Hierarchical Model Predictive Traffic Engineering

Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata,
Yousuke Takahashi, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto,

***Abstract*—Hierarchical traffic control is a promising approach for improving scalability in the face of network size. In this scheme, multiple controllers are introduced in a network, and these hierarchically decide operations. At the bottom layer, controllers decide specific operations in a small area, while controllers at the upper layer decide inter-area operations using abstracted information from the lower layers. These controllers depend mutually on controllers in other layers, which may cause control oscillations, disturbing the appropriate network state. The common way to handle such oscillations is to set the control interval of the upper layer to a large value. This approach, however, causes another problem: the delay of upper-level operations relative to environmental changes. To solve this problem, we introduce the concept of *model predictive control (MPC)* to hierarchical network control. In this method, each controller gradually changes operations based on the predicted future network state. By predicting the behavior of other controllers in the upper/lower layers, the controller can smoothly shift to the suitable operations. Furthermore, the impact of prediction error can be reduced by avoiding significant changes in operations. In this paper, we develop MPC-based hierarchical network control for effective *hierarchical traffic engineering (TE)*. Through extensive simulation, we show that the MPC-based hierarchical TE can avoid congestion even in cases where the existing TE method of setting long control intervals for the upper layer cannot accommodate dynamically changing traffic owing to operational delay.**

***Index Terms*—Model Predictive Control, Traffic Engineering, Topology Aggregation, Traffic Prediction**

I. Introduction

THE hierarchical control scheme is a promising approach to improving the scalability of network controls [1–6]. In this scheme, a network is divided hierarchically into multiple areas; the area in the lowest layer includes only a small number of nodes and links, and the area in the upper layer is constructed of multiple areas of the lower layer. One control server is deployed in each area, and each control server monitors the state of its corresponding area by collecting detailed information about the area from nodes within the area or about the aggregated information from the controllers of the lower layer. Then, each control server decides on control actions for the corresponding area. By doing so, effective network control can be achieved without a controller that considers detailed information from the whole network.

Tatsuya Otoshi, Yuichi Ohsita and Masayuki Murata are with the Graduate School of Information Science and Technology, Osaka University, Suita-shi, 565–0871, Japan(e-mail: t-otoshi, y-ohsita, murata@ist.osaka-u.ac.jp).

Yousuke Takahashi, Keisuke Ishibashi and Kohei Shiomoto are with NTT Network Technology Laboratories, NTT Corporation, Musashino-Shi, 180–8585, Japan(e-mail: takahashi.yousuke, ishibashi.keisuke, shiomoto.kohei@lab.ntt.co.jp).

Tomoaki Hashimoto is with Department of Mechanical Engineering, Osaka Institute of Technology, Osaka-shi, 535–8585, Japan(e-mail: tomoaki.hashimoto@oit.ac.jp).

Of course, there are several challenging problems in such hierarchical network control. One difficult problem is how to avoid the oscillation of operations. When the control server at the upper layer changes operations across the lower areas, the network states in the lower areas may change in a way that the control server at the lower layer does not expect. For example, consider a control server at the upper layer reallocating network resources at area A to another area B which requires more resources. Owing to this reallocation, the network resources at area A become smaller than expected by the controller of area A. On the other hand, the operations in lower areas change the state of those areas and stimulate operation changes at the upper layers. For example, when the area A is congested, the control server of the upper area considers that area A requires more resources. However, congestion may be mitigated by the control server of area A. In this case, the resources reallocated without knowledge of the behavior of the controller in area A are not necessary, and are again reallocated to other areas which are regarded as areas requiring more resources. Such interaction between layers occurs repeatedly, and global operation oscillates.

The common way to handle such unexpected oscillation in hierarchical network control is to set the control interval of the upper layer to a large value [7, 8]. By doing so, the control servers of the lower layers change operations with sufficient time before the upper layer changes inter-area operations. However, a large control interval increases the time required to respond to environmental changes; if all resources are used up in a certain area, the lack of resources cannot be mitigated until the control server of the upper layer reallocates inter-area resources. This tendency of large and/or frequent environmental changes is remarkable because of wide deployment of, e.g., content distribution networks (CDNs), user mobility, and so on.

To solve the above problem in existing hierarchical network control, we propose a hierarchical network control method with a new mechanism to avoid control oscillation without setting a large control interval. Our method is based on *model predictive control (MPC)* [9, 10], which changes the input adequately based on predictions so as to maintain system output near a target value. According to the concept of MPC, each control server predicts the traffic changes caused by the behavior of other control servers and then decides its own operations. By predicting the behavior of other control servers, the control servers are expected to smoothly shift to appropriate operations. Although operations may still oscillate if the controller changes operations based on an inaccurate prediction, MPC can effectively avoid the impact of prediction error by restricting large changes in its operation so as not to affect the other controllers, and by correcting

predictions with newly observed information at the next time slot. Therefore, MPC-based hierarchical control is capable of handling environmental changes without oscillations, which is the main subject of the current paper. To develop an effective MPC-based hierarchical control method, we utilize the idea of *hierarchical traffic engineering (TE)* [3, 5, 6]. In hierarchical TE, each control server changes the routes of the flows within its area to avoid congestion. We then apply MPC to the hierarchical TE, which we call *hierarchical MP-TE*. Through simulation of MP-TE, we show that our MPC scheme significantly absorbs the impact of interaction among layers without setting a large control interval at the upper layer.

We have already applied MPC to TE in the case where a central server controls the whole network [11, 12]. Our previous work showed that MPC-based TE follows the changing traffic even when prediction error occurs. However, in hierarchical TE, interaction between layers occurs, which is not considered in our previous work, causing route oscillation. To avoid route oscillation, in MPC-based hierarchical TE, (1) each controller predicts not only its own traffic variation but also the behavior of other controllers, and (2) avoids significant route changes which have large impact on other controllers. Through simulation, we demonstrate that such control avoids route oscillation with a short control interval, and that hierarchical MP-TE can accommodate changing traffic, which the existing hierarchical TE cannot accommodate. Additionally, we investigate the appropriate control policy for controllers at each layer based on the role of the layer in hierarchical TE. As a result, we find differences in appropriate control policies between the upper and lower layers: the controller at the upper layer should change routes more gradually while the control policy of the lower layer does not have a significant impact.

The rest of this paper is organized as follows. Section II surveys related work. Section III explains the overview of hierarchical network control. Section IV explains the framework of MPC-based hierarchical network control. In Section V, we propose a new hierarchical TE method called hierarchical MP-TE based on an MPC-based hierarchical network framework. Section VI evaluates hierarchical MP-TE. Section VII presents our concluding remarks.

II. Related work

A. Hierarchical Network Control

Hierarchical network control [1–4] is a promising approach for controlling large networks without a large control overhead and attendant computational complexity. In hierarchical network control, the network is hierarchically divided into multiple areas. A controller is deployed in each area of each layer. Each controller collects local information and calculates optimal operations within its area. Since each controller manages a relatively small network, large overhead for the controller is avoided.

One of the most representative cases of hierarchical network control is *hierarchical routing* [6, 7] in which each controller calculates routes so as to achieve a desired communication performance. For instance, Lui *et al.* proposed a hierarchical routing method that determines the route for each connection request so that the required bandwidth and delay are satisfied. In this method, the upper layer first calculates the inter-area routes based on the aggregated information about the bandwidth and delay within each area. Then, each area of the lower layers determines the inner-area routes with the actual delay and bandwidth observed within the local area.

The most challenging problem in hierarchical network control is oscillation due to interference between layers. The common way to handle oscillation is to set the control interval of the upper layer to a large value [7, 13]. For instance, Chang *et al.* used multiple policies for updating routing information to avoid route oscillation [7], which directly sets a long update time or introduces a threshold so that the controller does not update information unless the network state exceeds the threshold. Such methods, however, delay upper layer operations since the upper layer does not change routes unless the routing information is updated. To solve this problem, we propose a hierarchical network control method based on MPC that can avoid oscillations without setting a large control interval.

B. MPC and Its Application in Network Control

MPC is a method of system control based on predictions of system dynamics [9, 10]. MPC effectively handles environmental changes by combining the feedback and feedforward controls, whose detail is given in Subsection IV-A. Since systems often encounter dynamic changes in real environments, MPC is expected to be applied in various applications such as chemical plant controls, transportation controls, network controls, etc.

In our previous work [11], we proposed a non-hierarchical TE method based on MPC called *MP-TE*. In this method, the central control server behaves as an MPC controller that inputs routes to the network such that the link load is kept lower than a desired level. Furthermore, we developed a TE method called *SMP-TE* [12] to improve the robustness of MP-TE to prediction errors. In SMP-TE, the control server considers not only the expected value of future traffic but also its probability distribution in order to guarantee that the risk of congestion occurrence is less than a predefined probability.

In hierarchical TE, interaction among layers, which was not considered in our previous work on MPC-based TE, causes route oscillation and disturbs the controllers in accommodating traffic. Therefore, in this paper, we newly propose an MPC-based hierarchical TE method that considers interaction between layers. To avoid route oscillation caused by interaction between layers, each controller in hierarchical MP-TE predicts the behavior of other controllers and avoids significant route changes in order to avoid significantly affecting other controllers. Through simulation, we demonstrate the effectiveness of hierarchical MP-TE in handling interactions compared with the existing hierarchical TE, which sets a long control interval at the upper layer. In addition, we examine appropriate parameters for hierarchical MP-TE according to the role of each layer.

