Performance Improvement of Delay-Based TCPs in Asymmetric Networks

Cheng-Yuan Ho, *Member, IEEE,* Cheng-Yun Ho, and Jui-Tang Wang

***Abstract*—Delay-based TCPs detect network congestion in the early stage and successfully prevent periodic packet loss that usually occurs in loss-based TCPs. It has been demonstrated that delay-based algorithms outperform loss-based schemes in many aspects. However, a delay-based TCP may not prevent unnecessary throughput degradation in asymmetric networks and when the congestion occurs in the backward path over symmetric links. In this letter, we modify the mechanism of measuring RTT value for delay-based TCPs without clock synchronization. Based on the simulation results, we show the effectiveness and utilities.**

***Index Terms*—Transport protocol, delay-based TCP, congestion control, and asymmetric networks.**

I. Introduction

With the fast growth of Internet traffic, how to efficiently utilize network resources is essential to a successful congestion control. Currently, most applications use the Transmission Control Protocol (TCP) to transmit data over the Internet because TCP provides reliable data transmission with embedded congestion control algorithm which effectively removes congestion collapses in the Internet by adjusting the sending rate according to the network's available bandwidth.

TCP has several implementation versions classed as two categories. One is called loss-based approach that uses packet loss as the only indication of congestion, e.g., TCP Reno [1]. The other is named delay-based scheme which makes congestion decisions that reduce the transmission rate based on round-trip time (RTT) variations, e.g., TCP Vegas [2] and FAST TCP [3]. Many studies have demonstrated that delay-based TCP outperforms loss-based TCP in the aspects of overall network utilization, stability, fairness, and throughput [2], [3], [4], [5]. However, above studies assume symmetric networks and little or no reverse traffic. It has been argued that asymmetric networks, reverse traffic, and available reverse bandwidth significantly influence TCP performance, especially for delay-based TCPs, as this affects the flow of acknowledgments (ACKs) back to the TCP sources [6].

In today's Internet, there is a significant amount of peer-to-peer traffic which can lead to heavy asymmetry on access links. For example, a user is watching an MPEG4 based movie when others are trying to download files from his or her PC. The TCP traffic on the uplink side will see heavy reverse traffic, which is not elastic. Several works have been proposed to improve TCP performance for asymmetric networks, but they have some limits and drawbacks.

To reduce the impacts of backward congestion, RoVegas [6] obtains the backward queuing time by performing its accumulate queuing time (AQT) scheme in routers along the round-trip path. However, if all bottleneck routers along the round-trip path are not AQT-enabled, RoVegas will behave like Vegas. Fu et al. [7] employ an end-to-end method to measure the actual flow rate on the forward path at a Vegas source. Based on the differences between the expected rate along the round-trip path and the actual flow rate on the forward path, the source adjusts the congestion window size (CWND) accordingly. However, in a backward congestion environment, the self-clocking behavior of TCP will be disturbed. Then the TCP traffic with bursty nature may lead to an over-increased CWND and cause congestion on the forward path. Ge and Tan [8] divide a RTT into a forward trip time and a backward trip time to improve throughput of FAST TCP by removing the effects of backward path congestion. However, they assume the forward fixed delay time, including propagation delay and packet processing time, is similar to the backward fixed delay time. Ge et al. [9] propose a model of FAST TCP in asymmetric networks and show that asymmetry does not affect stability of FAST TCP, but throughput; however, no solutions are mentioned there.

In this work, we modify the mechanism of measuring RTT value for delay-based TCPs. In order to find out the minimum fixed delay on the backward path without clock synchronization, our mechanism uses two TCP timestamps options to record the time of a destination receiving a data packet and sending an ACK. Based on the simulation results, the proposed mechanism may improve the TCP performance in asymmetric networks and with backward path congestion.

The rest of this paper is organized as follows. The proposed mechanism is described in Section II. Section III shows the simulation results. Finally, Section IV summarizes this work.

II. The Proposed Mechanism

Different from loss-based mechanisms, which detect network congestion based on packet losses, delay-based TCPs control the congestion window size based on the minimum of ever measured RTTs (BaseRTT) and the average RTT (or the average queueing delay). For example, TCP Vegas estimates a proper amount of extra data (Δ) to be kept in the network pipe. Moreover, Δ is between two thresholds α and β, as shown in the following:

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

Manuscript received August 23, 2010. The associate editor coordinating the review of this letter and approving it for publication was S. Pierre.

C.-Y. Ho (corresponding author) and C.-Y. Ho are with the Department of Computer Science, National Chiao Tung University, No. 1001, Ta Hsueh Road, Hsinchu, 300, Taiwan (e-mail: {tommyho, cyho}@cs.nctu.edu.tw).

