Performance Analysis of TCP Variants

Soham Aurangabadkar
College of Computer and Information Science,
Northeastern University,
Boston, MA
NUID: 001941585

Kapil Deshpande
College of Computer and information Science,
Northeastern University,
Boston, MA
NUID: 001915390

***Abstract*- TCP is the most widely used connection oriented transport layer protocol and handles high amount of traffic over the Internet. Over the years, extensive research has been done to improve its performance which resulted into number of variants of TCP. This paper mainly concentrates on evaluation of performance of four variants of TCP namely Tahoe TCP, Reno TCP, NewReno TCP and SACK TCP using Network Simulator (Version 2). We also used a UDP flow in the form of a Constant Bit Rate to control the congestion in the simulated network. We then performed three experiments that helped us evaluate all the specified variants by comparing various environment variables. The experiments focused in turn on direct comparison between the variants, the competitive nature of each variant when run at the same time, and analyzing the importance of two queuing disciplines. We achieve this comparison by analyzing the distribution of performance parameters like throughput, packet drop count, and the round trip time. We also prove that performance of each variant depends on some parameters of environment which in turn will help to select correct variant as per varied requirements.**

I. Introduction

The transport layer of the ISO-OSI model has been widely used in the last few decades, and TCP and UDP are widely used to provide variety of services. Internet being primarily a best effort delivery network, it is the responsibility of transport layer to ensure reliability, flow control and error control in transmission of data. The originally proposed protocol had some performance issues which resulted in a number of variants of TCP with different features. But it is still debatable which variant performs better than other and in which environment conditions.

In this paper we thoroughly analyze performance of TCP by setting up a virtual network using NS-2. We compare different variants using parameters like throughput, round trip latency, packet drop count and bandwidth. We further verify fairness and bandwidth consumption when two variants of TCP are used at a time on single link having constant bandwidth. We also check performance of TCP protocol when DropTail and RED queuing algorithm is used for optimization.
In Section II, we will briefly analyze the experiments conducted and methodology to analyze results of experiments performed on variants of TCP. From Section III to V, we will thoroughly study experiments and their results. We list out our conclusions and inferences in section VI.

II. Methodology.

The aim of our experiments was to analyze performance of four variants of TCP when used in different configurations of network. To perform this analysis we conducted three experiments as follows:

Experiment 1: TCP Performance under various amounts of bandwidth consumption in the network
Experiment 2: Fairness between TCP variants
Experiment 3: Influence of queuing

We conduct these experiments using the following tools:
NS: NS is a discrete event simulator targeted at networking research. Ns provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. We perform our experiments by building topologies of networks with different parameters like CBR flow, Bandwidth, Queuing algorithm etc., using nodes as senders and receivers, while setting various environment variables as 10s of milliseconds for delays, DropTail queues where queuing was not analyzed, and 10s of packets of queue size

Tcl Scripting language: Tcl has extensive support of networking libraries and supports number of methods which helps us to define different network topologies and perform experiments. We used oTcl, which allows us to define variables and use them with NS-2

AWK Scripting: We make use of AWK scripting language to parse content of trace file generated by NS2 to calculate mainly throughput, latency and drop count.

Network Topology Setup: To perform experiments we use following topology as sample model. We define 6 nodes connected to each other by duplex links. Link N2 and N3 will be analyzed each time for congestion different factor are varied, as the CBR source is mainly between these nodes.

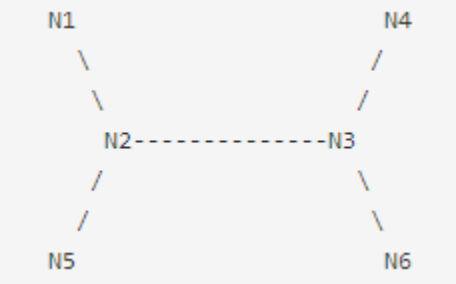

Fig.1: The experiment topology

III. TCP Performance under Congestion

In this experiment, we analyze the performance of the TCP variants under the influence of CBR. To study the behavior of the four TCP variants, we set up a network topology explained above. We add CBR source at N2 and sink at N3. We also set the advertised window of the receiver to 10000. Also we add TCP flows from N1 to N4. We vary the CBR rate from 1 MBPS to 10 MBPS so that we can achieve bottle neck condition in the link from N2 to N3. Using this setup we calculate throughput, latency and drop count of packets of different variants of TCP.

