A CONTROL THEORETICAL APPROACH TO A WINDOW-BASED FLOW CONTROL MECHANISM WITH EXPLICIT CONGESTION NOTIFICATION

Hiroyuki Ohsaki, Masayuki Murata, Toshimitsu Ushio, and Hideo Miyahara

Department of Information and Computer Sciences
Faculty of Engineering Science, Osaka University
1-3 Machikaneyama, Toyonaka, Osaka 560, Japan
oosaki@ics.es.osaka-u.ac.jp

Abstract — A window-based flow control mechanism is a sort of feedback-based congestion control mechanisms, and has been widely used in current TCP/IP networks. Recently proposed TCP Vegas is another version of the TCP mechanism and has potential to achieve much better performance than current TCP Tahoe and Reno. However, it has not been fully investigated how to determine control parameters of TCP Vegas. In this paper, we focus on a window-based flow control mechanism based on a congestion avoidance mechanism of TCP Vegas. We first present a control theoretic analysis for its stability. We then discuss several drawbacks of the window-based flow control mechanism, being inherited from TCP Vegas, through simulation experiments. We finally investigate how these drawbacks is solved by incorporating the ECN (Explicit Congestion Notification) mechanism into the window-based flow control mechanism.

I. Introduction

In a packet-switched network, a feedback-based congestion control mechanism is essential to provide data transfer services efficiently. Its main objective is to prevent packet losses in the network, and to utilize network resources effectively. The current Internet uses a window-based flow control mechanism in TCP, as the feedback-based congestion control mechanism. As an example, a version of TCP mechanism called *TCP Reno* uses packet losses in the network as feedback information since packet losses implies congestion occurrence in the network [1, 2]. In short, the congestion control mechanism of TCP Reno first increases its window size, and as soon as it detects packet losses in the network, it reduces its window size. TCP Reno repeats this process indefinitely during the connection.

Recently another version of TCP called *TCP Vegas* has been proposed by Brakmo *et al.*, which can achieve better performance than TCP Reno [3, 4]. TCP Vegas has following advantages over TCP Reno: (1) a new retransmission mechanism, (2) an improved congestion avoidance mechanism that controls buffer occupancy, and (3) a modified slow-start mechanism. With these features, it has been reported in [4] that total throughput of TCP Vegas becomes 37–71 % better than TCP Reno, and that the number of retransmitted packets of TCP Vegas can be reduced to about 1/5–1/2 of TCP Reno. The performance improvement is mainly achieved by the congestion avoidance mechanism of TCP Vegas, which uses a measured round-trip time of the packet — i.e., duration between the source host sends the packet and it receives its corresponding ACK (acknowledgment) packet. More specifically, TCP Vegas measures a round-trip time of a packet, and estimates the number of queued packets in the router's buffer. It then controls its window size to make it constant. There is no need for the source host to wait for packet losses to know occurrence of congestion in the network. The window size of TCP Vegas becomes stabilized when the network is in steady state, and therefore it can achieve much better throughput than TCP Reno.

In this paper, we first derive a condition that window sizes of TCP connections and a queue length (i.e., the number of packets waiting in the router's buffer) are stabilized in steady state, which we will call *a stability condition*. Our previous work has shown that the window-based flow control mechanism works quite efficiently in terms of stability and transient performance when several control parameters are chosen appropriately [5]. However, as we will discuss in Section III, it has several drawbacks. These drawbacks are directly inherited from the congestion avoidance mechanism of TCP Vegas. In this paper, we therefore discuss how the ECN (Explicit Congestion Notification) mechanism can be incorporated into the window-based flow control mechanism based on TCP Vegas, and demonstrate its effectiveness through simulation experiments.

Organization of this paper is as follows. In Section II, we explain our window-based flow control mechanism based on the congestion avoidance mechanism of TCP Vegas followed by our control theoretic analysis for its stability. In Section III, we perform simulation experiments to validate our analysis, and to investigate several drawbacks of the window-based flow control mechanism. In Section IV, we discuss the ECN mechanism, and how it can be incorporated into the window-based flow control mechanism in TCP/IP

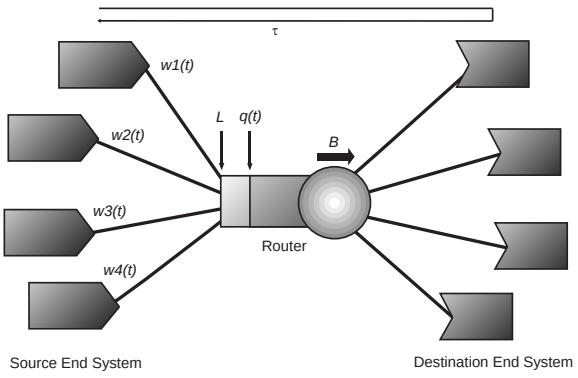

Fig. 1: Analytic model.

networks. Finally we conclude this paper and discuss future works in Section V.

