TCP Symbiosis: Congestion Control Mechanisms of TCP based on Lotka-Volterra Competition Model

Go Hasegawa
Cybermedia Center
Osaka University
1-32, Machikaneyama-cho, Toyonaka, Osaka 560-0043, JAPAN
Email: hasegawa@cmc.osaka-u.ac.jp

Masayuki Murata
Graduate School of Infomation Science and Technology
Osaka University
1-3, Yamadaoka, Suita, Osaka 560-0871, JAPAN
Email: murata@ist.osaka-u.ac.jp

Abstract— **In this paper, we propose TCP Symbiosis, which has a robust, self-adaptive and scalable congestion control mechanism for TCP. Our method is quite different from existing approaches. We change the window size of a TCP connection in response to information of the physical and available bandwidths of the end-to-end network path. The bandwidth information is obtained by an inline network measurement technique we have previously developed. Using the bandwidth information we can resolve the inherent problems in existing AIMD/MIMD-based algorithms such as periodic packet loss and unfairness caused by the difference in RTT. We borrow algorithms from biophysics to update the window size: the logistic growth model and the Lotka-Volterra competition model. This is because these models describe changes in the population size of a species that depends on the living environment. The population of a species can be viewed as the window size of a TCP connection and the living environment as the bandwidth of the bottleneck link. The greatest advantage of using these models is that we can refer to previous discussions and results for various characteristics of the mathematical models, including scalability, convergence, fairness and stability in these models. Through mathematical analysis and extensive simulation experiments, we compare the proposed mechanism with traditional TCP Reno, HighSpeed TCP, Scalable TCP and FAST TCP, and exhibit its effectiveness in terms of scalability to the network bandwidth and delay, convergence time, fairness among competing connections, and stability.**

I. Introduction

With increases in the heterogeneity and the complexity of the Internet, many problems have emerged in TCP Reno's congestion control mechanism ([1-3] for example). The primary reasons for these problems are that the congestion signals are only indicated by packet loss and that TCP Reno uses fixed Additive-Increase-Multiplicative-Decrease (AIMD) parameter values to increase and decrease window size, whereas those parameters should be changed according to the network conditions. Although many solutions have been proposed for there problems [4-6], most of them inherit the fundamental congestion control mechanism of TCP Reno: the AIMD mechanism triggered by the detection of packet losses in the network. The congestion control mechanism improves the throughput by adjusting the increasing and decreasing parameters statically and/or dynamically. However, most previous studies have focused on changing the AIMD parameters to accommodate particular network environments. Since these methods employ ad hoc modifications for a certain network situation, their performance when applied to other network environments is unclear.

Because window size indicates the maximum amount of packets that TCP can transmit for one Round Trip Time (RTT), an adequate window size for a TCP connection is equal to the product of the available bandwidth and the round-trip propagation delay between the sender and receiver hosts. TCP Reno measures the RTTs of the network path between sender and receiver hosts by checking the departure times of the data packets and the arrival times of the corresponding ACK packets. However, TCP Reno does not have an effective mechanism to recognize the available bandwidth. This explains the fundamental problem: TCP Reno cannot converge its window size to an adequate value when the network environment varies. In a sense, traditional TCP Reno can be considered to be a tool that measures available bandwidth because of its ability to adjust the congestion window size to achieve a transmission rate appropriate to the available bandwidth. However, it is ineffective because it only increases the window size until packet loss occurs. In other words, TCP Reno induces packet loss in order to obtain information about the available bandwidth(-delay product) of the network. That is, even when the congestion control mechanism of TCP works perfectly, the TCP sender experiences packet losses in the network at some intervals. Since all modified versions of TCP using the AIMD policy, including the generalized AIMD algorithm [7] and its variants, contain this essential problem, they cannot avoid packet losses in the network even if they behave ideally.

There are some TCP variants, including TCP Vegas [8] and FAST TCP [6], that utilize the RTT values for the congestion indication, based on the fact that the RTTs for a TCP connection usually increase before packet losses occur when the network is congested. However, such RTT-based approaches cannot be applied to high-speed networks due to an inherent problem, i.e., changes in RTT values of the end-to-end network path becomes invisible as the network bandwidth becomes large. We believe, therefore, that if a TCP sender recognizes the bandwidth information of the network path quickly and adequately, it can create a better mechanism for congestion control in TCP.

Although numerous measurement tools that measure the physical and available bandwidths of network paths have been proposed in the literature [9-14], we cannot directly employ these existing methods in TCP mechanisms, primarily because these methods utilize a lot of test probe packets. Moreover, these methods also require too much time to obtain one measurement result. Accordingly, we have proposed a method called Inline measurement TCP (ImTCP) that avoids these problems in [15, 16]. It does not inject extra traffic into the network, and instead it estimates the physical/available bandwidths of the network path from data/ACK packets transmitted by an active TCP connection in an inline fashion. Furthermore, since the ImTCP sender obtains bandwidth information every 1–4 RTTs, it is well able to follow the traffic fluctuation of the underlying IP network. We believe that, by directly measuring bandwidth information, the congestion control in TCP becomes truly *scalable* to the bandwidth delay product of the network. AIMD- and MIMD-based mechanisms such as HighSpeed TCP (HSTCP) [4] and Scalable TCP (STCP) [5] are more scalable than TCP Reno, but they have serious problems in parameter tuning. Since no knowledge of the bandwidth information is obtained, the control parameters are configured based on implicit/explicit assumptions about the network environment. For example, in [4], the recommended control parameters are to fill the network link with 10 Gbps bandwidth, 100 msec RTT, and a packet loss rate of 10^{-7}. One of the advantages of the proposed mechanism is that it is not necessary to configure the control parameters according to the network environment. In addition, because ImTCP is implemented at the bottom of the TCP layer, this measurement mechanism can be included in various types of TCP congestion control mechanisms.

In this work, we propose a new congestion control mechanism for TCP, which we call TCP Symbiosis, that utilizes the information of physical and available bandwidths obtained from an inline measurement technique. The proposed mechanism does not use ad hoc algorithms such as TCP Vegas and instead employs existing algorithms, which enable us to mathematically discuss and guarantee their behavior even though posing a simplification of the target system. More importantly, it becomes possible to give a reasonable explanation for our control parameter selections within TCP, instead of conducting intensive computer simulations and/or choosing parameters in an ad hoc fashion. We have designed a window size control algorithm whose purpose is to quickly adjust the window size to an adequate value based on bandwidth information in order to fairly distribute bandwidth among competing connections.

For this, we borrowed algorithms from the logistic growth model and the Lotka-Volterra competition model [17], both of which are used in biophysics to describe changes in the population of species. The biophysics models were chosen based on their intrinsic stability and robustness, which is achieved even when they behave without any interaction in an autonomous and distributed fashion. This is the case for the congestion control of TCP: each TCP connection behaves independently, but still we want to improve the bandwidth utilization and the throughput of the connection. When applying the logistic growth and Lotka-Volterra competition models to the congestion control algorithm of our TCP, the population of a species can be viewed as the window size of a TCP connection, the carrying capacity of the environment as the physical bandwidth, and interspecific competition among species as bandwidth sharing among competing TCP connections.

