Simulation based comparison of TCP variants

Deepanshu Lulla
Telecommunication Systems Management
lulla.de@husky.neu.edu
001798574

Smit Vasani
Information Assurance
vasani.s@husky.neu.edu
001901571

Abstract:

The objective of this project is to analyze the performance of different TCP variants. Several TCP variants have been proposed like TCP Tahoe, Reno, New Reno, Vegas, BIC, CUBIC, SACK, and others with mechanisms to improve the flow of data across the network. NS-2 network simulator was used to perform experiments that helped understand the behavior of the TCP variants under various load conditions and queuing algorithms. The results are analyzed and discussed in this paper.

Key Words: Network Simulation, TCP variants, Throughput, Latency, Queuing Algorithms

I. Introduction

The development of transport layer protocol leads to many implications out of which the TCP emerges as the most popular protocol. TCP provides in sequence deliverance of data and an unfailing data transmission among communicating nodes. One of the strengths of TCP is its high responsiveness towards network congestion. TCP is also a defensive protocol as it detects incident congestion and in result to that it tries to lessen the impacts of this congestion, which will prevent collapse of communication[1]. A set of mechanisms is put into place to detect occurrence of congestion and alleviate its affects

TCP is the fastest growing protocol even in future and this paper presents a comparative study of different TCP variants. These variants have been extensively studied in recent years by examining their performance under different traffic conditions. Understanding and analyzing the performance of these variants help determine which congestion control mechanism is most effective in different network topologies and different scenarios.

Tahoe is the very first variant of TCP that uses three mechanisms to organize the flow and handle congestion that is congestion avoidance, slow start, and fast retransmit. TCP Reno added another mechanism to the one's already used in Tahoe. This fourth mechanism or algorithm is known as Additive Increase and Multiplicative Decrease (AIMD). TCP New Reno introduced us to Fast Recovery while dealing with duplicate packets. Effectiveness of protocol is affected by again and again entering the Fast Recovery. TCP Vegas on the other hand detects congestion before it occurs and also follows AIMD. TCP Vegas switches to congestion avoidance phase as soon as it senses an early congestion by keep on calculating the difference of current and expected throughput. TCP CUBIC is an extended version of Binary Increase TCP. Here a cubic window increase function is used as opposed to the binary search strategy employed by BIC-TCP which makes the cwnd increases less aggressive.

Three experiments are conducted to analyze the performance of these variants. First experiment is used to measure the performance of these variance by analyzing the throughput, latency and packet drop rate of these variants. Second experiment combines two of these variants by implementing them simultaneously to analyze the bandwidth distribution. Third experiment is used to determine the effect of Drop tail and RED queuing on TCP variants Reno, SACK and CUBIC.

II. Methodology

Ns2 (Network Simulator 2) was used to analyze the TCP variants. NS (version 2) is an object-oriented, discrete event driven network simulator written in C++ and OTcl. NS is primarily useful for simulating local and wide area networks. To use NS, you program

[1] http://www.researchgate.net/publication/275337555_An_Experimental_Study_of_the_Performance_and_Effectiveness_of_TCP_Variants_in_IP_and_MPLS_Networks

in OTcl script language. To setup and run a simulation network, a user should write an OTcl script that initiates an event scheduler, sets up the network topology using the network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler. This simulator is used to generate a trace file (.tr) which is then used to analyze the variants [2].

The following diagram shows the network topology used in all the experiments.

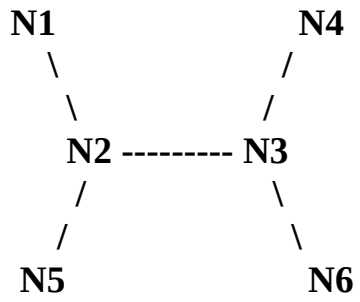


Fig 1: Topology used in Experiments

As seen in the figure there are six Nodes and five Duplex links connecting these nodes. The bandwidth of each node is 10 Mbps. A brief description of the topology is as follow:

1. N1 is the TCP source. It is assumed that an ftp application is running on this node
2. N2 is the source of an unresponsive UDP flow. It is a CBR (Constant Bit Rate) generator
3. N3 node is the CBR sink
4. N4 node is sink for the TCP source at N1
5. N5 is another TCP source(for experiment 2)
6. N6 is a sink for TCP source at N5 (for experiment 2)

CBR (Constant Bit Rate) is used between nodes N2 and N3. It uses a specific amount of bandwidth, it does not care about dropped packets, and does not perform congestion control. It just sends packets blindly at a constant rate. This CBR is used between the nodes in all three experiments.

