Performance Analysis of TCP Variants

Gauresh Pandit
College of Computer and information Science,
Northeastern University,
Boston, MA
NU id: 001945331

Anish Deshmukh
College of Computer and information Science,
Northeastern University,
Boston, MA
NU id: 001943901

***Abstract*- TCP is one of the most widely used protocol and amounts to a majority of Internet traffic. Over the years many different variants of TCP have been developed and deployed in the real world. In this paper we focus on comparing performance of mainly four TCP variants Tahoe, Reno ,NewReno, and Vegas. The performance and effect of different queuing algorithms on TCP traffic is also measured. With the aid of network simulator NS-2 , we introduce a CBR flow which acts as background traffic on a network. The aim is basically to do three small experiments in which first will be a comparison between TCP variants and to calculate on various performance parameters like Throughput, Drop rate, Latency etc. Secondly we analyse how two flows of different TCP variants fair against each other in presence of some traffic. These are then evaluated and analysed on the basis of simulation results rendered by the behaviours through graphical representation. Finally we show the distinction between different Queuing disciplines by assigning different TCP variants and comparing the results. This analysis may be useful for selecting appropriate TCP variants in different network conditions.**

I. Introduction

TCP has been working over the Internet for more than 2 decades and is an inherent part of the internet now. It is now one of the core protocols of the Internet protocol suite and is so common that the suite is called TCP/IP. TCP was inherently a reliable protocol but did not provide acceptable performance over large and congested networks. The main goal of this paper is to study and analyse how TCP variants like Tahoe, Reno, NewReno, Vegas work. It also helps to understand which variant works well in a particular situation and for what functionality or role required. The Research also helps to in depth analyse the currently running TCP on the internet and different possible variants which could be used as an alternative.

To study these variants we use several approaches mainly comparing each on different congestion levels on several parameters like Throughput, Packet Drop, Latency, Average bandwidth and Congestion Window. These help to analyse which variant gives better performance for different kinds of parameter in the same set up environment.

Secondly we study how TCP variants work in presence of each other and whether fairness is achieved among them or not. For this approach we analyse different TCP variants in combination with one another. We simulate working of different protocols with each other on parameters like throughput, latency and packet drop rate to determine fairness.

Lastly we determine the running of TCP in comparison with UDP on time intervals. We introduce UDP after the TCP gets constant and find the bandwidth consumption for the same. We compare two queuing algorithms namely DropTail and RED with Reno and Sack TCP variants to analyse the better queuing algorithm in accordance with these variants.

II. Methodology

The aim of this research was to analyse the performance of different TCP variants under changing and varied network conditions. To achieve this, following experiments were performed.

1. TCP Performance Under Congestion
2: Fairness Between TCP Variants
3: Influence of Queuing

We used the following tools to perform simulations on the network topology and extract the results from these simulations:-

1. NS2 :- A Network Simulation tool developed mainly for research and teaching and is a widely used simulation tool developed in UCB. We used ns-2 to run the simulation on the given topology and record the events happening in the network. Those events were then analysed to understand the network behavior.

2. NSG 2.1 (Network Scenario Generator) :- A Scenario generator which we used for generating topologies, creating flows of UDP and TCP and assigning different agents. This tool created TCL scripts for ns-2.

3. MS Excel :- We used Excel to plot the graphs using the data which we obtained from the simulations.

4. AWK Scripting :- AWK is an interpreted programming language designed for text processing and typically used as a data extraction and reporting tool. We used awk for parsing through simulation trace files generated by ns2 to calculate several results like throughput, latency etc.

All the experiments were conducted using the same network topology for all set of operations performed. The topology is as shown in Figure 1.

