VOL 3 (2019) NO 3

e-ISSN : 2549-9904
ISSN : 2549-9610

INTERNATIONAL JOURNAL ON INFORMATICS VISUALIZATION

Performance Evaluation of TCP Vegas over TCP Reno and TCP New Reno over TCP Reno

Tanjia Chowdhury[#], Mohammad Jahangir Alam[#]

[#] *Department of Computer Science and IT, Southern University Bangladesh, Bangladesh*
E-mail: tanjiacse5@gmail.com, jahangir@southern.edu.bd

***Abstract*— In the Transport layer, there are two types of Internet Protocol are worked, namely- Transmission Control Protocol (TCP) and User datagram protocol (UDP). TCP provides connection-oriented service and it can handle congestion control, flow control, and error detection whereas UDP does not provide any of service. TCP has several congestion control mechanisms such as TCP Reno, TCP Vegas, TCP New Reno, TCP Tahoe, etc. In this paper, we have focused on the behavior performance between TCP Reno and TCP Vegas, TCP New Reno over TCP Reno, when they share the same bottleneck link at the router. For instigating this situation, we used drop-tail and RED algorithm at the router and used NS-2 simulator for simulation. From the simulation results, we have observed that the performance of TCP Reno and TCP Vegas is different in two cases. In drop tail algorithm, TCP Reno achieves better Performance and throughput and act more an aggressive than Vegas. In Random Early Detection (RED) algorithm, both of congestion control mechanism provides better fair service when they coexist at the same link. TCP NewReno provides better performance than TCP Reno.**

***Keywords*— Drop-tail algorithm, Fairness, RED, TCP Congestion control.**

I. INTRODUCTION

There are two internet protocol are used in networking system namely: UDP and TCP. The protocol defines the collection of rules associated with the communication system. By using the protocol, data moves one host to another host. The protocol contains three types of characteristics like- Syntax, semantics, and timing. The transmission control protocol (TCP) defines a set of rules which used to send data in the form of bit streams between host/device over the Internet. TCP is a connection-oriented protocol, means before delivery of data TCP establish the handshaking procedure. Handshaking procedure defines when the sender sends data to the receiver or receiver receive data from the sender, they both of gets acknowledgment from both sides. TCP provides reliable data transfer, congestion and flow control, and error detection whereas UDP does not provide all of the services because UDP is a connectionless protocol. TCP ensures a long message is divided into the small packets and reunite the packets into the complete message at the receiver side. Congestion-control mechanism occurs in the network layer, it means there is more data in a queue when they are ready to send. If the sending rate and receiving rate is not equal in the network, then congestion occurs. To overcome this problem, we used the Flow control mechanism, means controls the sending and receiving transmission rate when congestion occurs in the network. In this paper, we have evaluated the performance of congestion-control mechanisms such as TCP Reno and TCP Vegas when they coexist at the same link and TCP New Reno over TCP Reno. Fairness defines sharing the bandwidth equally when they exist at the same link. To describe our simulation, we have to know some term related to network systems like- Throughput, propagation delay and Round-Trip time and window size. Throughput or network throughput defines the average rate of message delivery over a physical or logical link. The unit of throughput is measured in bits per second (bps). Propagation delay is the time required to propagate from one router to another router which is separated by the propagation speed. The formula of propagation delay is d/s, where d is the distance of two routers and s is the propagation speed of the link. The round-trip time is the travel time which means a bit takes time for visiting a sending device to a receiving device and then back again to the sending device on the same network, which is recognized as Round-Trip-Propagation delay (RTPD). Window-size defines the total amount of data which will be received before the acknowledgment. This sliding window is dynamically changeable. From the previous paper analysis, there are several related papers exists which explained about the fairness between TCP Reno and Vegas.

II. Literature Review

We have studied related paper on Fairness of TCP congestion control. KM. Avni Yadav et al. was used additive increase and multiplicative decrease (AIMD) algorithm for controlling TCP congestion. In this paper, they recommended a new Tax based AIMD method for TCP Congestion Control. This paper showed a comparison of a different queue management algorithm. From the comparison, it's showed that TAX AIMD algorithms provide better performance than other algorithms [1]. Dharamdas Kumar et al. was proposed an Active Queue Management (AQM) algorithm for evaluating performance as well as optimal (minimum) queue length in network congestion control. This method provides higher control in congestion control [2]. Domanski, Adam, et al. was used fluid flow approximation to investigate the performance of the TCP NewReno and Vegas congestion control mechanisms on CWND evolution, packet loss probability, queue length, and its variability. In this method, TCP Vegas provide fair service than Reno [3]. There is another paper which presents a comparison of TCP variants for Congestion Control in the network. They considered the basis of various performance metrics such as end-to-end wait, throughput, and queue dimension and packet delay rate using Network Simulator-2 (NS-2). In high congested network, Vegas gives the best performance and in the low cohesive network, Reno gives the best result [4]. Zhou, Liu Wenjia, et al. was introduced a modification approach called DYNAMIC Vegas which adopt the slow start algorithm and modifies the increase or decrease rate in congestion avoidance phase [5]. In this paper an analytic model [6] for the throughput of TCP NewReno is described that builds upon the well known Reno model. This paper used three important ways to differentiate NewReno from Reno.They are fast-recovery algorithm, timeout behavior- timeout during fast-recovery is also considered, and two-parameter loss model- considers both the loss event rate and burstiness of segment losses. There are lots of paper we studied [8]-[12], where TCP congestion control and their performance described.

