Performance Improvement of Congestion Avoidance Mechanism for TCP Vegas

Yi-Cheng Chan, Chia-Tai Chan, Yaw-Chung Chen, and Cheng-Yuan Ho
Department of Computer Science and Information Engineering
National Chiao Tung University, Hsinchu 300, Taiwan
ycchan@csie.nctu.edu.tw, ctchan@cht.com.tw, ycchen@csie.nctu.edu.tw, cyho@csie.nctu.edu.tw

Abstract

In this paper, we propose a router-based congestion avoidance mechanism (RoVegas) for TCP Vegas. 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 outperforms TCP Reno in many aspects. However, TCP Vegas suffers several problems that inhere in its congestion avoidance mechanism, these include issues of rerouting, persistent congestion, fairness, and network asymmetry. By performing the proposed scheme in routers along the round-trip path, RoVegas can solve the problems of rerouting and persistent congestion, enhance the fairness among the competitive connections, and improve the throughput when congestion occurs on the backward path. Through the results of both analysis and simulation, we demonstrate the effectiveness of RoVegas.

1. Introduction

With the fast growth of Internet traffic, how to efficiently utilize network resources is essential to a successful congestion control. Being a widely used end-to-end transport protocol on the Internet, Transmission Control Protocol (TCP) has several implementation versions (i.e., Tahoe, Reno, Vegas...) which intend to improve network utilization. Among these TCP variants, there are two notable approaches. One is Reno [1] which has been widely deployed on the Internet; the other is Vegas [2] with a claim of 37 to 71 percent throughput improvement over Reno was achieved.

To estimate the available bandwidth in the network, TCP Reno uses packet loss as an indicator for congestion. Its congestion window will be increased until packet loss is detected, at which point the congestion window is halved and then a linear increase algorithm will take over until further packet loss is experienced. It is known that the fast retransmit and recovery algorithm of TCP Reno can not recover multiple packet losses, the TCP may operate at a very low rate and lose a certain amount of throughput. TCP New Reno [3] is a modification to the fast retransmit and recovery. In TCP New Reno, a sender can recover from multiple packet losses without having to time out. TCP with selective acknowledgement (SACK) options [4] also has been proposed to efficiently recover from multiple packets loss. However, the additive increase and multiplicative decrease approach (AIMD) of Reno leads to periodic oscillations in the congestion window size, round-trip delay, and queue length of the bottleneck node. Recent works have shown that the oscillation may induce chaotic behavior into the network thus adversely affects overall network performance [5, 6].

TCP Vegas employs a fundamentally different congestion avoidance mechanism. It uses the difference between the expected and actual throughput to estimate the available bandwidth in the network. The idea is that when the network is not congested, the actual throughput will be close to the expected throughput. Otherwise the actual throughput will be smaller than the expected throughput. TCP Vegas uses the difference in throughput to gauge the congestion level in the network and update the congestion window size accordingly. As a result, TCP Vegas is able to detect network congestion in the early stage and successfully prevents periodic packet loss that usually occurs in Reno. Many studies have demonstrated that Vegas outperforms Reno in the aspects of overall network utilization [2, 7], stability [8, 9], fairness [8, 9], and throughput[2, 5, 7].

Although Vegas is superior to Reno in the aforementioned aspects, it suffers some problems that inhere in its congestion avoidance scheme, these include issues of rerouting [8], persistent congestion [8], fairness [9, 10, 11], and network asymmetry [12, 13, 14]. All these problems may be obstacles for Vegas to achieve a success.

In this work, we propose a router-based congestion avoidance mechanism for TCP Vegas (abbreviated as RoVegas hereafter). Through the proposed mechanism performed in routers along the round-trip path, RoVegas may solve the problems of rerouting and persistent congestion, enhance the fairness among the competitive connections,

Proceedings of the Tenth International Conference on Parallel and Distributed Systems (ICPADS'04)
1521-9097/04 $ 20.00 IEEE

and improve the throughput when congestion occurs on the backward path. Based on the results of analysis and simulation, we demonstrate the effectiveness of RoVegas.

The rest of this paper is organized as follows. Section 2 describes Vegas and its problems. Section 3 discusses the RoVegas. Section 4 present the simulation results. Lastly, we conclude this work in Section 5.

