Master's Thesis

Title

Performance Evaluation of TCP Congestion Control Mechanism based on Inline Network Measurement

Supervisor

Professor Masayuki Murata

Author

Yousuke Matsuura

February 15th, 2006

Department of Information Networking

Graduate School of Infomation Science and Technology

Osaka University

Master's Thesis

Perfomance Evaluation of TCP Congestion Control Mechanism based on Inline Network Measurement

Yousuke Matsuura

Abstract

TCP Reno is the most widely deployed variant of Transmission Control Protocol (TCP), meaning that almost all TCP implementations in the current OSs are based on that version of TCP. However, TCP has the problem that the performance deteriorates especially in large-bandwidth and long-delay networks. To solve the problem, our research group has proposed a congestion control mechanism as an alternative to the traditional TCP Reno. Our proposed mechanism uses the information of the physical and currently available bandwidth of the network path between sender and receiver hosts for congestion control. One of the novel idea of the proposed mechanism is that we deploy an algorithm based on a logistic growth model and a Lotka-Volterra competition model from biophysics in regulating the congestion window size of a TCP connection. The physical/available bandwidth information are obtained through an inline network measurement technique we has already proposed, which measures the bandwidth by using data/ACK packets belonging to the TCP connection, without using additional probing packets. Our research group has also proposed the Interrupt Coalescence-aware Inline Measurement (ICIM) for measuring bandwidth over 1 Gbps, which can not be obtained by existing measurement algorithms/tools. However, we have not evaluated the proposed congestion control mechanism with ICIM in high-speed and long-delay networks, whereas the proposed mechanism is intended to be deployed to such network environment.

In this thesis, through extensive simulation experiments, we evaluate the performance of our mechanism in high-speed and long-delay networks. We evaluate the performance of our mechanism in terms of throughput, the change in the congestion window size and queue length of the bottleneck link. Also, we compare the performance of our mechanism with those of other existing TCP variants, and present that our mechanism achieves good performance. It achieves almost

100 % of link utilization while keeping the queue length of the bottleneck link small, regardless of the change in available bandwidth of the network path.

Keywords

Transmission Control Protocol (TCP), Congestion Control, Simulation, Performance Evaluation

Contents

List of Figures

List of Tables

1 Introduction

With increases in the heterogeneity and the complexity of the Internet, many problems have emerged in TCP Reno's congestion control mechanism ([1-5] 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 the window size should be changed according to the network environment. Although many solutions have been proposed for there problems [5-8], 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 is an explanation of the fundamental problem: TCP Reno cannot adjust window size to an adequate value under various network environments. 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 AIMD policy contain this essential problem, they cannot avoid periodic packet losses.

There are some TCP variants, including TCP Vegas [9] and FAST TCP [7], 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.

Our research group has proposed a novel congestion control mechanism of TCP that utilizes the information of physical and available bandwidths obtained from an inline measurement technique [10]. The proposed mechanism does not use ad hoc algorithms such as TCP Vegas [9] and instead employs algorithms that have a mathematical background, which enable us to mathematically discuss and guarantee their behavior even though posing a simplification of the target system. For this, algorithms from the logistic growth model and the Lotka-Volterra competition model [11] are borrowed for proposed mechanism, both of which are used in biophysics to describe changes in the population of species. The biophysics models were chosen based on their essential nature of stability and robustness, which is achieved even when they behave independently 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 maximize the bandwidth utilization and the throughput of the connection.

In [10], our research group has evaluated the proposed congestion control mechanism has been evaluated. Since this mechanism requires physical/available bandwidth information of the network path to control congestion window size, Inline mesurement TCP (ImTCP) which is the inline network measurement mechanism proposed in [12] to get the available bandwidth information in the simulation evaluation in [10]. However, it is difficult for ImTCP to measure available bandwidth in high-speed networks over 100 [Mbps] due to its essential nature of the algorithm. Therefore, the proposed congestion control mechanism has not been evaluated in high-speed network yet. Also, in [10] we assumed that we can get accurate information on the physical bandwidth without measurement. Thet is,it is necessary to evaluate the proposed method with physical bandwidth measurement algorithm.

Fortunately, we have now a measurement algorithm for physical/available bandwidth of the high-speed network. Our research group proposed a new inline network measurement mechanism that works well in high-speed networks was proposed in [13]. That is called Interrupt Coalescence-

aware Inline Measurement (ICIM). ICIM estimates physical/available bandwidth based on changs in intervals of packet burst by chnging the number of packet in bursts. Unlike other active measurement tools, ICIM adjusts the number of packets that are transmitted in a burst caused by IC and estimates the available bandwidth by observing the number of packets in the burst as it passes through the network, rather than by observing the inter-intervals of the packets. ICIM does not set the sending interval of the packets, so the overhead for packet spacing at the sender is eliminated. The measurement results show that TCP with ICIM can transmit data with the same performance as Reno TCP and can measure the available bandwidth of high-speed networks.

In this thesis, we evaluate the effectiveness of congestion control mechanism proposed by our research group in [10] in high-speed networks through simulation expriments. We evaluate the fundamental characteristics of our congestion control mechanism by using a simple network model for the simulation. For performance comparison purpose, we conduct simulations with many congestion control mechanisms: TCP Reno, HighSpeed TCP, Scalable TCP, FAST TCP and BIC TCP. We utilize throughput, change in congestion window size and the queue length of the bottleneck link as performance metric.

The reminder of this thesis is organized as follows. In Section 2, we introduce the design of the congestion control mechanism based on inline network measurement and a measurement mechanism in high-speed networks. In Section 3, we show simulation settings and briefly summarize the congestion control algorithm and parameter setting of TCP variants for comparative evaluation. In Section 4, we present simulation results and discuss on the performance of our mechanisms and performance comparison with other TCP variants. Finally, in Section 5, we present our conclusions of this thesis and present future research plan.

2 Congestion Control Mechanism based on Inline Network Measurement

2.1 Congestion Control Mechanism based on Bandwidth Informatioin

The concept of the window updating algorithm of the proposed mechanism in [10] is borrowed from a biological system, which is often pointed out to be robust [14], 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., [15]). 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.