In the present paper, an analytic investigation of the proposed algorithm is performed based on previously reported discussions and results regarding various biophysical characteristics of the mathematical models, including scalability, convergence, fairness and stability. Endowing TCP with these characteristics is the primary objective of the present study. Furthermore, a performance analysis for the situation where the traditional TCP Reno and TCP Symbiosys share the bottleneck is present to show the fairness property of the proposed mechanism against the existing TCP version. We also present extensive simulation results in order to evaluate the proposed mechanism and show that, compared with traditional TCP Reno and other TCP variants for high-speed networks, the proposed mechanism utilizes network bandwidth effectively, quickly, and fairly.

II. Lotka-Volterra Model and Application to TCP Congestion Control Mechanisms

In this paper, we intend to build a robust self-adaptive congestion control mechanism for TCP. In this sense, the proposed method is quite different from existing approaches. The concept of the window updating algorithm of the proposed method is borrowed from a biological system, which is often pointed out to be robust [18], because in many biological systems, the actions of the entity (e.g., living organism) are not determined based on the results of direct interactions among entities, but rather on information obtained through the environment, which is a fundamental necessary condition for the system to be robust. The concept is often called “stigmergy” in the literature (see, e.g., [19]). With respect to the current case, the window increase/decrease strategy is determined based on the physical and available bandwidth, rather than on the packet loss or RTTs, which are direct consequences of the activities of the TCP connections.

A. Brief Introduction to the Lotka-Volterra Model

1) Logistic Model

The logistic equation is a formula that represents the evolution of the population of a single species over time. Generally, the per capita birth rate of a species increases as the population of the species becomes larger. However, since there are various restrictions on living environments, the environment has a carrying capacity (total population size), which is usually determined by the available sustaining resources. The logistic equation describes such changes in the population of a species as follows [17]:

$$\frac{d}{dt}N = \epsilon\left(1 - \frac{N}{K}\right)N \tag{1}$$

where t is time, N is the population of the species, K is the carrying capacity (total population size) of the environment, and ϵ is the intrinsic growth rate of the species ($0 < \epsilon$).

2) Lotka-Volterra Competition Model

The Lotka-Volterra competition model is a well known model for examining the population growth of two or more species that are engaged in interspecific competition. In the model, Equation (1) is modified to include the effects of both interspecific competition and intraspecific competition. The basic two-species Lotka-Volterra competition model with both species N_1 and N_2 having logistic growth in the absence of the other is comprised of the following equations [17]:

$$\frac{d}{dt}N_1 = \epsilon_1\left(1 - \frac{N_1 + \gamma_{12} \cdot N_2}{K_1}\right)N_1 \tag{2}$$

$$\frac{d}{dt}N_2 = \epsilon_2\left(1 - \frac{N_2 + \gamma_{21} \cdot N_1}{K_2}\right)N_2 \tag{3}$$

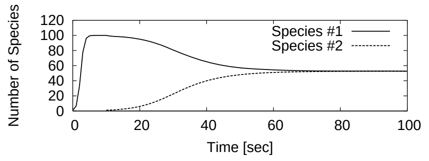


Fig. 1. Changes in population of two species with the Lotka-Volterra competition model

where N_i, K_i, and ϵ_i are the population of the species, the carrying capacity of the environment, and the intrinsic growth rate of the species i, respectively. In addition, γ_{ij} is the ratio of the competition coefficient of species i with respect of species j.

In this model, the population of species 1 and 2 does not always converge to a value larger than 0, and in some cases one species becomes extinct, depending on the values of γ_{12} and γ_{21}. Commonly, the following equations are sufficient conditions for the two species to survive in the environment [17]:

$$\gamma_{12} < \frac{K_1}{K_2}, \quad \gamma_{21} < \frac{K_2}{K_1} \tag{4}$$

Assuming that the two species have the same characteristics, they have the same values: $K = K_1 = K_2$, $\epsilon = \epsilon_1 = \epsilon_2$, and $\gamma = \gamma_1 = \gamma_2$. Then, Equations (2) and (3) can be written as follows:

$$\frac{d}{dt}N_1 = \epsilon\left(1 - \frac{N_1 + \gamma \cdot N_2}{K}\right)N_1 \tag{5}$$

$$\frac{d}{dt}N_2 = \epsilon\left(1 - \frac{N_2 + \gamma \cdot N_1}{K}\right)N_2 \tag{6}$$

In addition, Equation (4) can be written as $\gamma < 1$. Figure 1 shows the population changes in the two species using Equations (5) and (6), where K = 100, ϵ = 1.95 and γ = 0.90, and species 2 joins the environment 10 seconds after species 1. From the figure, we can observe from this figure that the population of the two species converges quickly to the same value.

We can easily extend Equations (5) and (6) for n species as follows:

$$\frac{d}{dt}N_i = \epsilon\left(1 - \frac{N_i + \gamma \cdot \sum_{j=1,i\neq j}^{n} N_j}{K}\right)N_i \tag{7}$$

Note that survival and convergence conditions are identical, i.e., $\gamma < 1$. Even when two or more species exist, each independently utilizes Equation (7) to obtain N_i, and the population of the species can converge to the value equally shared among competing species. We consider that the changing population trends of species depicted in Figure 1 is ideal for controlling the transmission speed of TCP. That is, by using Equation (7) for the congestion control algorithm of TCP, rapid and stable link utilization can be realized, whereas each TCP connection can behave independently as an autonomous distributed system. However, this model cannot be directly applied to the congestion control algorithm of TCP because the model must obtain N_j. This is discussed in the next subsection.

B. Application to Window Size Control Algorithm

To convert Equation (7) to a window increase/decrease algorithm, we consider N_i as the transmission rate of TCP sender i and K as the physical bandwidth of the bottleneck link. Furthermore, when applying Equation (7) to the congestion control algorithm for connection i, it is necessary for connection i to know the data transmission rates of all other connections that share the same bottleneck link. This assumption is quite unrealistic with respect to the current Internet. Therefore, we use the sum of the data transmission rates of all of the other connections using the physical and available bandwidths as follows:

$$\sum_{j=1,i\neq j}^{n} N_j = K - A_i$$

where A_i is the available bandwidth for connections i. Thus, Equation (7) becomes:

$$\frac{d}{dt}N_i = \epsilon\left(1 - \frac{N_i + \gamma \cdot (K - A_i)}{K}\right)N_i \tag{8}$$

Here we assume that all connections share the same bottleneck link K in the equation. Note that when each TCP connection has a different physical bandwidth, the proposed mechanism share the bottleneck link bandwidth in a reasonable manner, which we will discuss in Subsection IV-E.

The proposed mechanism requires modifications only with respect to sender-side TCP, and no change in receiver-side TCP is required. A TCP sender controls its data transmission rate by changing its window size. To retain the essential characteristics of TCP and decrease the implementation overhead, we employ window-based congestion control in the proposed TCP by converting Equation (8) to obtain an increasing algorithm of window size in TCP. The window size of connection i, w_i, is calculated from N_i, the transmission rate, using the following equation:

$$w_i = N_i \tau_i$$

where τ_i is the minimum value of the RTTs of connection i, which is assumed to equal the propagation delay without a queuing delay in the intermediate routers between sender and receiver hosts. Next, Equation (8) can be rewritten as follows:

$$\frac{d}{dt}w_i = \epsilon\left(1 - \frac{w_i + \gamma(K - A_i)\tau_i}{K\tau_i}\right)w_i \tag{9}$$

Finally, we integrate Equation (9) as follows:

$$w_i(t) = \frac{w_i(0)e^{\epsilon t\left\{1-\gamma\left(1-\frac{A_i}{K}\right)\right\}}\{K-\gamma(K-A_i)\}\tau_i}{w_i(0)\left(e^{\epsilon t\left\{1-\gamma\left(1-\frac{A_i}{K}\right)\right\}}-1\right)+\{K-\gamma(K-A_i)\}\tau_i} \tag{10}$$

In Equation (10), when we set the initial value of the window size ($w_i(0)$) and the current time to 0 (t = 0), we can directly obtain window size $w_i(t)$ for any time t. We use the above equation for the control algorithm of the window size of TCP connections.

