IEICE TRANS. COMMUN., VOL.E99–B, NO.2 FEBRUARY 2016

PAPER

# Proof Test of Chaos-Based Hierarchical Network Control Using Packet-Level Network Simulation

**Yusuke SAKUMOTO**[†a)], **Chisa TAKANO**[††b)], **Masaki AIDA**[†c)], ***Members***, ***and*** **Masayuki MURATA**[†††d)], ***Fellow***

**SUMMARY** Computer networks require sophisticated control mechanisms to realize fair resource allocation among users in conjunction with efficient resource usage. To successfully realize fair resource allocation in a network, someone should control the behavior of each user by considering fairness. To provide efficient resource utilization, someone should control the behavior of all users by considering efficiency. To realize both control goals with different granularities at the same time, a hierarchical network control mechanism that combines microscopic control (i.e., fairness control) and macroscopic control (i.e., efficiency control) is required. In previous works, Aida proposed the concept of chaos-based hierarchical network control. Next, as an application of the chaos-based concept, Aida designed a fundamental framework of hierarchical transmission rate control based on the chaos of coupled relaxation oscillators. To clarify the realization of the chaos-based concept, one should specify the chaos-based hierarchical transmission rate control in enough detail to work in an actual network, and confirm that it works as intended. In this study, we implement the chaos-based hierarchical transmission rate control in a popular network simulator, ns-2, and confirm its operation through our experimentation. Results verify that the chaos-based concept can be successfully realized in TCP/IP networks.

***key words:*** *chaos, hierarchical network architecture, congestion control, TCP global synchronization, TCP (transmission control protocol), RED (random early detection)*

## 1. Introduction

In computer networks (hereinafter simply referred to as "networks"), many users have to share the limited resources (e.g., link bandwidth). Such networks need control mechanism to realize fair resource allocation among users while providing efficient resource utilization. To realize fair resource allocation in a network, someone should control the behavior of each user by considering fairness; to provide efficient resource utilization, someone should control the behavior of all users by considering efficiency. These controls are characterized by the granularity of their control targets. Since each user is essentially a microscopic control target, the fairness control corresponds to microscopic control, whereas efficiency control for all users corresponds to macroscopic control. Therefore, as shown in Fig. 1(a), to realize fairness and efficiency simultaneously, we must realize a hierarchical network control mechanism that combines both microscopic and macroscopic controls.

In [1], Aida proposed the concept of chaos-based hierarchical network control. Chaos is the state of disorder observed at a microscopic scale in a deterministic system, but has a stable structure at a macroscopic scale. Aida's concept utilizes such a hierarchical structure of chaos, as shown in Fig. 1(b)), to design hierarchical network control. The hierarchical structure of chaos is briefly described as follows. We denote the network state by a point in a multidimensional space, and represent the time evolution of the network state by the trajectory of this point. In a deterministic system, the trajectory for an initial point is determined without randomness; however, if a deterministic system is chaotic, the future point is unpredictable unless its initial point is known with perfect accuracy. This is caused by *sensitivity to initial conditions of chaos*, which means that the difference between points that originate from slightly different points increases exponentially with time. Due to the sensitivity to initial conditions, any trajectory in a chaotic system appears to be random at a microscopic scale as illustrated in Fig. 1(b); however, regardless of the initial point, all trajectories are constrained by a common stable attractor (i.e., a chaos attractor) at a macroscopic scale. This property is called the *structural stability of chaos*. In [1], Aida explained that the hierarchical structure of chaos is useful for designing hierarchical network control schemes.

As an application of the above chaos-based concept, Aida also proposed a fundamental framework for hierarchical transmission rate control based on the chaos of coupled relaxation oscillators [2] (hereinafter simply referred to as "coupled oscillators"). He first designed the control rule for the transmission rate of each flow on the basis of the deterministic rule that underlies coupled oscillators. Due to the sensitivity to the initial conditions of chaos, transmission rates of flows appear to be random; thus, this control avoids severe network congestion. Such randomization would be effective for various network problems (e.g., the TCP global synchronization problem [3] and bufferbloat [4]). More specifically, this control rule deterministically restricts the flow with the largest transmission rate among all flows sharing a bottleneck link; thus, realizing fairness among flows. Note that the control rule involves a parameter that impacts the efficient usage of bottleneck links. Aida then designed the parameter setting necessary to achieve efficient usage

Manuscript received June 15, 2015.
Manuscript revised September 28, 2015.
[†]The authors are with Tokyo Metropolitan University, Hino-shi, 191-0065 Japan.
[††]The author is with Hiroshima City University, Hiroshima-shi, 731-3166 Japan.
[†††]The author is with Osaka University, Suita-shi, 565-0871 Japan.
a) E-mail: sakumoto@tmu.ac.jp
b) E-mail: takano@hiroshima-cu.ac.jp
c) E-mail: aida@tmu.ac.jp
d) E-mail: murata@osaka-u.ac.jp
DOI: 10.1587/transcom.2015EBP3256

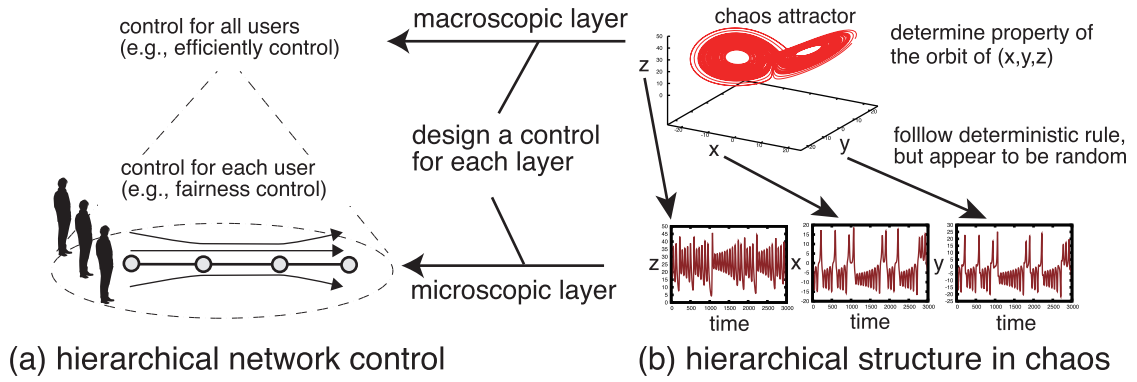



**Fig. 1** Hierarchical network control vs. the hierarchical structure of chaos.

on the basis of the chaos attractor established by the control rule. Since all flows use this parameter setting, it corresponds to controlling all flows. To confirm the feasibility of this chaos-based concept, we intend to specify the chaos-based hierarchical transmission rate control scheme such that it can work in an actual network, and confirm that it operates as intended.