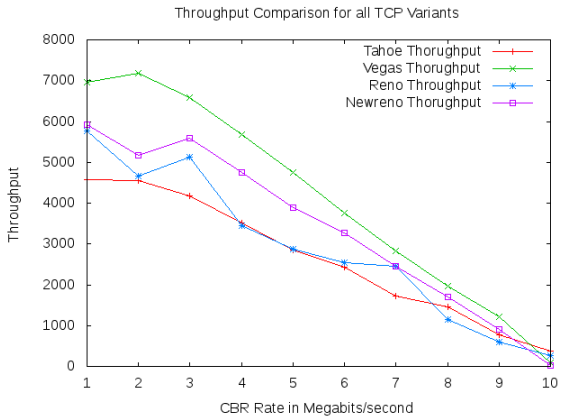

Fig. 2: Throughput (Y-Axis, kilobits/sec) versus CBR Rate (X-Axis, mbps)

Throughput:
The figure depicts throughput of different TCP variants (Tahoe, Vegas, Reno, and NewReno). We vary CBR from 1 to 10 Mbps to check congestion at common link N2 to N3 in above network topology. We ran the simulation for 60 seconds and started and ended both the flows simltaneously. Initially as CBR increases Reno and NewReno show almost the same throughput. We can see that Reno performs almost the same as TCP Tahoe over the complete duration of the experiment. Reno performs very well over Tahoe when the packet losses are small. But when we have multiple packet losses in one window then Reno doesn't do so. If there is multiple packet drop then the first info about the packet loss comes when we receive the duplicate ACK's. But the information about the second packet which was lost arrives only after the ACK for the retransmitted packet reaches the sender, introducing a delay of 1 RTT. TCP Vegas achieves better performance than TCP Reno since its bandwidth estimation does not rely on packet losses in order to estimate the available bandwidth in the network. It also introduces delay based congestion avoidance to further improve its estimation of congestion. TCP Reno and NewReno have similar behavior as both starts with fast transmission and fast recovery. The difference between Reno and NewReno fast recovery mechanism is, NewReno immediately triggers retransmission when it receives failure acknowledgment of single packet which reduces retransmission time whereas Reno waits for all the failure acknowledgements before triggering retransmission. Tahoe enters into slow start phase by reducing its congestion window to 1 which reduces its throughput. Since in case of Vegas, no packets have been dropped, throughput remains high.

As a conclusion we can say that throughput of Vegas is maximum as its congestion avoidance algorithm causes very few retransmissions, because of a highly accurate calculation of round trip time for the network.

Packet Drop Rate:
In this experiment, we vary CBR flow from 1 Mbps to 10 Mbps to check drop count of packets when congestion increases. The advertised window of the receiver is also set to 10000. From the graph below, it is clear that TCP Vegas has the lowest drop count.

Vegas, unlike other variants detect congestion at a very early stage using Round Trip Time. As congestion increases, time to taken by acknowledge to reach back increases dramatically which Vegas uses as a signal and reduces its window size. As RTT

decreases, Vegas predict that network is getting relived from congestion and increases its window size. On the other, other variants of TCP Reno, NewReno and Tahoe when detected congestion in the network initialize congestion avoidance algorithm because of which drop count is higher in all three cases.

Since TCP Vegas accurately detects RTT, its drop count is far less than others and doesn't contribute much to congestion happening in network.

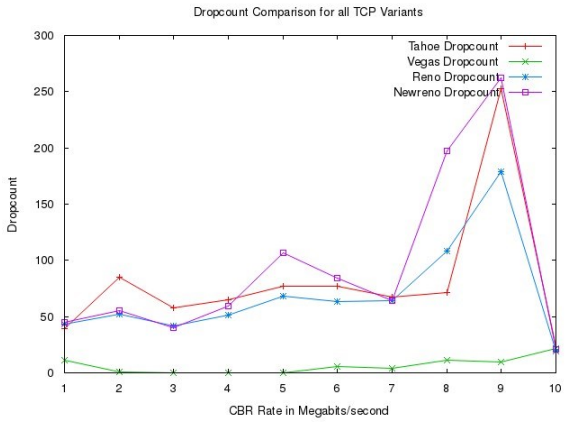

Fig. 3: Packet drop count (Y-axis) versus CBR Rate (X-Axis, mbps)

Latency:
In case of latency we record round trip time of each packet. Based on the analysis of number of packets we calculate average latency. Since TCP Vegas predicts congestion based on RTT, it reduces its window as soon as it detects congestion. Therefore in case of Vegas average latency is lowest when compared to other variants.

