Journal of Information Processing Vol.21 No.3 481–489 (July 2013)

[DOI: 10.2197/ipsjjip.21.481]

Regular Paper

A Method for Accelerating Flow-level Network Simulation with Low-pass Filtering of Fluid Models

Yusuke Sakumoto[1,a)] Hiroyuki Ohsaki[2,b)] Makoto Imase[2,c)]

Received: October 11, 2012, Accepted: March 1, 2013

Abstract: Conventional flow-level simulators use timescales around the round-trip time when numerically solving fluid-flow models for network simulations. In large-scale and high-speed network simulations, only understanding coarser behavior than that achieved with timescales around the round-trip time is sometimes sufficient for performance evaluation. In this paper, we propose a novel method for accelerating flow-level simulations; it omits timescale finer than that required by the performance evaluation. Through experiments, we investigate the effectiveness of the proposed method for accelerating flow-level simulations. Our findings show that the proposed method offers 60 times faster than the conventional flow-level simulator.

Keywords: flow-level simulator, accelerating simulation, fluid-flow model, low pass filter, large-scale/high-speed network

1. Introduction

Because of widespread deployment and advances in network technologies, the scale (i.e., number of nodes and network bandwidth) of networks has been expanding rapidly [1]. Such explosive expansion of networks makes it difficult to understand the behavior of the entire network. To understand the behavior of large-scale and high-speed networks, a network simulator with fast execution is needed. As such fast network simulator compared with the conventional network simulator [2], several flow-level simulators have been proposed [3], [4], [5], [6].

In Ref. [3], we proposed the flow-level simulator FSIM (Fluid-based SIMulator) for performance evaluation of large-scale and high-speed networks. FSIM performs flow-level simulation by numerically solving ODEs (Ordinary Differential Equations) of fluid models [7]. For accelerating flow-level simulation, FSIM adaptively controls the step-size of the numerical solver used to numerically solve ODEs.

Fluid models [4], [7] used in flow-level simulations describe the network behavior with timescales around the round-trip time. In other words, conventional flow-level simulators [3], [4] employ the same timescale order.

However, in large-scale simulations, only understanding coarser behavior than that realized with timescales around the round-trip time is sometimes sufficient (**Fig. 1**). Depending on the purpose of the simulation, such an understanding the performance of a file transfer protocol or a network routing protocol,

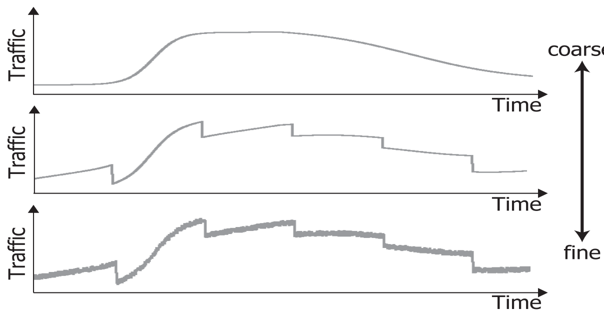


Fig. 1 An example of behavior at each timescale.

coarse network behavior at the timescales of seconds or minutes is sometimes sufficient [8], [9]. When designing overlay networks and communication networks, timescales of minutes or hours are acceptable in some cases [10], [11]. Since such protocols and networks behave with larger timescale than round-trip time, it becomes important to investigate the coarse network behavior in performance evaluation. Hence, for performance evaluation of large-scale networks, it would be sufficient that a flow-level simulator simulates the coarser behavior, and need not essentially simulate the finer behavior than that with the timescale required in performance evaluation.

In this paper, we propose a novel method that accelerates flow-level simulations by omitting the simulations whose timescales are finer than that required in the performance evaluation. Specifically, the frequencies higher than the cutoff frequency specified by the performance evaluator are attenuated by low-pass filtering the fluid models. By attenuating high frequencies, the change in network states is smoothened. As a result, the step-size in the ODE numerical solver can be increased. This will accelerate flow-level simulations.

We conduct experiments to investigate the effectiveness of the proposed method. We first implement three low-pass filters (Inte-

1 Graduate School of System Design, Tokyo Metropolitan University, Hino, Tokyo 191–0065, Japan
2 Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
a) sakumoto@sd.tmu.ac.jp
b) oosaki@ist.osaka-u.ac.jp
c) imase@ist.osaka-u.ac.jp

gral filter, RC filter, and RLC filter) into our flow-level simulator FSIM, and use it to evaluate the accuracy, speed, and memory consumption of the proposed method.

This paper is organized as follows. In Section 2, we introduce the low-pass filters used in this paper. In Section 3, we explain the flow-level simulation acceleration method. Section 4 details the experiments conducted on our FSIM implementation to evaluate the accuracy, speed, and memory consumption of the proposed method. Finally, in Section 5, we conclude this paper and discuss future works.

2. Low-pass Filters

In this section, we introduce the low-pass filters used to attenuate the frequencies higher than the cutoff frequency as defined by reducing the amplitude of input signals to 1/2. Although many low-pass filters exist, we select three low-pass filters (integral filter, RC filter, and RLC filter) [12], [13] since their filters are easy to implement. Hence, we can confirm the effectiveness of our basic idea by using easy way.

2.1 Integral Filter

The integral filter [12] smooths the input signals by integrating input function $y(t)$ over interval $[t-T, t]$. Output function $y_0(t)$ of the integral filter for input function $y(t)$ is defined by

$$y_0(t) = \frac{1}{T}\int_{t-T}^{t} y(\tau)d\tau. \qquad (1)$$

By differentiating both sides of Eq. (1), we obtain the following ODE

$$\frac{dy_0(t)}{dt} = \frac{1}{T}\{y(t) - y(t-T)\}. \qquad (2)$$

To investigate the frequency property of the RLC filter, we convert Eq. (2) from the time domain to the frequency domain. The converted equation of Eq. (2) in the frequency domain is given by

$$j\omega Y_0(j\omega) = \frac{1}{T}\left\{1 - e^{-j\omega T}\right\} Y(j\omega), \qquad (3)$$

where j is imaginary unit, and ω is angular frequency given by $\omega = 2\pi f$. $Y_0(j\omega)$ is the output function of the integral filter in the frequency domain, and $Y(j\omega)$ is the input function in the frequency domain. From Eq. (3), cutoff frequency f_c of the integral filter is given by

$$f_c = \frac{2}{\pi T}. \qquad (4)$$

The frequency responses (gain $G(f_c)$ and phase lag $\theta(f_c)$) of the integral filter with the cutoff frequency f_c are given by

$$G_0(f_c) = \frac{f_c}{4f}\sqrt{2\left\{1 - \cos\left(\frac{4f}{f_c}\right)\right\}}, \qquad (5)$$

and

$$\theta_0(f_c) = -\tan^{-1}\left\{\frac{\cos\left(1 - \frac{4f}{f_c}\right)}{\sin\left(\frac{4f}{f_c}\right)}\right\}. \qquad (6)$$

2.2 RC Filter

The RC filter [13] smooths the input voltage in an electrical circuit (RC circuit) consisting of a resistor and a capacitance. In a RC circuit, the relation of the input voltage (input function), $y(t)$, and the output voltage (output function), $y_1(t)$, is given by

$$RC\frac{dy_1(t)}{dt} + y_1(t) = y(t), \qquad (7)$$

where R and C are the resistance and the capacitance, respectively. If $RC > 1$, the output function $y_1(t)$ smoothened input function $y(t)$ is obtained.

