency and drop rate for each flow with respect to CBR flow rate.

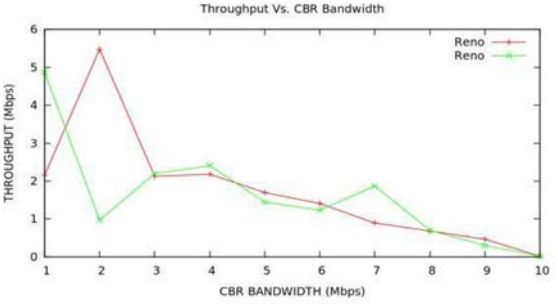

Figure 5. Fairness of Reno with Reno

Fig 5. shows that both the TCP Reno flows are fair to each

other in terms of bandwidth. This is because both the flows follow the same algorithm of congestion control. The throughput of both the TCP Reno flows decreases in a similar fashion under congestion.

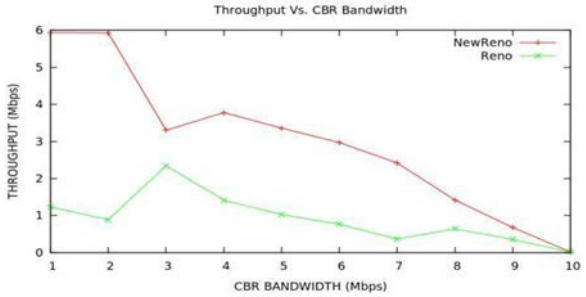

Figure 6. Throughput of Reno, New Reno

But, TCP New Reno is unfair to TCP Reno. This is because New Reno can handle multiple packet drops when congestion increases in the network. New reno does not come out of the fast recovery phase unless all packets present at the time when it entered fast recovery phase are acknowledged.

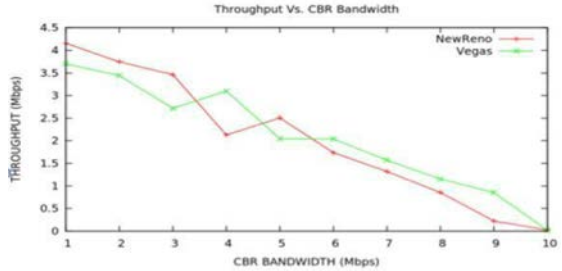

Figure 7. Fairness of New Reno and Vegas

Figure 7. shows that New Reno is unfair to Vegas. This is because as the New Reno traffic in the network increases, TCP Vegas detects congestion and reduces the number of packets it sends into the network, thus giving the network bandwidth to the other flow. But as the cbr traffic in the network increases the throughput of New Reno reduces as the number of packet drops increases. But at the same time the throughput of Vegas improves as it's a congestion detection mechanism.

It can be seen that both Vegas flows are fair to each other. When one of the TCP Vegas flow say Vegas_1 detects the occurance of congestion in the network it immediately backs-off by reducing the number of packets being sent on the network. Thus, the other TCP Vegas flow say Vegas_2 utilizes the bandwidth of the network and vice-versa. Therefore, we can see throughput of both the flows growing and then falling down. But the overall throughput of both the flows are almost the same.

C. Experiment 3

The throughput of TCP SACK and Reno are plotted in the same graph with CBR flows so as to better analyze their responses with RED and Droptail queuing algorithms.

