Comparison of TCP I-Vegas with TCP Vegas in Wired-cum-Wireless Network

Nitin Jain & Dr. Neelam Srivastava

Abstract—This paper compares the performance of TCP I-Vegas (where "I" stands for Improved) with conventional TCP Vegas, in wired-cum-wireless network. The performance is compared in terms of various parameters such as throughput, goodput, packet delivery ratio, number of packets received and number of packets dropped using Network Simulator (NS-2). Simulation results show that TCP I-Vegas performs well in all the parameters when compared with conventional TCP Vegas. Actually, TCP Vegas performs well as compared to TCP Reno but when sharing bandwidth with TCP Reno its performance degrades. TCP I-Vegas has been designed keeping in mind that whenever TCP variants like Reno has to share the bandwidth with TCP Vegas then instead of using TCP Vegas, if we may use TCP I-Vegas then the loss which TCP Vegas would have to bear will not be more.

Index Terms—NS-2, TCP, Tahoe, Reno, Sack, Vegas, Wired-cum-wireless networks, Throughput, goodput, .

1 Introduction

TCP Vegas proposed by L.S. Brakmo and L.L. Peterson [1], achieves 37 to 71 percent higher throughput than most used TCP version called TCP Reno [2]. S. Ahn, P.B. Danzig, Z. Liu and L. Yan [3] have evaluated the performance of Vegas and shown that it does achieve higher efficiency than Reno and causes much less packet retransmissions. However, they have also observed that Vegas when competing with other TCP variants like Reno, it does not receive a fair share of bandwidth, i.e., TCP Reno connections receive about 50 percent higher bandwidth. This incompatibility property is analyzed also by J. Mo and J. Walrand [4]. They show that due to the aggressive nature of Reno, when the buffer sizes are large, Vegas loses to Reno that fills up the available buffer space, forcing Vegas to back off. Hence, there is a need to improve the performance of Vegas, which is a conservative algorithm, so that whenever it shares the bandwidth with other TCP variants, the loss which conventional Vegas bears should not be more.

In this paper, we present an improved TCP I-Vegas and compared it with the conventional TCP Vegas in wired-cum-wireless network. The simulation has been done in NS-2 and the result shows that TCP I-Vegas achieves better, throughput, goodput and packet delivery ratio, when compared with conventional TCP Vegas.

The rest of paper is organized as follows: Section 2 presents background of Vegas.

- *Author Nitin Jain is currently pursuing Ph.D program in Electronics Engineering in Uttar Pradesh Technical University, Lucknow, India. E-mail: nitinjain_22@rediffmail.com*
- *Co-Author Dr. Neelam Srivastava is Professor in Electronics Engineering Department at Institute of Engineering & Technology, Lucknow, India. E-mail: neelam.srivastava@ietlucknow.edu*

Section 3 provides issues related with TCP Vegas. Section 4 gives algorithm of TCP I-Vegas which we have made in order to improve the performance of TCP Vegas. Section 5 presents simulation results and discussions. We conclude in Section 6.

2 Background of TCP Vegas

TCP Vegas, a conservative algorithm, when proposed, it differs with TCP Reno in its congestion avoidance scheme. In terms of congestion control schemes, Vegas is delay-based, while Reno and its variants like New Reno [5] are loss-based. Reno and New Reno rely on packet loss detection to detect network congestions, while Vegas uses a sophisticated bandwidth estimation scheme to proactively gauge network congestion.

In particular, since TCP throughput is inversely related to Round Trip Time (RTT), Vegas measure the difference between the expected and the actual throughput. The idea is that the actual throughput should match the expected throughput if there is no congestion along the network path. A lower actual throughput indicates increased delay, and hence congestion, on the network path. Similar to Reno, Vegas has slow start and congestion avoidance modes.

2.1 Slow-Start

During slow-start, Vegas maintains the threshold γ (the value of γ is generally set to 1). As long as diff, when comparing expected_thruput and actual_thruput is less than γ it increases the congestion window by 1 packet every other round trip time, rather than every RTT. Hence, during slow start the Vegas congestion window grows exponentially, though at a slower rate than in TCP Reno. At this point, Vegas needs correction so that it can be made some what aggressive.

When either the congestion window reaches the slow start threshold (ssthresh) or diff is larger than γ, Vegas enters the congestion avoidance. Upon exiting slow-start, Vegas decreases the congestion window by one eighth of its current size in order to ensure that the network does not remain congested.

2.2 Congestion-Avoidance

During congestion-avoidance, Vegas maintains two threshold values α and β (the value of α and β are usually set as 1 and 3 respectively). The adjustment of congestion window (cwnd) is done based on the value of diff given as follows:

$$\text{cwnd} = \begin{cases} \text{cwnd} + 1 & \text{if diff} < \alpha \\ \text{cwnd} - 1 & \text{if diff} > \beta \\ \text{cwnd} & \text{otherwise} \end{cases}$$

Where,
diff = (expected_thruput – actual_thruput).base_RTT

expected_thruput = cwnd/base_RTT, where cwnd is the current congestion window size and base_RTT is the minimum round trip time of that connection.

actual_thruput = cwnd/RTT, where RTT is the actual round trip time

Vegas tries to keep at least α packets but no more than β packets in the queues. Roughly speaking, α and β in Vegas represent respectively the minimum and the maximum number of packets the source can pipe in the network buffers; therefore α and β represent the aggressiveness degree of the TCP Vegas sources. The higher their value, the more Vegas approaches the behavior of Reno. Vegas always attempts to detect and utilize the extra bandwidth whenever it becomes available without congesting the network. This mechanism is fundamentally different from that used by Reno. It always updates its window size to guarantee full utilization of available bandwidth, leading to constant packet losses, whereas Vegas does not cause any oscillation in window size once it converges to an equilibrium point.

In congestion avoidance phase, two changes can be made in the algorithm of Vegas. Firstly, the values of α and β can be increased, because the aim is to make the algorithm of Vegas more aggressive. Secondly, when α < diff < β the size of the congestion window instead of keeping same, can be increased so that it will share the bandwidth more fairly as compared to other variants of TCP.

2.3 Loss Recovery

A packet loss can be detected via time out expiration or via three duplicated acks. In the first case, the ssthresh is set to half of the current congestion window value, the congestion window is set to 2, and Vegas performs again the slow-start. In second case, when Vegas source receives three duplicate acks, it performs Fast Retransmit and Fast Recovery as Reno does. Actually, Vegas develops a more refined fast retransmit mechanism based on a fine-grain clock. After fast retransmit Vegas sets the congestion window to ¾, instead of ½ of the current congestion window and performs again the congestion avoidance algorithm.

3 ISSUES WITH TCP VEGAS

3.1 Fairness

Vegas uses a conservative algorithm to decide how and when to vary its congestion window. Reno, in an effort to fully utilize the bandwidth, continues to increase the window size until a packet loss is detected. Thus, when TCP Vegas and Reno connections shares a bottleneck link, Reno uses up most of the link and router buffer space. Vegas, interpreting this as a sign of congestion, decreases its congestion window, which leads to an unfair sharing of available bandwidth in favor of Reno. This unfairness worsens when router buffer sizes are increased. G. Hasegawa, K. Kurata, M. Murata [6] proposed TCP Vegas+ as a method to tackle Vegas's fairness issue. However, Vegas+ assumes that an increase in the RTT value is always due to the presence of competing traffic and rules out other possibilities like rerouting. We feel that this is not a reasonable assumption. Furthermore, performance of Vegas+ depends on the choice of optimal value for the new parameter Countmax introduced in the protocol, which is an open question. G. Hasegawa, K. Kurata, M. Murata [6] and Raghavendra and Kinicki [7] showed that by using RED routers in place of the tail-drop routers, the fairness between Vegas and Reno can be improved to some degree. But there exists an inevitable trade-off between fairness and throughput, i.e. if the packet dropping probability of RED is set to a large value, the throughput share of Vegas can be improved, but the total throughput is reduced. In [8-9] Feng, Vanichpun and Weigle showed that choosing values of α and β as a function of the buffer capacity of the bottleneck router could improve the fairness condition. However, they do not propose any mechanism to measure this buffer capacity and to set appropriate values for α and β.

3.2 Rerouting

In Vegas, the parameter baseRTT denotes the smallest round-trip delay the connection has encountered and is used to measure the expected throughput. When rerouting occurs in between a connection, the RTT of a connection can change. When the new route has a longer RTT, the Vegas connection is not able to deduce whether the longer RTTs experienced are caused by congestion or route change. Without this knowledge, TCP Vegas assumes that the increase in RTT is due to congestion along the network path and hence decreases the congestion window size [10].

This is exactly opposite of what the connection should be doing. When the propagation delay increases, the bandwidth–delay product (bw*d) increases. The expression (cwnd-bw*d) gives the number of packets in the buffers of the routers. Since the aim of Vegas is to keep the number of packets in the router buffer between α and β, it should increase the congestion window to keep the same number of packets in the buffer when the propagations delay increases. In [10] the authors also proposed a modification to the Vegas to counteract the rerouting problem by assuming any lasting increase in RTT as a sign of rerouting. Besides the fact that this may not be a valid assumption in all cases, several new parameters K, N, L, δ and γ were introduced in this scheme and finding appropriate values for these variables remain an unaddressed problem.

4 TCP I-VEGAS

The algorithm of Vegas required making it little bit aggressive from conservative so that when compared with other TCP variants like Reno it should perform better than the conventional Vegas.

Modifications in Vegas has been confined to the sender side only because of this our I-Vegas with proposed changes is easy to implement.

Modifications does not introduce any further thresholds, generally hard to set, since it is completely adaptive to the status of the network; in this prospect our I-Vegas with proposed changes appears to be more efficient.

I-Vegas, behavior is not much different from that of the original Vegas in presence of other Vegas sources; so it is able to preserve all the nice features of the original Vegas: good throughput and goodput performance and ability in network congestion avoidance.

4.1 Algorithm

Following changes we have made in the algorithm of Vegas in order to make it more aggressive so that its performance get improved as compared to Vegas and it will fairly share the bandwidth when competing with other TCP variants like Reno.

During Slow-Start, Vegas need to change its cwnd more aggressively as Reno does.

In the case of rerouting, it should not decrease its cwnd, rather to increase the thresholds α and β to 3 and 6 respectively.

During RTO and on reception of Three dup ACKs, α and β are again set to 1 and 3 respectively.

During congestion avoidance, when diff lies between α and β, instead of keeping cwnd unchanged, it should change as it is changing when diff < α.

5 SIMULATION RESULTS AND DISCUSSIONS

We have created wired-cum-wireless environment in NS-2 [11] and compared the parameters like throughput, goodput and packet delivery ratio at different packet error probabilities.

5.1 Network Topology

Fig. 1 shows the simulation arrangements for wired-cum-wireless network topology. This topology has been created for comparing the performances of TCP I-Vegas with other TCP variants like Tahoe, New-Reno, Sack and Vegas. It is a simple dumbbell wired-cum-wireless network topology, in which, Node 5 and Node 6 behaves as Router and Base Station respectively. Whereas, Node 0 to Node 4 are sender nodes and Node 7 to Node 11 are receiver nodes, when data flows from wired to wireless part and vice-versa when data flows from wireless to wired part.

5.2 Network Parameters

Table 1 shows the network parameters. The wired network is a fast Ethernet LAN connected to a wireless network (WLAN) through a base station (WAN) which is at bottleneck. Table 2 shows the arrangement of Nodes.

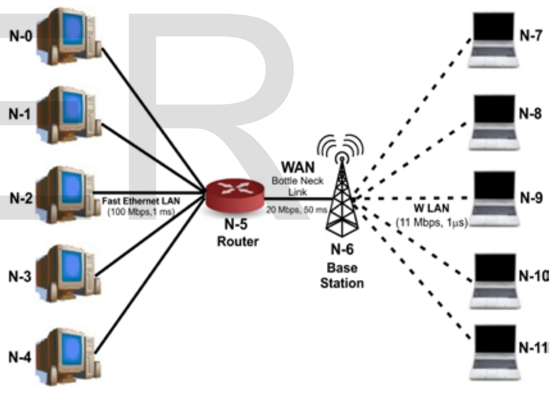


Fig 1. Wired-cum-Wireless Network

TABLE 1
SIMULATION PARAMETERS OF WIRED-CUM-WIRELESS NETWORK

Link	Bandwidth	Delay
Wired Network (Ethernet LAN)	100 Mbps	1 ms
Wireless Network (WLAN)	11 Mbps	1 μs
Bottleneck (WAN)	20 Mbps	50 ms

TABLE 2
SIMULATION ENVIRONMENT OF WIRED-CUM-WIRELESS NETWORK

TCP Variants	Wired to Wireless		Wireless to Wired	
	TCP Sender	TCP Receiver	TCP Sender	Node 0
Tahoe	Node 0	Node 7	Node 7	Node 1
New-Reno	Node 1	Node 8	Node 8	Node 2
Sack	Node 2	Node 9	Node 9	Node 3
Vegas	Node 3	Node 10	Node 10	Node 4
I-Vegas	Node 4	Node 11	Node 11	Node 0

5.3 Results

5.3.1 Data Flows from Wired to Wireless Side of the Network

Fig 1.2 (a), (b) & (c) shows the throughput comparison curves with 1%, 5% and 10% error respectively, Fig 1.3 (a), (b) & (c) shows the congestion window comparison curves with 1%, 5% and 10% error respectively. Fig 1.4 & 1.5 shows av. throughput & av. goodput comparison curves respectively. Fig 1.6, 1.7 & 1.8 shows packet generated, received and dropped respectively and Fig 1.9 shows the packet delivery ratio comparison curves, when data flows from wired to wireless part of the network.

5.3.2 Data Flows from Wireless to Wired Side of the Network

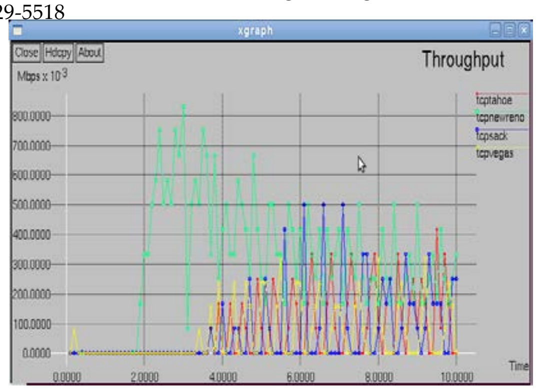
Fig 1.10 (a), (b) & (c) shows the throughput comparison curves with 1%, 5% and 10% error respectively, Fig 1.11 (a), (b) & (c) shows the congestion window comparison curves with 1%, 5% and 10% error respectively. Fig 1.12 & 1.13 shows av. throughput & av. goodput comparison curves respectively. Fig 1.14, 1.15 & 1.16 shows packet generated, received and dropped respectively and Fig 1.17 shows the packet delivery ratio comparison curves, when data flows from wireless to wired part of the network.

5.4 Discussion of Results

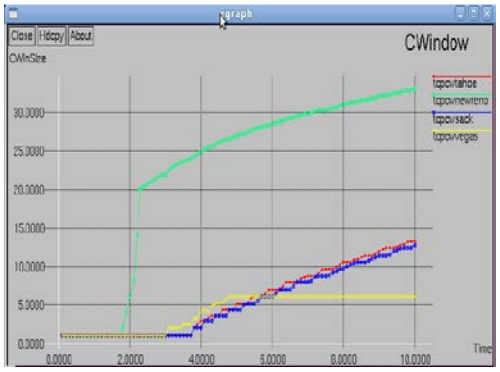
In Fig. 1.1 I-Vegas is sharing the bottleneck link with Tahoe, New-Reno, Sack and Vegas. Fig 1.2 to 1.9 and Fig. 1.10 to 1.17 shows the comparison curves of TCP I-Vegas with other TCP Variants like Tahoe, New-Reno and Sack, when data flows from wired to wireless part of the Network and from wireless to wired part of the network, respectively. In Fig. 1.2, 1.3, 1.10 & 1.11, conventional TCP Vegas curve is not considered as the difference in the performance of TCP I-Vegas and conventional TCP Vegas is not so much that it can visible in those curves simultaneously. However, in rest of the figures, the differences in their performances are clearly visible, whether the data is send from wired to wireless part of the network (Fig. 1.4 to 1.9) or from wireless to wired part of the network (Fig. 1.12 to 1.17). The performance of TCP I-Vegas as compared to conventional TCP Vegas is better but it doesn’t outperform with other TCP variant like Tahoe, New-Reno and Sack. This is only drawback of the congestion control algorithm of TCP I-Vegas. In terms of congestion window, whether the data is send from wired to wireless (Fig. 1.3) or from wireless to wired (Fig. 1.11) part of the network, the TCP I-Vegas shows some distortions, which is because of its aggressive nature. This again supports our algorithm which we have made for TCP I-Vegas. From the topology, it is also clear that as the number of connection increases, the performance of conventional Vegas as compared to other TCP variants decreases. Since the idea behind development of I-Vegas is to improve the performance of conventional Vegas when sharing the bottleneck link with other TCP variants so that the losses which conventional Vegas has to bear should not be more.

6 CONCLUSION

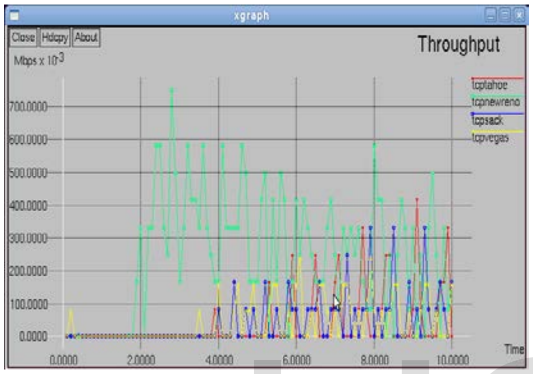
In this paper, we proposed a modified algorithm of Vegas and named it as I-Vegas, where “I” stands for “improved” and compared its performance with conventional Vegas in wired-cum-wireless network. We have also shown that making the algorithm of Vegas from conservative to some what aggressive, the performance of I-Vegas becomes much better than conventional Vegas. Simulation results proved that performance of I-Vegas in terms of av. throughput, av. goodput, packet delivery ratio, number of packets received and congestion window behavior becomes better than Vegas. Though, number of packets dropped has also becomes more in I-Vegas but it provided strength to our research that our proposed algorithm is correct because the aim is to make the Vegas aggressive than conservative.



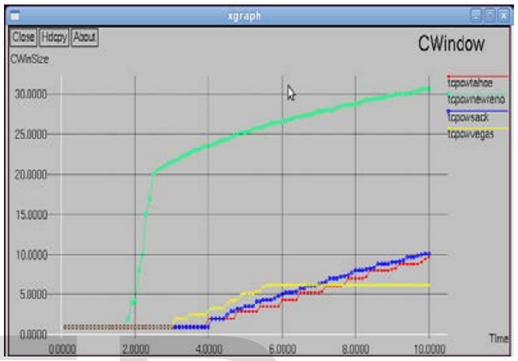
(a) with 1% Error



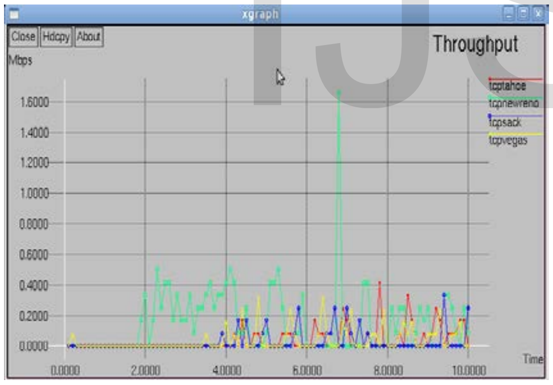
(a) with 1% Error



(b) with 5% Error

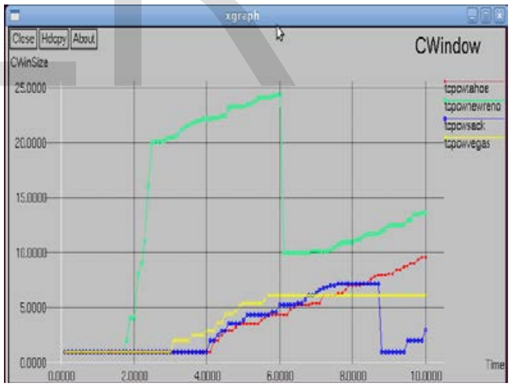


(b) with 5% Error



(c) with 10% Error

Fig. 1.2 Throughput Comparison Curves
(I-Vegas) (Wired to Wireless)



(c) with 10% Error

Fig. 1.3 CWND Comparison Curves
(I-Vegas) (Wired to Wireless)

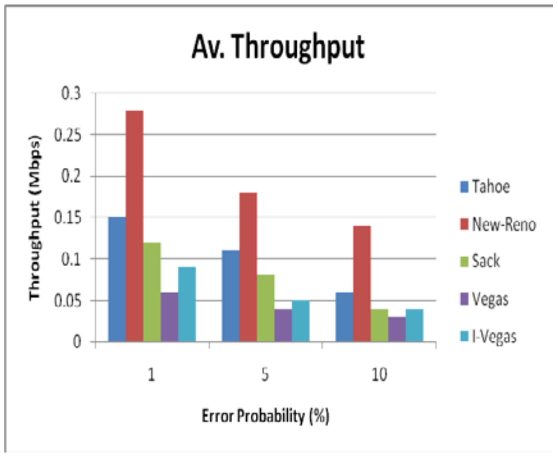


Fig. 1.4 Av. Throughput Comparison Curves (I-Vegas) (Wired to Wireless)

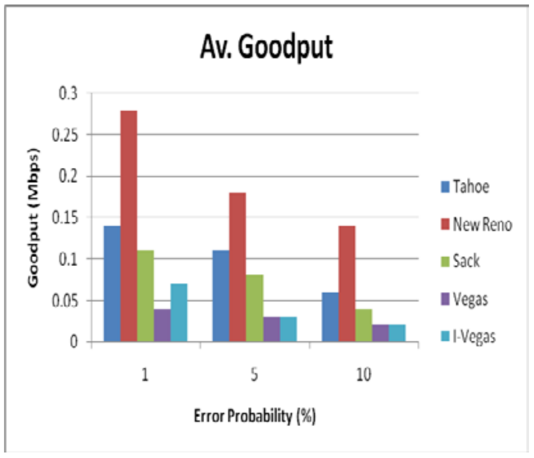


Fig. 1.5 Av. Goodput Comparison Curves (I-Vegas) (Wired to Wireless)

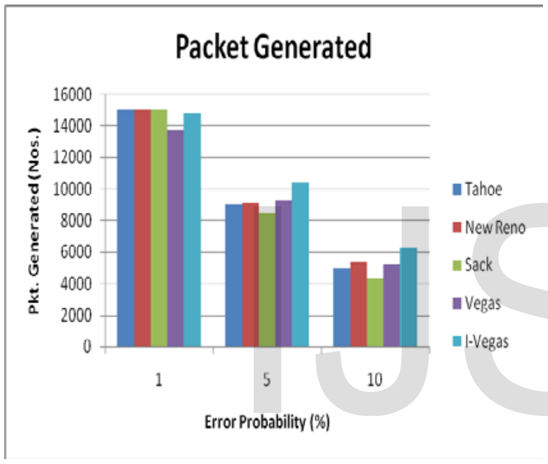


Fig. 1.6 Pkt. Generated Comparison Curves (I-Vegas) (Wired to Wireless)

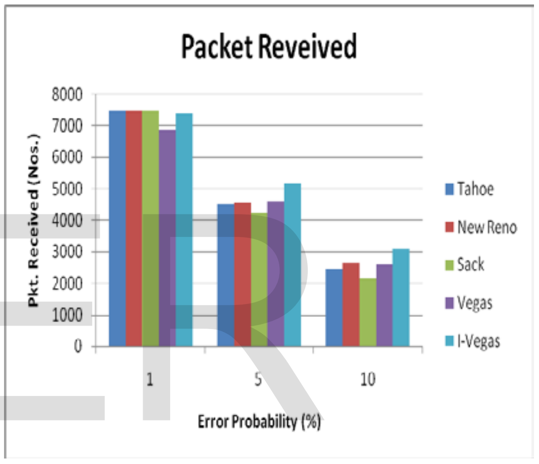


Fig. 1.7 Pkt. Received Comparison Curves (I-Vegas) (Wired to Wireless)

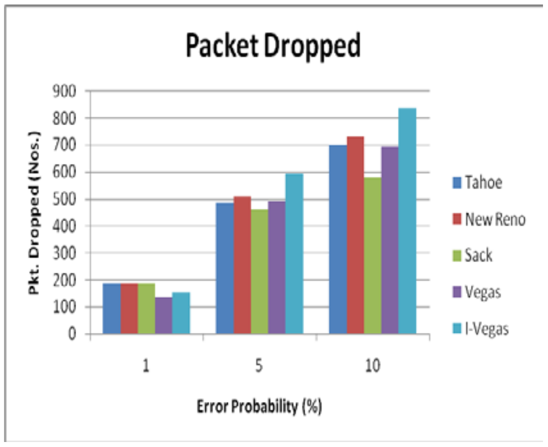


Fig. 1.8 Pkt. Dropped Comparison Curve (I-Vegas) (Wired to Wireless)

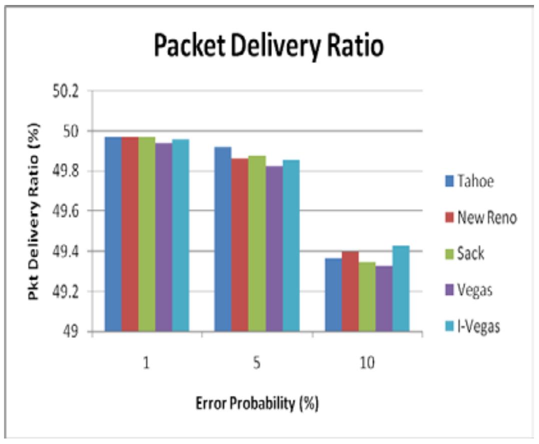
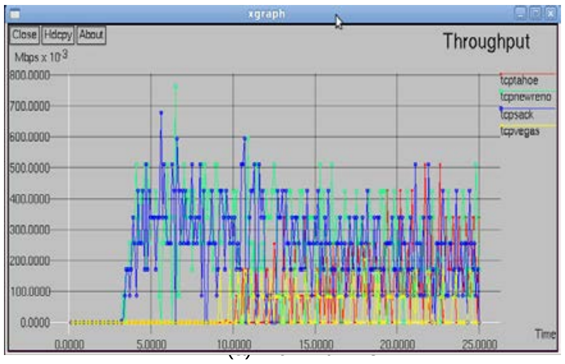
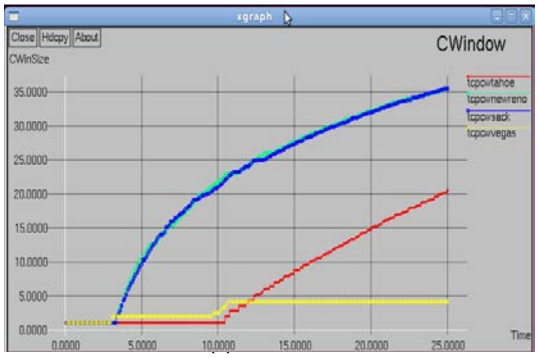
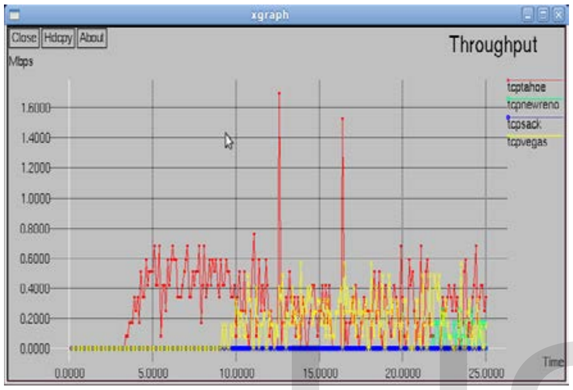
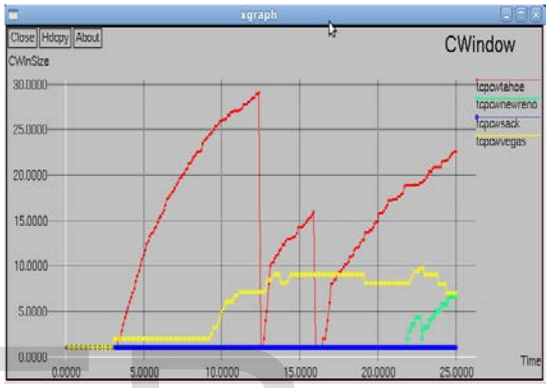


Fig.1.9 PDR Comparison Curves (I-Vegas) (Wired to Wireless)



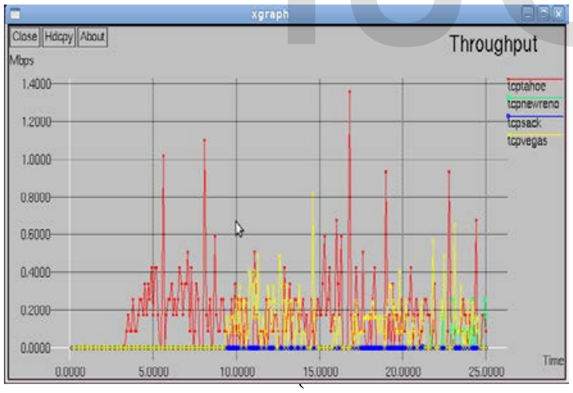


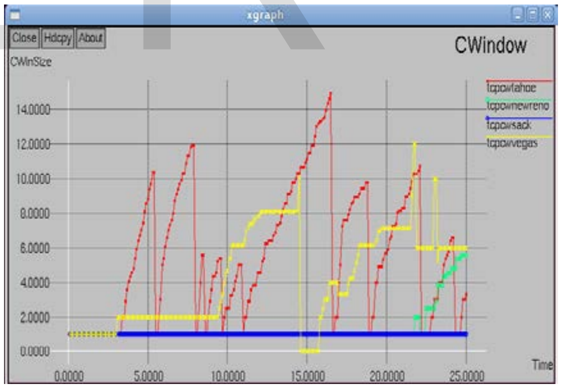




(b) with 5% Error

(b) with 5% Error





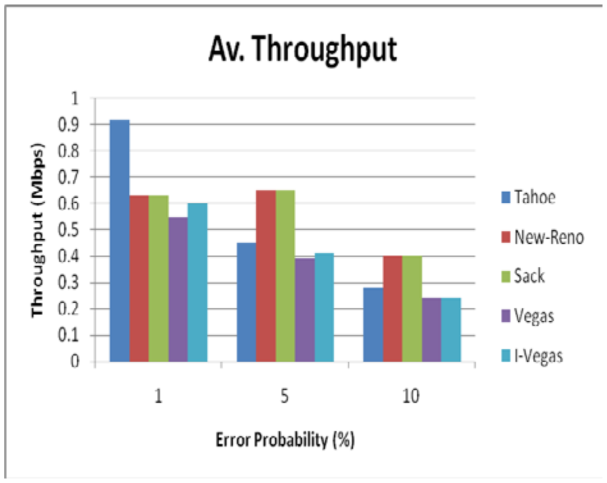
(c) with 10% Error

(c) with 10% Error

Fig. 1.10 Throughput Comparison Curves
(I-Vegas) (Wireless to Wired)

Fig. 1.11 CWND Comparison Curves
(I-Vegas) (Wireless to Wired)

Fig.1.12 Av. Throughput Comparison Curves



(I-
Vegas) (Wireless to Wired)

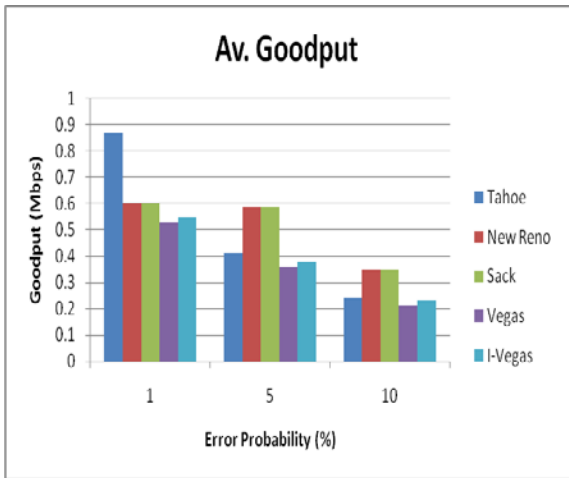


Fig. 1.13 Av. Goodput Comparison Curves
(I-Vegas) (Wireless to Wired)

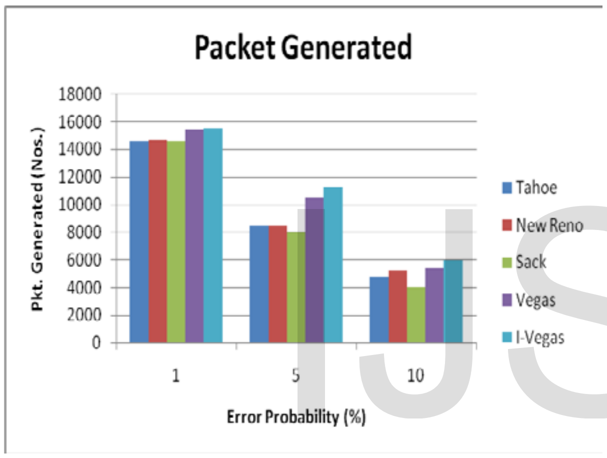


Fig.
1.14 Pkt. Generated Comparison Curves
(I-Vegas) (Wireless to Wired)

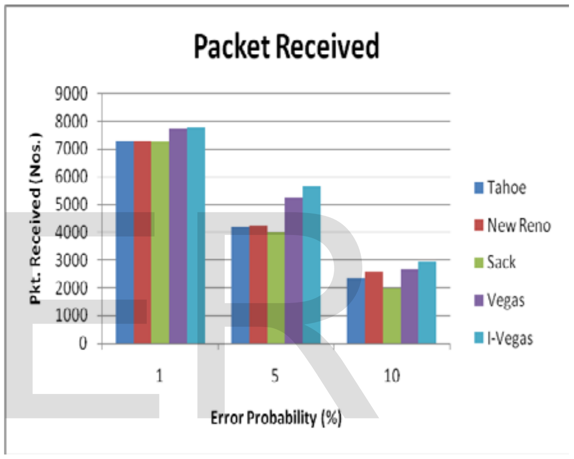


Fig. 1.15 Pkt. Received Comparison Curves
(I-Vegas) (Wireless to Wired)

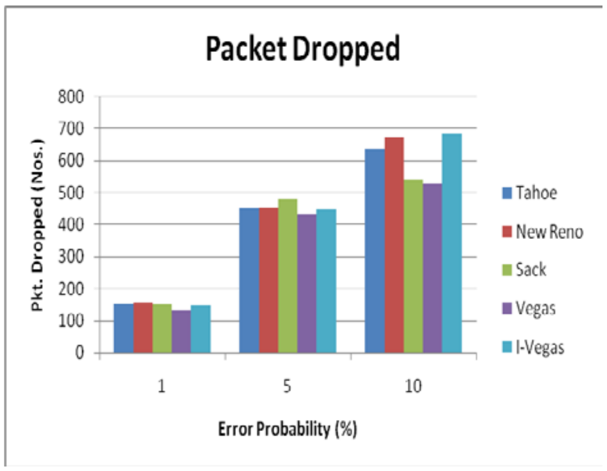


Fig. 1.16 Pkt. Dropped Comparison Curves
(I-Vegas) (Wireless to Wired)

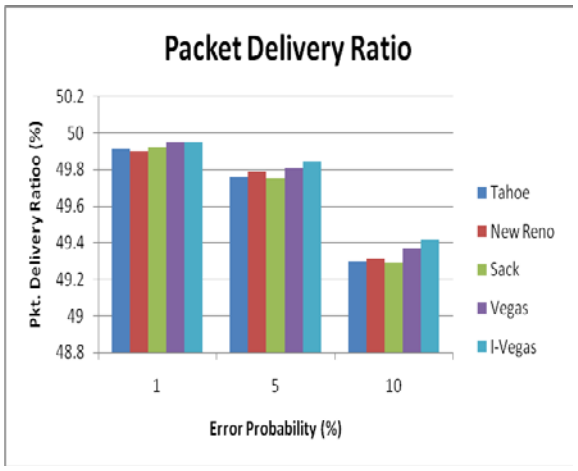


Fig. 1.17 PDR Comparison Curves
(I-Vegas) (Wireless to Wired)

REFERENCES

[1] L. S. Brakmo, L. L. Peterson, TCP Vegas: end-to-end congestion avoidance on a global Internet, IEEE Journal on Selected Areas in Communications,Vol.13, No.8, October 1995.J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[2] R. W. Stevens, TCP/IP Illustrated, Vol. I, The Protocols, Addison-Wesley, U.S.A., 1994.
[3] S. Ahn, P.B. Danzig, Z. Liu, and L. Yan, Evaluation of TCP Vegas: Emulation and Experiment. IEEE Transactions on Communications, 25(4):185-95, Oct 1995.
[4] J. Mo and J. Walrand, Fair End-to-end Window-based Congestion Control, SPIE '98 International Symposium on Voice, Video, and Data Communications, Nov. 1998.
[5] S. Floyd, T. Henderson, The NewReno modification to TCP's fast recovery algorithm, RFC 2582 April 1999.
[6] G. Hasegawa, K. Kurata, M. Murata, Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the internet, Proceedings of the IEEE International Conference on Network Protocols (ICNP 2000) November 2000.
[7] A.M. Raghavendra, R.R. Kinicki, A simulation performance study of TCP Vegas and random early detection, Proceedings of IPCCC'99, February 1999 pp. 169–176.
[8] E. Weigle, W. Feng, A case for TCP Vegas in high-performance computational grids, Proceedings of Ninth International Symposium on High Performance Distributed Computing August 2001.
[9] W. Feng, S. Vanichpun, Enabling compatibility between TCP Reno and TCP Vegas, IEEE Symposium on Applications and the Internet (SAINT 2003) January 2003.
[10] R.J. La, J. Walrand, V. Anantharam, Issues in TCP Vegas, July 1998.
[11] The Network Simulator - NS-2. URL:
http://www.isi.edu/nsnam/ns/index.html.