2. TCP Vegas and Its Problem Statements

TCP Vegas features three improvements as compared with TCP Reno: (1) a new retransmission mechanism, (2) an improved congestion avoidance mechanism, and (3) a modified slow-start mechanism. The detailed description of Vegas can be found in [2]. In this section, we first review the congestion avoidance mechanism of TCP Vegas and then describe its problems in detail.

2.1. Congestion Avoidance of TCP Vegas

Compared with Reno, TCP Vegas adopts a more sophisticated bandwidth estimation scheme that tries to avoid rather than to react to congestion. During the congestion avoidance phase, TCP Vegas does not continually increase the congestion window. Instead, it tries to detect incipient congestion by comparing the actual throughput to the expected throughput. Vegas estimates a proper amount of extra data to be kept in the network pipe and controls the congestion window size accordingly. It records the round-trip time (*RTT*) and sets *BaseRTT* to the minimum of ever measured *RTTs*. The amount of extra data is between two thresholds α and β, as shown in the following:

$$\alpha \leq (Expected - Actual) \times BaseRTT \leq \beta, \quad (1)$$

where *Expected* throughput is the current congestion window size divided by *BaseRTT*, and *Actual* throughput represents the current congestion window size divided by the newly measured *RTT*. The congestion window is kept constant when the amount of extra data is between α and β. If the amount 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 the amount is smaller than α, the connection is under a risk of under utilizing the available bandwidth. Hence, the congestion window size will be increased.

2.2. Problem Statements of Vegas

In Vegas, there are several problems inhere in its congestion avoidance mechanism that may have a serious impact on the performance. The problems are summarized as follows.

Rerouting: Since Vegas estimates the *BaseRTT* to compute the expected throughput and adjust its window size accordingly. Thus it is very important to estimate the quantity accurately for Vegas connections. Rerouting may cause a change of the fixed delay[1] that could result in substantial throughput degradation. When the route of a connection is changed, if the new route has a shorter fixed delay, it will not cause any serious problem for Vegas because most likely some packets will experience shorter round-trip time, 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 round-trip time is due to network congestion or route change. The source host may misinterpret the increased round-trip time as a signal of congestion in the network and decrease its window size. This is just the opposite of what the source should do.

Persistent Congestion: Persistent congestion is another problem caused by the incorrect estimation of *BaseRTT* [8]. Overestimation of the *BaseRTT* in Vegas may cause a substantial influence on the performance. Suppose that a connection starts while many other active connections also exist, the network is congested and the packets are accumulated in the bottleneck. Then, due to the queuing delay caused by extra data of other connections, the packets from the new connection may experience a round-trip time that are considerably larger than the actual fixed delay along the path. Hence, the window size of the new connection will be set to a value such that its expected amount of extra data lies between α and β; in fact, there may be much more extra data in the bottleneck queue due to the inaccurate estimation of the fixed delay. This scenario will repeat for each newly added connection, and it may cause the bottleneck node to remain in a persistent congestion. Persistent congestion is likely to happen in TCP Vegas due to its fine-tuned congestion avoidance mechanism.

Unfairness: Different from TCP Reno, Vegas is not biased against the connections with longer round-trip time [8, 9]. However, there is still unfairness comes with the nature of Vegas. According to the difference between the expected and actual throughputs, a Vegas source attempts to maintain an amount of extra data between two thresholds α and β by adjusting its congestion window size. The range between α and β induces uncertainty to the achievable throughput of connections. Since Vegas may keep different amount of extra data in the bottleneck even for the connections with the same round-trip path. Thus, it prohibits the better fairness achievement among the competitive connections.

Furthermore, the inaccurate computation of expected

[1] The fixed delay is the sum of propagation delay and packet processing time along the round-trip path. In other words, the fixed delay is the round-trip time without queuing delay.

throughput may also lead to unfairness. Recall that the computation of expected throughput is based on the measurement of *BaseRTT*. If Vegas connections can not estimate the *BaseRTT* accurately, it may affect the fairness achievement. When a new connection starts sending data while many other connections are also active, it may cause overestimation of the fixed delay and result in unfair distribution of bandwidth among the Vegas connections.

Network Asymmetry: Based on the estimated extra data kept on the bottleneck, Vegas updates its congestion window to avoid congestion as well as to maintain high throughput. However, a roughly measured *RTT* may lead to a coarse adjustment of congestion window size. If the network congestion occurs in the direction of ACK (backward path), it may underestimate the actual throughput 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 distinguish whether congestion occurs in the forward path or not and adjust the congestion window size accordingly.