III. Overview Of Drop-Tail And Red

In TCP Reno, the window size is increased when packet loss occurs at the buffer. In sender side TCP, it takes 1 RTT (round trip time) [sec] to detect the packet loss at the router buffer. In TCP Vegas connections, the window size is controlled by RTTs of the sending packets and the number of buffered packets remain in the router buffer from α to β [packets]. In TCP Vegas connections, the window size is decreased when the RTTs became larger and also reduced the throughput of the connection. If we controlled the window size, then packet loss doesn't occur in the connection. In TCP Reno connections, the window size is increased when the RTTs became larger. In TCP Vegas, the window size is updated by applying the congestion avoidance formula:

$$cwnd(t+t_A) = \begin{cases} cwnd(t)+1, if\ diff < \alpha \\ cwnd(t), if\ \alpha \le diff \le \beta \\ cwnd(t)-1, if\ diff > \beta \end{cases} \quad (1)$$

Therefore, the total window size, W_v and the average value of W_v, $\overline{W_v}[packets]$ of the TCP Vegas connections are achieved by using the formula [7]:

$$N_v\,\alpha < W_v < N_v\,\beta \quad (2)$$

$$\overline{W_v} = N_v \frac{\alpha+\beta}{2} \quad (3)$$

In TCP Reno connections, the total window size, W_r $[packets]$ is obtained as follows [7]:

$$W_r = 2\tau\mu + B - W_v \quad (4)$$

In TCP Reno, the window size is updated by applying the congestion avoidance formula:

$$cwnd(t+t_A) = \begin{cases} (Slow\ start\ phase:) \\ cwnd(t)+1, if\ cwnd(t) < ssth; \\ (congestion\ avoidance\ phase:) \\ cwnd(t)+\frac{1}{cwnd(t)}, if\ cwnd(t) \ge ssth; \end{cases} \quad (5)$$

The probability of the packet loss for each connection is proportional to its window size. In buffer overflow of the network, the numbers of packet losses of TCP Reno, $L_r[packets]$ and Vegas connections $L_v[packets]$, are obtained as follows [7]:

$$L_r = N_r \frac{W_r}{W_r+W_v} \quad (6)$$

$$L_v = N_r \frac{W_v}{W_r+W_v} \quad (7)$$

After the buffer overflow, we can obtain the total window size of the TCP Reno connections by using the below formula [7]:

$$\begin{aligned} W_r' &= \frac{1}{2}\frac{W_r}{N_r}.L_r + \frac{W_r}{N_r}(N_r - L_r) \quad (8) \\ &= \frac{1}{2}\times\frac{W_r}{N_r}\times N_r \times \frac{W_r}{W_r+W_v} + \frac{W_r}{N_r}\left(N_r - \frac{N_r W_r}{W_r+W_v}\right) \\ &= \frac{W_r^2}{2(W_r+W_v)} + \frac{W_r}{N_r}\left(\frac{W_r N_r + W_v N_r - W_r N_r}{W_r+W_v}\right) \\ &= \frac{W_r^2}{2(W_r+W_v)} + \frac{W_r W_v N_r}{N_r(W_r+W_v)} \\ &= \frac{W_r^2}{2(W_r+W_v)} + \frac{W_r W_v}{(W_r+W_v)} \\ &= \frac{W_r^2+2W_r W_v}{2(W_r+W_v)} \\ &= \frac{W_r+2W_v}{2(W_r+W_v)} W_r \quad (9) \end{aligned}$$

Finally, the average value of the total window size, $\overline{W_r}$ of TCP Reno connections [7]:

$$\overline{W_r} = \frac{\frac{1}{2}(W_r+W_r')\frac{W_r-W_r'}{N_r}+W_r}{\frac{W_r-W_r'}{N_r}+1} \quad (10)$$

Now, the average number of packets at the router buffer of TCP Reno and TCP Vegas,

$$\overline{B_r} = \overline{W_r}\frac{B}{2\tau\mu+B} \quad (11)$$

$$\overline{B_v} = \overline{W_v}\frac{B}{2\tau\mu+B} \quad (12)$$

Finally, the average throughput of the connections of TCP Reno and TCP Vegas can be calculated as follows [7]

$$\rho_r = \mu \frac{E_r}{E_r + E_v} \quad (13)$$

$$\rho_v = \mu \frac{E_v}{E_r + E_v} \quad (14)$$

In the RED algorithm, the probability p is used for all packet losses when buffer overflow free. Each connection can send 1/p packets in one cycle on the average. For one cycle, the number of packets transmission is

