Analysis of a Window-Based Flow Control Mechanism based on TCP Vegas in Heterogeneous Network Environment

Keiichi Takagaki † Hiroyuki Ohsaki ‡ Masayuki Murata ‡
†Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka, 560–8531, Japan
Phone +81–6850–6616, Fax +81–6850–6589
E-mail takagaki@ics.es.osaka-u.ac.jp

‡Cybermedia Center, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka, 560–8531, Japan
Phone +81–6850–6587, Fax +81–6850–6589
E-mail: {oosaki, murata}@cmc.osaka-u.ac.jp

Abstract—**Another version of TCP called *TCP Vegas* has been proposed and studied in the literature. It can achieve better performance than the current TCP Reno. In our previous studies, steady-state behavior of a window-based flow control mechanism based on TCP Vegas has been analyzed for a simple network topology. In this paper, we extend our analysis to a generic network topology where multiple bottleneck links exist. We first derive equilibrium values of a window size of a TCP connection and the number of packets waiting in a router's buffer in steady state. We also derive throughput of each TCP connection in steady state, and investigate the effect of control parameters of TCP Vegas on fairness among TCP connections. We then present several numerical examples, showing how control parameters of TCP Vegas should be configured for achieving both stability and better transient performance.**

I. Introduction

Another version of TCP called *TCP Vegas* has been proposed by Brakmo *et al.*, which can achieve better performance than TCP Reno [1], [2]. 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 [2] 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. In [3], it has been reported that TCP Vegas keeps less data in the network than TCP Reno, resulting in shorter end-to-end delays. 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 emission of a packet and receipt of 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. TCP Vegas then controls its window size to make it constant. There is no need for the source host to wait for packet losses to detect occurrence of congestion in the network. The window size of TCP Vegas becomes fixed when the network is in steady state, and therefore it can achieve much better throughput than TCP Reno.

In [4], [5], [6], we have analyzed the dynamics of a window-based flow control mechanism based on the congestion avoidance mechanism of TCP Vegas using simple network topologies. In those papers, we have quantitatively evaluated the effect of several parameters (e.g., the number of TCP connections, control parameters of the window-based flow control mechanism, and the propagation delay of a TCP connection) on the network performance. However, the network model has been limited to a fairly simple one; an existence of only a single bottleneck link has been assumed in our analyses. In reality, the TCP connection may traverse several bottleneck links along its path. In this paper, we relax the limitation of the analysis in [4], [5], [6], and analyze the window-based flow control mechanism based on TCP Vegas for realistic network models.

The window-based flow control mechanism of TCP is a distributed control in a sense that each TCP connection regulates its sending packets without exchanging information with other TCP connections. Thus, it is not an easy task to realize a fair bandwidth allocation among multiple TCP connections. In the first part of this paper, we focus on a fairness issue of the window-based flow control mechanism in a heterogeneous network. With the window-based congestion control mechanism based on TCP Vegas, operation of the network can be stabilized if control parameters are chosen appropriately. We first derive equilibrium values of the window sizes of source hosts and the number of packets in a router's buffer. We then derive the throughput values of TCP connections in steady state, and discuss how to configure control parameters of the window-based flow control mechanism for achieving better fairness among TCP connections in a heterogeneous network. Note that in [7], another approach has been taken to derive the throughput of a TCP connection. In [7], the throughput of the TCP connection is obtained by solving an optimization problem since the congestion avoidance mechanism of TCP Vegas can be viewed as a control that maximizes an objective function. Although the throughput obtained in this paper is identical to that of [7], our analysis has the following advantages: (1) an iterative computation is not necessary to obtain the throughput, and (2) the throughput of a TCP connection can be derived algebraically for rather simple network models.

More importantly, the window-based flow control mechanism of TCP is essentially a closed-loop control, where each TCP connection changes its window size according to implicit feedback information obtained from the network. Hence, an inappropriate choice of control parameters may lead to unstable operation of the network. In general, there exists a trade-off between stability and transient performance. Namely, a setting of control parameters aimed for the best transient performance never achieves the best stability, and vice versa. In the second part of this paper, we analytically show such a trade-off between stability and transient performance. More specifically, a control theoretic approach is utilized to quantitatively show the effect of a choice of control parameters on stability and transient performance. Several numerical examples are also presented to clearly understand how and why an inappropriate setting of control parameters causes severe performance degradation of the network.

