Performance Analysis of TCP Variants

Dharanish Kedarisetti
College of Engineering
Northeastern University
Boston, MA-02115
kedarisetti.d@husky.neu.edu

Anika Ramachandran
College of Engineering
Northeastern University
Boston, MA-02115
ramachandran.an@husky.neu.edu

I. Introduction

In this study different TCP variants are compared with simulated results. The NS-2 network simulator is used to perform experiments, helps understand the behavior of these TCP variants under various load conditions and queuing algorithms. Also, it gives an idea about how these TCP variants can be compared and analyzed. The paper is divided into five parts:

In the first part, explanation is provided on how the experiments were conducted. What tools were used? Why were these tools used, and how do we know if they are reliable or not? It also covers the key parameters used during the experiments and why were they chosen.

An analysis of the first experiment with probable explanation of the data is presented in the second part. Each TCP variants behavior in the presence of congestion is compared and analyzed.

Our second experiment discussed in the third part we simulate two-TCP stream to analyses how the TCP streams share bandwidth and try to understand the fairness with respect to b

andwidth.

In the fourth part, guidelines are provided as for the first two experimental sections. The impact of different queuing techniques on each TCP variant are discussed. Analysis is conducted on why each TCP variant react to different types of queues.

Finally, the significance of the study is stated and the key results are highlighted.

II. Methodology

Tool: Ns2 is object oriented network simulator which simulates various LANs and WANs. Network protocols such as TCP, UDP can be implemented on Ns2. It can also implement various traffic sources like FTP, CBR and mechanisms for queue management like RED and Drop Tail.

We have considered the benchmarks for deciding the “best” TCP variant to be the following:

1) Throughput: Ratio of total data transferred to the time taken to complete the transfer.

2) Latency: Roundtrip time for a packet

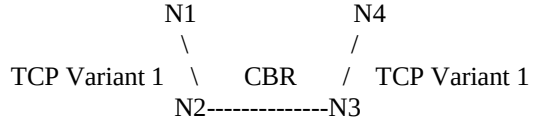
3) Packet Drop Rate: Fraction of packet drops during transmission for each packets sent

T-test is conducted for all combinations of TCP Variants for throughput for all possible conditions which checks weather the results derived are statistically significant.

Standard deviation is calculated for all combinations of TCP Variants.

III. Experiment 1

Topology:



In this experiment the performance of TCP variants (Tahoe, Reno, NewReno, and Vegas) are analyzed under the influence of various load conditions.

We used a network topology with four nodes. The bandwidth of the link from N1 to N2 and N3 to N4 was set to 20mbps and the link N2 to N3 bandwidth was set to 10mpbs. CBR source was added at N2 and a sink at N3, then we added a single TCP stream from N1 to a sink at N4. We recorded the performance of the TCP flow by changing the CBR's rate until it reached the bottleneck capacity.

We started the CBR flow at a rate of 1 Mbps and recorded the performance of the TCP flow. Then we increment the CBR flow's rate by 1 Mbps and repeat the test. We keep incrementing the CBR's rate until it reached the bottleneck capacity of 10mbps. We noticed that the TCP stream's performance will change depending on the amount of congestion. We conducted this experiment using the following TCP variants: Tahoe, Reno, NewReno, and Vegas. We analyzed the throughput, packet drop rate, and latency of the TCP stream as a function of the CBR flow rate.