III. Hierarchical Network Control

In hierarchical network control, the network is divided hierarchically into areas; the areas of the lowest layer are constructed of a small number of nodes, and the areas of the upper layer are constructed of multiple areas from the lower layer. Hereafter, we call the set of hierarchically divided networks the *hierarchical network*. A control server deployed in each area of each layer optimizes network operations within the area based on locally collected information about the network state. Thus, the observation overhead and computational complexity of each control server are kept small even when the network size becomes large. In the rest of this section, we describe the control methodology and problems of hierarchical network control.

We introduce three vectors; $\boldsymbol{z}(k)$ is a vector indicating the state of the network at time step k, $\boldsymbol{X}(k)$ is a vector indicating the observed information, and $\boldsymbol{u}(k)$ is a vector indicating the input from the controller. The observed information $\boldsymbol{X}(k)$ reflects the network state $\boldsymbol{z}(k)$.

$$\boldsymbol{X}(k) = g(\boldsymbol{z}(k)), \tag{1}$$

where $g()$ is a function mapping the network state to the observed information. The input from the controller changes the state of the network. That is,

$$\boldsymbol{z}(k) = f(\boldsymbol{z}(k-1), \boldsymbol{u}(k)) + \boldsymbol{\epsilon}(k), \tag{2}$$

where $f()$ is a function indicating the network state after the input $\boldsymbol{u}(k)$, and $\boldsymbol{\epsilon}(k)$ is a vector indicating the disturbance. In network control, the controller observes $\boldsymbol{X}(k)$, estimates the state of the network $\hat{\boldsymbol{z}}(k) = g^{-1}(\boldsymbol{X}(k))$, and sets the input $\boldsymbol{u}(k)$ so as to set $\boldsymbol{z}(k)$ into an appropriate state. As the network becomes large, the sizes of $\boldsymbol{X}(k)$, $\boldsymbol{z}(k)$, and $\boldsymbol{u}(k)$ become large, causing high observation overhead and computational cost for the controller if one controller controls the whole network.

In hierarchical network control, the network is divided hierarchically into areas, and a control server is deployed in each area. The control server in the area a of the lowest layer observes the local information $\boldsymbol{X}^{1;a}(k)$, which reflects the network state within the area a, $\boldsymbol{z}^{1;a}(k)$, which is a subset of the network state $\boldsymbol{z}(k)$. The control server calculates the input $\boldsymbol{u}^{1;a}(k)$, which is a subset of $\boldsymbol{u}(k)$ and has an impact only on $\boldsymbol{z}^{1;a}(k)$. $\boldsymbol{u}^{1;a}(k+1)$ is determined so as to set $\boldsymbol{z}^{1;a}(k+1)$ into an appropriate state.

In the upper layer m, the control server for area b collects the aggregated information $\boldsymbol{X}^{m;b}(k)$ from the areas of the lower layers. $\boldsymbol{X}^{m;b}(k)$ reflects the network state within area b, $\boldsymbol{z}^{m;b}(k)$, which includes the network states in the multiple areas of the lower layers and the network states that are not maintained by any area of the lower layers. The control server sets the input $\boldsymbol{u}^{m;a}(k)$, which has impact on the network state of the different lower layer areas but has an impact only on $\boldsymbol{z}^{m;a}(k)$ at the layer m. $\boldsymbol{u}^{m;a}(k)$ is set so as to set $\boldsymbol{z}^{m;b}(k)$ into an appropriate state.

The oscillation of operations is one of the important problems in hierarchical control. In hierarchical control, each control sever determines its input independently. For example, the control server at the lowest layer determines $\boldsymbol{u}^{1;a}(k+1)$ so that $\boldsymbol{z}^{1;a}(k+1)$ achieves an appropriate state. However, $\boldsymbol{z}^{1;a}(k+1)$ is also affected by the input of the upper layer $\boldsymbol{u}^{2;b}(k+1)$, which is determined independently by the control server at the upper layer. As a result, $\boldsymbol{z}^{1;a}(k+1)$ deviates from the state expected by the controller of the lowest layer, and the controller must change the input. Similarly, the input of the lower layer $\boldsymbol{u}^{1;a}(k+1)$ causes deviation of $\boldsymbol{z}^{2;b}(k+1)$ from the status expected by the controller of the upper layer, and the controller of the upper layer also changes the input $\boldsymbol{u}^{2;b}(k+1)$.

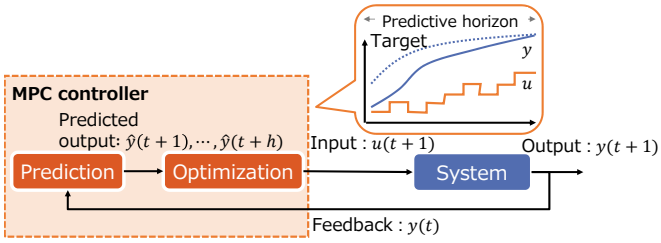


Fig. 1. Overview of MPC.

The typical approach to handling control oscillation is setting a long control interval at the upper layer. We denote s_m as the control interval of the layer m. The controller of the layer m observes the network state every s_m time steps by averaging the fine-grained observation as $\bar{\boldsymbol{X}}^{m;a}(k) = \frac{1}{s_m}\sum_{i=(k-1)s_m}^{ks_m-1} \boldsymbol{X}^{m;a}(i)$. By doing so, the operations of the lower layers converge before the operations of the upper layer change. This method, however, requires a long time to conduct appropriate operations because the long control interval delays the operations of the upper layer.

IV. Hierarchical Network Control Based on MPC

We introduce the concept of MPC into hierarchical network control to solve the above mentioned problems. In this section, we first explain MPC, and then propose hierarchical network control based on MPC.

A. Model Predictive Control

First, we briefly explain the concept of MPC. MPC is a method of system control that has been studied in recent years [9,10] based on predictions of system dynamics. Figure 1 shows an overview of MPC. An MPC controller sets an input so as to maintain system performance close to an operator-specified target. Unlike traditional system control, the MPC controller predicts how the output changes in order to calculate inputs for the *predictive horizon* over time steps $[t+1, t+h]$, where h is the distance to the predictive horizon. We denote the input and output at the time step k by $u(k)$ and $v(k)$, respectively. The MPC controller calculates $u(k)(k \in [t+1, t+h])$ so as to keep $v(k)$ close to the target value $r_v(k)$. In other words, the MPC controller minimizes an objective function $J_1 = \sum_{k=t+1}^{t+h} \|v(k) - r_v(k)\|^2$, where $\|\cdot\|$ represents the Euclidean norm:

$$(u(t+1), \cdots, u(t+h)) = \underset{(u(t+1), \cdots, u(t+h))}{\arg\min}\ J_1. \tag{3}$$

To calculate the optimal input, future outputs $v(t+1), \cdots, v(t+h)$ must be predicted from inputs $u(t+$

$1), \cdots, u(t+h)$. The future output under a given input is calculated by a system model that represents the system dynamics. In system control, a system model is often represented by a mathematical formula, the *state-space representation*, described as

$$z(k) = \phi(k, z(k-1), u(k)) \tag{4}$$

$$v(k) = \psi(k, z(k), u(k)), \tag{5}$$

where $z(k)$ is the state of the system at the time step k, and ϕ, ψ are functions that respectively map the current state and input onto the next state and output.

Modeling the system by a mathematical formula, however, may entail modeling errors, such as the use of ϕ, ψ, that do not accurately represent actual system dynamics. If the controller directly calculates the input based on such an inaccurate prediction, the controller changes the input unnecessarily. This unnecessary change in input causes destabilization of the system. The MPC controller therefore restricts the amount of allowed change to inputs, which mitigates the influence of prediction errors. We denote the amount of change in the input at the time step k by $\Delta u(k) = u(k) - u(k-1)$, and the aggregated amount of change during the predictive horizon by $J_2 = \sum_{k=t+1}^{t+h} \|\Delta u(k)\|^2$. Instead of the input values determined by Eq. (3), the controller calculates the input values by solving the following optimization problem:

$$(u(t+1), \cdots, u(t+h)) = \underset{(u(t+1), \cdots, u(t+h))}{\arg\min} (1-w)J_1 + wJ_2 \tag{6}$$

where $0 \leq w \leq 1$ is a parameter for weighting the two objective functions J_1 and J_2.

Though the controller calculates the input value during $[t+1, t+h]$, the controller actually implements only the first of the calculated inputs $u(t+1)$. Then, the MPC controller observes the output and corrects the prediction by using the output value as feedback. After prediction correction, the MPC controller recalculates the input value for the next time slot with the corrected prediction.

B. Applying MPC to Hierarchical Network Control

In this subsection, we propose hierarchical network control based on the MPC. In this method, each control server performs as an MPC controller that determines the local operations within its area. In the area a at the layer m, the input values are local operations $\boldsymbol{u}^{m;a}(k)$ and the output is the local network state $\boldsymbol{z}^{m;a}(k)$.

To estimate how the network states change, the control server should predict the behavior of the operations of other controllers. Since the behavior of other controllers is reflected in the local observations, the control server predicts how the future values of $\boldsymbol{X}^{m;a}(k)$ will be changed by the impact of other controllers. Using the predicted values $\hat{\boldsymbol{X}}^{m;a}(k)$, the controller calculates the future states $\hat{\boldsymbol{z}}^{m;a}(k)$ for deciding the input.

As mentioned in Section III, in hierarchical network control, the interaction of operations among layers causes control oscillation. The origin of the oscillation is that each control server calculates its own operations with uncertainty regarding the behaviors of other controllers. Thus, absorbing the impact of the prediction error in the behaviors of other controllers is critical in avoiding control oscillation.

