# A New Congestion Control Mechanism of TCP with Inline Network Measurement

Tomohito Iguchi, Go Hasegawa, and Masayuki Murata

Graduate School of Information Science and Technology, Osaka University
1-5 Yamadaoka, Suita, Osaka 565-0871, Japan
{t-iguti, hasegawa, murata}@ist.osaka-u.ac.jp

**Abstract.** In this paper, we propose a novel congestion control mechanism of TCP, by using an inline network measurement technique. By using information of available bandwidth of a network path between sender and receiver hosts, we construct quite a different congestion control mechanism from the traditional TCP Reno and its variants, based on logistic and Lotka-Volterra models from biophysics. The proposed mechanism is intensively investigated through analysis and simulation evaluations, and we show the effectiveness of the proposed mechanism in terms of scalability with the network bandwidth, convergence time, fairness among connections, and stability.

## 1 Introduction

Transmission Control Protocol (TCP) is the de facto standard transport layer protocol of the Internet. It was first designed in the 1970s, and the first Request for Comments (RFC) on TCP was released in 1981 [1]. Since the Internet has undergone such developmental changes as link bandwidth and number of nodes, TCP has also been frequently modified and enhanced according to such changes in the network.

One of the most important functions of TCP is its congestion control mechanism [2]. Its main purpose is to avoid and resolve network congestion, and to distribute network bandwidth equally among competing connections. TCP employs a window-based congestion control mechanism that adjusts data transmission speed by changing the window size. TCP's window updating mechanism is based on an Additive Increase Multiplicative Decrease (AIMD) policy: a TCP sender continues increasing window size additively until it detects a packet loss(es) and decreases it multiplicatively when a packet loss occurs. In [3], the authors argue that an AIMD policy is suitable for efficient and fair bandwidth usage in a distributed environment.

However, there are many problems in the congestion control mechanism of the current version of TCP (TCP Reno), which have emerged with increases of heterogeneity and the complexity of the Internet ([4-6] for some instances). The main reason is the fixed AIMD parameter values in increasing/decreasing window size, whereas they should be changed according to the network environment. For example, many previous papers [7-9] described that the throughput of TCP connections decreases when it traverses wireless links, since TCP cannot distinguish a congestion-oriented packet loss and a wireless-oriented (link loss and/or handoff) packet loss. In this case, the AIMD parameters, especially the decreasing parameters, must be changed dynamically according to the origins of the packet loss.

Another problem is the low throughput of TCP connections in high-speed and long-delay networks. In [10], the authors argued that a TCP Reno connection cannot fully utilize the link bandwidth of such networks, since the increasing parameter (1 packet per a Round Trip Time (RTT)) is too small and the decreasing parameter, which halves the window size when a packet loss occurs, is too large for networks with a large bandwidth-delay product.

Although there are many solutions against the above problems [8-13], almost all inherit the basic mechanism of the congestion control mechanism of TCP: the AIMD mechanism triggered by the detection of packet losses in the network. Most previous papers focused on changing the AIMD parameters according to the network environment. Since those methods may employ ad hoc modifications for a certain network situation, their performance is not clear when applied to other network environments.

TCP's performance is incomplete because the TCP sender does not have an effective mechanism to recognize the available bandwidth of the network path between sender and receiver hosts. In a sense, a traditional TCP Reno can be considered 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 since it only increases window size until a packet loss occurs. In other words, it induces packet losses to obtain information about the available bandwidth(-delay product) of the network. All modified versions of TCP using AIMD policy contain this essential problem.

If a TCP sender recognizes an available bandwidth quickly and adequately, we can create a further better mechanism for congestion control in TCP. Many measurement tools have been proposed in the literature [14-16] to measure the available bandwidth of network paths. However, we cannot directly employ those existing methods into TCP mechanisms since they utilize a lot of test probe packets; they also require a long time to obtain one measurement result. Fortunately, we have a method called Inline measurement TCP (ImTCP) that does not include these problems [17, 18]. It does not inject extra traffic into the network, and it estimates the available bandwidth from data/ACK packets transmitted by an active TCP connection in an inline fashion. Furthermore, the ImTCP sender can obtain the information of available bandwidth every 1–4 RTT that follows well the traffic fluctuation of the underlying IP network. Therefore, we can make a novel congestion control mechanism of TCP by using an inline network measurement mechanism.