Some prevalent networking technologies with asymmetry network characteristics, such as asymmetric digital subscriber line (ADSL), cable modem, and satellite-based networks, greatly increase the possibilities of backward path congestion. In these networks, it is very likely that the capacity of the forward direction is larger than that of backward direction. Both Reno and Vegas may suffer severe performance degradation in the case of backward path congestion, especially for Vegas [14]. Therefore, how to amend the deficiency of TCP Vegas in such situation becomes an important issue.

3. TCP RoVegas

From the above discussion, we can find that the coarse estimation of fixed delay along the round-trip path, *BaseRTT*, results in problems such as issues of rerouting, persistent congestion, and unfairness. A Vegas source is unable to distinguish whether congestion occurs in the forward path or not, this further leads to unnecessary throughput degradation when the congestion occurs on the backward path. In this work, we propose a router-based congestion avoidance mechanism (RoVegas) for TCP Vegas to deal with these problems. The details of the proposed mechanism is described as follows.

3.1. Proposed Mechanism

TCP Vegas estimates a proper amount of extra data to be kept in the network pipe and controls the congestion window size accordingly. The amount is between two thresholds α and β, as shown in Eq. (1). When backward congestion occurs, the increased backward queuing time will affect the *Actual* throughput and enlarge the difference between the *Expected* throughput and *Actual* throughput. It results in decreasing the congestion window size. Since the network resources in the backward path should not affect the traffic in the forward path, it is unnecessary to reduce the congestion window size when only backward congestion occurs.

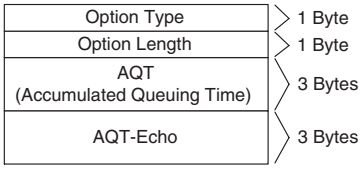

Figure 1. Fields of an AQT option.

A measured *RTT* can be divided into four parts: forward fixed delay (i.e., propagation delay and packet processing time), forward queuing time, backward fixed delay, and backward queuing time. To utilize the network bandwidth efficiently, we redefine the *Actual* throughput as

$$Actual' = \frac{CWND}{RTT - QT_b}, \tag{2}$$

where *RTT* is the newly measured round-trip time, QT_b is the backward queuing time, and *CWND* is the current congestion window size. Consequently, the $Actual'$ is a throughput that can be achieved if there is no backward queuing delay along the path.

To realize our scheme, we define a new IP option named AQT (accumulate queuing time) to collect the queuing time along the path. According to the general format of IP options described in [15], the fields of an AQT option are created as in Figure 1. The option type and length fields indicate the type and length of this IP option. The AQT field expresses the accumulated queuing time that a packet experienced along the route path. The AQT-Echo field echoes the accumulated queuing time value of an AQT option that was sent by the remote TCP.

A probing packet is a normal TCP packet (data or ACK) with AQT option in its IP header. When a RoVegas source sends out a probing packet, it sets the AQT field to zero. An AQT-enabled router (i.e., a router that is capable of AQT option processing) adds the queuing delay of a received probing packet to the AQT field. The queuing time is computed based on the queuing disciplines. The details regarding how to compute the queuing time of each received probing packet in various queuing disciplines is beyond the scope of this paper.

Whenever a RoVegas destination acknowledges a prob-

ing packet, it inserts an AQT option into the ACK. The AQT field is set to zero, and the AQT-Echo field is set to the value of the AQT field of the received packet. Through the AQT-enabled routers along the round-trip path, a RoVegas source is able to obtain both the forward queuing time (the value of AQT-Echo field) and backward queuing time (the value of AQT field) from the received probing packet. Moreover, for each probing packet received by a RoVegas source, the *BaseRTT* can be derived as follows:

$$BaseRTT = RTT - (\text{AQT} + \text{AQT-Echo}). \quad (3)$$

Notice that, the derived *BaseRTT* of a connection will be identical for each probing packet received when both the route and size of the probing packets are fixed. The derived *BaseRTT* of RoVegas represents the actual fixed delay along the round-trip path, if the path of a connection is rerouted and the fixed delay is changed, the newly derived *BaseRTT* may reflect the rerouting information. As a result, the issue of rerouting can be solved. Furthermore, since each connection of RoVegas is able to measure the fixed delay without bias, the problem of persistent congestion can be avoided and the fairness among the competitive connections can also be improved.

