PAPER *Special Issue on Traffic Control and Performance Evaluation in Internet*

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

Keiichi TAKAGAKI[†], **Hiroyuki OHSAKI**[††], ***and*** **Masayuki MURATA**[††], ***Members***

SUMMARY

A feedback-based congestion control mechanism is essential to realize an efficient data transfer services in packed-switched networks. TCP (Transmission Control Protocol) is a feedback-based congestion control mechanism, and has been widely used in the current Internet. An improved 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 previous studies, only performance analysis of a window-based flow control mechanism based on TCP Vegas for a simple network topology has been performed. In this paper, we extend the analysis to a generic network topology where each connection is allowed to have a different propagation delay and to traverse multiple bottleneck links. We first derive equilibrium values of window sizes of TCP connections and the number of packets waiting in a router's buffer. 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.

key words: *TCP (Transmission Control Protocol), TCP Vegas, control theory, fairness, stability, transient performance*

1. Introduction

In 1994, another version of TCP called *TCP Vegas* has been proposed by Brakmo *et al.*, which can achieve better performance than TCP Reno [1]. TCP Vegas has following advantages over TCP Reno: (1) a new timeout mechanism to detect packet losses much sooner than TCP Reno, (2) an improved congestion avoidance mechanism to control the number of in-flight packets within the network, and (3) a modified slow-start mechanism to prevent from sending packets at excessive rate. In particular, the congestion avoidance mechanism of TCP Vegas controls the number of in-flight packets in the network. A heart of TCP Vegas is its congestion avoidance mechanism; TCP Vegas measures an RTT (Round Trip Time), which is elapsed time from a packet transmission to the receipt of its corresponding ACK (ACKnowledgment) packet. It then uses the measured RTT as feedback information from the network. The simulation and experimental results show that the congestion control mechanism of TCP Vegas leads to 37–71 % higher throughput than that of TCP Reno [1].

In the litereture, there has been several simulation or experimental studies of TCP Vegas [2]–[4]. Also there has been analytic studies of TCP Vegas [5]–[8]. In those analytical studies, the evolution of a window size is approximated by a fluid model, and the throughput of each TCP Vegas connection is derived. However, those analytical studies use a very simplified network model: a single TCP connection [5], [6], two TCP connections [7], or homogeneous TCP connections [8]. Therefore, those results are not directly applicable to a real network. In addition, stability of TCP Vegas has not been fully investigated. Since TCP Vegas is essentially a feedback-based congestion control, a stable operation of the control mechanism is very important.

One exception can be found in [9], where the authors explicitly derive the stability condition and the optimal setting of control parameters by applying a control theory. Control parameters achieving the best transient performance are also derived. Further, the analytic model considered in [9] allows multiple TCP connections. However, the authors assume a homogeneous network environment where all TCP connections have identical propagation delays. Also, they assume a single bottleneck link in the network. In this paper, we analyze the network where each connection has a different propagation delay and traverses multiple bottleneck links by extending the previous work [9].

Besides stability and transient performance of the network, fairness among TCP connections is also an important issue. In the first part of this paper, We focus on a fairness issue of the window-based flow control mechanism in the heterogeneous network. We first derive equilibrium values of the window sizes of source hosts and the number of packets in the 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 the heterogeneous network. Note that in [10], [11], another approach has been taken to derive the throughput of a TCP connection. Although the throughput obtained in this paper is identical to that of [10], [11], our analysis has the following advantages: (1) an iterative computation is not necessary to

[†]The author is with the Matsushita Electric Industrial Co., Ltd, Kadoma, Osaka 571–8501 Japan

[††]The author is with the Cybermedia Center, Osaka University, Ibaraki, Osaka 558–8585 Japan

obtain the throughput, and (2) the throughput of the TCP connection can be derived algebraically for rather simple network models.

Next, we derive the stability condition of the window-based flow control mechanism by applying the control theory. Then, We quantitatively show how the stability region of control parameters is affected by network parameters such as the processing speed of the router and the propagation delay. Analyses of feedback-based congestion control mechanisms using control theoretic approaches can be found in the literature. In particular, several papers including [12]–[14] have analyzed a feedback-based congestion control mechanism for the network model where each connection has a different propagation delay. However, their approaches are not applicable to the window-based flow control mechanism since the behaviors of a rate-based and a window-based congestion control mechanisms are essentially different.