II. Steady State Analysis

A. Congestion Avoidance Mechanism of TCP Vegas and Analytic Model

In this paper, all source hosts are assumed to change their window sizes according to the congestion avoidance mechanism of TCP Vegas. We also assume that the buffer size of each router is sufficiently large so that no packet is lost in the network.

The congestion avoidance mechanism of TCP Vegas changes the window size w_c of a TCP connection c once per its RTT.

$$w_c \leftarrow \begin{cases} w_c + 1 & \text{if } d_c < \alpha_c \\ w_c - 1 & \text{if } d_c > \beta_c \\ w_c & \text{otherwise} \end{cases} \quad (1)$$

where d_c indicates severity of congestion in the network, and is computed at the source host as

$$d_c = \left(\frac{w_c}{\tau_c} - \frac{w_c}{r_c} \right) \tau_c \quad (2)$$

Here, τ_c and r_c are the round-trip propagation delay and the observed RTT of the TCP connection, respectively. For instance, d_c is zero if the expected throughput (w_c/τ_c) is same as the actual throughput (w_c/r_c). Otherwise, d_c takes a positive value being proportional to the difference between the actual and the expected throughput values. Two threshold values, α_c and β_c, are another control parameters of TCP Vegas, determining the number of in-flight packets in the network.

In our analysis, the congestion avoidance mechanism of TCP Vegas is used with a slight modification. Namely, Eq. (1) is changed to

$$w_c \quad \leftarrow \quad [w_c + \delta_c(\gamma_c - d_c)]^+ \quad (3)$$

where $[x]^+ \equiv \max(x, 0)$. γ_c is a control parameter that determines the number of in-flight packets in the network. The congestion avoidance mechanism of TCP Vegas does not change its window size when d_c lies in [α_c, β_c]. It has been reported in [8] that fairness among TCP connections is degraded because of this mechanism. In this paper, two threshold values, α_c and β_c, are unified into the single γ_c to prevent unfairness among TCP connections. In what follows, it is assumed that all TCP connections always send the number w_c of packets during their RTTs.

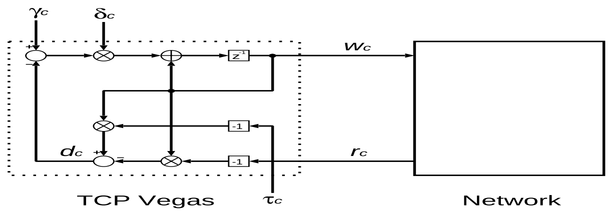

Fig. 1. Block diagram of the window-based flow control mechanism based on TCP Vegas

A block diagram of the window-based flow control mechanism based on TCP Vegas is shown in Fig. 1. Note that boundary conditions regarding w_c in Eq. (3) are not displayed. In this figure, " z^{-1}" represents a delay operator of an RTT, and -1 represents an inversion operator. The left part surrounded by a dotted line corresponds to the window-based flow control mechanism based on TCP Vegas, and the right part surrounded by a solid line corresponds to the network. As can be seen from this figure, the window-based flow control mechanism can be viewed as a SISO (Single-Input and Single-Output) nonlinear feedback-based control. The input is the observed RTT indicating severity of the congestion, and the output is the window size (i.e., the number of packets sent during its RTT).

In this paper, we analyze the window-based flow control mechanism based on TCP Vegas for an arbitrary network topology. For simplicity, we further make several assumptions: (1) all TCP connections traverse the same route if both the ingress and the egress node of those TCP connections are the same, and (2) the routing table is fixed so that a TCP connection's path is predetermined.

Each router is assumed to have separate output buffers for outgoing links, and an output buffer is modeled by a FIFO (First-In First-Out) queue that processes incoming packets in the order of its arrival. We also assume symmetry of the network; that is, the backward path (i.e., from a destination host to a source host) is always identical to the forward path (i.e., from a source host to a destination host). Table I defines several symbols used hereafter.

