Performance Analysis of TCP Variants

Prathamesh Kirloskar, Salim Ali Siddiq

College of Computer and Information Science, Northeastern University, Boston-02215

kirloskar.p@husky.neu.edu

siddiq.s@husky.neu.edu

Abstract – **The transmission control protocol (TCP) is a widely used connection oriented transport layer protocol that provides a reliable packet delivery over an unreliable network. However the initial version of TCP worked reliably but it was unable to perform satisfactorily as the network grew. Since then several TCP variants have been proposed to increase TCP performance by including mechanisms to control and avoid congestion. This paper tries to find out the best performing TCP variant under various load conditions and queuing algorithms. We have conducted different experiments using NS-2 simulations and plotted graph for different CBR rates and TCP flows. Finally, we have concluded the best TCP variant by measuring different network characteristics such as average throughput, latency and packet drops.**

I. INTRODUCTION

TCP or Transmission Control Protocol is a core protocol developed for reliable data transfer between hosts. Initially, when the networks were small and less congested, TCP performed well. But with the ubiquitous use of networks throughout the world, the networks have become congested. To overcome the problem of congestion, many new TCP variants were developed. Each variant of TCP went on to include one or more algorithms to tackle congestion problems. TCP Reno introduced Fast Recovery over TCP Tahoe which is the base TCP variant. TCP New Reno uses fast retransmit over TCP Reno. TCP Vegas uses its own congestion avoidance mechanism. We have conducted the experiments on some of the TCP variants such as TCP Tahoe, TCP Reno, TCP New Reno and TCP Vegas. The experiments have been performed to analyze TCP performance under Congestion, Fairness, when bid against each other, and the influence of Queuing. This paper will help in selecting an appropriate TCP variant in respective network to optimize traffic goal.

II. METHODOLOGY

We have performed network stimulations and studied the behavior of different TCP variants by creating a network topology as shown in Fig 1.0. This topology consists of 6 nodes which are connected by duplex links. We have created a TCL script to setup this network topology where we have defined the TCP variants and CBR flow rate depending upon the type of experiment performed. The TCL scripts also runs the NS simulator which then creates a trace file. The data obtained from trace files are then used to calculate throughput, latency and number of packets dropped. We consider the average throughput, average latency and average number of packets dropped to plot the graphs.

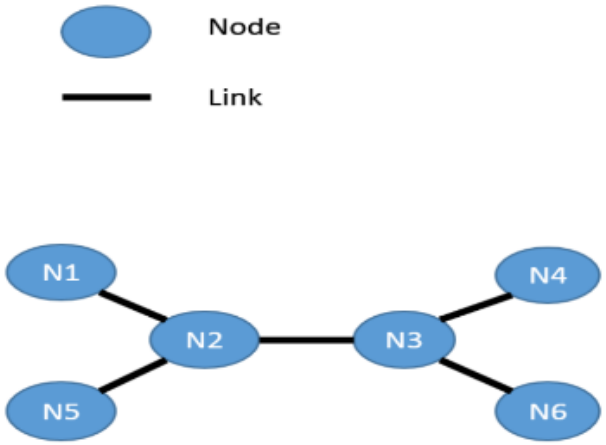

Fig 1.0

(1) Tools-

NS-2 Simulator: NS-2 network simulator can simulate networks in real world and record a large amount of real-time data into trace files. These files can be used to get the values required for calculating the parameters needed such as throughput, latency and packet drop rate.

Python 2.7: Python is a widely used general-purpose, high-level, object oriented and dynamic programming language. We have used Python programming language to obtain and analyze data from the trace files.

Microsoft Excel: Microsoft Excel is a spreadsheet developed by Microsoft. We have used it to find the average of data obtained and also plot the graph using graphing tools in it. We have also used it to perform TTEST on the throughput for experiment 1.

III. EXPERIMENT ONE - TCP Performance Under Congestion

Methodology:

In this experiment, we have setup a network topology as shown in Fig 1.0. An FTP agent has been created over TCP agent on Node N1. The FTP agent will trigger the TCP agent to transmit packets which flows from Node N1 to Node N2 then to Node N3 and finally it sinks on Node N4 where the TCP packets are acknowledged. The bandwidth of each link is set to 10 Mbps. A CBR flow over UDP has been created between node N2 and N3. Congestion is introduced in the link between nodes N2 and N3 by constantly incrementing the value of CBR flow rate. For a particular CBR flow rate, the TCP connection is started at different times. Say for example, the CBR flow may be started at time 0.0 and the TCP variant may be started at different times such 0, 1, 2 so on so forth. We take 50 such readings to get the average and plot a graph for the particular values. These multiple readings are also particularly important to get the standard deviation of each result for later t-test. This makes sure the difference is statistically different from each other, thus confirming whether the protocol caused the difference or it was just some random noise.