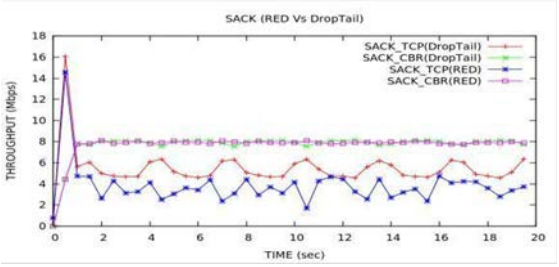

Figure 8. Thoughput of SACK with RED, Droptail

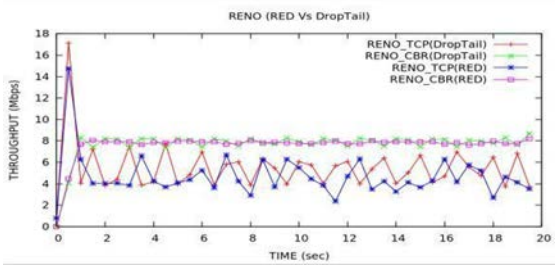

Figure 9. Thoughput of Reno with RED, Droptail

As seen from Fig. 8 and Fig. 9, the throughput of SACK (RED) and Reno (RED) is smaller than the throughput of SACK (Droptail) and Reno (Droptail). The reason is that in the RED algorithm packets are dropped on the basis of statistical algorithms. Three parameters are set as min, max and burst[3]. The packets dropped from a particular flow is proportional to the amount of network bandwidth utilized by that flow[3]. On the contrary, Droptail algorithm starts to drop packets irrespective of the flow type when its queue gets completely filled. Thus it shows that RED is more fair than Droptail to a respective network flow.

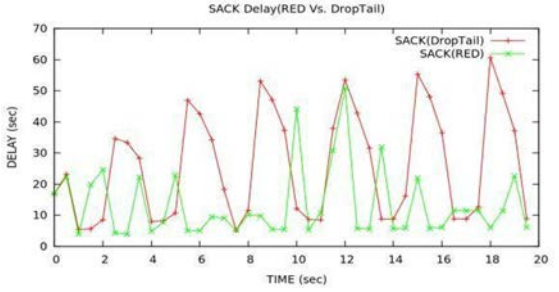

Figure 10. Delay of SACK with RED, Droptail

As seen from the fig. 10, the delay when Droptail algorithm is used is greater than when RED is used. This is because RED defines a limit over how much bursty traffic is to be allowed in the queue unlike droptail.

Thus, it can be seen from the Fig. 8 and Fig. 9 that TCP has very high throughput when there is no CBR (UDP)

flow in the network. The CBR flow does not take congestion into account as it just bursts the packets irrespective of congestion in the network. Therefore when there are TCP flows along with CBR flows (UDP), then the TCP flow backs-off and reduces its window size. Thus, TCP reduces its sending rate with an occurance of congestion.

In fig. 8, SACK when used with RED yields a much lower throughput than when SACK is used with Droptail. Also, SACK throughput is not stable at any point when RED queuing algorithm is used for a test condition. Thus, it is not a good idea to use SACK with RED queuing algorithm.

4. Conclusion

Different TCP variants have different performance characteristics in terms of throughput, latency and packet drop ratio under varying degree of congestion in the network.

In Experiment 1, depending upon the T-test, mean, variance and simulation analysis, we can conclude that TCP Vegas performs the best in terms of throughput, latency and packet drop rate for most of the times, specially under high congestion conditions in the network.

In Experiment 2, we can conclude from the simulation analysis that when two flows use the same TCP variant they are fair to each other. But when two different TCP flows are running at once, they are not fair to each other in terms of bandwidth utilization.

In Experiment 3, we observed from the simulation analysis that the TCP variants perform better in terms of throughput when DropTail queuing algorithm is used but gives small latency periods when RED queuing algorithm is used. Thus use of Droptail and RED is a trade-off between throughput and latency.

References

[1] A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas http://inst.eecs.berkeley.edu/~ee122/fa05/projects/Project2/SACKRENEVEGAS.pdf
[2] Fair comparisons of different TCP variants for future deployment of Networks http://www.academia.edu/183250/Fair_comparisons_of_different_TCP_variants_for_future_deployment_of_Networks
[3] Random Early Detection http://tldp.org/HOWTO/Adv-Routing-HOWTO/lartc.adv-qdisc.red.html
[4] TCP Vegas http://en.wikipedia.org/wiki/TCP_Vegas
[5] Ahsan Habib, Bharat Bhargava
[6] Unresponsive Flow Detection and Control
[7] using the Differentiated Services Framework https://www.cs.purdue.edu/homes/bb/unresp.pdf
[8] Steven H. Low, Fernando Paganini, Jiantao Wang, Sachin Adlakha, John C. Doyle,
[9] Dynamics of TCP/RED and a Scalable Control