To avoid the unnecessary reduction of congestion window size, the proposed router-based congestion avoidance mechanism is described as follows:

- Derive the *Expected* throughput that is defined as the current congestion window size divided by *BaseRTT*.
- Calculate the $Actual'$ as the current congestion window size divided by the difference between the newly measured *RTT* and backward queuing time.
- Let $Diff = (Expected - Actual') \times BaseRTT$.
- Let w_{cur} and w_{next} be the congestion window sizes for the current *RTT* and the next *RTT*, respectively. The rule for congestion window adjustment is as follows:

$$w_{next} = \begin{cases} w_{cur} + 1, & if\ Diff < \alpha \\ w_{cur} - 1, & if\ Diff > \beta \\ w_{cur}, & if\ \alpha \leq Diff \leq \beta \end{cases}. \quad (4)$$

Due to the space limitation, the discussion of implementation issue and the steady-state performance analysis of both Vegas and RoVegas please refer to [16]. We now demonstrate the proposed scheme can improve the performance of TCP Vegas using the performance evaluation results presented in the following section.

4. Performance Evaluation

In this section, we compare the performance of TCP RoVegas with TCP Vegas by using the network simulator ns-2.1b9a [18]. We show the performance results in backward congestion environments, the bias experiments, the fairness investigations among the competitive connections, and the study of gradual deployment.

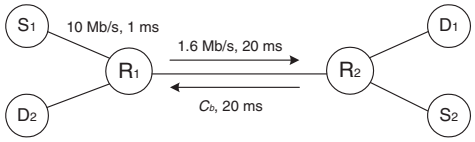

Figure 2. A single bottleneck network topology for investigating throughputs of Vegas and RoVegas when the congestion occurs on the backward path.

The FIFO service discipline is assumed. Every packet of RoVegas is a probing packet. Whenever a throughput of RoVegas is computed, the overhead induced by the AQT option will be subtracted from the throughput. Several VBR sources are used to generate backward traffic. These VBR sources are exponentially distributed ON-OFF sources. During ON periods, the VBR source sends data at 3.2 Mb/s. Unless stated otherwise, the size of each FIFO queue used in routers is 50 packets, the size of data packet is 1 Kbytes, and the sizes of ACKs are 40 and 48 bytes for Vegas and RoVegas respectively. To ease the comparison, we assume that the sources always have data to send.

4.1. Throughput Improvement

In this subsection, we investigate the throughputs of Vegas and RoVegas in two types of backward congestion. One is the congestion caused by network asymmetry, the other is the congestion caused by additional backward traffic.

The first network topology for the simulations is shown in Figure 2. Sources, destinations, and routers are expressed as S_i, D_i, and R_i respectively. A source and a 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, 1.6 Mb/s and 20 ms for the connection link from R_1 to R_2, and C_b and 20 ms for the connection link from R_2 to R_1, respectively. The C_b is set based on the normalized asymmetric factor *k* [17]. For example, if $k = 4$ and the size of data packet and ACK are 1 Kbytes and 40 bytes respectively, then the C_b is set to 16 Kb/s.

Asymmetric Networks: To evaluate the throughputs of Vegas and RoVegas in asymmetric networks, different values of *k* are used. A source S_1 of either Vegas or RoVegas sends data packet to its destination D_1. The size of each FIFO queue used in routers is 10 packets. Figures 3 and 4 exhibit the throughput performance of Vegas and RoVegas

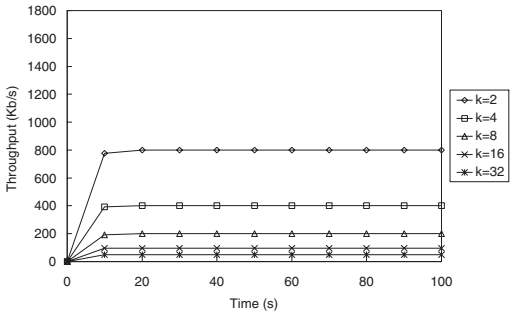

Figure 3. Throughputs of Vegas in asymmetric networks.