Organization of this paper is as follows. In Section 2, we explain a congestion control mechanism of TCP Vegas, and introduce our analytic model. In Section 3, we derive state transition equations, and focus on a fairness issue. Stability and transient analyses are also presented. In Section 4, we show numerical examples for the case of a single bottleneck link, where each TCP connection has a different propagation delay. In Section 5, we show numerical examples for the case of multiple bottleneck links. In Section 6, we conclude this paper and discuss future works.

2. Analytic Model

2.1 TCP Vegas

In this subsection, we explain the congestion avoidance mechanism of TCP Vegas. For detailed explanation, refer to [15]. In TCP Vegas, each source host maintains τ, which is the minimum RTT of connection obtained when the network is not congested. The minimum RTT τ corresponds to the sum of all propagation delays and processing delays at the routers. Hereafter, we call the minimum RTT, τ, as the *propagation delay.* The source host is allowed to emit packets of its current window-size (denoted by w) per RTT. Therefore, its effective throughput would be w/τ if there is no congestion in the network. Each source host obtains the actual RTT by measuring time duration between a transmission time of a packet and an arrival of its corresponding ACK packet. Let r be the actual RTT measured at the source host, and $\overline{w}$ be the number of packets the source host sent in the previous RTT. Its actual throughput is given by $\overline{w}/r$. TCP Vegas then computes the difference between expected throughput and actual throughput as $d = w/\tau - \overline{w}/r$. TCP Vegas changes its window size, w, according to relations among d and two threshold values, α and β, as follows.

$$w \leftarrow \begin{cases} w+1, & \text{if } d < \alpha \\ w-1, & \text{if } d > \beta \\ w, & \text{otherwise} \end{cases} \tag{1}$$

2.2 Network 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. In what follows, parameters regarding a TCP connection c is denoted by a subscript c. For example, the window size of a TCP connection c is denoted by w_c. In our analysis, the congestion avoidance mechanism of TCP Vegas is used with a slight modification. Namely, Eq. (1) is changed to

$$w_c \leftarrow [w_c + \delta_c(\gamma_c - d_c)]^+ \tag{2}$$

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 $[\alpha_c, \beta_c]$. It has been reported in [16] 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.

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 arrivals. 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).

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 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: ${}^\forall l \quad \tau_l \, \Delta_l = \tau$, 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.

3. Analysis

3.1 Derivation of State Transition Equations

In the analytic model described in Section 2, 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 convention 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

$$w_c(k) = \begin{cases} \left[w_c(k - \frac{\tau_c}{\tau}) + \delta_c(\gamma_c - d_c(k))\right]^+ \\ \qquad\qquad \text{if } k \equiv 0 \pmod{\frac{\tau_c}{\tau}} \\ w_c(k-1) \quad \text{otherwise} \end{cases} \tag{3}$$

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

$$d_c(k) = \left(\frac{w_c(k - \frac{\tau_c}{\tau})}{\tau_c} - \frac{w_c(k - \frac{\tau_c}{\tau})}{r_c(k)}\right)\tau_c \tag{4}$$

$$r_c(k) = \tau_c + \sum_{l \in \mathcal{L}(c)} \frac{q_l(k - \frac{1}{2}\frac{\tau_c}{\tau} - \sum_{m \in \mathcal{L}(c,l)} \Delta_m)}{\mu_l} \tag{5}$$

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

$$q_l(k) = \left[q_l(k-1) + \left(\sum_{c \in \mathcal{C}(l)} A_{c,l}(k-1) - \mu_l\right)\tau\right]^+ \tag{6}$$

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. However, we have a problem in the above formulation. The approximation error in Eq. (6) becomes quite large when the packet waiting time in the router's buffer is much larger than the propagation delay. If there is only a single bottleneck in the network, more rigorous equation can be obtained [17].

3.2 Fairness

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 the TCP connection and fairness among TCP connections. We will derive conditions for stable operation in Subsection 3.3.