In this paper, we describe our implementation of chaos-based hierarchical transmission rate control in the popular network simulator ns-2 [5]; furthermore, we experimentally confirm its operation. In our experiments, we clarify that (a) fairness among users is realized when applying the chaos-based control rule to each flow, and (b) bottleneck links are efficiently utilized when using the parameter setting proposed in [1]. Our results verify the possibility of successfully realizing the chaos-based concept in TCP/IP networks.

The rest of the paper is organized as follows. In Sect. 2, we introduce another hierarchical structure used in network control design as well as other network controls that are aware of chaos. In Sect. 3, we explain the chaos-based hierarchical transmission rate control. In Sect. 4, we provide details regarding our implementation of chaos-based hierarchical transmission rate control in ns-2. In Sect. 5, we show experimental results obtained from our implementation. In Sect. 6, we conclude this paper and describe scope for future work.

## 2. Related Work

### 2.1 Hierarchical Structure of Network Control

The standard network architecture — the Open Systems Interconnection model — has a hierarchical structure consisting of multiple abstraction layers to hide implementation details. Thanks to this hierarchical structure, developers can easily implement network controls that work within a specific abstraction layer; however, problems due to the hierarchical structure have been reported, and ad hoc solutions have been proposed in [6]–[9].

To solve the problems caused by the standard network architecture using a fundamental approach, Aida proposed a novel hierarchical network architecture inspired by systems in nature [10]. This network architecture has multiple layers to reflect different time/space scale dependencies, and is hierarchized by combining the behaviors of the microscopic and macroscopic layers quite similar to nature system models. As an example, to combine these behaviors, Aida introduced a renormalization group for networks in [10]. Hierarchical network control based on chaos [1] is another example of hierarchical network control based on Aida's hierarchical network architecture.

Some hierarchical network control mechanisms based on the hierarchical network architecture of [10] have been proposed [11]–[13]. In [11], [12], the authors use a partial differential equation (PDE) to combine the behaviors of multiple layers. A PDE and its solution respectively correspond to a microscopic rule and its macroscopic property. They designed the microscopic control based on a PDE, and the macroscopic control based on initial and boundary conditions that determine PDE solutions. In [13], the authors designed a microscopic control (i.e., autonomous decentralized control) based on Markov Chain Monte Carlo, which is a design method to control the probability distribution of a system macroscopic variable. The hierarchical network controls introduced in [11]–[13] were built by designing a microscopic rule through which macroscopic behavior can be indirectly controlled. By contrast, hierarchical network control based on chaos [1] is designed to be able to simultaneously control different behaviors (e.g., fairness and efficiency) at both the microscopic and macroscopic layers.

### 2.2 Chaos in Network Controls

The behaviors of a particular network controls show chaotic behaviors [14]–[16]. In [14], Chen et al. reported that the deterministic model of TCP congestion control is chaotic; as such, they designed a mechanism to stabilize the TCP congestion control by removing the chaotic behavior. While in existing research, it is common to remove the behavior of chaos from unstable network controls, Aida's concept attempts to deliberately utilize chaos in network control.

## 3. Chaos-Based Hierarchical Transmission Rate Control

In [1], as an application of the aforementioned chaos-based concept, Aida proposed a fundamental framework for hierarchical transmission rate control based on the chaos of coupled oscillators [2], which is a chaotic deterministic system. Each oscillator has displacement as its state, and the interaction among oscillators generates chaos with oscillator displacements appearing to be random at a microscopic scale.

### 3.1 Chaos-Based Control Rule for the Transmission Rate of Each Flow

In [1], Aida designed a control rule for the transmission rate of each flow according to the deterministic rule that models the displacement of each oscillator. The designed control rule adjusts transmission rate $r_i(t)$ of flow $i$ by using the following steps.

1. At the start time for flow $i$, $r_i(t)$ is set to 0, and increasing rate $s_i(t)$ of $r_i(t)$ is set to $\rho$ ($\rho > 0$).
2. Flow $i$ increases $r_i(i)$ by $s_i(t)$ per unit time until the condition of either step 3 or step 4 is satisfied.
3. If $r_i(t) = r_{\max}$, flow $i$ decreases from $r_i(t)$ to $\alpha\, r_i(t)$ where $0 \le \alpha < 1$, and $s_i(t)$ is set to $\rho$. Return to step 2.
4. If $r_j(t) = r_{\max}$ for flow $j$ that has an adjacent relationship to flow $i$, flow $i$ sets $s_i(t)$ to $\beta\rho$ where $0 < \beta < 1$. Return to step 2.

Note that threshold $r_{\max}$ is a positive parameter of the chaos-based control rule, and strongly affects the efficient utilization of bottleneck links. The dynamics of transmission rate $r_i(t)$ and the dynamics of increasing rate $s_i(t)$ are described by

$$\frac{\mathrm{d}\, r_i(t)}{\mathrm{d}\, t} = s_i(t) - s_i(t)\,(1-\alpha)\, r_{\max}\, \delta(r_{\max} - r_i(t)), \tag{1}$$

$$\frac{\mathrm{d}\, s_i(t)}{\mathrm{d}\, t} = \sum_{j \in \partial i} s_j(t)\ (\beta\rho - s_i(t))\ \delta(r_j(t) - r_{\max}) + s_i(t)\,(\rho - s_i(t))\,\delta(r_i(t) - r_{\max}), \tag{2}$$

where $\delta(x)$ is the delta function, and $\partial i$ is the set of flows that have an adjacent relationship to flow $i$. Flows in $\partial i$ are selected from the set of flows sharing a link according to a certain rule.

Next, we provide an execution example of the above chaos-based control rule. Figure 2 shows time series data of the transmission rates of flows A and B following the chaos-based control rule. In the execution example, flow A starts its transmission slightly earlier than flow B. Immediately after starting the transmission of flow B, transmission rates of flows A and B are approximately the same; however, they appear to be random over time. This property helps to prevent severe network congestion. The fairness between flows A and B seems to be realized since a flow with a higher transmission rate is always decreased according to step 3. In the execution example, flows appear to be random, but the trajectory of $(r_A(t), r_B(t))$ is constrained within the Lorenz attractor, as shown in Fig. 3 [2]. This is explained by the following reason. There are two combinations of $(s_A(t), s_B(t))$; thus, a trajectory of $(r_A(t), r_B(t))$ traverses the two vector fields shown in Fig. 4. In the figure, when a point of $(r_A(t), r_B(t))$ proceeds to the upper side or right side of a vector field, the point appears at the corresponding lower or left side. By pasting each pair of the corresponding sides shown in Fig. 4, we obtain a 3D structure similar to two jointed tori. The obtained structure is topologically equivalent to the Lorentz attractor [2]. In [1], Aida designed a setting method of threshold $r_{\max}$ on the basis of the chaos attractor, the details of which are described in the next subsection.