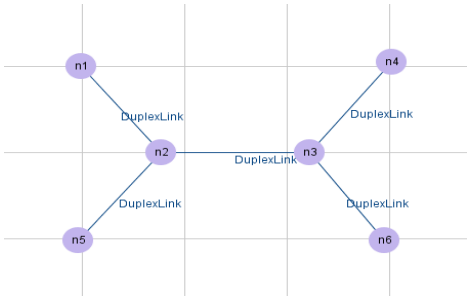

Figure 1

Our topology consisted of 6 nodes which were linked to each other using duplex-links of 10mbps. Two kinds of traffic sources were used; TCP flow of different TCP variants and a CBR flow for generating background traffic. FTP was used as TCP stream and an unresponsive UDP as the CBR flow. Different performance parameters like throughput, packet drop rate, and latency of the TCP stream was measured while varying the bandwidth of the CBR flow. The bottleneck link has the capacity of 10Mbps, a propagation delay of 10ms and a drop tail queue. Later, the queuing algorithm was also changed to RED(Random early detection) and its influence on the TCP connection was measured. The simulations were run for different values of the bandwidth of the CBR flow ranging from (1-10) for a period of 100 seconds and statistics such as total number of packets sent, number of packets dropped, RTT for each packet etc were recorded from the trace files. In addition, based on these values the throughput, packet drop rate, and latency of the TCP connection was calculated and analysed.

III. TCP Performance Under Congestion

In this experiment the performance of different TCP variants was analysed under load conditions. A simple simulation scenario was used were we added a TCP stream from source at Node1 to sink at Node4 and a CBR flow from source at Node2 to sink at Node3 where one UDP and one TCP share a bottleneck link. The simulation was run for different values of bandwidth of CBR flow and various performance quantities like the throughput, packet drop and latency of the TCP connection was calculated for different TCP variants (TCP Tahoe, Reno, Newreno and Vegas).The following simulation parameters were used:

SIMULATION PARAMETERS:

Access-link bandwidth: 10 Mbps
Access-link delay: 10 ms
TCP type: TCP Tahoe, Reno, Newreno and Vegas

Throughput:

The figure below shows the comparison of throughput of each variant as a function of the CBR rate. The figure depicts that TCP-Newreno has a higher overall throughput as compared to the other three variants. Initially, the throughput of Tahoe, Reno and Newreno is similar as all the three start from the slow start mechanism. But for Vegas it is slightly lower because it starts from the modified slow start mechanism where its rate of increasing its window size in slow start phase is one half of that in other variants.

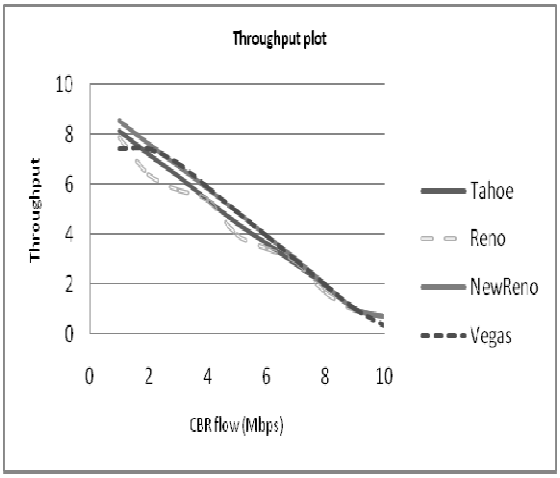

Figure 2

As the CBR rate is increased to around half the bandwidth of the link each variant behave according to their congestion avoidance algorithms which leads to different throughput values for each variant. Tahoe enters into slow start mechanism on detecting a packet loss and reduces its window size to 1 causing the low throughput. Reno and Newreno enters into the fast retransmit and fast recovery mode on detecting packet loss and reduces the window size by half. The difference between them is that Newreno triggers a retransmit on receiving one duplicate ack where as Reno waits until it receives three duplicates ack's.

Reno is not effective in detecting multiple packet loss and may cause the window size to be reduced twice for packets lost in one window causing the lower throughput. But for Vegas since no packet loss occurs, its size of congestion window remains

the same and results in higher utilization of the bandwidth.

In conclusion TCP Newreno has the highest throughputs among the others as it prevents many coarse timeouts and its congestion avoidance algorithm causes fewer retransmissions. TCP Vegas has high and comparable throughput to Newreno as it detects incipient congestion very accurately which causes fewer packet drops that leads to fewer retransmissions.

Packet Drop Rate:

To analyze the packet drop behaviour, we varied the bandwidth of the CBR flow and counted number of dropped packets at that bandwidth for TCP variants. From the data obtained from simulation we plotted the following figure.