First experiment requires us to add CBR source at N2 and a sink for the CBR at N3. Then a single TCP stream from N1 to a sink at N4 is added. The CBR is incremented in every step by 1 Mbps until it reaches the bottleneck capacity. Performance of TCP variants is then calculated by analyzing the average throughput, latency and packet drop rate.

Second Experiment uses the same topology but another TCP connection is added by attaching a TCP flow at node N5 and a sink at node N6. The performance of these two TCP flows is measured in presence of each other. Pairs that are used for this experiment are Reno-Reno, Vegas-Vegas, New Reno-Vegas, New Reno-Reno, Cubic-Tahoe, Cubic-Reno, Cubic-New Reno and Cubic-Vegas. CBR flow rate is changed from 1 to 10 Mbps.

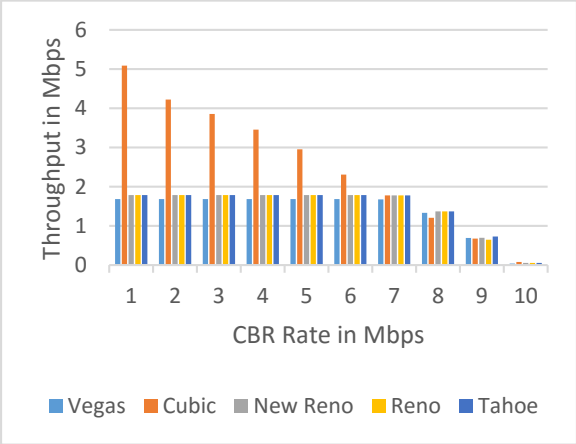
Final experiment uses a constant CBR rate of 7 Mbps. Here first the TCP starts and once it is steady, a constant CBR rate is included. Queuing algorithms Drop Tail and RED are used to analyze the performance of TCP Reno, Cubic and SACK.

III. Result:

Experiment 1:

This experiment was conducted to understand how the different TCP variants react to the presence of congestion. The result was obtained by analyzing and comparing the throughput, packet drop rate and end-to-end latency of different TCP variants.

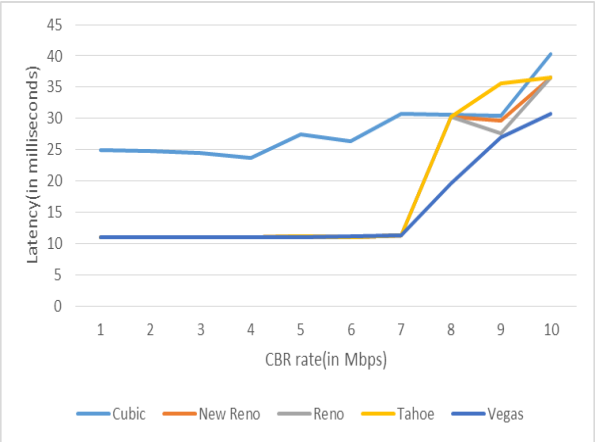
The graph shown below compares the throughput of TCP Tahoe, Reno, New Reno, Vegas and CUBIC in the presence of different CBR rates.



Graph 1: Throughput of TCP variants in varying CBR Rate

As seen in the graph, when compared between TCP Tahoe, Reno, New Reno, Vegas and CUBIC, CUBIC has a better throughput. It can be seen that the throughput for other variants isn't affected much till the CBR rate is increased till 7 Mbps. It can be seen that after 7 Mbps, for all the variants the overall throughput is decreased and this decrement continues till the Bottleneck Bandwidth is reached and the throughput becomes almost zero.

The graph shown below compares the end-to-end latency of the TCP variants discussed above.