The converted equation of Eq. (7) in the frequency domain is given by

$$\{j\omega RC + 1\} Y_1(j\omega) = Y(j\omega), \qquad (8)$$

where $Y_1(j\omega)$ is the output function of the RC filter in the frequency domain. From Eq. (8), cutoff frequency f_c of the RC filter is given by

$$f_c = \frac{\sqrt{3}}{2\pi RC}. \qquad (9)$$

The frequency responses (gain $G_1(f_c)$ and phase lag $\theta_1(f_c)$) of the RC filter with the cutoff frequency f_c are given by

$$G_1(f_c) = \frac{1}{\sqrt{1 + \left(\frac{\sqrt{3}f}{f_c}\right)^2}}, \qquad (10)$$

and

$$\theta_1(f_c) = -\tan^{-1}\left(\frac{\sqrt{3}f}{f_c}\right). \qquad (11)$$

2.3 RLC Filter

The RLC filter [13] smooths the input voltage in an electrical circuit (RLC circuit) consisting of a resistor, an inductor, and a capacitance. In this RLC circuit, the relation of the input voltage (input function), $y(t)$, and the output voltage (output function), $y_2(t)$, is given by

$$LC\frac{d^2y_2(t)}{dt^2} + RC\frac{dy_2(t)}{dt} + y_2(t) = y(t), \qquad (12)$$

where L is the inductance. It is well known that the RLC filter shows a resonating property for $k_2 < 1$ ($k_2 = \sqrt{LC}/(2RC)$) and a smoothing property for $k_2 > 1$ [14]. Hence, two parameters of the RLC filter, RC and LC, can be determined for given cutoff frequency f_c and $k_2 > 1$. Hereafter, we substitute $2k_2\sqrt{LC}(k_2 > 1)$ for RC to obtain the output function $y_2(t)$ smoothened input function $y(t)$.

The converted equation of Eq. (12) in the frequency domain is given by

$$\left\{(j\omega)^2 LC + j\omega RC + 1\right\} Y_2(j\omega) = Y(j\omega), \qquad (13)$$

where $Y_2(j\omega)$ is the output function of the RLC filter in the frequency domain. From Eq. (13), cutoff frequency f_c of the RLC filter is given by

$$f_c = \frac{1}{2\pi}\sqrt{\frac{1 - 2k_2^2 + \sqrt{(1-2k_2^2)^2 + 3}}{LC}}. \qquad (14)$$

The frequency responses (gain $G_2(f_c)$ and phase lag $\theta_2(f_c)$) of the integral filter with the cutoff frequency f_c are given by

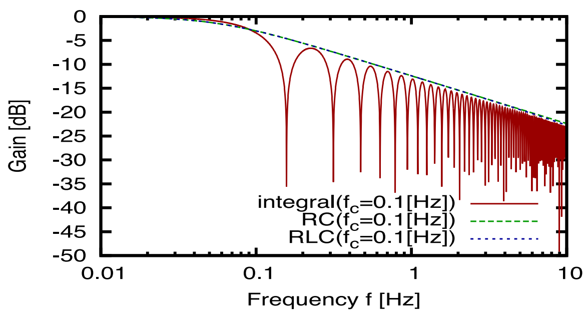


Fig. 2 Gains of the integral filter, the RC filter, and the RLC filter for the cutoff frequency f_c = 0.1 [Hz].

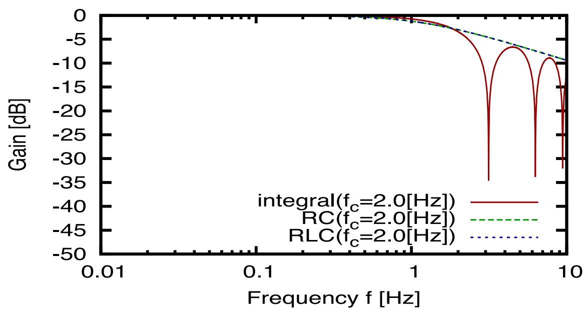


Fig. 3 Gains of the integral filter, the RC filter, and the RLC filter for the cutoff frequency f_c = 2 [Hz].

$$G_2(f_c) = \frac{f_c^2}{\sqrt{\left\{f_c^2 - (f\,k_2')^2\right\}^2 + (2\,k_2\,k_2'\,f\,f_c)^2}}, \tag{15}$$

and

$$\theta_2(f_c) = -\tan^{-1}\left\{\frac{2\,k_2\,k_2'\,f\,f_c}{f_c^2 - (f\,k_2')^2}\right\}, \tag{16}$$

where

$$k_2' = \sqrt{1 - 2\,k_2^2 + \sqrt{(1 - 2\,k_2^2)^2 + 3}}. \tag{17}$$

The computational complexity of solving Eq. (12) is larger than that of solving Eq. (7). This is because Eq. (12) is second-order ODE, but Eq. (7) is first-order ODE.

2.4 Plotting Frequency Response

For illustrating properties of three low-pass filers (i.e., integral filter, RC filter, and RLC filter), we plot the frequency responses of the low-pass filters. The effect of phase lags of low-pass filters on simulation results and accuracy will be discussed in Section 3.4. To obtain gains and phase lags of low pass filters, we use Eqs. (5), (6), (10), (11), (15), and (16) for $k_2 = 10$.

Figures 2 and **3** plot the gains of the integral filter, the RC filter, and the RLC filter as determined using cutoff frequencies f_c = 0.1 and 2.0 [Hz], respectively. Note that the lines for the RC filter have an overlap with those for the RLC filter in Figs. 2 and 3. The RLC filter with $k_2 = 10$ has approximately same frequency response as the RC filter. In these figures, the frequencies smaller than the cutoff frequency are somewhat attenuated. Hence, cutoff frequency f_c should be set with considering such attenuation in the output function.