In MPC, the controller overcomes the uncertainty of the prediction by avoiding significant changes in the input value. In addition, avoiding significant changes in the input value absorbs the interaction between layers. Therefore, in similar way as in Section IV-A, the control server minimizes the objective functions J_1 and J_2, which are determined as follows:

$$J_1 = \sum_{k=t+1}^{t+h} \|\hat{\boldsymbol{z}}^{m;a}(k) - r_z(k)\|^2 \tag{7}$$

$$J_2 = \sum_{k=t+1}^{t+h} \|\Delta \boldsymbol{u}^{m;a}(k)\|^2 \tag{8}$$

where $r_z(k)$ is the target value of $\boldsymbol{z}^{m;a}(k)$ and $\Delta \boldsymbol{u}^{m;a}(k) = \boldsymbol{u}^{m;a}(k) - \boldsymbol{u}^{m;a}(k-1)$. That is, the control server decides the future operations according to:

$$(\boldsymbol{u}^{m;a}(t+1), \cdots, \boldsymbol{u}^{m;a}(t+h)) = \underset{(\boldsymbol{u}^{m;a}(t+1), \cdots, \boldsymbol{u}^{m;a}(t+h))}{\arg\min} (1-w)J_1 + wJ_2. \tag{9}$$

The control server actually implements only the first of the calculated inputs $\boldsymbol{u}^{m;a}(t+1)$. Then, the control server observes $\boldsymbol{X}^{m;a}(t+1)$ and corrects the prediction $\hat{\boldsymbol{X}}^{m;a}(t+2), \cdots, \hat{\boldsymbol{X}}^{m;a}(t+h+1)$. After the prediction correction, the control server recalculates the operations for the next time step.

V. Hierarchical MP-TE

In this section, we propose a new hierarchical TE method based on the MPC-based hierarchical network control framework introduced in Section IV. In this section, we first formulate hierarchical TE; then we propose a new TE method.

A. Hierarchical TE

In hierarchical TE, multiple controllers are deployed in a hierarchy of areas to calculate routes within the areas. In the upper layer, the control server calculates routes of the flows between areas of the lower layers using the aggregated network topology. The control server at the lower layer calculates the specific routes of the flows within the area. In this subsection, we formulate hierarchical TE.

1) Construction of the Hierarchical Network:

First, we describe the construction of the hierarchical network, which is conducted by *area partitioning* and *topology aggregation*.

a) Area Partitioning: Area partitioning divides the network into multiple areas so that each area includes the connected subnetwork of the original network. Similarly to [1, 5, 6], we assume that the network is divided so that any nodes are included in one of the areas, and no nodes are included in multiple areas. Thus, the set of links within area a includes the set of links $\{(i,j) \in E | i; j \in V_a\}$, where E is the set of all links of the original network, and V_a is the set of nodes included in area a. In this link set, the links connecting

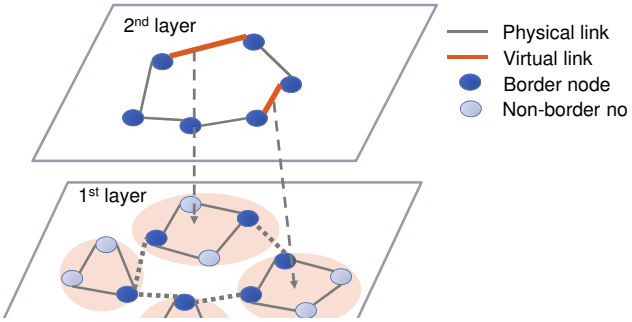


Fig. 2. The hierarchical network model.

nodes within different areas are not included in any areas, and are included in the upper layer.

Although we can use any area-partitioning strategy, e.g., [1], we manually divide the network into areas in the evaluation.

b) Topology Aggregation: Given area partitioning, the control server of the upper layer maintains the aggregated network topology instead of the original network topology so as to avoid large calculation time. Topology aggregation replaces each area of the lower layer with the set of a small number of nodes and links connecting them. There are many methods of aggregating topology information [5,13], and there is a trade-off between information accuracy and topology complexity.

In this paper, we use full-mesh topology to aggregate so as to maintain accurate information regarding the nodes at the borders of the areas. By using full-mesh topology, the abstracted topology of an area includes the set of nodes at the border and the set of links between all pairs of nodes at the border. Hereafter, we call the links generated by topology aggregation the *virtual links*. Figure 2 shows an example of the hierarchical network. In this network, the upper layer includes the virtual links and the physical links between different areas.

2) Traffic Engineering in Each Area: After deploying a control server at each area, each control server periodically 1) collects information on the traffic rates and link capacities within its area, and 2) calculates routes based on observations and configures network devices within its area.

a) Collection of Information:

The control server at area a of layer m collects the locally observable variables $\boldsymbol{X}^{m;a}(k)$. In hierarchical traffic engineering, $\boldsymbol{X}^{m;a}(k)$ includes the traffic rates $\boldsymbol{x}^{m;a}(k)$ and the residual link capacities $\boldsymbol{C}^{m;a}(k)$ within the area.

The traffic rates $\boldsymbol{x}^{m;a}(k)$ form a vector whose element $x_{i;j}^{m;a}(k)$ is the traffic rate from nodes i to j. Each node monitors traffic rates per source and destination address pair. The control server collects the traffic rates monitored by each node and calculates the sums of traffic rates for the flows from one node to another within the area.

The residual link capacities $\boldsymbol{C}^{m;a}(k)$ form a vector whose element $C_i^{m;a}(k)$ is the residual capacity of the link i, which represents how much additional traffic can be accommodated at link i. If the residual capacity is negative, then the link is overloaded and the controller should move traffic on that link to other links.

As mentioned in section V-A1, there are two types of links, i.e., physical and virtual. Since the physical link capacity is constant until the upgrade of link capacities, the actual capacity c_l of the physical link l is known by each control server. Thus, the residual capacity of the physical link l is represented by using only the local variables in

$$C_l^{m;a}(k) = c_l - y_l^{m;a}(k) \tag{10}$$

where $y_l^{m;a}$ is the traffic load on link l at area a of layer m.

On the other hand, the residual capacity of the virtual link depends on the network state of the lower layer. In this paper, the residual capacity of the virtual link is set to the total residual capacity of all available paths between both ends of the virtual link. That is, the capacity of the link l is set by

$$C_l^{m;a}(k) = \sum_{p \in P(i)} \min_{i \in L(p)} (C_i^{m-1;b}(k) - y_i^{m-1;b}(k)) \tag{11}$$

where area b is the area in which the virtual link l is constructed, $P(i)$ is the set of paths in the inner-area whose starting and ending nodes are the same as those of virtual link i, and $L(p)$ is the set of links included in path p. In this equation, $\min_{l \in L(p)}(C_l^{m-1;b}(k) - y_l^{m-1;b}(k))$ denotes the residual capacity of path p, which is equal to the residual capacity of the bottleneck link on the path, and the residual virtual link capacity sums the residual capacity for all available paths.

b) Route Calculation:

The control server calculates routes within the area based on the observed information $\boldsymbol{x}^{m;a}(k)$ and $\boldsymbol{C}^{m;a}(k)$. Here, the control variables $\boldsymbol{u}^{m;a}(k)$ include the routing matrix $R^{m,a}(k)$, whose element $R_{i;j}^{m;a}(k)$ indicates the fraction of traffic on the flow j that traverses the available path i. We also define the appropriate network state as the state in which no congestion occurs in the area. Thus, the control server adjusts $R^{m,a}(k)$ so as to accommodate traffic without congestion.

To achieve traffic accommodation, we introduce a metric called *excess traffic*. The excess traffic $\zeta_l^{m;a}(k)$ on the link l is defined by

$$\zeta_l^{m;a}(k) = [\Delta y_l^{m;a}(k) - C_l^{m;a}(k)]^+ \tag{12}$$

where $\Delta y_l^{m;a}(k) = y_l^{m;a}(k) - y_l^{m;a}(k-1)$ is the additional traffic on the link l caused by the route change at time step k, and $[x]^+$ equals x when $x \geq 0$ and equals 0 otherwise. When $\zeta_l^{m;a}(k)$ is zero, the additional traffic of link l falls under the residual capacity, meaning that congestion is avoided at link l. Therefore, the control server adjusts the routes $R^{m;a}(k)$ so that $\zeta_l^{m;a}(k)$ are minimized for all links. We also define $\boldsymbol{\zeta}^{m;a}(k)$ as a vector whose element is $\zeta_l^{m;a}(k)$.

To determine the appropriate routes, the control server has to calculate the value $\zeta_l^{m;a}(k)$ from local observation values $\boldsymbol{x}^{m;a}(k), \boldsymbol{C}^{m;a}(k)$ and local routes $R^{m,a}(k)$. According to the definition of $\zeta_l^{m;a}(k)$ in Eq. (12), the controller has to estimate how the link load $y_l^{m;a}(k)$ changes by setting the local routes $R^{m,a}(k)$. The control server calculates the link load based on the following relation between link and flow traffic:

$$\boldsymbol{y}^{m;a}(k) = G^{m;a} \cdot R^{m;a}(k) \cdot \boldsymbol{x}^{m;a}(k) \tag{13}$$

where $\boldsymbol{y}^{m;a}(k)$ is a vector whose elements represent the link load $y_l^{m;a}(k)$, and $G^{m;a}$ is a matrix whose element $G_{i;j}^{m;a}$ is 1 if the available path j traverses the link i and 0 otherwise.