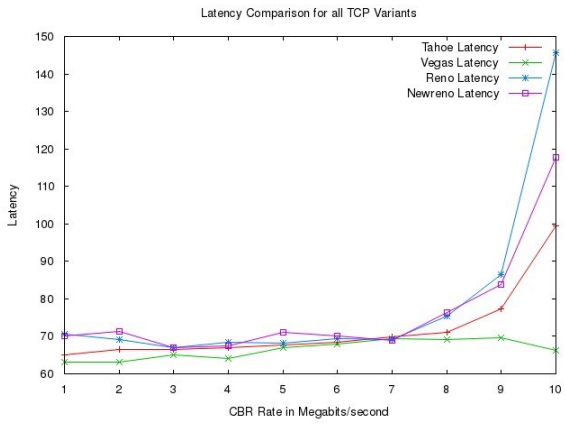

Fig. 4: Round trip time(Y-Axis, milliseconds) versus time (X-Axis,)

Thus we can conclude that the Vegas variant gets the better average throughput, the lowest overall latency and the fewest number of packets dropped. We can conclude that Vegas is the variant that gives overall better performance in this experiment.

IV. Fairness of TCP variants

In this set of experiments, we analyze performance of TCP variants in terms of bandwidth utilization when used in combination. For this we set up one CBR stream N2 to N3. We also add TCP stream from N1 to N4 and N5 to N6. Now we use series of combinations of variants to analyze the performance in turn analyzing the fairness.

NewReno and Reno:
In case of NewReno and Reno, throughput of the NewReno is greater, since it triggers immediate retransmission of packet when received partial acknowledgement unlike Reno. So as depicted in below graph as congestion increases throughput of Reno decreases. So NewReno is being unfair in terms of Bandwidth Congestion.

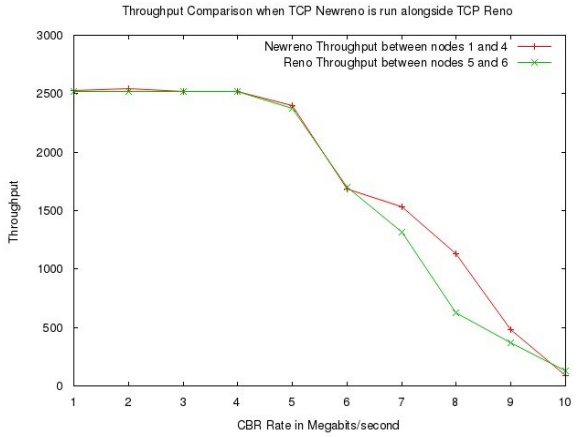

Fig. 5: Throughput(Y-Axis, kilobits/sec) Versus CBR Rate(X-Axis, mbps)

NewReno and Vegas:
In case of TCP NewReno and TCP Vegas, TCP Vegas performs very poorly. TCP NewReno aggressively triggers retransmissions it consumes more bandwidth. In contrast Vegas determined congestion in time based on RTT and hence reduces its window size reducing packets. So we can conclude that, when we will use TCP Vegas against any variant Vegas will perform poorly against other.

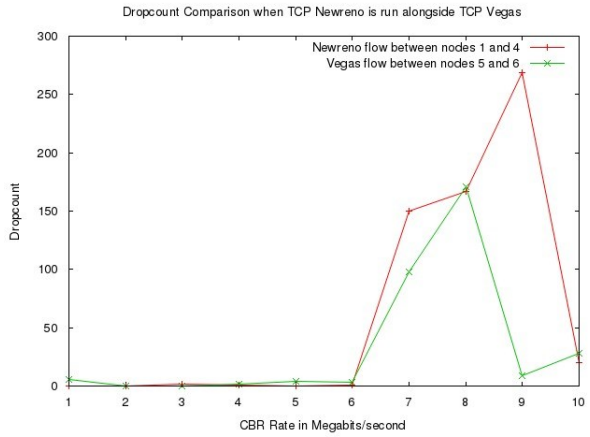

Fig. 6: Packet drop count (Y-Axis) versus CBR Rate (X-Axis, mbps)