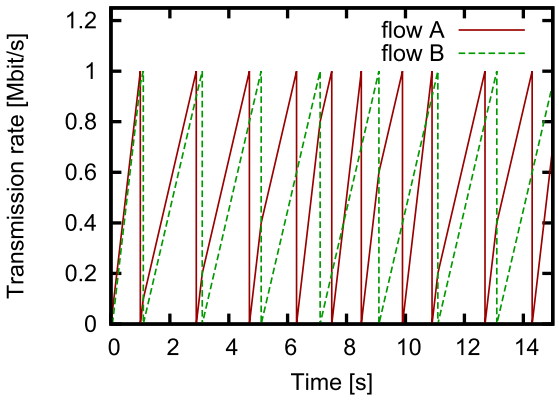



**Fig. 2** Time series of transmission rates of flows A and B when using the chaos-based control rule with $\alpha = 0.0, \beta = 0.5, \rho = 1.0$ Mbit/s$^2$, $r_{\max} = 1.0$ Mbit/s, $\partial A = \{B\}$, and $\partial B = \{A\}$.

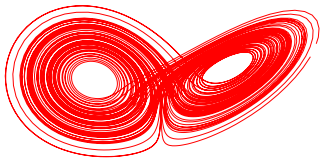


**Fig. 3** Lorenz attractor.

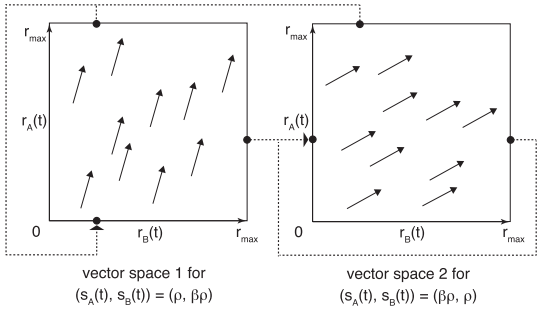



**Fig. 4** State space described by two vector fields.

### 3.2 Setting Threshold $r_{\max}$

Threshold $r_{\max}$ is the key to realizing the efficient utilization of bottleneck links. If $r_{\max}$ is set to too large a value, the source of each flow can inject many packets into the network. As a result, the total traffic rate of flows arriving at a bottleneck link can exceed the link bandwidth of the bottleneck link, and packet losses occur due to buffer overflow. Conversely, if $r_{\max}$ is set to too small a value, bottleneck links are not well utilized. Therefore, care should be taken when setting $r_{\max}$.

Threshold $r_{\max}$ is one of parameters that determines the shape of the chaos attractor established by the chaos-based control rule. In [1], Aida derives a setting for $r_{\max}$ on the basis of the chaos attractor as

$$r_{\max} = \frac{2\left[1 + (N-1)\beta\right]}{N\left[2 + (N-1)\beta + (N-1)\alpha\beta\right]}C, \tag{3}$$

where $N$ is the number of flows sharing a link, and $C$ is the link bandwidth.

### 3.3 Characteristics

In this subsection, we explain the characteristics of chaos-based hierarchical transmission rate control by comparing it with other transmission rate controls.

TCP transmits most of its traffic on the Internet, and the well-known TCP transmission rate control mechanism [17] attempts to avoid severe network congestion by monitoring packet losses in the network. In TCP control, the source of each TCP flow increases its transmission rate until packet loss is detected, whereupon the rate is then drastically decreased; however, such TCP control fails to avoid severe network congestion in certain situations [3]. One such situation, called *TCP global synchronization*, occurs when the time evolution of transmission rates of TCP flows sharing a bottleneck link are synchronized by TCP control. Owing to TCP global synchronization, severe network congestion and low utilization of the bottleneck link are alternately triggered. To solve TCP global synchronization problem, the time evolution of TCP transmission rates should be randomized.

Random early detection (RED) [18] is the most famous solution to the TCP global synchronization problem. RED randomly drops a packet at a bottleneck link, thus randomizing the transmission rates of TCP flows. Consequently, TCP control supported by RED is successful in avoiding severe network congestion and low utilization of a bottleneck link; however, RED causes a new undesirable situation. As an example, we consider the situation in which a bottleneck link is shared by one TCP flow with a transmission rate of 100 Mbit/s and 100 TCP flows with a transmission rate of 1 Mbit/s. In this situation, RED should drop a packet from the high-rate TCP flow with respect to fairness, but instead selects a packet from a low-rate TCP flow with a probability of 50% because 50% of packets arriving at the router are

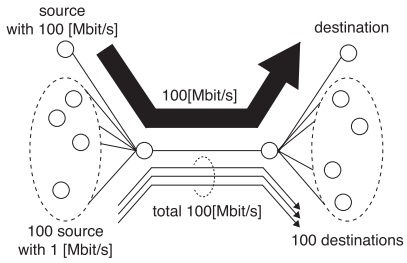



**Fig. 5** An example of unfairness caused by RED.

those of low-rate TCP flows. In this sense, RED imposes unfairness between high- and low-rate flows.

Chaos-based hierarchical transmission rate control can randomize transmission rates of flows by the property of chaos. Moreover, since chaos-based hierarchical transmission rate control deterministically restricts the flow with the largest transmission rate, it avoids the unfair situation caused by RED. Moreover, this restriction on the largest flow would be also effective to alleviate *bufferbloat* [4], which is a phenomenon in which the router at a bottleneck link persistently buffers too many packets.

## 4. Implementation in a Packet-Level Network Simulator

Here, we describe the fundamental framework of hierarchical transmission rate control based on chaos and implement it in the popular network simulator, ns-2 [5]. Network simulator ns-2 is designed on the basis of TCP/IP, and is often used to verify the operation of network control mechanisms in TCP/IP networks. Through experiments on ns-2, we clarify the feasibility of the concept of the chaos-based hierarchical network control proposed in [1].

### 4.1 Fundamentals

The steps of the chaos-based control rule explained in Sect. 3.1 should be executed at a router. If these steps are executed at the source host of a flow, the source host has to gather information regarding flows sharing a bottleneck link. In particular, to execute step 4 of the chaos-based control rule, the source host of flow $i$ has to gather transmission rates $r_j(t)$ of flow $j$ in $\partial i$; however, in general, it is difficult to exchange such information regarding flows between source hosts only using standard TCP functions. Even if source hosts can exchange flow information, the chaos-based control rule may fail to generate chaos because of the delay in executing step 4. To avoid this problem, we steps should be performed at a router. As shown in Fig. 6, each router calculates the transmission rates of flows according to the proposed steps, and notifies source hosts of calculated transmission rates using packet headers. To realize such notification via packet headers, we refer to the procedures of explicit congestion control (XCP) [19], [20], which passes information regarding transmission rates calculated by a router to a source host.