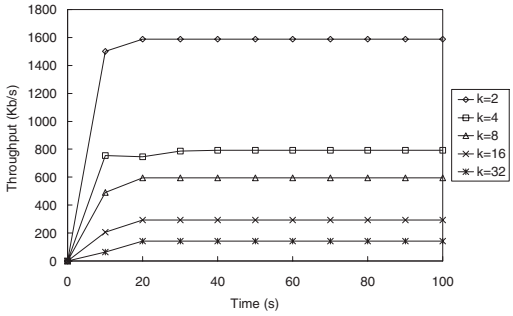

Figure 4. Throughputs of RoVegas in asymmetric networks.

in asymmetric networks respectively.

By observing the results shown in Figure 3, with the increasing value of *k* from 2 to 32, the throughput of Vegas degrades accordingly. As our analysis results, the throughput of Vegas in this scenario should be u_f/k (u_f is the forward link capacity). Obviously, the simulation results conform to our previous analysis.

Comparing the results of Figure 4 with that of Figure 3, we can find that the throughputs of RoVegas are much greater than that of Vegas. With $k = 2$, RoVegas maintains a high throughput at 1587.2 Kb/s in which the backward congestion seems not existing. The throughput ratios of RoVegas to Vegas in steady state are about 2 and 3 for k=2, 4 and k=8, 16, 32 respectively. Notably, all the simulation results shown in Figure 3 and Figure 4 are consistent with our analysis.

Symmetric Network With Backward Traffic: Asymmetric networks should not be the only reason causing backward congestion. Actually, even in a symmetric network the

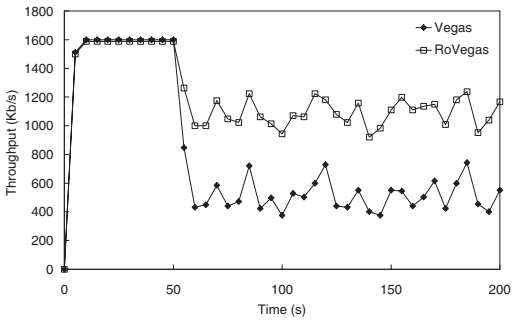

Figure 5. Throughput comparison between Vegas and RoVegas with the backward traffic load is 0.9 in the single bottleneck network topology.

backward congestion may still occur. In this simulation, a VBR source is used to generate backward traffic. The capacity of the backward bottleneck, C_b, is set to 1.6 Mb/s. A source of either Vegas or RoVegas 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 50 second. Figure 5 depicts the throughput comparison between Vegas and RoVegas with the VBR source which has 1.44 Mb/s averaged sending rate.

As shown in Figure 5, when the traffic source is Vegas only (0–50 second), it achieves high throughput and stabilizes at 1.6 Mb/s. However, the performance of Vegas degrades dramatically as the VBR source starts sending data. Although the overhead induced by AQT option slightly lower the throughput of RoVegas (0.8 %) during the preceding 50 seconds, nevertheless, RoVegas maintains a much higher throughput than that of Vegas while the backward congestion occurs. With the inference of the backward VBR traffic, the average throughput of Vegas is 521 Kb/s and RoVegas is 1092 Kb/s. Since we use the same traffic pattern of the VBR source while the throughput of Vegas or RoVegas is examined. Thus there appear some synchronized fluctuations of throughputs between Vegas and RoVegas.

To evaluate the average throughput of Vegas and RoVegas with different backward traffic loads, we set the VBR traffic loads to vary from 0 to 1. The traffic sources are the same as the above descriptions but the sources of either Vegas or RoVegas and VBR start at 0 second. The simulation period is 200 seconds for each sample point. From the simulation results shown in Figure 6, we can find that when the backward traffic load is not zero, RoVegas always achieves a higher average throughput than Vegas. For example, as the backward traffic load is 1, RoVegas achieves a 4.1 times higher average throughput in comparison with

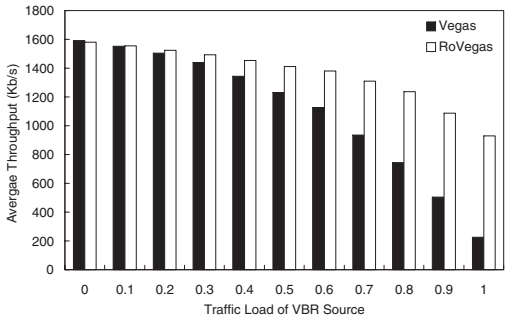

Figure 6. Average throughput versus different backward traffic loads for Vegas and RoVegas in the single bottleneck network topology.

