PAPER

Traffic Engineering Based on Model Predictive Control

Tatsuya OTOSHI[†a)], *Nonmember*, Yuichi OHSITA[†b)], *Member*, Masayuki MURATA[†c)], *Fellow*, Yousuke TAKAHASHI[††d)], Noriaki KAMIYAMA[††e)], Keisuke ISHIBASHI[††f)], *Members*, Kohei SHIOMOTO[††g)], *Fellow*, *and* Tomoaki HASHIMOTO[†††h)], *Nonmember*

SUMMARY In recent years, the time variation of Internet traffic has increased due to the growth of streaming and cloud services. Backbone networks must accommodate such traffic without congestion. Traffic engineering with traffic prediction is one approach to stably accommodating time-varying traffic. In this approach, routes are calculated from predicted traffic to avoid congestion, but predictions may include errors that cause congestion. We propose prediction-based traffic engineering that is robust against prediction errors. To achieve robust control, our method uses model predictive control, a process control method based on prediction of system dynamics. Routes are calculated so that future congestion is avoided without sudden route changes. We apply calculated routes for the next time slot, and observe traffic. Using the newly observed traffic, we again predict traffic and re-calculate the routes. Repeating these steps mitigates the impact of prediction errors, because traffic predictions are corrected in each time slot. Through simulations using backbone network traffic traces, we demonstrate that our method can avoid the congestion that the other methods cannot.

key words: *Model Predictive Control, Traffic Engineering, Traffic Prediction, Multi-path Routing*

1. Introduction

In recent years, the time variation of Internet traffic has increased due to the growth of streaming and cloud services. Backbone networks must accommodate such traffic without congestion.

Until now, backbone networks have addressed this problem by reserving redundant link capacity to accommodate not only average traffic but also traffic surges [1,2]. However, this approach incurs higher costs as the average and variance of traffic increases. Moreover, this approach wastes energy due to the poor utility of network resources; this approach reserves more than double the capacity required to accommodate the actual traffic. Hence, a method for accommodating network traffic without congestion and with limited resources is required in order to reduce costs and power consumption caused by over-provisioning.

Routing optimization such as load balancing by splitting traffic among paths is effective for accommodating traffic with limited resources. A routing method called *oblivious routing* [3–5] tries to accommodate traffic demands without prior knowledge of traffic statistics by using fixed routes that are calculated in advance. In this method, the route is calculated so as to minimize a metric called *oblivious ratio* which is the worst ratio of maximum link load to its optimal value. Such worst-case-guarantee routes, however, degrade performance under normal situations. In [5], numerical evaluations show that oblivious routing will spend most of its time in a congested state when the number of source–destination pairs is large, despite achieving a low oblivious ratio.

Many dynamic traffic engineering (TE) methods have addressed the problem of accommodating time-varying traffic by using limited resources effectively [6–9]. In dynamic TE methods, a control server periodically observes network traffic and dynamically reroutes flow to accommodate the observed traffic. These methods set routes for only observed traffic, however. This renders the configured routes unsuitable after significant traffic changes because routes are not changed until the next control cycle. Control servers can quickly respond to traffic changes by shortening the control cycle interval, but frequent route changes cause routing oscillations that degrade TCP session throughput; oscillations cause packet reordering by delivering the packets of a given TCP session via different paths, which reduces the TCP session window size. Routing oscillations also cause overly frequent changes in round-trip time (RTT), which decreases the throughput of delay-based TCP [10]. Hence, a method that avoids congestion without significant route changes is required.

TE with traffic prediction is one approach to solving such problems. In this method, routes are calcu-

[†]The authors are with the Graduate School of Information Science and Technology, Osaka University, Suita, Osaka, 560–0871, Japan

[††]The authors are with the NTT Network Technology Laboratories, NTT Corporation, Musashino, Tokyo, 180–8585, Japan

[†††]The author is with the Graduate School of Engineering Science, Osaka University, Toyonaka, 560–8631, Japan

a) E-mail: t-otoshi@ist.osaka-u.ac.jp
b) E-mail: y-ohsita@ist.osaka-u.ac.jp
c) E-mail: murata@ist.osaka-u.ac.jp
d) E-mail: takahashi.yousuke@lab.ntt.co.jp
e) E-mail: kamiyama.noriaki@lab.ntt.co.jp
f) E-mail: ishibashi.keisuke@lab.ntt.co.jp
g) E-mail: shiomoto.kohei@lab.ntt.co.jp
h) E-mail: thashi@sys.es.osaka-u.ac.jp

DOI: 10.1587/transcom.E0.B.1

lated on the basis of predicted future traffic. Prediction methods for network traffic have been studied for various time scales, with variation ranging from milliseconds or seconds [11–14] up to daily [15, 16] and even monthly or yearly long-term variation [17, 18]. Traffic prediction that considers both daily and short-term variation has also been proposed for TE [19]. However, no prediction methods are without error, and routes calculated from incorrect traffic information become inappropriate for actual traffic and may cause congestion. Therefore, TE with traffic prediction should be robust to prediction errors.