In this paper, we propose a new congestion control mechanism of TCP that utilizes available bandwidth information obtained from inline measurement techniques. The proposed mechanism does not use ad hoc algorithms such as TCP Vegas [19], instead employs algorithms which have a mathematical background, by which we are able to mathematically discuss and guarantee its behavior even though it poses a simplification of the target system. More importantly, it becomes possible to give a reasonable background on our selection of control parameters within TCP, instead of conducting intensive computer simulation and/or choosing parameters in an ad-hoc fashion. We borrowed the algorithm from biophysics; a logistic equation and a Lotka-Volterra competition model [20] that describe changes in the population of species are applied to the window updating mechanism of our TCP. This application can be done by considering the number of a single species as a window size of a TCP connection, a carrying capacity as a bottleneck link bandwidth, and interspecific competition among species as a bandwidth share among competing TCP connections. We present in detail how to apply the logistic equation and the Lotka-Volterra competition model to the congestion control algorithm of TCP as well as analytic investigations of the proposed algorithm. Then, we can utilize the existing discussions and results on various characteristics of the model, including stability, fairness, and robustness. Giving those characteristics to TCP is our main purpose of the current study. We also present some preliminary simulation results to evaluate the proposed mechanism and show that, compared with traditional TCP Reno and other TCP variants it utilizes network bandwidth effectively, quickly, and fairly.

The rest of this paper is organized as follows. In Section 2, we import two mathematical models from biophysics: the logistic equation and the Lotka-Volterra competition model. The transition of those models to the data transmission rate control algorithm in computer networks is presented. Then we propose a new congestion control mechanism with inline network measurement and discuss its characteristics in Section 3. In Section 4, we show some simulation results to evaluate the performance of the proposed mechanism. We finally conclude this paper and offer future work in Section 5.

## 2 Mathematical Models Applied to a Congestion Control Mechanism

In this section, we briefly summarize the mathematical models from biophysics utilized by our proposed mechanism to control the congestion window size of a TCP connection.

### 2.1 Logistic Equation

The logistic equation is a formula that approximates the evolution of the population of a species over time. Generally, the increasing rate of a species population becomes larger as the species population becomes larger. However, since there are various restrictions about living environments, the environmental capacity, which is the maximum of the population of the species, exists. The logistic equation approximates such changes in the species population:

$$\frac{dN}{dt} = \varepsilon \left(1 - \frac{N}{K}\right) N$$

where $t$ is time, $N$ is the number of species, $K$ is the carrying capacity of the environment, and $\varepsilon$ is the intrinsic growth rate of the species. Fig. 1 shows changes of the species population ($N$) as a function of time where $K$ = 100 and $\varepsilon$ changes to 0.6, 1.8, 2.4, and 3.0. Looking at lines with $\varepsilon$ = 0.6 and 1.8, we can observe the following characteristics of the logistic equation; when $N$ is much smaller than $K$, the increasing speed of $N$ becomes larger as $N$ increases. On the other hand, when $N$ becomes close to $K$, the increasing rate decreases and $N$ converges to $K$. As $\varepsilon$ increases from 0.6 to 1.8, the convergence time becomes small at an expense of some overshoot. When $\varepsilon$ is 2.4 or 3.0, however, $N$ does not converge to $K$ and remains unstable. This is a well-known characteristic of the logistic equation, where $\varepsilon$ should be less than 2.0 to successfully converge $N$ to $K$.

We consider that the increasing trend of $N$ in the logistic equation can be applied to the control of the data transmission speed of TCP. That is, by considering $N$ as the transmission rate of a TCP sender and $K$ as the physical bandwidth of the bottleneck link, rapid and stable link utilization can be realized. However, the logistic equation describes the population of one species, whereas there are two or more TCP connections in the practical network. In the next subsection, we introduce an extended model that describes the changes of the population of two species with interaction between themselves.

### 2.2 Lotka-Volterra Competition Model

The Lotka-Volterra competition model is a famous model for the population growth of two species including interspecific competition between them. In the model, a logistic equation is modified to include the effects of interspecific competition as well as intraspecific competition. The Lotka-Volterra model of interspecific competition is comprised of the following equation for the population of species $i$ and $j$:

$$\frac{dN_i}{dt} = \varepsilon_i \left(1 - \frac{N_i + \gamma_{ij} \cdot N_j}{K_i}\right) N_i \tag{1}$$

where $N_i$, $K_i$, and $\varepsilon_i$ are the population, the environmental capacity, and the intrinsic growth rate of the species $i$, respectively. $\gamma_{ij}$ is the ratio of the competition coefficient of species $i$ on species $j$.

