An Aided Congestion Avoidance Mechanism for TCP Vegas*

Cheng-Yuan Ho[1], Chen-Hua Shih[1], Yaw-Chung Chen[1], and Yi-Cheng Chan[2]

[1] Department of Computer Science and Information Engineering, National Chiao Tung University, No. 1001, Ta Hsueh Road, Hsinchu City, 300, Taiwan
{cyho, shihch, ycchen}@csie.nctu.edu.tw

[2] Department of Computer Science and Information Engineering, National Changhua University of Education, No. 1, Jin-De Road, Changhua, Taiwan
ycchan@cc.ncue.edu.tw

Abstract. TCP Vegas detects network congestion in the early stage and successfully prevents periodic packet loss that usually occurs in TCP Reno. It has been demonstrated that TCP Vegas achieves much higher performance than TCP Reno in many aspects. However, TCP Vegas cannot prevent unnecessary throughput degradation when congestion occurs in the backward path, it passes through multiple congested links, or it reroutes through a path with longer round-trip time (RTT). In this paper, we propose an aided congestion avoidance mechanism for TCP Vegas, called Aid-Vegas, which uses the relative one-way delay of each packet along the forward path to distinguish whether congestion occurs or not. Through the results of simulation, we demonstrate that Aid-Vegas can solve the problems of rerouting and backward congestion, enhance the fairness among the competitive connections, and improve the throughput when multiple congested links are encountered.

1 Introduction

With the fast growth of Internet traffic, how to efficiently utilize network resources is essential to a successful congestion control. Transmission Control Protocol (TCP) is a widely used end-to-end transport protocol on the Internet; it has several implementation versions (i.e., Tahoe, Reno, Vegas...) which intend to improve network utilization. Among these TCP versions, Vegas can achieve much higher throughput than that of others [1].

TCP Vegas uses the difference between the expected and actual throughput to estimate the available bandwidth in the network, control the throughput, and avoid congestion. The idea is that when the network is not congested, the actual throughput will be close to the expected throughput. Otherwise, it will be much smaller than expected. TCP Vegas uses the congestion window size and measured RTT to estimate the amount of data in the network pipe, maintains

* This work was sponsored in part by National Science Council under grant no. NSC 93-2752-E-009-006-PAE.

© Springer-Verlag Berlin Heidelberg 2005

extra data with amount between the lower threshold (α) and the upper threshold (β), gauges the congestion level in the network, and updates the congestion window size accordingly. As a result, Vegas is able to detect network congestion in the early stage and successfully prevents periodic packet loss that usually occurs in Reno. Furthermore, Vegas keeps an appropriate amount of extra data in the network to avoid congestion as well as to maintain high throughput. Many studies have demonstrated that Vegas outperforms Reno in the aspects of overall network utilization, stability, fairness, and throughput [1], [2], [3], [4]. However, it suffers some problems that inhere in its congestion avoidance scheme, including those issues of rerouting [3], [5], fairness [4], network asymmetry [5], [6], [7], [8], [9], and path with multiple congested links [10]. All these problems may be obstacles for Vegas to achieve a success.

In this work, we propose an aided congestion avoidance mechanism for TCP Vegas (abbreviated as Aid-Vegas hereafter). By using the relative one-way delay of each packet along the forward path to distinguish whether congestion occurs or not, Aid-Vegas may solve the problems of rerouting and backward congestion, enhance the fairness among the competitive connections, and improve the throughput when passing through multiple congested links. We demonstrate the effectiveness of Aid-Vegas based on the results of simulation.

The rest of this paper is organized as follows. Section 2 describes Vegas and its problems. Section 3 outlines prior related work. Section 4 discusses the Aid-Vegas. The simulation results are presented in Section 5. Finally, Section 6 makes some concluding remarks.

2 TCP Vegas and Its Problems

In this section, we review the congestion avoidance mechanism of TCP Vegas and describe its problems in detail. The detailed description of Vegas can be found in [1].