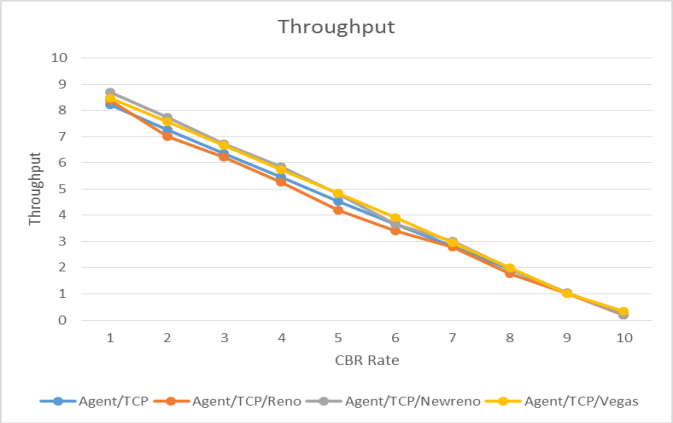


Fig. 1. Throughput of all TCP Variants w.r.t CBR rate

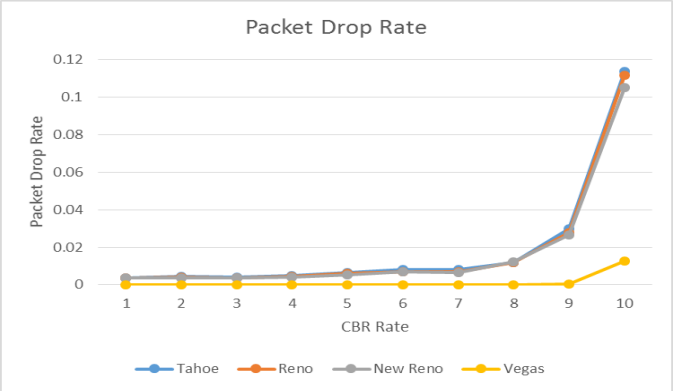


Fig. 2. Packet Drop Rate of all TCP Variants w.r.t CBR rate

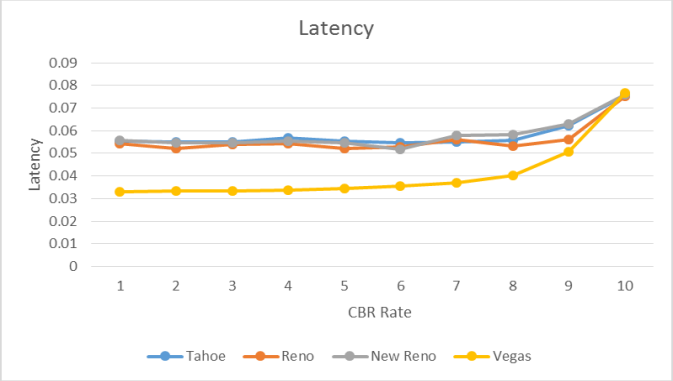


Fig. 3. Latency of all TCP Variants w.r.t CBR rate

Tahoe makes use of 'Additive Increase Multiplicative Decrease'. When a packet loss occurs, Tahoe sets the threshold value to half of instantaneous window. It sets CWD equal to one and performs slow start until it reaches the threshold. Then it linearly increases until a packet loss is encountered. As in Fig. 1. , although the throughput is better than Reno, Tahoe has a high latency as well as packet drop rate as compared to the other variants. This is because it takes Tahoe a complete timeout interval to detect a packet loss and assumes all the outstanding packets following the lost packet to be lost.

Reno uses slow start and re-transmit timer of the Tahoe TCP. Also, it detects loss of packets earlier and the pipeline is not emptied every time a packet is lost as was in case of TCP Tahoe by using the 'Fast Retransmit' algorithm. When three duplicate ACK's are received, it identifies that loss of the segment and re-transmits the segment without waiting for a timeout. Hence, retransmission of the segment is done with the pipe being almost full. As observed, the packet drop rate and latency is less than that of Tahoe, but it has a lower throughput than that of Tahoe. This is because TCP Reno reduces the window size more than required and hence it fails to recover fast.

NewReno is able to detect packet losses in the event of multiple packet losses and increase the efficiency. Similar to Reno, when multiple duplicate packets are received, it also enters fast-retransmit. The differentiating factor is that, unlike Reno, it does not exit fast-retransmit until all the acknowledgement for the data in the pipeline has been received. This is how it overcomes the issue of reducing the CWD multiple times faced by Reno. The graph shows that the throughput is high and the packet drop rate is low for New Reno. But the latency is higher for New Reno compared to Tahoe and Reno because TCP New Reno takes an entire RTT for detecting single packet loss.