At the time step t, the control server does not know the actual traffic rates and virtual link capacity at the next time step. Thus, the control sever uses the observation values $\boldsymbol{x}^{m;a}(t)$ and $C_l^{m;a}(t)$ instead of the actual values at $t+1$ to estimate the excess traffic $\zeta_l^{m;a}(t+1)$. As a result, the routes at time step $t+1$ are determined as the solution of the following optimization problem:

$$minimize: \left|\left|\hat{\boldsymbol{\zeta}}^{m;a}(t+1)\right|\right|^2 \tag{14}$$

$$subject\ to: \boldsymbol{y}^{m;a}(t+1) = G^{m;a} R^{m;a}(t+1) \cdot \boldsymbol{x}^{m;a}(t) \tag{15}$$

$$\forall l, \zeta_l^{m;a}(t+1) = [\Delta y_l^{m;a}(t+1) - C_l^{m;a}(t)]^+ \tag{16}$$

$$\forall f, p, R_{p;f}^{m;a}(l) \in [0,1] \tag{17}$$

$$\forall f, \sum_{p \in \wp^{m;a}(f)} R_{p;f}^{m;a}(t+1) = 1 \tag{18}$$

where $\wp^{m;a}(f)$ is the set of available paths of flow f and $N_L^{m;a}, N_P^{m;a}$ are the numbers of links and paths, respectively. Here, $\boldsymbol{x}^{m;a}(t), G^{m;a}, C_l^{m;a}(t)$ are given variables and $R^{m;a}(t+1), \boldsymbol{y}^{m;a}(t+1), \boldsymbol{\zeta}^{m;a}(t+1)$ are the variables to be optimized. Eq. (15) represents the relation between the traffic rates of the flows and links. Eq. (16) is the definition of ζ. Eqs. (17) and (18) mean that all traffic on each flow is allocated to some available paths.

Since the observation values $\boldsymbol{x}^{m;a}(t)$ and $C_l^{m;a}(t)$ are different from the actual state of the next time slot, the controller sometimes sets inappropriate routes, causing an oscillation of routes. The common method to avoid routing oscillation is to set a long control interval at the upper layer. However, setting a long control interval induces delay in response to the changing network state.

B. Hierarchical MP-TE

In this subsection, we show the hierarchical TE method called hierarchical MP-TE, which is based on the MPC methodology. Similarly to the simple hierarchical TE mentioned in Section V-A, the network is divided into multiple areas, and multiple control server are deployed in the areas. The control server observes the traffic rates of flows and residual link capacities in a similar way to V-A2a. Based on the observed values, the control server predicts future traffic rates and residual link capacities. Then, the control server calculates routes using the prediction, and implements the routes in the network. In the rest of this subsection, we explain the prediction and route calculation processes, which contain the main difference from the simple hierarchical TE method.

1) Prediction: Based on previously observed values, each control server predicts future traffic rates and residual link capacities. Although the controller can adopt any prediction model, e.g., ARIMA [14, 15], ARCH [16], GARCH [17], or neural networks [18, 19], we use a simple prediction method in this evaluation.

The prediction we use in evaluation is determined as follows. First, the control server at area a of layer m finds a best-fit straight line $l(k) = ak + b$ that minimizes the sum of squared distances from the previously observed traffic $x^{m;a}(t-\tau), x^{m;a}(t-\tau+1), \cdots, x^{m;a}(t) (\tau \leq 1)$, denoted as $\sum_{k=0}^{s}(x^{m;a}(t-\tau+k) - l(t-\tau+k))^2$. The control server then obtains the future traffic rate as $\hat{x}^{m;a}(t+k) = l(t+k)$. In a similar way, the control server predicts the future residual capacity $\hat{C}^{m;a}(t+k)$. Even if traffic changes linearly, this prediction method cannot predict future traffic and residual capacity accurately because the traffic rates and residual capacities maintained by each controller are affected by the route changes other layers. Using this simple prediction method, we show that hierarchical MP-TE works well even with an inaccurate prediction method. In the evaluation, we set $\tau = 1$.

2) Route Calculation: After the prediction of traffic rates of flows and residual link capacities, the control server calculates routes using predicted values. As mentioned in Section V-A2b, observable variables $\boldsymbol{X}^{m;a}(k) = (\boldsymbol{x}^{m;a}(k), \boldsymbol{C}^{m;a}(k))$, control variables $\boldsymbol{u}^{m;a}(k) = R^{m;a}(k)$, and the appropriate network state is defined as $\boldsymbol{\zeta}^{m;a}(k) = \boldsymbol{0}$. Then, according to Section IV-B, the control server calculates the routes by solving the following optimization problem:

$$minimize: \sum_{k=t+1}^{t+h} \left(\frac{1-w}{N_L^{m;a}} \left\| \frac{\hat{\boldsymbol{\zeta}}^{m;a}(k)}{Z^{m;a}} \right\|^2 + \frac{w}{N_P^{m,a}} \|\Delta R^{m;a}(k)\|^2 \right) \tag{19}$$

$$subject\ to: \forall k, \hat{\boldsymbol{y}}^{m;a}(k) = G^{m;a} \cdot R^{m;a}(k) \cdot \hat{\boldsymbol{x}}^{m;a}(k) \tag{20}$$

$$\forall k, l, \hat{\zeta}_l^{m;a}(k) = [\Delta \hat{y}_l^{m;a}(k) - \hat{C}_l^{m;a}(k)]^+ \tag{21}$$

$$\forall k, f, p, R_{p;f}^{m;a}(l) \in [0,1] \tag{22}$$

$$\forall k, f, \sum_{p \in \wp^{m;a}(f)} R_{p;f}^{m;a}(k) = 1 \tag{23}$$

where $\hat{\boldsymbol{y}}^{m;a}(k), \hat{C}_l^{m;a}(k), \hat{\boldsymbol{\zeta}}^{m;a}(k)$ are the predicted values of link load, residual link capacity, and excess traffic, respectively. $Z^{m;a} = \max_{l,k}[\{G^{m;a} \cdot R^{m;a}(t) \cdot \Delta\hat{\boldsymbol{x}}^{m;a}(k)\}_l - C_l^{m;a}(k)]^+$ is the maximum excess traffic if the current routes $R^{m;a}(t)$ are used during the predictive horizon. Eq. (19) is the objective function, which is the weighted summation of excess traffic $\boldsymbol{\zeta}^{m;a}(k)$ and the amount of route change $\Delta R^{m;a}(k)$. To clarify the effect of the weighting parameter w, we normalize the objective function by dividing $\hat{\boldsymbol{\zeta}}^{m;a}(k)$ by $Z^{m;a}$ and dividing the excess traffic on links and route changes on paths by $N_L^{m;a}$ and $N_P^{m;a}$, respectively.

Although the above optimization problem is not defined when $Z^{m;a} = 0$, this case is not critical for TE because the current routes $R^{m;a}(t)$ minimize both $\boldsymbol{\zeta}^{m;a}(k)$ and $\Delta R^{m;a}(k)$ when $Z^{m;a} = 0$. Therefore, in this paper we calculate the routes using the above optimization problem only when $Z^{m;a} \neq 0$.

VI. Evaluation

In this section, we evaluate MP-TE by simulation to verify how well the MPC concept performs in a hierarchical control scheme. At first, we use stationary traffic to demonstrate that hierarchical MP-TE can avoid routing oscillations by absorbing the interactions between layers even with a short control time interval. Second, we demonstrate the behavior of MP-TE under dynamic traffic with unpredictable fluctuations, which is a more realistic situation encountered in actual networks.

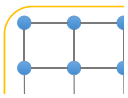


Fig. 3. Lattice topology with 64 nodes.

A. Evaluation in the Stationary Traffic Case

We first use stationary traffic to evaluate routing convergence with interactions between layers. Our interest here is whether hierarchical MP-TE achieves routing convergence by avoiding significant route changes and whether the short control interval helps in achieving quick responses to traffic changes. To clarify these questions, we compare hierarchical MP-TE with simple hierarchical TE and vary the control interval of the upper layer.

1) Simulation Environment:

a) Network Topology: In the following evaluation, we use the lattice topology shown in Figure 3. The network contains 64 nodes, and all links have the same link capacity of 2×10^9 units. We divide the network into four areas as shown in the figure.

b) Traffic: To investigate the interaction between layers, we generate traffic so that congestion cannot be solved by route changes at the lowest layer. We generate a traffic pattern such that traffic in an area increases linearly from 1.0×10^7 to 7.0×10^7 during the time steps 6–10 while the traffic in other areas does not change. The traffic pattern is shown in Figure 4. In this situation, an area becomes congested without the control of the upper layer.

Similarly, we also generate a traffic pattern such that traffic between a certain pair of areas increases from 1.0×10^7 to 3.0×10^7 during the time steps 6–10 while other traffic does not change. In this situation, the links between the areas become congested.

c) Routing Calculation: The routes in MP-TE are determined as a solution of the optimization problem (19)–(23). In a similar way to [11], the optimization problem is equally transformed as a convex quadratic programming problem which can be solved by common solvers. We use the CPLEX [20], which is an optimization problem solver. We run CPLEX on computers equipped with four Intel Xeon Processors, each having 10 cores and 30 MB of cache memory.

d) Comparison Method: For comparison, we use a basic hierarchical TE method without the MPC concept described in Section V-A, which we call *simple TE*. To avoid routing oscillations, a long control interval is set at the upper layer, and the averaged observation value is used to decide the routes. Comparing with this method, we verify the effect of MPC: that each controller handles interactions between layers by predicting the behavior of other controllers and avoiding drastic route changes.