II. Analysis of Window-Based Flow Control Mechanism

In this section, we present our control theoretical analysis of the window-based flow control mechanism based on TCP Vegas. Refer to [5] for more detail.

A. Analytic Model

We first explain the congestion avoidance mechanism of TCP Vegas. For detailed explanation, refer to [4]. In TCP Vegas, each source host maintains τ, which is a minimum round-trip time obtained when the network is not congested. That is, the minimum round-trip time τ corresponds to the sum of all propagation delays and processing delays at the routers. Hereafter, we call the minimum round-trip time, τ, the *propagation delay* for brevity. The source host is allowed to emit packets of its current window-size (denoted by w) per round-trip time. Therefore, its effective throughput would be w/τ if there is no congestion in the network. Each source host obtains the actual round-trip time by measuring time duration between a transmission time of a packet and arrival of its corresponding ACK packet. Let r be the actual round-trip time measured at the source host, and $\overline{w}$ be the number of packets the source host sent in the previous round-trip time. Its actual throughput is given by $\overline{w}/r$. TCP Vegas then computes the difference between expected throughput and actual throughput as

$$d = \frac{w}{\tau} - \frac{\overline{w}}{r}.$$

TCP Vegas changes its window size, w, according to relations among d and two threshold values, α and β. If d is less than α, the window size is linearly increased by one packet in the next round-trip time. If d is greater than β, the window size is linearly decreased by one packet in the next round-trip time. Otherwise, the window size is unchanged.

In this paper, we model the above-mentioned congestion control mechanism of TCP Vegas as follows. Figure 1 depicts our analytic model used throughout this paper. The number N of source hosts are connected to corresponding destination hosts through a single bottleneck router. TCP Vegas changes its window size once every round-trip time. We therefore consider the system as a discrete-time model, where each time slot corresponds to the round-trip time. Note that since the round-trip time changes as the network status changes, the length of one slot is not fixed in our model.

Let $w_n(k)$ be the window size of the source host n ($1 \leq n \leq N$) at slot k. This indicates that the source host n can inject $w_n(k)$ packets into the network during slot k. We assume that each source host always has packets to transmit so that the number $w_n(k)$ of packets are sent at slot k. Let $q(k)$ be the number of packets queued in the router's buffer at slot k, and L be the buffer size of the router. At the router, all packets coming from source hosts are processed in a FIFO (First-In First-Out) manner; that is, all packets are first queued in the single buffer, and then transmitted onto the output link in order. We denote the bandwidth of the router (i.e., the processing speed of the router or the bandwidth of the output link) by B. Note that $w_n(k)$ (the window size), $q(k)$ (the number of packets in the router's buffer), and L (the buffer size) are represented in units of packets.

In a round-trip time, TCP Vegas allows the source host consume the bandwidth being worth of its given window size. Provided that round-trip times of all connections are equal, the number of packets in the buffer at slot $k+1$, $q(k+1)$, is given by the following equation.

$$q(k+1) = \min(\max(\sum_{n=1}^{N} w_n(k) - B\, r(k), 0), L),$$

where $r(k)$ denotes the round-trip time at slot k.