$$N_p = \frac{1}{p}$$

Here, retransmission timeout expiration and fast retransmit algorithm are used to detect the packet loss based on the window size. In retransmission timeout expiration, throughput remains large. We can calculate the probability of occurring timeout expiration in the window by [8]

$$P_{to} = \sum_{i=2}^{W_r} \binom{W_r}{i} . p^i . (1-p)^{W_r - i} \quad (15)$$

If retransmission timeout expiration occurs, the window size is reset to 1 [packet] and the slow start is updated by $\frac{W_r}{2}$ $[packets]$. So, the time duration of the slow start phase, $T_{to,1}[sec]$ [7]

$$T_{to,1} = rtt . \log_2 \frac{W_r}{2} \quad (16)$$

The number of packets, $A_{to,1}[packets]$ transmitted in the slow start phase as follows [7]:

$$A_{to,1} = \frac{W_r}{2} - 1 \quad (17)$$

In the congestion avoidance phase, the window size is increased by 1 [packet] per RTT [sec] and the time duration, $T_{to,2}[sec]$ is [7]:

$$T_{to,2} = rtt . \left(\overline{W_r} - \frac{W_r}{2} \right) \quad (18)$$

The number of transmitted packets, $A_{to,2}[packets]$ in the congestion avoidance phase, is:

$$A_{to,2} = \frac{1}{2} \left(\overline{W_r} + \frac{W_r}{2} \right) \left(\overline{W_r} - \frac{W_r}{2} \right) \quad (19)$$

In the slow start phase, the time durations, $T_{fr,1}$ and the number of transmitted packets $A_{fr,1}$ is zero. In the congestion avoidance phase, the time durations and the number of transmitted packets is [7],

$$T_{fr,2} = rtt . \left(\overline{W_r} - \frac{W_r}{2} \right) \quad (20)$$

$$A_{fr,2} = \frac{1}{2} \left(\overline{W_r} + \frac{W_r}{2} \right) \left(\overline{W_r} - \frac{W_r}{2} \right) \quad (21)$$

Finally, the number of transmitted packets in one cycle and window size, $\overline{W_r}$ can be calculated by using the formula [7]:

$$\frac{1}{p} - P_{to} \left(A_{to,1} + A_{to,2} \right) + (1 - P_{to}) \left(A_{fr,1} + A_{fr,2} \right) \quad (22)$$

$$\overline{W_r} = rtt . P_{to} \left(\frac{A_{to1} + A_{to2}}{T_{to,1} + T_{to,2} + rto} \right) + rtt . (1 - P_{to}) \left(\frac{A_{fr,1} + A_{fr,1}}{T_{fr,:} + T_{fr,1}} \right) \quad (23)$$

The total window size of all TCP Reno connections;

$$\overline{W_r} = N_r . \overline{W_r} \quad (24)$$

Finally, ρ_r and ρ_v in the RED case is similar to the drop-tail case.

TABLE I
DESCRIPTION OF PARAMETERS

Parameter	Definition
α ,β	Parameters for controlling the change of cwnd
μ	Bandwidth of the bottleneck link
τ	Total propagation delay
B	Buffer size
rto [sec]	Retransmission timeout value of the connection.
rtt [sec]	Mean value of RTT of sending packet
N_r	Sender hosts of TCP Reno
N_v	Sender hosts of TCP Vegas

IV. PROPOSED NETWORK MODEL AND SIMULATION RESULTS

For TCP Reno and TCP Vegas:

In the simulation, a simple dumbbell network topology is used where TCP Reno and TCP Vegas connections coexist at the same bottleneck link. We used the bandwidth of access link between S1 to R1 is 2Mb and propagation delay is 10ms. The bandwidth of access link between S2 to R1 is 2Mb and propagation delay is 10ms and so on. The bandwidth of bottleneck link is 0.3Mb and the propagation delay is 200ms. The bottleneck capacity is 10 packets. For the RED queue management, the minthresh and maxthresh are set to 5[packets] and 0.6[packets]. The advertised window size by the receiver is 8000bytes. The packet size is 552 byte. To compare the fairness between TCP Reno and Vegas, drop-tail and RED algorithms are used at the router. The dumbbell topology which considers for implementation is shown below:

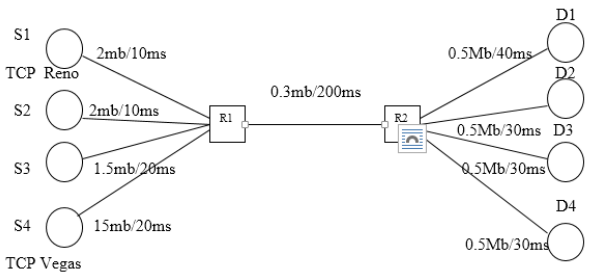

Fig. 1 Proposed Dumbbell topology

For TCP NewReno over TCP Reno:

The simulation is done for both single and multiple bottleneck-link. In the case of single bottleneck-link, two types of queue management mechanisms are used- DropTail and RED. All experiments have two long-duration flows: one NewReno flow and one Reno flow. Here three TCP sources and one UDP source are used. The bottleneck bandwidths vary from 0.7Mbps to 2Mbps. Here two types of traffic flows are used- FTP and constant bit rate UDP flows. The propagation delays vary from 1ms to 20ms. For RED

queue management, the minthresh and the maxthresh are set to 1/10 and 3/10 of the corresponding queue size limit. The maximum congestion window size is 8KB and packet size is 0.5KB. The UDP flow uses a rate of 0.01Mbps, the packet size is 1KB, and the propagation delay is 7ms. The FTP flows start at uniformly distributed times between 0 and 7 seconds.