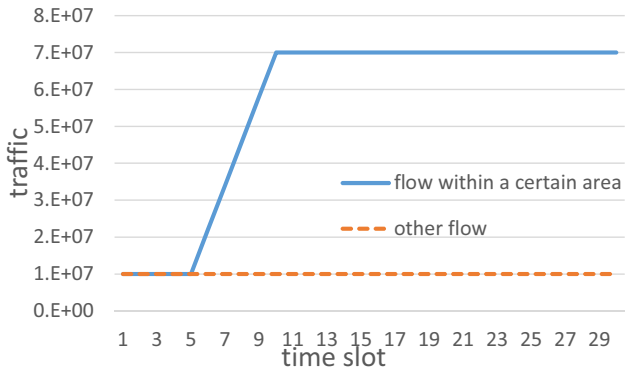


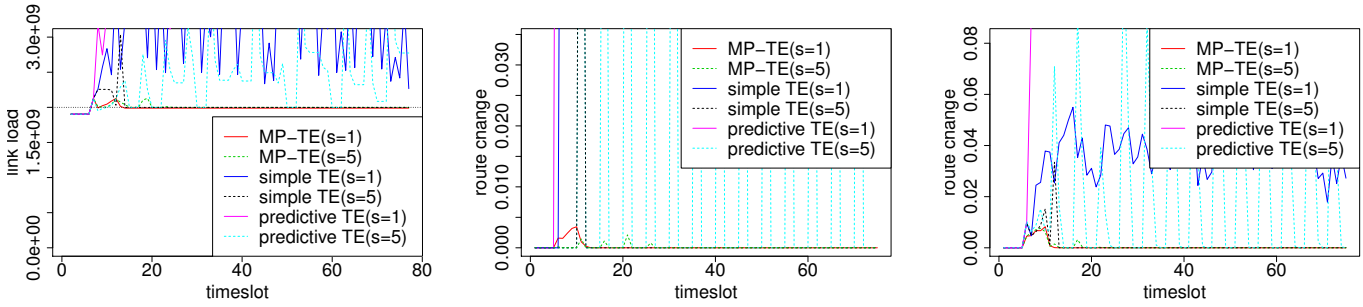
Fig. 4. Time series of traffic causing inner-area congestion.

e) Metrics: We use the maximum link load $\max_l y_l^{m;a}(t)$ as the metric to evaluate hierarchical TE. If the maximum link load is lower than the targeted capacity, the calculated routes accommodate all traffic under the targeted capacity. On the other hand, $|\Delta R_{i;j}^{m;a}(t)|$ is used to check whether or not the routing has converged.

2) Results:

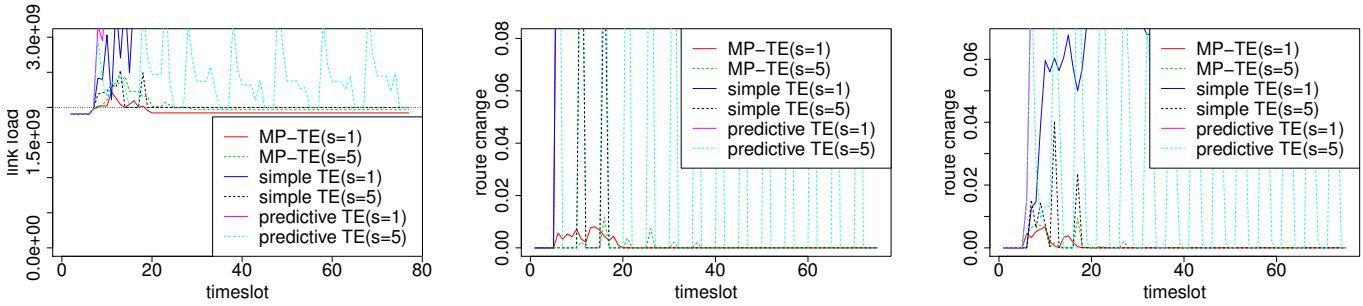
Figure 5 shows results with increasing inner-area traffic for the cases of MP-TE and simple TE. In the figure, "MP-TE" indicates the result of MP-TE with parameters $(h = 3, w = 0.6)$, "simple TE" means the result of simple TE, and "predictive TE" denotes the result of simple TE using the predicted value instead of the observed value. Predictive TE is also a special case of MP-TE with parameters $(h = 1, w = 0)$ in which the controller determines the routes with predicted information for the next time step without restricting route changes. In each case for the TE methods, we show the time series for maximum link load $\max_l y_l^{m;a}(t)$ and average values for the route change $|\Delta R^{m;a}(t)|$ in the upper and lower layers. The horizontal dotted line in the figure denotes the targeted capacity. In the figure, s represents the control interval at the upper layer. In MP-TE, we also change the control interval at the upper layer in a similar way to the simple TE method. Although we show only the result of MP-TE with parameters $(h = 3, w = 0.6)$ here, a detailed discussion of parameter setting for MP-TE is given in Section VI-B. Additionally, Figure 6 shows results with increasing inter-area traffic.

In both cases of increasing inner- and inter-area traffic, MP-TE $(h = 3, w = 0.6)$ quickly achieves route convergence and traffic accommodation with $s = 1$ while simple TE with $s = 1$ consistently causes congestion. In simple TE with $s = 1$, the controllers at both upper and lower layers set inappropriate routes because a route change at a layer causes unexpected changes of the network state at other layers. Repeating the wrong route changes, simple TE with $s = 1$ causes route oscillation. On the other hand, each control server in MP-TE avoids significant route changes at each time step, absorbing the impact from route changes at other layers and avoiding route change impacts on other layers. Thus, route oscillation is avoided without setting a longer control interval at the upper



(a) Maximum link load. (b) Route changes in the upper layer. (c) Route changes in the lower layer.

Fig. 5. Time series for maximum link load and average route changes with increasing inner-area traffic.



(a) Maximum link load. (b) Route changes in the upper layer. (c) Route changes in the lower layer.

Fig. 6. Time series for maximum link load and average route changes with increasing inter-area traffic.

layer.

Routing convergence is also achieved by simple TE with $s = 5$. By setting a long control interval at the upper layer, the control server at the lower layer temporarily completes the route changes while operations at the upper layer are unchanged. Thus, route oscillation is avoided by setting a long control interval at the upper layer.

However, the amount of traffic exceeding the targeted capacity in simple TE ($s = 5$) is larger than that of MP-TE, especially before a route change is conducted in the upper layer. This is because the routes change in simple TE ($s = 5$) delays the changing traffic for two reasons: control delay at the upper layer and observed information delay at the lower layer. Since congestion cannot be solved only by inner-area routing in this simulation, congestion continues until the routes change at the upper layer. On the other hand, in MP-TE ($s = 1$), the controller at the upper layer gradually changes routes from early time steps to reduce area congestion. In addition, simple TE delays changing the routes even in the lower layer because the controller calculates routes based on the observed value at the previous time step. Thus, the amount of excess traffic in simple TE ($s = 5$) is even larger than that of MP-TE($s = 5$) when the traffic is first increasing where both methods do not change routes at the upper layer.

Although using the prediction value is effective for following traffic changes, simple predictive TE does not achieve routing convergence even when setting a long control interval at the upper layer. This is because the impact of the prediction error becomes large when the route changes are not restricted. When the controller of the upper layer overestimates the capacity of the area a and moves traffic from area b to area a, congestion occurs in area a. By observing changes in virtual link capacities, the controller of the upper layer predicts that the area a will be badly congested in the future even if the current congestion is small. Then, the control server at the upper layer moves traffic from area a to area b based largely on the underestimation of the capacity of area a. Repeating the above process, route oscillation occurs. Since the prediction is conducted at each control interval, the impact of the prediction error cannot be mitigated by setting a long control interval. Thus, setting a long control interval at the upper layer is not always effective in prediction-based control, and introducing a restriction of route changes is required to avoid oscillation in prediction-based control.

B. Evaluation in a Dynamic Traffic Case

In the above evaluation, we investigated the behavior of MP-TE with only stationary traffic. Such traffic can be predicted accurately even with a simple prediction method, although prediction error certainly occurs owing to control interactions between layers. In an actual network, traffic changes dynamically with a certain tendency and noisy fluctuation. In this situation, the predicted traffic always includes prediction errors owing to unpredictable fluctuations, and such inaccurate predictions may impact the performance of hierarchical MP-TE. Therefore, we verify the impact of unpredictable traffic changes on MP-TE by simulation. In addition, we investigate appropriate parameter values in MP-TE considering the role of each layer in hierarchical TE.

TABLE I
σ VALUES AND STANDARD DEVIATIONS OF PREDICTION ERROR.

σ	Standard deviation of prediction error
0	3.4×10^5
3.9×10^5	9.9×10^5
7.8×10^5	1.9×10^6
1.2×10^6	2.7×10^6
1.6×10^6	3.6×10^6
1.9×10^6	4.5×10^6
2.3×10^6	5.3×10^6

1) Simulation Environment: The simulation environment is almost the same as that mentioned in subsection VI-A1, the main difference being the traffic pattern. In this simulation, we generate traffic which includes cyclic variation and noisy variation as [21]. The traffic rate from node i to node j at time step k is given as

$$x_{i,j}(k) = \mu \left(1 + \sin\left(\frac{2\pi}{T}k + \theta_{i,j}\right)\right) + W(k) \quad (24)$$

where μ is the mean value of traffic, T is the cycle length of cyclic variation, $\theta_{i,j}$ is the phase, and $W(k)$ is the noisy fluctuation, which follows the zero-mean Gaussian distribution $N(0, \sigma^2)$. We set $\mu = 7 \times 10^6, T = 24$ and randomly change $\theta_{i,j}$ such that the maximum difference $|\theta_{i,j} - \theta_{i',j'}|$ is 3 time steps.