In this model, the population of species $i$ and $j$ does not always converge to some value larger than 0, and in some cases one of them sometimes dies out. It depends on the value of $\gamma_{ij}$. It is a well-known characteristic that when the following condition is satisfied, the two species survive in the environment:

$$\gamma_{ij} < \frac{K_i}{K_j} \tag{2}$$

Assuming that the two species have the same characteristics, they have the same values of $K$, $\varepsilon$, and $\gamma$, Equation (1) can be written as follows:

$$\frac{dN_i}{dt} = \varepsilon \left(1 - \frac{N_i + \gamma \cdot N_j}{K}\right) N_i \tag{3}$$

Note that the condition in Equation (2) becomes $\gamma < 1$. Fig. 2 shows the change in the population of the two species by using Equation (3), where species 2 join the environment in 10 time units after species 1. We can observe from this figure that the population of the two species converges quickly at the same value, which is considered an ideal behavior for the control of the transmission rate in computer networks.

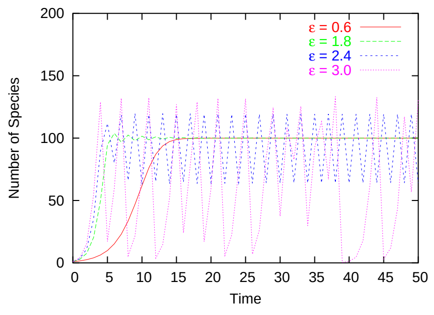


**Fig. 1.** Logistic equation ($K = 100$)

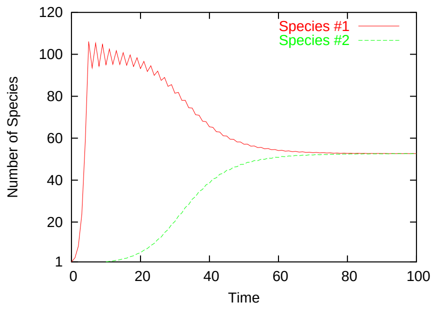


**Fig. 2.** Changes in species population with Lotka-Volterra competition model ($\varepsilon = 1.95$, $\gamma = 0.9$, $K = 100$)

## 2.3 Application to Transmission Rate Control Algorithm

In a practical network there are usually more than two TCP connections sharing a network bandwidth. We can easily extend Equation (3) for $n$ species as follows:

$$\frac{dN_i}{dt} = \varepsilon\left(1 - \frac{N_i + \gamma \cdot \sum_{j=1, i \neq j}^{n} N_j}{K}\right) N_i \tag{4}$$

When we consider Equation (4) as the control algorithm for the data transmission rate for TCP connection $i$ ($N_i$), 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 in the current Internet. However, when we obtain the available bandwidth for connection $i$ with the inline measurement mechanism [17], we can approximate the sum of the data transmission rates of all of other connections as follows:

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

Thus, Equation (4) becomes as follows;

$$\frac{dN_i}{dt} = \varepsilon\left(1 - \frac{N_i + \gamma \cdot (K - A_i)}{K}\right) N_i \tag{5}$$

where $N_i$, and $A_i$ are the data transmission rate and the available bandwidth for connection $i$. $K$ is the physical bandwidth of the bottleneck link, where we assume that all connections share the same bottleneck link. Our proposed mechanism assumes that we can obtain $A_i$ and $K$ by using the inline network measurement. The current version of ImTCP [17, 18] can measure $A_i$ with high accuracy in various conditions of the network. Therefore, we consider that the proposed mechanism can set $A_i$ by ImTCP. On the other hand, because a physical bandwidth measurement algorithm is now under consideration, we directly set $K$ to the correct value. However, the change of the physical bandwidth of the network path is smaller than that of the available bandwidth, so we can expect that the measurement error is also smaller. Hence, we consider that the effect of the measurement error of the physical bandwidth ($K$) on the performance of the proposed mechanism is negligible when we use the measurement results of the physical bandwidth. In our proposed mechanism, we use the above equation as a rate control algorithm of a TCP sender host. In the next section, we present the control algorithm of the window size of the TCP sender host, using the above equation.