In this paper, we propose a TE method that uses traffic prediction in a way that is not impacted by prediction errors. Our method uses model predictive control (MPC) [20, 21], which has been recently studied as a method of system control based on the prediction of system dynamics. In MPC, a controller inputs system parameters so as to maintain system output at close to a target value. The MPC controller predicts the system output, which reflects changes in the input values, and calculates the optimal input values for future time slots. Input values are implemented for only the next time slot. The MPC controller then observes the output and corrects the predictions by using the output value as feedback. After correction of the predictions, the MPC controller recalculates the input value for the next time slot with the corrected predictions. By repeatedly performing the above steps, the MPC controller can calculate accurate input for future time slots even when prediction errors occur. Moreover, the MPC controller avoids overreaction to temporary prediction errors by avoiding drastic changes in the input value. In this paper, we apply MPC to TE to propose a method that follows predicted traffic variation and is robust against prediction errors.

We summarize the contribution of this paper as follows. (i) This paper proposes a new prediction-based TE method, which is robust to the prediction errors by applying the idea of MPC. (ii) This paper demonstrates the advantage of our TE method by simulation using the actual traffic trace. (iii) This paper discusses the suitable parameter setting of our TE method.

The rest of this paper is organized as follows. Section 2 describes TE and TE with traffic prediction. Section 3 describes our TE method, to which we apply MPC. Section 4 presents an evaluation of basic behavior in our TE method. Section 5 gives an evaluation of our TE method as applied to an actual backbone network. Section 6 surveys related work. Section 7 presents our conclusions.

2. Traffic Engineering and Traffic Prediction

2.1 Traffic Engineering

TE has been studied as an approach to accommodating changing traffic by dynamically changing routes. The process of TE is composed of the following three steps: (1) traffic rates at network devices are observed, (2) routes are calculated so as to accommodate the current traffic, and (3) the calculated routes are applied to the actual network. These steps are periodically repeated to follow traffic changes. The details of the above steps are discussed below.

Traffic rates are observed at a fixed interval (e.g., one second, one minute, or one hour), with the times between observations called *time slots.* Because there are a huge number of traffic flows, aggregate traffic rates are observed instead of individual rates. In [6,8], multiple flows are aggregated as an origin–destination (OD) flow that traverses from the ingress point-of-presence (PoP) router to the egress PoP router. Similar to these work, we too aggregate individual flows as OD flows. Hereafter, we denote the traffic rate of OD flow i at the kth time slot by $x_i(k)$, and the vector $\boldsymbol{x}(k) = {}^t(x_1(k), \cdots, x_q(k))$ represents the traffic rates of all OD flows at the kth time slot, where q is the number of OD flows. The traffic rates of the OD flows are monitored by routers or traffic monitors attached to the routers. This information can be collected by using the Netflow protocol or similar.

After the traffic information is collected, routes are calculated on the basis of the observed traffic rates. The routes are defined by the fraction of traffic of each OD flow sent to each path. We denote the fractions by a matrix $R(k)$ whose (i, j) element $R_{i,j}(k)$ indicates the fraction of traffic on the OD flow j that traverses the available path i. Under the assumption that the traffic pattern will not change between the current and next time slots, the expected traffic rates on links are calculated as

$$\hat{\boldsymbol{y}}(t+1) = G \cdot R(t+1) \cdot \boldsymbol{x}(t) \tag{1}$$

where $\hat{\boldsymbol{y}}(t+1) = {}^t(\hat{y}_1(t+1), \ldots, \hat{y}_l(t+1))$ is a vector whose component $\hat{y}_i(t+1)$ indicates the expected traffic rate of link i at the next time slot, l is the number of links, and G is a matrix whose (i, j) element $G_{i,j}$ is 1 if the available path j traverses the link i and 0 otherwise. TE is the process of calculating routes $R(t+1)$ so as to minimize a *cost function* $f(\hat{\boldsymbol{y}}(t+1))$, such as link load, delay, or packet loss rate for traffic rates on the links. The TE is formalized as the following optimization problem:

$$\textit{minimize} \,:\, f(\hat{\boldsymbol{y}}(t+1)) \tag{2}$$
$$\textit{subject to} \,:\, \hat{\boldsymbol{y}}(t+1) = G \cdot R(t+1) \cdot \boldsymbol{x}(t). \tag{3}$$