Graph 2: Latency of TCP variants in varying CBR Rate

The above graph shows that latency of CUBIC is worse when compared to other variants. For a brief range, when CBR rate is between 7-9 Mbps; Tahoe performs worse than others. Vegas performs better in all cases. Latency, just like throughput, shows sudden changes after the CBR Rate reaches 7 Mbps. The increase in latency of Tahoe, Reno and New Reno is quite similar till CBR of 8 Mbps, but by the time bottleneck bandwidth is reached New Reno and Reno shows better performance as compared to Tahoe. CUBIC and Vegas show better performance than the rest of the variants.

The next in the line is to study the amount of packets dropped in varying CBR rates.

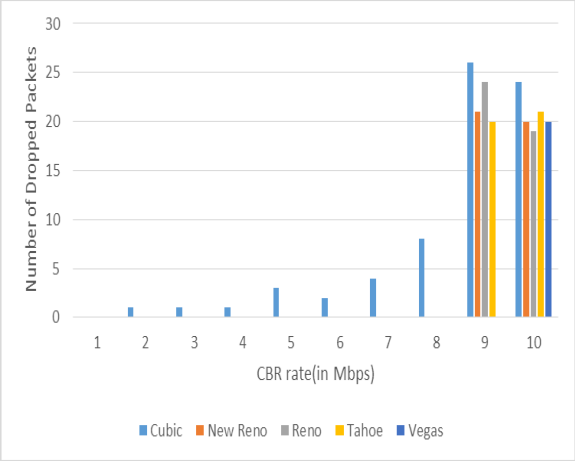


Figure 3: Dropped Packet Rate for TCP variants in varying CBR Rate

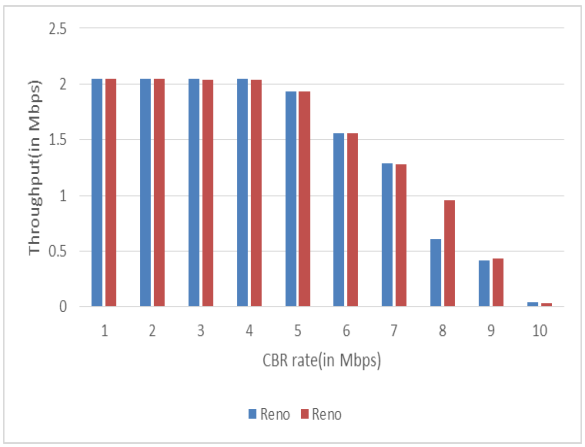
The graph clearly shows that there is minimum packet drops for all TCP variants, except CUBIC, till the CBR Rate reaches 8 Mbps. Vegas has the minimum number of dropped packets when compared to other variants. Vegas has its first dropped packet after the CBR Rate reaches bottleneck capacity.

Throughput and latency of CUBIC was better when compared to other variants, but the packet drop rate of CUBIC was higher than other variants. Vegas on the other hand had the best latency and the least packet drop rate as compared to other variants. Whereas, Tahoe, Reno and New Reno had more latency and packet drop rate. Thus the performance of TCP Vegas was better than other variants. It had a decent throughput with a very good latency and minimum packet drop rate.

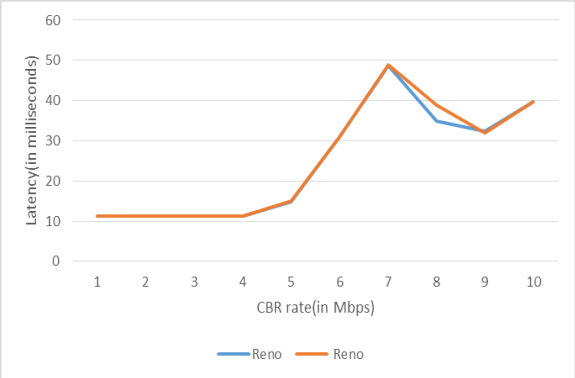
Experiment 2:

This experiment was conducted to analyze the fairness between different TCP variants. Two TCP variants were implemented simultaneously to determine the fairness. The following combinations were implemented when trying to determine their performance with each other.