2.1 Congestion Avoidance Mechanism of TCP Vegas

Different from Tahoe and Reno, which detect network congestion based on packet losses, Vegas estimates *a proper amount of extra data*, called Δ for short, to be kept in the network pipe and controls the congestion window size accordingly during the congestion avoidance phase. It records the RTT and sets BaseRTT to the minimum of ever measured RTTs. The Δ is between two thresholds α and β, as shown in the following:

$$\alpha \leq (Expected - Actual) \times BaseRTT \leq \beta, \tag{1}$$

where *Expected* throughput is the current congestion window size divided by BaseRTT, and *Actual* throughput is the current congestion window size divided by the newly measured RTT. Both throughput and congestion window size are kept constant when Δ is between α and β. If Δ is greater than β, it is taken as a sign for incipient congestion, thus the congestion window size will be reduced. On the other hand, if Δ is smaller than α, the connection may be underutilizing the available bandwidth. Hence, the congestion window size will be increased.

2.2 Problems in Vegas

There are several problems in Vegas that may have a serious impact on the performance during the congestion avoidance phase. We summarize these problems as follows.

Network Asymmetry: By adjusting source congestion window size, Vegas keeps an estimated extra data on the bottleneck to avoid congestion as well as to maintain high throughput. However, a roughly measured RTT may lead to an improper adjustment of congestion window size. If the network congestion occurs in the direction of ACK packets (backward path), it may underestimate the actual rate and cause an unnecessary decreasing of the congestion window size. Ideally, congestion in the backward path should not affect the network throughput in the forward path, which is the data transfer direction. Obviously, the control mechanism must be able to distinguish the direction of congestion and adjust the congestion window size only if necessary.

Rerouting: Vegas estimates the BaseRTT and RTT to compute the expected and actual throughput respectively and then adjusts its window size accordingly. This idea works well in usual situations. However, rerouting may cause a change of the fixed delay, which is the sum of propagation delay and packet processing time along the round-trip path, and result in substantial throughput degradation. When the route of a connection is changed, if the new route features shorter fixed delay, it will not cause any serious problem for Vegas because most likely some packets will experience shorter RTT, and BaseRTT will be updated eventually. On the other hand, if the new route for the connection has a longer fixed delay, it would be unable to tell whether the increased RTT is due to network congestion or route change. The source host may misinterpret the increased RTT as a signal of congestion in the network and decrease its window size. This is just the opposite of what the source should do.

Unfairness: Different from TCP Reno, Vegas is not biased against the connections with longer RTT [3], [4]. However, there is still unfairness which comes with the nature of Vegas. Since Vegas attempts to maintain Δ between two thresholds α and β by varying its congestion window size, but the range between α and β features uncertainty that affects the achievable throughput of connections. Furthermore, Vegas may keep different extra data in the bottleneck for connections with the same round-trip path. As a result, it prohibits better fairness among the competing connections.

Multiple Congested Links: In the paper about Vegas [1], it is assumed that only one bottleneck in the connection so the extra data is just queued in one router. However, when Vegas passes through multiple congested links, it could not tell whether the packets are queued in a single or multiple routers because

Vegas uses the RTT to estimate the backlog in the path. As a result, it tends to decrease congestion window size and hence degrade the throughput of multi-hop connections due to reducing the sending rate unnecessarily [10].

3 Related Works

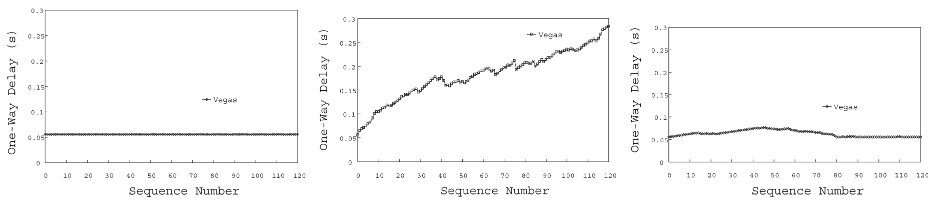
Congestion control for TCP is a very active research area; solutions to TCP congestion control address the problem either at the intermediate routers in the network [5], [11] or at the endpoints of the connection [6], [7], [8], [9], [12].

Router-based support for Vegas congestion avoidance mechanism can be provided as RoVegas [5], a solution which uses a normal TCP packet (data or ACK) with AQT (accumulate queuing time) option in its IP header as a probing packet to collect the queuing time along the path. As an alternative to packet dropping, an ECN (Explicit Congestion Notification) [11] bit can be set in the packet header for prompting the source to slow down. However, current TCP and router implementations do not support these two methods.