The most used cost function is maximum link utilization [6,8] for accommodating unexpected traffic surges.

Finally, the calculated routes are implemented. One approach to implementing the routes is to set the MPLS label-switched paths (LSPs) between the OD pair along the calculated routes [7, 22, 23]. In this approach, a control server calculates the set of links used by each LSP and the split ratio of OD flow among LSPs from $R(t+1)$. Then, the calculated routes are implemented by establishing the LSPs.

In the existing TE method, the control server calculates the next routes $R(t+1)$ from the latest observed traffic rates $\boldsymbol{x}(t)$. These routes $R(t+1)$, however, are not exactly suited to the traffic rates at time slot $t+1$, because the actual rates will differ from those of time slot t. Under drastically changing traffic, the difference between $\boldsymbol{x}(t+1)$ and $\boldsymbol{x}(t)$ becomes large and routes calculated from $\boldsymbol{x}(t)$ may no longer accommodate the actual traffic at the $(t+1)$th time slot. Frequent control with narrow time slots is one way to quickly respond to such traffic fluctuations. In such methods, routes are frequently calculated to respond to traffic changes. However, frequent and significant route changes degrade the throughput of TCP sessions because of the induced packet reordering or frequent changes in RTT. To solve these problems, the TE and traffic prediction must cooperate. By using predicted traffic, the TE method directly sets routes fitting the traffic at future time slots.

2.2 TE with Traffic Prediction

Traffic prediction is useful for TE to prevent route change delays due to differences between actual and observed traffic. Fig. 1 shows an overview of TE with traffic prediction. Unlike existing observation-based TE methods, observed traffic rates are not directly used to calculate routes. Rather, observed traffic is used to calculate future traffic rates by the traffic prediction process, and routes are then calculated from prediction results. This process is periodically repeated to follow traffic trends. The details are shown below.

The traffic prediction is estimation of future traffic rates of OD flows. First, a model of traffic dynamics is constructed from the observed traffic rates. The model represents a time evolution such as $x(k+1) = F(x(1), \cdots, x(k))$ where F is a model of traffic dynamics. Future traffic rates are then predicted in accordance with the model. If we observe the traffic rate until time slot t, the traffic rate at time slot $t+1$ is calculated as

$$\hat{x}(t+1) = F(x(1), \cdots, x(t)), \tag{4}$$

where $\hat{x}(t+1)$ is the predicted traffic at time slot $t+1$. The traffic rates from time slot $t+2$ are iteratively calculated, by using the previous predicted values instead of observation values, as $\hat{x}(t+k) = F(x(1), \cdots, x(t), \hat{x}(t+1), \cdots, \hat{x}(t+k-1))$.

Using the predicted traffic on the OD flows, traffic rates on the links can also be predicted; the predicted traffic rates on links in the case of routes $R(t+1)$ are calculated as

$$\hat{\boldsymbol{y}}(t+1) = R(t+1)\hat{\boldsymbol{x}}(t+1). \tag{5}$$

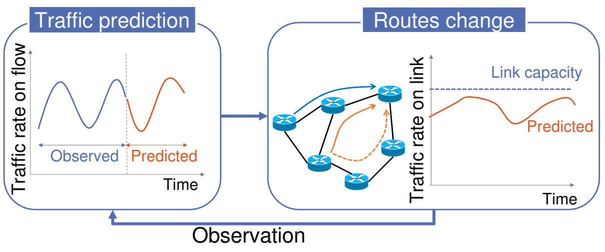


Fig. 1 Overview of traffic engineering with traffic prediction

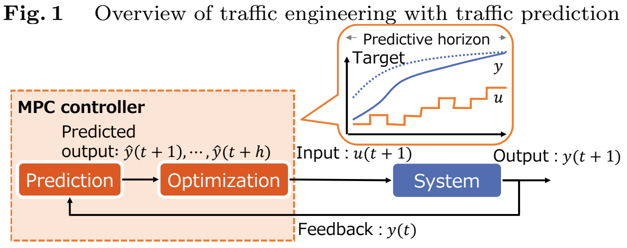


Fig. 2 Overview of MPC

In TE with traffic prediction, the routes are calculated by considering the cost function of $\hat{\boldsymbol{y}}(t+1)$.

