An Enhanced Congestion Avoidance Mechanism for TCP Vegas

Yi-Cheng Chan, Chia-Tai Chan, and Yaw-Chung Chen, *Member, IEEE*

***Abstract*—TCP Vegas detects network congestion in the early stage and successfully prevents periodic packet loss that usually occurs in traditional schemes. It has been demonstrated that TCP Vegas achieves much higher throughput than TCP Reno. However, TCP Vegas cannot prevent unnecessary throughput degradation when congestion occurs in the backward path. In this letter, we propose an enhanced congestion avoidance mechanism for TCP Vegas. By distinguishing whether congestion occurs in the forward path or not, it significantly improves the connection throughput when the backward path is congested.**

***Index Terms*—Congestion avoidance, TCP vegas, transport protocol.**

I. Introduction

With the fast growth of Internet traffic, how to efficiently utilize network resources becomes an important issue. 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 attempts to control and avoid congestion by monitoring the difference between the measured throughput and expected throughput. It uses the congestion window size and measured round-trip time (*RTT*) to estimate the amount of data in the network pipe and maintain extra data with amount between the lower threshold (α) and the upper threshold (β). By adjusting source congestion window size, it keeps an appropriate amount of extra data in the network 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 distinguish whether congestion occurs in the forward path or not and adjust the congestion window size precisely.

Several works have been proposed to improve TCP performance for asymmetric networks. These mechanisms obtain either the forward trip time [2] or the actual flow rate on the forward path [3] 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 [4], these approaches are not effective for handling the Vegas' asymmetry problem [3]. By using the relative delay estimation along the forward path, TCP Santa Cruz [5] is able to identify the direction of congestion. However, it is not for rate-based Vegas. In this work, we propose an enhanced congestion avoidance mechanism for TCP Vegas (Enhanced Vegas), our mechanism uses TCP timestamps option to estimate queueing delay on the forward and backward path separately without clock synchronization. By distinguishing the direction along where congestion occurs, Enhanced Vegas significantly reduces the impact and improves the throughput in the case of backward congestion.

The rest of this letter is organized as follows. Section II describes the Enhanced Vegas. Section III shows the simulation results. Section IV summarizes this work.

II. Enhanced Vegas

Different from Tahoe and Reno, which detect network congestion based on packet losses, 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 the following:

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

where *Expected* throughput is the current congestion window size divided by *BaseRTT* (i.e., the minimum of ever measured *RTT*s), and *Actual* throughput represents the current congestion window size divided by the newly measured *RTT*.

When backward congestion occurs, the increasing backward queueing time will affect the *Actual* throughput and enlarge the difference between the *Expected* throughput and *Actual* throughput. This results in decreasing the congestion window size. Since the network resources in the backward path should not affect that in the forward path, it is unnecessary to reduce the congestion window size when backward congestion occurs.

A measured *RTT* can be divided into four parts: forward fixed delay time (i.e., propagation delay and packet processing time), forward queueing time, backward fixed delay time, and

Manuscript received January 10, 2003. The associate editor coordinating the review of this letter and approving it for publication was Prof. D. Petr. This work was supported in part by the National Science Council, Taiwan, R.O.C., under Grant NSC 91-2213-E-009-017.

Y.-C. Chan and Y.-C. Chen are with the Department of Computer Science and Information Engineering, National Chiao Tung University, Hsinchu 30050, Taiwan, R.O.C. (e-mail: ycchan@csie.nctu.edu.tw; ycchen@csie.nctu.edu.tw).

C.-T. Chan is with the Telecommunication Laboratories, Chunghwa Telecom Company, Ltd, Taipei 106, Taiwan, R.O.C. (e-mail: ctchan@cht.com.tw).

Digital Object Identifier 10.1109/LCOMM.2003.814715