Figures 4 and **5** plot the phase lags of the integral filter, the RC filter, and the RLC filter as determined using cutoff frequency

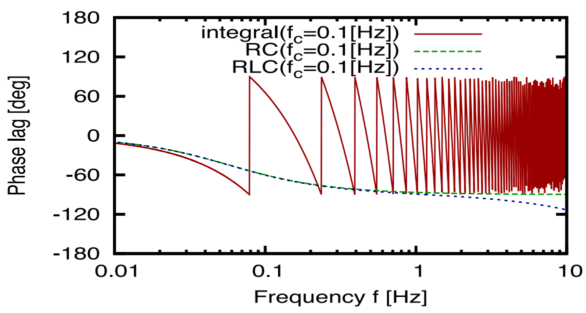


Fig. 4 Phase lags of the integral filter, the RC filter, and the RLC filter for the cutoff frequency f_c = 0.1 [Hz].

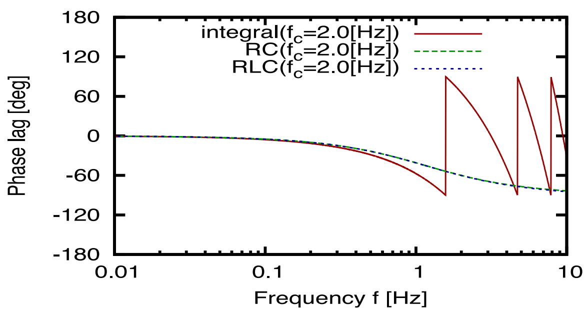


Fig. 5 Phase lags of the integral filter, the RC filter, and the RLC filter for the cutoff frequency f_c = 2.0 [Hz].

f_c = 0.1 and 2.0 [Hz], respectively. Note that the lines for the RC filter have an overlap with those for the RLC filter in Fig. 5. From Figs. 4 and 5, the input function of low-pass filters is somewhat delayed in the output function according to the phase lags. Hence, cutoff frequency f_c should be set with considering the delay in the output function.

From these results, the integrate filter obviously has oscillated frequency response unlike the RC filter and the RLC filter. The difference of frequency response would affect how well a low-pass filter attenuates the frequencies higher than the cutoff frequency f_c.

3. Accelerating Flow-level Network Simulation Method with Low-pass Filtering

3.1 Basic Idea

To accelerate a flow-level simulation, we omit the network behavior at a finer timescale than that required by performance evaluation. Specifically, we use low-pass filtering of fluid models to attenuate the frequencies higher than the cutoff frequency specified by the performance evaluator. Cutoff frequency f_c of the low-pass filter should be set according to the requirements of the performance evaluator. By attenuating high frequencies, the change in network states is smoothened. As a result, the step-sizes in the numerical ODE solver can be increased. This accelerates the flow-level simulation.

3.2 Numerical Algorithm

The algorithm of the proposed method repeatedly solves the ODE using the step-size of Δ (**Fig. 6**). The step-size Δ is determined using the adaptive step-size control explained in Section II.4 of Ref. [15]. Please refer to Section II.4 of Ref. [15] for details. We denote the network state vector and the low-pass filter's state vector at time t by $\mathbf{y}(t)$ and $\mathbf{y}_l(t)$, respectively. Net-

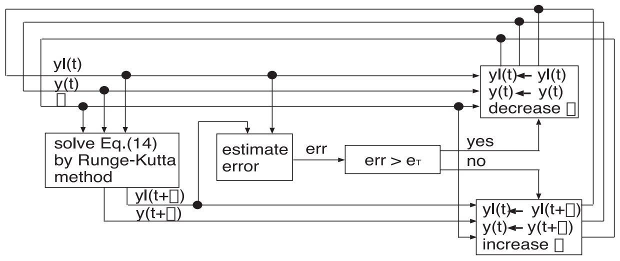


Fig. 6 Block figure of network updates.

work state vector is a set of network variables in fluid models. Low-pass filter's state vector is a set of low-pass filter's output variables corresponding to network variables in the network state vector.

First, we obtain network state vector $\mathbf{y}(t+\Delta)$ from the network state vector, $\mathbf{y}(t)$ by using the numerical ODE solver [16]. Specifically, we numerically solve the following ODEs

$$\frac{d\,\mathbf{y}(t)}{d\,t} = \mathbf{f}(\mathbf{y}(t)), \tag{18}$$

where $\mathbf{f}(\mathbf{y}(t))$ is the first derivative given by the fluid models.

Next, by using the network state vector $\mathbf{y}(t+\Delta)$ and the low-pass filter's state vector $\mathbf{y}_l(t)$, we obtain the low-pass filter's state vector $\mathbf{y}_l(t+\Delta)$ by numerically solving ODEs of low-pass filters. Specifically, we numerically solve the following ODEs

$$\frac{d\,\mathbf{y}_l(t)}{d\,t} = \mathbf{h}(\mathbf{y}(t), \mathbf{y}_l(t)), \tag{19}$$

where $\mathbf{h}(\mathbf{y}(t), \mathbf{y}_l(t))$ is the first derivative given by Eqs. (2), (7), and (12). Called *low-pass filtering*, this procedure attenuates the high frequencies when evolving the fluid model. With the numerical ODE solver [16], fourth-order and fifth-order numerical solutions of $\mathbf{y}_l(t+\Delta)$ can be obtained. We denote fourth-order and fifth-order numerical solutions $\mathbf{y}_l(t+\Delta)$ by $\mathbf{y}_l^{(4)}(t+\Delta)$ and $\mathbf{y}_l^{(5)}(t+\Delta)$, respectively. The numerical solutions $\mathbf{y}_l^{(4)}(t+\Delta)$ and $\mathbf{y}_l^{(5)}(t+\Delta)$ are utilized to estimate the numerical error in $\mathbf{y}_l(t+\Delta)$.

To guarantee that the numerical error lies within the predetermined tolerance error e_T, we estimate the numerical error in the low-pass filter's state vector, $\mathbf{y}_l(t+\Delta)$. Note that we do not intend that network state vector $\mathbf{y}(t+\Delta)$ be used for this purpose. This is because that the purpose of flow-level simulation with the proposed method is to simulate the coarse behavior obtained by low-pass filters, and not to simulate the fine behavior obtained by fluid models. Specifically, in the adaptive step-size control [15], the numerical error *err* in $\mathbf{y}_l(t+\Delta)$ is given by

$$err = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{yi_l^{(4)}(t+\Delta) - yi_l^{(5)}(t+\Delta)}{\max(yi_l^{(4)}(t+\Delta), yi_l^{(5)}(t+\Delta))}\right)^2}, \tag{20}$$

where $yi_l^{(4)}(t+\Delta)$ and $yi_l^{(5)}(t+\Delta)$ are the i-th element in fourth-order and fifth-order numerical solutions $\mathbf{y}_l^{(4)}(t+\Delta)$ and $\mathbf{y}_l^{(5)}(t+\Delta)$. N is the number of variables in the low-pass filter's state vector. According to the conventional adaptive step-size control, if the numerical error is larger than the tolerance error e_T set by the performance evaluator, the step-size in the numerical ODE solver is decreased, and network state vector $\mathbf{y}(t+\Delta)$ is recalculated. On the contrary, if the numerical error is smaller than the tolerance error e_T, the step-size in the numerical ODE solver is increased, and the flow-level simulation is put forward.