TE with traffic prediction configures routes so as to avoid future congestion without frequent route changes. One approach is to configure the fixed routes R that minimize a cost function at future time slots from $t+1$ to $t+h$. The optimal fixed routes R are obtained by solving the following optimization problem:

$$\textit{minimize} \ : \ f(\hat{\boldsymbol{y}}(t+1), \cdots, \hat{\boldsymbol{y}}(t+h)) \tag{6}$$

$$\textit{subject to} \ : \ \hat{\boldsymbol{y}}(k) = R\hat{\boldsymbol{x}}(k), k = t+1, \cdots, t+h. \tag{7}$$

The predicted traffic, however, includes the prediction error. Thus, the TE method must configure the appropriate routes even when the prediction errors occur.

3. Traffic Engineering Based on MPC

3.1 MPC

MPC is a method of system control based on predictions of system dynamics; this has been studied in recent years. Fig. 2 shows an overview of MPC. In MPC, a controller sets an input parameter so as to keep the output performance of the system close to an operator-specified target. Unlike traditional system control, the MPC controller predicts changes in the output value to calculate the inputs for the *predictive horizon*, time slots $[t+1, t+h]$ where h is the distance to the predictive horizon. We denote the input and output at the kth time slot by $u(k)$ and $y(k)$, respectively. The MPC controller calculates the inputs for the predictive horizon $[t+1, t+h]$ so as to keep $y(k)$ close to the target value $r_y(k)$. The inputs $u(t+1), \cdots, u(t+h)$ that keep $y(k)$ close to $r_y(k)$ are obtained by using the objective function $J_1 = \sum_{k=t+1}^{t+h} \|y(k) - r_y(k)\|^2$, where $\|\cdot\|$ represents the Euclidean norm:

$$(u(t+1), \cdots, u(t+h)) = \underset{(u(t+1),\cdots,u(t+h))}{\arg\min} J_1. \tag{8}$$

To solve the above optimization problem, the future outputs $y(t+1), \cdots, y(t+h)$ must be predicted from the inputs $u(t+1), \cdots, u(t+h)$. The future output under the 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

$$z(k+1) = \phi(k, z(k), u(k)) \tag{9}$$
$$y(k) = \psi(k, z(k), u(k)), \tag{10}$$

where $z(k)$ is the state of the system at the kth time slot, and ϕ and 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 or ψ functions that do not well represent actual system dynamics. Predictions of system output will be inaccurate under an incorrect model, and prediction errors become increasingly large with more distant predictive horizons. The MPC controller therefore implements only the first of the calculated inputs $u(t+1), \cdots, u(t+h)$ for the predictive horizon. 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.

Prediction errors may also significantly change input values, destabilizing the system. The 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 slot k by $\Delta u(k) = u(k) - u(k-1)$, and the aggregated amount of change during the predictive horizon by $J_2 = \sum_{k=t+1}^{t+h} \|\Delta u(k)\|$. Instead of the input values determined by Eq. (8), the controller calculates the input values by the following optimization problem:

$$(u(t+1), \cdots, u(t+h)) = \underset{(u(t+1), \cdots, u(t+h))}{\arg\min} \ (1-w)J_1 + wJ_2 \tag{11}$$

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

3.2 Applying MPC to TE

We apply MPC to TE to achieve a prediction-based TE that is robust against prediction errors. Fig. 3 shows an overview of our TE method, to which MPC is applied. We assume that a control server collects all traffic information and sets the routes. In the TE, a central control server acts as the MPC controller, which inputs the routes $R(k)$ and measures the outputs of the network and the traffic rates on the links $\boldsymbol{y}(k)$. The control server periodically changes the routes by repeating the following two steps: 1) The control server predicts the traffic rates of OD flows for the target time slots from the previously observed traffic rates using a certain prediction model. 2) The control server calculates the routes from the prediction so as to minimize a cost function $f(\hat{\boldsymbol{y}}(k))$, such as link load, delay, or packet loss rate.