Vegas resolves the issue of the requirement of enough duplicate ACKs for detecting packet losses, as it uses packet delay along with packet losses to detect network congestion. It also prevents the congestion of the network by using a modified slow start algorithm. Simulation results show Vegas has the less throughput, latency and least packet drop rate before congestion. As congestion is increased it showed increase in throughput share due to faster congestion detection.

Best TCP variant: Although New Reno has a higher average throughput before congestion but Vegas gains throughput when congestion increases and also has the lowest average latency with fewest drops. In this experiment, TCP Vegas can be considered as an overall "best" TCP variant.

T-Test:

T-test	Tahoe	Reno	NewReno	Vegas
Tahoe	0	8.41	5.86	15.3
Reno	8.41	0	16.7	29.1
NewReno	5.86	16.7	0	14.7
Vegas	15.3	29.1	14.7	0

T-Test calculated values

All P values calculated are < .1 indicating strong evidence against the null hypothesis. T-test was performed for the throughput for all CBR rates and by varying TCP start and stop time. It was observed that the values were high for all variants, showing that all the data recorded are statistically significant.

	Tahoe	Reno	NewReno	Vegas
Standard Deviation	0.070475	0.064758	0.040885	0.002257

Standard deviation was performed for the throughput for all the variants. It was observed that Vegas had the least standard

deviation compared to other variants, showing that TCP Vegas is most stable among the lot.

IV. Experiment 2

Topology:

```
                N1                  N4
TCP Variant 1   \                   /   TCP Variant 1
                  \      CBR      /
                   N2-------------N3
TCP Variant 2   /                   \   TCP Variant 2
                /                     \
               N5                     N6
```

This experiment was included to analyze the fairness between different TCP variants. Two variants are compared in the same environment and have to share bandwidth with a common CBR. The fairness of the protocols to one another was determined by comparing the throughput of each TCP flow.

We used a network topology of six nodes each set to a link bandwidth 10mbps. A CBR source was added at N2 and its sink at N3, a TCP stream was added from N1 and its sink at N4 and another TCP stream from N5 to a sink at N6. The performance at these TCP flows were measured by changing the CBR's rate until it reached the bottleneck capacity.

As in Experiment 1, average throughput of each variant was recorded by varying CBR bandwidth in steps of 1Mbps till bottleneck capacity of 10mbps. The throughputs for four pairs of TCP streams was analyzed as a function of the bandwidth used by the CBR flow.

For determining the fairness of the TCP variant, throughput, latency and packet drop rate is taken as the measuring parameter. Ideally, each variant should share equal bandwidth but it varies due to different protocol designs.