The algorithm does not guarantee the numerical error in the numerical solution of Eq. (18) but it does guarantee the numerical error in the numerical solution of Eq. (19). Note that numerical error means local error for a single state update. If the numerical solution of Eq. (19) is obtained in two-steps (i.e., solving Eqs. (18) and (19) sequentially), the numerical error in both numerical solutions of Eqs. (18) and (19) must be guaranteed. However, the algorithm directly solves Eq. (18), which enables efficient computation of Eq. (19) by compromising the numerical error in the numerical solution of Eq. (18).

3.3 Notes on Numerical Error

We discuss the numerical error in the algorithm. The algorithm guarantees the numerical error in a numerical solution of $y_l(t+\Delta)$ is smaller than e_T.

To obtain the numerical solution $\mathbf{y}_l^{(p)}(t+\Delta)$, the algorithm performs the adaptive step-size control after sequentially solving Eqs. (18) and (19) in two steps. On the other hand, the conventional algorithm used in Ref. [3] performs the adaptive step-size control after only solving Eq. (18). According to the discussion on numerical error in two steps explained in Section II.4 of Ref. [15], the numerical error in the p-th order of numerical solution $\mathbf{y}_l^{(p)}(t+\Delta)$ obtained by the algorithm is given by

$$(C + C_l)\Delta^{p+1} + O(\Delta^{p+2}), \tag{21}$$

where C and C_l are the constants related to Eqs. (18) and (19), respectively. From Eq. (20), the numerical error in the fourth order of numerical solution $\mathbf{y}_l^{(4)}(t+\Delta)$ given by Eq. (21) can be estimated. Since the algorithm ensures the estimated numerical error to be less than e_T, the numerical error in the numerical solution of $y_l(t+\Delta)$ becomes smaller than e_T.

3.4 Effect of Phase Lag in Low-pass filter

As we have shown in Section 2.4, low-pass filters have phase lag properties. Phase lags of low-pass filters bring some delay in simulation results compared with results of ideal coarse network behavior, leading to slight degradation of accuracy. Hence, for a given low-pass filter, the performance evaluator should be aware of the effect of its phase lag on simulation results and accuracy. We should emphasize that the phase lag is at most 360 degrees (i.e., one cycle), which should have little impact on performance evaluation in practice.

3.5 Setting of Cutoff Frequency

The performance evaluator can intuitively determine the setting of cutoff frequency f_c according to the timescale focused by the performance evaluator. First, the performance evaluator chooses a desired timescale value of the network behavior. Then, the cutoff frequency f_c is set to the inverse of the desired timescale value.

By using the above setting of cutoff frequency f_c, the performance evaluator would be able to maximize the simulation speed under the condition to acceptable accuracy for the performance evaluator. The acceptable accuracy is related to acceptance attenuation of the obtained network behavior at the desired timescale value. In the above setting of cutoff frequency f_c, we suppose that the acceptable attenuation of the obtained network behavior at the desired timescale value for the performance evaluator is 50% since we define the cutoff frequency f_c by reducing the amplitude of input signals to $1/2$. If the acceptable attenuation for the performance evaluator is different from 50%, the performance evaluator needs to further set the LPF parameters (i.e., T, RC and LC) based on definition of cutoff frequency corresponding to the acceptable attenuation.

3.6 Parameter Setting in Low-pass Filters

The integral filter and RC filter have essentially one parameter T and RC, respectively. Parameter T in the integral filter can be determined for a given cutoff frequency f_c using Eq. (4). Parameter RC in the RC filter can be determined for a given cutoff frequency f_c using Eq. (9).

The RLC filter has essentially two parameters RC and LC. The parameters RC and LC can be determined for a given cutoff frequency f_c and $k_2 > 1$ ($k_2 = \sqrt{LC}/(2RC)$) using Eq. (14). In practice, k_2 should not be set to a value close to 1 since the resonance of output signals sometimes occurs. For utilizing the smoothing effect of the RLC filter, a performance evaluator should use the $k_2 >> 1$.

4. Experiment

Through experiments, we investigate the effectiveness of the proposed method for accelerating a flow-level simulation. We implement the numerical algorithm using low-pass filters into our flow-level simulator FSIM. For the experiment, we use the low-pass filters (Integral filter, RC filter, and RLC filter). We evaluate the accuracy, the speed, and the memory consumption of each combination.

4.1 Experiment Setup

We use a dumbbell topology (**Fig. 7**) as the network topology. **Table 1** shows the parameter configuration used in experiments. We purposely set the parameter k_2 of the RLC filter to 10 for fairly comparing the computational efficiency of the RLC filter with that of the RC filter under the condition that the RLC filter has the approximately same frequency response of RC filter.

In the experiment, we use two simulation scenarios: a simple scenario and a complex scenario. The simple scenario is used to investigate the basic property of the proposed method. On the other hand, complex scenario is used to investigate the effectiveness of the proposed method in a network with complex behavior is used.

- Simple scenario
  In the simple scenario, all TCP flows are activated at the start of simulation, and are deactivated at the termination of simulation.
- Complex scenario
  In the complex scenario, each TCP flow with a given file size is activated at a given time, and are deactivated at the time of its file transfer completion. We randomly give the activation time and file size of each TCP flow according to specified distributions. More specifically, the distribution of time intervals between each pair of consecutive TCP flow activations is given by the exponential distribution with mean 1.0 [s]. The distribution of file sizes of TCP flow is given by the Pareto distribution with the average 20,000 [packet] and the shape 3.0. The complex scenario simply represents part of complex behavior in a large-scale network.

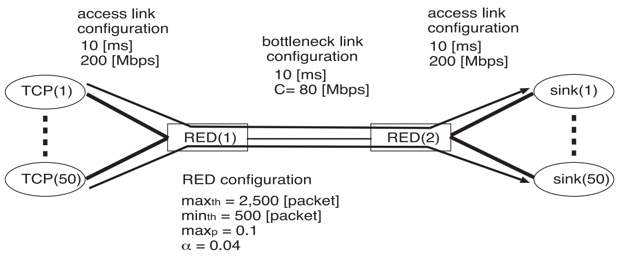


Fig. 7 Dumbbell topology.