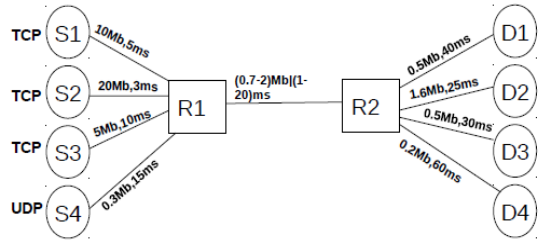

Fig. 2 A dumbbell topology for single bottleneck link

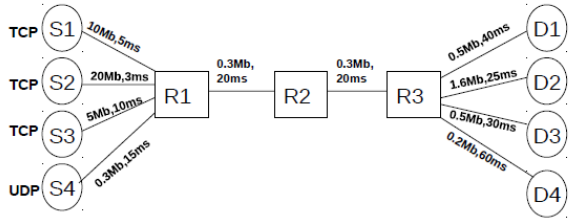

Fig. 3 Dumbbell topology for multiple bottleneck link

A. *Effect of drop-tail*

In TCP Vegas, the window size is converged to the fixed value and no packet loss occurs. It is different from TCP Reno in which the packet loss is necessary to invoke the congestion control. Comparison between congestion window of TCP Reno and TCP Vegas is shown in figure 1. Figure 4(a) and 4(b) shows that the time-dependent behavior of the window size of TCP Reno and TCP Vegas.

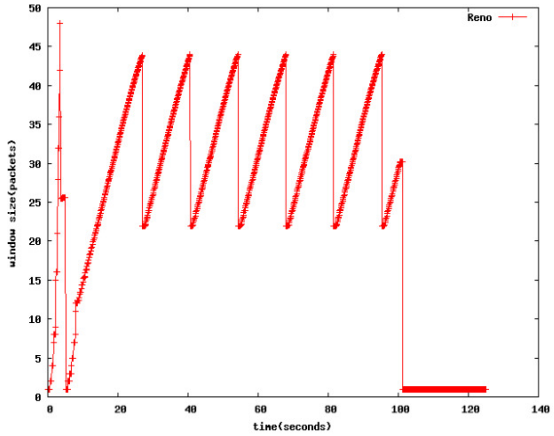

Fig. 4(a) The change of window size of TCP Reno

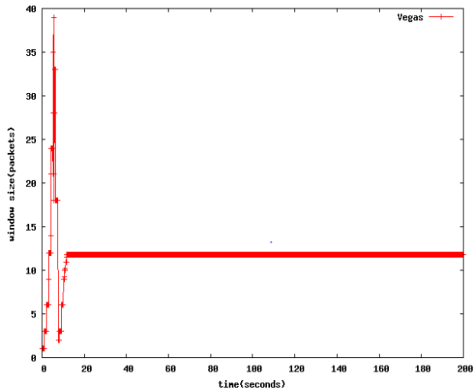

Fig. 4(b) The change of window size of TCP Vegas

In TCP Vegas, the window size is met to the fixed value and no packet loss occurs. It is different from TCP Reno in which the packet loss is necessary to invoke the congestion control which is shown in figure 4(a).

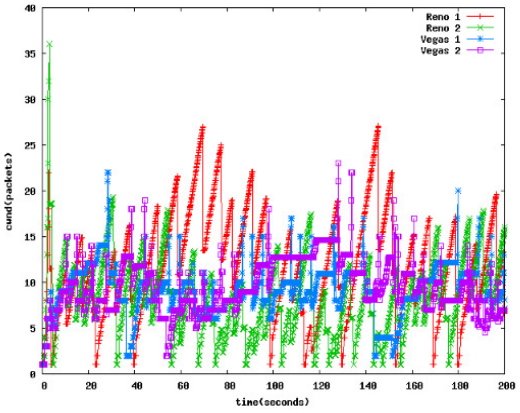

Fig. 5 Comparing the congestion window of TCP Reno and TCP Vegas

To investigate fairness between two connections, we mainly focus on changes of the congestion window. Figure 5 shows that TCP Vegas changes its congestion window according to RTT whereas TCP Reno does not depend on RTT. TCP Reno increases its congestion window until it detects any loss and it has an ability to keep a fair service among connections, on the other hand, TCP Vegas can change congestion window before any loss occur, by observing RTT of sending packets. TCP Vegas cannot achieve fairness between connections because of the difference of base RTT of two connections. When connection 2 joins the network, the switch buffer is occupied by several segments of connection 1. Thus, base RTT of connection 2 includes some buffering delay at the switch and it becomes larger than that of connection 1. Therefore, the window size of connection 2/TCP Vegas becomes lower to satisfy the equation (7).

