A Remedy for Performance Degradation of TCP Vegas in Asymmetric Networks

Cheng P. Fu, *Member, IEEE*, and Soung C. Liew, *Senior Member, IEEE*

***Abstract*—Many previous studies have indicated that TCP Vegas outperforms TCP Reno. This letter shows that in asymmetric networks in which the bottleneck is on the reverse path rather than on the forward path, Vegas underutilizes the available bandwidth on the forward path by a large margin. A solution that makes use of the TCP timestamp option can effectively restore the throughput on the forward path.**

***Index Terms*—Asymmetric networks, congestion control, TCP Vegas.**

I. Introduction

TCP Reno [1] is a widely deployed transport protocol on the Internet. TCP Vegas [2], which employs a fundamentally different congestion control algorithm from that in Reno, was proposed and shown to be able to achieve 37% to 71% throughput improvement over Reno. Comparison of Reno and Vegas is a contentious issue and the study of their relative advantages [3]–[5] and disadvantages [4]–[6] have been investigated extensively and debated heatedly by the research community. This letter adds to the understanding of Vegas by considering its performance in asymmetric networks, in which the transmission bottleneck is in a reverse rather than a forward link.

Network asymmetry is becoming more and more prevalent in the Internet. Satellite technology and access technologies, such as ADSL and HFC, often have vastly different bandwidths in the two directions of the link. So much so that when used to transport TCP packets in the downlink, the ratio of the forward data rate (in data packets per second) to the reverse data rate (in acks per second) is often larger than one. In such a scenario, throughput on the forward path can be limited by ack congestion on the reverse path, as has been demonstrated in the case of TCP Reno [7], [8]. It is therefore interesting to investigate if TCP Vegas exhibits the same degradation in asymmetric networks.

This letter presents results showing that not only does Vegas suffer from performance degradation in asymmetric networks, but the degradation can be so severe that the data transport rate becomes unacceptably low. Fortunately, this degradation can be remedied with a simple mechanism that makes use of the timestamp option of TCP.

Manuscript received April 22, 2002. The associate editor coordinating the review of this letter and approving it for publication was Dr. J. Kim. This work was supported in part by the Area of Excellence in IT of Hong Kong Special Administrative Region, China.

The authors are with the Information Engineering Dept., Chinese University of Hong Kong, Shatin, Hong Kong (e-mail: cpfu@ie.cuhk.edu.hk; soung@ie.cuhk.edu.hk).

Digital Object Identifier 10.1109/LCOMM.2002.805544

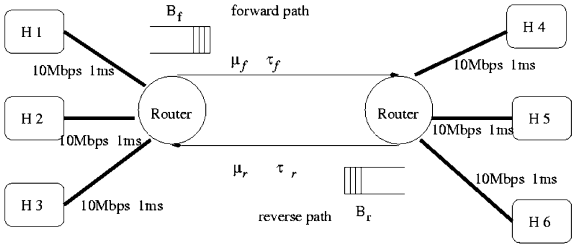


Fig. 1. Asymmetric network model.

II. Simulation Results of Asymmetric Networks

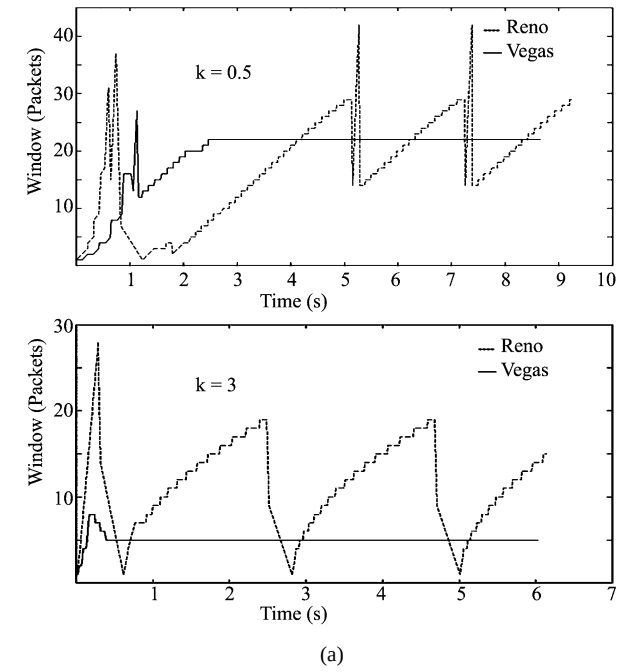
We make use of the network simulator (*ns*) developed at Lawrence Berkeley Laboratory to investigate TCP Vegas in asymmetric networks based on the model in Fig. 1. In the network model, the forward-path capacity and propagation delay are μ_f (in packets/second) and τ_f, respectively, and the reverse-path capacity and propagation delay are μ_r (in acks/second) and τ_r, respectively. In our experiments, the packet size is 1 kbytes and ack size is 40 bytes. Noted the diagram in Fig. 3 describes how to measure the forward data rate in our suggested modified Vegas, which will be addressed in Section III.