Several end-to-end congestion control approaches have been proposed to improve TCP performance for asymmetric networks. These approaches obtain either the forward trip time [7] or the actual flow rate on the forward path [8] depending on TCP timestamps option. Although solutions such as ACC (ack congestion control), AF (ack filtering), SA (sender adaptation), and AR (ack reconstruction) have improved the Reno's performance under asymmetric networks [12], these are not effective for handling asymmetry problems of Vegas [8]. By using the relative delay estimation along the forward path, TCP Santa Cruz [9] is able to identify the direction of congestion. However, it is not for rate-based Vegas. Enhanced Vegas [6] is a mechanism that works under asymmetric networks and uses TCP timestamps option to estimate queuing delay on the forward and backward path separately without clock synchronization. Nevertheless, clock skew issue such as the convergence speed of the clock ratio is still a problem of Enhanced Vegas.

4 Aid-Vegas

Vegas estimates a proper amount of extra data to be kept in the network pipe and controls the congestion window size accordingly. It works well during the congestion avoidance phase when no other competing sources exist. However, the aforementioned problems such as rerouting, unfairness, and multiple congested links may be encountered. Also it leads to unnecessary throughout degradation when the congestion occurs on the backward path. In other words, Vegas does a good job on its increasing part, but it may mistakenly decrease the congestion window easily when there are some variations in the network. In this work, we propose Aid-Vegas, which preserves the advantages of TCP Vegas, to deal with these problems. The detail of the proposed mechanism is explained as follows.



(a) Steady one-way delays (b) Increase trend (c) Short-term intervals

Fig. 1. The relative one-way delay of packets for Vegas

4.1 Mechanism Description

Suppose that Src (source) transmits packets to Dest (destination), Src timestamps each packet i with a timestamp t_i prior to its transmission, and a_i is the arrival time of the i^{th} packet at Dest. Dest computes the *relative one-way delay* of each packet as $D_i = a_i - t_i$. As will become clear later, the measurement methodology does not need synchronized clocks, because we are only interested in the relative magnitude of *one-way delays.* If a Vegas source keeps a constant throughput and a stable Δ between α and β during the congestion avoidance phase, so the relative one-way delay of each packet is steady as shown in Fig. 1 (a). We could see two situations when other sources want to compete the resource with it [13], [14]. One is that while the total flow rates are larger than the link capacity consistently, the relative one-way delays of Vegas would experience an increasing trend as shown in Fig. 1 (b). The other is the short-term relative one-way delays are increasing, as shown in Fig. 1 (c) when transient traffic of other sources passes by. As a result, Aid-Vegas utilizes one-way delays relationship to decide whether congestion happens in the forward path. The detailed mechanism is described as follows.

In the destination side, after computing D_i for each packet, Dest tells the result of D_i comparing with D_{i-1} to Src by using two reserved bits in TCP header, called ROD bits. In case $|D_i - D_{i-1}| \leq \tau$, where τ is a predefined time interval, or it is the first packet of a connection (i.e., $i = 1$), the ROD value is 00_{binary}. If $(D_i - D_{i-1}) > \tau$, the ROD value is 10_{binary}. On the other hand, the ROD value is 01_{binary} when $(D_{i-1} - D_i) > \tau$. In the source host, we set the trend D_t, whose default value is 0 in every RTT, to represent a congestion in the forward path. Whenever Src receives an ACK, it sums ROD value to D_t, and the values for 00_{binary}, 01_{binary} and 10_{binary} of ROD are set to 0, 1, and -1 respectively. The detailed description of Src and Dest's behaviors with variations of one-way delays is shown in Table 1.