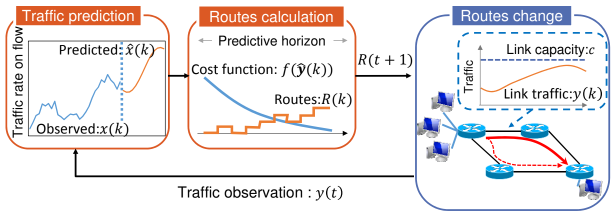


Fig. 3 Overview of traffic engineering based on MPC

3.2.1 Traffic Prediction

The control server predicts future traffic from the previous observations in accordance with a prediction model. The predicted traffic is used as an input of the route calculation. In our TE, any prediction method can be used. Though a prediction method may have an impact on prediction errors, the suitable prediction method is out of the scope of this paper. Instead, we use one of the simplest prediction models in our evaluation to demonstrate that our TE works properly even in the case of inaccurate prediction.

3.2.2 Routes Calculation

The control server computes the routes by minimizing the objective function $J_1 = \sum_{k=t+1}^{t+h} f(\hat{\boldsymbol{y}}(k))$, which indicates the summation of the cost function during the predictive horizon. In addition, the control server also minimizes $J_2 = \sum_{k=t+1}^{t+h} \|\Delta R(k)\|^2$ where $\Delta R(k)$ is a matrix whose (i,j) element $\Delta R_{i,j}(k) = R_{i,j}(k) - R_{i,j}(k-1)$. By minimizing J_2, the overreaction to prediction error is avoided. This multi-objective optimization is conducted by minimizing the weighted sum $(1-w)J_1 + wJ_2$, where $0 \leq w \leq 1$ weights the importance of the restriction on the route changes.

In our TE method, the control server solves the following optimization problem at each time slot t:

$$\textit{minimize}: \sum_{k=t+1}^{t+h} \left((1-w)f(\hat{\boldsymbol{y}}(k)) + w\|\Delta R(k)\|^2\right) \tag{12}$$

$$\textit{subject to}: \forall k, \hat{\boldsymbol{y}}(k) = G \cdot R(k) \cdot \hat{\boldsymbol{x}}(k) \tag{13}$$
$$\forall k, \forall i, \forall j, R_{i,j}(k) \in [0,1] \tag{14}$$
$$\forall k, \forall j, \sum_{i \in \wp(j)} R_{i,j}(k) = 1. \tag{15}$$

Here, $\hat{\boldsymbol{x}}(k), G$ are given variables and $R(k), \hat{\boldsymbol{y}}(k)$ are the variables to be optimized. Eq. (13) represents the relation between the traffic rates of the OD flows and links. Eqs. (14) and (15) mean that all traffic on each OD flow is allocated to an available path.

Although all of the routes $R(t+1), \cdots, R(t+h)$

during the predictive horizon are obtained by solving the above optimization problem, the control server implements only the next routes $R(t+1)$. After the route change, the control server corrects the traffic prediction $\hat{\boldsymbol{x}}(k)$ by using the newly observed traffic rate and recalculates the next routes by solving the optimization problem again.

4. Evaluation of Basic Behavior of MPC-based TE

In this section, we investigate the behavior of the MPC-based TE in a basic situation. In this evaluation, we generate the average traffic rate of each time slot of each OD flow. At the beginning of each time slot, we calculate the routes of the OD flows by TE methods using the traffic rates of the past time slots. Then, we map the OD flows on the links according to the calculated routes. Finally, we evaluate the performance of the TE based on the average traffic rate on each link. In this evaluation, we do not assume a specific time scale of the time slot, but the length of the time slot is sufficiently large so that the route change does not affect the traffic rate.

4.1 Simulation Environment

4.1.1 Network Topology

We use the simple network topology shown in Fig. 4. Each link has a capacity of 100 units of traffic and delay of 0.1 unit time. In this simple network there are only two OD flows, from node 0 to node 1 and from node 4 to node 5. Each OD flow has two available paths, shown by the arrows in Fig. 4, the paths 0–1 and 0–2–3–1 for the OD flow between node 0 and node 1 and the paths 4–5 and 4–2–3–5 for the other OD flow. Due to the overlap between paths 0–2–3–1 and 4–2–3–5 (on link 2–3), the control server has to adjust the split ratio of traffic among the paths. For example, if the traffic rates increase at the OD flow 0–1, more traffic should be bypassed on the path 0–2–3–1, and traffic at OD flow 4–5 should not traverse the path 4–2–3–5 so much to avoid the congestion.