Table 1 Parameter configuration.

initial value of TCP transmission rate	0.001	[Mbit/s]
packet size	1,000	[byte]
minimum threshold of RED router	500	[packet]
maximum threshold of RED router	2,500	[packet]
maximum packet marking probability of RED router	0.1	
weight of the exponential average of RED router	0.04	
buffer size of RED router	5,000	[packet]
number of TCP flows	50	
bandwidth of bottleneck link	80	[Mbit/s]
link delay of link	10	[ms]
simulation time	200	[s]
initial value of step-size	0.001	[ms]
minimum of step-size	0.001	[ms]
maximum of step-size	10.0	[ms]
tolerance error	10^{-14}	
coefficient of RLC filter k_2	10	

In the experiment, we repeated 10 simulations, and measured the average and 95% confidence interval of measurements (e.g., simulation execution time and maximum memory consumption).

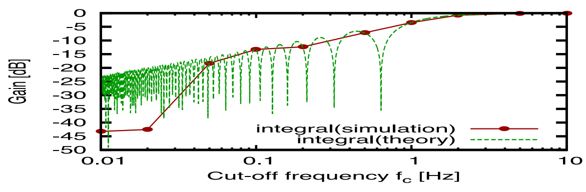
All processing was done on an Intel Core i5 2.8 [GHz] processor with 12 [Gbyte] memory running Mac OS X 10.7 (Lion).

4.2 Accuracy

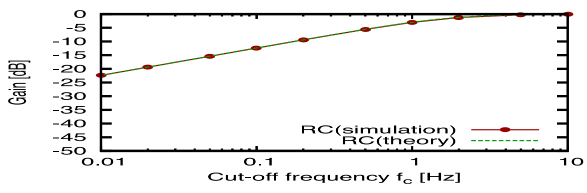
We investigate the accuracy of the numerical algorithm used in the proposed method.

We first compare the frequency responses of the low-pass filters obtained from the equations (Eqs. (5), (6), (10), (11), (15), and (16)) and the flow-level simulation results yielded by the proposed method, to conform that the low-pass filters are accurately implemented into the flow-level simulator. In this experiment, we use the simple scenario.

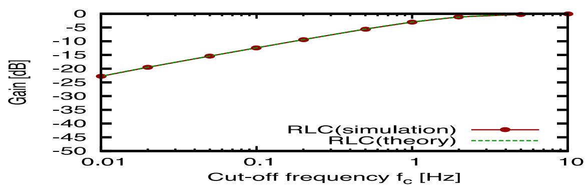
Figures 8 and **9** plot the gain and phase lag of the integral filter, the RC filter, and the RLC filter for different cutoff frequencies, f_c. In these figures, *theory* means the frequency responses yielded by the equations. On the contrary, *simulation* means the frequency responses obtained from flow-level simulations with the proposed method. Since TCP transmission rates have a frequency with approximately 1 [Hz] in the simple scenario, we use the frequency $f = 1$ [Hz] in the equations for calculating theory values. From Figs. 8 and 9, the simulated frequency responses of the RC filter and the RLC filter coincide with those obtained from



(a) Integral filter

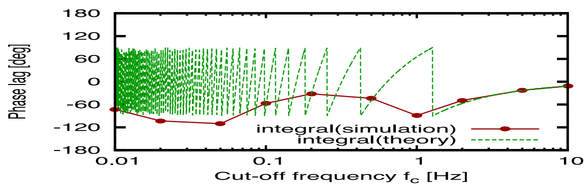


(b) RC filter

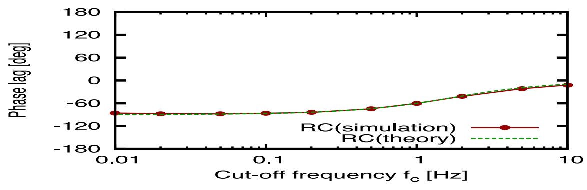


(c) RLC filter

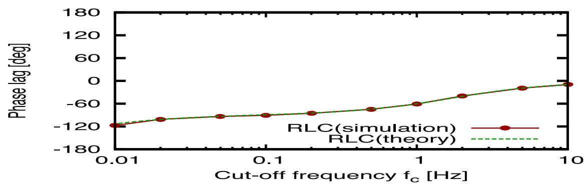
Fig. 8 Gains of the integral filter, the RC filter, and the RLC filter for different cutoff frequencies f_c when using the simple scenario; lines in Figs. (b) and (c) are almost identical.



(a) Integral filter



(b) RC filter



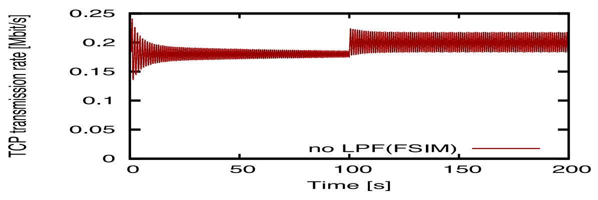
(c) RLC filter

Fig. 9 Phase lag of the integral filter, the RC filter, and the RLC filter for different cutoff frequencies f_c when using the simple scenario; lines in Figs. (b) and (c) are almost identical.

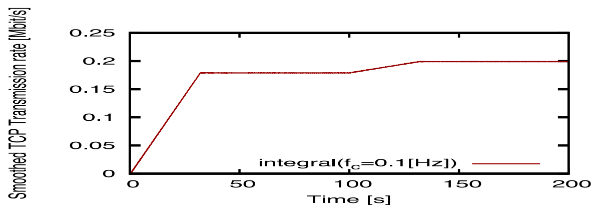
the equations. On the contrary, the simulated frequency response of the integral filter do not coincide with that obtained from the equations. This is because the frequency response the integral filter exhibits wild fluctuations.

These results confirm that the RC filter and the RLC filter implemented in FSIM have frequency responses consistent with theory regardless of the cutoff frequencies. As we discussed in Section 3.4, there is the effect of phase lag on simulation results and accuracy. The performance evaluator should use the proposed method with knowledge of the phase lag shown in Fig. 9.