Equation (10) requires measurement of the physical and available bandwidths of a network path. Therefore, we utilize the inline network measurement technique in ImTCP [15, 16]. In [15, 16], the authors proposed ImTCP, which is an inline network measurement technique for the physical and available bandwidths of network paths between TCP sender and receiver hosts. ImTCP can continuously measure bandwidth by using data and ACK packets of a TCP connection under data transmission. That is, the TCP sender transmits data packets at intervals determined by an inline measurement algorithm and checks the arrival interval times of the corresponding ACK packets to estimate bandwidth. Since ImTCP performs the measurement without transmitting additional probe packets over the network, the effect on other network traffic is negligible. ImTCP can also quickly update the latest changes in bandwidths by frequently performing measurements (one result per 1–4 RTTs) as long as TCP transmits data packets. The authors have also proposed an implementation design of ImTCP, in which the measurement program is located at the bottom of the TCP layer. The proposed implementation design maintains the transmission/arrival intervals of TCP data/ACK packets by introducing a FIFO buffer between the TCP and IP layers. Note that the measurement algorithm has limited effect on TCP's congestion control algorithm [15], meaning that the measurement algorithm can be applied to any TCP variant including our method proposed in this paper.

Note that the inline network measurement algorithm can estimate both of the physical and available bandwidths based on the assumption that the narrowest link on the physical bandwidth of the end-to-end network path becomes the tightest link on the available bandwidth. According to the algorithm in [16], when such an assumption is not satisfied, that is, when the narrowest link and the tightest link are different in the path, the physical bandwidth cannot be measured exactly, whereas the available bandwidth can be obtained successfully. However, in that case, since the physical bandwidth is likely to be underestimated, this measurement error does not cause a serious problem for the proposed congestion control mechanism, because underestimation of the physical bandwidth does not result in injecting too many packets into the network.

III. Characteristics of Proposed Mechanism

In this section, we analyze various characteristics of the proposed mechanism, such as scalability, convergence, parameter setting issues and fairness against TCP Reno. This analysis illustrates that the proposed mechanism essentially solves the problems inherent in TCP Reno.

A. Convergence Time and Scalability

In this subsection, we assume that the physical bandwidth K and available bandwidth A are constant, which means that the utilization of the bottleneck link are stable. In the proposed mechanism, the window size then converges to a certain value in the proposed mechanism. The converged window size, which is denoted as w^*, can be obtained by setting dw/dt = 0 in Equation (9):

$$w^* = \{(1-\gamma)K + \gamma A\}\tau \tag{11}$$

where τ is the round-trip propagation delay the TCP connection. In what follows, we consider the time which is required to increase the window size from w_0 to $\rho \cdot w^*$ ($0 < \rho < 1$, $w_0 < \rho w^*$). In the proposed mechanism, using Equation (10), the time T becomes as follows:

$$\begin{aligned} T &= \frac{1}{\epsilon\left\{1-\gamma\left(1-\frac{A}{K}\right)\right\}} ln\left(\frac{\rho}{1-\rho}\frac{w^*-w_0}{w_0}\right) \\ &\leq \frac{1}{\epsilon(1-\gamma)} ln\left(\frac{\rho}{1-\rho}\frac{w^*-w_0}{w_0}\right) \\ &= \frac{1}{\epsilon(1-\gamma)} ln\left(\frac{\rho}{1-\rho}\frac{((1-\gamma)K+\gamma A)\tau - w_0}{w_0}\right) \end{aligned} \tag{12}$$

because $0 \leq A \leq K$ is satisfied. Note that ϵ and γ are fixed parameters of the proposed mechanism. The issue of setting these parameters will be discussed in the next subsection. This equation indicates that time T of the proposed mechanism increases logarithmically with respect to link bandwidth (K) and propagation delay (τ).

In the case of TCP Reno, we can easily calculate T_{reno}, the time necessary to increase window size from w_0 to w^*, as follows:

$$T_{\mathrm{reno}} = (w^* - w_0)\bar{\tau} = [\{(1-\gamma)K + \gamma A\}\tau - w_0]\bar{\tau} \tag{13}$$

where $\bar{\tau}$ is the average value of the RTTs of the TCP connection. Here, we ignore the effect of the delayed ACK option [20] and focus only on the congestion avoidance phase of TCP Reno. In the case of HSTCP, which is essentially based on the AIMD policy as in the case of TCP Reno, T_{hstcp} is given by:

$$T_{\mathrm{hstcp}} \geq \frac{w^* - w_0}{a_{\max}}\bar{\tau} = \frac{\{(1-\gamma)K + \gamma A\}\tau - w_0}{a_{\max}}\bar{\tau} \tag{14}$$

where $a_{\max}$ is a parameter of HSTCP that indicates the maximum window size increase during one RTT (equivalent to $a(W)$ in [4]). Equations (13) and (14) indicate that the time required to increase the window size is proportional to physical bandwidth K and propagation delay τ. This illustrates that the time required to fully utilize the bandwidth-delay product of the network path is proportional to the bandwidth-delay product. HSTCP was designed as a new congestion control mechanism to resolve problems inherent in TCP Reno for high-speed and long delay networks. However, since the window size control algorithm of HSTCP is essentially based on the AIMD policy, this algorithm suffers from poor scalability to the bandwidth-delay product.

STCP has a window size control algorithm based on Multiplicative Increase Multiplicative Decrease (MIMD) policy and describes logarithmic increases in time with respect to increases in link bandwidth [5]. We calculate its convergence time T_{stcp} as follows:

$$T_{\mathrm{stcp}} = \frac{1}{a}\left(ln\frac{w^*}{w_0}\right)\bar{\tau} = \frac{1}{a}\left(ln\frac{\{(1-\gamma)K+\gamma A\}\tau}{w_0}\right)\bar{\tau} \tag{15}$$

where a is an STCP parameter that indicates the increase in window size when receiving one ACK packet. In [5], a = 0.01 [packet] is the default value. This equation indicates that STCP has good scalability to network bandwidth: however, STCP has poor scalability to propagation delay.

FAST TCP has the same equilibrium properties as TCP Vegas, and the window size is updated at intervals based on the RTT [6]. This means that FAST TCP does not have good scalability to the propagation delay of the end-to-end network path, as will be shown in Section IV.

B. Stability and Fairness

In this Subsection, we utilize the microeconomics analogy as in [21] and its followers to discuss the stability and fairness property of the proposed congestion control mechanism.