A. Vegas vs Vegas

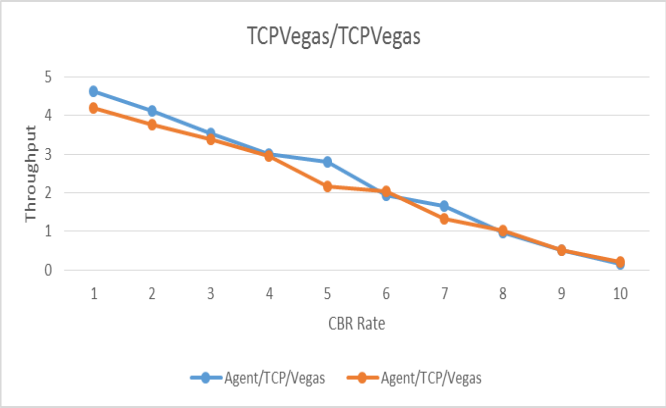


Fig. 4. Throughput comparison of Vegas vs Vegas

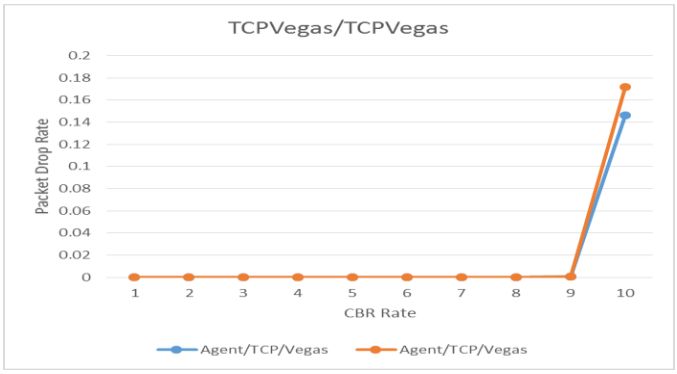


Fig. 5. Packet Drop Rate comparison of Vegas vs Vegas

Vegas differs from other TCP variants because it considers large queue building in the router as a sign of congestion rather than considering loss of segment for congestion detection. It brings down the sending rate lower than the expected rate. Therefore, in Vegas, when it notices that the calculated rate is too low compared to the expected rate, it increases its rate of transmissions to utilize the bandwidth available. It also decreases the transmissions when the calculated rate is very near to the expected rate which helps to avoid over saturation of bandwidth of the network. Unlike all other algorithms, which during the slow-start phase have no idea about the bandwidth of the connection available and can possibly over shoot the value of bandwidth share by a big amount (during exponential increase) contributing in congestion, Vegas increases its bandwidth share exponentially only after every RTT. Difference between the actual sending throughput and the expected one is calculated and when a specific threshold is crossed, slow start phase is exited and congestion avoidance phase is entered.

When there are two TCP Vegas, and one Vegas tries to increase the sending rate, the second one detects it and decreases its sending rate. As seen from the graph, the throughputs keep on changing. As soon as they become close to each other, one of it decreases its rate to avoid congestion. Both Vegas are fair to each other, and the oscillating graph is due to the congestion avoidance algorithm followed by TCP Vegas. It is also observed that the latency and packet drop for both Vegas are similar in value.

B. Vegas vs NewReno

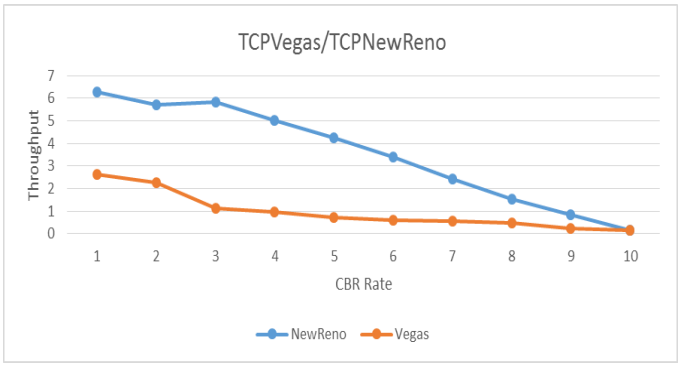


Fig. 6. Throughput comparison of Vegas vs NewReno

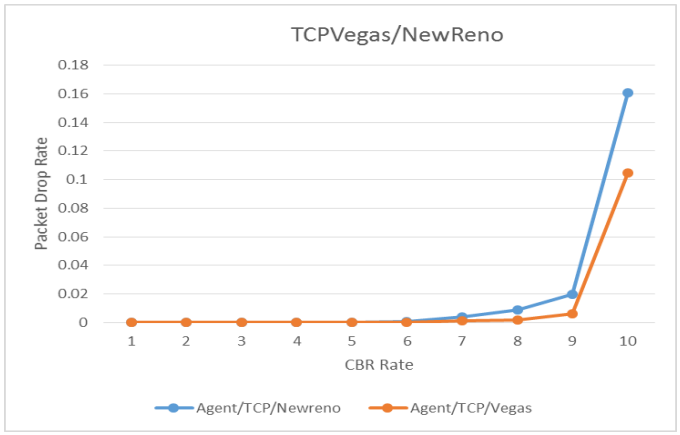


Fig. 7. Packet Drop Rate comparison of Vegas vs NewReno

Irrespective of the start time of the TCP variant, TCP NewReno takes most of the bandwidth share when it is paired with TCP Vegas. Detection of congestion in Vegas is faster as it happens after every RTT. Hence, it lowers its transmission rate. During this period, NewReno will take up majority of the bandwidth share as Vegas is not accessing it at that time and then it tries to maintain it. Hence, NewReno has a better throughput compared to Vegas. In this pair, TCP New Reno is unfair to Vegas even though it is observed that NewReno has higher latency as well as packet drop rate compared to Vegas.