Table 1. Src and Dest's behaviors with variations of one-way delays

variation	Dest	Src
$\lvert D_i - D_{i-1}\rvert \leq \tau$ or $i = 1$	ROD $= 00_{binary}$	$D_t = D_t + 0$
$(D_{i-1} - D_i) > \tau$	ROD $= 01_{binary}$	$D_t = D_t + 1$
$(D_i - D_{i-1}) > \tau$	ROD $= 10_{binary}$	$D_t = D_t - 1$

Table 2. 9 situations and congestion window size adjustment

	9 situations								
	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
Src	+	+	+	$\diamond$	$\diamond$	$\diamond$	−	−	−
Dest	+	$\diamond$	−	+	$\diamond$	−	+	$\diamond$	−
	Adjustment								
CWD	+1	+1	+1	+1	±0	±0	+1	±0	–1

When this calculation is performed once per RTT, Src adjusts the congestion window size according to both values of Δ and D_t. If D_t is positive, it indicates that there is no congestion in the network, so Src increases the congestion window. Congestion may occur in the forward path and the throughput may be reduced when D_t is negative. The flow rates in the network are balanced while D_t equals 0, so Src doesn't adjust anything. Accordingly, the combinations of Src and Dest's notifications for adjusting the congestion window size results in 9 situations. Table 2 shows these 9 situations and how Src adjusts the congestion window size, where +, $\diamond$, and - represent $\Delta < \alpha$, $\alpha \leq \Delta \leq \beta$, and $\beta < \Delta$ in Src column, and $D_t > 0$, $D_t = 0$, and $D_t < 0$ in Dest column respectively, and the signification of CWD is the congestion window size. We reserve the increasing part (i.e., (1)~(3) notifications in Table 2) of Vegas and modifies both the steady and decreasing parts (i.e., the others in Table 2).

Since the idea is simple, we omit the pseudo codes in this section. The proposed scheme can improve the performance of TCP Vegas according to the simulation results in the following section.

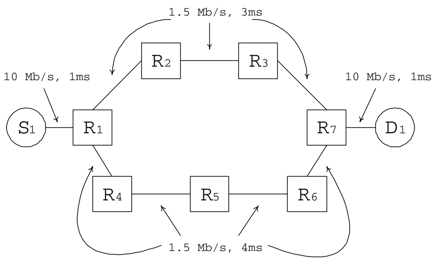
5 Simulation Results

In this section, we compare the performance of Aid-Vegas with Vegas by using the network simulator *ns2* [16], version 2.26. We show the simulation results for backward congestion, rerouting, fairness among the competing connections, and multiple congested links. The FIFO service discipline is assumed. Several VBR sources construct both forward and backward traffic, these are distributed ON-OFF sources with the Pareto model. During ON periods, some VBR sources generate backward traffic and send data at 3.0 Mb/s, while the others send data at 0.8 Mb/s or 1.0 Mb/s. Unless otherwise stated, the size of each FIFO queue used in routers is 50 packets, the size of data packet is 1 Kbytes, and the size of ACK is 40 bytes for both Vegas and Aid-Vegas. To ease the comparison, we assume that the sources are back logged.

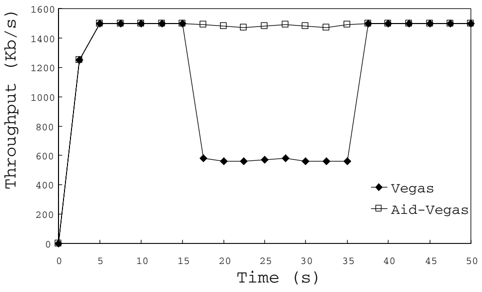
5.1 Rerouting

From Section 2, we could see that there is no serious problem for Vegas when it reroutes with a shorter RTT, so does Aid-Vegas because it is based on Vegas.

However, if packets are rerouted with a longer RTT, Vegas may suffer throughput degradation, but not for Aid-Vegas. The reason is that Vegas could not differentiate whether the increased RTT is due to route change or network congestion. On the other hand, Aid-Vegas uses the relative one-way delay to distinguish the congestion, so rerouting to a longer RTT doesn't affect the throughput. In the following, we show the simulation result of both Vegas and Aid-Vegas when packets are rerouted to a longer path with larger RTT.



(a) A single network topology for the simulation of Vegas and Aid-Vegas.



(b) Throughput comparison between Vegas and Aid-Vegas with rerouting.