TCP Vegas changes its window size based on the measured round-trip time. By letting $r(k)$ be the round-trip time observed at the source host n at slot k, the difference between the expected throughput and the actual throughput, $d(k)$, is computed as

$$d(k) = \frac{w_i(k)}{\tau} - \frac{w_i(k)}{r(k)}, \tag{1}$$

where τ is the round-trip time when there is no waiting packets in the router's buffer. The round-trip time, $r(k)$, is determined by τ and the number of packets in the buffer; Namely,

$$r(k) = \tau + \frac{q(k)}{B}.$$

TCP Vegas linearly increases or decreases its window size based on $d(k)$. The window size of the source host n at slot $k+1$, $w_n(k+1)$, is determined as

$$w_n(k+1) = \begin{cases} w_n(k)+1 & \text{if } d(k) < \alpha \\ w_n(k)-1 & \text{if } d(k) > \beta \\ w_n(k) & \text{otherwise} \end{cases}. \tag{2}$$

In the above equation, two threshold values, α and β, are control parameters at the source host, which specify the amount of excess packets the source host is permitted to send in a round-trip time. However, we modify Eq. (2) as follows.

$$w_i(k+1) \quad = \quad \max(w_n(k) + \delta(\gamma - d(k)), 0) \quad (3)$$

where δ is a control parameter that determines the amount of increase/decrease of the window size in a round-trip time. The purpose of introducing δ is not only for enabling application of a control theory, but also for improving transient performance. In [6], it has been reported that fairness among connections cannot be satisfied when $d(k)$ lies in $[\alpha, \beta]$. In our analytic model, we therefore unify both α and β in Eq. (2) into a single one, γ, as in Eq. (3). With this modification, fairness among connections can be improved [6]. Intuitively, γ controls the number of on-the-fly packets in the network for each connection.

B. Stability Analysis

For simplicity, we assume that the initial window sizes of all source hosts are equal, and that all source hosts change their window sizes according to Eq. (3). The number of packets in the router's buffer at slot $k+1$, $q(k+1)$, is given by

$$q(k+1) \quad = \quad \min(\max(N\, w(k) - B\, r(k), 0), L), \quad (4)$$

where $w(k) \equiv w_n(k)$ $(1 \le n \le N)$.

Let w^*, q^*, and d^* be the fixed points of $w(k)$, $q(k)$, and $d(k)$, respectively. By using Eqs. (1), (3), and (4), and assuming $r(k) \simeq \tau$ in Eq. (3), w^*, q^*, and d^* can be obtained as follows.

$$w^* \quad = \quad \tau \left(\frac{B + \gamma N}{N} \right) \quad (5)$$

$$q^* \quad = \quad \gamma N \tau \quad (6)$$

$$d^* \quad = \quad \gamma \quad (7)$$

Since $w(k)$ is given by a non-linear equation, we linearize it around the fixed point. Let $\mathbf{x}(k)$ be the difference from the fixed point, which is defined by

$$\mathbf{x}(k) \quad = \quad \begin{bmatrix} w(k) - w^* \\ q(k) - q^* \end{bmatrix}.$$

$\mathbf{x}(k+1)$ is given by

$$\mathbf{x}(k+1) = \mathbf{A}\, \mathbf{x}(k) \quad (8)$$

where

$$\mathbf{A} \quad = \quad \begin{bmatrix} 1 - \frac{\delta}{\tau} + \frac{B\delta}{(B+\gamma N)\tau} & -\frac{B\delta}{N(B+\gamma N)\tau} \\ N & 0 \end{bmatrix}.$$

In the system defined by Eqs. (1), (3), and (4), the fixed point (w^*, q^*) is locally exponentially stable when the roots of the characteristic equation, s_i $(i = 1, 2)$, satisfy $|s_i| < 1$. Note that the characteristic equation is given by

$$D(s) \equiv |s\mathbf{I} - \mathbf{A}| = 0. \quad (9)$$

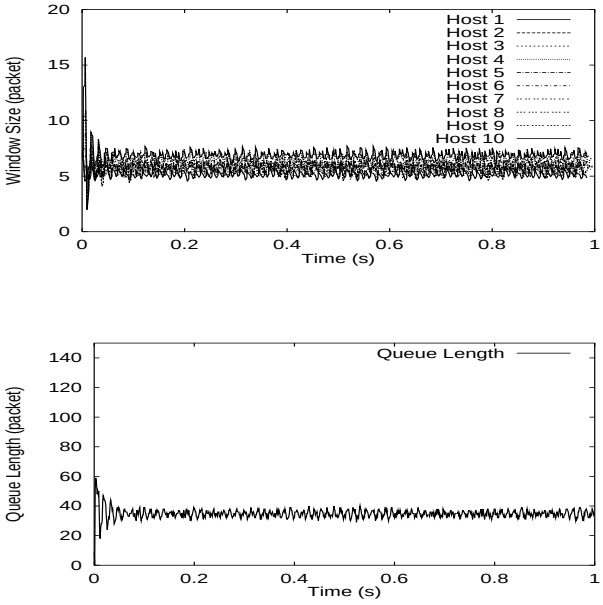

Fig. 2: Stable behavior ($\delta = 2.0$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet).