Define the asymmetry ratio as $k = \mu_f/\mu_r$. The upper part of Fig. 2(a) plots the evolution of the congestion windows of a Reno and a Vegas connection from one source to one destination, with $\tau_r = \tau_f = 50$ ms and $k = 0.5$— i.e., the network is symmetric and the forward path is the bottleneck. It shows that Vegas' congestion window remains rather constant while Reno's congestion window fluctuates over time. Although during the initial slow-start phase, Vegas performs worse than Reno, the averages of the window size, however, are comparable. The throughputs are therefore comparable.

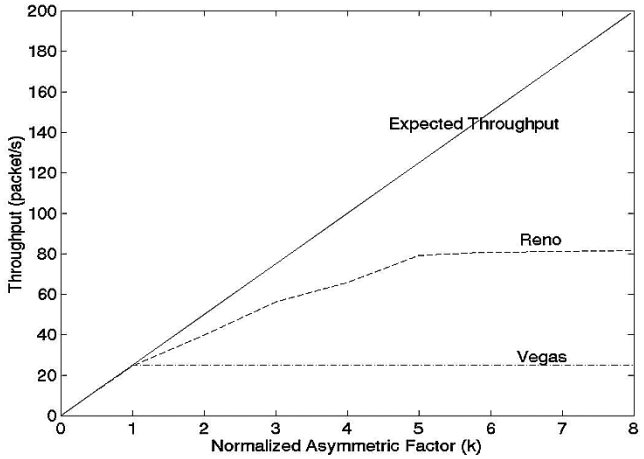
The lower part of Fig. 2(a) plots the results $k = 3$— i.e., the network is asymmetric and the reverse path is the bottleneck. We see that the performance of Vegas is significantly worse than that of Reno in this case.

Table I shows more results for various k values for Reno, Vegas and a modified version of Vegas to remove penalty due to asymmetry. We first address the figures for Reno and Vegas here. B_f and B_r are the buffer size at the forward and reverse paths, respectively. As can be seen, although the performance of both Reno and Vegas degrade as the degree of asymmetry increases, the degradation of Vegas is a lot more severe.

This point is further illustrated with Fig. 2(b). In this experiment, the expected throughput, which is the available bandwidth



(a)



(b)

Fig. 2. (a) Reno's and Vegas' congestion window evolution for $k = 0.5$ and $k = 3$. (b) TCP throughput versus degree of asymmetric network, given $\mu_r = 25$ acks/s network, given $\mu_r = 25$ acks/s.

on the forward path, is varied, while the reverse-path bandwidth is kept constant. It is clearly shown that the bandwidth utilization of the forward path in Vegas is limited to that of the reverse path when $k \geq 1$. Although Reno cannot make full use of the forward bandwidth either, the degradation is less severe.

A. Reason for Performance Degradation of Vegas

A main idea in Vegas is to maintain the packet backlog at the bottleneck link to between α and β by dynamic adjustment of TCP window at the sender. The values of α and β are typically set to 1 and 3, respectively [2]. If the backlog is at least 1 at all time, the bottleneck link will always have packets to transmit and it will therefore be fully utilized. By limiting the backlog to be no more than 3, packet loss due to buffer overflow is avoided.

An implicit assumption in Vegas is that congestion is always encountered on the forward path and that the bottleneck link is on the forward path.

TABLE I
THROUGHPUT AND UTILIZATION OF TCP VEGAS AND RENO
($\mu_f = 1.6$ Mbps, $B_f = 15$ packets $B_r = 10$ acks $\tau_f = \tau_r = 50$ ms)

	RENO / VEGAS / MODIFIED VEGAS			
	Non-asymmetric	Asymmetric Networks		
	k=0.5	k=3	k=5	k=7
Throughput (packet/s)	185/195/194	138/65/190	109/43/192	71.5/26/189
Forward link utilization	.93/.98/.97	.69/.31/.95	.54/.19/.96	.36/.14/.95

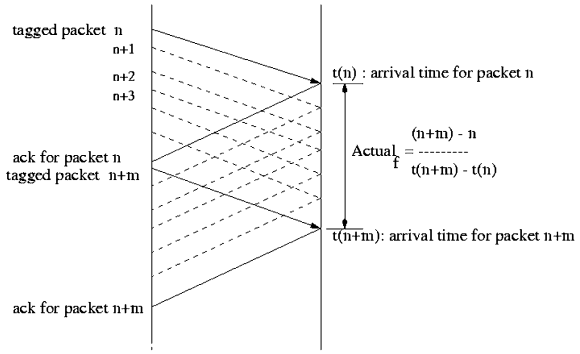


Fig. 3. Measurement of forward data rate.

In asymmetric networks, Vegas misinterprets the accumulation of acks at the reverse bottleneck link as indication of congestion on the forward path and decreases the TCP window accordingly. Meanwhile, the forward buffer can be relatively empty. This proactive congestion control mechanism prevents ack loss rather packet loss, causing unnecessary throttling of data rate.

