Simulation-based Performance Analysis of TCP Variants

Karthik Chandranna and Sujith Narayan

**1. Introduction**

Transmission Control Protocol (TCP) is one of the core protocols of communication networks. It is a reliable connection-oriented end-to-end protocol. Since its origin, many enhancements have been proposed to the original design. Analysis of these variants is important to determine the best protocol variant for a given network scenario.

This paper describes the results of simulation-based experiments to analyze the performance of several TCP variants, namely: TCP Tahoe, Reno, New-Reno, Vegas and TCP SACK. The focus is on comparing the different variants by studying them in simulated environments, either in presence of each other or in the presence of a Constant Bitrate flow (CBR).

We evaluate the variants based on their performance under congestion, fairness to each other, and we also study the influence of queueing algorithms on the overall performance of flows. TCP Tahoe and TCP Reno exhibit similar throughput patterns over increasing CBR bandwidth. TCP Vegas minimizes packet drops but suffers a dip in throughput in presence of TCP New-Reno.

**2. Methodology**

The experiments were conducted using Network Simulator NS-2. Our considerable confidence in NS is due to the fact that it is the most popular choice of simulator used in research papers, and is constantly maintained and updated by its large user base. Trace files that contain traces of all the packets sent during the experiment were generated in order to analyze the results. Python scripts were written to parse the data in the trace file, extract the required fields, and perform relevant computation. The output of these python scripts were stored in a file, and this file was used in plotting graphs using MS Excel.

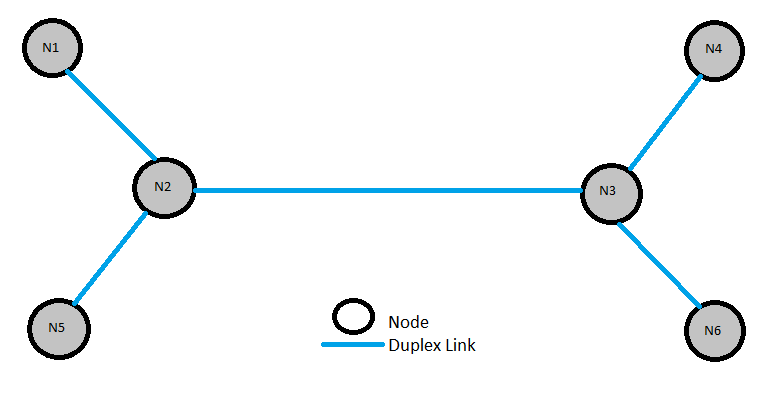


Fig. 1

The topology used for the experiments (Fig. 1) has 6 nodes and 5 duplex links connecting the nodes. Each node can be either a source or a sink. The nodes are one of TCP agent, CBR agent, UDP agent, FTP Application, or a null agent. Each duplex link has a bandwidth of 10Mbps, delay of 10ms and a queue limit of 5. Each flow is assigned an identifier - fid.

Experiment 1 deals with the analysis of the performance of four TCP variants – Tahoe, Reno, New-Reno, and Vegas as a function of the bandwidth used by the CBR flow. A single TCP stream is added from Node N1 to a TCP sink at Node N4. FTP over TCP connection is implemented by attaching FTP Application to the source TCP agent at Node N1. A UDP agent is attached to Node N2, and a null agent is attached to Node N3. We then add a CBR source at Node N2 and a CBR sink at Node N3. The queue uses a Droptail queueing algorithm and its limit is set to 5.

The simulation is scheduled to begin from 0.1s and end after 11s. The CBR source is scheduled to start transmitting on start of the simulation. The FTP application starts transmitting after 1s. The CBR source stops transmitting at 10.5s. The bandwidth of the CBR flow is the parameter that is varied for each trial. For the first trial, the bandwidth is set to 1Mbps, and it is incremented by 0.1 for each trial. The test is performed until the bottleneck capacity is reached. The same test is performed for all the four TCP variants in consideration, and the recorded values are parsed and corresponding graphs are plotted.

Experiment 2 deals with the analysis of fairness between different TCP variants. One CBR flow and two TCP flows are used. Two TCP streams are added from Node N1 to Node N4 and from Node N5 to Node N6, respectively. FTP over TCP connection is implemented by attaching FTP application to each of the source TCP nodes N1 and N5. A UDP agent is attached to Node N2, and a null agent is attached to Node N3. We then add a CBR source at Node N2. The queue uses a Droptail queueing algorithm and its limit is set to 5.