Of course the up-to-date and reliable available bandwidth is necessary in order to realize such a mechanism for TCP congestion control. Fortunately, the inline measurement method of TCP (ICIM [13]) can quickly obtain such information within several RTTs. Then the resultant control method has good scalability with respect to both RTT and capacity, which has not been achieved in the previous proposals. This is the main motivation for inline network measurement congestion control method. In this subsection, at first we briefly introduce the mathematical models borrowed from biophysics and present the proposed mechanism.

2.1.1 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, which is usually determined by the available sustaining resources. The logistic equation describes such changes in the population of a species as follows [11]:

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

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

2.1.2 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 [11]:

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

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

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, when the following conditions are satisfied, two species can survive in the environment [11]:

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

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:

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

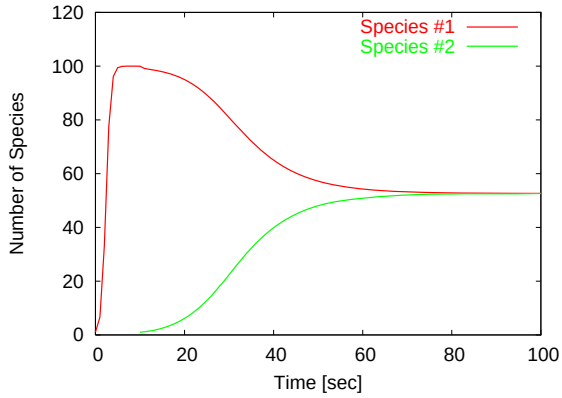


Figure 1: Changes in population of two species with the Lotka-Volterra competition model

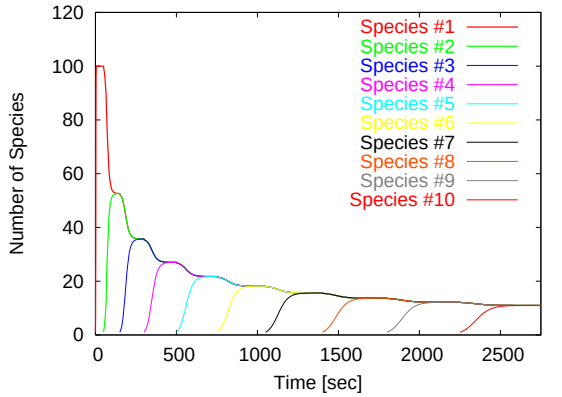


Figure 2: Changes in population of 10 species with the Lotka-Volterra competition model

Figure 2 shows the population changes among the ten species when using Equation (7), where $K = 100$, $\epsilon = 1.95$ and $\gamma = 0.90$, where new species join the environment one after another. We can observe that ten species converge in the same manner as two species, as shown in Figure 1. 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 Figures 1 and 2 are 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 con-

gestion control algorithm of TCP because the model must obtain N_j. This is discussed in the next subsection.

2.1.3 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:

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

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) contains the e^x calculation. Generally, exponentiation cannot be operated in the system kernel because of the lack of a library and the processing overhead. Therefore, for function e^x, we give the Taylor polynomial of degree 4 around $x = a$ as follows,

$$e^x \sim e^a \sum_{k=0}^{4} \frac{1}{k!}(x-a)^k$$

where a is the integer part of x (e.g., a = 0 when $0 \leq x < 1$). By preparing e^a on a memory table for a limited range of a, we can calculate e^x with minimal processing overhead. In determining the maximum value of a in the proposed mechanism, we consider the following equation:

$$x = \epsilon t \left\{ 1 - \gamma \left(1 - \frac{A_i}{K} \right) \right\} \leq \epsilon t$$

That is, when we assume that the maximum RTT of TCP connection is 10 [sec], we can determine the maximum value of a to $\lfloor 10\ \epsilon \rfloor$.

Equation (10) requires measurement of the physical and available bandwidths of a network path. Therefore, we utilize the inline network measurement technique in ICIM [16]. In [16], the authors proposed ICIM, which is an inline network measurement technique for the physical and available bandwidths of network paths between TCP sender and receiver hosts. ICIM can continuously measure bandwidth by using data and ACK packets of a TCP connection under data transmission. That is, the TCP sender transmits data packet bursts at intervals determined by an inline measurement algorithm and checks the arrival interval times of the corresponding ACK packet bursts to estimate bandwidth. Since ICIM 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.

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 [17], 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.

The inline network measurement based congestion control algorithm is based on traditional TCP Reno, and we use the same algorithm as TCP Reno for the window updating algorithm until measurement results are obtained through inline network measurements. That is, the slow start phase is used as other TCP congestion control methods. If two or more TCP connections were to open large windows at the same time, many packets would be lost, and so probes on the path are still necessary. Therefore, the slow start phase is utilized during several RTTs for accurate estimation of the bandwidth information. As well, in cases of packet loss, the window size is decreased in a manner identical to that of TCP Reno in both cases of timeout and fast retransmit [18]. When bandwidth information is obtained, the congestion control algorithm adjusts its window size using Equation (10). That is, when the j-th ACK packet is received at the TCP sender, we use Equation (10) to obtain the new value of the congestion window size of the TCP connection by the following calculation: set $w_i(0)$ to the current window size and t to the time duration from the arrival time of the (j-1)-th ACK packet to that of the j-th ACK packet.

2.2 Measurement Mechanism of Physical/Available Bandwidth in High-speed Networks

2.2.1 Measuring in High-speed Networks

In the present study, the authors of ImTCP focus on a new challenge regarding active measurement. Specifically, they investigate the bandwidth measurement of 1-Gbps or faster network paths, which are becoming increasingly popular. In such high-speed networks, ImTCP, Pathload and other active measurement tools based on packet spacing [19-22] must overcome the following problems. First, measurement in fast networks requires short transmission intervals of the probe packets (for example, 12 μs for a 1-Gbps link). However, regulating such short intervals causes a heavy load on the CPU. Second, network cards for high-speed networks usually employ Interrupt Coalescence (IC) [23, 24], which rearranges the arrival intervals of packets and causing bursty transmission, so that the algorithms utilizing the packet arrival intervals do not work properly.