Vegas and Vegas:

When Vegas/Vegas are used to check fairness of bandwidth consumption, both perform very fairly as both predicts congestion intime so consumption is fair when congestion occurs. Similar is case with Reno/Reno. Based on this analysis we can conclude that when two same variants are used against each other Bandwidth consumption is fair.

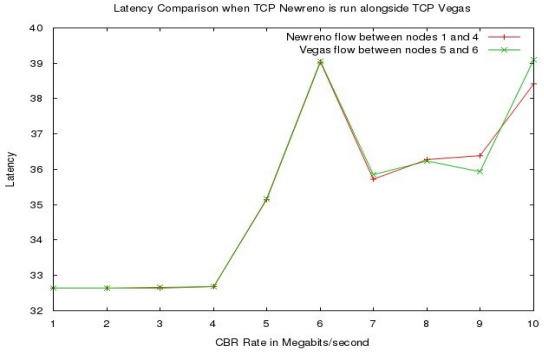

Fig. 7: Latency(Y-Axis, milliseconds) versus CBR Rate (X-Axis, mbps)

In case of latency, variants of TCP perform quite well and remain fair to each other except n NewReno vs Vegas. In case of NewReno vs Vegas, since Vegas estimates its window size based on its RTT value latency of Vegas is less compared to NewReno. As RTT value increases, latency of Vegas increases but still remains below NewReno.

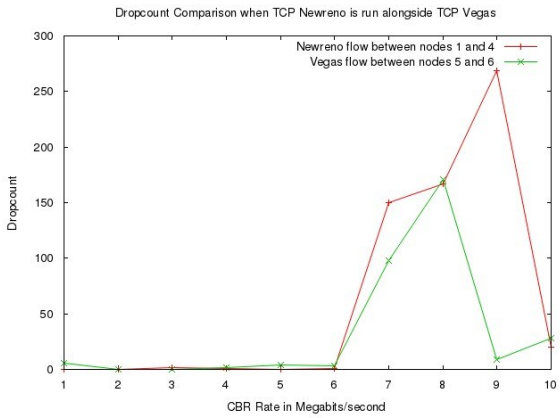

Fig. 8: Packet drop count(Y-Axis) versus CBR Rate (X-Axis, mbps)

In case of drop count, again every variant perform fairly except TCP NewReno and TCP Vegas. In case of TCP Vegas, since it uses very passive approach by reducing its congestion window in time, packet loss is very low. In case of NewReno since it uses very active approach by rapid retransmission, packet drop count is very high. Hence when CBR increases bottleneck arrives and this leads to higher drop rate in NewReno.

V. Influence of Queuing

In this experiment we check throughput of queuing algorithms RED (Random Early Drop) and DropTail when used with TCP variants TCP Reno and TCP SACK. To conduct this experiment we set up TCP flow from N1 to N4 and CBR UDP flow from N5 to N6 having bandwidth consumption limit at 7 MB. Initially from the start of time till five seconds TCP is allowed to steady itself. Thus we see that CBR throughput, while initially zero increases with time, dominating TCP as time progresses. As CBR flow starts it climbs up quickly consuming 7 Mb of bandwidth on the connecting link as per the setup.

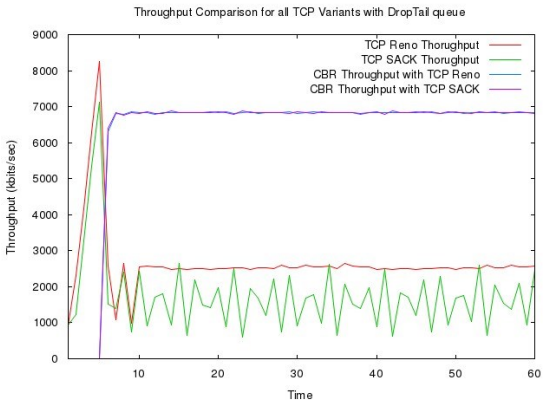

Fig. 9: Throughput(Y-Axis, mbps) versus time (X-Axis, seconds)