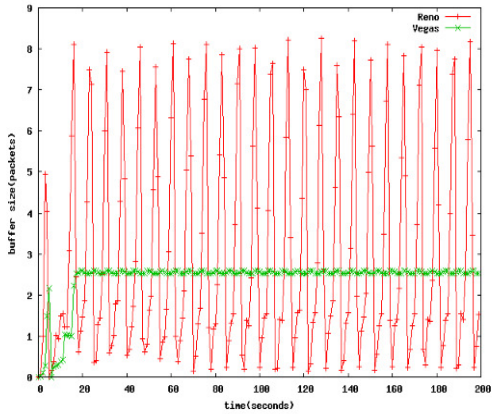

Fig. 6: Comparing the buffer occupancy of TCP Reno and TCP Vegas

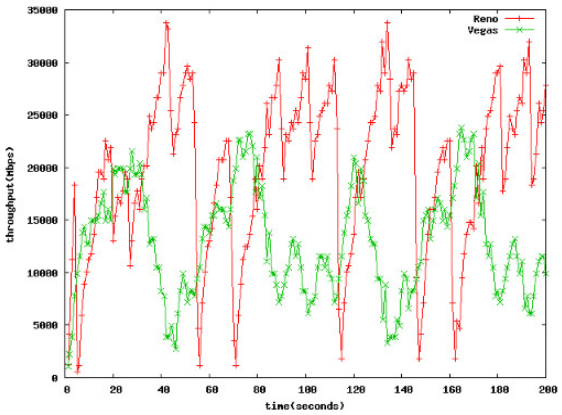

Fig. 7 Comparisons of average throughput of TCP Reno and TCP Vegas

Figures 6 and 7 show that the buffer occupancy of the TCP Reno connections and TCP Vegas connections and the average throughput of two versions of TCP. From the above figures, we can say that when TCP Reno and TCP Vegas connections share the bottleneck link, TCP Reno connection occupies the link bandwidth. Then, TCP Vegas connection suffers from significant performance dilapidation and low throughput compared with TCP Reno connection because of the difference of buffer occupancy at the router. TCP Vegas does not provide fair service when the buffer size is too small, because the window size of TCP Vegas connection is fixed and cannot increase their window size larger than β whereas TCP Reno connections increase their window size until the buffer becomes full and packet loss occurs.

TCP Vegas increases its window size conservatively, which means that if RTT becomes large, it decreases the window size. On the other hand, TCP Reno increases the window size aggressively until a packet is lost.

For TCP NewReno and TCP Reno:

In case of Drop Tail queue router, a single bottleneck link is used that has 0.7Mbps bandwidth, buffer limit is of 20 packets. The congestion window is of 8KB, the roundtrip propagation delay for each TCP flow is 20ms. In this simulation process, the change of congestion window size over time is graphically shown in figure 8 and 9.

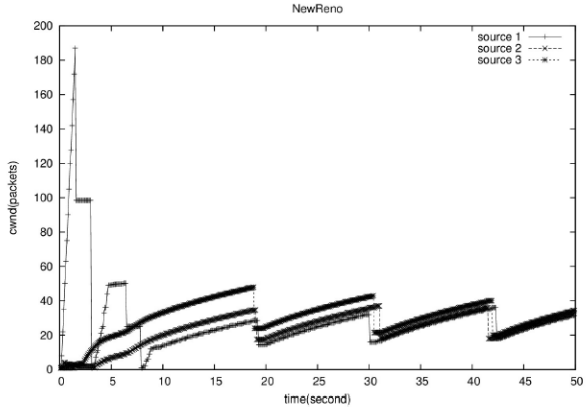

Fig. 8 Congestion window change in NewReno

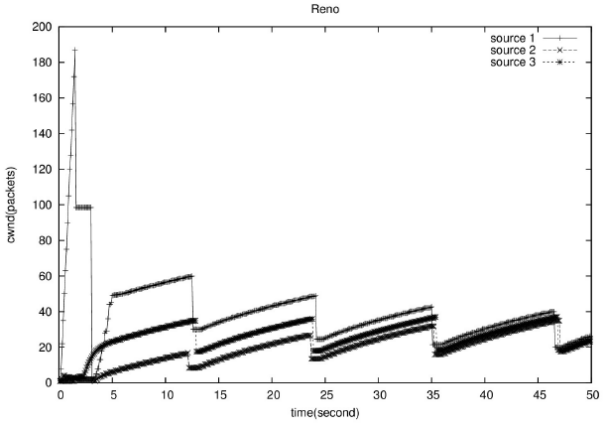

Fig. 9 Congestion window change in Reno

From the above figures, it is shown that the Reno needs more multiple reductions of the congestion window than the NewReno because of its multiple entries. The figure given below shows the throughput obtained in Reno and NewReno.