4.1.2 Network Traffic

We use the artificial traffic shown in Fig. 5. This artificial traffic includes traffic increases and decreases, which will cause congestion unless the routes are appropriately changed.

4.1.3 Prediction Method

In this evaluation, we use a simple prediction method detailed as follows. First, we find a best-fit straight line $l(k) = ak + b$ that minimizes the sum of squared

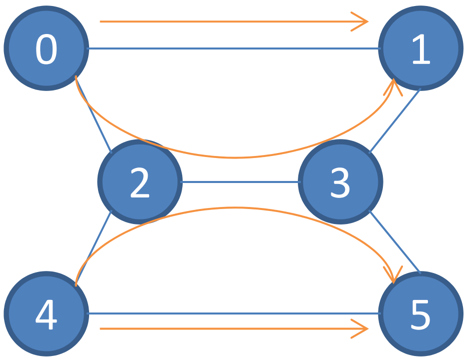


Fig. 4 Simple network topology

t

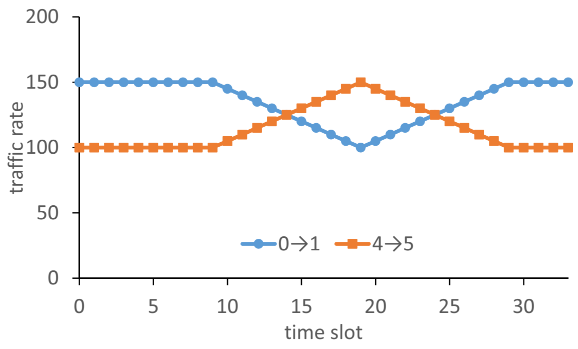


Fig. 5 Network traffic for simple network topology

distances from the previous observed traffic rates $x(t-s), x(t-s+1), \cdots, x(t) (s \geq 1)$, denoted as $\sum_{k=0}^{s}(x(t-s+k) - l(t-s+k))^2$. We then obtain the future traffic rate as $\hat{x}(t+k) = l(t+k)$. Though there are many more sophisticated prediction methods, we use the above simple prediction with $s = 1$ to verify the effect of correcting the prediction with feedback from new observations, which is one of the main effects of MPC.

4.1.4 Cost Function

In this evaluation, we use a cost function that is based on link utilization, which is similar to existing work [24]. While most previous studies have minimized link utilization, high link utilization does not affect communication performance unless congestion occurs. We therefore use a cost function that indicates whether congestion occurs. In this cost function, we define the congestion level $\zeta_j(k) \geq 0$ for each path j at time slot k. To distinguish whether congestion occurs or not, we introduce a threshold value called *target capacity* $c_i = \rho C_i$ where ρ is an allowable upper limit of link utilization which is defined by the performance requirements such as delay or loss rate. In this evaluation, we set the value of ρ to 0.9. If traffic on any links over path j does not exceed the target capacity, then j is regarded as an uncongested path. If there are the links with traffic exceeding the target capacity, then j is regarded as a congested path. The congestion level of a path is determined under the following policies: (1) the congestion on a link equally affects paths that traverse the link, and (2) the congestion level on a path is determined by the bottleneck link, defined as the most congested intermediate link on the path. Based on these polices, we set $\zeta_j(k)$ as

$$\zeta_j(k) = \max_{i \in \mathscr{P}(j)} [y_i(k) - c_i]^+ / n_i, \tag{16}$$

where $[x]^+$ equals x if x is positive and equals 0 otherwise, n_i is the number of paths which traverse the link i, and $\mathscr{P}(j)$ is the set of all links the path j traverse. In the following evaluation, we use the congestion level normalized by scaling the value of $\zeta_j(k)$ with the maximum link capacity

$$\zeta'_j(k) = \zeta_j(k) / \max_i c_i \tag{17}$$

instead of $\zeta_j(k)$. If $\zeta_j(k)$ is 0 for any path j, the TE succeeds in accommodating traffic with satisfying performance requirements. Therefore, we use the sum of $\zeta_j(k)$ as the metric to evaluate the TE methods.