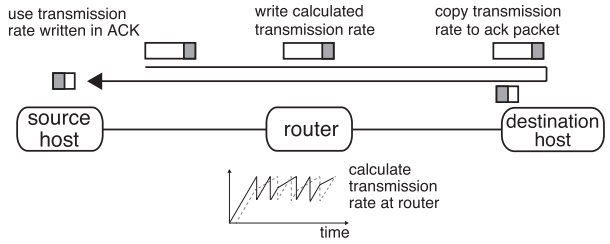



**Fig. 6** Notification of transmission rate calculated by a router to source host.

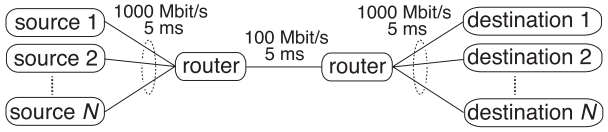



**Fig. 7** Dumbbell topology used in our experiments.

Each router sets threshold $r_{\max}$ by Eq. (3); here, the router uses the number of flows sharing its managed link, $N$, as well as its own link bandwidth $C$.

This study assumes that the chaos-based control mechanism is activated in all routers at links that may possibly become bottlenecks. We detail here the chaos-based hierarchical transmission rate control required to confirm its operation on the dumbbell topology shown in Fig 7. Efficient implementation and support of general topologies are beyond the scope of this paper.

### 4.2 Design Details

#### 4.2.1 Procedure for Source Hosts

Regarding the procedure for source hosts, each source host transmits packets according to XCP [20]. Each XCP source host restricts the number of packets available for simultaneous release by its congestion window size, and updates its congestion window size using the change amount in the header of ACK packets. Similar to XCP source hosts, each source host updates its congestion window size using header information (e.g., the transmission rate calculated by the router at a bottleneck link). In this update, each source host calculates its new congestion window size as the product of the notified transmission rate and the base round trip time (RTT), which is the RTT without queuing delay time. Some congestion control mechanisms [21], [22] also use the base RTT to control transmission rates.

#### 4.2.2 Procedure for Routers

Regarding routers, each router forwards packets according to XCP. We therefore explain the procedures that differ from XCP below.

Each router has a list of flows passing through its link. If a router receives a packet of a flow that is absent from its list, the router adds the flow to the bottom of its list. If a router receives no packets of a flow in its list within a certain time period, the router removes the flow from its list. If any updates to its list occur, the router resets $r_{\max}$ by Eq. (3). To set $\partial i$ of flow $i$, adjacency relationships among flows are determined by the entry order in the list. Namely, $\partial i$ if $N \geq 2$ is given by

$$\partial i = \begin{cases} \{o^{-1}(2)\} & \text{if } o(i) = 1 \\ \{o^{-1}(N-1)\} & \text{if } o(i) = N \\ \{o^{-1}(o(i) \pm 1)\} & \text{otherwise} \end{cases}, \tag{4}$$

where $o(i)$ is the function that returns the index number of flow $i$ in the list, and $o^{-1}(k)$ is the function that returns the identifier of the $k$-th flow in the list. In general, $o^{-1}(o(i)) = i$. If $N = 1$, $\partial i$ is $\emptyset$ (i.e., empty set). Consider the example when a router holds list $(A, B, C)$. In this example, $o(A) = 1$, $o(B) = 2$, and $o(C) = 3$. According to Eq. (4), $\partial A = \{o^{-1}(2)\} = \{B\}$, $\partial B = \{o^{-1}(1), o^{-1}(3)\} = \{A, C\}$ and $\partial C = \{o^{-1}(3-1)\} = \{B\}$.

Each router calculates the transmission rates of flows in its list according to the steps explained in Sect. 3.1. When the router receives a packet of a flow, it writes the transmission rate of the flow to the header of the received packet just as in an XCP router.

#### 4.2.3 Procedures for Destination Hosts

Regarding destination hosts, each destination host handles received packets according to XCP. Similar to XCP destination hosts, each destination host reads the header information of a received packet of a flow, and then writes the transmission rate of the flow to the header of the corresponding ACK packet.

## 5. Experimentation

We confirm the validity of the chaos-based hierarchical transmission rate control through experimentation. More specifically, we confirm that (a) fairness among flows is realized when using the chaos-based control rule, and (b) efficient usage of the bottleneck link is realized when using the parameter setting of threshold $r_{\max}$ proposed in [1]. Through our experiments, we will clarify the feasibility of realizing the chaos-based concept in TCP/IP networks.

Our experiments compare the performance of chaos-based control with that of RED. Note that RED randomizes flow behavior using a pseudo-random number generator, whereas chaos-based control randomizes them by the property of chaos. We investigate the differences between RED and chaos-based control in terms of fairness and efficiency. Through our experiments, we clarify the performance differences produced by both mechanisms to randomize flow behavior. Chaos-based control uses more information (i.e., the number of flows, $N$, and the link capacity $C$) than RED, so we expect that chaos-based control will outperform RED.

### 5.1 Experimental Setting

For our experimental setup, we use the dumbbell topology

**Table 1** Parameter settings.

| | | |
|---|---|---|
| chaos-based hierarchical transmission rate control | | |
| relaxation ratio $\alpha$ | 1/2 | |
| deceleration ratio $\beta$ | 1/2 | |
| increasing rate $\rho$ | $\sqrt{2}$ | Mbit/s$^2$ |
| RED | | |
| minimum threshold $\min_{\text{th}}$ | 100 | packet |
| maximum threshold $\max_{\text{th}}$ | 500 | packet |
| packet loss probability at $\max_{\text{th}}$ | 0.1 | |
| weight parameter $w_q$ | 0.002 | |
| common | | |
| number of sender, $N$ | 10 | |
| bandwidth of non-bottleneck link, $C_{\text{non}}$ | 1,000 | Mbit/s |
| bandwidth of bottleneck link, $C$ | 100 | Mbit/s |
| propagation delay of a link, $\tau$ | 5 ms | |
| buffer size of a link, $Q$ | 5,000 | packet |
| packet size (i.e., MSS) $P$ | 512 | byte |

shown in Fig. 7. The bottleneck link is shared by $N$ flows. When we use chaos-based control, queue management at the bottleneck link is set to Droptail. When we use RED for the queue management, transmission rates of flows are controlled by TCP Reno [17], and we use *TCP-Reno+RED* for notational distinction. Overall, we use the default parameters shown in Table 1. Note that we use 512 bytes as the TCP packet size (i.e., the maximum segment size (MSS)), similar to other studies [23]–[25]. Below, we show results for different number of flows, $N$, while keeping the same packet size. Results for different packet sizes can be estimated from those for different $N$'s because of the following reason. Chaos-based control is a transmission rate control mechanism using congestion window, and its behavior for a parameter configuration would be identical to that for other parameter configurations that have the same number of transmitted packets per unit time. The doubling of the packet size halves the number of transmitted packets per unit time as same as the doubling of $N$. Hence, we can estimate results when the packet size doubles from those when $N$ doubles.