The above figure shows performance of TCP Reno and TCP SACK when the DropTail queuing algorithm is used. As per DropTail queuing algorithm once the FIFO queue used in TCP is filled with packets it will drop the packets till room is available for new packets. It is probably the simplest queuing algorithm. So when using TCP SACK few packets gets dropped. As per implementation, TCP SACK will retransmit the packet and until it gets second consecutive duplicate acknowledgement. So probability of second packet which will remain ahead in dropped packet is high. So congestion window will not go in slow start giving maximum throughput of 7000 Kbps

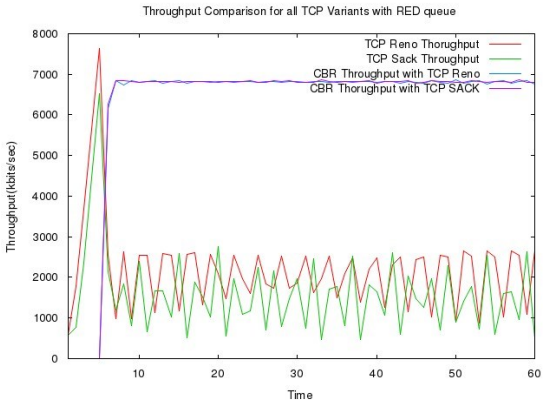

Fig. 10: Throughput(Y-Axis, milliseconds) versus CBR Rate (X-Axis, mbps)

The above graph shows the performance of the included TCP variants when the Random Early Drop queuing algorithm is used. CBR dominates the bandwidth later on as expected. The average throughput for TCP SACK over time as compared to TCP Reno is very less. The goals of the RED algorithm are to reduce queuing delay and packet loss, to maintain high link utilization, to better accommodate bursty sources, and to provide a low-delay environment for interactive services by maintaining a small queue size. It is an active queue management mechanism, and operates on the basis of statistical probabilities while deciding the dropping of packets. If the buffer is empty, all incoming packets are acknowledged. As the queue size increases, the probability for discarding the packets also increases, and all incoming packets are dropped when the probability increases to 1. It is most helpful in evading global synchronization of TCP flows. TCP SACK will have a higher unused window before the queue gets completely filled.

We also compared latencies (round trip times) for both the variants using both queuing disciplines. The latency of TCP Reno and TCP SACK over 60 seconds with the DropTail queuing algorithm, along with the CBR flow are shown in the figure below.

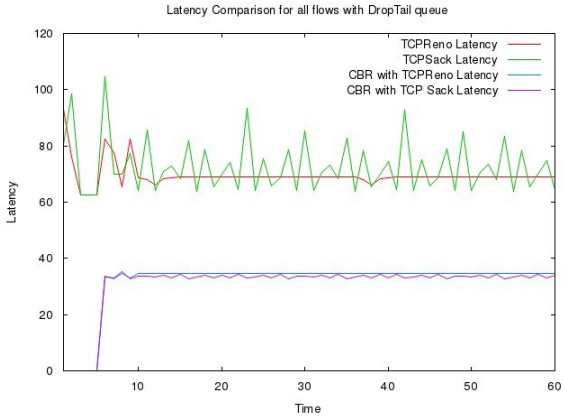

Fig. 11: Latency(Y-Axis, milliseconds) versus time (X-Axis, seconds)

As the graph shows the performance of TCP Reno is better than TCP SACK. This is because the implementation of TCP Reno allows Fast Retransmit and Fast Recovery. In a normal implementation of TCP, if a segment is lost, there is a long wait for the retransmit timer (RTO) to run out. TCP Reno provides for retransmit after three duplicate ACKs. It also provides for Fast Recovery. If the sender enters fast retransmit, the sender in TCP Reno goes into fast recovery, i.e. it avoids unnecessary slow-starts to prevent expensive timeouts. Thus overall, TCP Reno gives better performance over TCP SACK.