2.2.2 Packet Burst-based Bandwidth Measurement Algorithm

As shown in Figure 3, we consider the situation in which two bursts of packets are sent at the interval S. The number of packets in the first burst (Burst 1) is N. Assume that C is the capacity of the bottleneck link. C_{Cross} is the average transmission rate of cross traffic over the bottleneck link when the two bursts pass the link, and P is the packet size. Then, the amount of traffic that enters the bottleneck link during the period from the point at which the first packet of Burst 1 reaches the link until the point at which the first packet of Burst 2 reaches the link will be the sum of the packets in Burst 1 and the cross traffic packets arriving in S, i.e., $C_{Cross} \cdot S + N \cdot P$. If the amount is larger than the transfer ability of the link during this period, considered to be $C \cdot S$, then Burst 2 will go to the buffer of the link. This results in a tendency for the interval between the two bursts to increase after leaving the bottleneck link.

We can write that the burst interval will be increased if

$$C_{Cross} \cdot S + N \cdot P > C \cdot S \tag{11}$$

or,

$$\frac{N \cdot P}{S} > C - C_{Cross}$$

Note that $C - C_{Cross}$ is the available bandwidth (A) of the bottleneck link. Therefore, Eq. (1) becomes

$$\frac{N \cdot P}{S} > A$$

Since we assume that the absolute timer is used, S is always larger than $RxAbsIntDelay$. Therefore, at the NIC of the TCP receiver, since the arrival interval of the two bursts are larger or equal to S, the two bursts are passed to the kernel in two different interrupts. The TCP receiver then sends the ACK of the two bursts in the same intervals to the sender TCP. Thus, by checking the arrival intervals of the corresponding ACK packets of the two bursts, the TCP sender can determine if $A > NP/S$. By sending numerous bursts with various values of NP/S (by changing N), we can search for the value of the available bandwidth A. This is the measurement principle of the proposed inline measurement mechanism.

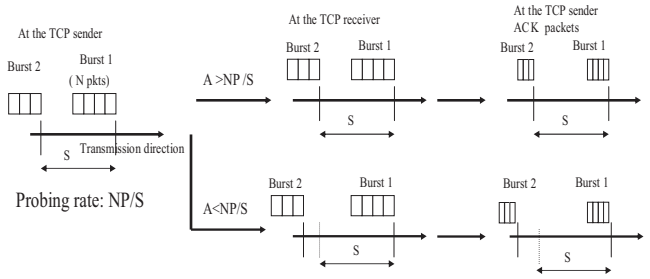


Figure 3: Packet burst-based available-bandwidth measurement principle

2.2.3 TCP with Interrupt Coalescence-aware Inline Measurement (ICIM)

ICIM inherits the concept of the *search range* from the measurement algorithm in ImTCP [25]. This is the idea of limiting the bandwidth measurement range using statistical information from previous measurement results rather than searching from 0 bps to the upper limit of the physical bandwidth for every measurement. By limiting the measurement range, we can keep the number of probe packets small.

At first, we explain how to search for the available bandwidth in a determined search range and then we present an overview of the measurement algorithm.

Assume that the search range for a measurement is (B_l, B_u). The algorithm then check k values in the range to determine which is nearest to the real available bandwidth. We use $k = 4$ in the following simulations. The k points are:

$$B_i = B_l + \frac{B_u - B_l}{k-1}(i-1) \;\; (i = 1, ..., k)$$

The TCP sender then sends k consequence bursts and the number of packets are adjusted so that the probe rate of Burst i is B_i:

$$\frac{N_i \cdot P}{S_i} = B_i \tag{12}$$

We illustrate the setting in Figure 4.

Realization of Eq. (2) requires the following:

- The value of S_i must be estimated at the timing of the transmission of Burst i. In fact, S_i is unknown until Burst $i + 1$ is transmitted. But we need the value at the timing of the transmission of Burst i in order to guarantee Eq. (3). We therefore estimate the value of S_i

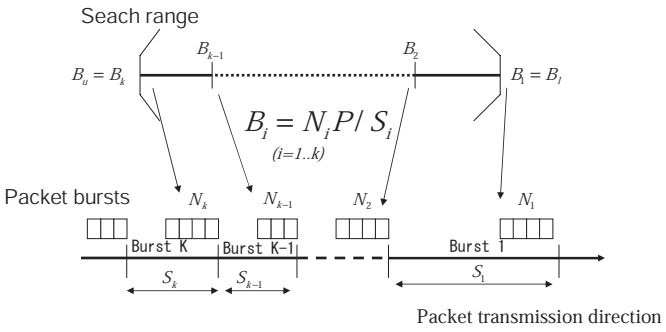


Figure 4: Probing a search range in ICIM

by assuming that the amount of data in Burst i is proportional to the length of the interval as follow:

$$S_i = \frac{N_i \cdot P}{T} \tag{13}$$

where T is the average throughput of TCP.

- In case the number of packets in Burst i is smaller than N_i, additional packets must be added to the burst so that the packet number becomes N_i. ICIM utilizes a buffer located at the bottom of the TCP layer in order to store the packets temporarily before sending them to the IP layer, in the manner of ImTCP. ICIM stores all of the packets of the burst that preceded Burst 1 in the buffer. Packets are added to Burst i ($i = 1..k$) when necessary in order to maintain the desired number of packets (N_i) in these bursts.

  ICIM sends k bursts and checks the corresponding ACK of the bursts. If from burst number j, $j = 1..k$, the arrival interval of the bursts becomes larger, then B_j is considered to be the value of the available bandwidth in that measurement. Here, the burst interval is consider to become larger if the arrival interval is larger then λ times of the sending interval. We set λ to 1.01 in the following simulations.

The measurement algorithm of ICIM is as follows:

(1) Set the initial search range

We set the initial search range as $(T, 2 \cdot T)$ where T is the throughput of TCP.

(2) Search for the available bandwidth in the decided search range.