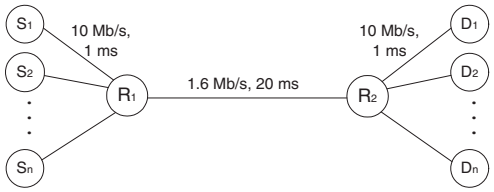

Figure 7. Network topology for studying the bias and fairness issues of Vegas and RoVegas.

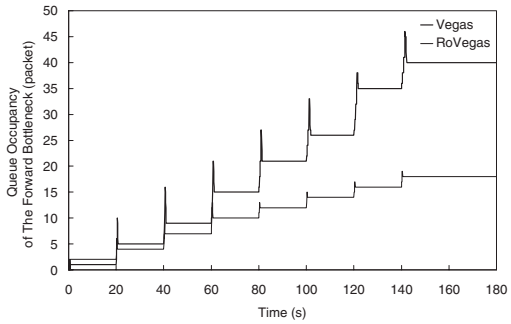

Figure 8. Queue occupancy of the forward bottleneck for Vegas and RoVegas.

that of Vegas. Obviously, we have demonstrated that RoVegas significantly improves the connection throughput when the backward path is congested.

4.2. Persistent Congestion

As a connection starts when there exist many other connections, the new connection may experience round-trip times that are considerably larger than the actual fixed delay along the path. Thus the *BaseRTT* of this new connection will be larger than it should be. Therefore the new connection will put a larger amount of extra data than its expected amount on the network. This bias may possibly drive the system to a persistent congestion.

In this subsection, we study the bias of Vegas through simulation. The simulation network topology is illustrated as in Figure 7 in which the bandwidth and propagation delay for each full-duplex link are depicted.

Eight connections of Vegas from S_1 to S_8 successively enter the network every 20 seconds. The α and β of Vegas are set to 1 and 3 respectively. Thus, the amount of extra data for each connection should be kept between 1 and 3 packets. From the results shown in Figure 8, we can observe that when the fourth connection of Vegas joins the network, the queue occupancy of the bottleneck increases to 15 packets. This amount of extra data is larger than the expected maximum amount (12 packets). Even worse, as the eighth connection goes into the network, the queue occupancy of the bottleneck is 40 packets. That is, averagely each connection of Vegas contributes 5 packets to the bottleneck. This situation will become worse and worse when more connections enter the network.

In contrast to the Vegas connections, each RoVegas connection keeps a proper amount of extra data in the bottleneck. When the eighth connection of RoVegas joins the network, the queue occupancy of the bottleneck is 18 packets. Since each connection of RoVegas is able to derive the fixed delay along the round-trip path, the bias of Vegas no longer exists in RoVegas.

4.3. Fairness Enhancement

Fairness is another important issue of Vegas. Although Vegas is not biased against the connections with longer round-trip time like Reno does, there is still unfairness occurred in Vegas. In this subsection, we investigate the fairness metric of Vegas and RoVegas.

The network topology for simulations is shown in Figure 7. Five connections of either Vegas or RoVegas from S_1 to S_5 successively join the network every 30 seconds. In order to remove the uncertainty induced by the range between α and β, we set α equal to β in two simulation scenarios. Figure 9 shows the results of simulations.

From the simulation results of Vegas presented in Figure 9(a) and Figure 9(b) we can see that no matter the values of α and β are equal or not, connections are unable to share the bandwidth fairly. According to our previous discussion, there are two criteria for achieving the fairness among the