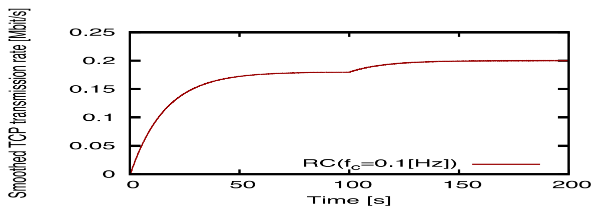
To confirm that the proposed method accurately obtains the



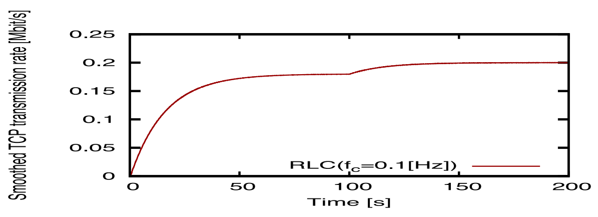
(a) FSIM



(b) Integral filter



(c) RC filter



(d) RLC filter

Fig. 10 Time evolutions of smoothed TCP transmission rates of the integral filter, the RC filter, and the RLC filter for the cutoff frequency $f_c = 0.1$ [Hz].

coarse behavior in flow-level simulations, we then compare simulation results of the original FSIM and FSIM with a low-pass filter for different cutoff frequencies. In this experiment, we use the simple and complex scenarios.

Figures 10 and **11** plot the time evolution of smoothed TCP transmission rates of the integral filter, the RC filter, and the RLC filter versus the cutoff frequencies $f_c = 0.1$ and 2.0 [Hz] when using the simple scenario. In these figures, we also show the time evolution of TCP transmission rate obtained from the original FSIM, for comparison. The TCP transmission rate obtained from the original FSIM oscillates with a frequency of approximately 1 [Hz]. In this simulation, we generated UDP traffic with the transmission rate of 8 [Mbit/s] with ON/OFF cycling every 100 [s] as the coarser behavior than the TCP flow in the original FSIM. From the result for $f_c = 0.1$ [Hz] shown in Fig. 10, we confirm that the flow-level simulator with the proposed method simulated only the coarse behavior (i.e., UDP traffic behavior). On the contrary, the result for $f_c = 2.0$ [Hz] shown in Fig. 11, confirms that the flow-level simulator with the proposed method simulated the coarse and fine network behavior (i.e., UDP traffic behavior and TCP flow behavior). If we carefully look at Fig. 11, the amplitudes of the smoothed TCP transmission rate for the low-pass filters are smaller than that yielded by the original FSIM. Hence, the cutoff frequency, f_c, of the performance evaluator should be

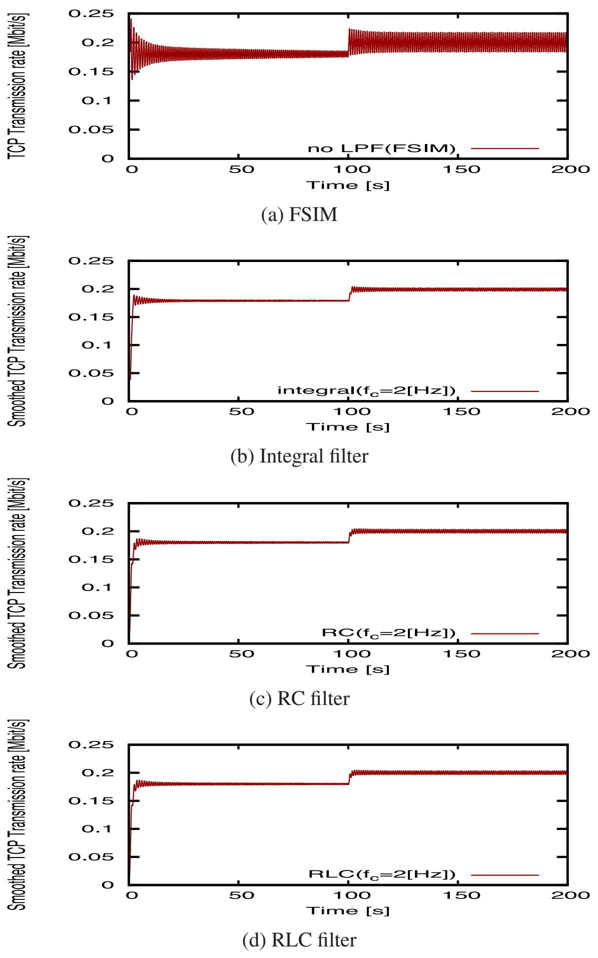


Fig. 11 Time evolutions of smoothed TCP transmission rates of the integral filter, the RC filter, and the RLC filter for the cutoff frequency $f_c = 2.0$ [Hz].

set considering the theoretical gain, Eqs. (5), (10) and (15), of the low-pass filter used.

From these results, we confirm that the proposed method yields the coarse behavior if the cutoff frequency is appropriately set.

Figures 12 and **13** plot the time evolution of the smoothed RED queue length of the integral filter, the RC filter, and the RLC filter versus the cutoff frequencies $f_c = 0.1$ and 2.0 [Hz] when using the complex scenario. In these figures, we also show the time evolution of RED queue obtained from the original FSIM, for comparison. From the result for $f_c = 0.1$ [Hz] shown in Fig. 12, we confirm that the flow-level simulator with the proposed method simulated only the coarse behavior. On the contrary, the result for $f_c = 2.0$ [Hz] shown in Fig. 13, confirms that the flow-level simulator with the proposed method simulated fine network behavior.

From these results, we confirm that the proposed method also yields the coarse behavior if the simulation scenario is complex.

4.3 Speed

Next, we investigate the speed of the proposed method by measuring the simulation execution time, which is the time required to simulate a 1,000 [s] trial, and the average step-size of the numerical solver.

Figures 14 and **15** show simulation execution times and av-

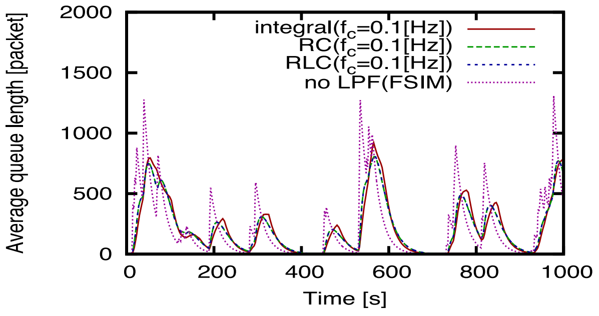


Fig. 12 Time evolution of smoothed RED queue length of the integral filter, the RC filter, and the RLC filter [Hz] with $f_c = 0.1$ [Hz] when using the complex scenario.