Fig. 2. The simulation topology and results of rerouting

Figure 2 (a) shows the first network topology. A source S_1 of either Vegas or Aid-Vegas sends data packet to its destination D_1. The bandwidth and propagation delay are 10 Mb/s and 1 ms for each full-duplex access link, 1.5 Mb/s and 3 ms for the full-duplex trunk link from R_1 to R_2, from R_2 to R_3 and from R_3 to R_7, and 1.5 Mb/s and 4 ms for the full-duplex trunk link from R_1 to R_4, from R_4 to R_5, from R_5 to R_6 and from R_6 to R_7. At the beginning, the packets are routed through S_1, R_1, R_2, R_3, R_7, and D_1 in order. At 15^{th} second, the connection link from R_2 to R_3 is broken and then recovered at 35^{th} second. Therefore, the packets pass through the other path from 15^{th} till 35^{th} second. As shown in Fig. 2 (b), when the packets are routed through the path with shorter RTT, Vegas achieves high throughput and stabilizes at 1.5 Mb/s. However, the performance of Vegas degrades dramatically when the packets are rerouted through the other path. On the other hand, Aid-Vegas always maintains a steady throughput regardless of the route change.

5.2 Network Asymmetry

In this subsection, we are interested in the throughput of different mechanisms when the congestion is caused by additional backward traffic. Since TCP RoVegas [5] and Enhanced Vegas [6] improve the performance of Vegas when the congestion occurs in the backward path, we also compare Aid-Vegas with them. Therefore, we will see the performance of four mechanisms in this part.

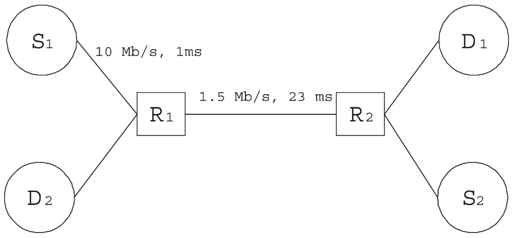
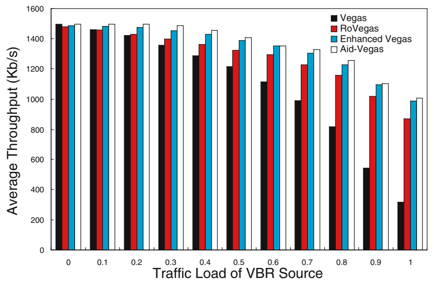
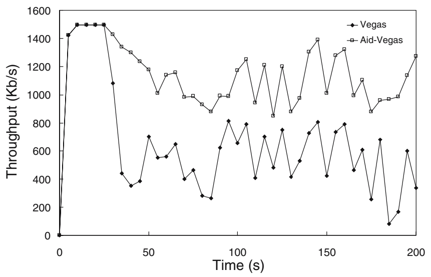


Fig. 3. A single bottleneck network topology for investigating throughput of different mechanisms when the congestion occurs in the backward path



(a) Average throughput versus different backward traffic loads for four schemes in the single bottleneck topology.



(b) Throughput comparison between Vegas and Aid-Vegas with the backward traffic load is 0.9.

Fig. 4. The simulation results of network asymmetry

The second network topology for the simulations is shown in Fig. 3. Sources, destinations, and routers are represented as S_i, D_i, and R_i respectively. A source and destination with the same suffix value represent a traffic pair. The bandwidth and propagation delay are 10 Mb/s and 1 ms for each full-duplex access link, 1.5 Mb/s and 23 ms for the full-duplex trunk link. A source of Vegas, RoVegas, Enhanced Vegas, or Aid-Vegas is attached to S_1 and a VBR source is attached to S_2. The S_1 starts sending data at 0 second, while S_2 starts at 25^{th} second.

From the results shown in Fig. 4 (a), we can observe that when different backward VBR traffic loads varies from 0 to 1, the average throughput of Aid-Vegas is higher than those of other mechanisms. For example, as the backward traffic load is 1, Aid-Vegas achieves a 3.2 times higher average throughput in comparison with that of Vegas. In addition, Fig. 4 (b) depicts the throughput comparison between Vegas and Aid-Vegas with a VBR source which has 1.35 Mb/s averaged sending rate. Obviously, we have demonstrated that Aid-Vegas significantly improves the connection throughput when the backward path is congested.