J.-T. Wang is with the Industrial Technology Research Institute (ITRI), Rm. 505, Bldg. 51, 195, Sec.4, Chung Hsing Rd. Chutung, Hsinchu, 300, Taiwan (e-mail: rtwang@itri.org.tw).

Digital Object Identifier 10.1109/LCOMM.2011.011311.101535

where *Expected* throughput is the current CWND divided by BaseRTT, and *Actual* throughput represents the CWND divided by the newly measured RTT. The CWND is 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 CWND will be reduced. Otherwise, the connection may be under utilizing the available bandwidth. Hence, the CWND will be increased. Similarly, FAST adjusts its congestion window according to:

$$W \longleftarrow \min\{2W, (1-\gamma)W + \gamma(\frac{BaseRTT}{RTT}W + \alpha)\}, \quad (2)$$

where *W* is the CWND, $\gamma \in (0, 1]$, and α is a positive protocol parameter that determines the total number of packets queued in routers in equilibrium along the flow's path. In addition, FAST TCP can be regarded as a scaled version of Vegas [3].

When backward congestion occurs, the increasing backward queueing time will affect the RTT value and enlarge the difference between BaseRTT and RTT. This results in decreasing the CWND. Since the network resources in the backward path should not affect that in the forward path, it is unnecessary to reduce the CWND when backward congestion happens.

A RTT can be divided into four parts: forward fixed delay, forward queuing time, backward fixed delay, and backward queuing time. Let the difference of system clocks in the source and the destination be T_D. Assume the sender transmits the data packet i at the time T_{1i} and this data packet arrives the destination at the time T_{2i} called arriving time (AT). Then, the ACK packet for the data packet i leaves the receiver at the time T_{3i} named leaving time (LT) and the sender gets this ACK packet at the time T_{4i}. Therefore, the end-to-end trip time of the forward data packet i (T_{F_i}) and the backward ACK packet (T_{B_i}) will be $T_{2i} - T_{1i} + T_D$ and $T_{4i} - T_{3i} - T_D$, respectively. Support $T_{F_{min}}$ and $T_{B_{min}}$ denote the minimum forward time and backward time that the source ever measured from N packets. In other words, $\exists\ h, k \in [1, N]$ such that $T_{F_{min}} = T_{2h} - T_{1h} + T_D$ and $T_{B_{min}} = T_{4k} - T_{3k} - T_D$.

In order to realize the implementation, we make use of the TCP timestamps option[1] to obtain the AT and LT. When a destination receives a packet, it records the current time. As the receiver acknowledges this packet, it inserts two timestamps including AT and LT into the ACK packet. In addition, a delay-based TCP with the proposed mechanism is called xxx-ALT, where xxx is the name of the delay-based TCP, as Vegas-ALT.

To utilize the network bandwidth efficiently, ALT only redefines the $BaseRTT$ and RTT_i as $BaseRTT = T_{F_{min}} + T_{B_{min}} = T_{2h} - T_{1h} + T_{4k} - T_{3k}$ and $RTT_i = T_{F_i} + T_{B_{min}} = T_{2i} - T_{1i} + T_{4k} - T_{3k}$, respectively. Consequently, the RTT may be achieved if there is no backward queueing delay along the path. The way of adjusting congestion window is same as the TCP mechanism using ALT. Thus, avoiding the unnecessary reduction of TCP congestion window size, a delay-based TCP with ALT is more effective in improving the throughput.

Note that we only modify the RTT measurement mechanism of delay-based TCPs. Other mechanisms such as fast retransmit and fast recovery are the same. For example, when a severe congestion happens in backward path, Vegas-ALT and Vegas will make the same decisions. If none of the lost ACKs are duplicate ACKs, Vegas-ALT and Vegas will adjust their CWNDs based on Eq. (1). On the other hand, when a duplicate ACK is received, they check whether they need to retransmit a packet. If so, the packet is retransmitted. After retransmission, the CWND is reduced to alleviate the network congestion. If the lost packet has been transmitted just once, the CWND will be three fourth of the previous size. Otherwise, one-half of the previous CWND is set as the current CWND.

[1]RFC 1323 defines 10 bytes to the TCP timestamps option. Using timestamps will increase 25% space to the basic TCPIP header, 40 bytes; however, compared to the data packet, the timestamps option is much smaller.

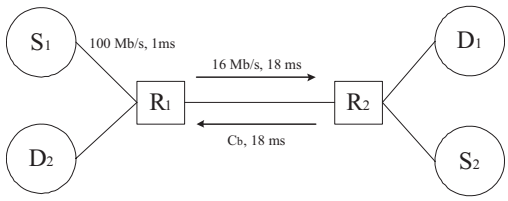

Fig. 1. A single bottleneck network topology with backward congestion.