1. Reno/Reno
2. New Reno/Reno
3. Vegas/Vegas
4. New Reno/Vegas

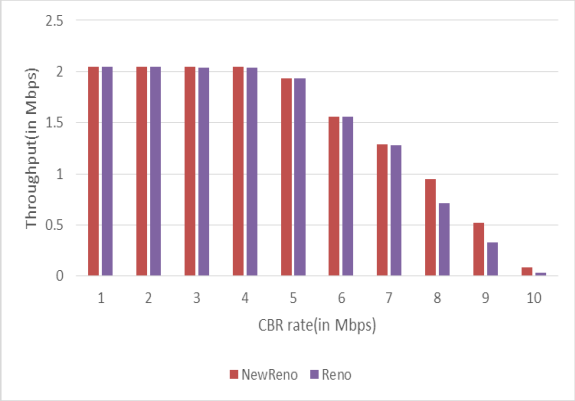


Graph 4: Throughput of TCP variants Reno/Reno in varying CBR Rate

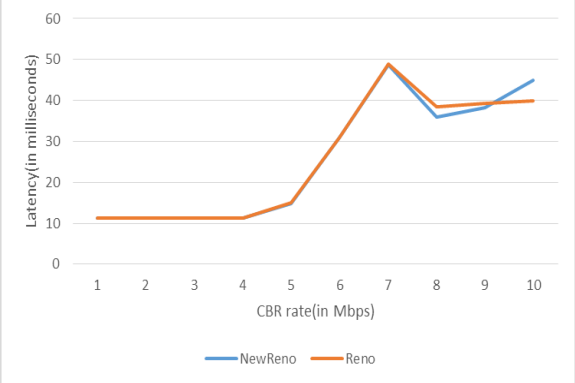


Graph 5: Latency of TCP variants Reno/Reno in varying CBR Rate

As it can be understood from these graph, the throughput and latency of both the variants is almost the same. The only exception was for the CBR Rate of 8 Mbps when New Reno dominated over Reno. But overall we can conclude that this scenario showed a good fairness.

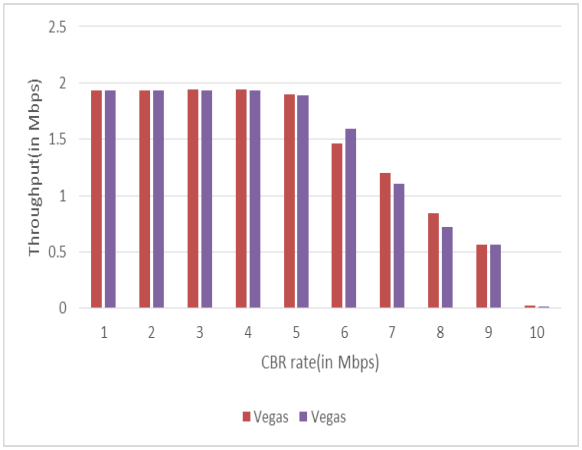


Graph 5: Throughput of TCP variants New Reno/Reno in varying CBR Rate

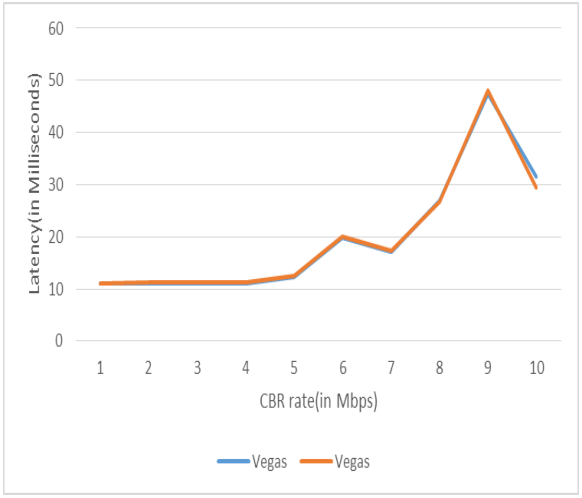


Graph 6: Latency of TCP variants New Reno/Reno in varying CBR Rate

The use of TCP variants New Reno and Reno also shows good fairness till CBR Rate of 7 Mbps. This fairness is affected once the CBR increases and New Reno dominates Reno.