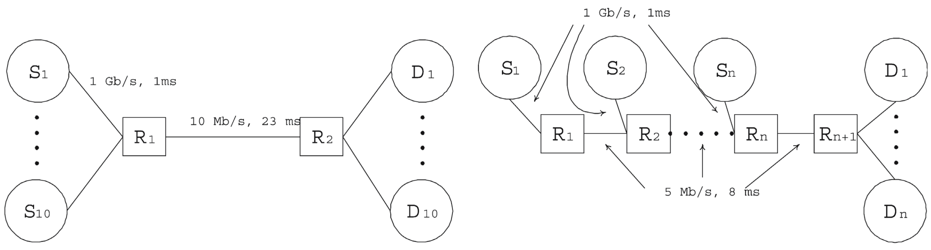
5.3 Fairness Improvement

Vegas experiences unfairness because it attempts to maintain Δ between two thresholds α and β by adjusting its congestion window size, but the range between α and β includes uncertainty to the achievable throughput of connections. Here, we show the comparison between Vegas and Aid-Vegas with multiple users competing network resources.

First, we are interested in those sources with same RTT. The network topology for simulations is shown in Fig. 5 (a), where the bandwidth and propagation delay are 1 Gb/s and 1 ms for full-duplex access link, and 10 Mb/s and 23 ms for full-duplex trunk link respectively. The sources are either Vegas or Aid-Vegas. We consider two cases here. One is that ten sources start at the same time, and the other is that ten connections from S_1 to S_{10} successively join the network one by one every 30 seconds. In addition, we don't change the default value of α and β. We use the fairness index [15] to represent the result of these two cases, as shown in Table 3, from which we observe that the fairness index of Aid-Vegas is higher than that of Vegas, especially when the sources start at the same time.

Table 3. The fairness index for Vegas and Aid-Vegas

	Vegas	Aid-Vegas
start at the same time	0.967	0.999
start at the different time	0.932	0.985



(a) With same RTT. (b) With different RTTs.

Fig. 5. The network topology for ten sources

Table 4. The fairness index for Vegas and Aid-Vegas

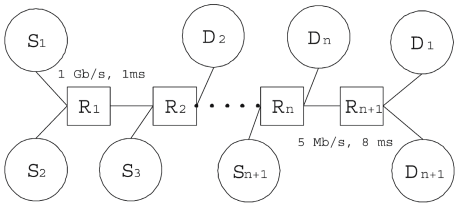
	Vegas	Aid-Vegas
$i = 2$	0.973	1.000
$i = 3$	0.943	0.998

Second, we simulate the source with different RTTs, and the network topology is shown in Fig. 5 (b). The bandwidth and propagation delay are 1 Gb/s and 1 ms for full-duplex access link, and 5 Mb/s and 8 ms for full-duplex trunk link respectively. The sources are either Vegas or Aid-Vegas. Table 4 depicts the fairness index for Vegas and Aid-Vegas with $i = 2$ and 3. We observe that the fairness index of Aid-Vegas is bigger than that of Vegas from Table 4.

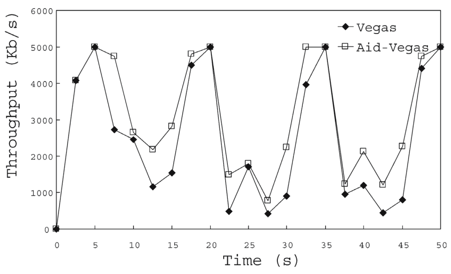
From these simulation results, we can observe that the Aid-Vegas is more suitable than Vegas for multiple sources with same mechanism.

5.4 Multiple Congested Links

Vegas adjusts the congestion window size according to the comparison result of Δ with α and β. However, packet routing though multiple congested links causes the RTT increased. As a result, Vegas mistakenly judges that congestion occurs and decreases the congestion window size. On the other hand, Aid-Vegas uses the relative one-way delay of each packet to distinguish congestion. If the traffic of all flows is steady, the curve of relative one-way delays will be short-term intervals. Figure 6 (a) shows the general network topology, where the bandwidth and propagation delay are 1 Gb/s and 1 ms for the full-duplex access link, and 5 Mb/s and 8 ms for the full-duplex trunk link respectively . The S_1 is either Vegas or Aid-Vegas source, and the other sources are VBR sources with 0.8 Mb/s or 1.0 Mb/s during ON periods. Figure 6 (b) depicts the throughput of Vegas and Aid-Vegas when there are two VBR sources (i.e., n = 2) in the network. The throughput is 0.8 Mb/s and 1.0 Mb/s for S_2 and S_3 respectively. S_2 is ON from 7^{th} to 16^{th} second, S_3 is ON from 21^{th} to 30^{th} second, and both are ON from 37^{th} to 45^{th} second. From Fig. 6 (b), we can observe that the performance of Aid-Vegas is higher than Vegas.