We consider a single link with the following link cost function:

$$C(x) = \frac{1}{2}px^2 \tag{16}$$

for possible constant p, which implicitly represents the parameter of Active Queue Management (AQM) at the link buffer. x is the total traffic arrival rate to the link. By the definition of the available bandwidth, we have;

$$K - A_i = \sum_{k \neq i} \frac{w_k}{\tau_k} \tag{17}$$

By using the above equation, we can re-write Equation (9) as follows:

$$\frac{d}{dt}w_i = \frac{\epsilon}{K}\left(K - (1-\gamma)\frac{w_i}{\tau_i} - \gamma\sum_k \frac{w_k}{\tau_k}\right)w_i$$

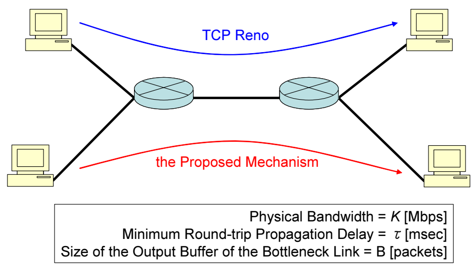

TCP Reno
the Proposed Mechanism
Physical Bandwidth = K [Mbps]
Minimum Round-trip Propagation Delay = τ [msec]
Size of the Output Buffer of the Bottleneck Link = B [packets]

Fig. 2. Model used for fairness analysis

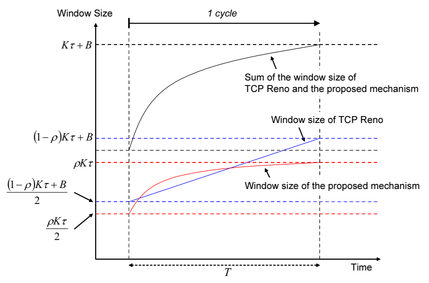

Window Size
1 cycle
$K\tau+B$
$(1-\rho)K\tau+B$
$\rho K\tau$
$\frac{(1-\rho)K\tau+B}{2}$
$\frac{\rho K\tau}{2}$
Sum of the window size of TCP Reno and the proposed mechanism
Window size of TCP Reno
Window size of the proposed mechanism
T
Time

Fig. 3. Changes in the window sizes of TCP Reno and the proposed mechanism

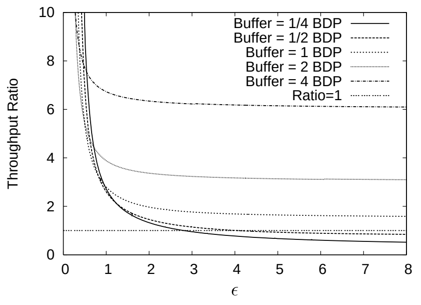


Fig. 4. Ratio of throughput for various buffer sizes

We define the sending rate of connection i as $x_i = w_i/\tau_i$, and assume τ_i is constant. Then we can derive the dynamics of x_i as follows:

$$\frac{d}{dt}x_i = \frac{\epsilon\gamma x_i}{Kp}\left(\frac{p}{\gamma}(K-(1-\gamma)x_i) - p\sum_k x_k\right) \tag{18}$$

We here introduce the following function $U_i(x)$:

$$U_i(x) = \frac{p}{\gamma}\left(K - \frac{1}{2}(1-\gamma)x\right)x$$

Then we can obtain the following eqation from Equation (18):

$$\frac{d}{dt}x_i = \frac{\epsilon\gamma x_i}{Kp}(U_i'(x_i) - C'(\sum_k x_k))$$

From this equation, we can regard the proposed mechanism as the source with the utility function $U_i(x)$ and link cost function $C(x)$ under the prevailing assumptions. We first note that $U_i(x)$ is independent of the RTT τ_i, which means that there is no RTT bias in the proposed mechanism. We confirm this characteristics in Section IV.

The next remark is that $U_i(x)$ depends on the parameters K (physical capacity of the link), p (parameter pof link cost), and γ. For K, since we assume that each TCP connection obtains the value of physical capacity by using ImTCP, we can say that the proposed mechanism provides fair performance when the same value of K is obtained by competing connections. The effect of different values of K is discussed in Subsection IV-E. For γ, which is the control parameter of the proposed mechanism, we show the setting guideline of γ in the next Subsection, and show that the value of γ would affect on the buffering behavior of the bottleneck link, not on the behavior of each connections. This means that the source of the proposed mechanism should use the identical value for γ, which does not affect on the fairness property. For p, on the other hand, it means that we have a problem in the proposed mechanism that its behavior depends on the AQM parameter. Modifying the algorithm to remove this dependency is one of the future research topic of this work.

We also note that since $U_i''(x) = -p(1-\gamma)/\gamma$ and $\gamma < 1$, the utility function $U_i(x)$ is strictly concave. This guarantees that the social welfare problem as follows has a unique solution:

$$\begin{aligned} &SYSTEM \\ &\text{maximize } \textstyle\sum_i U_i(x_i) - C(\sum_k x_k) \\ &\qquad\text{over} \quad x_i \geq 0, \text{ for all } i \end{aligned}$$

This characteristics clearly shows the stability of the proposed congestion control mechanism.

C. Parameter Settings

The congestion control algorithm of the proposed mechanism has two parameters, γ and ϵ. In this subsection, we discuss the effect of these parameters and present some guidelines for configuring γ and ϵ.

1) γ Setting

The parameter γ indicates the degree of the influence of the other competing connections that share the same bottleneck link. To converge window size to a positive value despite the physical bandwidth K_i of each connection, it is necessary to satisfy the condition $0 < \gamma < 1$. Furthermore, based on Equations (11) and (12), we need to consider the trade-off between convergence speed and the final number of packets accumulated within the buffer at the bottleneck link. That is, although smaller γ leads to faster convergence speed, it increases the queue size of the bottleneck router buffer when the window size is converged. Using Equation (11) we can easily obtain the sum of the window size of n TCP connections as follows:

$$\sum_{i=1}^{n} w_i = \frac{n}{1+(n-1)\gamma}K\tau \tag{19}$$

where we assume that the physical bandwidth K and the delay τ of each connection are identical. From Equation (19) queue size Q at the bottleneck link is given by:

$$Q = \frac{(n-1)(1-\gamma)}{1+(n-1)\gamma}K\tau \tag{20}$$

This equation shows that Q increases as n becomes larger. However, as n goes to infinity, we can obtain the following equation:

$$\lim_{n\to\infty} Q = \frac{1-\gamma}{\gamma}K\tau \tag{21}$$

That is, there exists an upper bound of the queue size with respect to an increase in the number of concurrent TCP connections. Therefore, if the bottleneck link has a large enough buffer, the proposed mechanism will induce no packet losses regardless of the number of TCP connections. TCP Reno, HSTCP, and STCP, on the other hand, increase their window size until they fully utilize the buffer at the bottleneck link, and as a result, they cannot avoid periodic packet losses.

2) ϵ Setting

ϵ determines convergence speed, as shown in Equation (9). Generally, when we convert Equation (1) into a discrete equation, the population of the species does not converge with $\epsilon \geq 2$ [17]. In contrast, the window size updating algorithm proposed in Subsection II-B converts Equation (10) into a discrete equation in such a way that it does not cause oscillation. Therefore, in the proposed algorithm, there is no limitation on ϵ, which means that as ϵ becomes larger, the window size converges faster. However, an excessively large value of ϵ causes the TCP sender to transmit numerous packets in bursty fashion, which may reduce the network performance.

D. Competition with TCP Reno

In this subsection, we investigate the fairness property of the proposed mechanism with respect to competing TCP Reno connections. For this purpose, we compare the throughput of two TCP connections which TCP Reno and the proposed mechanism share a bottleneck link, by analyzing changes in congestion window sizes. Figure 2 depicts the network model for analysis, where K is the physical bandwidth, τ is the minimum round-trip propagation delay, not including the queuing delay, and B is the size of the output buffer adopting a TailDrop scheme, of the bottleneck link.