## 3 Proposed Congestion Control Mechanism of TCP

A TCP sender controls its data transmission rate by changing its window size when it receives an ACK packet. Here we convert Equation (5) to obtain an increasing algorithm of the window size in TCP. The window size of connection $i$, $w_i$, is calculated from $N_i$, the transmission rate, by the following simple equation:

$$w_i = N_i \cdot base\_rtt_i$$

where $base\_rtt_i$ is the minimum value of the RTTs of connection $i$. Then Equation (5) can be rewritten as follows:

$$\frac{dw_i}{dt} = \varepsilon \left(1 - \frac{w_i + \gamma \cdot base\_rtt_i \cdot (K - A_i)}{K \cdot base\_rtt_i}\right) w_i$$

We next change the equation in RTT.

$$\frac{dw_i}{drtt} = \varepsilon \left(1 - \frac{w_i + \gamma \cdot base\_rtt_i \cdot (K - A_i)}{K \cdot base\_rtt_i}\right) w_i \tag{6}$$

Finally, we derive the amount of the increase in window size when an ACK packet is received at the TCP sender by considering that $w_i$ ACK packets are received in one RTT:

$$\Delta w_i = \varepsilon \left(1 - \frac{w_i + \gamma \cdot base\_rtt_i \cdot (K - A_i)}{K \cdot base\_rtt_i}\right)$$

This is the fundamental equation in increasing window size in our proposed mechanism. Since this equation requires the measurements of the available bandwidth and physical bandwidth of a network path, we use the same algorithm as TCP Reno for window updating algorithm until the measurement results are obtained through inline network measurements. In cases of packet loss(es), window size is decreased in identical way to TCP Reno. When a timeout occurs, sender TCP discards all measurement results, window size is reset to 1, and the slow-start phase begins as a TCP Reno sender does.

## 4 Simulation Results

In this section, we present some simulation results to evaluate the performance of the congestion control mechanism proposed in Section 3. We used ns-2 [21] for the simulation experiments. The traditional TCP Reno, HighSpeed TCP (HSTCP) [10] and

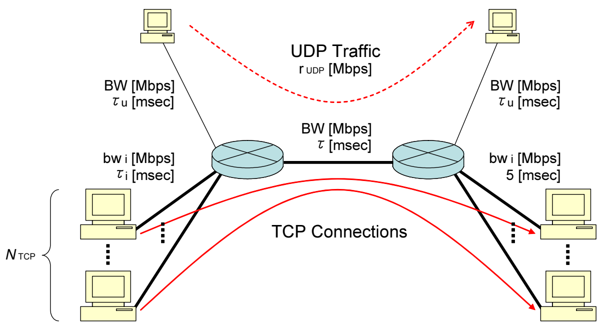


**Fig. 3.** Network topology in simulation experiments

Scalable TCP [12] were chosen for performance comparison. In our proposed mechanism, the available bandwidth information was obtained through an inline measurement mechanism. Note that we directly give the physical bandwidth information to the TCP sender since there is currently no effective mechanism to measure the physical bandwidth in an inline fashion. We set $\varepsilon = 1.95$ and $\gamma = 0.9$ for the proposed mechanism. For HSTCP, we use the parameters described in [10].

The network model used in the simulation is depicted in Fig. 3. It consists of sender/receiver hosts, two routers, and links between the hosts and routers. $N_{\mathrm{tcp}}$ TCP connections are established between TCP sender $i$ and TCP receiver $i$. For creating the background traffic, we inject UDP packets at the rate of $r_{\mathrm{udp}}$ into the network. That is, $N_{\mathrm{tcp}}$ TCP connections and an UDP flow share the bottleneck link between the two routers. The bandwidths of the bottleneck link and the access link for the UDP sender/receiver are all set to $BW$, and the propagation delays are $\tau$ and $\tau_{\mathrm{u}}$, respectively. The bandwidth and the propagation delay of the access link for TCP sender $i$ are $bw_i$ and $\tau_i$, respectively. We deployed a Taildrop discipline at the router buffer, and the buffer size is set to twice the bandwidth-delay product of the bottleneck link between the two routers.