4.1.5 Route Calculation

Though the congestion level defined by Eq. (17) is nonlinear, it can be rewritten as a linear constraint in the optimization problem. The calculation of $[\hat{y}_l(k) - c_l]^+$ can be replaced by a linear constraint $[\hat{y}_l(k) - c_l]^+ = \hat{y}_l(k) - c_l + S_l(k)$, where $S_l(k) \geq 0$ is a slack variable. The operation $max_{l \in \mathscr{P}(p)}$ is translated by inequality constraints $n_l \zeta_p(k) \geq \max_{l \in \mathscr{P}(p)} [\hat{y}_l(k) - c_l]^+ / \max c_l$ for all links l in the path p. As a result, the MPC-based TE using the congestion level as a cost function is rewritten as the following convex quadratic programming problem:

$$minimize : \sum_{k=t+1}^{t+h} \left((1-w) \|\boldsymbol{\zeta}'(k)\| + w \|\Delta R(k)\| \right) \tag{18}$$

$$subject\ to : \forall k, \forall p, \forall l \in \mathscr{P}(p), n_l \zeta'_p(k) \geq \frac{\alpha_l(k)}{\max c_l} \tag{19}$$

$$\forall k, \forall l, \alpha_l(k) = \hat{y}_l(k) - c_l(k) + S_l(k) \tag{20}$$

$$\forall k, \forall l, \alpha_l(k) \geq 0 \tag{21}$$

$$\forall k, \forall l, S_l(k) \geq 0 \tag{22}$$

$$\forall k, \hat{\boldsymbol{y}}(k) = G \cdot R(k) \cdot \hat{\boldsymbol{x}}(k) \tag{23}$$

$$\forall k, \forall i, \forall j, R_{i,j}(k) \in [0, 1] \tag{24}$$

$$\forall k, \forall j, \sum_{i \in \wp(j)} R_{i,j}(k) = 1. \tag{25}$$

Here, $\alpha_l(k) \geq 0$ represents the value of $[\hat{y}_l(k) - c_l]^+$. The solution of this optimization problem satisfies the original congestion level because the variables satisfy the inequality formulation $n_l \zeta'_p(k) \geq \max_{l \in \mathscr{P}(p)} \frac{\alpha_l(k)}{\max c_l} \geq \max_{l \in \mathscr{P}(p)} \frac{[\hat{y}_l(k) - c_l]^+}{\max c_l}$, and the equality is attained when $\zeta'_p(k)$ is minimized.

To solve the optimization problem of Eqs. (18)–(25), we use the CPLEX [25] package, 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.

4.1.6 Compared Method

(1) Observation-based TE

We use an observation-based TE to compare with our MPC-based TE. In the observation-based TE, the control server uses only the observed traffic rates instead of the predicted rates. Comparing the MPC-based TE with this observation-based TE demonstrates the effect of considering future traffic variation.

(2) Simple Prediction-based TE

We also use a simple prediction-based TE in our comparison. In this method, the controller simply calculates the routes without restricting the routes changes. For the prediction, the controller uses the same prediction model for MPC-based TE. This TE method is a special case for our method when parameters are set to $h = 1$ and $w = 0$. Comparison with this method demonstrates the effect of restricting route change to avoid the impact of prediction errors.

4.2 Congestion Level

Fig. 6 shows the sum of $\zeta_j(k)$ for all paths, which is the amount of traffic exceeding the target link capacity at each time slot. The labels “MPC”, “prediction base”, and “observation base” represent the results of the MPC-based TE, simple prediction-based TE, and observation-based TE, respectively. We compares two cases of MPC-based TE with $h = 1$ and $h = 3$ to verify the effect of considering the future traffic variation.

Fig. 6 indicates the advantages of MPC-based TE. In Fig 6, congestion occurs at some time slots for all TE methods except MPC($h = 3, w = 0.5$). The observation-based TE cannot avoid the congestion because the routes based on the previous traffic rates are no longer suited to the next time slot. Thus, the traffic prediction is required to avoid congestion. However, the prediction based TE also cannot avoid the congestion due to the prediction errors. In our evaluation, the prediction error occurs at time slots 10, 20, and 30, because the slope of traffic rate changes at those time slots. Due to these prediction errors, the prediction-based TE poorly configures the routes: the capacity of the shared link 2–3 is under-allocated for the under-predicted flow, while it is over-allocated for the over-predicted flow. As a result, the actual traffic on the under-predicted flow overshoots the target capacity and causes congestion. The prediction error, however, is corrected after observation of the implemented routes at time slots 10, 20,