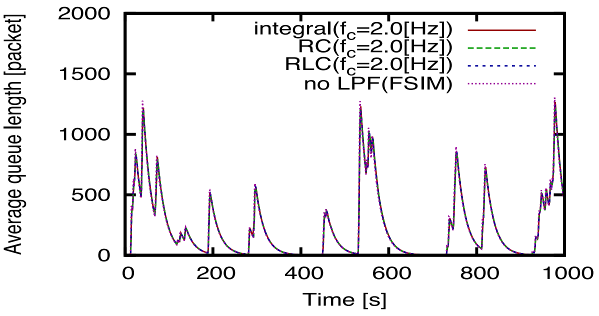


Fig. 13 Time evolution of smoothed RED queue length of the integral filter, the RC filter, and the RLC filter [Hz] with $f_c = 2.0$ [Hz] when using the complex scenario; all lines are almost identical.

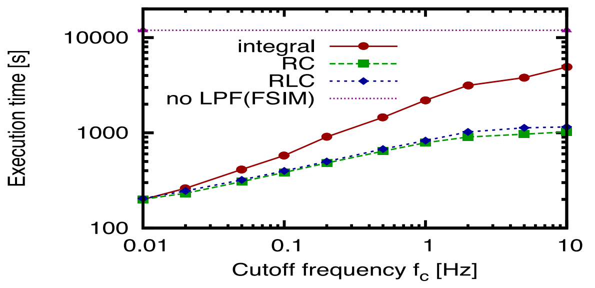


Fig. 14 Simulation execution times of the integral filter, the RC filter, and the RLC filter for different cutoff frequencies f_c when using the simple scenario.

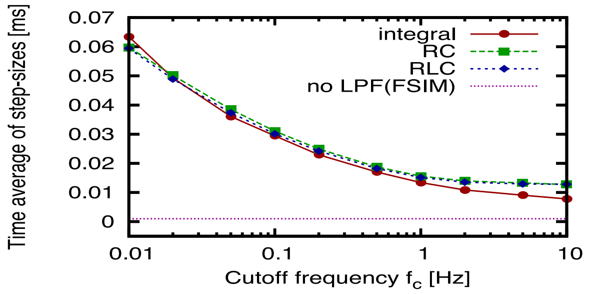


Fig. 15 Average step-size of the integral filter, the RC filter, and the RLC filter for different cutoff frequencies f_c when using the simple scenario.

erage step-sizes of the integral filter, the RC filter, and the RLC filter for different cutoff frequencies, f_c. To obtain these figures, we use the simple scenario. In Figs. 14 and 15, we also show the simulation execution time and average step-size obtained from the original FSIM, for comparison. The figures show that simulation execution times decrease as the cutoff frequency, f_c, of the low-pass filter decreases. This is because decreasing the cutoff frequency increases the average step-size of the low-pass filter.

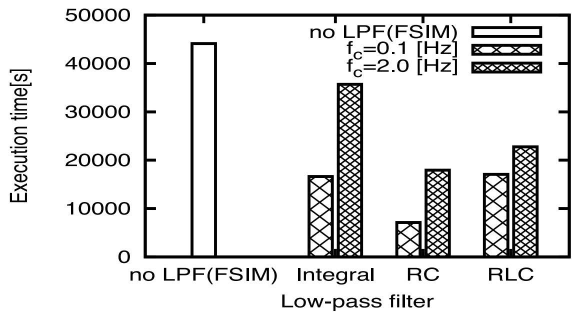


Fig. 16 Execution times of the integral filter, the RC filter, and the RLC filter for cutoff frequencies f_c = 0.1, 2.0 [Hz] when using the complex scenario.

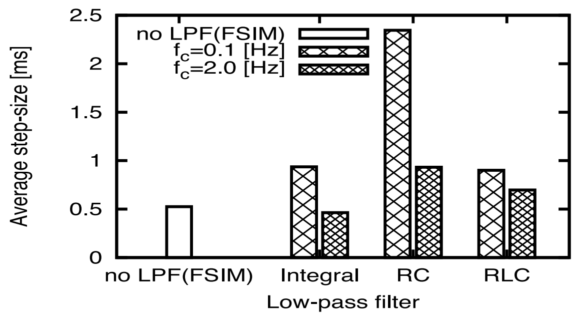


Fig. 17 Average step-sizes of the integral filter, the RC filter, and the RLC filter for cut-off frequencies f_c = 0.1, 2.0 [Hz] when using the complex scenario.

In this experiment with the simple scenario, the proposed method was up to 60 times faster than the original FSIM.

Figures 16 and **17** plot the execution times and average step-sizes corresponding to Figs. 12 and 13 with the complex scenario. From this figure, we confirm the acceleration effect of the low-pass filtering of flow-level simulation. In particular, the execution time with the RC filter with the cutoff frequency f_c = 0.1 [Hz] is approximately 4 times smaller than that for the original FSIM. Despite the approximately-same frequency response shown in Figs. 2 and 3, the execution times with the RLC filter are larger than that with the RC filter. This phenomenon is caused by the following reason. Since the RLC filter is given by a second order ODE, the stability of numerical calculation with the RLC filter is relatively low compared with the RC filter. Hence, the RLC filter should not drastically increase the step-size for the complex scenario.

In this experiment with the complex scenario, the RC filter is suitable for accelerating of flow-level simulation.

4.4 Memory Consumption

Finally, we investigate the memory consumption of the proposed method by measuring the maximum consumption during simulation execution.

Figure 18 plots maximum memory consumption of the integral filter, the RC filter, and the RLC filter versus the cutoff frequency, f_c. In Fig. 18, we also show the memory consumption obtained from the original FSIM, for comparison. From Fig. 18, (a) the maximum memory consumption of the integral filter increases as cutoff frequency f_c decreases, (b) regardless of cutoff frequency f_c, the RC and RLC filters have almost the same maximum memory consumption as the simulation with no low-pass filter. We

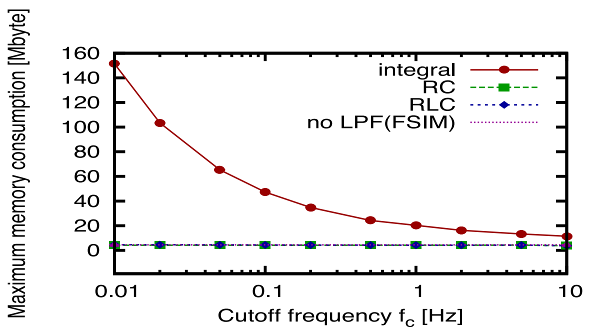


Fig. 18 Maximum memory consumptions of the integral filter, the RC filter, and the RLC filter for different cutoff frequencies f_c.