In our simulations, each source host starts transmitting randomly within the time range $[0, 1]$ s. After starting a transmission, each source host continues transmitting until the end of the simulation. We repeat the simulation experiments with different random sequences of start times until the 95% confidence interval of measurements becomes sufficiently small. Destination hosts have sufficient processing power to handle received packets, so transmission rates of flows are not restricted by the processing performance of destination hosts. As a simple setting for the base RTT used in the calculation of congestion window size, we set 0.03 s for source hosts.

### 5.2 Confirmation of Operation of the Chaos-Based Transmission Rate Control Rule

To confirm the operation of the chaos-based control rule, we investigate the behavior of flows and the fairness among flows.

Figures 8 and 9 plot time series data of transmission

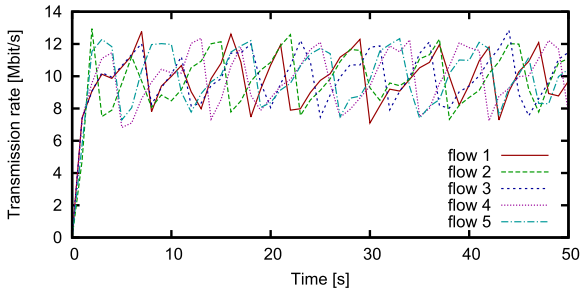



**Fig. 8** Time series of transmission rates of flows 1–5.

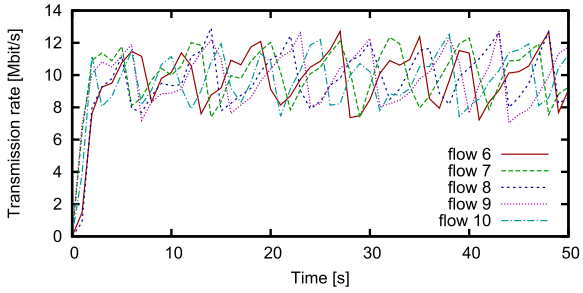



**Fig. 9** Time series of transmission rates of flows 6–10.

rates of flows. In the figures, the transmission rates of flows are approximately the same immediately after transmission begins; however, they appear to be random over time. From these results, we can confirm the randomization effect of chaos-based control.

Next, we investigate the fairness among flows. For our fairness metric, we use the standard deviation (SD) of the transmission rates of flows. The SD at time $t$ is defined as

$$\mathrm{SD}(t) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\overline{r}_i^2(t) - \left(\frac{1}{N}\sum_{i=1}^{N}\overline{r}_i(t)\right)^2}, \tag{5}$$

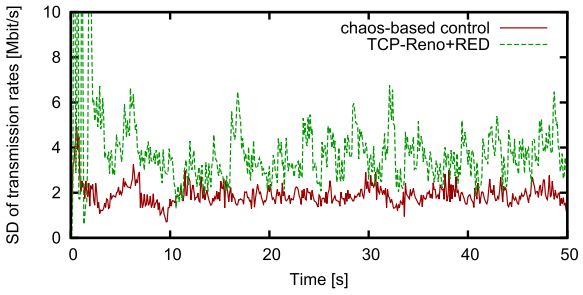
where

$$\overline{r}_i(t) = \frac{1}{\Delta}\int_{t-\Delta}^{t} r_i(x)\,\mathrm{d}\,x. \tag{6}$$

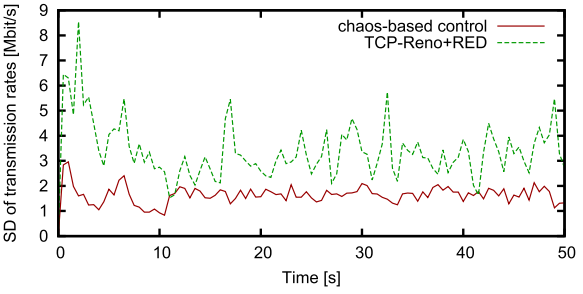
In the above equation, $\Delta$ is the calculation interval time of SD, representing the time scale of interest in the fairness determination. The integration on the right hand side of Eq. (6) is obtained by counting the number of packets transmitted by the source host of flow $i$ from $t - \Delta$ to $t$.

Figures 10(a)–10(c) show time series data for the SD of transmission rates of flows for $\Delta = 0.1$, 0.5, and 1.0 s, respectively. According to the figures, the SD for chaos-based control is almost always smaller than that for TCP-Reno+RED, regardless of $\Delta$. Then, from Fig. 11, chaos-based control has a smaller SD than TCP-Reno+RED, regardless of $N$. Therefore, chaos-based control has higher fairness than RED. This is because chaos-based control can avoid the unfair situation of RED explained in Sect. 3.3.

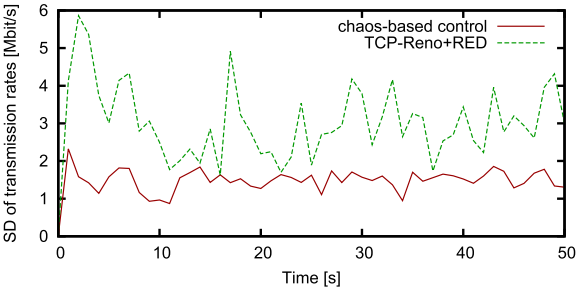
Chaos-based control uses $N$ and $C$ when calculating threshold $r_{\max}$ by Eq. (3). As another control using the same




(a) $\Delta = 0.1$ s




(b) $\Delta = 0.5$ s




(c) $\Delta = 1.0$ s

**Fig. 10** Time series of standard deviation (SD) of transmission rates of flows.

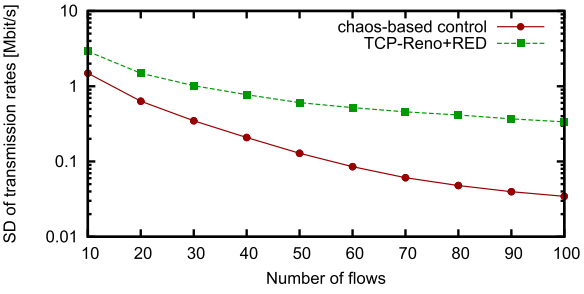



**Fig. 11** Number of flows vs. standard deviation (SD) of transmission rates of flows.

information, we can design a simple control that sets transmission rates of flows to $C/N$. Note that the simple control is equivalent to chaos-based control if relaxation ratio $\alpha$ of chaos-based control approaches to 1. When we use this simple control, the transmission rates of all flows are same. Hence, the SD of transmission rates of flows under the simple control is zero. According to Figs. 11, the fairness of chaos-based control is worse than that of the simple control.