From Eqs. (3)–(5), following equations are satisfied in steady state.

$$\gamma_c = d_c^* \tag{7}$$

$$d_c^* = \left(\frac{w_c^*}{\tau_c} - \frac{w_c^*}{r_c^*}\right)\tau_c \tag{8}$$

$$r_c^* = \tau_c + \sum_{l \in \mathcal{L}(c)} \frac{q_l^*}{\mu_l} \tag{9}$$

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 (7)–(9) yield

$$\gamma_c = \frac{\theta_c^*}{\tau_c + \theta_c^*} w_c^* \tag{10}$$

Letting ρ_c^* ($\equiv w_c^*/r_c^*$) be the throughput of TCP connection c in steady state, Eq. (10) gives

$$\gamma_c = \rho_c^* \, \theta_c^* \tag{11}$$

It indicates that the throughput of the TCP connection is determined by the control parameter γ_c and the packet waiting time at all routers θ_c^*. Note that the above 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. (10) and (11).

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

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

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

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.(11) gives the throughput ratio of these TCP connections as $\rho_c^*/\rho_{c'}^* = \gamma_c/\gamma_{c'}$, 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 $\rho_c^*/\rho_{c'}^* = \gamma_c \, \theta_{c'}^*/(\gamma_{c'} \, \theta_c^*)$. It indicates that the throughput of the TCP connection c is relatively improved as θ_c^* decreases. From Eq. (9) 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 Section 5.2, we will discuss it in more detail with numerical examples.

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

$$\sum_{c \in \mathcal{C}(l)} \rho_c^* = \mu_l \tag{12}$$

Solving equations given by Eqs. (10) and (12) 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.

3.3 Stability and Transient Behavior

All TCP connections belonging to the same set of connections traversing the same route (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:

$$\mathbf{x}(k) \equiv \begin{bmatrix} w_{c_1}(k) & - & w_{c_1}^* \\ & \vdots & \\ w_{c_{|\mathcal{N}|}}(k) & - & w_{c_{|\mathcal{N}|}}^* \\ q_{l_1}(k) & - & q_{l_1}^* \\ & \vdots & \\ q_{l_{|\mathcal{L}|}}(k) & - & q_{l_{|\mathcal{L}|}}^* \end{bmatrix} \tag{13}$$

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. (3) and (6) 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

$$\mathbf{x}(k+\Delta_{LCM}) = \mathbf{A}\,\mathbf{x}(k) \tag{14}$$

where $\mathbf{A}$ is a state transition matrix. This model is the $(|\mathcal{N}|+|\mathcal{L}|)$ -th order system without input. So the stability of the network is determined by eigenvalues of the state transition matrix $\mathbf{A}$ [18]. More specifically, the network is stable if all eigenvalues λ_i $(1 \le i \le |\mathcal{N}|+|\mathcal{L}|)$ 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 [18]. It is well known that transient performance of the system is heavily affected by the maximum value of λ_i. The maximum value of λ_i denoted by $|\lambda|$ is given as follows.

$$|\lambda| \equiv \max_{1 \le i \le |\mathcal{N}|+|\mathcal{L}|} (|\lambda_i|) \tag{15}$$

4. Case of Single Bottleneck Link

4.1 Network Configuration

In this subsection, we show numerical examples for a network configuration shown in Fig. 1. TCP connections are divided into two groups called $\mathcal{C}_{sh}$ and $\mathcal{C}_{lo}$ according to their routes. For brevity, we call a TCP connection belonging to $\mathcal{C}_{sh}$ and $\mathcal{C}_{lo}$ as c_{sh} and c_{lo}, respectively. The link between router n_1 and n_2 (i.e., l_B) are assumed to be the bottleneck link in this model. In other words, link capacities of all other links are assumed to be larger than that of l_B. Also, the propagation delays of links l_{sh} and l_{lo} (i.e., $\tau_{l_{sh}}$ and $\tau_{l_{lo}}$) are the half of the propagation delay of TCP connections $\tau_{c_{sh}}$ and $\tau_{c_{lo}}$, respectively. In other words, propagation delays of other links are assumed to be negligible. Without loss of generality, we assume $\tau_{l_{sh}} \le \tau_{l_{lo}}$. In this configuration, two types of TCP connections with different propagation delays share the bottleneck link and the ratio of propagation delays is $\Delta_{l_{sh}}$:$\Delta_{l_{lo}}$.