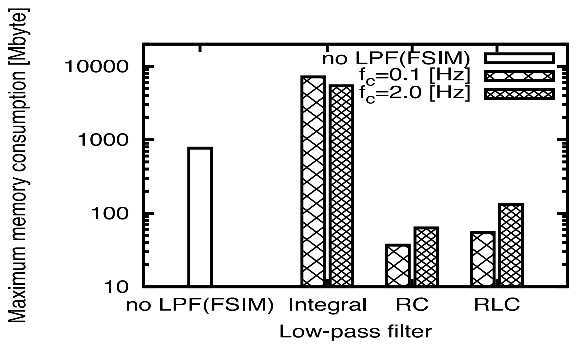


Fig. 19 Maximum memory consumptions of the integral filter, the RC filter, and the RLC filter for cut-off frequency f_c = 0.1, 2.0 [Hz] when using the complex scenario.

can explain the increase in memory consumption triggered by the integral filter from the ODE defined by Eq. (2). According to Eq. (2), the integral filter integrates network states over interval $[t - T, t]$. Since T increases as cutoff frequency f_c decreases, the integral filter must store the past state for long periods to integrate the network state over $[t - T, t]$. Storing the past state for long periods increases memory consumption.

Figure 19 plots maximum memory consumptions corresponding to Figs. 12 and 13 with the complex scenario. From this figure, we confirm that the low-pass filtering of flow-level simulation can be saved the memory consumption. This is caused by the large average step-size when using low-pass filters. FSIM stores the past network state to calculate the ODE of the fluid model. As the step-size increases, the memory size required by storing the past network state decreases. This phenomenon does not appear in Fig. 18. This is because the percentage of memory consumption for past network state is small since the number of TCP flows is small when using the simple scenario.

From the viewpoint of memory consumption, the RC and RLC filters are more effective than the integral filter.

5. Conclusion

In this paper, we have proposed a novel method for accelerating flow-level simulations; it omits simulations whose timescales are finer than that required by performance evaluation. We implemented three low-pass filters (Integral filter, RC filter, and RLC filter) into our flow-level simulator FSIM. Through experiments, we investigated the effectiveness of the proposed method for accelerating a flow-level simulation. Our findings show that the proposed method is 60 times faster than the conventional flow-level simulator when obtaining coarse behavior at timescale of minutes.

As future work, we are planning to investigate the effectiveness of the proposed method for large-scale networks, which has many network nodes.

Acknowledgments This work was supported by the Grant-in-Aid for Scientific Research (B) (24700072) by the Ministry of Education, Culture, Sports, Science and Technology (MEXT).

References

[1] Internet Systems Consortium: Internet Domain Survey, available from ⟨http://www.isc.org/solutions/survey⟩.

[2] ISI: The network simulator – ns2, available from ⟨http://www.isi.edu/nsnam/ns/⟩.

[3] Sakumoto, Y., Asai, R., Ohsaki, H. and Imase, M.: Design and Implementation of Flow-Level Simulator for Performance Evaluation of Large Scale Networks, *Proc. 15th Annual Meeting of the IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (*MASCOTS 2007*), pp.166–172 (2007).

[4] Liu, Y., Presti, F.L., Misra, V., Towsley, D. and Gu, Y.: Fluid models and solutions for large-scale IP networks, *Proc. ACM SIGMETRICS 2003*, pp.91–101 (2003).

[5] Kumar, S., Park, S. and Iyengar, S.S.: A loss-event driven scalable fluid simulation method for high-speed networks, *Computer Networks*, Vol.54, No.1, pp.112–132 (2010).

[6] Kim, H. and Hou, J.: Enabling network calculus-based simulation for TCP congestion control, *Computer Networks*, Vol.53, No.1, pp.1–24 (2009).

[7] Ohsaki, H., Ujiie, J. and Imase, M.: On Scalable Modeling of TCP Congestion Control Mechanism for Large-Scale IP Networks, *Proc. 5th IEEE/IPSJ International Symposium on Applications and the Internet* (*SAINT 2005*), pp.361–369 (2005).

[8] Barford, P. and Crovella, M.: A performance evaluation of hyper text transfer protocols, *ACM SIGMETRICS Performance Evaluation Review*, Vol.27, No.1, pp.188–197 (1999).

[9] Ericsson, M., Resende, M. and Pardalos, P.: A genetic algorithm for the weight setting problem in OSPF routing, *Journal of Combinatorial Optimization*, Vol.6, No.3, pp.299–333 (2002).

[10] Andersen, D., Balakrishnan, H., Kaashoek, F. and Morris, R.: Resilient Overlay Networks, *Proc. 18th ACM Symposium on Operating Systems Principles* (*SOSP '01*), pp.131–145 (2001).

[11] Buriol, L., Resende, M. and Thorup, M.: Survivable IP network design with OSPF routing, *Networks*, Vol.49, No.1, pp.51–64 (2007).

[12] Visioli, A.: *Practical PID Control*, Springer (2005).

[13] Kuriakose, C.: *Circuit Theory: Continuous and Discrete Time Systems, Elements of Network Synthesis*, Prentice-Hall of India Pvt.Ltd (2005).

[14] Mayergoyz, I. and Lawson, W.: *Basic Electric Circuit Theory: A One-Semester Text*, Academic Press (1997).

[15] Hairer, E., Nørsett, S.P. and Wanner, G.: *Solving Ordinary Differential Equations I: Nonstiff Problems*, Springer (2007).

[16] Dormand, J. and Prince, P.: A family of embedded Runge-Kutta formulae, *Journal of Computational and Applied Mathematics*, Vol.6, pp.19–26 (1980).

Yusuke Sakumoto received his 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 performance evaluation of computer networks. He is a member of IEEE, IPSJ and IEICE.

Hiroyuki Ohsaki received his M.E. degree in the Information and Computer Sciences from Osaka University, Osaka, Japan, in 1995. He also received his Ph. D. degree from Osaka University, Osaka, Japan, in 1997. He is currently an associate professor at Department of Information Networking, Graduate School of Information Science and Technology, Osaka University, Japan. His research work is in the area of traffic management in high-speed networks. He is a member of IEEE, IEICE, and IPSJ.

Makoto Imase received his B.E. and M.E. degrees in Information Engineering from Osaka University in 1975 and 1977, respectively. He received his D.E. degree from Osaka University in 1986. From 1977 to 2001, he was engaged in Nippon Telegraph and Telephone Corporation (NTT). From 2002 to 2012, he was a Professor of Graduate School of Information Science and Technology at Osaka University. He is currently a Visiting Professor of Graduate School of Information Science and Technology at Osaka University. His research interests are in the area of information networks, distributed systems and graph theory. He is a member of IPSJ, JSIAM, and IEICE.