In the simulation we change σ values from 0 to 2.3×10^6 in order to verify the impact of unpredictable traffic on MP-TE. Table I lists the σ values we use and the standard deviation of one-step-ahead prediction error caused when applying the simple prediction method to the generated traffic. As expected, the prediction error also becomes large when the noisy fluctuation becomes large. When $\sigma = 2.3 \times 10^6$, the standard deviation of prediction error is 5.3×10^6, which is about 76 % of average traffic. Since the actual error of one-step-ahead prediction is about 30% [22], our simulation covers the case where prediction error is much larger than the error actually expected.

2) Results: Figures 7–10 show the results of MP-TE. Figs. 7 and 8 show the cases of $\sigma = 0$ and $\sigma = 2.3 \times 10^6$, respectively, with various weights of route changes. In these figures, we change the value of w at lower and upper layers separately. We denote the value of w at the lower layer as w_l and that of the upper layer as w_u.

Similarly, Figs. 9 and 10 show the cases of $\sigma = 0, 2.3 \times 10^6$ with various lengths of predictive horizon. In these figures, we change the value of h at the lower layer (denoted as h_l) and the upper layer (denoted as h_u) separately.

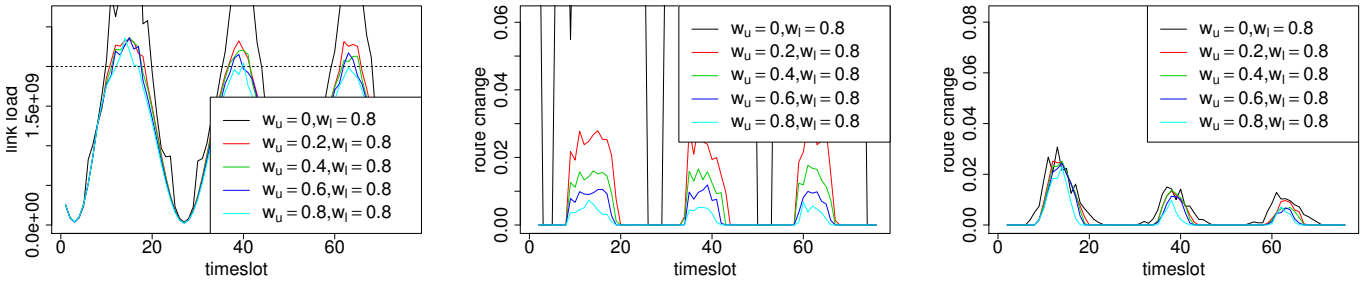
In addition, we show the result of simple TE in Figure 11 setting various control intervals at the upper layer. The figure shows that the maximum link load largely exceeds the targeted capacity around the peak time of all three cycles for any s. The reason for this is different for small and large s. When s is small, the interaction between the layers cannot be avoided since the length of the interval is not sufficient to complete the route change at the lower layer. Therefore, the interaction between layers causes routing oscillation and disturbs the controller in setting appropriate routes. When s is large, a route change at the upper layer simply delays the dynamically changing traffic. Moreover, the control server cannot grasp the congestion situation correctly since the averaged traffic rates and virtual link capacities become inappropriate when s becomes large. Thus, simple TE causes large excess traffic for any s.

On the other hand, MP-TE keeps excess traffic to nearly zero with appropriate setting of parameters in Figs 7–10. As mentioned in VI-A2, MP-TE can follow traffic changes quickly with prediction and setting a small control interval while routing oscillation is avoided by restricting route changes. Thus, MP-TE quickly sets better routes by responding to the dynamically changing traffic. That is, MP-TE outperforms the existing hierarchical TE approach, especially with dynamically changing traffic. The rest of this subsection discusses the impact of prediction error in MP-TE and how to determine appropriate parameter values for MP-TE.

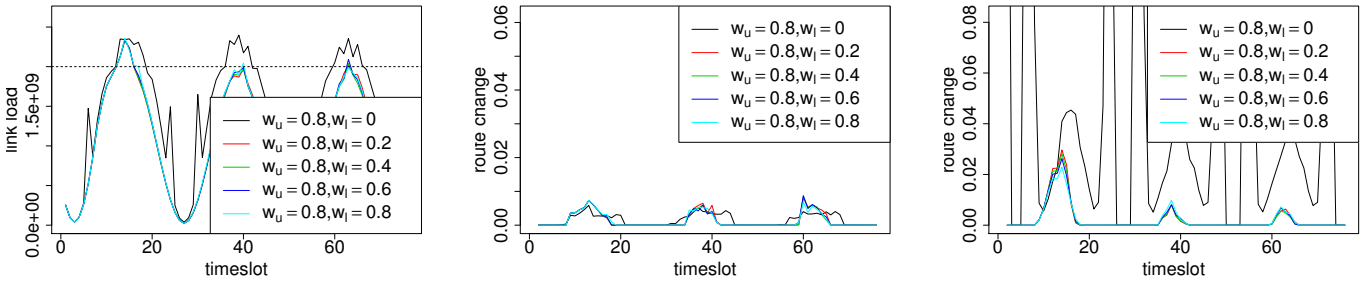
a) Impact of Unpredictable Traffic Fluctuation: First, we discuss the impact of unpredictable traffic fluctuation. Comparing the cases of $\sigma = 0$ (Figs. 7 and 9) and $\sigma = 2.3 \times 10^6$ (Figs. 8 and 10), we cannot see a clear difference in the behavior of MP-TE. This means that unpredictable traffic fluctuation does not significantly affect the performance of MP-TE. This is because the control server avoids setting routes that are highly unsuitable even when significant prediction error occurs, since the control server restricts route changes. Moreover, prediction errors in the link loads, which are more critical for the route calculation than the flow traffic rates, are relatively small owing to the *statistical multiplexing effect*. Since noisy fluctuations and prediction error in the generated traffic are independent between the flows, the increasing and decreasing noises cancel each other. Thus, the control server calculates routes with relatively small prediction error even with large fluctuations in the flows. As mentioned before, the prediction error when $\sigma = 2.3 \times 10^6$ is much larger than realistic prediction error values, and the statistical multiplexing effect is common in realistic networks [23]; hence, MP-TE should work well even in actual situations. Although we only show the cases of $\sigma = 0, 2.3 \times 10^6$, we conducted the simulation with other σ listed in Table I and did not observe a clear difference among these cases.

b) Setting Appropriate Parameters: In this subsection, we discuss parameter setting in MP-TE. First, we investigate the appropriate value of w in hierarchical control. Figs. 7 and 8 show that significant congestion occurs when either w_u or w_l are 0. This is because the control server significantly changes the routes when $w = 0$ and causes control interference between layers, disturbing appropriate routes. Thus, the idea of MPC, which avoids significant changes, is necessary for both layers to avoid the interference of other layers.

Moreover, Figs. 7 and 8 show that traffic exceeding the targeted capacity is reduced by setting a large w_u whereas there is no certain difference among $w_l > 0$. This is because route changes in the upper layers have wider impact than those in the lower layers. When a route change occurs in an upper layer, the control at all areas of the lower layers is affected by changes in the traffic pattern. On the other hand, a route change in the lower layer only affects the residual capacity on the virtual link in the upper layer. Thus, the upper layer should avoid large route changes by setting large w_u whereas the performance of the lower layers is not very sensitive to

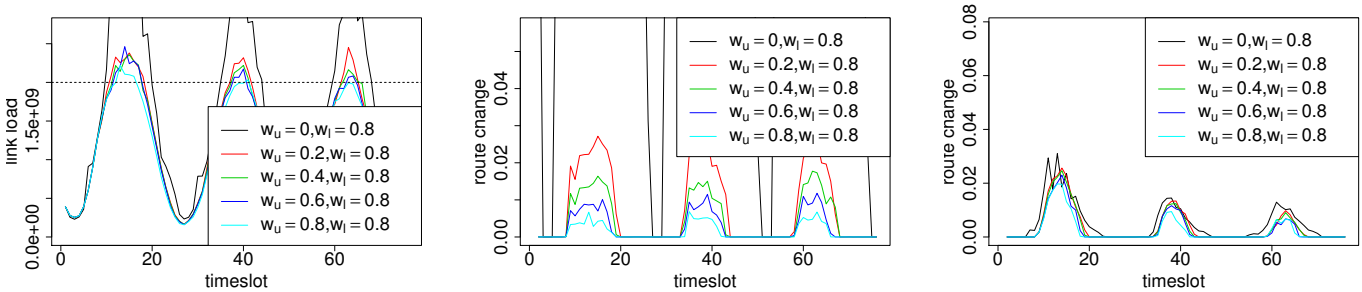


(a) Maximum link load with changing w_u. (b) Route changes in the upper layer with changing w_u. (c) Route changes in the lower layer with changing w_u.

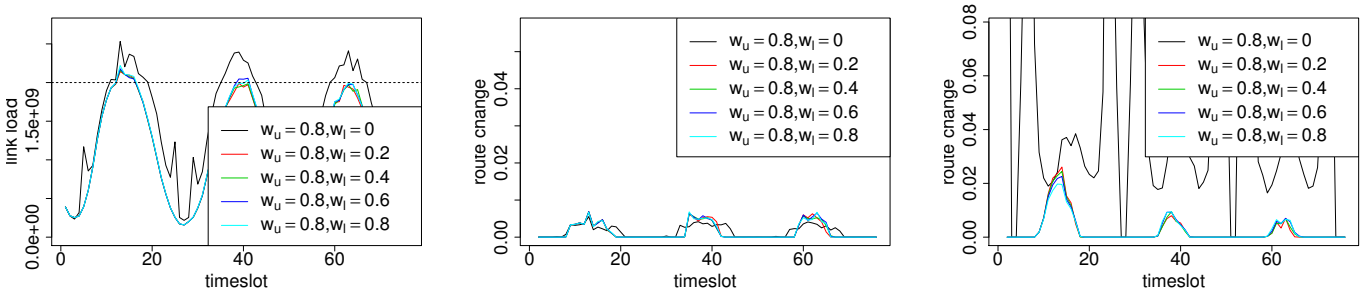


(d) Maximum link load with changing w_l. (e) Route changes in the upper layer with changing w_l. (f) Route changes in the lower layer with changing w_l.