Analysis:

(1) Throughput-

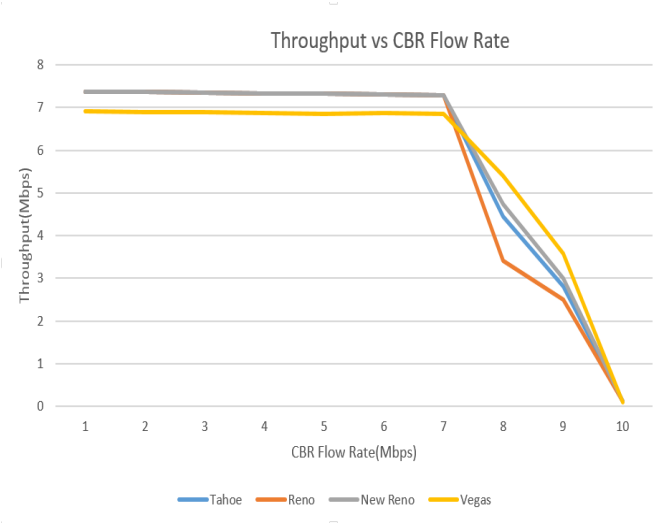

Fig 2.0: Throughput vs CBR flow rate

In case of low CBR flow rates, the throughput of all four TCP variants are steady. TCP Vegas has a slightly lower throughput compared to the other TCP variants. With the increase in CBR flow rate, all four TCP variants start suffering congestion and the throughput starts dropping. But TCP Vegas has distinct better average throughput than others. TCP Reno, on the other hand, is the worst performer under such situations. TCP Vegas' ability to detect congestion earlier by observing the RTT (Round Trip Time) gives it a better performance in congested networks. But under a non-congested network, TCP Vegas has lower throughput because of its sensitive nature and prudent congestion control mechanism.

- *T-Test:*

The T-Test values are calculated to get the significance of difference between the performances of the different TCP variants. We compare TCP Vegas to others as it seems to perform better than other TCP variants. The T value can be computed using the following formula:

$$t = \frac{\overline{x_1} - \overline{x_2}}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}}$$

Where,

x_1 is the mean of first data set

x_2 is the mean of first data set

S_1^2 is the standard deviation of first data set

S_2^2 is the standard deviation of first data set

N_1 is the no. of elements in the first data set

N_2 is the no. of elements in the first data set

T value of TCP Vegas over TCP Reno = 0.5605

T value of TCP Vegas over TCP New Reno = 0.2546

T value of TCP Vegas over TCP Tahoe = 0.3749

(2) Drop Rate-

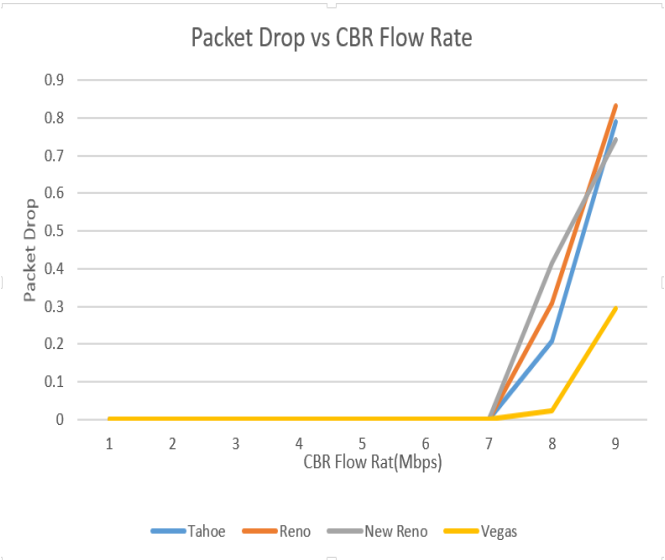

Fig 2.1: Packet Drop vs CBR flow rate

In case of low CBR flow rates, no packets are dropped for all four TCP variants. With the increase in CBR flow rate, it is clearly evident from the graph that TCP Vegas performs better than the other three variants. We observe that except TCP Vegas, all other TCP variants start dropping packets earlier than Vegas. TCP Tahoe, Reno and New Reno use 'Additive Increase Multiplicative Decrease' algorithms to control the congestion window and they detect the packet loss by getting the required number of duplicate acknowledgements. Instead of this, the TCP Vegas uses RTT detection rather than the number of packet-drops to control the congestion window. This makes TCP Vegas to detect the congestion earlier than others and hence has relatively lower drop rate under all congestion conditions. Thus, in terms of average drop rate, TCP Vegas performs the best.