As explained above, the proposed mechanism converges its window size to a certain value whereas TCP Reno continues to increase its window size until a packet loss occurs. Hence, even when both TCP connections compete at the bottleneck link bandwidth, periodic packet loss occurs at the buffer. We, therefore, assume that both TCP connections experience packet loss when the buffer becomes fully utilized. Therefore, the window size of the two TCP connections changes cyclically, triggered by packet loss. Figure 3 describes such changes in the window size. Here, we define one cycle as the period between two packet losses and denote the length of the cycle as T. We assume that the received socket buffer of each TCP connection is large enough not to limit the congestion window size evolution.

In this analysis, we assume that the sender of the proposed mechanism can obtain precise physical bandwidth information. From Figure 3, by using ρ $(0 < \rho < 1)$, the window size of the proposed mechanism just before packet loss occurs is represented as $\rho K\tau$. Since the sum of the window size of both connections is $K\tau + B$ when the buffer becomes full, the window size of TCP Reno connection at that time can be described as $(1-\rho)K\tau + B$. Then, the window size of the proposed mechanism immediately after packet loss occurs becomes decreased to $\rho K\tau/2$, and that of TCP Reno becomes $((1-\rho)K\tau + B)/2$. Since TCP Reno increases its window size by one packet every RTT, T, which is the duration time of one cycle, can be calculated as follows:

$$T = \frac{(1-\rho)K\tau + B}{2}\bar{\tau} \tag{22}$$

where $\bar{\tau}$ is the average value of the RTTs of the TCP connection. the window size of the proposed mechanism can be obtained from Equation (10) by substituting K for A as follows:

$$w(t) = \frac{w(0)e^{\epsilon t}K\tau}{w(0)(e^{\epsilon t}-1) + K\tau} \tag{23}$$

From Equations (12) and (23), we can calculate T, which is equal to the time required for the window size to increase from $\rho K\tau/2$ to $\rho K\tau$, as follows:

$$T = \frac{1}{\epsilon}ln\left(\frac{\rho}{1-\rho}\frac{K\tau - \rho K\tau/2}{\rho K\tau/2}\right) = \frac{1}{\epsilon}ln\left(\frac{2-\rho}{1-\rho}\right) \tag{24}$$

From Equations (22) and (24), we obtain the following equation:

$$\frac{(1-\rho)K\tau + B}{2}\bar{\tau} = \frac{1}{\epsilon}ln\left(\frac{2-\rho}{1-\rho}\right) \tag{25}$$

Note that the ratio of the throughput of the TCP Reno connection to that of the proposed mechanism is equal to the ratio of areas enclosed by the the x axis and each line, indicating changes in the window size, as depicted in Figure 3. The area for TCP Reno, S_{reno}, is given by:

$$S_{\mathrm{reno}} = \frac{3}{4}\left\{(1-\rho)K\tau + B\right\}^2\bar{\tau}$$

On the other hand, the area for the proposed mechanism, denoted as S_{proposed}, is calculated as follows:

$$S_{\mathrm{proposed}} = \int_0^T w(t)dt = \frac{K\tau}{\epsilon}ln\frac{2-\rho}{2(1-\rho)}$$

Finally, the average ratio of the throughput of TCP Reno to that of the proposal mechanism is given by:

$$\lambda = \frac{S_{\mathrm{reno}}}{S_{\mathrm{proposed}}} = \frac{\frac{3}{4}\left\{(1-\rho)K\tau + B\right\}^2}{\frac{K\tau}{\epsilon}ln\frac{2-\rho}{2(1-\rho)}} \tag{26}$$

Note that ρ is given by solving Equation (25).

From Equations (25) and (26), we can understand the relationship between the variables (ϵ, K and B) and the ratio of throughput λ. Next, we show some numerical examples of the throughput ratio. Here we ignore the queuing delay and assume $\bar{\tau} = \tau$. Figure 4 shows changes in the throughput ratio with respect to ϵ, where we set K = 10 [Mbps] and τ = 50 [msec]. The five lines represent the results when the buffer size B is 1/4, 1/2, 1, 2, and 4 times the bandwidth-delay product (BDP) of the bottleneck link, respectively. In Figure 5, we show the results when we set τ = 50 [msec] and B to 41 [packets] (equal to BDP when K = 10 [Mbps]), where the five lines describe the results when K = 10, 50, 100, 500, and 1000 [Mbps].

These results show that ϵ, which realizes fairness between TCP Reno and the proposed mechanism, drastically changes when we modify K and/or B. Furthermore, in some situations, especially when the buffer size is large compared with the bandwidth-delay product, fairness cannot be realized by configuring ϵ. One reason is that the proposed mechanism converges its window size to $K\tau$, whereas TCP Reno continues increasing its window size until the buffer has been fully used. The primary reason of this unfairness is the characteristics of ImTCP [15, 16] which we deployed in the proposed mechanism for bandwidth measurement: ImTCP estimates an available bandwidth of the end-to-end network path, not a *fair shair* of the bottleneck link bandwidth. In other words, if there exists an inline measurement algorithm which can estimate a fair bandwidth share of the network, we can employ it to our proposed congestion control mechanism.

From another point of view, the congestion control algorithm of the proposed mechanism is essentially more conservative than TCP Reno. In contrast, TCP Reno has an aggressive window size control algorithm. Therefore, the unfairness between the proposed mechanism and TCP Reno cannot be avoided when they co-exist in the network. A similar discussion can also be found in the literature regarding TCP Vegas [22, 23], and we believe this is the primary reason that TCP Vegas was not successfully deployed in the Internet. In the case of TCP Reno and its variants using AIMD/MIMD

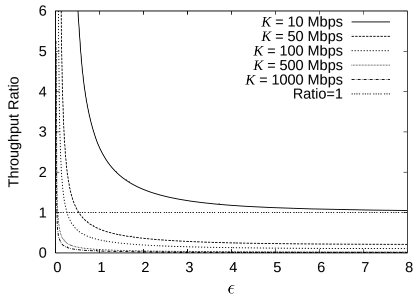


Fig. 5. Ratio of throughput for various physical bandwidths

policies, the window size just after packet loss occurs depends on the bottleneck link buffer size. That is, the throughput of these connections is improved as the buffer size increases. However, as buffer size becomes larger, the packets within the buffer also become larger, which means that the queuing delay is also increased.

IV. Simulation Results

In this section, we present simulation results by which to evaluate the performance of the congestion control mechanism proposed in Section II.

A. Simulation Settings

We use ns-2 [24] for the simulation experiments. Traditional TCP Reno, HighSpeed TCP (HSTCP), Scalable TCP (STCP), and FAST TCP are chosen for performance comparison. We set ϵ = 1.95 and γ = 0.9 for the proposed mechanism according to the discussion in Subsections II-A and III-C. Note that we have confirmed that changes in these parameters have a limited effect on the performance of the proposed mechanism, especially on the transient behavior, and that the characteristics of the proposed mechanism shown below does not change. The parameters in HSTCP and STCP are set to the value described in [4] and [5], respectively, and SACK option [25] is set to be enabled for both protocols. FAST TCP has the parameter α, which should be changed according to the link bandwidth. According to the guidelines in [26] we set α = 10, 20, 50, 100, 200, 500, and 1000 for link bandwidths K = 10, 20, 50, 100, 200, 500, and 1000 [Mbps], respectively.