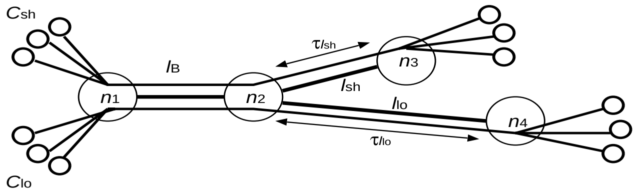

Fig. 1 Network configuration in the case of single bottleneck link.

4.2 Stability

Figure 2 shows boundary lines of the stability region on the $\delta_{c_{sh}} - \delta_{c_{lo}}$ plane for several values of the link capacity μ_{l_B}. In this figure, the following parameters are used: the number of connections $|\mathcal{C}_{sh}| = |\mathcal{C}_{lo}| = 10$, the propagation delay $\tau_{c_{sh}} = 1$ [ms] and $\tau_{c_{lo}} = 2$ [ms], and the control parameter $\gamma_{c_{sh}} = \gamma_{c_{lo}} = 3$ [packet]. The link capacity μ_{l_B} is changed from 2 to 2,000 [packet/ms]. This figure suggests that the system is stable when the point $(\delta_{c_{sh}},\ \delta_{c_{lo}})$ lies inside the boundary line. In other words, we should choose the point $(\delta_{c_{sh}},\ \delta_{c_{lo}})$ in the region surrounded by the boundary line and x- and y-axes for stable operation.

We can find from the figure that the stability region heavily depends on the link capacity μ_{l_B}. We can

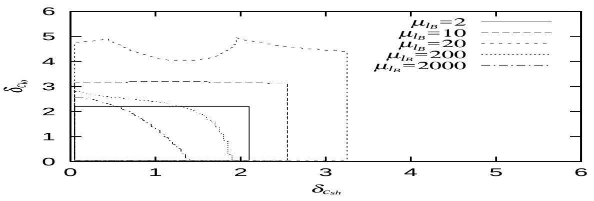

Fig. 2 Stability region in the $\delta_{c_{sh}}$–$\delta_{c_{lo}}$ plane for different link capacity μ_{l_B}.

also find that the maximum value of $\delta_{c_{lo}}$ for connections with a large propagation delay is larger than the maximum value of $\delta_{c_{sh}}$ for connections with a smaller propagation delay. This tendency becomes more noticeable as the link capacity μ_{l_B} becomes large. For example, when $\mu_{l_B} = 2,000$ [packet/ms], the maximum values of $\delta_{c_{sh}}$ and $\delta_{c_{lo}}$ for stable operation are about 1.3 and 2.6, respectively.

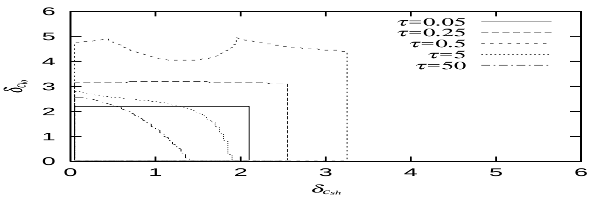

Fig. 3 Stability region in the $\delta_{c_{sh}}$–$\delta_{c_{lo}}$ plane for different propagation delays τ.

Figure 3 shows the stability region for different values of propagation delays τ from 0.05 [ms] to 50 [ms]. In this figure, values of control parameters and system parameters are equal to those in Fig. 2, whereas the processing speed of the router μ_{l_B} is fixed at 20 [packet/ms]. By comparing Figs. 2 and 3, one can find that boundary lines in these figures are almost identical. Such a correspondence can be easily explained from Eqs. (3)–(6). Namely, all μ_l's and τ_c's in these equations take the product form of $\mu_l \times \tau_c$.