(3) Latency-

In case of low CBR flow rates, all TCP variants perform well. The average latency is almost the same as optimal at lower CBR flow rates. As CBR rate increases, all TCP variants start to suffer longer delays. While TCP Vegas has much lower latency, TCP Tahoe, Reno and Vegas have higher latencies as seen in fig2.2. TCP Vegas employs a more efficient algorithm to measure congestion which detects congestion before the packet losses occur. Thus TCP Vegas could detect and avoid the congestion more easily and efficiently, hence having a reduced latency.

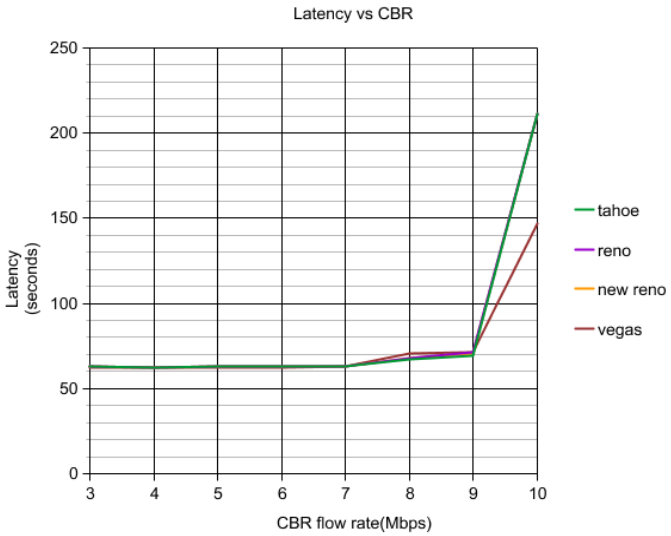

Fig 2.2: Latency vs CBR flow rate

Is there an overall "best" TCP variant in this experiment, or does the "best" variant vary depending on other circumstances?

Overall, TCP Vegas variant performs best in congested networks. Under non-congested networks, the performance all TCP variants is almost same.

IV. EXPERIMENT TWO – Fairness between TCP Variants

Methodology:

In this experiment, we have setup a network topology as in the first experiment. An FTP agent has been created over TCP agent on Node N1. The FTP agent will trigger the TCP agent to transmit packets which flows from Node N1 to Node N4. Similarly another FTP agent has been created over TCP agent for path Node N5 to Node N6. The bandwidth of each link is set to 10 Mbps. A CBR flow over UDP has been created between node N2 and N3. Congestion is introduced in the link between nodes N2 and N3 by constantly incrementing the value of CBR flow rate. For a particular CBR flow rate, the TCP connections is started at different times. Say for example, the CBR flow may be started at time 0.0 and the TCP variant may be started at different times such as 0, 1, 2 so on so forth. The start time and end time of each TCP variant is changed and the readings are noted. We take 50 such readings to get the average and plot a graph for the particular values.

Analysis:

(1) Reno/Reno-

As the CBR rate increases, both Reno change in the same way. In all three factors, i.e. throughput, latency and packet drop rate, no single Reno dominates the other. It is seen that both of the TCP Reno utilize resources in a fair manner. Though there are certain oscillations at certain CBR flow rates, it still justifiable to say that TCP Reno is fair to another TCP Reno. In a congested network, both TCP Reno compete for the remaining resources and eventually reach a common point of sharing resources. The reason for this fairness is that both the TCP Reno use the same congestion detection algorithms, i.e. duplicate ACKs detection.