ICIM waits until the window size ($cwnd$) is larger than C_{min} (large enough to create bursts for measurement). We use $C_{min} = 50$ in the following simulations. Data packets are then sent in order to search the available bandwidth in the decided search range, as described above.

(3) Add the new measurement result to the database and calculate the new search range.

The measurement result in the last step is added to a dabatase of measurement results. We then calculate the new search range (B'_l, B'_u) from the database. We use the 95% confidential interval of the data stored in the database as the width of the next search range, and the current available bandwidth is used as the center of the search range. The search range is calculated as follow:

$$B'_l = R - \max\left(1.96\frac{V}{\sqrt{q}}, \frac{R}{10}\right)$$

$$B'_u = R + \max\left(1.96\frac{V}{\sqrt{q}}, \frac{R}{10}\right)$$

where R is the latest measurement result. V is the variance of stored values of the available bandwidth and q is the number of stored values. $R/10$ is a value that ensures that the search range does not become too small. Moreover, when measurement result in Step 3 falls to B_l (B_u), it is possible to consider that the network has changed greatly so that the real value of the available bandwidth is lower (higher) than the search range. In this case, we discard the accumulated measurement results because they become unreliable as statistic data and enlarge the search range (B_l, B_u) twice towards the lower (higher) direction to create (B'_l, B'_u).

(4) Wait for Q seconds then return to Step 2 and start the next measurement. During the waiting time Q, TCP transmits packets in the normal manner. The waiting time is needed for the TCP transmission to return to the normal state after the packets store-and-forward process at Step 2.

2.2.4 Measurement Algorithm for Physical Bandwidth

For measuring the physical bandwidth of the network path, we employ the same algorithm in [17]. The largest advantage of the algorithm is that we can obtain the accurate estimation

results for physical bandwidth even when the network is highly loaded, while other existing mechanism can not derive the valid results in high-loaded network. See [17] for detailed algorithm for physical bandwidth measurement.

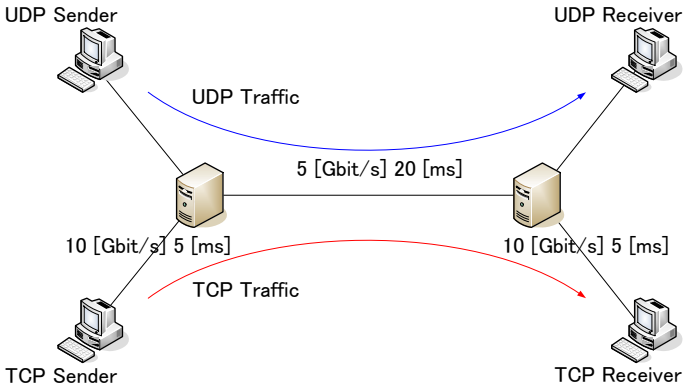


Figure 5: Network topology in simulation experiments

3 Simulation Settings

In this section, we explain the network model and various settings for the simulation. We use ns-2 [24] for simulation experiments.

3.1 Network Model and Parameters

The network model used in the simulation is depicted in Figure 5. The model consists of sender/receiver hosts, two routers, and links between the hosts and routers. A TCP connection is established between the TCP sender and the TCP receiver. To create background traffic, we injected UDP packets at a rate of r_{udp} into the network, where the packet size distribution follows the traffic observation results in the Internet [26]. That is, a TCP connection and an UDP flow share a bottleneck link between the two routers. The bandwidth of the bottleneck link is 5 [Gbps], and the propagation delay is 20 [msec]. The bandwidth and the propagation delay of the access link for TCP sender is 10 [Gbps] and 5 [msec]. 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 host. The buffer size of sender/reciever hosts is sufficient. The packet size is set to 1500 [KByte]. We set ϵ = 1.95 and γ = 0.9 for the evaluated mechanism, the effect on the performance caused by these parameter change is mentioned in Subsection 4.2.

3.2 TCP Variants for Comparative Evaluation

In this subsection, we explain TCP congestion control mechanisms for comparison candidates focusing on the algorithms of changing congestion window size at sender host, which make the largest influence on the performance evaluated in the next section. To understand their detailed mechanisms, please refer to [27] for TCP Reno, [5] for HighSpeed TCP (HSTCP), [28] for Scalable TCP, [7] for FAST TCP and [8] for BIC TCP.

3.2.1 TCP Reno

TCP Reno is the most popular version of TCP used on the current Internet. It increases the congestion window size on the reception of an ACK packet and decreases it when packet loss occurs as:

$$\begin{aligned} w_{\mathrm{reno}} &\leftarrow w_{\mathrm{reno}} + \frac{1}{w_{\mathrm{reno}}} \\ w_{\mathrm{reno}} &\leftarrow \frac{w_{\mathrm{reno}}}{2} \end{aligned}$$

3.2.2 HighSpeed TCP

The congestion control mechanism of HighSpeed TCP is designed to slove the problem of TCP Reno in high-speed networks. It increase congestion window size more aggressively and less drastically than TCP Reno. HighSpeed TCP acts equally TCP Reno when its congestion window size is not greater than the parameterLow_Window. Otherwise, the congestion windows size is regulated as:

$$\begin{aligned} w_{\mathrm{hstcp}} &\leftarrow w_{\mathrm{hstcp}} + \frac{a(w_{\mathrm{hstcp}})}{w_{\mathrm{hstcp}}} \\ w_{\mathrm{hstcp}} &\leftarrow (1 - b(w_{\mathrm{hstcp}}))w_{\mathrm{hstcp}} \end{aligned}$$

$a(w_{\mathrm{hstcp}})$ and $b(w_{\mathrm{hstcp}})$ are defined as

$$\begin{aligned} a(w_{\mathrm{hstcp}}) &= \frac{w_{\mathrm{hstcp}}^2 \cdot p(w_{\mathrm{hstcp}}) \cdot b(w_{\mathrm{hstcp}})}{2 - b(w_{\mathrm{hstcp}})} \\ b(w_{\mathrm{hstcp}}) &= (High\_Decrease - 0.5)\frac{\log(w_{\mathrm{hstcp}}) - \log(Low\_Window)}{\log(High\_Window) - \log(Low\_Window)} + 0.5 \end{aligned}$$

and $p(w_{\mathrm{hstcp}})$ is

$$p(w_{\mathrm{hstcp}}) = \exp(\log(Low\_P) + (\log(High\_P) - \log(Low\_P))$$

$$\frac{\log(w_{\mathrm{hstcp}}) - \log(Low\_Window)}{\log(High\_Window) - \log(Low\_window)})$$

$$Low\_P = \frac{1.5}{Low\_Window^2}$$

In the simulation experiments, we set parameters for HighSpeed TCP according to [5] as: $Low\_Window = 38$, $High\_Window = 83000$, $High\_P = 10^{-7}$ and $High\_Decrease = 0.1$.