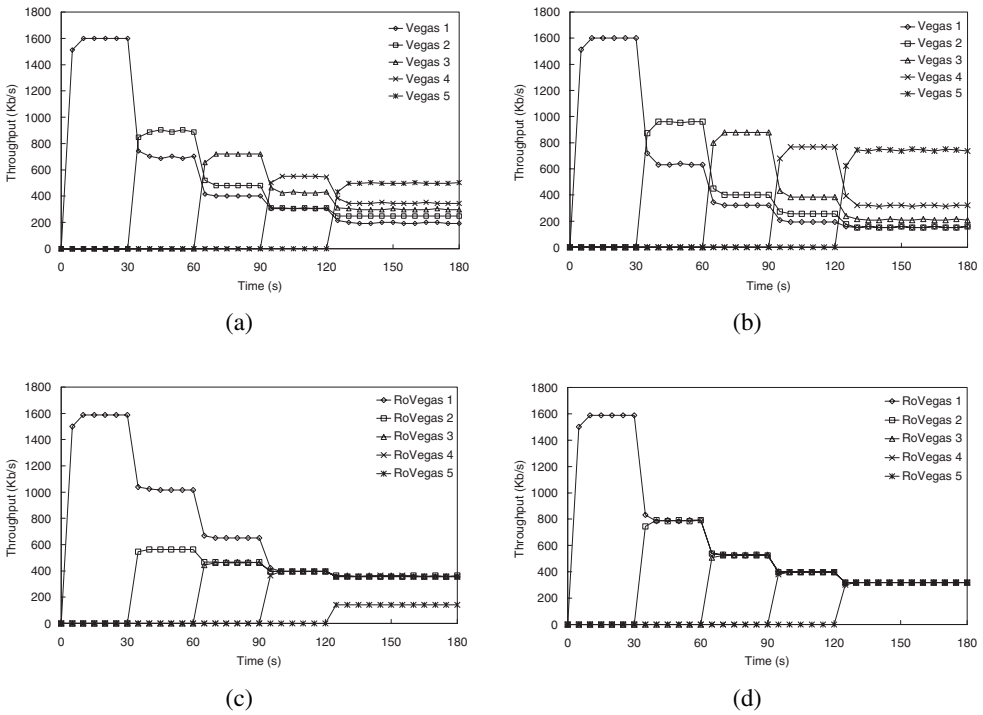

Figure 9. Fairness investigation of Vegas and RoVegas in which connections with the same propagation delay and successively enter the network every 30 second. (a) Vegas ($\alpha = 1,\ \beta = 3$). (b) Vegas ($\alpha = \beta = 2$). (c) RoVegas ($\alpha = 1,\ \beta = 3$). (d) RoVegas ($\alpha = \beta = 2$).

competitive connections of Vegas. One is that the measured *BaseRTT* must be precise enough. The other is that the uncertainty induced by the range between α and β must be removed. Connections in Figure 9(a) do not meet both criteria. Connections in Figure 9(b) do not conform to the first criterion. Therefore, both connections in these two figures are unable to fairly share the bandwidth of the bottleneck.

By observing the results of RoVegas shown in Figure 9(c) and Figure 9(d). Since connections in Figure 9(c) do not meet the second criterion of fairness, the throughputs of these connections are not identical. Finally, connections in Figure 9(d) meet both criteria, hence, all the connections share the bandwidth fairly.

4.4. Gradual Deployment

It can not be expected that all routers on the Internet are AQT-enabled while the AQT option is a new defined IP option. To consider the gradual deployment issue, for each ACK received by a RoVegas source, the *BaseRTT* should be measured as the following pseudo codes:

if (the ACK is a probing packet)
$BaseRTT_{temp} = RTT -$ (AQT + AQT-Echo)
/* *RTT* is the newly meausred round-trip time */
if ($BaseRTT_{temp} < BaseRTT$)
$BaseRTT = BaseRTT_{temp}$
else /* the ACK is not a probing packet) */
if ($RTT < BaseRTT$)
$BaseRTT = RTT$

If all bottleneck routers along the round-trip path are not AQT-enabled, RoVegas may behave like Vegas. Since RoVegas cannot obtain the backward queuing time (QT_b) to reduce the impacts of backward congestion, and may not estimate a precise *BaseRTT* to enhance the fairness and solve the persistent congestion. In this subsection, we try to explore whether a single AQT-enabled router on the end-to-end path may achieve the benefits from the RoVegas mechanism.

A parking lot network as shown in Figure 10 is used to examine the throughput of Vegas and RoVegas separately, here only R_2 is AQT-enabled. The bandwidth and propagation delay of each full-duplex access link and connection link are 10 Mb/s, 1 ms and 1.6 Mb/s, 10 ms respectively. Three VBR sources each with 1.28 Mb/s averaged sending to generate backward traffic. A connection of either Vegas or RoVegas from S_1 to D_1 and three VBR flows from

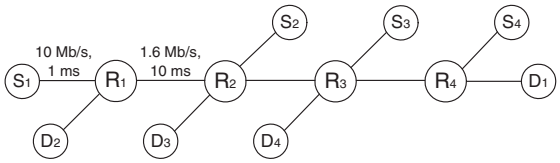

Figure 10. A parking lot network topology for investigating throughputs of Vegas and RoVegas in which only R_2 is AQT-enabled.