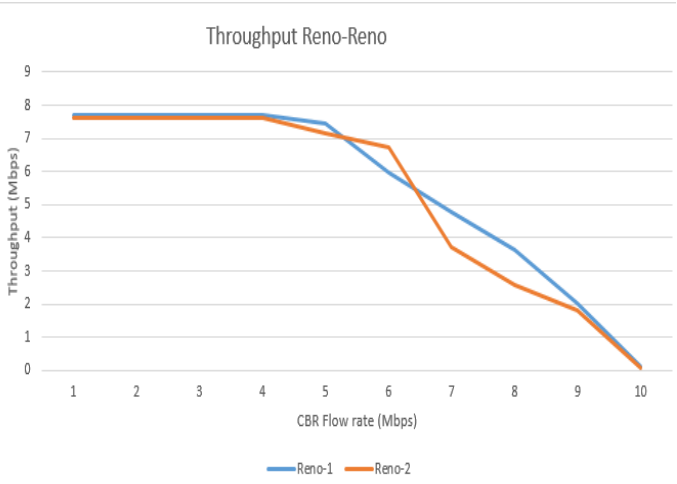

Fig 3.0: Throughput graph of Reno-Reno

(2) New Reno/Reno-

From the average throughput graph plotted, we can see that at CBR flow rates greater than 6Mbps, TCP New Reno performs better than TCP Reno. In terms of packet drop rate and latency with respect to CBR flow rates, there is no distinguishable difference between the two TCP variants. Hence we can say that TCP New Reno and TCP Reno are fair to each other in terms of latency and packet drop rate but unfair in terms of throughput. The reason for this is TCP New Reno has an improved retransmission policy as compared to Reno. TCP New Reno is eager to send more packets than TCP Reno under congested environments. TCP New Reno is aggressive in sending retransmissions at single duplicate ACK whereas TCP Reno retransmits only after triple duplicate ACKs.

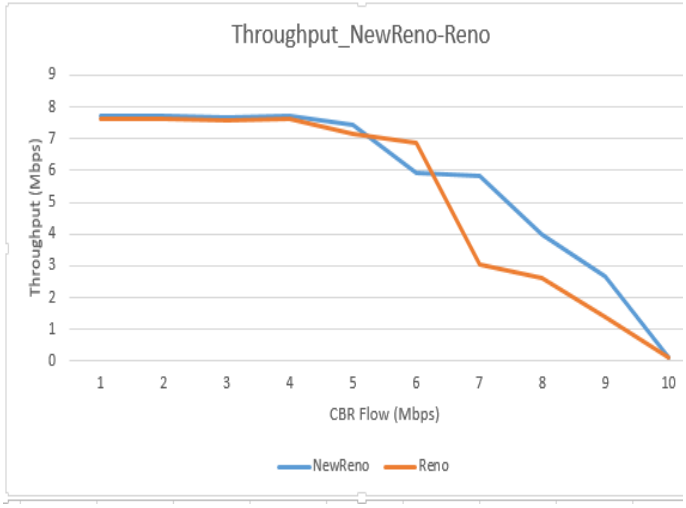

Fig 3.1: Throughput graph of New Reno-Reno

(3) Vegas/Vegas-

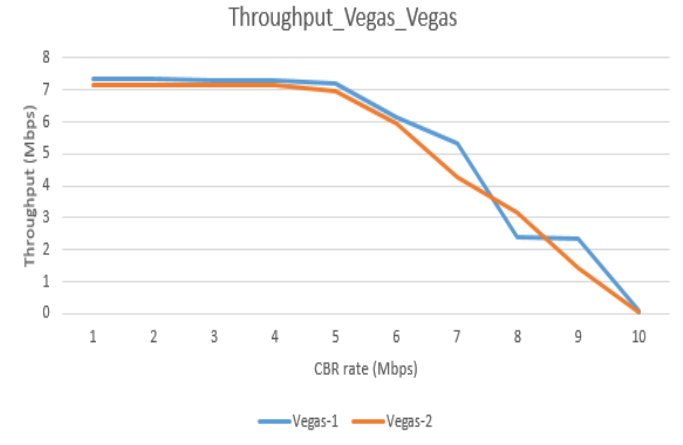

Fig 3.2: Throughput graph of Vegas-Vegas

Similar to Reno-Reno combination, TCP Vegas performs fairly with the other TCP Vegas. Fig 3.2 shows the throughput vs CBR flow rate graph which shows similar characteristics of both the TCP Vegas variants. Although there are a few oscillations, we can evidently say that both perform the same. Again, the reason for this fairness between two TCP Vegas variants is the use of same congestion detection algorithm, i.e. RTT detection.

(4) New Reno/Vegas-

As seen in Fig 3.3, we can say that as CBR rate increases, TCP New Reno performs better than TCP Vegas. This means that as the network starts getting congested and competence is needed for the limited resource, TCP New Reno is dominant over TCP Vegas and utilizes the resources more. Also, TCP New Reno is steadier than TCP Vegas. This means that TCP New Reno is more aggressive for resources and is able to control its own resources, while TCP Vegas has to depend on other variants. This is because TCP Vegas uses an early congestion detection and is more sensitive to congestion. Hence, it detects congestion early and slows down immediately. This in turn gives more bandwidth to TCP New Reno. Thus, we can conclude that TCP New Reno is not fair to TCP Vegas. For the very same reason of early congestion detection, we think that all other TCP Variants also may not be fair to TCP Vegas.