The simulation is scheduled to begin from 0.1s and end after 11s. The CBR source is scheduled to start transmitting on start of the simulation. Both the FTP applications attached to TCP agents start transmitting after 1s. The CBR source stops transmitting at 10.5s. The bandwidth of the CBR flow is the parameter that is varied for each trial. For the first trial, the bandwidth is set to 1Mbps, and it incremented by 0.25 for each trial. The test is performed until the bottleneck capacity is reached. The same test is performed for each pair of the TCP variants – Reno/Reno, New-Reno/Reno, Vegas/Vegas and New-Reno/Vegas - by using corresponding TCP source and sink agents for the two flows. The recorded values are parsed and graphs are plotted.

Experiment 3 deals with the study of the influence of queueing algorithms used by the nodes – DropTail or RED - on the overall performance of flows. A single TCP stream is added from Node N1 to Node N4. FTP application is attached to the TCP source at Node N1. A UDP agent is attached to Node N2, and a null agent is attached to Node N3. We then add a CBR source at Node N2.

The simulation is scheduled to begin from 0.1s and end after 26s. The FTP application is scheduled to start transmitting at 0.1s and end after 25s. The TCP flow does not have contention in the first 10s of the simulation, where it is allowed to reach a steady state. The bandwidth of the CBR source is set at 8Mbps in order to introduce high contention in the network and in turn, observe throughput and latency patterns during congestion. It starts transmitting from 10s. The throughput and latency values are recorded and the graph is plotted over time on the X-axis. The experiment is conducted by varying the queueing discipline used by the nodes, i.e. DropTail and RED. The recorded values are parsed and graphs are plotted.

**3. Analytical Report on Performance of TCP Variants**

Fig.2 and Fig 3

Fig. 4

To analyze the throughput, packet drops and latency of the variants, we varied the CBR bandwidth for each trial. From fig. 2, we observe that the average throughput of all the four TCP variants is almost equal with a standard deviation of 25.51Kbps. The similarity is also observed in the see-saw pattern that is seen during bottleneck bandwidth. This happens due to the Additive Increase and Multiplicative Decrease of the congestion window. When the congestion window is increased, packets are dropped as the network has reached bottleneck capacity. This leads to a multiplicative decrease of the congestion window. This cycle repeats to produce a see-saw pattern on the throughput plot. Compared to the other variants, Reno provides a lower average throughput of 1915Kbps as its performance dips when multiple packets are dropped in the same window.

From fig. 3, we observe that TCP Vegas has the fewest packet drops for CBR bandwidth greater than 5Mbps. The average packet drop for Vegas when the CBR bandwidth was varied from 1 to 10 was 2.6, whereas the other variants suffered packet drops in the range 6.4-8. This is a significant difference when compared to TCP Vegas. The Vegas congestion algorithm is based on Round-trip time (RTT) and delay rather than packet drops. It accurately estimates the RTT value periodically and adjusts the sending rate based on the RTT values of the packets. This mechanism ensures that fewer packets are sent during congestion, and this leads to a lower drop rate.

From fig. 4, we observe that TCP Vegas has the least latency for CBR bandwidth greater than 7Mbps. The average latency for Vegas when the CBR bandwidth was varied from 1 to 10Mbps, with 10Mbps being the bottleneck capacity, was 75ms. This is an improvement over the other variants, each of which had a latency of around 85ms. The congestion control algorithm of TCP Vegas is based on estimation of RTT. Since TCP Vegas does not wait for a Retransmission timeout or a duplicate acknowledgement to signal packet loss, and preemptively retransmits packets, the latency is significantly reduced.

From our first set of experiments, we conclude that there is no “best” TCP variant. Choosing the right TCP variant for the network depends on characteristics of the network such as bandwidth and congestion. If the bandwidth of the network is high, any of the three variants – TCP Tahoe, New-Reno or Vegas would be preferred. If the network suffers from high packet loss or high latency, TCP Vegas should be the preferred variant.

**4. Analytical Report on Fairness of TCP Variants**

Fig 5

Fig 6

Fig 7

Fig 8

Fig 9

Fig 10

From Fig. 5, we observe that the throughput plots for both flow1 and flow2 of TCP Reno are alike. Further, the average throughput of flow1 and flow2 is around 1420Kbps, with a standard deviation of 8. Similarly, in our experiments, it has been observed that the latency for both flow1 and flow2 is around 100ms. In addition, the drop rate of both the flows has been observed to be around 12-14%. Since both the flows use the same congestion control algorithm which is TCP Reno, the throughput is approximately equal. Both the flows begin with slow start wherein the congestion window is increased exponentially. When flow1 is utilizing more bandwidth, the packet drops are more because there is congestion in the network due to flow2 and the CBR flow. This leads to more duplicate acknowledgements, which when greater than 3 causes the congestion window of flow1 to reduce by half. This gives an opportunity for flow2 to utilize more bandwidth. The same scenario holds true when flow2 uses more bandwidth.