We next investigate how a choice of control parameters and/or system parameters affects stability and transient performance of the network. After investigating the stability region ($\delta_{c_{sh}}$,$\delta_{c_{lo}}$) for a number of parameter sets on μ_{l_B}, γ_c, and τ_c, we have found that the stability region is the same if the following value is identical

$$F_c \equiv \frac{|\mathcal{C}|\,\gamma_c}{\mu_{l_B}\,\tau} \tag{16}$$

F_c has the following meaning. As Eq. (2) indicates, the source host adjusts its window size to transmit extra γ_c packets into the network per its RTT. In the network configuration used in this section, since extra γ_c packets of all connections are queued in the single router's buffer, the number of packets in the router's buffer is given by the following equation.

$$q^*_{l_B} = \sum_{c\in\mathcal{C}(l_B)} \gamma_c = |\mathcal{C}_{sh}|\gamma_{c_{sh}} + |\mathcal{C}_{lo}|\gamma_{c_{lo}}$$

Since the packet waiting time in the router's buffer is given by q^*/μ_{l_B}, F_c is thought of as the packet waiting time for a TCP connection, divided by its propagation delay. Namely, F_c can be seen as an index representing the amount of the packet waiting time in the router's buffer compared to the RTT. In what follows, we therefore discuss based on the value of F_c.

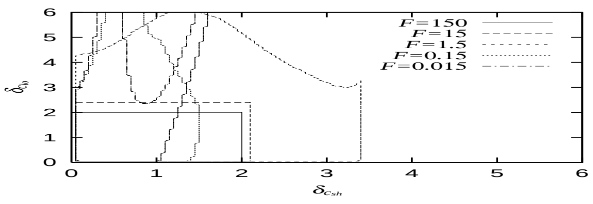

Fig. 4 Stability region in the $\delta_{c_{sh}}$–$\delta_{c_{lo}}$ plane for propagation delay for $\Delta_{l_{sh}}/\Delta_{l_{lo}} = 1/4$.

In Figs. 4 and 5, we show stability regions for $\Delta_{l_{sh}}/\Delta_{l_{lo}} = 1/4$ and for $\Delta_{l_{sh}}/\Delta_{l_{lo}} = 2/3$, respectively. In these figures, F_c is changed from 0.015 to 150 while satisfying the relation $F_{c_{sh}} = F_{c_{lo}}$. Recalling that $\Delta_{l_{sh}} < \Delta_{l_{lo}}$, these figures suggest that when F_c is small, the maximum value of $\delta_{c_{lo}}$ is larger than that of $\delta_{c_{sh}}$. When F_c is small, the RTT observed by a source host is mostly determined by the propagation delay. So the maximum value of $\delta_{c_{lo}}$ is larger than the maximum value of $\delta_{c_{sh}}$. On the other hand, when F_c is large, the maximum value of δ_c achieving stability seems to be independent of the difference in propagation delays of TCP connections. Figures 4 and 5 indicate, for large F_c, the stability region is $0 < \delta_{c_{sh}}, \delta_{c_{lo}} < 2$. From these observations, when F_c is large (i.e., the packet waiting time is larger), the network can always be stabilized with, for example, $(\delta_{c_{sh}}, \delta_{c_{lo}}) = (1, 1)$. However, when F_c is small (i.e., the packet waiting time is smaller), δ_c must be chosen carefully according to the difference in propagation delays.

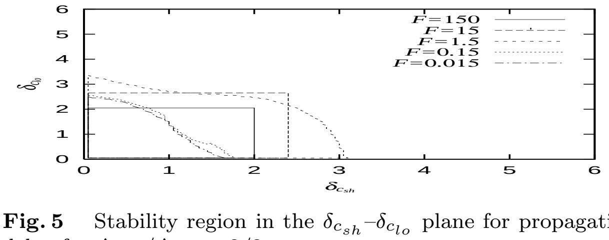

Fig. 5 Stability region in the $\delta_{c_{sh}}$–$\delta_{c_{lo}}$ plane for propagation delay for $\Delta_{l_{sh}}/\Delta_{l_{lo}} = 2/3$.

4.3 Transient Behavior

For an efficient operation of the window-based flow control mechanism, control parameters should be configured by taking account of transient performance, as well as stability. In what follow, we therefore investigate how control parameters are chosen to optimize the transient performance of TCP connections while preserving the stability of the network. Namely, we will derive the optimal values of δ_c and γ_c to minimize the settling-time: i.e., the time taken for the window size of the source host and the number of packets in the router's buffer to converge.

As discussed in Section 3.3, transient performance of the network is mostly determined by $|\lambda|$ as indicated by Eq.(15). Note that the actual transient performance is affected not only by the value of $|\lambda|$ but also by the length of $\tau\Delta_{LCM}$ since the state changes per $\tau\Delta_{LCM}$ in Eq. (13). So, when $|\lambda|$ is same, the ramp-up time is not same for a different $\tau\Delta_{LCM}$.