TABLE I
Definition of Symbols

$\mathcal{N}$	:	set of source/destination hosts and routers
$\mathcal{L}$	:	set of links
μ_l	:	capacity of link l
τ_l	:	propagation delay of link l
q_l	:	the number of packets in router's output buffer destined to link l
$\mathcal{C}$	:	set of TCP connections
$C_i \subset \mathcal{C}$	:	set of TCP connections traversing the same route
$\mathcal{L}(c) \subset \mathcal{L}$	:	set of links that TCP connection c traverses
$\mathcal{C}(l) \subset \mathcal{C}$	:	set of TCP connections traversing link l
$\mathcal{L}(c,l)$	:	set of links that TCP connection c traverses after link l

A destination host sends ACK packet back to the corresponding source host immediately after its receipt of a data packet. Provided that the backward path is never congested, the round-trip propagation delay τ_c is denoted by

$$\tau_c = 2 \sum_{l \in \mathcal{L}(c)} \tau_l$$

where τ_l is the propagation delay of a link l. Let Δ_l ($l \in \mathcal{L}$) be the irreducible positive integer that is a ratio of the propagation delay of the link l:

$$\forall_l \quad \frac{\tau_l}{\Delta_l} = \tau$$

where τ is a constant. In what follows, the network including TCP connections is modeled by a discrete-time system where a time slot is given by τ.

B. Derivation of State Transition Equations

In the analytic model described above, the network state is uniquely defined by w_c (i.e., the window size of a TCP connection c) and q_l (the number of packets in the router's buffer destined to a link l). We derive a set of state transition equations, representing evolutions of w_c and q_l between two adjacent slots. Let us denote $w_c(k)$ and $q_l(k)$ as values of w_c and q_l in slot k, respectively. We also use this notation for all other variables. Provided that the RTT can be approximated by the propagation delay (i.e., $r_c \simeq \tau_c$), the window size w_c is given by

$$w_c(k) = \begin{cases} \left[w_c(k - \frac{\tau_c}{\tau}) + \delta_c(\gamma_c - d_c(k)) \right]^+ \\ \qquad \text{if } k \equiv 0 \pmod{\frac{\tau_c}{\tau}} \\ w_c(k-1) \quad \text{otherwise} \end{cases} \tag{4}$$

where $d_c(k)$ and $r_c(k)$ are defined as

$$d_c(k) = \left(\frac{w_c(k - \frac{\tau_c}{\tau})}{\tau_c} - \frac{w_c(k - \frac{\tau_c}{\tau})}{r_c(k)} \right) \tau_c \tag{5}$$

$$r_c(k) = \tau_c + \sum_{l \in \mathcal{L}(c)} \frac{q_l(k - \frac{1}{2}\frac{\tau_c}{\tau} - \sum_{m \in \mathcal{L}(c,l)} \Delta_m)}{\mu_l} \tag{6}$$

Also the number of packets in the router's buffer $q_l(k)$ is given by

$$q_l(k) = \left[q_l(k-1) + \left(\sum_{c \in \mathcal{C}(l)} A_{c,l}(k-1) - \mu_l \right) \tau \right]^+ \tag{7}$$

where $A_{c,l}(k)$ is a packet arrival rate coming from the TCP connection c at slot k. Namely,

$$A_{c,l}(k) = \begin{cases} \frac{w_c(k)}{r_c(k)} & \text{if } l = l_c \\ \frac{\mu_l A_{c,b(c,l)}(k - \Delta_{b(c,l)})}{\sum_{d \in \mathcal{C}(l)} A_{d,b(d,l)}(k - \Delta_{b(d,l)})} & \text{if } l \neq l_c \text{ and } q_l(k) > 0 \\ A_{c,b(c,l)}(k - \Delta_{b(c,l)}) & \text{if } l \neq l_c \text{ and } q_l(k) = 0 \end{cases}$$

Here, $b(c,l)$ is the previous link to the link l for TCP connection c, and l_c ($\in \mathcal{L}$) is the link to which the source host of TCP connection c is connected.

C. Consideration on Throughput and Fairness

With the window-based flow control mechanism based on TCP Vegas, if control parameters are configured appropriately, the network is stabilized; i.e., window sizes of TCP connections and the numbers of packets in routers buffer converge to fixed values. In this subsection, by assuming a stable operation of the network, equilibrium values of the window size of a TCP connection (denoted by w_c^*) and the number of packets in the router's buffer (denoted by q_l^*) are derived, followed by discussions on the throughput of a TCP connection and fairness among TCP connections. We will derive conditions for stable operation in Section II-D.