III. Performance Evaluation

In this section, we compare the performance of TCP Vegas-ALT with TCP Vegas and FAST-ALT with FAST by using the network simulator *ns-2* [10]. The FIFO service discipline is assumed. Every packet of Vegas-ALT and FAST-ALT is a probing packet. Whenever a throughput of proposed mechanism is computed, the overhead induced by the ALT option will be subtracted from the throughput. Several variable bit rate (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 30 Mb/s. Unless stated otherwise, the size of each FIFO queue used in routers is 200 packets, the size of data packet is 1 KB, and the sizes of ACKs are 40 and 50 bytes for Vegas (or FAST) and Vegas-ALT (or FAST-ALT), respectively. To ease the comparison, we assume sources always have data to send.

The network topology for the simulations is shown in Fig. 1. Sources, destinations, and routers are expressed as S_i, D_i, and R_i, respectively. A source and a destination with the same subscript value represent a traffic pair. The bandwidth and propagation delay are 100 Mb/s and 1 ms for each full-duplex access link, 16 Mb/s and 18 ms for the connection link from R_1 to R_2, and C_b and 18 ms for the connection link from R_2 to R_1, respectively. The C_b is set based on the normalized asymmetric factor k [11]. For example, if $k = 4$ and the size of data packet and ACK are 1 KB and 40 bytes, respectively, then the C_b is set to 160 Kb/s.

A. Asymmetric Networks

To evaluate throughputs of TCP Vegas and Vegas-ALT in asymmetric networks, different values of k are used. A source S_1 of either Vegas or Vegas-ALT sends data packet to its destination D_1. Figures 2 and 3 exhibit the throughput performance of Vegas and Vegas-ALT in asymmetric networks, respectively. By observing the results shown in Fig. 2, with the increasing value of k from 2 to 32, the throughput of Vegas degrades accordingly. Comparing the results of Fig. 3 with that of Fig. 2, we can find that the throughput of Vegas-ALT is much greater than that of Vegas. With $k = 2$, Vegas-ALT maintains a high throughput at 16 Mb/s in which

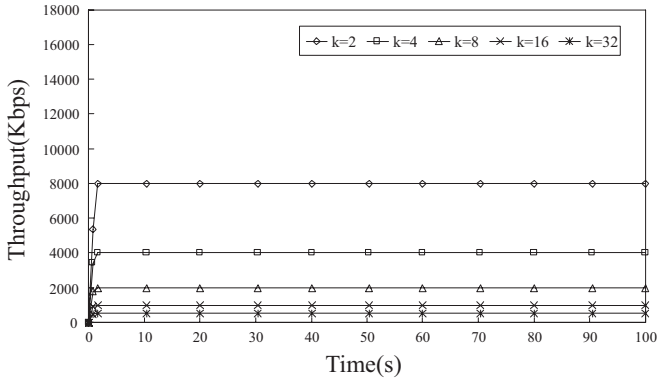

Fig. 2. Throughput of Vegas in asymmetric networks.

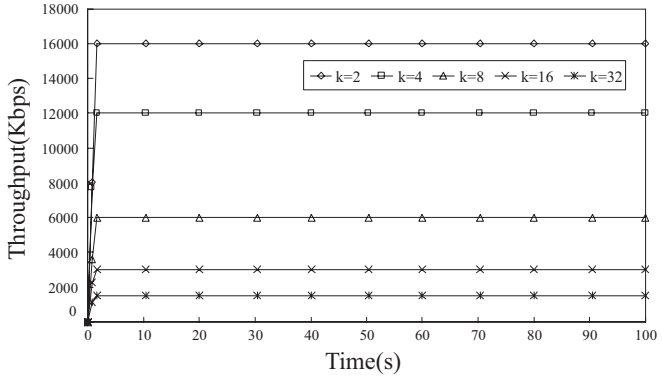

Fig. 3. Throughput of Vegas-ALT in asymmetric networks.

the backward congestion seems not existing. The throughput ratios of Vegas-ALT to Vegas in steady state are about 2 and 3 for $k = 2$ and $k = 4$, 8, 16, 32, respectively. On the other hand, the outcome of comparing FAST-ALT with FAST is similar to the above. However, the space is limited, so the simulation results of FAST and FAST-ALT are omitted.