The network model used in the simulation is depicted in Figure 6. The model consists of sender/receiver hosts, two routers, and links between the hosts and routers. $N_{\rm tcp}$ TCP connections are established between TCP sender i and TCP receiver i. To create background traffic, we injected UDP packets at a rate of $r_{\rm udp}$ into the network, where the packet size distribution follows the traffic observation results in the Internet [27]. That is, $N_{\rm tcp}$ TCP connections and an UDP flow share a bottleneck link between the two routers. The bandwidth of the bottleneck link is denoted as BW, and the propagation delay is τ. The bandwidth and the propagation delay of the access link for TCP sender i are bw_i and τ_i, respectively. We deployed the TailDrop scheme at the router buffer, and the buffer size is set to be equivalent to the bandwidth-delay product between sender and receiver hosts.

B. Basic Behavior

First, we confirm the fundamental behavior of the proposed mechanism with one TCP connection. Figure 7 shows the changes in window size of TCP Reno, HSTCP, STCP, FAST TCP, and the proposed mechanism, where we set $N_{\rm tcp}$ = 1, BW = 100 [Mbps], τ = 25 [msec], bw_1 = 200 [Mbps], and τ_1 = 5 [msec]. In this case, we do not inject UDP traffic into the network. The result shows that TCP Reno, HSTCP, and STCP connections experience periodic packet loss due to buffer overflow, because these connections continue increasing the window size until packet loss occurs. On the other hand, since the window sizes of FAST TCP and the proposed mechanism converge quickly to an ideal value, no packet loss occurs. The speed of window size increase is much higher for FAST TCP and the proposed mechanism than for HSTCP and STCP, meaning that FAST TCP and the proposed mechanism can more effectively utilize the link bandwidth. Furthermore, Figure 8 describes the results for the case in which BW = 1 [Gbps] and bw_1 = 2 [Gbps]. Based on these results, we observe that TCP Reno and HSTCP increase their window size slowly. However, the speed of the window size increase of the other mechanisms remains fast regardless of the link bandwidth. Note also that HSTCP and STCP, which rapidly increase their window size, cause more packet losses than TCP Reno. In the case of Figure 7, the SACK mechanism works well, and the sender host avoids timeouts. However, as shown in Figure 8, many retransmission timeouts occur because the SACK mechanism cannot recover all of the lost packets as the link capacity becomes large.

C. Scalability to Network Bandwidth and Delay

We next investigate the scalability to the link bandwidth of the proposed mechanism by checking the convergence time, defined as the time required for the TCP connection to utilize 99% of the link bandwidth. We set $N_{\rm tcp}$ = 1, τ_1 = 5 [msec], τ = 25 [msec], and $\tau_{\rm u}$ = 5 [msec]. Figure 9 shows changes in the convergence time when we change BW from 10 [Mbps] to 1 [Gbps], where $r_{\rm udp}$ is set to (0.2 BW) [Mbps] and bw_1 is set to be equal to BW. In the figure, the average values and the 95% confidence intervals for 10 simulation experiments are shown. From this figure, we can see that the TCP Reno connection requires a great deal of time to fully utilize the link bandwidth since the increasing speed of the window size is fixed at a small value, regardless of the link bandwidth. HSTCP dramatically reduces the convergence time, but the larger the link bandwidth becomes, the greater the convergence time that is required in order to fill the bottleneck link bandwidth. This means that HSTCP is fundamentally unable to resolve the scalability problem of TCP Reno. In the case of STCP and FAST TCP, the convergence time remains constant regardless of the link bandwidth, which is also confirmed in [5] and [6]. The proposed mechanism retains an approximately constant convergence time regardless of the link bandwidth, which shows good scalability to network bandwidth.

We also note that the convergence time of the proposed mechanism is a slightly worse than that of FAST TCP, especially in Figure 9. This is because of the choice of the control parameters in both mechanisms. In other words, with a different set of the control parameters for FAST TCP and the proposed mechanism, the opposite results may be obtained. In addition, since the congestion control mechanism of FAST TCP is based on that of TCP Vegas, it is considered that FAST TCP has the same difficluty in parameter setting as TCP Vegas described in [28]. Anyway, the most important

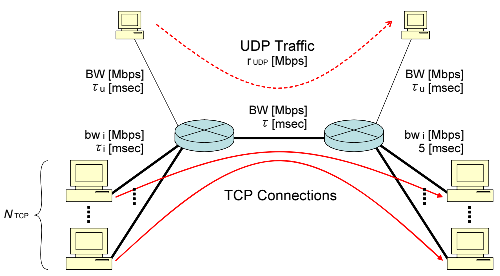


Fig. 6. Network topology in simulation experiments

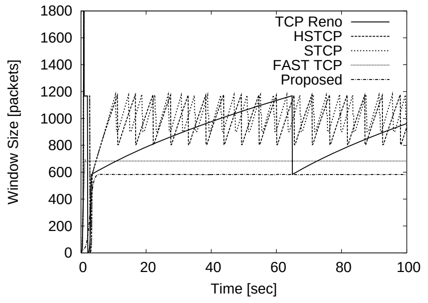


Fig. 7. Changes in window size (BW=100 [Mbps])

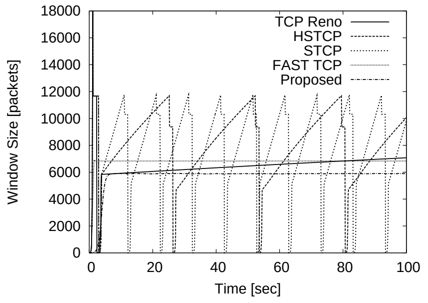


Fig. 8. Change in window size (BW=1 [Gbps])

characteristics observed in Figures 9 and 10 is scalability to the bandwidth-delay product of the network, which means that the convergence time changes as the bandwidth and/or delay become large.

Moreover, we investigate the scalability to the propagation delay of the proposed mechanism. We set $N_{\rm tcp}$ = 1, BW = 100 [Mbps], bw_1 = 200 [Mbps], τ_1 = 5 [msec], and $r_{\rm udp}$ = 20 [Mbps]. Figure 10 shows the changes in the convergence time when we change τ from 10 [msec] to 500 [msec]. This figure shows that the TCP Reno connection requires quite a long time to fully utilize the link bandwidth because it only increases its window size by one packet per RTT. The convergence time of HSTCP and FAST TCP is less than that of TCP Reno. However, the greater the increase in propagation delay, the larger the convergence time becomes. STCP has good scalability to link bandwidth as described in Figure 9, but the convergence time increases when the delay becomes larger because HSTCP, STCP, and FAST TCP increase their window size when receiving ACK packets, which depends on RTT. The proposed mechanism maintains the best scalability to the network delay, because, as shown in Subsection III-A, the convergence time increases logarithmically with increases in the delay or bandwidth.

D. Adaptability and Fairness

We also investigate the adaptability and fairness of the proposed mechanism by checking the effect of changes in the number of TCP connections. We set $N_{\rm tcp}$ = 5, BW = 100 [Mbps], τ = 25 [msec], bw_i = 100 [Mbps] ($1 \le i \le 5$), and τ_i = 5 [msec]. We do not inject UDP traffic into the network. TCP connections 1–5 join the network at 0, 100, 300, 500, and 700 [sec] and stop sending data packets at 900, 950, 1000, 1050, and 1100 [sec], respectively. Figure 11 shows changes in window size for the five TCP connections with respect to the time for HSTCP, STCP, FAST TCP, and the proposed mechanism.