3.2.3 Scalable TCP

Scalable TCP changes the congestion window size as:

$$w_{\mathrm{stcp}} \leftarrow w_{\mathrm{stcp}} + a$$

$$w_{\mathrm{stcp}} \leftarrow w_{\mathrm{stcp}} - \lceil b \cdot w_{\mathrm{stcp}} \rceil$$

Similarly to HighSpeed TCP, Scalable TCP behaves equally to TCP Reno when the current congestion window size is lower than $lwnd$. Owing to this algorithm, the time for recovery congestion window size after packet loss is independent from the link capacity. In the simulation experiments, in accordance with [28], we set parameters as $a = 0.01$, $b = 0.125$ and $lwnd = 16$. The parameters a and b are chosen because those value make good balance between rate fluctuation and convergence time of congestion window size [28].

3.2.4 FAST TCP

FAST TCP has a delay-based congestion control mechanism. It uses queueing delay as multi-bit congestion signal, while the traditional TCP Reno uses a binary congestion signal (packet loss).

$$w_{\mathrm{fast}} \leftarrow \min 2w_{\mathrm{fast}}, (1-\gamma)w_{\mathrm{fast}} + \gamma(\frac{baseRTT}{RTT}w_{\mathrm{fast}} + \alpha(w_{\mathrm{fast}}, qdelay))$$

Where $\gamma \in (0, 1]$. $baseRTT$ is the minimum RTT of the lifetime of a TCP connection. $qdelay$ is a queueing delay. In [7], $\alpha(w, qdelay)$ is defined as a constant value α. α corresponds to the number of FAST TCP's packets kept in the bottleneck link queue. α is set to 1000 in the following simulation experiments. This value is an instance to maintain about 2.5 [msec] of queueing delay with 5 [Gbps] link capacity for each FAST TCP connection [29]. It is difficult to set α according to the network condition because the adequate value of α is changed by link capacity. Too big α or too small α causes the oscillation of congestion window size [30], leading to the performance degradation.

3.2.5 BIC TCP

The congestion control mechanism of BIC TCP searches an adequate value for the congestion window size by using binary search algorithm. When a packet loss event occures, BIC TCP reduces the congestion window size by a multiplicative factor (β). The congestion window size just before the reduction is set to the maximum and the congestion window size just after the reduction is set to the minimum. Then, BIC TCP performs a binary search using these two parameters between W_{max} and W_{min}. Since packet losses have occurred at W_{max}, the appropriate congestion window size that the network can currently handle without packet losses must be somewhere between these two window sizes. BIC TCP increase congestion window size `Wbic` by following pseudo code.

```
while (Wmin <= Wmax){
    inc = (Wmin+Wmax)/2 - Wbic;
    if (inc > Smax)
        inc = Smax;
    else if (inc < Smin)
        inc = Smin;
    Wbic = Wbic + inc;
    if (no packet losses)
        Wmin = Wbic;
    else
        break;
}
```

BIC TCP decrease congestion window as

$$w_{\mathrm{bic}} \quad \leftarrow \quad (1-\beta) * w_{\mathrm{bic}}$$

In the simulation experiments, in accordance with [8], we set the parameters for BIC TCP as $\beta = 0.125$, $S_{min} = 32$, $S_{max} = 0.01$ and $low\_window = 14$. Similarly to HighSpeed TCP and Scalable TCP, BIC TCP acts equally TCP Reno when current congestion window size is lower than low_window.

4 Performance comparison with other TCP

In this section, we present simulation results to evaluate the performance of the congestion control mechanism based on inline metwork measurement. Additionally, we compare it with other congestion control mechanisms.

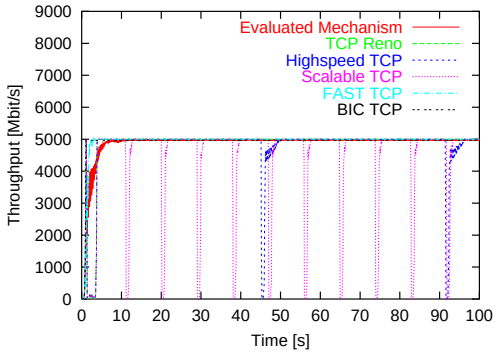
4.1 Fundamental behavior

First, we observe the behavior of congestion control mechanisms when there is no background traffic and the available bandwidth for a TCP connection remains unchanged. Figure 6 shows the simulation results of the change in throughput, congestion window size, and queue length of the bottleneck link.

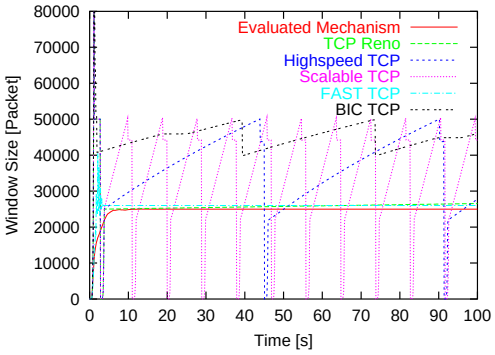
The congestion control mechanism based on inline network measurement achieves good performance. It fully utilizes the physical bandwidth. The congestion window size is regulated to bandwidth-delay product, and it keeps the queue length of the bottleneck link very small. TCP Reno achieves good performance too. However, this is a quite "lucky" case for TCP Reno, meaning that $ssthresh$ is set to the window size almost equal to bandwidth-delay product. Otherwise we cannot expect good performance since it increase congestion window size very slightly. High-Speed TCP and Scalable TCP has the same shortcoming. Since they increase the congestion window size until the router buffer becomes fully utilized, it causes periodical packet losses at the router buffer. Since many packet losses occur in a short period of time, the SACK mechanism does not function properly. Therefore, they begin slow start. FAST TCP achieves good performance in terms of throughput. However, because of its mechanism, FAST TCP needs to keeps some packet in the bottleneck link queue. Therefore, the congestion window size becomes slightly larger than the window size corresponding to bandwidth-delay product. BIC TCP achieves good throughput, but it increases its congestion window size too large and fills the bottleneck link buffer until packet loss occurs.