Since the characteristic equation, $D(s)$, is quadratic, this condition is equivalent to the following inequalities.

$$D(1) > 0; \quad D(-1) > 0; \quad |D(0)| < 1$$

That is, the fixed point of the system, (w^*, q^*), is locally exponentially stable if and only if the following inequalities hold.

$$\delta \quad > \quad 0 \quad (10)$$

$$\frac{\delta(B - \gamma N)}{(B + \gamma N)\tau} + 2 \quad > \quad 0 \quad (11)$$

$$\frac{B\delta}{(B + \gamma N)\tau} \quad < \quad 1 \quad (12)$$

III. Simulation Results

As have been mentioned in [7], one drawback of the window-based flow control mechanism based on TCP Vegas is its incapability to measure the propagation delay exactly. Namely, the congestion avoidance mechanism of TCP Vegas relies on the assumption that the propagation delay (i.e., the round-trip time without any queueing delay at the router) is known in advance. This assumption is valid if all routers in the network have separate output buffers for different connections, or if the offered traffic load is low so that no packets are built up at the router's buffer. However, these are rarely the case in real networks. As we will demonstrate later, inaccurate measurement of the propagation delay causes unfairness among connections.

Another drawback of the window-based flow control mechanism based on TCP Vegas is its lack of scalability regarding the number of connections. Namely, the number of packets in the router's buffer in steady state is given

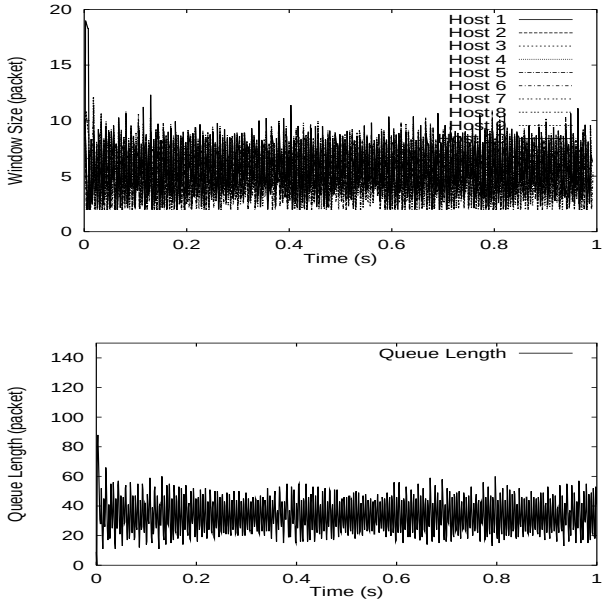

Fig. 3: Unstable behavior ($\delta = 3.0$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet).

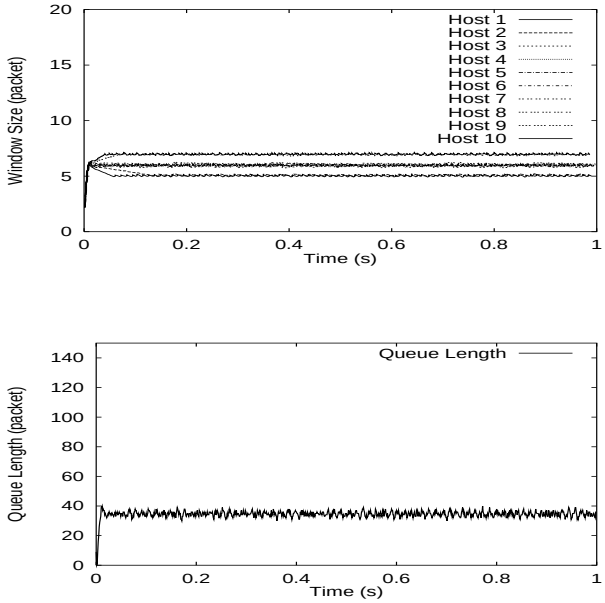

Fig. 4: Case of appropriate δ ($\delta = 0.4$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet).

by Eq. (6). It indicates that the number of packets in the router's buffer linearly increases as the number of connections, N, increases. If the number of connections gets extremely large, the number of packets in the buffer becomes quite large. So it does not operate correctly unless either a great amount of buffer is provided or γ is set to be quite small [5].