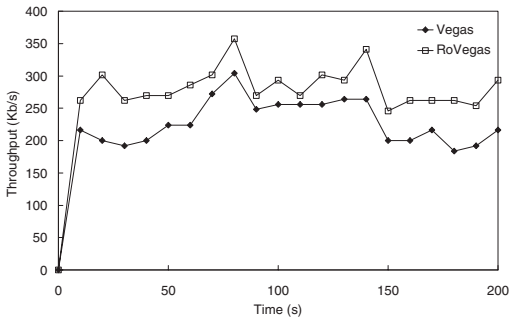

Figure 11. Throughput comparison between Vegas and RoVegas for only R_2 is AQT-enabled in the parking lot network.

S_2–S_4 to D_2–D_4 respectively. All traffic sources start sending data at 0 second. Despite only one AQT-enabled router R_2 located on the routing path, we can find that RoVegas still maintains a higher throughput than that of Vegas, as depicted in Figure 11. The simulation results imply that the proposed mechanism is amenable to gradual deployment for reducing the impacts of backward congestion. This feature may encourage the gradual adoption of RoVegas on the Internet.

5. Conclusion

In this research, we propose a router-based congestion avoidance mechanism, RoVegas, for TCP Vegas. Comparing with other previous studies, RoVegas provides a more effective way to solve the problems of rerouting and persistent congestion, to enhance the fairness among the competitive connections, and to improve the throughput when congestion occurs on the backward path. Through both analysis and simulation, the effectiveness of RoVegas has been shown. Nevertheless, there is still an issue that needs more attentions. It is enhancing the throughputs of Vegas when it performs with Reno head-to-head. Therefore, how to enable compatibility between Reno and Vegas would be one of our future works.

References

[1] V. Jacobson, "Modified TCP congestion avoidance algorithm," Tech. Rep., Apr. 1990.

[2] 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.

[3] J. C. Hoe, "Start-up dynamics of TCP's congestion control and avoidance schemes," Master's thesis, MIT, Jun. 1995.

[4] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," Internet Draft, April 1996.

[5] W. Feng and P. Tinnakornsrisuphap, "The failure of TCP in high-performance computational grids," *SC 2000: High-performance Networking and Computing Conf.*, Nov. 2000.

[6] A. Veres and M. Boda, "The chaotic nature of TCP congestion control," *IEEE INFORCOM'2000*, vol. 3, pp. 1715-1723, Mar. 2000.

[7] 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.

[8] 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.

[9] G. Hasegawa, M. Murata, and H. Miyahara, "Fairness and stability of congestion control mechanism of TCP," *Telecommunication Systems Journal*, pp. 167-184, Nov. 2000.

[10] C. Boutremans and J. L. Boudec, "A note on the fairness of TCP Vegas," *2000 International Zurich Seminar on Broadband Communications*, pp. 163-170, Feb. 2000.

[11] D. D. Luong and J.Biro, "On the proportional fairness of TCP Vegas," *IEEE GLOBECOM'01*, vol. 3, pp. 1718-1722, Nov. 2001.

[12] O. Elloumi, H. Afifi, and M. Hamdi, "Improving congestion avoidance algorithms for asymmetric networks," *1997 IEEE Int. Conf. Communications*, pp. 1417-1421, June 1997.

[13] 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.

[14] C. P. Fu, L. C. Chung, and S. C. Liew, "Performance degradation of TCP Vegas in asymmetric networks and its remedies," *2001 IEEE Int. Conf. Communications*, vol. 10, pp. 3229-3236, June 2001.

[15] J. Postel, "Internet Protocol," RFC791, Sep. 1981.

[16] Y. Chan, C. Chan, and Y. Chen, "RoVegas: Performance improvement of congestion avoidance mechanism for TCP Vegas," Tech. Rep., Sept. 2003, http://w3.nctu.edu.tw/∼u8517810/.

[17] T. V. Lakshman, U. Madhow, and B. Suter, "Window-based error recovery and flow control with a slow acknowledgement channel: A study of TCP/IP performance," *IEEE INFOCOM'97*, vol. 3, pp. 1199-1209, Apr. 1997.

[18] UCB/LBNL/VINT Network Simulator – ns (version 2), http://www.isi.edu/nsnam/ns/.