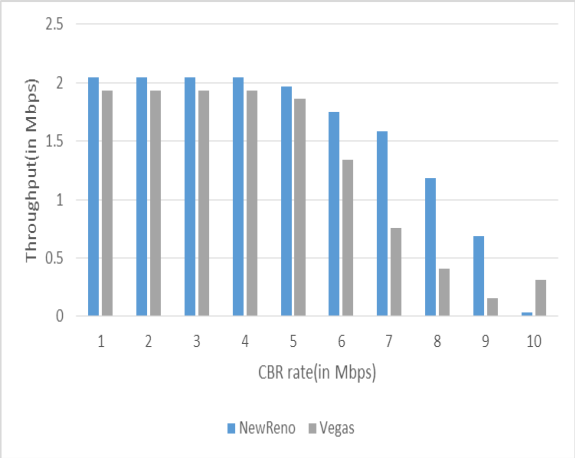


Graph 7: Throughput of TCP variants Vegas/Vegas in varying CBR Rate

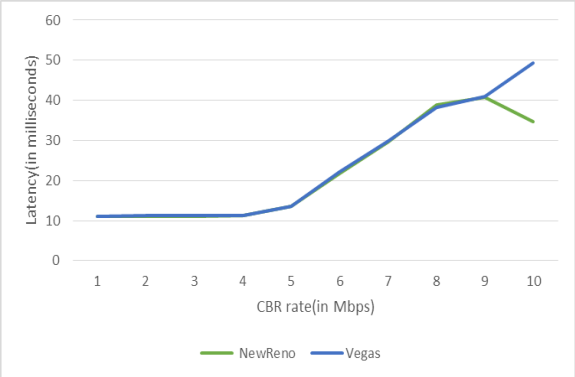


Graph 8: Latency of TCP variants Vegas/Vegas in varying CBR Rate

The fairness between TCP variants Vegas/Vegas is affected once CBR Rate reaches 6 Mbps and the latency is not affected at all. It's after this that one variants starts dominating the other.

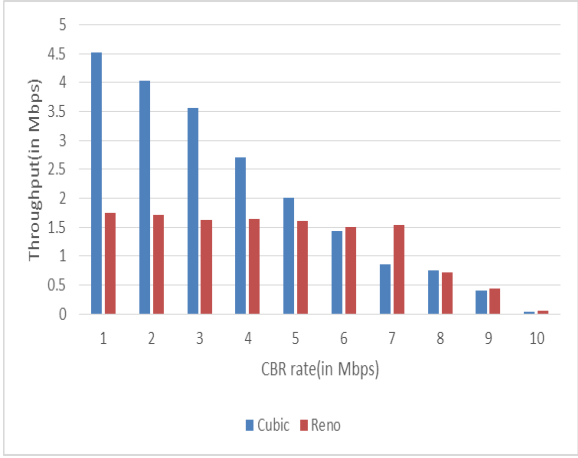


Graph 9: Throughput of TCP variants New Reno/Vegas in varying CBR Rate



Graph 10: Latency of TCP variants New Reno/Vegas in varying CBR Rate

As it can be seen from the graph 9 and graph 10 New Reno dominates Vegas. Initially till the CBR Rate reaches 5 Mbps the fairness is pretty good and acceptable. But as the CBR is further increased, New Reno dominates Vegas. The reason for this is because of the fact that Vegas implements RTT for computation of network congestion. So on sensing congestion due to another TCP source and CBR it reduces its throughput.



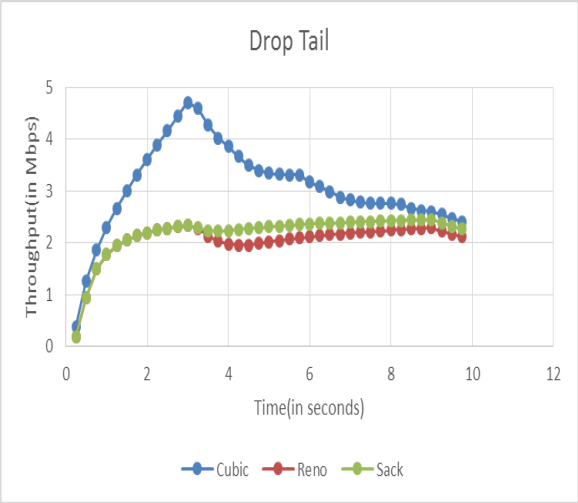
Graph 11: Throughput of TCP CUBIC/Reno in varying CBR Rate

Graph 12: Latency of TCP variants CUBIC/Reno in varying CBR Rate

This graphs 11 and 12 clearly state the dominance of TCP CUBIC in presence of TCP Reno.