In what follows, we present several simulation results for the window-based flow control mechanism based on TCP Vegas. The simulation model is equivalent to the analytic model shown in Fig. 1. We have implemented the window-based flow control mechanism based on TCP Vegas on the ns (Network Simulator) package. The packet size is fixed at 1,000 bytes, the number of connections, N, is 10, and the control parameter δ is changed to 0.4, 2.0, and 3.0. For other control parameters, following parameters are used: the router's bandwidth, B, is set to 20 packet/ms, the propagation delay, τ, is 1 ms, and the control parameter, γ, is 3 packet.

In Figs. 2 and 3, we first show simulation results for $\delta = 2.0$ and $\delta = 3.0$, each of which presents stable and unstable behavior of the window-based flow control mechanism based on TCP Vegas. These figures show dynamical behaviors of the window size of each connection and the number of packets in the router's buffer. One can find that Fig. 2 (stable behavior) shows the slight oscillation of both the window size and the number of packets in the router's buffer. This is caused by the disturbance in measurement of the round-trip time (e.g., variation in processing delays at both TCP and IP layers and timer granularity of the source host). Thus, a smaller value of δ would be appropriate for achieving a better stability. Figure 3 (unstable behavior) shows the drastic oscillation of the window size and the number of packets in the buffer. These figures suggest the validity of our stability analysis presented in Section II-B.

We next show the case of appropriate value of δ in Fig. 4, where δ is set to 0.4. This figure shows the more stable operation than Fig. 2. However, it should be noted that fairness among connections is not fully satisfied. This problem is caused by the drawback of the window-based flow control mechanism based on TCP Vegas as have discussed above — incapability to measure the propagation delay exactly.

This problem becomes more apparent and has a serious impact when source hosts begin their data transmissions at different times. In Fig. 5, each connection is activated every 20 ms. This figure shows that the window size of the source host 10 is stabilized at 22 packets whereas that of the source host 1 at 6 packets, indicating severe unfairness among connections. To illustrate the cause of this problem clearly, the estimated propagation delays of all source hosts are shown in Fig. 6.

One can find that the estimated propagation delays range from 1 ms to 3.7 ms. The reason of the propagation delay of the source host 10 being larger than the source host 1 can be explained as follows. When the source host 1 is activated, there is no packet in the router's buffer. So it is able to measure the propagation delay accurately. However, when the source host 10 starts, approximately 40 packets are queued in the router's buffer. Thus, the propagation delay seen by the source host 10 is the actual propagation delay *plus* the queueing delay for these packets. As our analysis suggests in Eq. (5), the window size in steady state is a linear function of the propagation delay. Hence, the difference in the *estimated* propagation delay directly affects the difference

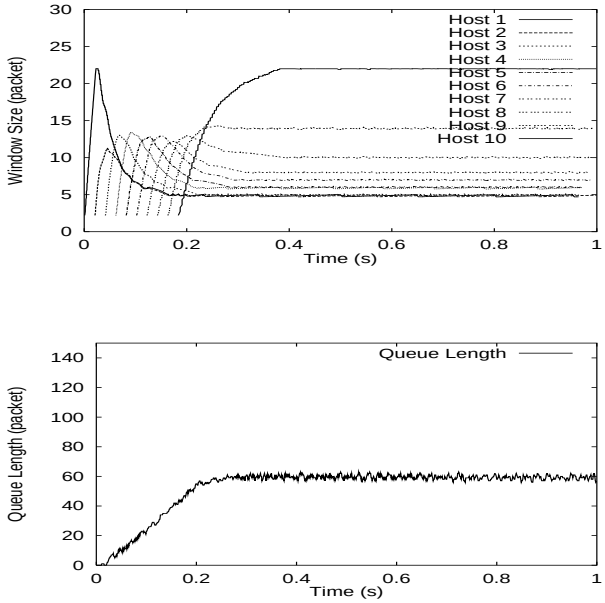

Fig. 5: Case of staggered activation ($\delta = 0.4$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet).