Next, we observe the behavior of TCP variants with background traffic. In this case, UDP traffic is generated in order to change available bandwidth to 3 [Gbps] at 0–30 [sec], 4 [Gbps] at 30–70 [sec], 3 [Gbps] at 70–85 [sec] and 2 [Gbps] at 85–100 [sec]. Figure 7 shows the simulation results.

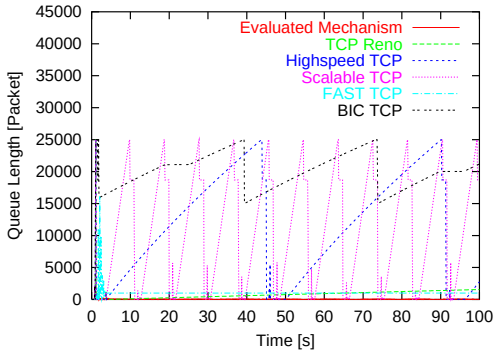
The congestion control mechanism based on inline network measurement achieves good per-



(a) Changes in throughput



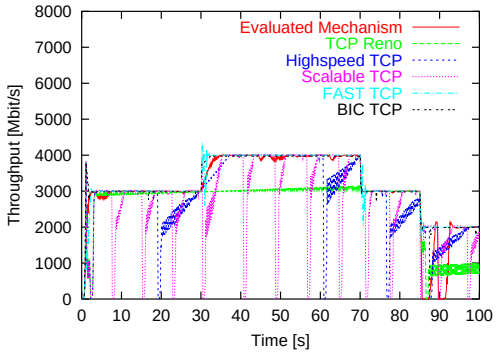
(b) Changes in congestion window size



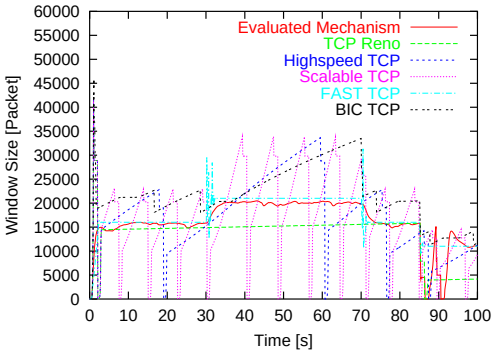
(c) Changes in queue length of the bottleneck link

Figure 6: Behavior of TCP variants without background traffic

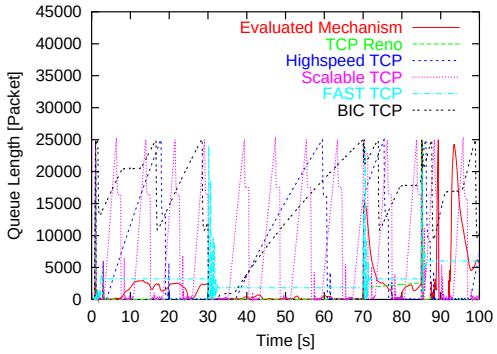
formance even in this case. It shows quite a good adaptability tothe change in the available bandwidth. It also achieves good utilization of bandwidth, changes the congestion window size properly, and keeps the queue length small. When the available bandwidth becomes small the queue length become large, but by changing congestion window size properly the queue length become small in a short time and the throughput is quickly recovered. The performance of TCP Reno is also bad in this case. Once it decrease its congestion window size, it requires quite a long time to recover congestion window size. HighSpeed TCP is not good for same reason as the case of no background traffic. However, they are superior to TCP Reno in terms of higher average throughput. FAST TCP achieves good performance. It shows a good adaptability to the available bandwidth as well as the congestion control meachanism based on inline network measurement.



(a) Changes in throughput



(b) Changes in congestion window size



(c) Changes in queue length of the bottleneck link

Figure 7: Behavior of TCP variants with background traffic

However, the queue length of the bottleneck link becomes larger as background traffic becomes larger. When available bandwidth changes the queue length temporarily become very large, but by changing congestion window size properly it maintains its stability. Equivalent to the case of no background traffic, BIC TCP can fully utilize the available bandwidth, but it introduce the long queue length at the bottleneck link.

We summarize the characteristics of TCP variants evaluated in this subsection in Table 1.

4.2 Effect of Parameter Setting

The congestion control meachanism based on inline network measurement has two parameters, γ and ϵ. In this subsection, we discuss the effect of these parameters.

congestion control	congestion control based on inline network measurement	TCP Reno	HighSpeed TCP	Scalable TCP	FAST TCP	BIC TCP
throughput	◯	×	△	△	◯	◯
packet loss	◯	×	×	×	△	×
parameter settings	◯	◯	△	◯	×	△
queueing delay	◯	×	×	×	△	×
ease of process	congestion cotrol △ measurement method ×	◯	△	◯	△	△
responsiveness to available bandwidth	◯	×	△	△	◯	△