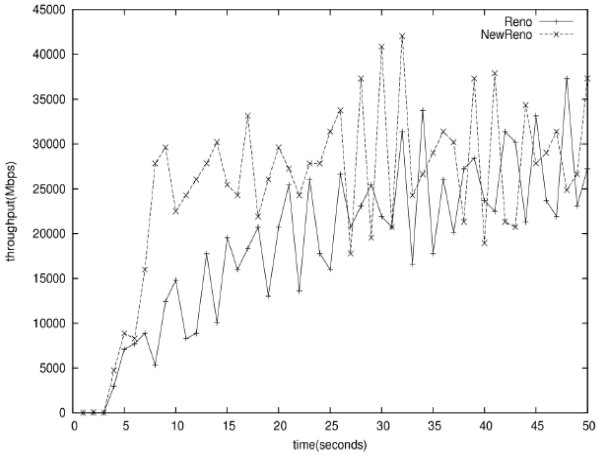

Fig. 10 Throughput change over time

From the above figure, it obtained that in case of the DropTail queue management the throughput of TCP NewReno is about 15%-25% higher than the Reno. The packet loss against time during data transmission is shown in the below figure-

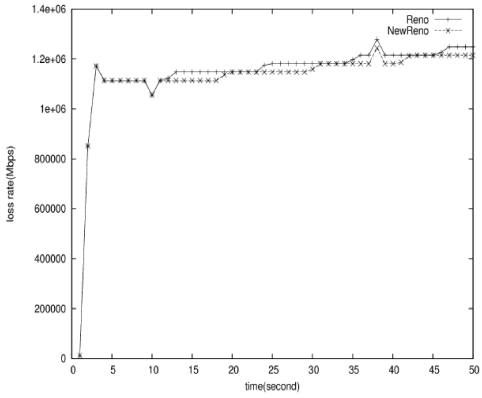

Fig. 11 Packet loss rate with time.

From the simulation result, we obtain that the NewReno suffers from 2%-3% less loss rate than the Reno model.

B. Effect of Random Early Detection

RED uses a probability for dropping packets. Packets losses occur when buffer overflow is created. Comparison between the congestion window of TCP Reno and TCP Vegas are shown in figure 12.

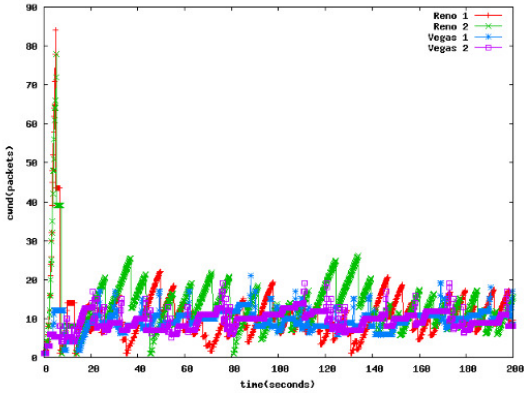

Fig. 12 Comparing the congestion window of TCP Reno and TCP Vegas

From this figure, we can see that, TCP Reno connection cannot expand their window sizes until the router buffer become fully-utilized, since packet losses occur before the buffer becomes full, which is caused by the RED algorithm. Figure 13 and 14 shows that the buffer occupancy of the TCP Reno connections and TCP Vegas connections and the average throughput of two versions of TCP.

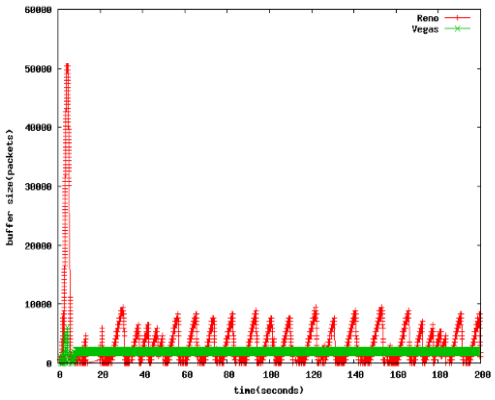

Fig. 13 Comparing the buffer occupancy of TCP Reno and TCP Vegas

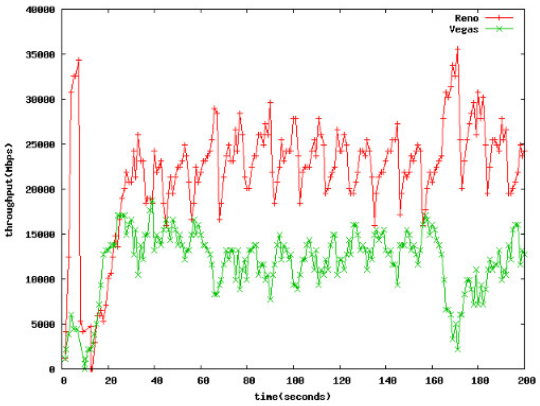

Fig. 14 Comparisons of average throughput of TCP Reno and TCP Vegas

When the RED algorithm is used in the router buffer the buffer occupancy of TCP Reno connections is weakened which decreases the difference of the throughput of the TCP Reno and TCP Vegas connections.