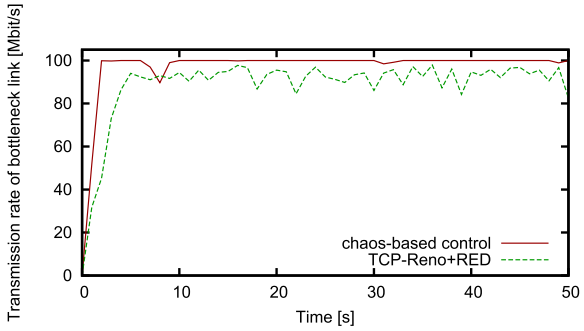



**Fig. 12** Time series of the transmission rate at the bottleneck link.

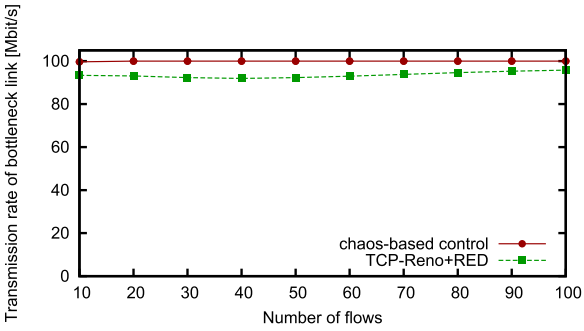



**Fig. 13** Number of flows vs. the time average of the transmission rate at the bottleneck link.

### 5.3 Effectiveness of Setting Threshold $r_{\max}$ by Eq. (3)

Through further simulation, we confirm whether the $r_{\max}$ setting proposed in [1] realizes efficient utilization of the bottleneck link.

Figure 12 plots time series data of the transmission rate of all flows at the bottleneck link. As a reference, we also show results for TCP-Reno+RED in Fig. 12. From the figure, chaos-based control almost always fully uses the link bandwidth. Figure 13 shows the time average of the transmission rate at the bottleneck link for different numbers of flows, $N$. According to this result, chaos-based control realizes complete utilization of the bottleneck link, regardless of $N$.

If link bandwidth is fully utilized but packet losses occur owing to buffer overflow, we cannot say that chaos-based control efficiently utilizes the bottleneck link. Hence, we confirm the queue length of the bottleneck link. Figure 14 shows time series data for the queue length of the bottleneck link. As a reference, we show results for TCP-Reno+RED in Fig. 14. From the figure, we observe that chaos-based control always yields smaller queue lengths than the buffer size (i.e., 5,000 packets), so there is no possibility of packet loss owing to buffer overflow. The same result is obtained, regardless of $N$, as shown in Fig. 15. The simple control introduced at the end of Sect. 5.2 can realize zero queue length and complete link utilization at the bottleneck link. From Fig. 15, we observe that the queue length of chaos-based control is larger than that of the simple control that

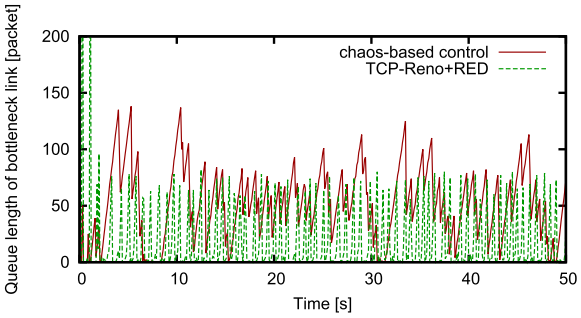



**Fig. 14** Time series of the queue length of the bottleneck link.

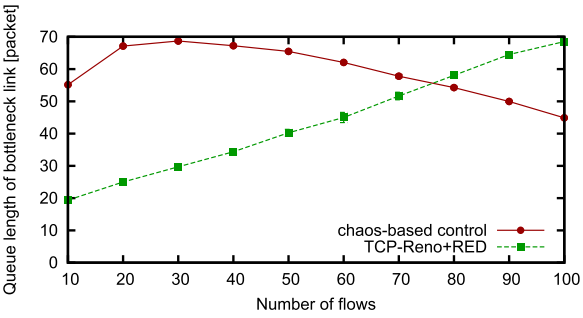



**Fig. 15** Number of flows vs. time average of the queue length of the bottleneck link.

sets transmission rates of flows to $C/N$.

Finally, we compare the the results for different $r_{\max}$ settings to clarify the effectiveness of the settings proposed in [1]. In this experiment, we use $r'_{\max}$, which is given by

$$r'_{\max} = k_{\max}\, r_{\max}, \tag{7}$$

where $k_{\max}$ is a positive constant. In the right side of the above equation, $r_{\max}$ is given by Eq. (3). Hence, if $k_{\max} = 1$, the setting by Eq. (3) is same as that by Eq. (7).

Figures 16 and 17 show time average values of the transmission rate at the bottleneck link and the time average values of the queue length of the bottleneck link for different values of $k_{\max}$, respectively. According to Fig. 16, the transmission rate of the bottleneck link is equal to the bottleneck link bandwidth if $k_{\max} \geq 1$. Conversely, from Fig. 17, we observe that the queue length of the bottleneck link increases as $k_{\max}$ increases. Therefore, we conclude that $k_{\max} = 1$ is the best parameter setting, regardless of $N$.

As evident from above, chaos-based control can realize efficient utilization of the bottleneck link with the $r_{\max}$ setting proposed in [1].

### 5.4 Experimentation Considering Actual Networks

It is unrealistic to expect all flows to support chaos-based control in actual networks. There are usually some flows not under chaos-based control that would share a bottleneck link with flows under chaos-based control. Hence, we perform experiments using the network topology shown in Fig. 18. This topology contains background traffic not under chaos-based control (i.e., UDP traffic with 20 Mbit/s).