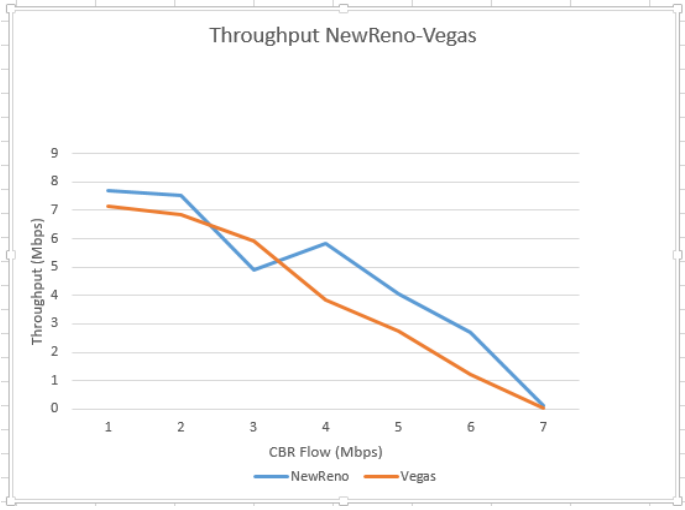

Fig 3.3: Throughput graph of New Reno-Vegas

V. EXPERIMENT THREE: Influence of Queuing

Methodology:

In this experiment, we study the influence of queuing disciplines like DropTail and Random Early Drop (RED).These algorithms are widely used by Internet routers to decide when to drop packets. To perform this experiment, we have setup a network topology as in the first experiment. An FTP agent has been created over TCP agent on Node N1. The FTP agent will trigger the TCP agent to transmit packets which flows from Node N1 to Node N4. The bandwidth of each link is set to 10 Mbps. A CBR flow over UDP has been created between node N5 and N6. First, TCP flow is started at time 0. Once the TCP flow is steady, the CBR source is started at 3

seconds and stopped at 10 seconds. The CBR flow rate is kept constant and we have observed the performance of TCP Reno, TCP SACK and CBR over time. For each pair of TCP variant/queue method, we set the short period of time granularity of 0.5s.

(1) Throughput Analysis:

We have analyzed the effect of queuing algorithms by comparing the throughput of various combinations like: SACK_DropTail vs SACK_RED vs Reno_DropTail vs Reno_RED.

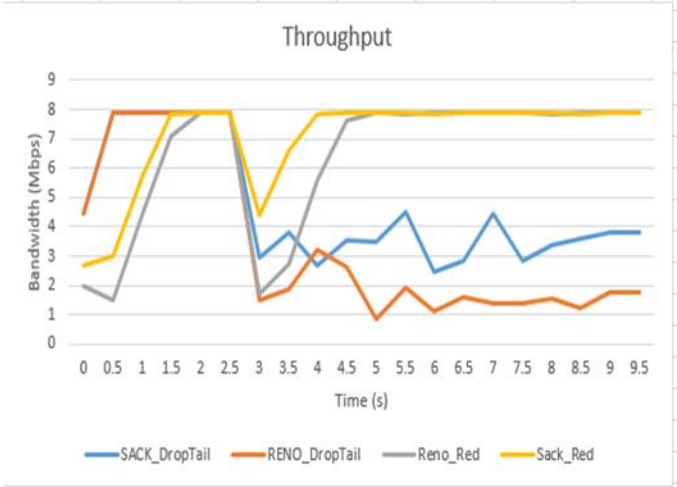

Fig 4.0: Throughput of SACK & Reno- RED & Drop-tail

From the Fig 4.0, we can say that before the CBR flow starts, the behavior of all the four variants in terms of bandwidth don't vary much. The Reno-Droptail variant performs better than the other variants before the CBR flow starts. When we start the CBR flow after 3s, we can infer from the graph that TCP RED provides higher throughput than TCP DropTail, irrespective of the type of TCP variants used. This is because, RED allows the congested link to slow more gracefully by statistically dropping packets from flows before it reaches its limit. On the other hand, DropTail allows queuing up of packets up to a certain amount, then starts to drop packets once its queue is filled. Overall we can say that, regardless of the TCP variant, RED performs better than DropTail when CBR flow exists. Also TCP SACK performs better than TCP RENO with RED queuing algorithm. This is because it works around the problem faced by TCP TENO, namely detection of multiple lost packets, and re-transmission of more than one lost packet per RTT.