Fig. 7. Time series of maximum link load and average route changes of MP-TE with various weights of route changes ($h = 3, \sigma = 0$).

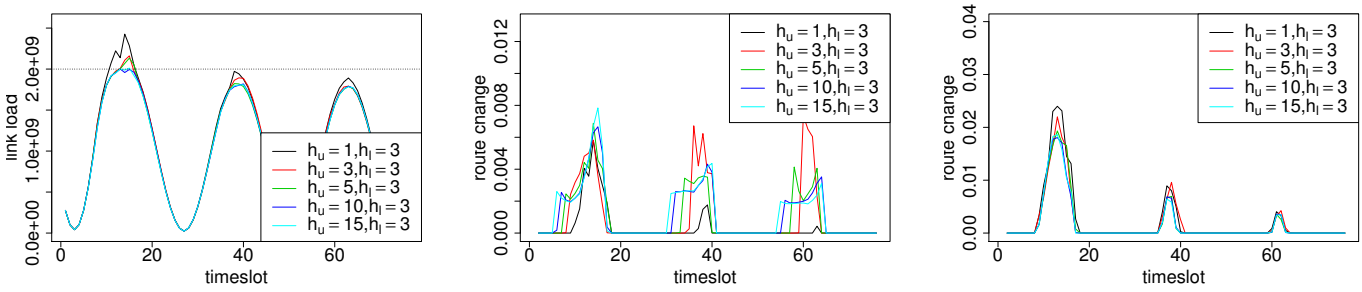


(a) Maximum link load with changing w_u. (b) Routes changes in the upper layer with changing w_u. (c) Route changes in the lower layer with changing w_u.

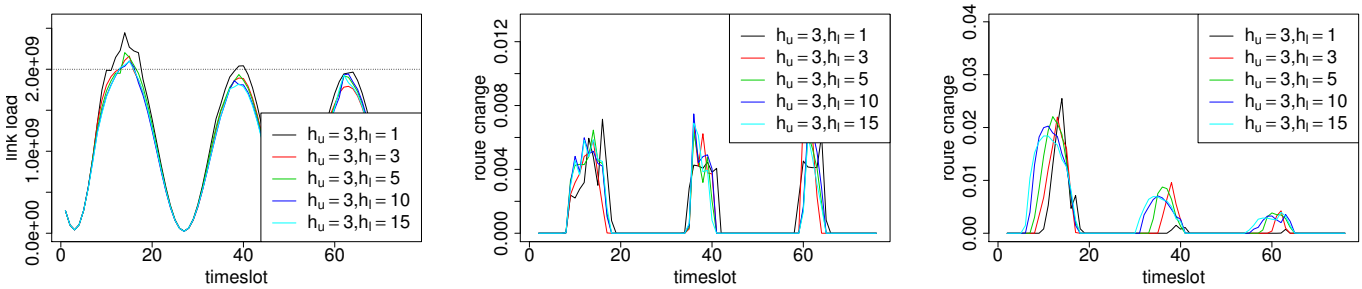


(d) Maximum link load with changing w_l. (e) Route changes in the upper layer with changing w_l. (f) Route changes in the lower layer with changing w_l.

Fig. 8. Time series of maximum link load and average route changes of MP-TE with various weights of routes changes ($h = 3, \sigma = 2.3 \times 10^6$)

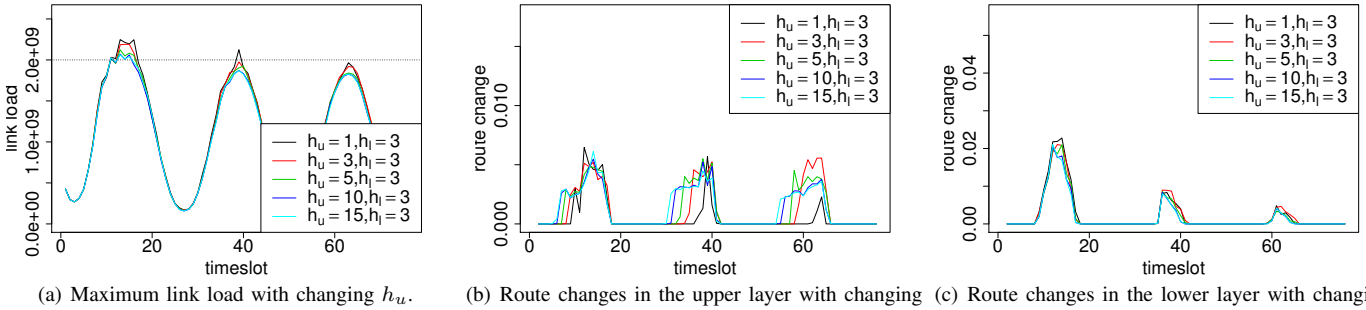


(a) Maximum link load with changing h_u. (b) Route changes in the upper layer with changing h_u. (c) Route changes in the lower layer with changing h_u.

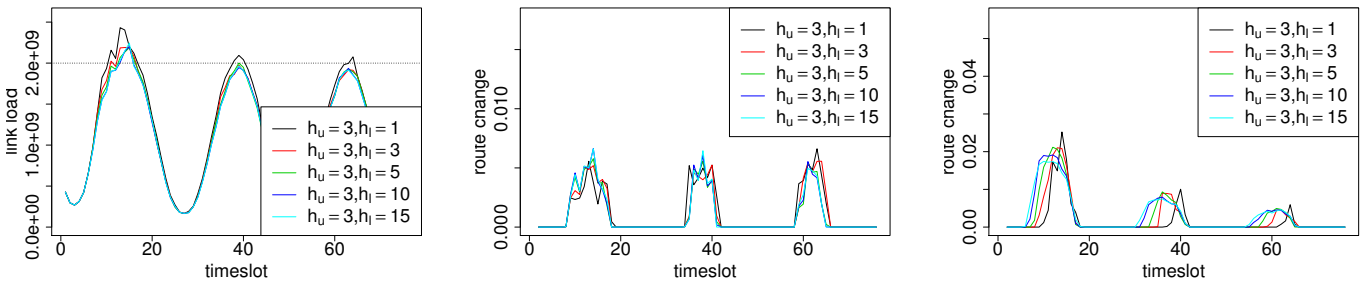


(d) Maximum link load with changing h_l. (e) Route changes in the upper layer with changing h_l. (f) Route changes in the lower layer with changing h_l.

Fig. 9. Time series of maximum link load and average route changes of MP-TE with various lengths of prediction ($w = 0.8, \sigma = 0$).

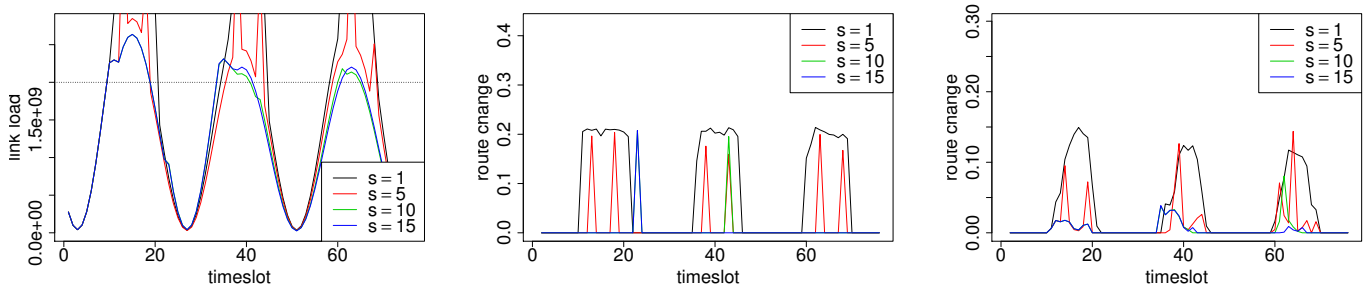


(a) Maximum link load with changing h_u. (b) Route changes in the upper layer with changing h_u. (c) Route changes in the lower layer with changing h_u.



(d) Maximum link load with changing h_l. (e) Route changes in the upper layer with changing h_l. (f) Route changes in the lower layer with changing h_l.

Fig. 10. Time series of maximum link load and average route changes of MP-TE with various lengths of prediction ($w = 0.8, \sigma = 2.3 \times 10^6$).



(a) Maximum link load. (b) Route changes in the upper layer. (c) Route changes in the lower layer.

Fig. 11. Time series of maximum link load and average route changes of simple TE ($\sigma = 0$).

$w_l > 0$.

Finally, we investigate the appropriate value of h in hierarchical control. Figs. 9 and 10 show that worse congestion occurs when either h_u or h_l are 1. This is because the control server with $h = 1$ suddenly changes the routes just before congestion occurs, and other control servers scarcely cooperate with such sudden route changes. The sudden route change causes unexpected changes in the information observed by other control servers. Then the control servers wrongly set routes with incorrect information, causing significant congestion.

On the other hand, the control server with $h > 1$ gradually changes routes in advance of the occurrence of congestion. When a control server gradually changes routes, other control servers can predict how the traffic rates and residual link capacities will change in response to future route changes. Thus, MP-TE with $h > 1$ achieves better collaboration between the layers and keeps the congestion small.