Unlike Vegas, which adopts proactive congestion control mechanism, Reno adopt a reactive mechanism to aggressively increases its TCP window until data packet loss actually occurs. Although acks on the reverse path may occur before the data packet loss on the forward path, the cumulative property of acks (i.e., the loss of an ack can be compensated by the reception of a subsequent ack) allows the window size to continue increase. Therefore, Reno does not suffer from the same degree of degradation as Vegas does in asymmetric networks.

III. SOLUTIONS

Various techniques have been proposed to improve the performance of Reno in asymmetric networks. These techniques, however, are not effective for handling Vegas' problems.

ACC (Ack Congestion Control) uses a gateway on the reverse link to aid congestion control [8]. It tries to detect impending congestion by tracking the average queue size in the recent past and then informs the receiver to dynamically decrease the frequency of acks so that each ack effectively acknowledges several packets. However, Vegas' queue size only fluctuates between 1 and 3 and it is difficult for ACC to react based on queue size of this narrow range.

AF (Ack Filtering) [8] is a router-based technique that attempts to remove accumulated acks in the reverse buffer to ease up the congestion. Again, given the small backlog fluctuation in Vegas, it is difficult for the router to react unless it is aware of

the fact that the acks are from Vegas connections and not from Reno connections.

Our solution employs an end-to-end method. Before explaining this method, let us review the basic window adjustment mechanism in Vegas. Specifically, each time a measurement of RTT, the round trip time, is made, the following algorithm is performed at the sender:

```
if (Diff*BaseRTT < α)
  /*where Diff = (Expected − Actual)*/
 cwnd = cwnd + 1
  /*increase congestion window size by
 one*/
if (Diff*BaseRTT > β)
 cwnd = cwnd − 1
else cwnd = cwnd
  /*keep congestion window unchanged*/
```

where $cwnd$ is the window size, $BaseRTT$ is minimum of all RTT s measured and $Diff = Expected - Actual$ (the difference between expected and actual data rate). $Expected$ is computed from $cwnd/BaseRTT$ and $Actual$ is computed from $cwnd/RTT$.

The problem with Vegas in asymmetric networks arises because the $Actual$ data rate computed as above is not the forward data rate, but the ack rate. Our solution consists of using the timestamp option in TCP to encode the arrival times of packets at the destination in their acks back to the sender. As illustrated in Fig. 3, $Actual_f$, the actual data arrival rate in the forward path can then be easily computed by the number of packets between two tagged packets divided by the difference of the reception times of the two tagged packets.

We re-conducted the experiments with this mechanism installed. As shown in Table I, with the modified Vegas, the performance degradation has effectively been removed.

IV. Conclusions

Many previous studies have indicated that TCP Vegas outperforms TCP Reno. This letter shows that in asymmetric networks, Vegas algorithm fails to make full use of the available bandwidth on the forward path and points out that the Vegas congestion detection mechanism is the fundamental cause behind such degradations. We have proposed and demonstrated that the TCP timestamp option can be used for accurate measurement of data rate to remove the limitation of Vegas in asymmetric networks.

This study has focused on identifying the key reason behind the performance degradation of Vegas in asymmetric networks and providing a simple solution to it. As such, simple experimental set-ups have been considered. More extensive experiments involving large numbers of concurrent reverse and forward connections, such as those conducted in paper [9], will be worthwhile to validate our observation and solution in future work.

References

[1] V. Jacobson, "Congestion avoidance and control," in *ACM SIGCOMM 88*, Stanford, CA, Aug. 1988, pp. 314–329.

[2] L. Brakmo, S O'Malley, and L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *ACM SIGCOMM'94*, London, U.K., Aug. 1994, pp. 24–35.

[3] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan, "Evaluation with TCP Vegas: Emulation and experiment," in *ACM SIGCOMM 1995*, Cambridge, MA, Aug. 1995, pp. 185–205.

[4] G. Hasegawa, M. Murata, and H. Miyahara, "Fairness and stability of congestion control mechanism of TCP," in *INFOCOM '99*, New York, Mar. 1999, pp. 1329–1336.

[5] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," in *Proc. INFOCOM '99*, vol. 3, New York, Mar. 1999, pp. 1556–1563.

[6] U. Hengartner, J. Bolliger, and T. Gross, "TCP Vegas revisited," *Proc. IEEE INFOCOM'00*, vol. 3, pp. 1546–1555, Mar. 2000.

[7] T. V. Lakshman, U. Madhow, and B. Sutter, "Window-based error recovery and flow control with a slow acknowledgment channel," in *INFOCOM '97*, Kobe, Japan, Apr. 1997, pp. 1199–1209.

[8] H. Balakrishnan, V. N. Padmanabhan, and R. H. Katz, "The effects of TCP/IP performance," in *Proc. INFOCOM'97*, Budapest, Hungary, Sept. 1997, pp. 77–89.

[9] S.H. Low, L. Peterson, and L. Wang, "Understanding Vegas: A duality model," *J. ACM* , vol. 49, no. 2, pp. 207–235, Mar. 2002.