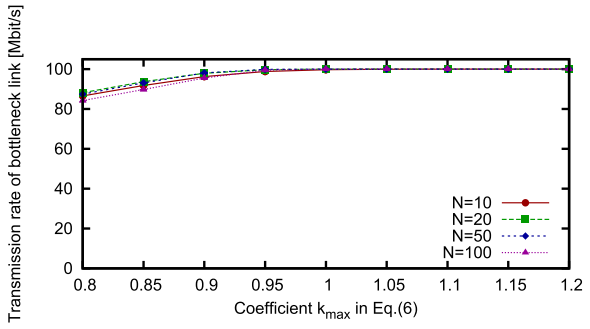



**Fig. 16** $k_{\max}$ vs. the time average of the transmission rate at the bottleneck link

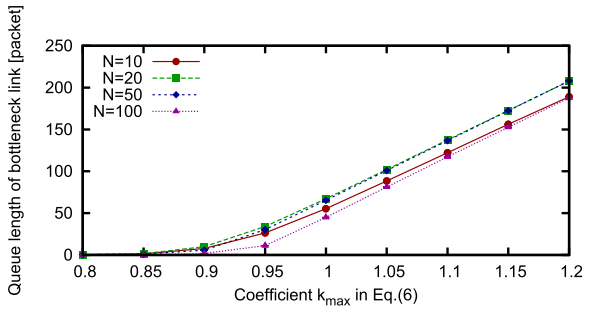



**Fig. 17** $k_{\max}$ vs. the time average of the queue length of the bottleneck link

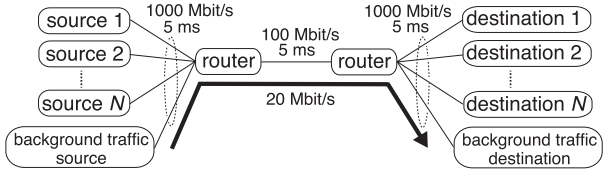



**Fig. 18** Dumbbell topology used in the experiment when generating background traffic.

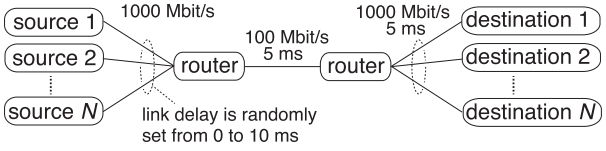



**Fig. 19** Dumbbell topology used in the experiment using settings with different RTTs.

In general, flows with different RTTs share a bottleneck link. Hence, we then perform experiments using the network topology shown in Fig. 19. In this network topology, link delay $l_i$ between source of flow $i$ and the router at the bottleneck link are set to a random value from 0 to 10 ms. Furthermore, the base RTT of flow $i$ is set to $2\,l_i + 20$ ms.

Figures 20 and 21 show transmission rates and queue lengths of the bottleneck link when generating background traffic, respectively. Note that we showed the transmission rate without background traffic to investigate the efficiency of flows under chaos-based control. From these figures, we

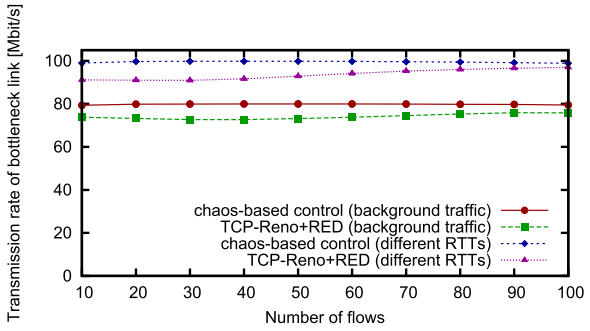



**Fig. 20** Number of flows vs. the time average of the transmission rate at the bottleneck link in the experiments with considering actual networks.

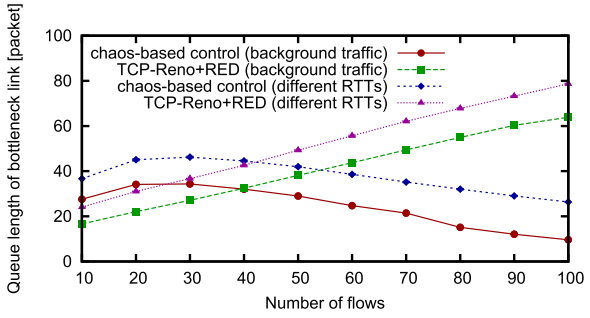



**Fig. 21** Number of flows vs. time average of the queue length of the bottleneck link in the experiments with considering actual networks.

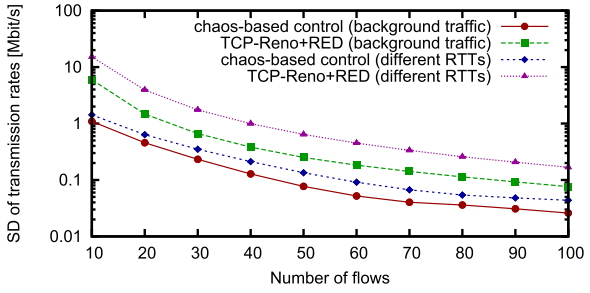



**Fig. 22** Number of flows vs. standard deviation (SD) of transmission rates of flows in the experiments with considering actual networks.

can confirm that chaos-based control realizes efficiency similar to the results of Sect. 5.3. Figure 22 shows the SD of transmission rates of flows under chaos-based control. According to these results, chaos-based control has similar effectiveness in terms of fairness to those of results presented in Sect. 5.2. Figures 20–22 also show results from experiments using different RTTs. For such experiments, we obtain the same conclusions regarding fairness and efficiency when using scenarios with background traffic and those presented in Sects. 5.2 and 5.3.

## 6. Conclusions and Future Work

We implemented chaos-based hierarchical transmission rate control in the popular network simulator, ns-2, and confirmed its successful operation through various experiments. More specifically, we confirmed that (a) the chaos-based control rule realizes fairness among users, and (b) the setting of the parameter used in the control rule proposed in [1] realizes efficient utilization of bottleneck links. These results clarified the feasibility of realizing the chaos-based concept in TCP/IP networks.

As future work, we plan to apply the concept of chaos-based hierarchical network control to other network control designs.

## Acknowledgements

The authors thank Ms.Yuri Takahashi of KDDI Corporation for her help in implementing chaos-based transmission rate control in the network simulator, ns-2. The authors also thank Mr. Liu Yongchao of Tokyo Metropolitan University for his help to in gathering simulation results. This work was supported by JSPS KAKENHI Grant Number 25540032 and 15K15985.