B. Symmetric Network with Backward Traffic

The reason of causing backward congestion should not only be asymmetric networks. Actually, even in a symmetric network, the backward congestion may still occur. We use a VBR source with different averaged sending rates to examine the throughputs of Vegas, Vegas-ALT, FAST, and FAST-ALT separately in the single bottleneck network as shown in Fig. 1. In addition, the VBR traffic loads vary from 0 to 1. The capacity of the backward bottleneck, C_b, is set to 16 Mb/s. A source of Vegas, Vegas-ALT, FAST, or FAST-ALT is attached to S_1, and a VBR source is attached to S_2. Both traffic sources (S_1 and S_2) start sending data at the 0^{th} second. The simulation period is 200 seconds for each sample point. From the simulation results shown in Fig. 4, we can find that when the backward traffic load is not zero, Vegas-ALT always achieves a higher average throughout than Vegas, that is same as FAST-ALT and FAST. For example, when the VBR traffic load is 0.9, the average throughput of Vegas is about 1.6 Mb/s, Vegas-ALT is 6.1 Mb/s, FAST is about 4.7 Mb/s and FAST-ALT is 5.7 Mb/s. While the backward traffic load is 1, for instance, Vegas-ALT achieves a 3.75 times higher average throughput in comparison with Vegas, and the average throughput of FAST-ALT is 1.33 times greater than that of FAST. Note that we use the VBR source with same traffic

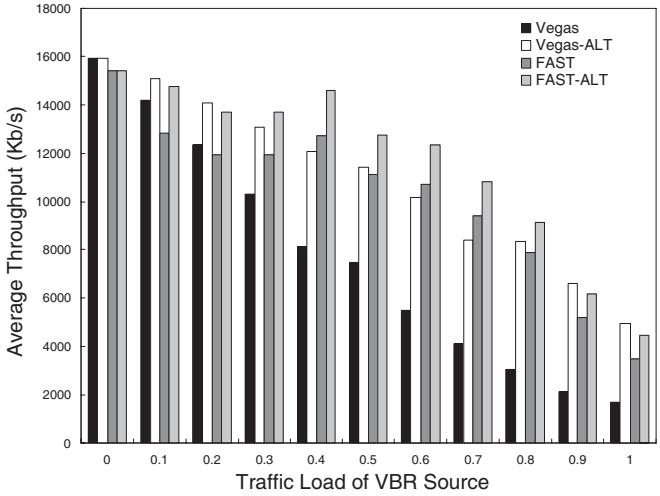

Fig. 4. Four TCPs' average throughput vs. different backward traffic loads.

pattern to examine four TCP versions; therefore, there appear some synchronized fluctuations of throughput among them.

IV. Conclusions

In this article, we modify the mechanism of measuring RTT value for delay-based TCPs. Comparing with other previous studies, delay-based TCPs with ALT provide a more effective way to improve the connection throughput in asymmetric networks and when congestion occurs on the backward path. The simulation results show the effectiveness of our proposed mechanism. Nevertheless, there is still an issue that needs more attentions. It is enhancing the throughput of the delay-based mechanism when it performs with loss-based scheme head-to-head. Therefore, how to enable compatibility between delay-based and loss-based TCPs would be our future work.

References

[1] V. Jacobson, "Modified TCP congestion avoidance algorithm," technical report, Apr. 1990.
[2] L. S. Brakmo and L. L. Peterson, "TCP Vegas: end to end congestion avoidance on a global Internet," *IEEE J. Sel. Areas Commun.*, vol. 13, pp. 1465-1480, 1995.
[3] D. X. Wei, C. Jin, S. H. Low, and S. Hegde, "FAST TCP: motivation, architecture, algorithms, performance," *IEEE/ACM Trans. Networking*, vol. 14, no. 6, pp. 1246-1259, Dec. 2006.
[4] W. Feng and P. Tinnakornsrisuphap, "The failure of TCP in high-performance computational grids," in *Proc. ACM/IEEE SC 2000 Conference (SC'00)*, p. 37, Nov. 2000.
[5] L. Tan, L. Dong, Y. Cao, and M. Zukerman, "Fairness comparison of FAST TCP and TCP Reno," *Computer Commun.*, vol. 30, no. 6, pp. 1375-1382, Mar. 2007.
[6] Y.-C. Chan, C.-T. Chan, and Y.-C. Chen, "RoVegas: a router-based congestion avoidance mechanism for TCP Vegas," *Computer Commun.*, vol. 27, no. 16, pp. 1624-1636, Oct. 2004.
[7] C. P. Fu and S. C. Liew, "A remedy for performance degradation of TCP Vegas in asymmetric networks," *IEEE Commun. Lett.*, vol. 7, no. 1, pp. 42-44, Jan. 2003.
[8] F. Ge and L. Tan, 'Improving FAST TCP performance in asymmetric networks," in *Proc. IEEE AusWireless'07*, p. 56, Aug. 2007.
[9] F. Ge, L. Tan, and M. Zukerman, "Throughput of FAST TCP in asymmetric networks," *IEEE Commun. Lett.*, vol. 12, no. 2, pp. 158-160, Feb. 2008.
[10] http://www.isi.edu/nsnam/ns/
[11] 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," in *Proc. IEEE INFOCOM*, vol. 3, pp. 1199-1209, Apr. 1997.