C. Reno vs Reno

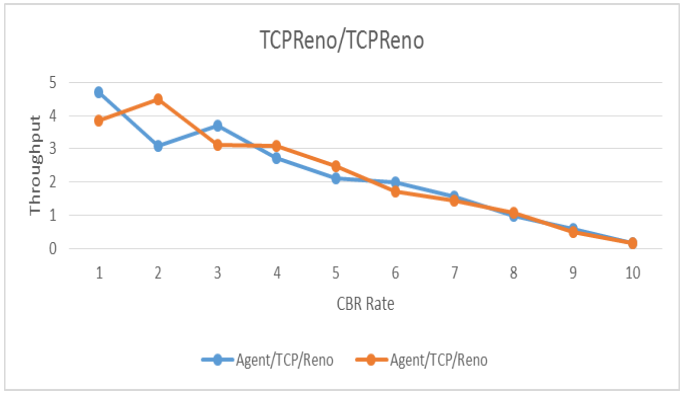


Fig. 8. Throughput comparison of Reno vs Reno

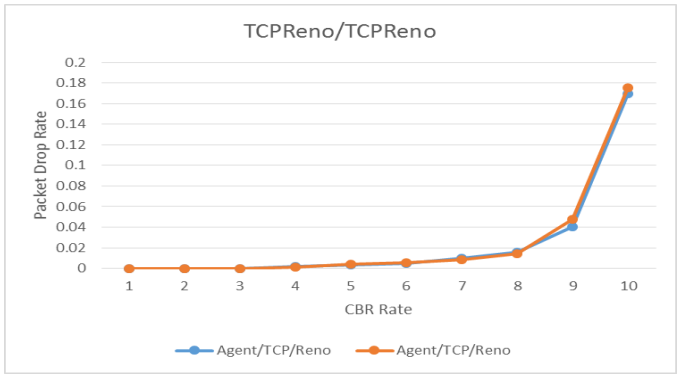


Fig. 9. Packet Drop Rate comparison of Reno vs Reno

From figure, we can observe that the throughput of both the TCP Reno's are almost similar. This shows that bandwidth sharing in the network is equal between them. It is also observed to have the same latency, therefore, when a pair of Reno and Reno shares a network bandwidth, they are fair to each other.

D. Reno vs NewReno

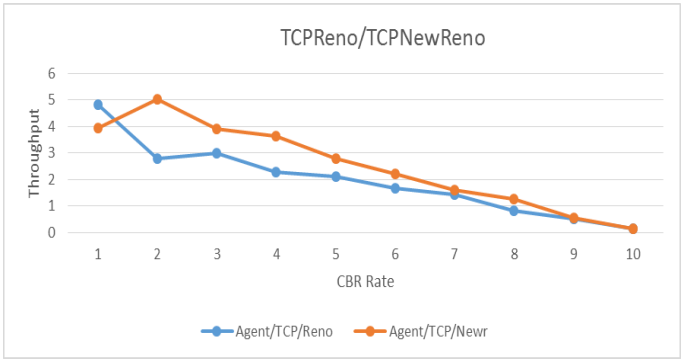


Fig. 10. Throughput comparison of Reno vs NewReno

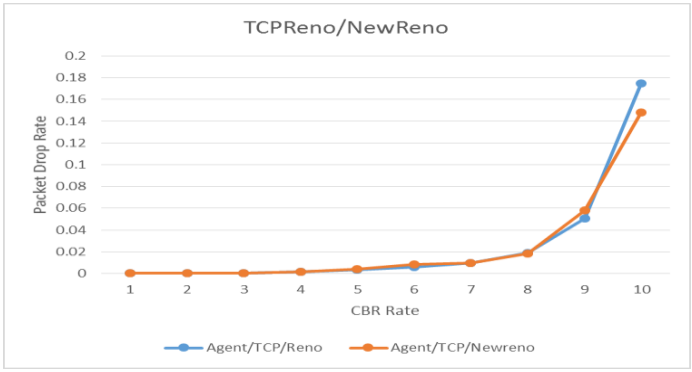
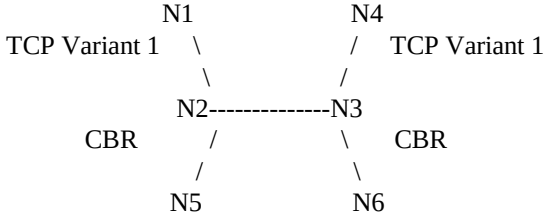