### References

[1] M. Aida, "Concept of chaos-based hierarchical network control and its application to transmission rate control," IEICE Trans. Commun., vol.E98-B, no.1, pp.135–144, Jan. 2015.
[2] K. Ito, Y. Oono, H. Yamazaki, and K. Hirakawa, "Chaotic behavior in great earthquakes — coupled relaxation oscillator model, billiard model and electronic circuit model —," J. Phys. Soc. Jpn., vol.49, no.1, pp.43–52, July 1980.
[3] L. Qiu, Y. Zhang, and S. Keshav, "On individual and aggregate TCP performance," Proc. Seventh International Conference on Network Protocols, pp.203–212, 1999.
[4] J. Gettys and K. Nichols, "Bufferbloat: Dark buffers in the Internet," Commun. ACM, vol.55, no.1, pp.57–65, Jan. 2012.
[5] "The network simulator — ns-2," available at http://www.isi.edu/nsnam/ns/
[6] S. Shakkottai, T.S. Rappaport, and P.C. Karlsson, "Cross-layer design for wireless networks," IEEE Commun. Mag., vol.41, no.10, pp.74–80, Oct. 2003.
[7] T. ElBatt and A. Ephremides, "Joint scheduling and power control for wireless ad hoc networks," IEEE Trans. Wireless Commun., vol.3, no.1, pp.74–85, Jan. 2004.
[8] X. Lin and N.B. Shroff, "The impact of imperfect scheduling on cross-Layer congestion control in wireless networks," IEEE/ACM Trans. Netw., vol.14, no.2, pp.302–315, April 2006.
[9] L. Chen, S.H. Low, M. Chiang, and J.C. Doyle, "Cross-layer congestion control, routing and scheduling design in ad hoc wireless networks," Proc. 25TH IEEE International Conference on Computer Communications, IEEE INFOCOM 2006, pp.1–13, 2006.
[10] M. Aida, "Using a renormalization group to create ideal hierarchical network architecture with time scale dependency," IEICE Trans. Commun., vol.E95-B, no.5, pp.1488–1500, May 2012.
[11] C. Takano and M Aida, "Diffusion-type autonomous decentralized flow control for end-to-end flow in high-speed networks," IEICE Trans. Commun., vol.E88-B, no.4, pp.1559–1567, 2005.
[12] C. Takano, M. Aida, M. Murata, and M. Imase, "Proposal for autonomous decentralized structure formation based on local interaction and back-diffusion potential," IEICE Trans. Commun., vol.E95-B, no.5, pp.1529–1538, 2012.
[13] Y. Sakumoto, M. Aida, and H. Shimonishi, "An autonomous decentralized control for indirectly controlling system performance variable in large-scale and wide-area network," Proc. 16th International Telecommunications Network Strategy and Planning Symposium

(Networks 2014), pp.1–7, 2014.
[14] L. Chen, X. Wang, and Z. Han, "Controlling chaos in Internet congestion control model," Chaos, Solitons & Fractals, vol.21, no.1, pp.81–91, July 2004.
[15] N. Bigdeli, M. Haeri, S. Choobkar, and F. Jannesari, "Characterization of complex behaviors of TCP/RED computer networks based on nonlinear time series analysis methods," Physica D: Nonlinear Phenomena, vol.233, no.2, pp.138–150, Sept. 2007.
[16] M.A. Sharbafi and M.J. Yazdanpanah, "IDFC: A new approach to control bifurcation in TCP/RED," J. Netw. Comput. Appl., vol.34, no.6, pp.2042–2050, Nov. 2011.
[17] M. Allman, V. Paxson, and W. Stevens, "TCP congestion control." RFC 2581, April 1999.
[18] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Trans. Netw., vol.1, no.4, pp.397–413, Aug. 1993.
[19] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," SIGCOMM Comput. Commun. Rev., vol.32, no.4, pp.89–102, Oct. 2002.
[20] A. Falk, Y. Pryadkin, and D. Katabi, "Specification for the explicit control protocol (XCP)," Network Working Group Internet-Draft 03, Jan. 2007.
[21] L.S. Brakmo, S.W. O'Malley, and L.L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," SIGCOMM Comput. Commun. Rev., vol.24, no.4, pp.24–35, Oct. 1994.
[22] D.X. Wei, C. Jin, S.H. Low, and S. Hegde, "FAST TCP: Motivation, architecture, algorithms, performance," IEEE/ACM Trans. Netw., vol.14, no.6, pp.1246–1259, Dec. 2006.
[23] V. Jacobson, "Congestion avoidance and control," SIGCOMM Comput. Commun. Rev., vol.18, no.4, pp.314–329, Aug. 1988.
[24] K. Xu, M. Gerla, L. Qi, and Y. Shu, "Enhancing TCP fairness in ad hoc wireless networks using neighborhood RED," Proc. 9th Annual International Conference on Mobile Computing and Networking, MobiCom'03, pp.16–28, 2003.
[25] X.M. Zhang, W.B. Zhu, N.N. Li, and D.K. Sung, "TCP congestion window adaptation through contention detection in ad hoc networks," IEEE Trans. Veh. Technol., vol.59, no.9, pp.4578–4588, Nov. 2010.

**Yusuke Sakumoto** received M.E. and Ph.D. degrees in the Information and Computer Sciences from Osaka University in 2008 and 2010, respectively. He is currently an assistant professor at Graduate School of System Design, Tokyo Metropolitan University, Japan. His research work is in the area of analysis and design of network controls. He is a member of IEEE, IPSJ and IEICE.

**Chisa Takano** received the B.E. degree in Telecommunication Engineering from Osaka University, Japan, in 2000, and received the Ph.D. in Telecommunications Engineering from Tokyo Metropolitan University, Japan, in 2008. In 2000 she joined the Traffic Research Center, NTT Advanced Technology Corporation (NTT-AT). Since April 2008, she has been an Associate Professor of the Graduate School of Information Sciences, Hiroshima City University. Her research interests are in the area of compute networks and distributed systems. She received the IEICE's Young Researchers' Award in 2003.

**Masaki Aida** received the B.S. degree in Physics and M.S. degree in Atomic Physics from St. Paul's University, Tokyo, Japan, in 1987 and 1989, respectively, and received the Ph.D. in Telecommunications Engineering from the University of Tokyo, Japan, in 1999. After joining NTT Laboratories in April 1989, he has been engaged in research on traffic issues in computer communication networks. From April 2005 to March 2007, he was an Associate Professor at the Faculty of System Design, Tokyo Metropolitan University. He has been a Professor of the Graduate School of System Design, Tokyo Metropolitan University since April 2007. He received the Best Tutorial Paper Award of IEICE Communications Society in 2013. Prof. Aida is a member of the IEEE, IEICE, and the Operations Research Society of Japan.

**Masayuki Murata** received the M.E. and D.E. degrees in Information and Computer Sciences from Osaka University, Japan, in 1984 and 1988, respectively. In April 1984, he joined IBM Japan's Tokyo Research Laboratory, as a Researcher. From September 1987 to January 1989, he was an Assistant Professor with the Computation Center, Osaka University. In February 1989, he moved to the Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University. In April 1999, he became a Professor of Osaka University, and since April 2004, he has been with the Graduate School of Information Science and Technology, Osaka University. He has contributed more than four hundred and fifty papers to international and domestic journals and conferences. His research interests include computer communication networks and performance modeling and evaluation. He is a member of the IEEE, ACM, IEICE, the Internet Society, and IPSJ.