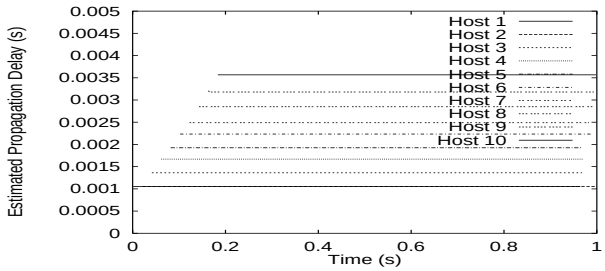

Fig. 6: Case of staggered activation ($\delta = 0.4$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet).

in throughput.

IV. ECN (Explicit Congestion Notification) in TCP Vegas

An ECN (Explicit Congestion Notification) is a mechanism to explicitly notify source hosts of congestion occurrence in the network. Several variants of ECN mechanisms have been used in various congestion control mechanisms [7]. For instance, the DECbit congestion avoidance scheme uses an ECN bit in the header of data packets. In ATM networks, an EFCI (Explicit Forward Congestion Indication) bit in the header of data cells and a CI (Congestion Indication) bit of RM (Resource Management) cells are used. ECN mechanisms can be implemented in TCP/IP networks in several ways. In [8], *ICMP Source Quench message* is defined for conveying congestion information from the congested router to source hosts. One-bit use of the DS-byte in the differentiated service architecture has been proposed in [9].

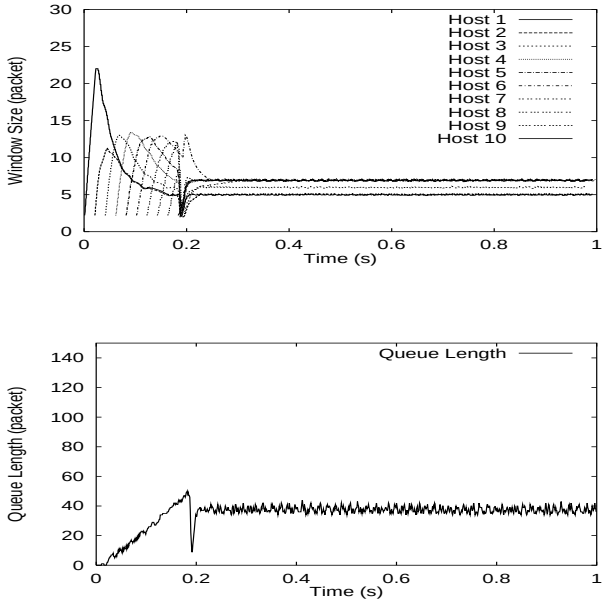

Fig. 7: Case of staggered activation with ECN mechanism ($\delta = 0.4$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet).

According to [9], an example implementation of the ECN mechanism in TCP/IP networks is as follows. A one-bit in the header of the data packet is reserved for the ECN bit. The router in the network uses the ECN bit for notifying source hosts of its incipient congestion. The router computes the average number of packets in the buffer. If it exceeds a threshold value (e.g., p % of the buffer capacity), the router sets the ECN bits of all arriving packets. This information is then carried to source hosts via corresponding destination hosts as the ACK packet with the ECN bit set. The source host responds to the ECN message by, for example, reducing its window size as in the case of packet losses [7]. The advantage of the ECN mechanism is that unnecessary packet losses can be prevented if source hosts respond to the ECN message appropriately. In [7], it has been reported that the ECN mechanism can avoid unnecessary packet delays for low-bandwidth and delay-sensitive TCP connections. It has also been reported that another advantage of the ECN mechanism is that the source host can detect congestion rapidly regardless of coarse granularity of the TCP's timer.

The ECN mechanism has a possibility to solve drawbacks of the window-based flow control mechanism based on TCP Vegas. When the ECN message is received by the source host, it implies that the control algorithm works inappropriately. In this case, the source host should throttle its window size. However, the problem is how much of the window size should be reduced by receipt of the ECN message. One-bit information of the ECN message is apparently insufficient to fine control of the window size. We therefore use the probabilistic number of ECN messages. More specifically, the source host counts the number of received ECN messages,

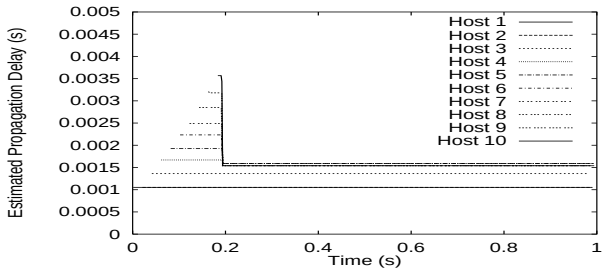

Fig. 8: Case of staggered activation with ECN mechanism ($\delta = 0.4$, $B = 20$ packet/ms, $N = 10$, $\tau = 1$ ms, $\gamma = 3$ packet).