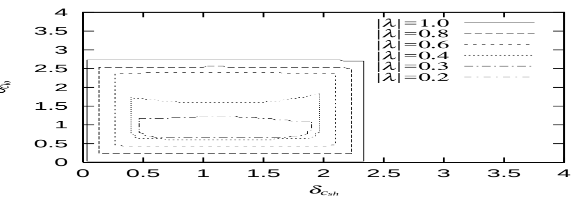

Fig. 6 $|\lambda|$ in the $\delta_{c_{sh}}$–$\delta_{c_{lo}}$ plane for $F_{c_{sh}} = F_{c_{lo}} = 4.5$, $\Delta_{l_{sh}}/\Delta_{l_{lo}} = 1/2$.

Figure 6 shows lines for different values of $|\lambda|$ in the $\delta_{c_{sh}}$–$\delta_{c_{lo}}$ plane when the ratio of propagation delays is 1:2. The following parameters are used: the bottleneck bandwidth $\mu_{l_B} = 2$ [packet/ms], the number of TCP connections $N_{c_{sh}} = N_{c_{lo}} = 3$, the propagation delay $\tau = 1$ [ms], the control parameter $\gamma_{c_{sh}} = \gamma_{c_{lo}} = 3$ [packet]. The value of F_c is $F_{c_{sh}} = F_{c_{lo}} = 4.5$. This figure indicates that, when the point ($\delta_{c_{sh}}$, $\delta_{c_{lo}}$) is outside of the line $|\lambda| = 1.0$, the network is unstable. Also the transient performance is almost optimal when inside the line $|\lambda| = 0.2$.

We next show the value of $|\lambda|$ with a small F_c in the $\delta_{c_{sh}}$–$\delta_{c_{lo}}$ plane. Figure 7 uses the same parameters with Fig. 6, but the bottleneck bandwidth μ_{l_B} and the propagation delay τ are changed to $\mu_{l_B} = 200$ [packet/ms] and $\tau_c = 10$ [ms], respectively. The F_c is $F_{c_{sh}} = F_{c_{lo}} = 0.0045$ in this case. One can find that, regardless of a choice of ($\delta_{c_{sh}}$, $\delta_{c_{lo}}$), $|\lambda|$ is always larger than 0.99. In Fig. 7, the optimal parameter set is about $(\delta_{c_{sh}}, \delta_{c_{lo}}) = (0.6, 1.7)$. This result suggests that a larger F_c is desirable to improve the transient performance.

Although the number of TCP connections $|\mathcal{C}|$, the bottleneck bandwidth μ_{l_B}, and the propagation delay

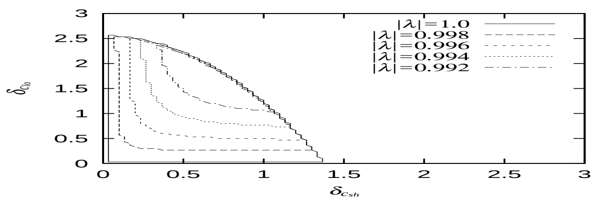

Fig. 7 $|\lambda|$ in the $\delta_{c_{sh}}$–$\delta_{c_{lo}}$ plane for $F_{c_{sh}} = F_{c_{lo}} = 0.0045$, $\Delta_{l_{sh}}/\Delta_{l_{lo}} = 1/2$.

τ are uncontrollable parameters, the control parameter γ_c can be chosen freely at the source host. Hence, a larger γ_c is required to achieve a better transient performance. As suggested by Eq. (17), the number of packets queued in the router's buffer is proportional to γ_c. In other words, there is a trade-off between the packet waiting time in the router's buffer and the transient performance. As we have discussed in Section 3.3, if the control parameter γ_c can be set to a sufficiently large value, the network can be stabilized with $0 < \delta_c < 2$. However, because of some reason such as the limited buffer size of the bottleneck router, γ_c sometimes may not take a large value. In such cases, δ_c should be configured carefully by taking account of the difference in propagation delays of TCP connections.