Fig. 11. Packet Drop Rate comparison of Reno vs NewReno

NewReno has higher throughput than Reno when they are sharing the bandwidth. This shows that this combination is unfair to each other. NewReno takes up more bandwidth irrespective of the start time of the TCPs. It is also observed that the packet drop and latency is on the higher side for Reno as compared to NewReno as NewReno can detect multiple packet losses and does not wait for the retransmit timeout to occur for retransmission of packet as in the case of Reno. This gives NewReno more priority in bandwidth share.

V. EXPERIMENT 3

Topology:



Experiment is performed to analyze the influence of queuing discipline used by the nodes on the throughput of the TCP flows. The effect on packet losses, throughput and latency is analyzed with different queue managements. Effect of CBR on a steady TCP is studied as well.

Network topology with four nodes is used. The bandwidth of all the links are set to 10mbps. A TCP stream from N1 to a sink at N4 is started and then a CBR source was added at N5 and its sink at N6. The performance of the TCP flow is analyzed by changing the CBR's rate until it reaches the bottleneck capacity. TCP Reno and SACK with the following two Queuing disciplines were used to study the influence on the overall throughput of flows:

1) DropTail
2) Random Early Drop (RED)

The CBR flow is not affected by the queuing algorithm as CBR does not care about the packet drops. However, TCP flows are affected by the queueing algorithm. Hence, it is clear that the algorithms are unfair to every flow eg: TCP v/s CBR.

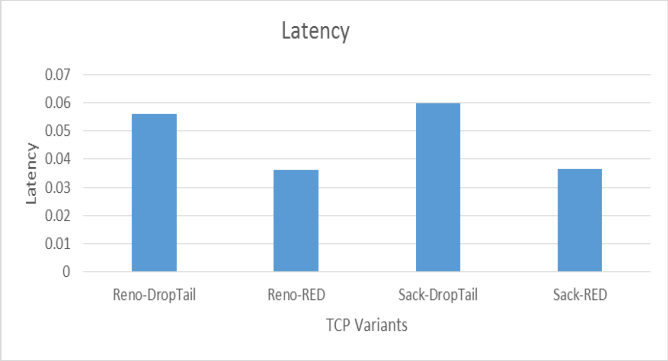


Fig. 12. Latency affected by the queuing algorithms

RED queuing protocol monitors the queue size and drops packets based on the queue capacity. Thus any packets reaching their destination will have a considerably smaller delay as they go through a monitored queue length. But, in case of DropTail, the queue size increases over time. Latency is therefore less for RED compared to DropTail.