From Fig. 6, we observe that the throughput plots for New-Reno and Reno are mostly alike except for CBR bandwidth in the range of 8-10Mbps. In this range, the flow which uses New-Reno algorithm has a higher throughput. However, the average throughput of each of the flows for the entire duration of the experiment is around 1425Kbps with a standard deviation of 16. It has been observed from the experiments that the latency and the packet drop rate, as seen in Fig. 10, of the Reno flow is slightly higher than that of the New-Reno flow. The New-Reno variant retransmits a packet after receiving only one duplicate acknowledgement whereas Reno retransmits a packet after receiving three duplicate acknowledgements. This leads to New-Reno utilizing a higher share of bandwidth, and thus not being totally fair to Reno.

From Fig. 8, we observe that the throughput plots for the two Vegas flows are almost identical. This is further emphasized by the average throughput values for each of the flows, which is around 1430Kbps with a standard deviation of 11. Also, from Fig. 7 and our experiments on latency, the average latency for both the flows is 75ms. Both the flows follow the TCP Vegas algorithm where congestion is detected based on estimation of RTT rather than packet drops. This leads to fewer packet drops, which has been observed in the results of our experiments. From the above test results, we infer that both the TCP Vegas flows are being fair to each other.

From Fig. 9, we observe that the throughput of the New-Reno flow is much higher than that of the TCP Vegas flow over increasing CBR bandwidth. The average throughput of the New-Reno flow was observed to be 1480.3Kbps which is 11%

higher than the average throughput of the Vegas flow which is 1315Kbps. The congestion control in New-Reno is detected based on packet loss i.e. a packet is retransmitted immediately after receiving a single duplicate acknowledgement.

Whereas in Vegas, congestion control is done based on the observed RTT of the sent packets i.e. the congestion window/sending rate is increased or decreased dynamically. In the event of congestion in the network, New-Reno would wait for a packet drop to recognize congestion. TCP Vegas senses congestion before New-Reno as it does not wait for a packet drop but instead senses the increase in RTT. Hence, in New-Reno, the congestion window continues to increase until there is a packet drop whereas in TCP Vegas, the congestion window/sending rate would have already decreased. This leads to New-Reno utilizing more bandwidth. Thus we can conclude that TCP New-Reno is unfair to TCP Vegas.

**5. Analytical Report on the Influence of Queueing**

Fig 11

Fig 12

Fig 13

Fig 14

Fig 15

As observed in Fig 11, DropTail queueing discipline provides fair share of bandwidth to the TCP Reno flow when it is under contention from the CBR flow. Considering that the CBR flow is at a constant bandwidth of 8Mbps, the TCP Reno’s throughput, under contention from CBR flow, has decreased to an average of 1900Kbps. This is after TCP Reno has reached a steady state of 2500Kbps average throughput.

Similar to TCP Reno, TCP SACK gets a fair share of bandwidth when the queueing discipline used is DropTail. After reaching a steady state of throughput 2500Kbps average throughput, it reduces to an average of 1890Kbps when under contention from the CBR flow.

Thus we can infer that both TCP Reno and TCP SACK, when under contention with the CBR flow, get fair share of bandwidth when DropTail queueing discipline is used.

As observed in Fig 14, the Random Early Detection (RED) queueing discipline does not provide fair share of bandwidth to the TCP SACK flow when it is under contention from the CBR flow. Before the introduction of the CBR flow, the average throughput of the TCP SACK flow was observed to be at a steady state of 2500Kbps average throughput. After introducing the CBR flow, the average throughput of TCP SACK was found to be 1324Kbps when under contention from the CBR flow. Thus the throughput of TCP SACK decreased by 1200Kbs but the CBR flow was still able to provide a throughput of 7750Kbps.

Similar to TCP SACK, TCP Reno does not get a fair share of bandwidth under contention from the CBR flow when the queueing discipline used is RED. Before the introduction of the CBR flow, the throughput of the TCP Reno flow was observed to be at a steady state of 2500Kbps average throughput. After introducing CBR flow, the average throughput was found to be 1145Kbps while the CBR flow still maintained an average of 7770Kbps average throughput.

Thus we can infer that both TCP Reno and TCP SACK, when under contention with the CBR flow, do not get a fair share of bandwidth when RED queueing discipline is used.