For TCP NewReno over TCP Reno:

With RED queue the bottleneck link has the bandwidth of 2Mbps, propagation delay is 1ms, buffer limit is of 50 packets, the minthresh is set to 5, maxthresh is 15, and congestion window size is 2KB. Here 25 flows are used with per TCP source. The congestion window changes over time during this session are shown in figure 15 and 16:

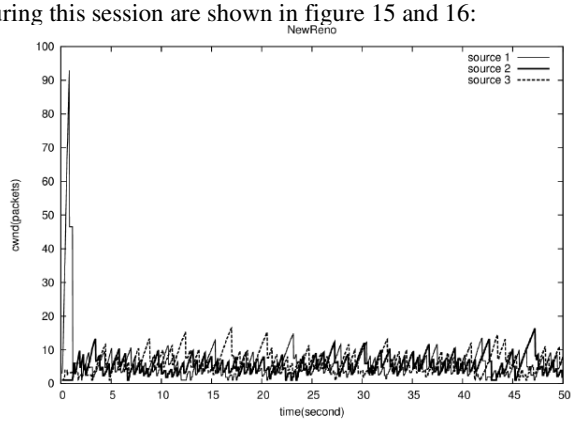

Fig. 15 Congestion window change in NewReno

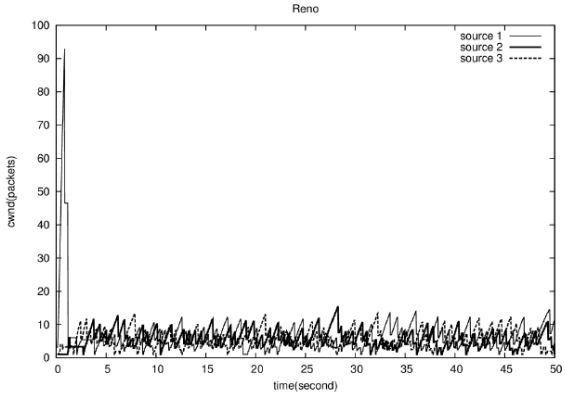

Fig. 16 Congestion window change in Reno

Comparison of throughput changes over time between these two models has shown in figure 17:

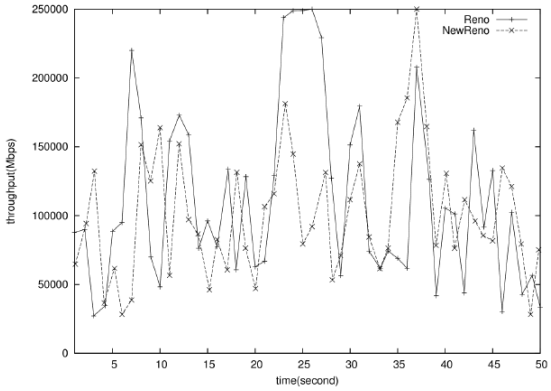

Fig. 17 Throughput change over time

From the simulation, it is found that TCP NewReno provides 10%-15% higher throughput than the Reno. The loss rate scenarios are shown below-

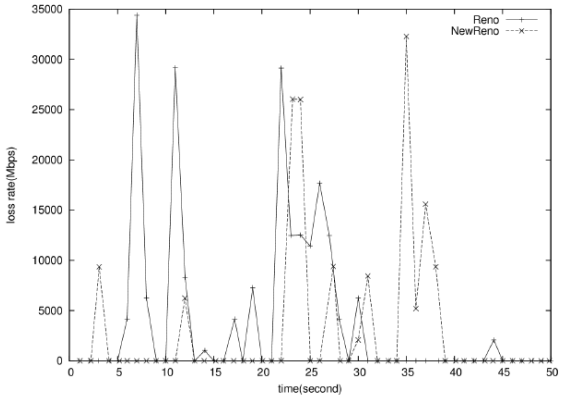

Fig. 18 Packet loss rate with time

The simulation result shows that Reno suffers from 20%-25% more losses than NewReno.

Multiple Bottleneck

Now the experiment is conducted with two bottleneck links connected in series. Each link has a bandwidth of 0.3Mbps, propagation delay 20ms, and buffer size is 10 packets. The congestion window size is 8KB and packet size is 552bytes. The queue management algorithm is DropTail. The throughput change for each model with time is shown in figure 19-

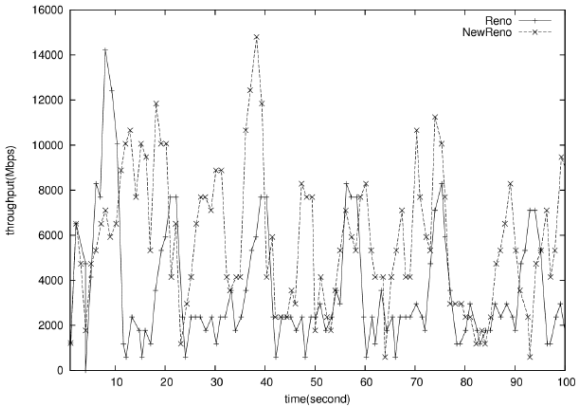

Fig. 19 Throughput change over time

In case of multiple bottlenecks, TCP NewReno provides 10%-11% more throughput than TCP Reno. In figure 10 loss rates for these two model with multiple bottlenecks are shown