We first confirm the fundamental behavior of the proposed mechanism with one TCP connection ($N_{\mathrm{tcp}} = 1$). Fig. 4 shows the change in the window size of TCP Reno, HSTCP, Scalable TCP and the proposed mechanism, where we set $bw_1 = 100$ Mbps, $\tau_1 = 5$ msec, $BW = 100$ Mbps, $\tau = 40$ msec, $\tau_{\mathrm{u}} = 5$ msec and $r_{\mathrm{udp}} = 50$ Mbps. This result shows that TCP Reno, HSTCP and Scalable TCP connections experience periodic packet losses due to buffer overflow, since they continue increasing the window size until packet loss occurs. On the other hand, the window size of the proposed mechanism converges to an ideal value quickly and no packet loss occurs. Furthermore, the increasing speed is much larger than that of HSTCP and Scalable TCP, meaning that the proposed mechanism effectively utilizes the link bandwidth.

We next investigate the scalability with link bandwidth of the proposed mechanism by checking the convergence time, which is defined as the time it takes for the TCP connection to utilize 99% of the link bandwidth. We set $N_{\mathrm{tcp}} = 1$, $\tau_1 = 5$ msec, $\tau = 40$ msec and $\tau_{\mathrm{u}} = 5$ msec. Fig. 5 shows the change of the convergence time when we change $BW$

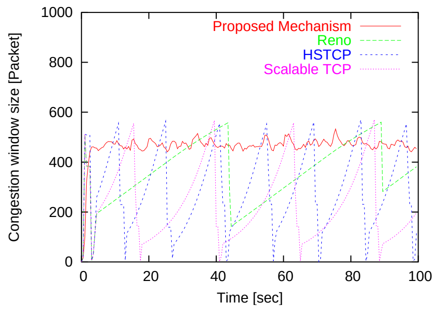


**Fig. 4.** Change of window size in one connection case

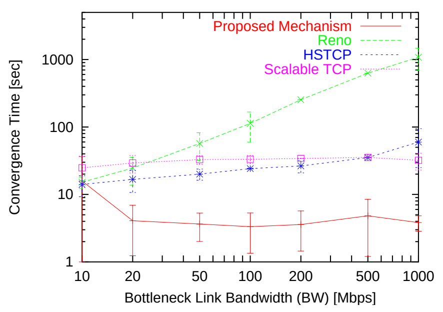
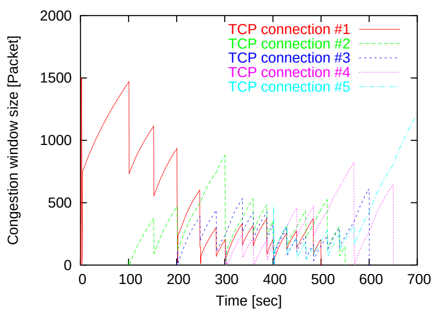


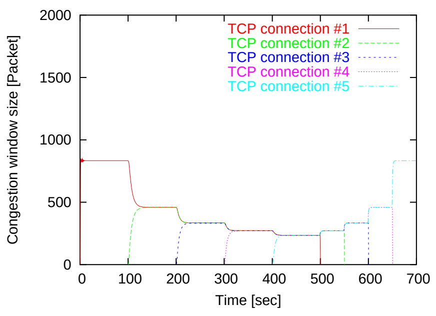
**Fig. 5.** Convergence time as a function of bottleneck link bandwidth

from 10 Mbps to 1 Gbps, where $r_{\mathrm{udp}}$ is set to $(0.2 \cdot BW)$ Mbps and $bw_1$ is set equal to $BW$. In the figure, the average values and the 95% confidence intervals for 10 simulations experiments are shown. From this figure, we can see that the TCP Reno connection requires quite a large 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 larger convergence time requires 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 Scalable TCP, the convergence time remains constant regardless of the link bandwidth, which was confirmed in [12]. However, it is quite larger than that of the proposed mechanism. The proposed mechanism, however, keeps the smallest and the almost constant convergence time regardless of the link bandwidth, which shows good scalability with the network bandwidth. The confidence interval of the proposed mechanism is large because the measurement results have some errors.