From Eqs. (4)–(6), following equations are satisfied in steady state.

$$\gamma_c = d_c^* \tag{8}$$

$$d_c^* = \left(\frac{w_c^*}{\tau_c} - \frac{w_c^*}{r_c^*} \right) \tau_c \tag{9}$$

$$r_c^* = \tau_c + \sum_{l \in \mathcal{L}(c)} \frac{q_l^*}{\mu_l} \tag{10}$$

Let θ_c^* ($\equiv r_c^* - \tau_c$) be the sum of waiting times at all routers for a packet belonging to TCP connection c. Equations (8)–(10) yield

$$\gamma_c = \frac{\theta_c^*}{\tau_c + \theta_c^*} w_c^* \tag{11}$$

Letting ρ_c^* ($\equiv w_c^*/r_c^*$) be the throughput of TCP connection c in steady state, Eq. (11) gives

$$\gamma_c = \rho_c^* \, \theta_c^* \tag{12}$$

This indicates that the throughput of a TCP connection is determined by the control parameter γ_c and the packet waiting time at all routers θ_c^*. Note that this equation is regarded as a Little's law; ρ_c^* is a packet arrival rate from TCP connection c, θ_c^* is a packet waiting time in the network, and γ_c is the number of packets waiting in the network.

Moreover, the following relation is obtained from Eqs. (11) and (12).

$$w_c^* = \rho_c^* \tau_c + \gamma_c$$

This equation clearly shows that the window size of the TCP connection converges to the value given by

$$\text{(bandwidth)} \times \text{(propagation delay)} + \text{(control parameter } \gamma_c) \tag{13}$$

As will be shown below, the control parameter γ_c directly affects fairness among TCP connections. Let us consider two TCP connections, c and c', which traverse the same route. These two connections has the identical packet waiting time (i.e., $\theta_c^* = \theta_{c'}^*$). Hence, using Eq.(12) gives the throughput ratio of these TCP connections as

$$\frac{\rho_c^*}{\rho_{c'}^*} = \frac{\gamma_c}{\gamma_{c'}}$$

which suggests that the throughput ratio of two TCP connections traversing the same path is solely dependent on control parameters γ_c and $\gamma_{c'}$.

On the contrary, if two TCP connections c and c' traverse different routes, the throughput ratio is given by

$$\frac{\rho_c^*}{\rho_{c'}^*} = \frac{\gamma_c \, \theta_{c'}^*}{\gamma_{c'} \, \theta_c^*}$$

It indicates that the throughput of the TCP connection c relatively increases as θ_c^* decreases. From Eq. (10) and the definition of θ_c^*, it can be seen that the throughput of a TCP connection is relatively large, when the TCP connection traverses the small number of bottleneck links or when it traverses a bottleneck link with large capacity. It means that it is difficult to achieve fairness among TCP connections in the heterogenous network. Hence, careful configuration of γ_c is necessary for achieving better fairness.

In steady state, the sum of throughputs of all TCP connections traversing the link l is equal to its link capacity; i.e.,

$$\sum_{c \in \mathcal{C}(l)} \rho_c^* = \mu_l \tag{14}$$

Solving equations given by Eqs. (11) and (14) for all TCP connections and bottleneck links yields equilibrium values of w_c^* and q_l^*. If a network topology is simple, it can be solved algebraically. Otherwise, a numerical computation is necessary.

D. Stability and Transient Behavior

All TCP connections belonging to C_i are assumed to have the same initial window size and to behave identically. Let $\mathbf{x}(k)$ be the difference between the network state at slot k and its equilibrium value:

$$\mathbf{x}(k) \equiv \begin{bmatrix} w_{c_1}(k) & - & w^*_{c_1} \\ & \vdots & \\ w_{c_{|\mathcal{N}|}}(k) & - & w^*_{c_{|\mathcal{N}|}} \\ q_{l_1}(k) & - & q^*_{l_1} \\ & \vdots & \\ q_{l_{|\mathcal{L}|}}(k) & - & q^*_{l_{|\mathcal{L}|}} \end{bmatrix} \qquad (15)$$

where $|\mathcal{N}|$ and c_i are defined as the number of different sets $\mathcal{C}_i$ and a TCP connection belonging to $\mathcal{C}_i$, respectively.