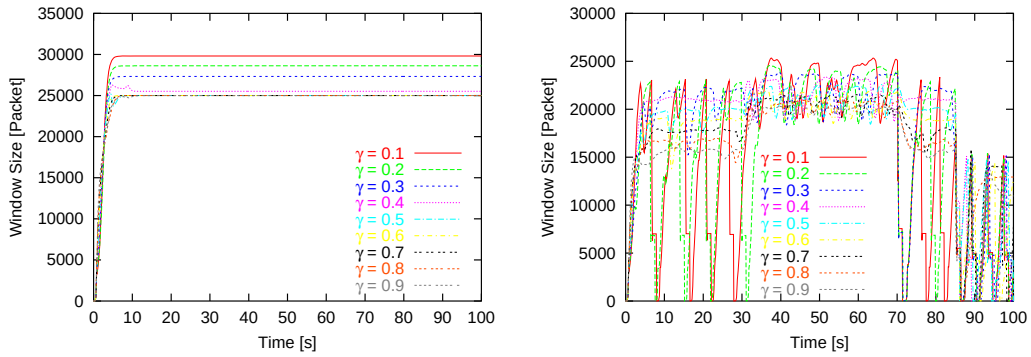
Table 1: Performanc comparison

4.2.1 γ Setting

The parameter γ indicates the degree of the influence of the other competing connections that share the same bottleneck link. From Equation (9), we get convergence window size as:

$$w^* = \{(1-\gamma)K + \gamma A_i\}\tau_i \tag{14}$$

From Equation (14), greater γ leads to smaller convergence window size. This can be observed in Figure 9, which depicts the effect of the γ value on the change in the congestion window size. The converged window size increases as γ decrease, then the queue size of the bottleneck link becomes larger, shown in Figure 9. In particular, when the available bandwidth is small, the performance of congestion control mechanism based on inline network measurement deteriorates because of causing packet losses. To converge the congestion window size, $0 < \gamma < 1$ is needed [11]. Therefore, we recommend to set γ close to 0.9. But greater γ causes larger convergence time especially when multiple TCP connections share the bottleneck link. The performance evaluation when more than one TCP connections exists in the network is one of the important future work.



(a) Available bandwidth does not change

(b) Available bandwidth changes

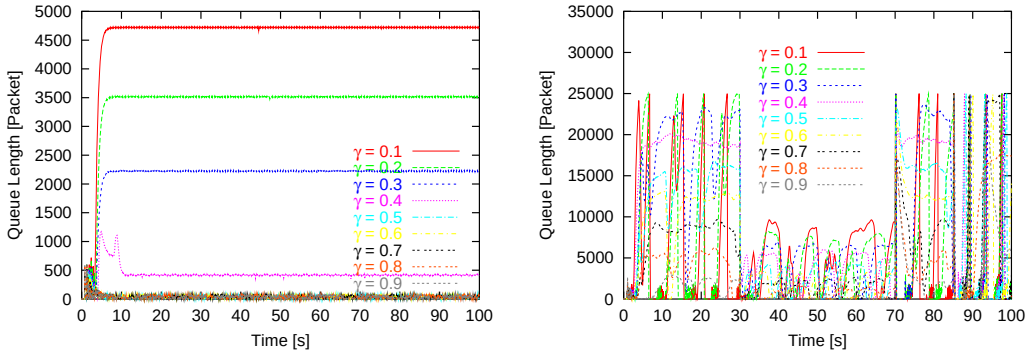
Figure 8: Changes in congestion window size with various γ values

4.2.2 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 $\epsilon \geq 2$ [11]. In contrast, the window size updating algorithm proposed in Subsection 2.1.3 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. Figure 10 shows the change in the congestion window size and we can see that the congestion window size converges quickly when ϵ is large. Figure 11 shows the change in the queue length of the bottleneck link, presenting that the queue length does not become larger when ϵ become greater. On the other hand, to avoid the oscillation of the congestion window size, we need to consider delayed feedback [10]. ϵ is needed to satisfy

$$\epsilon \leq \frac{\pi}{2\tau_d}$$

where τ_d is the delay of the feed back information. Due to the nature of ImTCP's bandwidth measurement algorithm, the delay in the proposed mechanism corresponds to the time required for the data (ACK) packets to traverse from the bottleneck link to the sender hosts. Since ICIM needs up to 4 RTTs to measure the bandwidth information, the delay is approximately 2 RTTs since ImTCP estimates the average values of the physical and available bandwidth for the 4 RTTs. That is, the length of the delay depends on the RTT value of a TCP connection. In other words, by

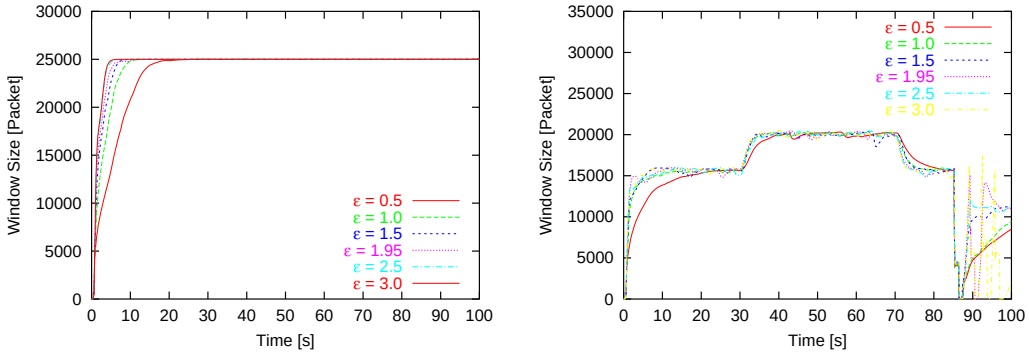


(a) Available bandwidth does not change

(b) Available bandwidth changes

Figure 9: Changes in queue length of bottleneck link with various γ values

setting ϵ according to the RTT for each TCP connection, the congestion control mechanism based on inline network measurement can avoid window size oscillation. However, using different values of ϵ results in different convergence speeds as shown in Equation (9), and short-term unfairness among connections with different RTTs might occur. Therefore, we suggest that the fixed value for ϵ is used, which is around 2.0 under the rough assumption that the maximum RTT value of a TCP connection is about 500 [msec].