Figure 11(a) shows that HSTCP control their window size with the AIMD policy and realize fairness among connections by inducing periodic packet losses. From Figure 11(b), we can see that STCP cannot realize fairness among connections because its window size control algorithm is based on the MIMD policy. In Figure 11(c), we can see that the nature of FAST TCP is as follows. Since FAST TCP utilizes queuing delay as a congestion signal, it can adjust its window size without inducing any packet loss when a new TCP connection joins the network. However, FAST TCP cannot achieve fairness among existing connections and a new connection.

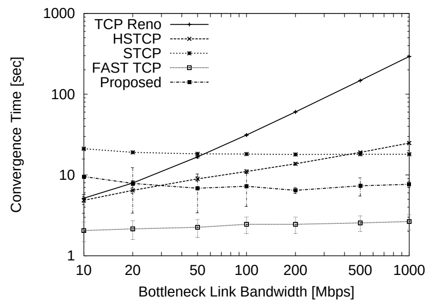


Fig. 9. Convergence time with respect to bottleneck link bandwidths

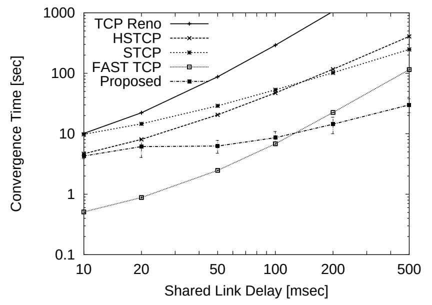
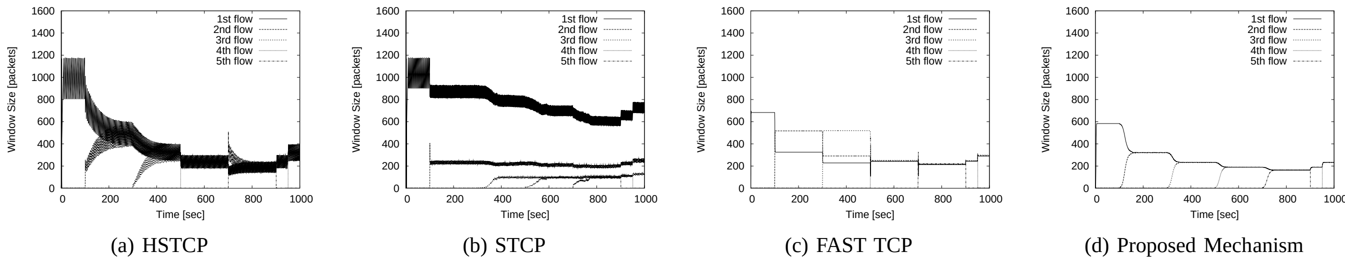


Fig. 10. Convergence time with respect to bottleneck link delays

Although FAST TCP needs RTT information to control the window size, the new connection cannot successfully measure the minimum RTT due to the queuing delay caused by the existing connection. When a connection stops a transmission and exits from the network, the remaining connections enjoy equal throughput because the buffer becomes temporarily empty, and the existing connections can measure the precise values for minimum RTT. On the other hand, Figure 11(d) shows that the proposed mechanism converges the window sizes very quickly, so that no packet loss occurs when a new connection joins the network. Furthermore, when the TCP connection leaves the network, the proposed mechanism connections quickly fill the unused bandwidth. Borrowing the terminology of biophysics, we say that TCP connections are competitive, but still symbiotic even against the environmental changes.

Adaptability to changes in the available bandwidth is also an important characteristic of the transport layer protocol. To confirm that performance of the proposed mechanism, we set $N_{\rm tcp}$ = 1, BW = 100 [Mbps], τ = 25 [msec],



(a) HSTCP (b) STCP (c) FAST TCP (d) Proposed Mechanism

Fig. 11. Effect of changes in number of connections

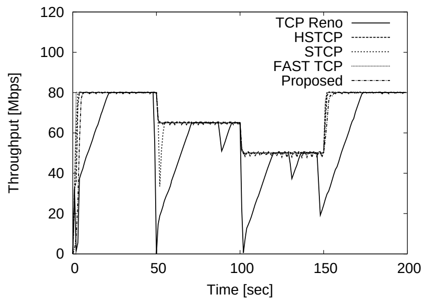


Fig. 12. Adaptability to change in available bandwidth (throughput)

bw_1 = 100 [Mbps], τ_1 = 5 [msec], and change $r_{\rm udp}$ so that the available bandwidth of the bottleneck link is 80 [Mbps] at 0–50 [sec], 65 [Mbps] at 50–100 [sec], 50 [Mbps] at 100–150 [sec], and 80 [Mbps] at 150–200 [sec]. Figures 12 and 13 present the changes in the throughput of a TCP connection and the queue size of the bottleneck link buffer for TCP Reno, HSTCP, STCP, and the proposed mechanism. The results obviously show the effectiveness of the proposed mechanism, which gives good adaptability to the changes in the available bandwidth. Furthermore, no packet loss occurs even when the available bandwidth suddenly decreases. On the other hand, TCP Reno connections experience packet losses during simulation time, and link utilization is much lower than 100%. Although HSTCP and STCP can retain their link utilization because of a sufficient buffer, they have largely fluctuating RTTs caused by queuing delays. FAST TCP and the proposed mechanism experience no packet loss and retain their link utilization with small RTTs, but the proposed mechanism has a smaller queue size than FAST TCP. This is one of the advantages of the proposed mechanism, which uses an inline measurement technique, which means that the proposed mechanism is quite robust against environmental changes of the network.

E. Effect of Heterogeneity in Physical Bandwidth

In the above subsections we demonstrated the effectiveness of the proposed mechanism with respect to various aspects. However, in a sense, these results are expected as a result of the newly developed congestion control mechanism based on the bandwidth measurement technique. A more striking feature of the proposed mechanism is detailed in the following results. Here, we investigate the effects of the heterogeneity of access networks such as the differences in access link bandwidth. We set $N_{\rm tcp}$ = 2, τ = 40 [msec], bw_1 = 10 [Mbps], bw_2 = 20 [Mbps], τ_1 = τ_2 = 5 [msec], and we change BW from 5 [Mbps] to 30 [Mbps]. UDP traffic is not injected into the network. Figure 14 shows the changes in the throughput of the two TCP connections in TCP Reno and the proposed mechanism with respect to BW. The figure shows that TCP Reno shares the bottleneck link bandwidth fairly, regardless of the value of BW. On the other hand, the proposed mechanism shows an interesting characteristic. When $BW < bw_1$, the two TCP connections share bottleneck link bandwidth fairly. However, when $bw_1 < BW < bw_2$, the bottleneck link bandwidth is distributed proportionally to the ratio of bw_1 and bw_2. This property can be explained using the equation of the proposed mechanism. Using Equation (8), the converged transmission rate for connection i, denoted by $\hat{N}_i$, which has a different physical link bandwidth (K_i), can be calculated as follows:

$$\hat{N}_i = \frac{K_i}{\sum_{i=1}^{n} K_i} \cdot BW \tag{27}$$

This equality is satisfied when $\gamma < 1$. This equation means that the bottleneck link bandwidth is shared proportionally to the physical bandwidth of each TCP connection. Since the physical bandwidth of the network path is defined as the bandwidth of the tightest link between TCP hosts (a sender and a receiver), the simulation results shown in Figure 14 agree with Equation (27). We argue that this characteristic is ideal for an Internet congestion control strategy. Throughout the history of the Internet, the ratio of the bandwidth of access networks to backbone networks has been changing over time [29]. Compared with access networks, the resources of backbone networks are sometimes scarce and sometimes plentiful. We believe that when backbone resources are few, they should be shared fairly between users, regardless of their access link bandwidth. On the other hand, when backbone resources are sufficient, they should be shared according to the access link bandwidth. The characteristics of the proposed mechanism, shown in Figure 14 and Equation (27), realize such a resource sharing strategy.