The discrete-time system defined by Eqs. (4) and (7) has non-linearity. So we linearize it around equilibrium values. Let Δ_{LCM} be the LCM (Lowest Common Multiple) of τ_c/τ's of all TCP connections. Assuming that all TCP connections start their packet transmission at slot k, all TCP connections are synchronized every Δ_{LCM} slots. Hence, the state transition between $\mathbf{x}(k)$ and $\mathbf{x}(k + \Delta_{LCM})$ is written as

$$\mathbf{x}(k + \Delta_{LCM}) = \mathbf{A}\,\mathbf{x}(k) \qquad (16)$$

where $\mathbf{A}$ is a state transition matrix. This model is the $(|\mathcal{N}| + |\mathcal{L}|)$ -th order system with no input. So the stability of the network is determined by eigenvalues of the state transition matrix $\mathbf{A}$ [9]. More specifically, the network is stable if all eigenvalues lie in the unit circle in the complex plane (i.e., absolute values are less than one). It is also known that the smaller absolute values of eigenvalues, the better the transient performance becomes [9].

III. Numerical Examples

In this section, several numerical examples are presented to investigate how control parameters affect fairness among TCP connections, stability and transient behavior of the network.

A. Network Model

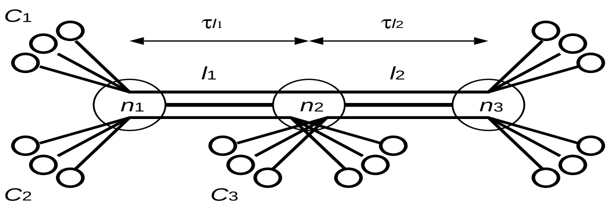

Fig. 2. Network model used in numerical examples

As shown in Fig. 2, a rather simple network model is used in the following numerical examples. The model consists of three routers n_i $(1 \le n \le 3)$ connected in serial. TCP connections are classified into three groups called C_i $(1 \le i \le 3)$ according to their routes. We call a TCP connection belonging to C_i as c_i $(1 \le i \le 3)$ for brevity. Links between routers (i.e., l_1 and l_2) are assumed to be bottleneck links in this model. In other words, link capacities of all other links are assumed to be larger than those of l_1 and l_2. Hence, TCP connection c_1 traverses two bottleneck links, whereas TCP connections c_2 and c_3 do a single bottleneck link. We set $\Delta_{l_1} = \Delta_{l_2} = 1$ but $\Delta_l = 0$ for all other links. So the propagation delay of each TCP connection is proportional to the number of bottleneck links that it traverses.

B. Fairness

Figure 3 shows the throughput of TCP connections. In this figure, the capacity of the link l_1 is fixed to μ_l = 20 [packet/ms] while the capacity of the link l_2 is changed. For other parameters, the following values are used: $|C_i| = 10$ $(1 \le i \le 3)$, $\tau = 1$ [ms], $\gamma_{c_i} = 3$ [ms] $(1 \le i \le 3)$.

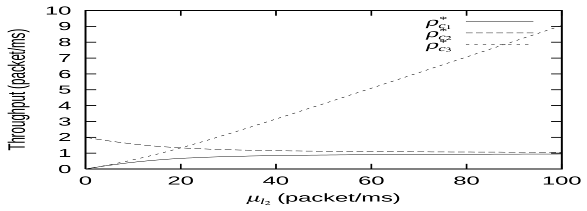

Fig. 3. Throughput values of TCP connections

As can be seen from Fig. 3, when two bottleneck links have the same capacity (i.e., $\mu_{l_2} = 20$), we have a relation

$$\rho^*_{c_2} = \rho^*_{c_3} = 2\rho^*_{c_1} \qquad (17)$$

Namely, the throughput of the TCP connection is inversely proportional to the number of bottleneck links it traverses. On the contrary, when μ_{l_2} is sufficiently large, TCP connections c_1 and c_2 receive the same throughput.