(2) Latency Analysis:

We have analyzed the effect of queuing algorithms by comparing the latency of various combinations like: SACK_DropTail vs SACK_RED vs Reno_DropTail vs Reno_RED.

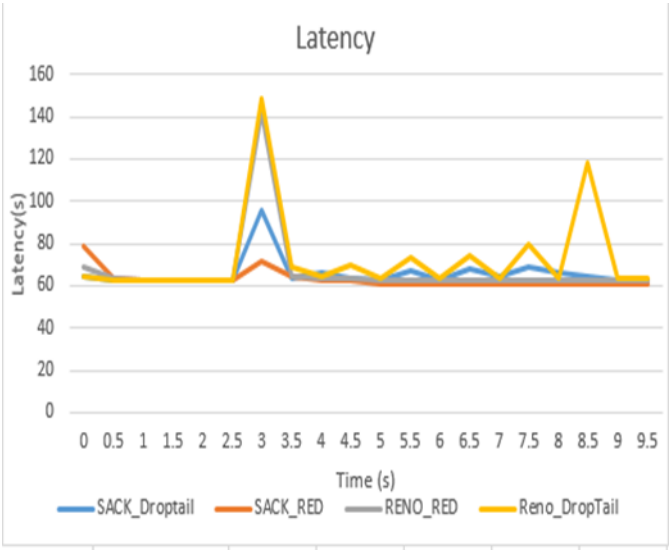

Fig 4.1: Latency of SACK & Reno- RED and Drop-tail

As we can see in the graph in Fig 4.1, during CBR slow start, TCP variant Reno and queue method DropTail have extremely high latency. This is because once the DropTail buffer queue is full, it enters retransmit synchronization and drops the packets arrived after its buffer queue is full. This sudden burst of packet drops will cause a delayed burst of retransmits, which will overfill the already congested network leading to higher latency. Overall, the flow with RED has a better performance of latency than DropTail regardless of the TCP variant used. This is because, RED statistically drops packet from flows before it reaches its hard limit. This allows the congested network to grow more gracefully thus preventing retransmit synchronization. RED overcomes some of the problems discovered in Drop-Tail such as synchronization of TCP flows and correlation of the drop events within a TCP flow thus achieving low end-to-end latency.

VI. CONCLUSION

In this project, we have simulated the TCP variants using NS-2 simulator and have compared the behavior of these under various load conditions and queuing algorithms. We have analyzed and compared these on the basis of certain parameters such as throughput, latency and packet drop rate. From the three experiments we conclude the following:

1. By observing the readings from T-value, mean and simulated graphs, we can conclude that TCP Vegas performs the best under different load conditions when compared to other TCP variants. It particularly has much better performance during heavy congestions.

2. By observing the simulated graphs of different combinations of TCP variants run together, we can conclude that when two connections of same variant are used, they act fair to each other. But when different TCP variants and used in a single link, they

are unfair to each other. Particularly, TCP Vegas has poor fairness when used with other TCP variants because of its early congestion detection algorithm.

3. By observing the simulated graphs of different combinations of TCP variants run together, we can conclude that RED performs well with both the TCP variants. RED performs exceptionally well with TCP SACK and provides low latency, higher throughput and stable transmission.

VII. REFERENCES

[1] Comparison and Analysis of Drop Tail and RED Queuing Methodology in PIM-DM Multicasting Network

http://www.ijcsit.com/docs/Volume%203/Vol3Issue2/ijcsit20120120302.pdf

[2] A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas

http://inst.eecs.berkeley.edu/~ee122/fa05/projects/Project2/SACKRENEVEGAS.pdf

[3] TCP Variants and Network Parameters: A Comprehensive Performance Analysis

http://www.iaeng.org/publication/IMECS2009/IMECS2009_pp351-353.pdf

[4] Studying TCP's Throughput and Goodput using NS

http://www.mathcs.emory.edu/~cheung/Courses/558/Syllabus/A4-TCP-Sim/TCP-Throughput.html

[5] Performance Analysis of TCP Congestion Control Algorithms

http://www.universitypress.org.uk/journals/cc/cc-27.pdf

[6] NS by Example

http://nile.wpi.edu/NS/

[7] Issues of TCP with SACK

http://www.icir.org/floyd/papers/issues_sa.pdf