5. Case of Multiple Bottleneck Links

5.1 Network Configuration

As shown in Fig. 8, a rather simple network model with multiple bottleneck links is used in the following numerical examples. The model consists of three routers n_i $(1 \leq n \leq 3)$ connected in serial. TCP connections are classified into three groups called C_i $(1 \leq i \leq 3)$ according to their paths. We call a TCP connection belonging to C_i as c_i $(1 \leq i \leq 3)$ 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.

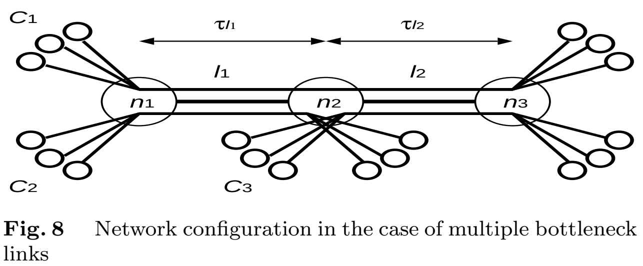

Fig. 8 Network configuration in the case of multiple bottleneck links

5.2 Throughput and Fairness

Figure 9 shows the throughput values of TCP connections. In this figure, the capacity of the link l_1 is fixed at $\mu_l = 20$ [packet/ms] while the capacity of the link l_2 is changed. For other parameters, the following values are used: $|C_i| = 10$ $(1 \leq i \leq 3)$, $\tau = 1$ [ms], $\gamma_{c_i} = 3$ [ms] $(1 \leq i \leq 3)$. As can be seen from Fig. 9, when two bottleneck links have the same capacity (i.e., $\mu_{l_2} = 20$), we have a relation: $\rho^*_{c_2} = \rho^*_{c_3} = 2\rho^*_{c_1}$. 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.

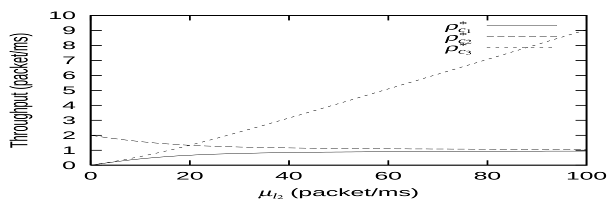

Fig. 9 Throughput values of TCP connections for different link capacities μ_l.

5.3 Stability and Transient Performance

We show the stability region in Fig. 10. 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 of Fig. 9. Figure 10 shows that the stability region becomes $0 \leq \delta_{c_1}, \delta_{c_2} \leq 2$ 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 the TCP connection as well as the number of bottleneck links is also a key factor in determining the Section 4.2

As described in Section 3.3, stability and transient performance of the network is determined by the eigenvalues λ_i. Although λ, the maximum of λ_i, has the significant impact on stability and transient performance, investigation of all eigenvalues λ_i gives us more information to understand the window-based flow control mechanism based on TCP Vegas. In what follows, we investigate how each eigenvalue is affected by the control parameter δ_c. Figure 11 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. 10. 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.

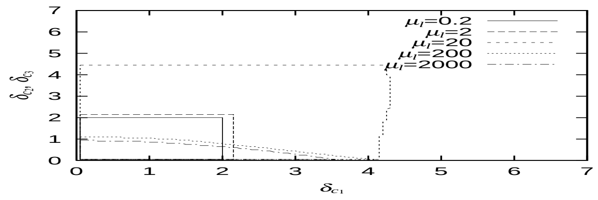

Fig. 10 Stability region in the δ_{c1}–δ_c plane for the case of multiple bottleneck links

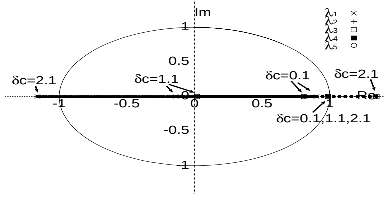

Fig. 11 Trajectories of eigenvalues in the complex plane ($\mu_l = 0.2$ [packet/ms])

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., $\mu_l \to 0$), eigenvalues λ_i converge to

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

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. 12, trajectories of eigenvalues λ_i's 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's show complex trajectories in this figure. In particular, λ_4 go 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 go outside of the unit circle so that the network becomes unstable. Recall 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.