Finally We investigate the adaptability and fairness of the proposed mechanism by investigating the effect of changes in the number of TCP connections. We set $N_{\mathrm{tcp}} = 5$, $bw_i = 100$ Mbps, $\tau_i = 5$ msec $(1 \leq i \leq 5)$, $BW = 100$ Mbps and $\tau = 40$ msec. We do not inject UDP traffic into the network. TCP connections 1–5 join the network at 0, 100, 200, 300, and 400 sec and stop sending data packets at 500, 550, 600, 650, and 700 sec, respectively. Fig. 6 shows change of window size for the five TCP connections as a function of simulation time in TCP Reno (Fig. 6(a)) and the proposed mechanism (Fig. 6(b)). This figure shows that TCP Reno cannot maintain the fairness among connections at all, mainly because it takes long time for all connections to have the fair window sizes. Furthermore, TCP Reno connections suffer from cyclic packet losses. On the other hand, the proposed mechanism converges the window size very quickly and no packet loss occurs when a new connection joins the network. Furthermore, when the TCP connection leaves the network, the proposed mechanism quickly fill the unused bandwidth.



(a) TCP Reno



(b) Proposed mechanism

**Fig. 6.** Effect of changes in number of connections

## 5 Conclusion and Future Work

In this paper, we proposed a new congestion control mechanism of TCP which utilized an inline network measurement technique. The proposed mechanism is based on the logistic equation and the Lotka-Volterra competition model that represents population changes of species. We applied the two models to the transmission rate control in the computer network and constructed a new algorithm to change the window size of the TCP connections. Through analysis and simulation evaluations, we confirmed the effectiveness of the proposed mechanism for scalability, convergence speed, fairness, stability, and so on.

We consider that by obtaining the important information for congestion control, for example, the available and physical bandwidth of the network path, we can create a much better mechanism for the congestion control of TCP. As research on inline network measurement techniques advances, other kinds of congestion control for the Internet will be realized that enhance the performance of TCP. The mechanism proposed in this paper is the first step in that challenge.

## Acknowledgement

This work was partly supported by the Ministry of Education, Science and Culture, Grant-in-Aid for (A)(16680003), 2004.

## References

1. J. B. Postel, "Transmission control protocol," *Request for Comments 793*, Sept. 1981.
2. W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Reading, Massachusetts: Addison-Wesley, 1994.

3. D.-M. Chiu and R. Jain, “Analysis of the increase and decrease algorithms for congestion avoidance in computer networks,” *Journal of Computer Networks and ISDN Systems*, pp. 1–14, June 1989.
4. 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.
5. 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.
6. L. Guo and I. Matta, “The War Between Mice and Elephants,” *Technical Report BU-CS-2001-005*, May 2001.
7. 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.
8. K. Xu, Y. Tian, and N. Ansari, “TCP-Jersey for wireless IP communications,” *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 747–756, May 2004.
9. E. H.-K. Wu and M.-Z. Chen, “JTCP: Jitter-based TCP for heterogeneous wireless networks,” *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 757–766, May 2004.
10. S. Floyd, “HighSpeed TCP for large congestion windows,” *RFC 3649*, Dec. 2003.
11. M. Gerla, M. Y. Sanadidi, R.Wang, A. Zanella, C. Casetti, and S. Mascolo, “TCP Westwood: Congestion window control using bandwidth estimation,” in *Proceedings of IEEE GLOBECOM ’01*, Nov. 2001.
12. T. Kelly, “Scalable TCP: Improving performance in highspeed wide area networks,” in *proceedings of PFLDnet ’03: workshop for the purposes of discussion*, Feb. 2003.
13. C. Jin, D. X. Wei, and S. H. Low, “FAST TCP: motivation, architecture, algorithms, performance,” in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.
14. B. Melander, M. Bjorkman, and P. Gunningberg, “A new end-to-end probing and analysis method for estimating bandwidth bottlenecks,” in *Proceedings of IEEE GLOBECOM 2000*, Nov. 2000.
15. 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.
16. 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.
17. M. L. T. Cao, “A study on inline network measurement mechanism for service overlay networks,” Master’s thesis, Graduate School of Information Science, Osaka University, Feb. 2004.
18. M. L. T. Cao, G. Hasegawa, and M. Murata, “Available bandwidth measurement via TCP connection,” to be presented at IFIP/IEEE MMNS 2004, Oct. 2004.
19. L. S. Brakmo, S. W.O’Malley, and L. L. Peterson, “TCP Vegas: New techniques for congestion detection and avoidance,” in *Proceedings of ACM SIGCOMM’94*, pp. 24–35, Oct. 1994.
20. J. D. Murray, *Mathematical Biology I: An Introduction*. Springer Verlag Published, 2002.
21. The VINT Project, “UCB/LBNL/VINT network simulator - ns (version 2).” available from `http://www.isi.edu/nsnam/ns/`.