As observed in Fig 12 and Fig 13, the end-to-end latency plot of the TCP flow is steady when the queueing discipline used is DropTail. This behavior of DropTail queueing discipline is the same for both TCP Reno and TCP SACK. But when the queueing discipline used is RED, we observe a see-saw pattern in the end-to-end latency plot of the TCP flow. This behavior of RED queueing discipline is the same for both TCP Reno and TCP SACK. However, the average end-to-end latency of the TCP flows at the end of the simulation was observed to be the same irrespective of the queueing discipline used. On further analysis, RED was found to have a slight edge over DropTail with a miniscule decrease in latency of 1ms.

As observed in the given graphs, upon the introduction of the CBR flow, the throughput of the TCP flow decreases due to contention from the CBR flow.

Since the network is congested after the introduction of the CBR flow, more number of packets are queued in the buffer. The end-to-end latency increases because of this queueing delay at the network components which is similar in both the DropTail and RED queueing disciplines. However, packet drops are negligible in DropTail queueing discipline as it gives an average packet drop of 0.2 for both TCP SACK and TCP Reno. RED queueing discipline, on the other hand, gives an average packet drop of 2-2.5 for both TCP SACK and TCP Reno.

From our experiments, we observed that the average throughput of TCP SACK decreases from 2500Kbps to 1678Kbps upon the introduction of CBR flow when the queueing discipline used is RED. But the average throughput of TCP SACK under contention from the same CBR flow reduces only to 2054Kbps when the queueing discipline used is DropTail. Further, the average drop count of TCP SACK is 2.5 when under contention from the CBR flow when the queueing discipline used is RED. On the other hand, the average drop count remains at zero when the queueing discipline used is DropTail. Lastly, the average latency of TCP SACK when CBR flow is introduced increases to 66.2ms from 60ms when the queueing discipline used is RED. For DropTail, the latency increases to 69ms, which is slightly higher compared to the latency when the queueing discipline used is RED.

Thus, we can infer that RED is not a good option for TCP SACK as it results in a lower average throughput and a higher number of packet drops in comparison to DropTail queueing discipline.

**6. Conclusion**

Through this paper, we have analyzed the performance of several TCP variants, namely: Tahoe, Reno, New-Reno and Vegas under congestion. In addition, we have analyzed the degree of fairness between the different TCP variants under varying network conditions. Lastly, we have studied the influence of queueing disciplines on the overall performance of the TCP variants. Some of the key results of our experiments include the fairness graph of TCP New-Reno vs TCP Vegas, where TCP New-Reno deprives TCP Vegas of bandwidth. Also, the improvement that TCP Vegas provides over the other variants in terms of latency and packet drop rate is significant.

**7. References**

[1] Kevin Fall and Sally Floyd. “Simulation-based Comparisons of Tahoe, Reno, and SACK TCP”. Technical Report URL http://david.choffnes.com/classes/cs4700fa14/papers/tcp-sim.pdf

[2] Kenji Kurata, Go Hasegawa and Masayuki Murata. “Fairness Comparisons Between TCP Reno and TCP Vegas for Future Deployment of TCP Vegas”.

[3] Sonia Fahmy and Mridul Sharma. “Deployment Considerations for the TCP Vegas Congestion Control Algorithm”. Technical Report 2003.

[4] Monia Ghobadi, Soheil Hassas Yeganeh, Yashar Ganjali. “Rethinking End-to-End Congestion Control in Software-Defined Networks”. URL <http://conferences.sigcomm.org/hotnets/2012/papers/hotnets12-final85.pdf>

[5] Manish Devendra Chawhan and Dr Avichal R.Kapur. “PERFORMANCE EVALUATION OF TCP VARIANTS IN WI-FI NETWORK USING CROSS LAYER DESIGN PROTOCOL AND EXPLICIT CONGESTION NOTIFICATION”. International Journal of Wireless & Mobile Networks (IJWMN) Vol. 3, No. 3, June 2011.

[6] Md. Shohidul Islam, M.A Kashem, W.H Sadid, M. A Rahman, M.N Islam, S. Anam. ” TCP Variants and Network Parameters: A Comprehensive Performance Analysis”. Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I IMECS 2009, March 18 - 20, 2009.

[7] M.A. Rahman, A.H.Kabir, K. A. M. Lutfullah, Z.Hassan, and M. R. Amin. “Fair comparisons of different TCP variants for future deployment of Networks”. International Conference on Electronics, Computer and Communication (ICECC 2008).