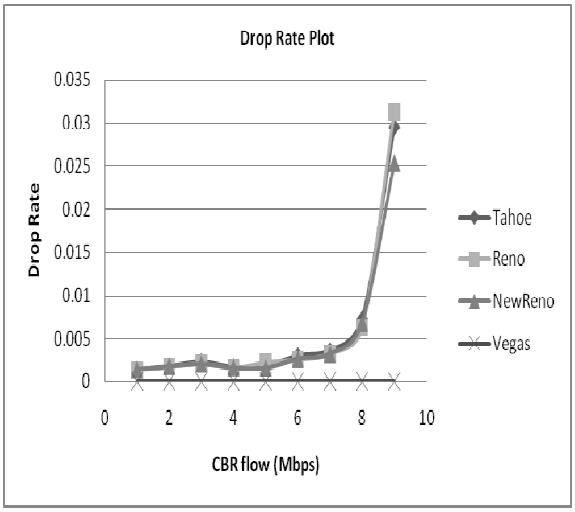

Figure 3

The figure clearly depicts that the drop rate of TCP Vegas is very less (almost zero) in comparison to Tahoe, Reno and Newreno. This behaviour is observed because Vegas detects congestion at an early stage based on increasing Round-Trip Time (RTT) while the other variants detect congestion only when packets gets dropped. As the CBR rate is increased, it starts utilizing more and more bandwidth and the TCP flow starts getting affected. As a result it takes more time for a packet to reach to the destination and the acknowledgment to arrive. TCP Vegas detects this change in the RTT. If observed RTTs become large, TCP Vegas recognizes that the network is becoming congested, and reduces the window size. If RTTs become small, the sender host of TCP Vegas determines that the network is relieved from the congestion, and increases the window size again. On the other hand the other TCP variants keeps increasing their window size until they detect a packet loss. When a packet loss is detected Tahoe, Reno and Newreno reduces their window based on the congestion avoidance algorithm.

Since Vegas detects congestion at an early stage by accurately measuring the available bandwidth unlike the other variants it starts sending less number of packets hence doesn't contribute to the congestion and reduces the number of dropped packets.

Latency:

To calculate latency the RTT(Roundtrip Time) of each packet sent was recorded and then cumulative latency was calculated based on total number of packets sent during the time the simulation was run.

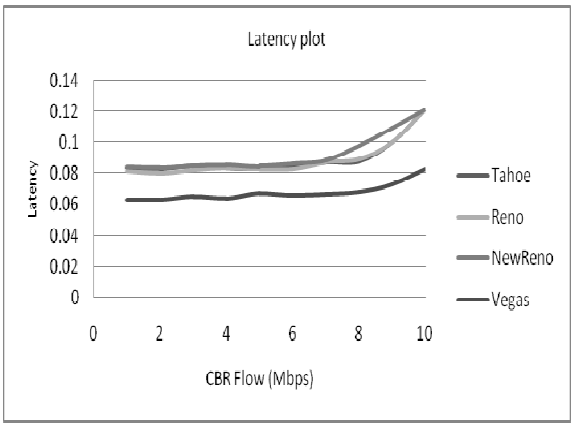

Figure 4

As seen from the figure 4, TCP Vegas has the lowest average latency. The other three variants have comparable and high latency than TCP Vegas. The other variants keep on increasing their window size until they detect a packet loss after which they enter into congestion avoidance mechanism. Since they react only after congestion has started to occur the packet drops tends to increase which in turn causes the latency to increase. But as Vegas reduces packet loss by carefully matching the sending rate based on the RTT required for packet delivery, it doesn't add to congestion and thus reduces latency.

IV. Fairness Between TCP Variants

This experiment is basically a comparison of TCP Variants in combination with each other to check fairness of the algorithms in presence of one another. We analyse the fairness of protocols based on three parameters basically Throughput, Packet Drop Rate & Latency.

The results show that many of the TCP Variants show fairness in throughput when used in combination with each other. The Reno works well with itself as another variant and gives oscillating throughput but a overall good average latency.

As shown in figure 5 NewReno gives better throughput when compared to Reno. One of the major reasons for the same is that in NewReno

during fast retransmit 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 transmission window full. Unlike Reno where the retransmission is done after every 3 duplicates ACKs that are received.