C. Stability and Transient Behavior

We illustrate the stability condition (the condition for all eigenvalues to be in the unit circle) of in Fig. 4. In this figure, two links l_1 and l_2 are given the same capacity (denoted by μ_l) but μ_l is changed from 0.2 to 2,000 [packet/ms]. The network is stabilized when the point $(\delta_{c_1}, \delta_{c_2})$ resides in the region surrounded by the boundary line. TCP connections c_2 and c_3 are given the same value of δ_c. Other parameters are equal to those in obtaining Fig. 3.

Figure 4 shows that the stability region becomes $0 \le \delta_{c_1}, \delta_{c_2} \le 2$ as the link capacity μ_l converges to zero. Namely, when the link capacity is very small, the control parameter δ_c of a TCP connection can be chosen regardless of the number of bottleneck links that it traverses. On the other hand, as the link capacity μ_l becomes large, the upper-bounds of δ_{c_1} and δ_{c_2} for stability converge to 3.5 and 1.0, respectively. It indicates that a larger value δ_c can be assigned to TCP connection traversing more bottleneck links without deteriorating network stability. Note that the propagation delay of a TCP connection as well as the number of bottleneck links is also a key factor in determining the maximum value of δ_c [6]. We note that the stability region of Fig. 4 almost matches that obtained from simulation experiments, which are not included here due to space limitation.

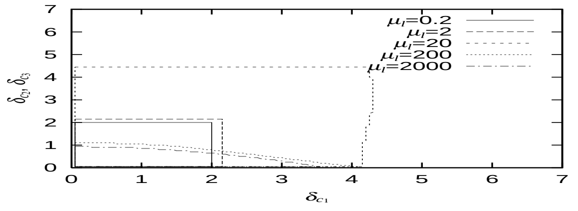

Fig. 4. Stability region in the δ_{c_1}–δ_c plane (Model I)

The effect of changing the control parameter δ_c on the dynamics of the network behavior can be understood by investing movement of eigenvalues of the state transition matrix **A**. Figure 5 shows the trajectories of the eigenvalues λ_i $(1 \leq i \leq 5)$ in the complex plane for $\mu_l = 0.2$ [packet/ms]. In this figure, δ_c $(= \delta_{c_i}, 1 \leq i \leq 3)$ is changed from 0 to 2.1 while other parameters are unchanged from Fig. 4. One can find that three eigenvalues, λ_1, λ_2 and λ_3, go outside of the unit circle as the control parameter δ_c becomes large. On the contrary, other eigenvalues, λ_4 and λ_5, almost stay fixed. Namely, stability of the network is determined only by those three eigenvalues.

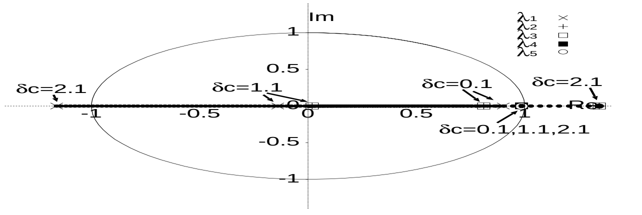

Fig. 5. Trajectories of eigenvalues in the complex plane (μ_l = 0.2 [packet/ms])

Physical meaning of the eigenvalue λ_i can be interpreted through its corresponding eigenvector. Let us explain this with an example. When the link capacity becomes zero (i.e., $\mu_l \rightarrow 0$), eigenvalues λ_i converge to

$$\begin{aligned}&[\lambda_1 \quad \lambda_2 \quad \lambda_3 \quad \lambda_4 \quad \lambda_5] \\ &\quad = \quad [1 - \delta_{c_1} \quad (1 - \delta_{c_2})^2 \quad (1 - \delta_{c_3})^2 \quad 1 \quad 1]\end{aligned}$$

This indicates that the stability condition is $0 \leq \lambda_i \leq 2$ $(1 \leq i \leq 3)$. With a little calculation, one can confirm that eigenvectors for λ_i $(1 \leq i \leq 3)$ have zeros, which corresponds to w_{c_j} $(i \neq j)$. Namely, this means that the control parameter δ_c of a TCP connection has no effect on stability of window sizes of other TCP connections.