Experiment 3:

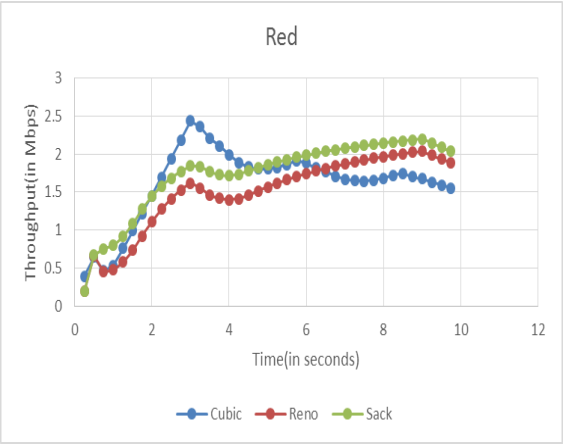
This experiment was conducted to help study the influence of the queuing discipline RED and Drop Tail on the overall throughput of flows. Queuing disciplines like Drop Tail and Random Early Drop (RED) are algorithms that control how packets in a queue are treated. The TCP flow between nodes 1 and 4 was started first and the CBR flow of 7 Mbps was added after 3 seconds. Let's analyze the performance of these queuing algorithms.



Graph 13: Throughput of TCP variants TCP Reno and SACK while using Drop Tail queuing

The graph 13 displays the performance of TCP Reno Cubic and SACK when Drop Tail queuing is used. It can be clearly understood from the graph that the throughput increases initially till 3 seconds. This is because the CBR source is off and the TCP flow tries to slowly increase to 10 Mbps bandwidth. But as the CBR rate of 7 Mbps is started the throughput reduces. The throughput and thus performance of CUBIC surpasses that of Reno and SACK in this experiment. But when throughput of Reno and SACK are compared, SACK wins it by a close margin. The throughput is same for these two variants till the CBR flow is started. But as CBR initiates, Reno's throughput takes a hit.

Now let us analyze the performance of these same variants when RED queuing is used.



Graph 14: Throughput of TCP variants TCP Reno and SACK while using RED queuing

The above graph uses the same topology as used by Drop Tail queuing. We can see the similar rise in throughput for the TCP variants when a steady and lone TCP flow is used. But there is a different variation in the performance when the CBR source is started. After the source is started initially there is a dip in the throughput, but as the time passes the performance for TCP Reno and SACK increases whereas the performance of CUBIC decreases. RED queuing performs better when the CBR source is attached and thus is a better queuing algorithm.

IV. Conclusion

The three experiments performed above were very useful in understanding the performance of various TCP variants in different topologies and conditions.

The first experiment helped us understand the performance of TCP variants under congestion (in this case in presence of a CBR flow). Here Vegas had a better performance as compared to its counterparts Tahoe, Reno, New Reno and CUBIC. A good throughput and a very small end-to-end latency and drop packet rate helped us determine that.

The second experiment helped understand the fairness factor between TCP variants when used simultaneously. The combination of CUBIC and Vegas had the least fairness where CUBIC dominated Vegas. Whereas the combination of similar variants of TCP (Reno/Reno and Vegas/Vegas) showed the most fairness when used together.

Last experiment showed how performances are affected when different queuing algorithms are used. Performance of SACK, CUBIC and Reno was better when used with RED queuing. Drop Tail queuing when used provided the best throughput in the absence of CBR flow. But after the CBR flow was included the performance of these variants was better when RED queuing was used.

V. References

[1] Simulation-based Comparisons of Tahoe, Reno, and SACK TCP by Kevin Fall and Sally Floyd

[2] http://nile.wpi.edu/NS/