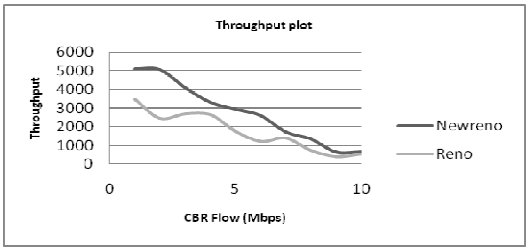

Figure 5

Considering TCP Vegas in comparison with TCP Vegas performs very fair and gives approximately the same throughput over increasing traffic and contention. TCP Vegas though does not cause congestion in the network and thus does not consume all available bandwidth as it prevents packet drops. Thus the average throughput for both remains the same but does not go as high.

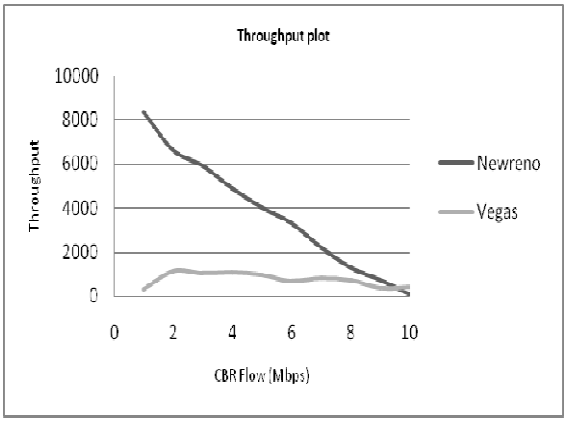

Figure 6

The last result of comparison between TCP NewReno and TCP Vegas as shown in figure 6 tell how badly Vegas performs in presence of other variants. The NewReno is a very aggressive as it retransmits as soon as it receives the first ACK and the throughput of TCP Vegas goes very low as if it detects congestion on the stream it lowers the window size. Vegas detects congestion by having a check on the RTT required, which increases as NewReno takes up most of the bandwidth after increasing exponentially after a slow start. Thus, Vegas never gets much of the bandwidth for most of the time and the throughput for TCP Vegas remains low in presence of the aggressive TCP NewReno.

In Packet Drop ratios every variant performs very similar to every other variant except when it comes to TCP NewReno and TCP Vegas.

The other TCP variants have similar values when it comes to packet loss as the AIMD process is the same for all other algorithms and thus similar packet drops are visible after they hit congestion. But when TCP Vegas comes into consideration the packet loss is much less as when considered with TCP NewReno as shown in figure 7.

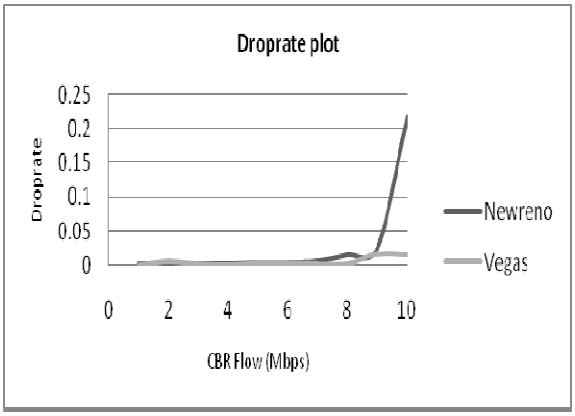

Figure 7

The main reason for this is that NewReno being very aggressive hits congestion more times thus having more packet losses at the same time Vegas being more passive as discussed above drops less packets. As you can see in the figure the distinction is visible more at higher traffic (CBR) values this is because when there is lesser bandwidth TCP NewReno hits congestion faster after slow start and fast retransmit losing more packets.

When it comes to latency the TCP Variants again perform quite well and act as fair contenders. The latency comparison for TCP NewReno and TCP Vegas though is not fair again.