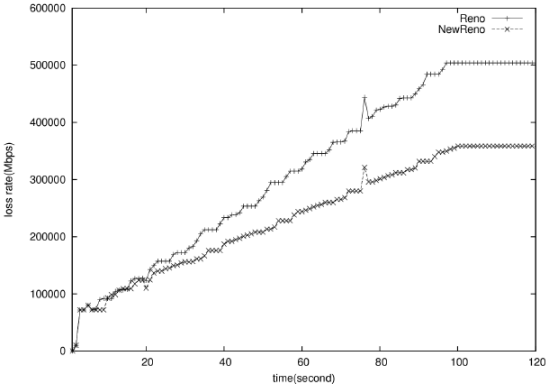

Fig. 20 Packet loss rate with time

From the simulation results, it is found that Reno suffers from 20%-25% losses than New Reno

V. CONCLUSIONS

In this paper, we described the TCP congestion control mechanism like TCP Reno and TCP Vegas; TCP NewReno and TCP Reno on based their window size, buffer occupancy, average throughput and loss of the packet. From the simulation results, it is derived that the TCP Vegas does not provide better performance with drop-tail because of the difference of buffer occupancy of the router buffer. But in the case of the RED algorithm, they provide better performance. In the simulation, the performance of TCP NewReno and TCP Reno are also compared using ns-2 simulator. The performance is evaluated in case of single and multiple bottleneck links, and two queue management mechanisms- DropTail and RED. The results from the simulation illustrate the significant performance advantages of NewReno over Reno. The simulation results indicate that TCP NewReno is more advantageous than TCP Reno, as the later one suffers from more losses and gives lower throughput between the two.

REFERENCES

[1] KM. Avni Yadav, DR. Sachin Kumar, “Novel Additive Increase And Multiplicative Decrease Algorithm For Congestion Control In Tcp,” International Journal of Pure and Applied Mathematics, vol. 118, pp1059-1066, 2018

[2] Kumhar, Dharamdas, “Performance Analysis of AQM Algorithms in Network Congestion Control,” International Journal of Advanced Research in Computer Science ,vol 8(3), pp.204- 208 ,March – April 2017

[3] Adam Doma´nski, Joanna Doma´nska, Michele Pagano,and Tadeusz Czach´orski, “The Fluid Flow Approximation of the TCP Vegas and Reno Congestion Control Mechanism,” ISCIS, pp. 193–200, 2016.

[4] Chaudhary, Pooja, and Sachin Kumar. “A review of comparative analysis of TCP variants for congestion control in network,” International Journal of Computer Applications, vol. 160(8), pp. 28 - 34 , February 2017 .

[5] Zhenwei Zhu, Yu Qian, Zhou, and Liu Wenjia,Keren. "Dynamic Vegas: Efficient Congestion Control Mechanisms," in International Conference on Information Technology and Computer Science, Springer India, pp. 333-340,2014.

[6] Nadim Parvez, Anirban Mahanti, and Carey Williamson, “An Analytic Throughput Model for TCP NewReno,” IEEE/ACM Transactions on Networking, vol. 18, No.2, April 2010.

[7] Kenji Kurata, Go Hasegawa, Masayuki Murata, "Fairness comparisons between TCP Reno and TCP Vegas for future deployment of TCP Vegas," in Proceedings of INET, pp-1-9, 2000.

[8] C. Samios and M. Vernon, “Modeling the Throughput of TCP Vegas,” In Proc. Of ACM SIGMETRICS, San Diego, USA, June 2003.

[9] Khalid Mohamed, Said Hussein and Asad Abdi, Abubakr El Seddiq ,"Studying the TCP Flow and Congestion Control Mechanisms Impact on Internet Environment." International Journal of Computer Science and Information Security (IJCSIS) ,vol. 16, No. 11, pp. 174- 179, November 2018

[10] Mudassar Ahmad, Majid Hussain, Beenish Abbas, Omar Aldabbas, Uzma Jamil, Rehan Ashraf, Shahla Asadi, "End-to-End Loss Based TCP Congestion Control Mechanism as a Secured Communication Technology for Smart Healthcare Enterprises," IEEE Access 6, vol.6, pp. 11641-11656, 2018

[11] Yesin Sahraoui, Atef Ghanam, Sofiane Zaidi, Salim Bitam, Abdelhamid Mellouk, "Performance evaluation of TCP and UDP based video streaming in vehicular ad-hoc networks." 2018 International Conference on Smart Communications in Network Technologies (SaCoNeT). IEEE, pp.- 67- 72, 2018.

[12] G. Hasegawa, M. Murata, and H. Miyahara, “Fairness and stability of the congestion control mechanism of TCP,” in Proceedings of IEEE INFOCOM 99, pp. 1329–1336, March 1999.