We then focus on the case where the link capacity is normal. In Fig. 6, trajectories of eigenvalues λ_i are plotted for $\mu_l = 20$ [packet/ms]. In this figure, δ_c $(= \delta_{c_i}, 1 \leq i \leq 3)$ is changed from 0 to 4.6 while other parameters are unchanged. All eigenvalues λ_i show complex trajectories in this figure. In particular, λ_4 goes left on the real axis, and λ_2 and λ_3 go right along the real axis as δ_c increases. Consequently, when δ_c reaches about 4.5, λ_2, λ_3, and λ_4 goes outside of the unit circle so that the network becomes unstable. Recalling that the smaller absolute values of eigenvalues, the better the transient performance becomes. Thus, it is expected that the transient performance is improved by choosing the control parameter δ_c appropriately.

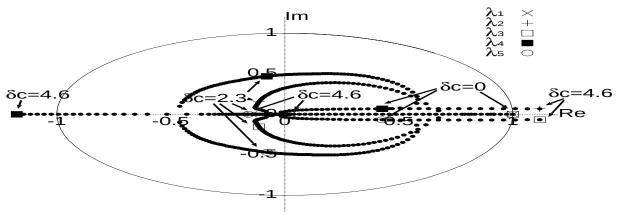

Fig. 6. Trajectories of eigenvalues in the complex plane (μ_l = 20 [packet/ms])

However, when the link capacity μ_l is large, transient performance cannot be improved, which can be explained from Fig. 7. In this figure, trajectories of λ_i are plotted for μ_l = 2000 [packet/ms] and δ_c $(= \delta_{c_i}, 1 \leq i \leq 3)$ are changed from 0.1 to 1.0. One can find from this figure that λ_1 never gets close to the origin. It is also true for other eigenvalues. Hence, the transient performance cannot be improved regardless of the value of the control parameter δ_c.

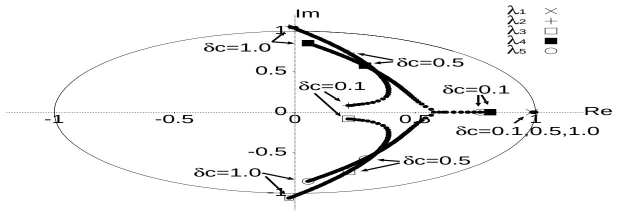

Fig. 7. Trajectories of eigenvalues in the complex plane (μ_l = 2000 [packet/ms])

IV. Conclusion and Future Work

In this paper, we have analyzed the window-based flow control mechanism based on TCP Vegas for realistic network models. We have found that the window-based flow control mechanism based on TCP Vegas has a bias on the number of bottleneck links and the bottleneck link capacity. We have quantitatively show that the control parameter δ_c has an important role to control stability and transient performance of the network. For the network model shown in Fig. 2, we have confirmed validity of our analysis by comparing with several simulation results, which cannot be included in this paper due to space limitation. However, more extensive simulation studies are necessary to validate our analysis for more complex network topologies. One problem of the window-based flow control mechanism based on TCP Vegas is its undesirable transient performance with a large bandwidth–delay product. As a future work, designing a compensator for the window-based flow control mechanism to improve its transient performance would be interesting.

Acknowledgments

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] 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.
[2] 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.
[3] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan, "Evaluation with TCP Vegas: Emulation and experiment," *ACM SIGCOMM Computer Communication Review*, vol. 25, pp. 185–195, August 1995.
[4] H. Ohsaki, M. Murata, T. Ushio, and H. Miyahara, "Stability analysis of window-based flow control mechanism in TCP/IP networks," *1999 IEEE International Conference on Control Applications*, pp. 1603–1606, August 1999.
[5] H. Ohsaki, M. Murata, T. Ushio, and H. Miyahara, "A control theoretical analysis of a window-based flow control mechanism in TCP/IP networks," submitted to *IEEE Transactions on Control Systems Technology*, August 2000.
[6] K. Takagaki, H. Ohsaki, and M. Murata, "Stability analysis of a window-based flow control mechanism for TCP connections with different propagation delays," in *Proceedings of INET 2000: The Internet Global Summit*, July 2000.
[7] L. P. Steven Low and L. Wang, "Understanding TCP Vegas: Theory and practice." available at `http://www.ee.mu.oz.au/staff/slow/research/`.
[8] 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.
[9] G. F. Franklin and J. D. Powell, *Digital Control of Dyanamic Systems*. Addision-Wesley Publishing Company, 1980.