Moreover, Figs. 9 and 10 show that setting large h_u is more effective in reducing excess traffic whereas there is no certain difference among h_l, similar to w_u and w_l. This is because significant route changes should be avoided in the upper layer. Setting a long predictive horizon enables the controller to change routes more smoothly since the controller can begin the route change earlier, before congestion actually occurs. Thus, setting large h_u reduces interference between controllers and results in a quick shift to the appropriate network state. On the other hand, route changes at the lower layer have a small impact on the network state.

VII. Conclusion

Setting a long control interval at the upper layer is a common approach for avoiding oscillations in hierarchical network control. However, doing so requires a long time to respond to environmental changes which cannot be solved by only operations in the lower layers. To solve this problem, we have proposed introducing the idea of MPC into hierarchical network control. Utilizing the basic concept of hierarchical TE, we have developed an MPC-based hierarchical network control called hierarchical MP-TE which achieves routing convergence while setting a short control period. In hierarchical MP-TE, a network is divided hierarchically into multiple areas, and multiple controllers are deployed to calculate routes in a similar way to other hierarchical TE methods. To avoid route oscillation, in hierarchical MP-TE each controller gradually changes routes based on predicted traffic instead of setting a long control interval. Through simulation, we demonstrated that hierarchical MP-TE achieves routing convergence by restricting route changes even when setting a short control interval. We also showed that setting a short control interval improves the convergence time of hierarchical routing. In addition, considering a realistic situation, we evaluated MP-TE under large prediction error and verified that MP-TE is not sensitive to prediction errors. Moreover, we clarified the appropriate parameter values to be set in MP-TE.

Future work will include a method to determine appropriate partitioning of a given network. Furthermore, we will develop a more sophisticated prediction method suitable to MP-TE.

References

[1] X. Li, P. Djukie, and H. Zhang, "Zoning for hierarchical network optimization in software defined networks," in *Proceedings of IEEE Network Operations and Management Symposium 2014*, May 2014, pp. 1–8.

[2] Y. Honma, M. Aida, and H. Shimonishi, "New routing methodology focusing on the hierarchical structure of control time scale," *WSEAS Transactions on Communications*, vol. 13, pp. 505–512, 2014.

[3] Y. Ohsita, T. Miyamura, S. Arakawa, S. Kamamura, D. Shimazaki, K. Shiomoto, A. Hiramatsu, and M. Murata, "Aggregation of traffic information for hierarchical routing reconfiguration," *Computer Networks*, vol. 76, pp. 242–258, Jan. 2015.

[4] S. K. Singh, M. P. Singh, and D. K. Singh, "A survey of energy-efficient hierarchical cluster-based routing in wireless sensor networks," *International Journal of Advanced Networking and Applications*, vol. 2, no. 2, pp. 570–580, 2010.

[5] K.-S. Lui, K. Nahrstedt, and S. Chen, "Routing with topology aggregation in delay-bandwidth sensitive networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 17–29, Feb. 2004.

[6] F. Hao and E. W. Zegura, "On scalable QoS routing: performance evaluation of topology aggregation," in *Proceedings of IEEE INFOCOM 2000*, Mar. 2000, pp. 147–156.

[7] B.-J. Chang and R.-H. Hwang, "Distributed cost-based update policies for QoS routing on hierarchical networks," *Information Sciences*, vol. 159, no. 1–2, pp. 87–108, Jan. 2004.

[8] M. Chamania, X. Chen, A. jukan, F. Rembach, and M. Hoffmann, "An adaptive inter-domain PCE framework to improve resource utilization and reduce inter-domain signaling," *Optical Switching and Networking*, vol. 6, no. 4, pp. 259–267, Dec. 2009.

[9] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control Engineering Practice*, vol. 11, no. 7, pp. 733–764, Jul. 2003.

[10] M. N. Zeilinger, D. M. Raimondo, A. Domahidi, M. Morari, and C. N. Jones, "On real-time robust model predictive control," *Automatica*, vol. 50, no. 3, p. 683?694, Mar. 2014.

[11] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, N. Kamiyama, K. Ishibashi, K. Shiomoto, and T. Hashimoto, "Traffic engineering based on model predictive control," *IEICE Transactions on Communications*, vol. E09-B, no. 6, pp. 996–1007, Jun. 2015.

[12] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, K. Ishibashi, K. Shiomoto, and T. Hashimoto, "Traffic engineering based on stochastic model predictive control for uncertain traffic change," in *Proceedings of The Seventh IFIP/IEEE International Workshop on Management of the Future Internet*, Ottawa, May 2015, pp. 1165–1170.

[13] S. Uludag, K. shan Lui, K. Nahrstedt, and G. Brewster, "Analysis of topology aggregation techniques for QoS routing," *ACM Computing Surveys*, vol. 39, no. 3, Sep. 2007.

[14] M. F. Zhani, H. Elbiaze, and F. Kamoun, "Analysis and prediction of real network traffic," *Journal of Networks*, vol. 4, no. 9, pp. 855–865, Nov. 2009.

[15] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, "Long-term forecasting of Internet backbone traffic: Observations and initial models," in *Proceedings of IEEE INFOCOM 2003*, vol. 2, Mar. 2003, pp. 1178–1188.

[16] B. Krithikaivasan, T. Zenf, K. Deka, and D. Medhi, "ARCH-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic," *IEEE/ACM Transactions on Networking*, vol. 15, no. 3, pp. 683–696, Jun. 2007.

[17] B. Zhou, D. He, Z. Sun, and W. H. Ng, "Network traffic modeling and prediction with ARIMA/GARCH," in *Proceedings of the Fourth International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks*, Sep. 2006, pp. 1–10.

[18] M. L. F. Miguel, M. C. Penna, J. C. Nievola, and M. E. Pellenz, "New models for long-term Internet traffic forecasting using artificial neural networks and flow based information," in *Proceedings of IEEE Network Operations and Management Symposium 2012*, Apr. 2012, pp. 1082–1088.

[19] C. Guang, G. Jian, and D. Wei, "Nonlinear-periodical network traffic behavioral forecast based on seasonal neural network model," in *Proceedings of the Second International Conference on Communications, Circuits and Systems*, vol. 1, Jun. 2004, pp. 683–687.

[20] "IBM ILOG CPLEX Optimizer," optimization software : http://www-01.ibm.com/software/integration/optimization/cplex-optimizer.

[21] A. Nucci, A. Sridharan, and N. Taft, "The problem of synthetically generating IP traffic matrices: initial recommendations," *ACM SIGCOMM Computer Communication Review*, vol. 35, no. 3, pp. 19–32, Jul. 2005.

[22] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, K. Ishibashi, and K. Shiomoto, "Traffic prediction for dyanamic traffic engineering considering traffic variation," in *Proceedings of IEEE GLOBECOM 2013*, Dec. 2013, pp. 1592–1598.

[23] M. Johnston, H.-W. Lee, and E. Modiano, "Robust network design for stochastic traffic demands," *Journal of Lightwave Technology*, vol. 31, no. 18, p. 3104?3116, Sep. 2013.

Tatsuya Otoshi received an M.E. degree in information science and technology in 2014 from Osaka University, where he is currently a postgraduate studying for a Ph.D. degree. His research interests include traffic engineering and traffic prediction. He is a student member of the IEEE.

Yuichi Ohsita received M.E. and Ph.D. degrees in information science and technology in 2005 and 2008 from Osaka University, where he is currently an assistant professor in the Graduate School of Information Science and Technology. His research interests include traffic matrix estimation and countermeasures against DDoS attacks. He is a member of IEICE, IEEE, and the Association for Computing Machinery (ACM).

Masayuki Murata received M.E. and Ph.D. degrees in information science and technology from Osaka University in 1984 and 1988. In April 1984, he joined the Tokyo Research Laboratory at IBM Japan 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. From 1992 to 1999, he was an associate professor with the Graduate School of Engineering Science, Osaka University, and since April 1999, he has been a professor. He moved to the Graduate School of Information Science and Technology, Osaka University, in April 2004. He has published more than 300 papers in international and domestic journals and conferences. His research interests include computer communication networks, performance modeling, and evaluation. He is a fellow of IEICE and a member of IEEE, the Association for Computing Machinery (ACM), The Internet Society, and IPSJ.

Yousuke Takahashi received an M.E. degree in information science and technology from Osaka University in 2009. He is currently a member of the Traffic Engineering Group, Communication Traffic & Service Quality Project at NTT Network Technology Laboratories. His research interests include network economy and traffic analysis. He is a member of IEICE.

Keisuke Ishibashi received an M.S. degree in mathematics from Tohoku University in 1995 and a Ph.D. degree in information science and technology from the University of Tokyo in 2005. He is currently a senior research engineer in the Traffic Engineering Group, Communication Traffic & Service Quality Project at NTT Network Technology Laboratories. His research interests include Internet measurement and traffic analysis. He is a member of IEICE, IEEE, and the Operation Research Society of Japan.

Kohei Shiomoto received M.E and Ph.D. degrees in information and computer science from Osaka University in 1989 and 1998. He is currently a senior manager in Communication Traffic & Service Quality Technology at NTT Network Technology Laboratories. His research interests include traffic engineering, routing, network algorithms, optimization, and protocols. He is a fellow of IEICE and a member of IEEE and the Association for Computing Machinery (ACM).

Tomoaki Hashimoto received B.E, M.E, and Ph.D degrees from the Tokyo Metropolitan Institute of Technology in 2003, 2004, and 2007, respectively, all in aerospace engineering. He was a research assistant at the RIKEN Brain Science Institute from 2007 to 2008, an assistant professor at Shinshu University from 2008 to 2009 and an assistant professor at Osaka University from 2009 to 2013. He is currently a lecturer at the Osaka Institute of Technology. His research interests are in the area of dynamical and control systems.