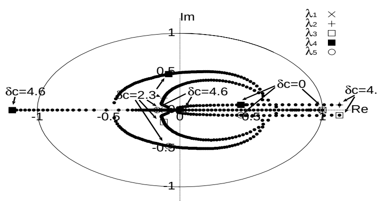

Fig. 12 Trajectories of eigenvalues in the complex plane ($\mu_l = 20$ [packet/ms])

However, when the link capacity μ_l is large, transient performance cannot be improved. It can be explained from Fig. 13. In this figure, trajectories of λ_i are plotted for $\mu_l = 2000$ [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.

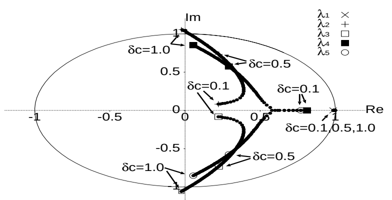

Fig. 13 Trajectories of eigenvalues in the complex plane ($\mu_l = 2000$ [packet/ms])

6. Conclusion

In this paper, we have focused on the window-based flow control mechanism based on the congestion avoidance mechanism of TCP Vegas. Its behavior in steady state has been analyzed by applying a control theory. A network model considered in this paper consists of TCP connections with different propagation delays and multiple bottleneck links. We have first derived the equilibrium values of the window size and the number of packets in the bottleneck router's buffer. We have shown, in the case of a single bottleneck link, that a fair bandwidth allocation to all connections can be realized by setting γ_c's of all connections identically, regardless of the difference in their propagation delays. We have also shown 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 shown that the control parameter δ_c has an important role to control stability and transient performance of the network. If the TCP connection, which has a small propagation delay or traverses less bottleneck links, change its window size excessively (i.e., δ_c is set large), the network becomes unstable. We have found that stability condition and transient performance of the network are heavily dependent on a ratio of the propagation delay and the RTT. We have also shown how to choose the control parameter γ_c for achieving better transient performance. 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.

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] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan, "Evaluation of TCP Vegas: emulation and experiment," *ACM SIGCOMM Computer Communication Review*, vol. 25, pp. 185–195, August 1995.

[3] P. Danzig, Z. Liu, and L. Yan, "An evaluation of tcp vegas by live emulation," 1995.

[4] U. Hengartner, J. Bolliger, and T. Gross, "TCP vegas revisited," in *INFOCOM (3)*, pp. 1546–1555, 2000.

[5] O. A. Hellal and E. Altman, "Analysis of TCP Vegas and TCP Reno," in *Proceedings of IEEE ICC '97*, June 1997.

[6] A. Kumar, "Comperative performance analysis of versions of TCP in a local network with a lossy link," *IEEE/ACM Transactions on Networking*, vol. 6, pp. 485–498, August 1998.

[7] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and TCP Vegas," in *Proceedings of IEEE INFOCOM '99*, March 1999.

[8] T. Bonald, "Comparison of TCP Reno and TCP Vegas via fluid approximation," novemver 1998.

[9] 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.

[10] L. W. S. H. Low, Larry Peterson, "Understanding TCP Vegas: A duality model," june 2001.

[11] L. P. Steven Low and L. Wang, "Understanding TCP Vegas: Theory and practice," February 2000. available at `http://www.ee.mu.oz.au/staff/slow/research/`.

[12] B.-K. Kim and C. Thompson, "Optimal feedback control of ABR traffic in ATM networks," in *Proceedings of IEEE GLOBECOM '98*, pp. 844–848, November 1998.

[13] H. Zhang and O. W. Yang, "Design of robust congestion controllers for ATM networks," in *Proceedings of IEEE INFOCOM '97*, April 1997.

[14] A. Kolarov and G. Ramamurthy, "A control theoretic approach to the design of closed-loop rate based flow control for high speed ATM networks," in *Proceedings of IEEE INFOCOM '97*, pp. 293–301, April 1997.

[15] 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.

[16] 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.

[17] Keiichi Takagaki, Hiroyuki Ohsaki and Masayuki Murata, "Analysis of a window-based flow control mechanism based on TCP Vegas in heterogeneous network environment," to appear in *Proceedings of ICC2001*, June 2001.

[18] G. F. Franklin and J. D. Powell, *Digital Control of Dyanamic Systems*. Addision-Wesley Publishing Company, 1980.