N_e, in the last number N_a of ACK packets. It then computes the ratio of the ECN messages at slot k, $e(k)$, as

$$e(k) \quad = \quad \frac{N_e}{N_a}, \qquad 0 \leq e(k) \leq 1.$$

If $e(k)$ is close to 1, it suggests that the network is heavily congested so that the window size should be reduced quickly. On the contrary, if $e(k)$ is close to 0, the network is lightly congested so that the window size should be reduced slightly. Thus, the controller function of the window-based flow control mechanism given in Eq. (3) is changed as

$$w_i(k+1) \quad = \quad \max(w_n(k) + \delta(\gamma - d(k)) - w_n(k)e(k), 0).$$

Note that the objective of the above controller is to minimize the difference between γ and $d(k)$ and to minimize $e(k)$.

We finally demonstrate how the drawbacks of the window-based flow control mechanism are solved by introduction of the ECN mechanism. Figures 7 and 8 show simulation results when the ECN mechanism is used. In this case, the router marks the ECN bit in the packet header whenever the number of packets in the buffer is over 50 packets. And the source host changes its window size according to the above equation. The number of ACK packets, N_a, which is used to compute the probabilistic number of ECN messages, is set to 10. These figures correspond to Figs. 5 and 6, where the ECN mechanism is not used. It can be found from Fig. 7 that fairness among connections is dramatically improved compared with Fig. 5. Note that performance improvement in fairness is mostly resulted from the probabilistic number of ECN messages, $e(k)$. Let us consider a situation where a source host happens to gain a much larger window size than others due to inaccurate estimation of the propagation delay. In this case, it receives more ECN messages because of our probabilistic approach. It therefore decreases its window size more quickly than others. Consequently, unfairness among connection is considerably relieved.

V. Conclusion

In this paper, we have focused on a window-based flow control mechanism based on the congestion avoidance mechanism of TCP Vegas, and have analyzed its stability using control theory. We have performed simulation experiments and have shown its several drawbacks. To overcome these drawbacks, we have investigated how the ECN mechanism can be incorporated into the window-based flow control mechanism. We have also demonstrated its effectiveness through simulation experiments.

As a future work, we should extend our analysis to more realistic networks. For example, interference between different types of traffic should be considered. Since both of TCP and UDP traffic co-exist in real TCP/IP networks, performance of a congestion control mechanism for TCP traffic must be affected by existence of UDP traffic.

ACKNOWLEDGMENT

This work was supported in part by Research for the Future Program of Japan Society for the Promotion of Science under the Project "Integrated Network Architecture for Advanced Multimedia Application Systems" (JSPS-RFTF97R16301).

REFERENCES

[1] V. Jacobson, "Congestion avdoidance and control," in *Proceedings of SIGCOMM '88*, pp. 314–329, August 1988.

[2] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. New York: Addison-Wesley, 1994.

[3] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of ACM SIGCOMM '94*, pp. 24–35, October 1994.

[4] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1465–1480, October 1995.

[5] H. Ohsaki, M. Murata, T. Ushio, and H. Miyahara, "A control theoretic analysis of a window-based flow control mechanism in TCP/IP networks," submitted to *IEEE INFOCOM 2000*, July 1999.

[6] G. Hasegawa, M. Murata, and H. Miyahara, "Fairness and stability of congestion control mechanism of TCP," in *Proceedings of 11th ITC Special Seminar*, pp. 255–262, October 1998.

[7] S. Floyd, "TCP and explicit congestion notification," *ACM Computer Communication Review*, vol. 24, pp. 10–23, October 1994.

[8] B. Braden and J. Postel, "Requirements for internet gateways," *Request for Comments (RFC) 1009*, June 1987.

[9] S. Kalyanaraman, D. Harrison, S. Arora, K. Wanglee, and G. Guarriello, "A one-bit feedback enhanced differentiated services architecture," *Internet Draft* <draft-shivkuma-ecn-diffserv-01.txt>, March 1998.