(a) Available bandwidth does not change

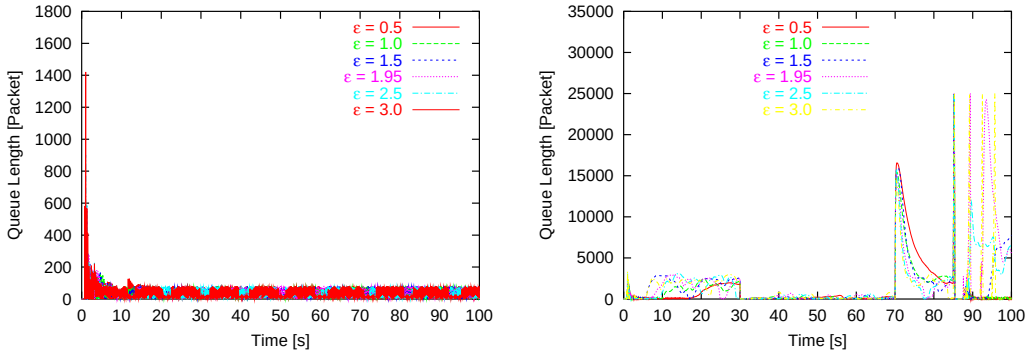
(b) Available bandwidth changes

Figure 10: Changes in window size with various ϵ values

5 Conclusion

In this thesis, we evaluate the congestion control mechanism based on inline network measurement in high-speed networks through simulation experiments. From the extensive simulation results, we indicate the effectiveness of the mechanism in high-speed networks. In addition, we confirm that the mechanism is superior to other existing TCP congestion control mechanisms, in terms of enough throughput and small queue length at the bottleneck link, whereas almost no packet loss occurs. We also show the guidelines in setting the parameters in our congestion control scheme, by validating the analysis results from the simulation results.

For the future work, we will evaluate the interaction between multiple connections of our mechanism and the fairness among TCP Reno. Implementation experiments are also remaining task.



(a) Available bandwidth does not change

(b) Available bandwidth changes

Figure 11: Changes in queue length of the bottleneck link with various ϵ values

Acknowledgement

I would like to express my deepest appreciation to Professor Masayuki Murata. Without his help, I could not carry on a study.

I feel grateful to Associate Professor Go Hasegawa for his proper and continuous guidance.

I am grateful to Professors Koso Murakami, Makoto Imase, Teruo Higashino, Hirotaka Nakano, and Tetsuji Satoh of Osaka University, for their appropriate guidance.

I would like to express sincere thanks to Cao Le Thanh Man for his kind help.

I would like to express gratitude to the member of my research group and laboratory too.

I wish sincerely to express my gratitude to all the people who supported me.

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, pp. 30–39, Oct. 1990.

[2] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme of TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, pp. 270–280, Oct. 1996.

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

[4] 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.

[5] S. Floyd, "HighSpeed TCP for large congestion windows," *Request for Comments (RFC) 3649*, Dec. 2003.

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

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

[8] L. Xu and K. H. andInjong Rhee, "Binary increase congestion control (BIC) for fast long-distance networks," in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.

[9] 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.

[10] T. Iguti, G. Hasegawa, and M. Murata, "A new congestion control mechanism of TCP with inline network measurement," in *Proceedings of The International Conference on Information Networking (ICOIN) 2005*, pp. 109–121, Jan. 2005.

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

[12] C. L. T. Man, G. Hasegawa, and M. Murata, "An inline measurement method for capacity of end-to-end network path," in *Proceedings of the 3rd IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (E2EMON 2005)*, May 2005.

[13] C. L. T. Man, G. Hasegawa, and M. Murata, “ICIM: An inline network measurement mechanism for highspeed networks,” *to be presented in Fourth IEEE/IFIP Workshop on End-to-End Monitoring Techniques and Services (E2EMON ’06)*, Apr. 2006.

[14] 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.

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

[16] C. L. T. Man, G. Hasegawa, and M. Murata, “An inline network measurement mechanism for High-Speed networks,” *technical report of IEICE* (IN2005-123), pp. 79–84, Dec. 2005.

[17] C. L. T. Man, G. Hasegawa, and M. Murata, “A merged inline measurement method for capacity and available bandwidth,” in *Proceedings of the 6th Passive and Active Measurement Workshop PAM 2005*, Mar. 2005.

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

[19] N.Hu and P.Steenkiste, “Evaluation and characterization of available bandwidth probing techniques,” *IEEE Journal on Selected Areas in Communications*, no. 6, pp. 879–894.

[20] J.Strauss, D.Katabi and F.Kaashoek, “A measurement study of available bandwidth estimation tools,” in *Proceedings of Internet Measurement Conference 2003*, Oct. 2003.

[21] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil and L. Cottrell, “PathChirp: Efficient available bandwidth estimation for network paths,” in *Proceedings of Passive and Active Measurement Workshop*, 2003.

[22] J. Navratil and R. Cottrell, “ABwE: A practical approach to available bandwidth estimation,” in *Proceedings of the 4th Passive and Active Measurement Workshop PAM 2003*, Apr. 2003.

[23] Intel, “Interrupt Moderation Using Intel Gigabit Ethernet Controllers,” available at `http://www.intel.com/design/network/applnots/ap450.pdf(2003)`.

[24] Syskonnect, “SK-NET GE Gigabit Ethernet Server Adapter,” available at http://www.syskonnect.com/syskonnect/technology/SK-NET_GE.PDF(2003).

[25] C. L. T. Man, G. Hasegawa, and M. Murata, “Available bandwidth measurement via TCP connection,” in *Proceedings of the 2nd Workshop on End-to-End Monitoring Techniques and Services E2EMON*, Oct. 2004.

[26] Agilent Technologies, “Mixed packet size throughput.” available at http://advanced.comms.agilent.com/n2x/docs/journal/JTC_003.html.

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

[28] T. Kelly, “Scalable TCP: Improving performance in highspeed wide area networks,” *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83–91, Apr. 2003.

[29] C. Jin, D. X. Wei, and S. H. Low, “FAST TCP for High-Speed Long-Distance networks,” June 2003. available at http://netlab.caltech.edu/pub/papers/draft-jwl-tcp-fast-01.txt.

[30] “NS Simulation result of FAST TCP.” available at http://netlab.caltech.edu/~weixl/research/summary/fast-ns2/.