The figure below shows the performance of the TCP variants when the RED queuing algorithm is used. As can be observed from the graph, the performance of SACK is erratic and Reno gives a constantly low latency distribution due to its implementation. RED gives better performance because packet arrival can be bursty and erratic and its algorithm can drop packets to avoid synchronous flows as compared to

the DropTail algorithm. The overall congestion in the connected network is greatly reduced, leading to a better round trip time for all packets

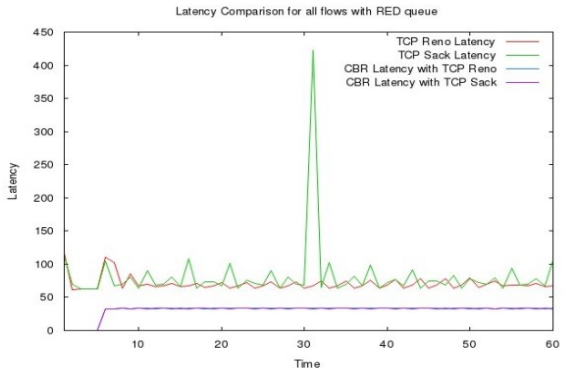

Fig. 12: Latency(Y-Axis, milliseconds) versus time (X-Axis, seconds)

We can summarize that the CBR flow occupies most of the bandwidth available steadily and dominates the TCP flow irrespective of the queuing mechanism. When the CBR flow starts, the TCP throughput falls from 80% of the available bandwidth to 25%. Thus queuing disciplines are not enough to ensure fairness of bandwidth for each flow. RED gives a better overall latency, which is significantly improved when TCP Reno is used in comparison to TCP SACK. Thus queuing disciplines are not enough to compare two TCP variants as the throughput of the variants is nearly identical while latency comparison gives TCP Reno a clear edge. We can also conclude that SACK is not a good option while using RED queues because it gives higher latency due to no congestion avoidance algorithms being present

VI. Conclusions

For the Internet to flourish incessantly its congestion control mechanism must remain efficient and effective as the network gestates. Our extensive simulation analysis illustrate that some of the TCP Variants maintains good utilization and steadiness over a band of bandwidths or a scale of delays.

In first, we observed that when different variants are tested using simulation to check bandwidth, latency and drop count, TCP Vegas outperformed compared to others. We also deduce logical reasoning behind this by observing a special property of Vegas which enables it to make intelligent assumption of possible congestion based on RTT calculation and adjusts its window accordingly.

In second window we studied performance of TCP variants by testing their performance by choosing pair of variants at a time. By careful analysis of its output we inferred that in this case TCP New Reno outperforms its pair. We can say from our analysis that, modified retransmission mechanism enables New Reno to consume bandwidth unfairly dominating other variant. Also Vegas performs very poorly because of its careful nature leading to under performance. Thus if we want to deploy TCP variant on the end system, TCP New Reno will be the best choice and when used TCP Vegas will cause huge performance degradation.

In third experiment we demonstrated how queuing algorithms RED and Droptail, affects performance of TCP Variant. We analyzed that RED having probabilistic packet drop strategy gives a better overall latency, which is significantly improved when TCP Reno is used in comparison to TCP SACK.

After conducting these experiments we have come to conclusion that these parameters though depicts very good picture, how TCP Variants will perform in real world, are not sufficient. Conclusions may vary when we consider different scenarios like Homogeneous Wired Network, Heterogeneous Wired Network and Wired-Cum-Wireless Networks. We also can consider more performance metrics like Routing Overhead, Bandwidth Delay Product, Total route requests sent, Retransmission attempts etc to tune our results. Expanding the range of analysis by considering other new TCP‘s like HS-TCP, TCP WESTWOOD etc and under different routing protocols like DYMO, OLSR etc will definitely help us to understand performance of Variants in real world. Hence deploying these variants in real world and testing them will only give us in depth analysis of the same

VII. References

[1] Yuvaraju B. N and Niranjan N Chiplunkar *“Scenario Based Performance Analysis of Variants of TCP using NS2-Simulator”*

[2] Shakeel Ahmad, Adli Mustafa and Bashir Ahmad *“Comparative Study Of Congestion Control Techniques In High Speed Networks.”*

[3] Lawrence S. Brakmo, Sean W. O’Malley and Larry L. Peterson *“TCP Vegas: New Techniques for Congestion Detection and Avoidance”*

[4] Luigi A. Grieco and Saverio Mascolo *“Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion*