TCP I-Vegas in Mobile-IP Network

Nitin Jain
Electronics & Communication Engineering
BBDESGI
Lucknow, India

Dr. Neelam Srivastava
Electronics Engineering
IET
Lucknow, India

***Abstract*—Mobile Internet Protocol (Mobile-IP or MIP) provides hosts with the ability to change their point of attachment to the network without compromising their ability to communicate. However, when TCP Vegas is used over a MIP network, its performance degrades because it may respond to a handoff by invoking its congestion control algorithm. TCP Vegas is sensitive to the change of Round-Trip Time (RTT) and it may recognize the increased RTT as a result of network congestion. This is because TCP Vegas could not differentiate whether the increased RTT is due to route change or network congestion. This paper presents a new and improved version of conventional TCP Vegas, which we named as TCP I-Vegas (where "I", stands for Improved). Vegas performs well when compared to Reno but when sharing bandwidth with Reno its performance degrades. I-Vegas has been designed keeping in mind that whenever TCP variants like Reno has to share the bandwidth with Vegas then instead of using Vegas, if we use I-Vegas then the loss which Vegas would have to bear will not be more. We compared the performance of I-Vegas with Vegas in MIP environment using Network Simulator (NS-2). Simulation results show that I-Vegas performs better than Vegas in terms of providing better throughput and congestion window behavior.**

Keywords—TCP Vegas; Mobile-IP; NS-2

I. INTRODUCTION

A large number of heterogeneous computer networks interconnected together using TCP/IP protocol suite (Transmission Control Protocol/Internet Protocol) forms Internet. With the fast prevalence of Internet users demand the mobility of hosts, i.e., they expect that the hosts can change their locations continuously without interrupting current communication sessions.

TCP is a reliable, connection-oriented protocol that ensures in-order delivery of a byte stream supplied by an application. It provides reliable service by implementing flow control, error detection, error recovery, in-order delivery, and removing duplicate data. Both the sending and the receiving node must keep state to support reliable delivery, therefore a connection is setup before data are transferred.

MIP provides hosts with the ability to change their point of attachment to the network without compromising their ability in communications. The mobility support provided by MIP is transparent to other protocol layers so as not to affect those applications which do not have mobility features. MIP introduces three new entities required to support the protocol: the Home Agent (HA), the Foreign Agent (FA) and the Mobile Node (MN). The MIP Working Group of the Internet Engineering Task Force (IETF) has compiled a series of Internet Drafts and Request for Comments (RFC) to define MIP for providing an economical solutions which implements mobility support over the existing Internet infrastructure.

There are several problems of using TCP Vegas in a MIP network. Since TCP Vegas is tuned to perform well in traditional wired networks in which most packet losses are due to congestion. However, in a wireless mobile network, packet losses usually occur due to either random loss or handoff. After a handoff, the throughput of TCP Vegas may be decreased due to a longer BaseRTT of the new routing path, which is usually caused by either triangular routing or route optimization.

In this paper, we present a new and improved version of conventional TCP Vegas which we named as TCP I-Vegas (where "I" stands for Improved). I-Vegas proves to be better in terms of throughput and congestion window behavior, when compared with conventional Vegas. Simulation results proved that our proposed new and improved I-Vegas performs better than Vegas in MIP wired-cum-wireless network.

The rest of paper is organized as follows: Section II presents background of TCP Vegas and Mobile-IP networks. Section III provides issues related with TCP Vegas. Section IV gives algorithm of TCP I-Vegas which we have made in order to improve the performance of TCP Vegas. Section V presents simulation results and discussions. We conclude in Section VI.

II. BACKGROUND: MOBILE-IP & TCP VEGAS

A. *Mobile-IP*

In order to achieve the mobility function, the Internet Protocol (IP) has extended to become the Mobile Internet Protocol (Mobile IP or MIP). MIP provides hosts with the ability to change their point of attachment to the network without compromising their ability to communicate. The mobility support provided by MIP is transparent to the other protocol layers so as not to affect the operation of applications which do not have the mobile capability. Among various IP mobility proposals, Mobile IPv4 [1] & [2] is the oldest and probably the most widely known mobility management proposal with IP. MIPv4 introduces three new entities required to support the protocol: the Home Agent (HA), the Foreign Agent (FA) and the Mobile Node (MN). HA and FA are introduced for mobility management. The basic idea is to use an authenticated registration procedure between a MN and a HA in its home network, and via a FA while MN is visiting a foreign network. Each time a mobile host connects to a network at a new location, it will obtain a temporary address, called Care-of Address (COA) from a foreign agent in the local network. Then the mobile host must inform its home agent of the new address by a registration procedure, which begins when the mobile host, possibly with the assistance of the foreign agent, sends a registration request with the COA. When the home agent receives this request, it may typically add the necessary information to its routing table, approve the request, and send a registration reply back to the mobile host. Further information on MIP functionality can be found in [1] & [2].

B. TCP Vegas

TCP Vegas, a conservative algorithm, which is delay based, first proposed by L.S. Brakmo and L.L. Peterson [3] ensures end-to-end integrity of data transfer, while IP performs datagram routing and internetworking functions. It achieves 37 to 71 percent higher throughput than most used TCP version called TCP Reno [4], which is loss based. S. Ahn, P.B. Danzig, Z. Liu and L. Yan [5] 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 [6]. 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 like Reno or New Reno [7], the loss which conventional Vegas bears should not be more.

TCP throughput is inversely related to 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.

C. 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 somewhat 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.

D. 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 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 $\alpha < diff < \beta$ 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.

E. 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.

III. ISSUES WITH TCP VEGAS

A. 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 [8] 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 $Count_{max}$ introduced in the protocol, which is an open question. G. Hasegawa, K. Kurata, M. Murata [6] and Raghavendra and Kinicki [9] 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 [10-11] 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.

B. 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 [12].

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 [12] 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.

IV. 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.

A. 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, we change the *cwnd* of Vegas more aggressively as Reno does.

In the case of rerouting, Vegas should not decrease its *cwnd*, rather to increase the thresholds α and β to 3 and 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, Vegas should change its *cwnd* as it is changing when $diff < \alpha$.

V. SIMULATION RESULTS AND DISCUSSION

We have created wired-cum-wireless MIP environment in NS-2 [13] and compared the parameters like throughput and congestion window behavior at different packet error probabilities.

A. Network Topology

Fig. 1 shows the network topology which is a wired-cum-wireless MIP network. In fig. 1, node 0 and node 1 are W(0) and W(1) wired nodes respectively, node 2 and node 3 are base station nodes behaves like a HA and FA respectively and node 4 behaves like MN that moves between its HA and FA. Table I gives the details. We set up a TCP flow between node 0 to node 4 i.e. between W(0) and MH. As MH moves out from the domain of its HA, into the domain of FA, we observe how packets destined for MH is redirected by its HA to the FA as per MIP protocol definitions.

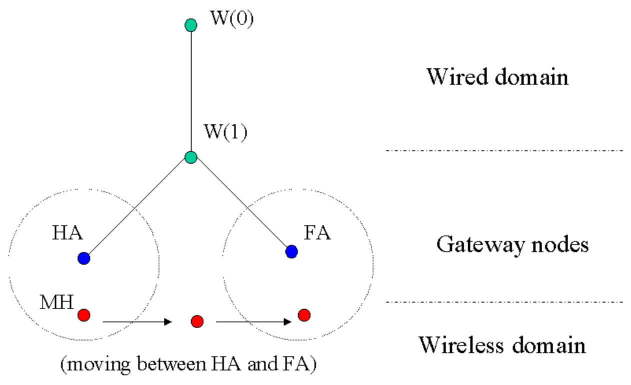

Fig. 1. Wired-cum-Wireless MIP Network

TABLE I. NODE DETAILS

Node	Nature	TCP Connection
Node 0	Wired Node, W(0) (Source Node)	Vegas/I-Vegas
Node 1	Wired Node, W(1)	
Node 2	Base Station Node Home Agent (HA)	
Node 3	Base Station Node Foreign Agent (FA)	
Node 4	Mobile Node (MN) (Sink Node)	

B. Comparison Curves for TCP Vegas and TCP I-Vegas

Fig. 2 to 9 shows the comparison curves in terms of congestion window behavior for TCP Vegas and TCP I-Vegas at different error probabilities. Similarly, fig. 10 to 17 shows the comparison curves in terms of throughput of TCP Vegas and TCP I-Vegas at different error probability.

Figure shows that I-Vegas performs better than Vegas in terms of both congestion window behavior and throughput at different error probabilities.

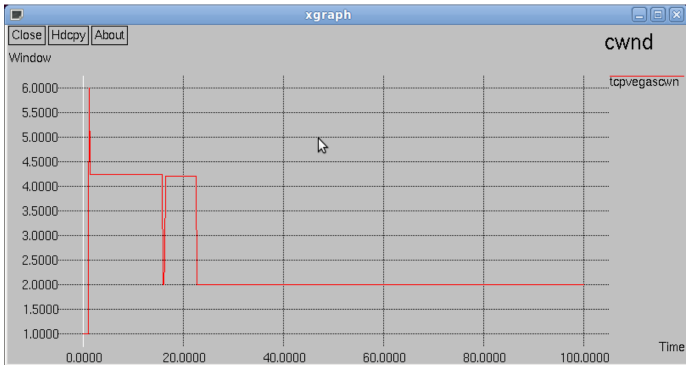

Fig. 2. Congestion Window for TCP Vegas at 0% Error

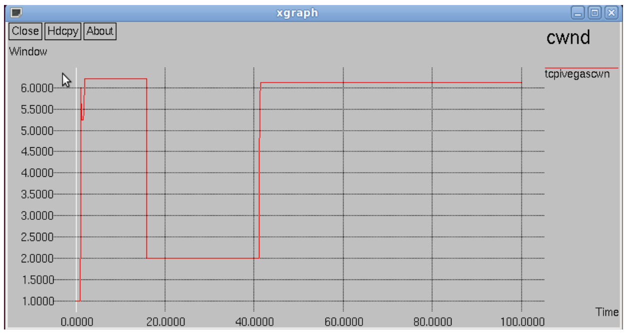

Fig. 3. Congestion Window for TCP I-Vegas at 0% Error

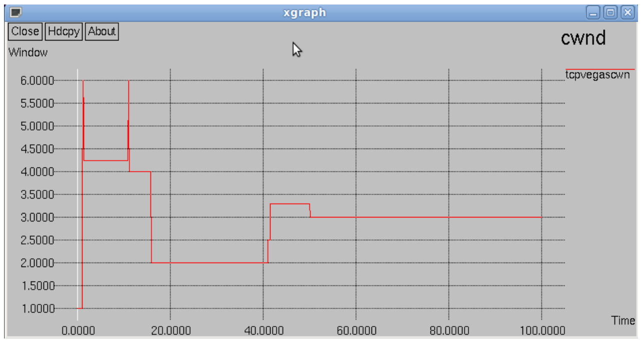

Fig. 4. Congestion Window for TCP Vegas at 1% Error

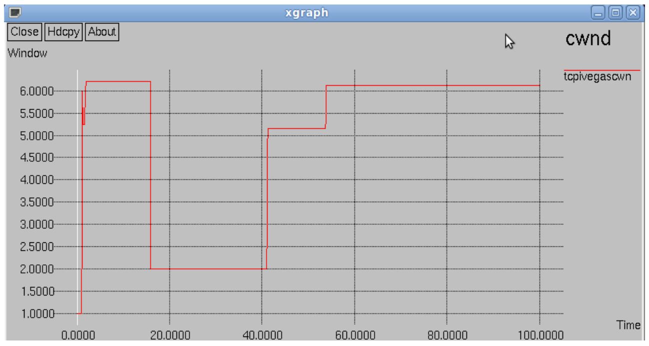

Fig. 5. Congestion Window for TCP I-Vegas at 1% Error

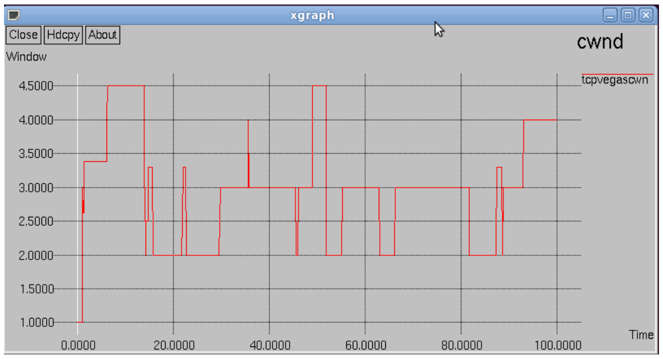

Fig. 6. Congestion Window for TCP Vegas at 5% Error

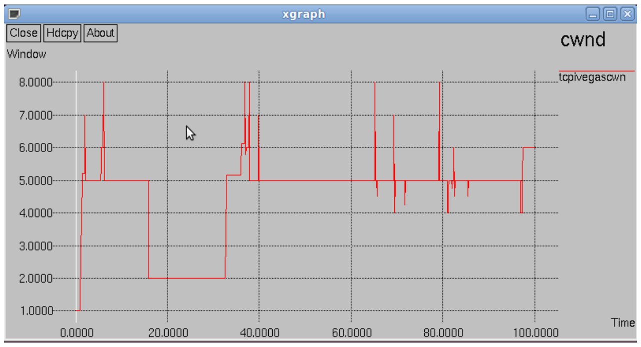

Fig. 7. Congestion Window for TCP I-Vegas at 5% Error

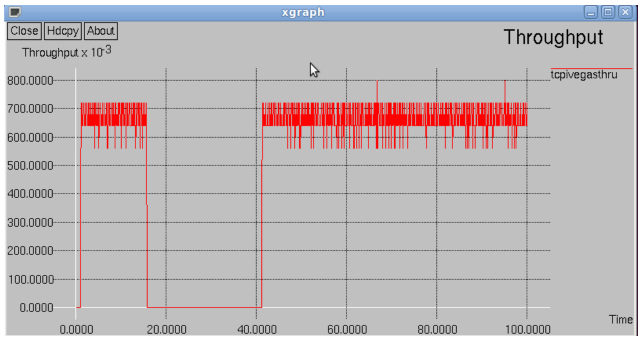

Fig. 11. Throughput of TCP I-Vegas at 0% Error

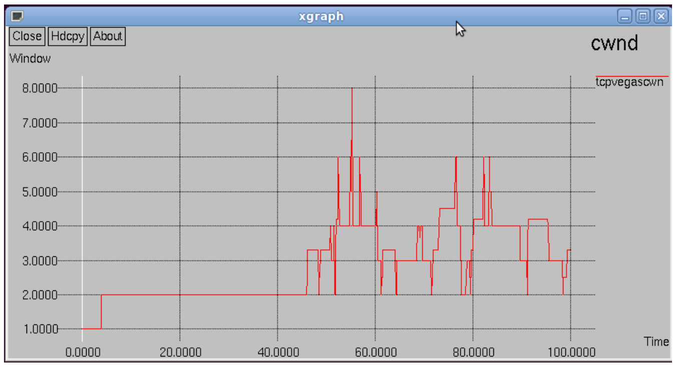

Fig. 8. Congestion Window for TCP Vegas at 10% Error

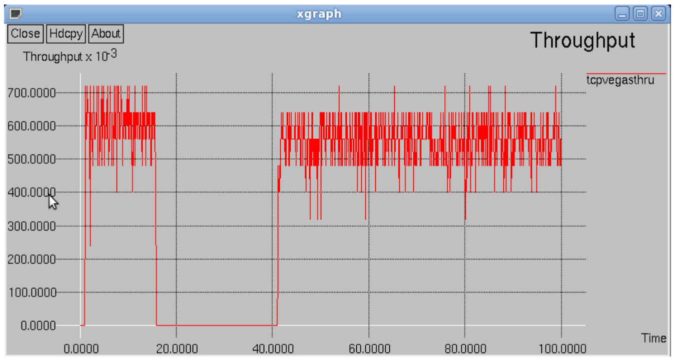

Fig. 12. Throughput of TCP Vegas at 1% Error

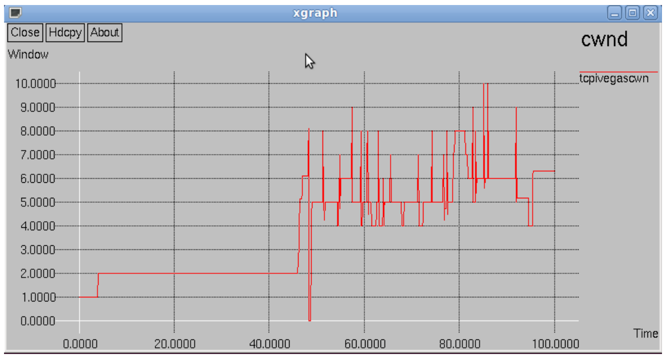

Fig. 9. Congestion Window for TCP I-Vegas at 10% Error

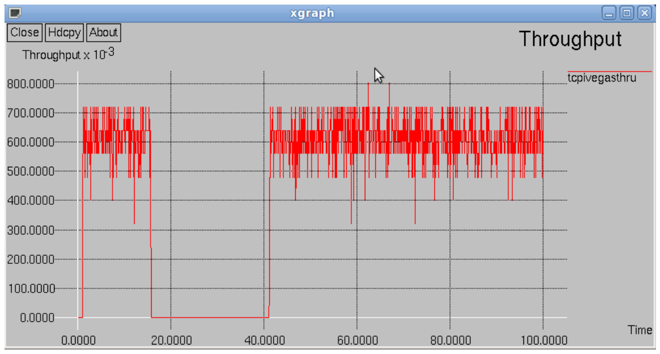

Fig. 13. Throughput of TCP I-Vegas at 1% Error

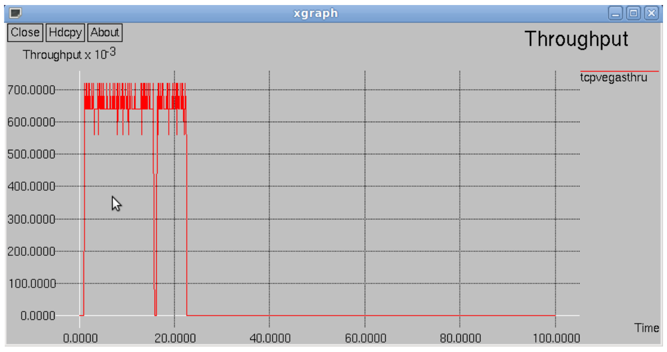

Fig. 10. Throughput of TCP Vegas at 0% Error

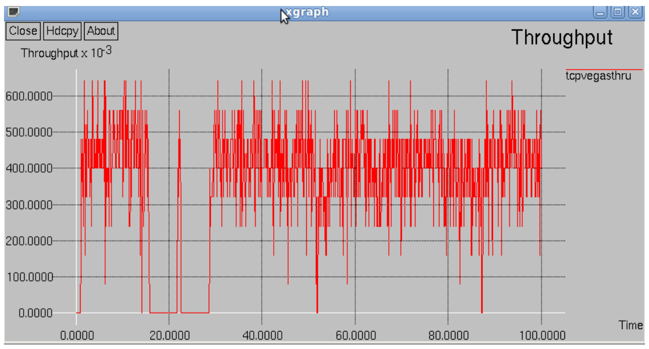

Fig. 14. Throughput of TCP Vegas at 5% Error

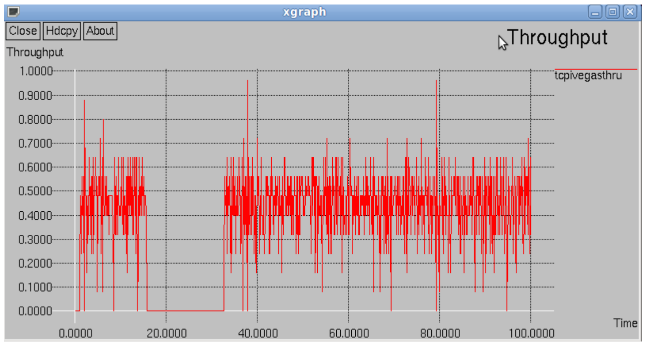

Fig. 15. Throughput of TCP I-Vegas at 5% Error

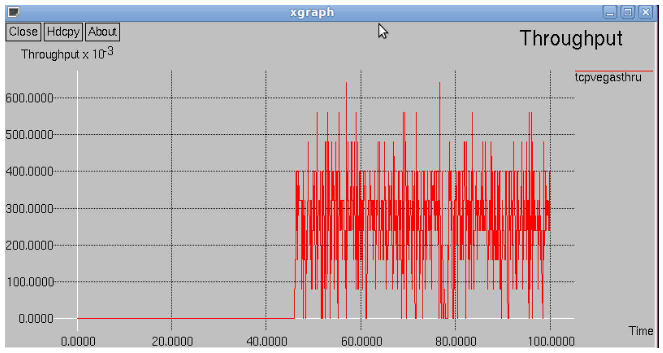

Fig. 16. Throughput of TCP Vegas at 10% Error

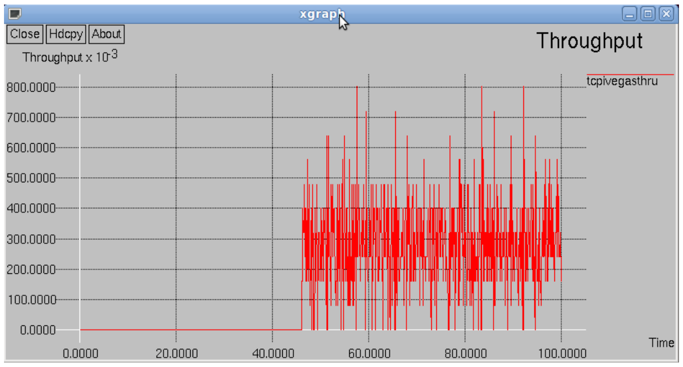

Fig. 17. Throughput of TCP I-Vegas at 10% Error

VI. Conclusion & Future Scope

In this paper, we have proposed a modified algorithm of Vegas and named it as I-Vegas, where "I" stands for "improved". We have also shown that making the algorithm of Vegas from conservative to somewhat 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 and congestion window behavior becomes better than Vegas in MIP network.

Mobile IP is a newly defined protocol which supports mobile users but also is compatible with the current IP. It is still in the process of being standardized, and there are still many items that need to be worked on and enhanced, such as the security issue and the routing issue. We are working on these issues.

References

[1] C. Perkins, 'IP Mobility Support', IETF RFC 3220, January 2002

[2] C. Perkins, 'IP Mobility Support for IPv4', IETF RFC 3344, August 2002

[3] 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.

[4] R. W. Stevens, TCP/IP Illustrated, Vol. I, The Protocols, Addison-Wesley, U.S.A., 1994.

[5] 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.

[6] 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.

[7] S. Floyd, T. Henderson, The NewReno modification to TCP's fast recovery algorithm, RFC 2582 April 1999.

[8] 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.

[9] 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.

[10] 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.

[11] W. Feng, S. Vanichpun, Enabling compatibility between TCP Reno and TCP Vegas, IEEE Symposium on Applications and the Internet (SAINT 2003) January 2003.

[12] R.J. La, J. Walrand, V. Anantharam, Issues in TCP Vegas, July 1998.

[13] The Network Simulator - NS-2. URL: http://www.isi.edu/nsnam/ns/index.html.