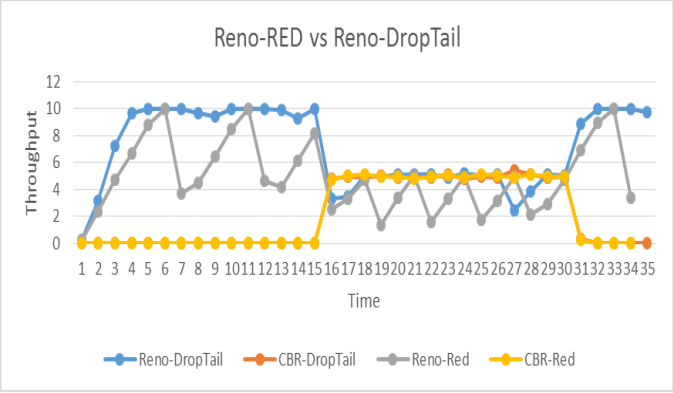


Fig. 13. Throughput comparison of Reno-RED vs Reno-DropTail

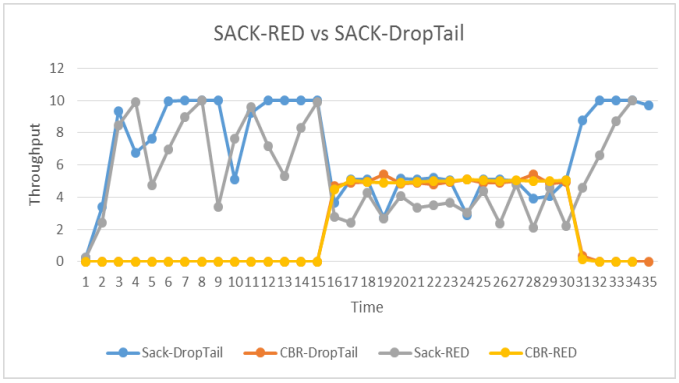


Fig. 14. Throughput comparison of SACK-RED vs SACK-DropTail

TCP throughput decreases significantly with the creation of the CBR flow. Starting of CBR increases the packet loss as the queue at node N2 is queued with packets from CBR as well as TCP. The queuing algorithm drops packets from CBR and TCP based on the queue size causing decrement in TCP throughput. As the CBR flow is stopped, the queue size decreases and the queuing algorithm accepts more packets from TCP. TCP then gradually increases its throughput till it detects a congestion.

As seen from the graphs, TCP Reno as well as SACK give a better throughput when using DropTail algorithm. When RED algorithm is being used, the throughput is seen to be fluctuating and comparatively less than it was when using DropTail, with each increase in CBR. This is because, DropTail only drops packet when the buffer is full and RED drops packet randomly when it detects congestion. It estimates the congestion of the network and drops packet accordingly. RED calculates the packet drop probability by using average queue length. Hence, the graph of throughput when the TCP uses RED algorithm is haphazard and not stable.

RED drops the packets randomly when it detects congestion. This can affect the TCP variants as this calls for resending of a large chunk of packets. But, in the case of SACK, it can detect selectively, which packet was lost and resend that particular one again. Hence, it can be concluded that RED is a good idea when dealing with SACK as latency is lower for RED and SACK can address random packet drops selectively.

	Reno-DropTail	Reno-RED	SACK-DropTail	SACK-RED
Standard Dev	2.415	1.714	2.397	1.909

Standard Deviation results show DropTail to be more stable than RED in both the cases.

CONCLUSION

By analyzing the results, the following conclusion can be made:

Different TCP Variants perform in different ways with respect to the congestion in the network. TCP Vegas has a better performance when compared to the other variants when congestion is increased. It gives a higher throughput and lower packet drop rate as well as latency than the other variants.

When combination of two TCP variants are used, it is not always fair as expected. Reno/Reno pair as well as Vegas/Vegas pair are fair to each other whereas, NewReno/Vegas and Reno/Reno are unfair to each other.

Though RED queuing algorithm gives lower throughput than DropTail and DropTail has higher latency than RED, these values are dependent on the TCP variants used to study them.

Each variant though behaves as expected individually, variations are observed depending on the interfering flows and queuing protocols under different circumstances. In real life scenarios the observed results would be much complex as the number of nodes combined with individual source and sink parameters would probably make the network behave in a different manner than predicted. Experiments such as these help us understand the basic in-depth behavior and help us choose parameters, protocols according to the network requirements as the implementation of each variant plays an important role in any network.

It would be interesting to see how TCP Vegas would perform in networks using RED queuing algorithm, as RED drops packets on congestion and has low latency while Vegas would always detect no congestion (if in case its packets weren't dropped). We would like to cover this in our future work.

REFERENCES

[1] A Comparative Analysis of TCP Tahoe, Reno, New-Reno, SACK and Vegas, http://inst.eecs.berkeley.edu

[2] An Investigation into Transport Protocols and Data Transport Applications Over High Performance Networks, Yee-Ting Li, University College London

[3] Analytic Models of TCP Performance, Debessay Fesehaye Kassa

[4] Analytic Models for the Latency and Steady-State Throughput of TCP Tahoe, Reno and SACK, B. Sikdar, S. Kalyanaraman and K. S. Vastola Dept. of ECSE, Rensselaer Polytechnic Institute