(a) Network topology.



(b) Throughputs.

Fig. 6. The simulation with multiple congested links

6 Conclusions

In this research, we propose Aid-Vegas for TCP Vegas. Comparing with other previous studies, Aid-Vegas provides a more effective way to solve the problems of rerouting and backward congestion, to enhance the fairness among the competing connections, and to improve the throughput when passing through multiple congested links. Through simulation, we demonstrate the effectiveness of Aid-Vegas. We will focus on its coexistence with other TCP implementations as well as its performance over wireless networks in our future work.

References

1. L. S. Brakmo and L. L. Peterson, 'TCP Vegas: End to End Congestion Avoidance on a Global Internet', *IEEE J. Select. Areas Commun., vol. 13, pp. 1465-1480, Oct. 1995.*
2. J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan, 'Evaluation of TCP Vegas: Emulation and Experiment', *ACM SIGCOMM'95, vol. 25, pp. 185-195, Aug. 1995.*
3. J. Mo, R. J. La, V. Anantharam, and J. Walrand, 'Analysis and Comparison of TCP Reno and Vegas', *IEEE INFORCOM'99, vol. 3, pp. 1556-1563, Mar. 1999.*
4. G. Hasegawa, M. Murata, and H. Miyahara, 'Fairness and Stability of Congestion Control Mechanism of TCP', *Telecommunication Systems Journal, pp. 167-184, Nov. 2000.*
5. Y. C. Chan, C. T. Chan, Y. C. Chen, and C. Y. Ho, 'Performance Improvement of Congestion Avoidance Mechanism for TCP Vegas', *IEEE ICPADS'2004, pp. 605-612, Jul. 2004.*
6. Y. C. Chan, C. T. Chan, and Y.C. Chen, 'An Enhanced Congestion Avoidance Mechanism for TCP Vegas', *IEEE Commun. Lett., vol. 7, issue 7, pp. 343-345, Jul. 2003.*
7. O. Elloumi, H. Afifi, and M. Hamdi, 'Improving Congestion Avoidance Algorithms for Asymmetric Networks', *in Conf. Rec. 1997 IEEE Int. Conf. Communications, pp. 1417-1421.*
8. C. P. Fu and S. C. Liew, 'A Remedy for Performance Degradation of TCP Vegas in Asymmetric Networks', *IEEE Commun. Lett., vol. 7, pp. 42-44, Jan. 2003.*
9. C. Parsa and J. J. Garcia-Luna-Aceves, 'Improving TCP Congestion Control over Internet with Heterogeneous Transmission Media', *in Conf. Rec. 1999 IEEE Int. Conf. Network Protocols, pp. 213-221.*
10. H. Cunqing and T.-S. P. Yum, 'The Fairness of TCP Vegas in Networks with Multiple Congested Gateways', *High Speed Networks and Multimedia Communications 5th IEEE International Conference on, pp. 115-121, July 2002.*
11. V. Jacobson and S. Floyd, 'TCP and Explicit Congestion Notification', *In Computer Communication Review, vol. 24, No. 5, pp. 8-23, Oct. 1994.*
12. H. Balakrishnan and V. N. Padmanabhan, 'How Network Asymmetry Affects TCP', *IEEE Commun. Mag., vol. 39, pp. 60-67, Apr. 2001.*
13. M. Jain and C. Dovrolis, 'End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput', *Networking, IEEE/ACM Transactions on, vol. 11, issue 4, pp. 537-549, Aug. 2003.*
14. M. Jain and C. Dovrolis, 'Pathload: A Measurement Tool for End-to-End Available Bandwidth', *In Passive and Active Measurements, Fort Collins, CO, Mar. 2002.*
15. R. Jain, D. Chiu, and W. Hawe, 'A Quantitative Measure Of Fairness and Discrimination for Resource Allocation in Shared Computer Systems', *DEC Research Report TR-301, Sep. 1984.*
16. http://www.isi.edu/nsnam/ns/