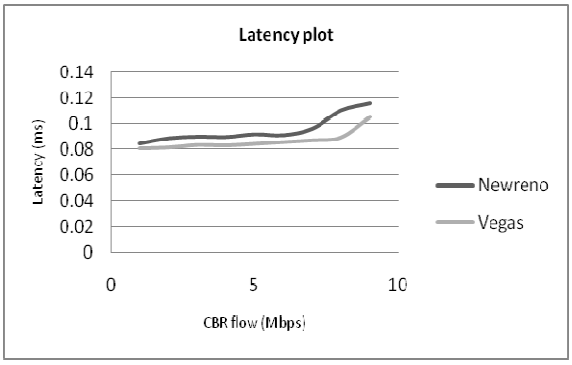

Figure 8

The TCP Vegas has again lesser delay in comparison. One of the reasons for this is Vegas sends packets only if the RTT time is low enough to send a packet. It checks for the RTT each time even during slow start till it crosses a desired threshold value and then increases slowly depending on the RTT. Hence the latency will always be less as the congestion window value for Vegas is very less and

thus having lower latency values in comparison to TCP NewReno.

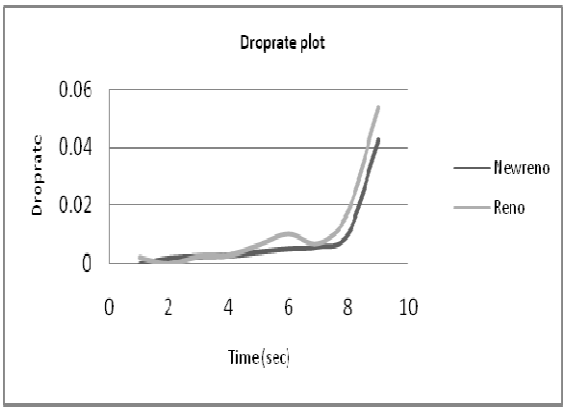

Figure 9

As we can see in the figure above, when Reno and NewReno are compared Reno gives lesser packet drops and lower latency values. This is majorly because NewReno hits congestion similar to the way Reno does the only difference is the approach for retransmission.

V. Influence of Queuing

This experiment shows the throughput given by the use of two different queuing disciplines DropTail and Random Early Drop (RED). Queuing disciplines on two different algorithms are used namely Reno and Sack to check their performance. The set-up for the experiment has a TCP flow flowing from node N1 -> N4 till it gets constant after which a CBR (UDP) is introduced flowing from nodes N5 -> N6. On creation the CBR consumes the whole allotted bandwidth in this case 6 Mbps and thus the TCP receives several packet drops due to which it faces timeout and slow starts again. It keeps climbing trying to reach the available bandwidth which is now 4 Mbps in this scenario.

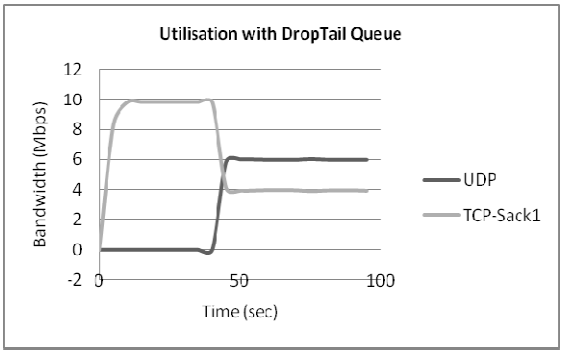

Figure 10

The figure 10 shows the performance of TCP Sack with DropTail Queuing discipline. Near complete throughput is achieved on using DropTail as seen in the figure. DropTail queue follows the FIFO algorithm and drops packets after the queue gets full. Thus few packets of Sack get dropped giving a partial ack from the receiver on every such drop. According to the implementation of Sack it will retransmit the packets and increment its pipe until it gets a second consecutive duplicate ack. The probability of the packet to get delivered the second time gets higher as the dropped packets will be the first in order to be sent to the receiver. Thus the congestion window won't go into slow start as often giving maximum throughput which in this case near 5mbps.

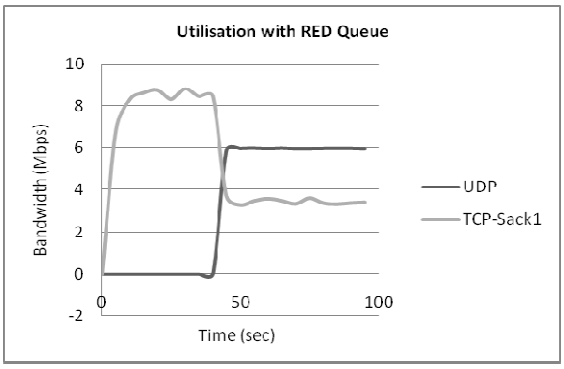

Figure 11

Unlike DropTail the performance of Sack with RED Queuing discipline is not as high. As you can see in the figure 11. The throughput for SACK over time is less than compared to TCP Reno as a variant. RED queuing discipline works on the basis of probabilistic drops where the probability of a packet being dropped is based on the count of the packets. Thus RED drops few packets when the Sack will have a higher window size before the queue gets completely filled. Also RED doesn't work well with misbehaving flows like UDP giving it similar throughput like when used with DropTail. UDP packets are also dropped slightly when compared to DropTail but that is a very trivial.