V. Conclusion

In this paper, we propose a new congestion control mechanism for TCP based on inline network measurement. The proposed mechanism obtains information about the physical and available bandwidths from inline network measurement via ImTCP. Using bandwidth information, the proposed mechanism adjusts its window size with an algorithm based on mathematical models borrowed from biophysics. Consequently, the proposed mechanism can converge its window size to an ideal value and avoid the periodic packet loss experienced by TCP Reno.

Through mathematical analysis, we confirm that the proposed mechanism has good scalability to not only link band-

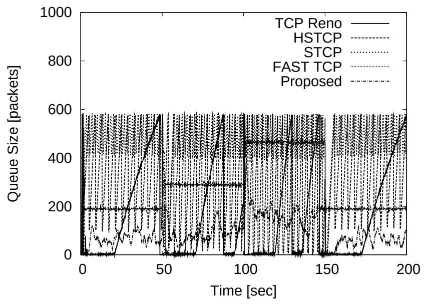
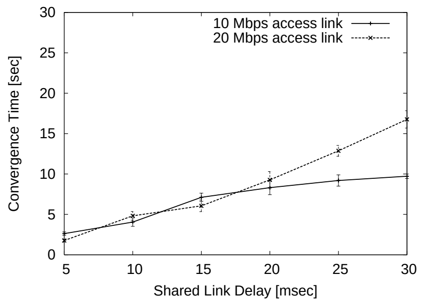
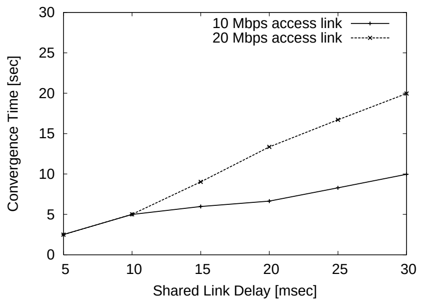


Fig. 13. Adaptability to change in available bandwidth (queue size)



(a) TCP Reno



(b) Proposed Mechanism

Fig. 14. Effect of different access link bandwidths

width but also propagation delay between the sender and receiver hosts. Other transport layer protocols such as TCP Reno, HighSpeed TCP, Scalable TCP, and FAST TCP cannot provide such scalability. Furthermore, based on the results of mathematical analysis regarding competition between TCP Reno and the proposed mechanism, although the realization of fairness between them was observed to be difficult, we believe that the proposed mechanism is a possible solution for transport layer protocols for future high-speed networks. Furthermore, through extensive simulations, we have confirmed that the proposed mechanism does exhibit the analytically determined characteristics. Therefore, the proposed mechanism is effective regardless of network bandwidth or delay and can solve many of the problems associated with TCP Reno and its variants.

For future work, we will confirm additional characteristics of the proposed mechanism, which include fairness among connections with different RTTs and the effect of measurement errors on the physical and available bandwidths. We are now implementing the proposed mechanism on the FreeBSD system. Experiments on the actual network are also important research tasks.

References

[1] S. Shenker, L. Zhang, and D. D. Clark, "Some observations on the dynamics of a congestion control algorithm," *ACM Computer Communication Review*, vol. 20, no. 5, pp. 30–39, Oct. 1990.

[2] L. Guo and I. Matta, "The war between mice and elephants," *Technical Report BU-CS-2001-005*, May 2001.

[3] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proceedings of IEEE INFOCOM 2003*, Apr. 2003.

[4] S. Floyd, "HighSpeed TCP for large congestion windows," *Request for Comments 3649*, Dec. 2003.

[5] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," in *proceedings of PFLDnet '03: workshop for the purposes of discussion*, Feb. 2003.

[6] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.

[7] Y. R. Yang and S. S. Lam, "General AIMD congestion control," in *Proceedings of IEEE ICNP 2000*, November 2000.

[8] 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*, Oct. 1994.

[9] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proceedings of ACM SIGCOMM 2002*, Aug. 2002.

[10] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proceedings of NLANR PAM2003*, Apr. 2003.

[11] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" in *Proceedings of IEEE INFOCOM 2001*, Apr. 2001.

[12] V. Jacobson, "Pathchar - a tool to infer characteristics of internet paths," available from http://www.caida.org/tools/utilities/others/pathchar/, Apr. 1997.

[13] R. King, R. Baraniuk, and R. Riedi, "TCP-Africa: An adaptive and fair rapid increase rule for scalable TCP," in *Proceedings of IEEE INFOCOM 2005*, Mar. 2005.

[14] S. Bhandarkar, S. Jain, and A. L. N. Reddy, "Improving TCP performance in high bandwidth high RTT links using layered congestion control," in *Proceedings of PFLDnet 2005*, Feb. 2005.

[15] M. L. T. Cao, G. Hasegawa, and M. Murata, "Available bandwidth measurement via TCP connection," in *Proceedings of IFIP/IEEE MMNS 2004*, Oct. 2004.

[16] ——, "A merged inline measurement method for capacity and available bandwidth," in *Proceedings of NLANR PAM 2005*, Mar. 2005.

[17] J. D. Murray, *Mathematical Biology I: An Introduction*. Springer Verlag Published, 2002.

[18] A. Montresor, H. Meling, and O. Babaoglu, "Toward self-organizing, self-repairing and resilient distributed systems," chapter 22, pages 119-124. Number 2584 in *Lecture Notes in Computer Science*. Springer-Verlag, June 2003.

[19] E. Bonabeau, M. Dorigo, and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*. New York: Oxford University Press, 1999.

[20] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Massachusetts: Addison-Wesley, 1994.

[21] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: Shadow prices, proportional fairness, stability," *Journal of Operations Research Society*, vol. 49, pp. 237–252, 1998.

[22] G. Hasegawa, K. Kurata, and M. Murata, "Analysis and improvement of fairness between TCP Reno and Vegas for deployment of TCP Vegas to the Internet," in *Proceedings of IEEE ICNP 2000*, Nov. 2000.

[23] G. Hasegawa, M. Murata, and H. Miyahara, "Fairness and stability of the congestion control mechanism of TCP," in *Proceedings of IEEE INFOCOM'99*, Mar. 1999, pp. 1329–1336.

[24] The VINT Project, "UCB/LBNL/VINT network simulator - ns (version 2)," available from http://www.isi.edu/nsnam/ns/.

[25] M. Mathis, "TCP selective acknowledgement options," *Request for Comments 2018*, Oct. 1996.

[26] C. Jin, D. X. Wei, and S. H. Low, "Internet draft: FAST TCP for high-speed long-distance networks," *Internet draft draft-jwl-tcp-fast-01.txt*, June 2003.

[27] A. Technologies, "Mixed packet size throughput," available from http://advanced.comms.agilent.com/n2x/docs/journal/JTC_003.html.

[28] T. A. Trinh and S. Molnar, "A game-theoretic analysis of TCP Vegas," in *Proceedings of QofIS 2004*, Oct. 2004.

[29] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield, "QoS's downfall: At the bottom, or not at all!" in *Proceedings of ACM SIGCOMM 2003 Workshop on Revisiting IP QoS (RIPQOS)*, Aug. 2003.