Both these queuing disciplines don't provide fair bandwidth when it comes to UDP (CBR) flow and TCP flow as the whole requested bandwidth is given to UDP. As seen in the above figures the UDP covers 60% bandwidth. The latency for RED is less than what is achieved with DropTail. One of the reasons being that in RED the speed of all flows is lowered before completely filling the queue and thus there are fewer very bursty drops. Thus the network is never very congested which reduces the latency.

VI. Conclusion

The study helped to give an in depth analysis how several TCP variants work in several competing parameters. It also showcased how the queuing disciplines worked on different TCP Variations and what discipline fared better. The first

experiment helped us know how Vegas had better performance over its other variants on the basis of parameters like throughput, packet drops and latency. Vegas nearly outperformed its other variants which was mainly experienced due to its intelligent calculation of window size which completely depends on the base RTT time calculation. In the second experiment the shortcoming of the very fine working Vegas was highlighted. Vegas did not perform well in accordance with other TCP variants giving lower throughput values because Vegas detects congestion on getting a lesser RTT value thus giving preference to the other algorithm. The analysis also revealed that NewReno was better when contending with the other TCP variants. TCP NewReno is more aggressive when it comes to getting more bandwidth and thus we also saw that it had more packet losses and more latency too. The second experiment thus conveyed that TCP NewReno seemed to be a better option as it worked better with itself and also performed fairly when it came to latency and packet drops. NewReno thus seems to be an overall winner in comparison with the TCP variants as contenders and also for combination specifically due to the caveat of Vegas performance with other variants. Thus if we have to deploy the TCP Variants on to the system NewReno with probably be the best choice causing lesser congestion having better performance and also having fairness in all terms. On the other hand if Vegas was to be deployed for a client it would cause huge performance degradation for the user while improving performance for other available variants in the system.

The last experiment exhibited DropTail queuing discipline to be better than RED in comparison with taking a single TCP flow into consideration. DropTail gave better performance with both Reno and Sack which were the used TCP Variants. RED having a probabilistic packet drop strategy drops packets before it reaches its maximum throughput levels thus giving less throughput and better latency over time.

Though we have now analysed how TCP and its variants work on the given network the conclusion may not stand true for wireless networks and one also needs to find which protocol maybe suitable for both wireless and wired connections. The experiments conducted were also done for individual TCP Variants but it may not necessarily mean that the same with many flows working together at a given time. Deploying these variants onto the internet and testing them in real world scenarios would gives us more in depth analysis of the same.

REFERENCES:

[1] Peng-Jung Wu "https://sites.google.com/site/pengjungwu/nsg".

[2] L. Qiu, Y. Zhang, and S. Keshav, *“Understanding the performance of many TCP flows,” Computer Networks,Nov. 2001*.

[3] Kevin Fall and Sally Floyd *"Simulation-based Comparisons of Tahoe, Reno, and SACK TCP"*.

[4] V. Jacobson and M. Karels. “Congestion Avoidance and Control.”

[5] Junsoo Lee , Joao˜ P. Hespanha, Katia Obraczka, Stephan Bohacek *"A Study of TCP Fairness in High-Speed Networks"*.

[6] Kenji Kurata Go Hasegawa Masayuki Murata " *Fairness Comparisons between TCP Reno and TCP Vegas for Future Deployment of TCP Vegas"*.

[7]M. May, J. Bolot, C. Diot, and B. Lyles, “Reasons not to deploy RED,” in IWQoS’99, 1999.

[8]The ns Manual (formerly ns Notes and Documentation), The VINT Project, a collaboration between researchers at UC Berkeley, LBL, USC/ISI, and Xerox PARC,Oct. 2000.