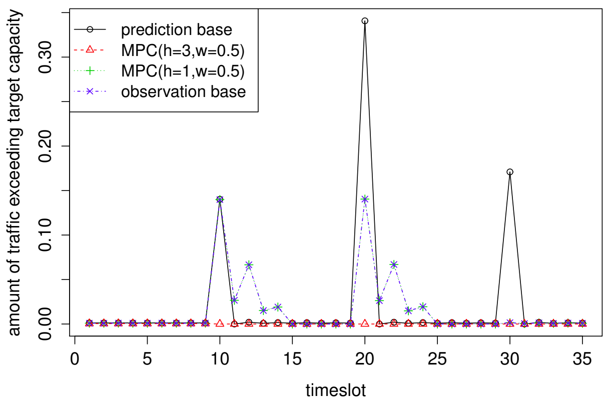


Fig. 6 Traffic exceeding target link capacity in a simple network

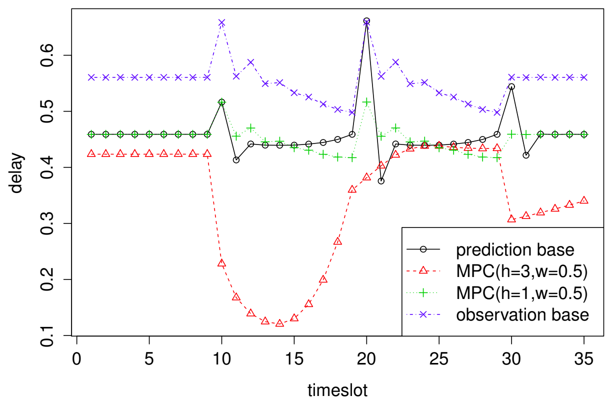


Fig. 7 Average end-to-end delay of all OD flows in a simple network

and 30. Therefore, the routes are corrected with exact predictions after these time slots.

Restricting the route changes avoids the overreaction to prediction errors. However, restricting the route changes may prevent the required route changes. As a result, MPC($h = 1, w = 0.5$) cannot avoid the congestion. This problem can be solved by starting route changes in advance. MPC($h = 3, w = 0.5$) starts to change the routes when the future congestion is predicted. Thus, MPC($h = 3, w = 0.5$) configures the routes so as to follow the traffic changes without changing the routes significantly at any time slot. As a result, MPC($h = 3, w = 0.5$) avoids congestion at all time slots.

The above results indicate that the idea of MPC that controls input on the basis of predictions and mitigates the influence of prediction errors is effective for TE. MPC-based TE avoids future congestion, while the simple prediction- or observation-based TE cannot avoid congestion induced by prediction errors or traffic changes.

4.3 End-to-end Delay

In the previous subsection, we demonstrated that MPC-based TE keeps the congestion level close to 0. In this subsection, we demonstrate the impact of keeping the congestion level close to 0.

One of the important impacts is the end-to-end delay; keeping the congestion level to 0 keeps the queuing delay of links small. Therefore, we compare the end-to-end delay in this subsection.

We calculate the link delay from link utilization by approximating packet processing on the Internet by the M/M/1 queuing model. According to queuing theory, link delay is calculated as $\frac{\bar{L}}{C_l - y_l} + p_l$, where $\bar{L}$ is the average packet length, p_l is the propagation delay, and C_l is the actual capacity of the link l. The delay of OD flow j at time slot k is the weighted sum of the delays of all available paths $\sum_{i \in \wp(j)} R_{i,j}(k) d_j$, where d_i is the delay of the path i as the sum of delays on all links over the path. Large delays are caused not only by congestion, but also by path length. Therefore, if most traffic traverses a long path, the delay of OD flow becomes large even under low congestion.

Figs. 7 and 8 show the average delay and maximum delay of all OD flows, respectively. These figures show that MPC($h = 3, w = 0.5$) avoids the large delay at all time slots. This is because MPC($h = 3, w = 0.5$) keeps the congestion level low.

In Figs. 7 and 8, MPC($h = 3, w = 0.5$) decreases the delay significantly from time slot 10 to 19. This is because MPC($h = 3, w = 0.5$) selects the shorter paths without congestion.

This significant change of the end-to-end delay does not degrade the TCP throughput, because the length of the time slot can be set larger than the length of TCP flows; frequent route change is not required since MPC-based TE avoids congestion at all time slots. In the evaluation using the actual traffic traces described in Section 5, the length of the time slot is set to 1 or 10 seconds, while most of observed TCP sessions ends withing 1 second.

In actual situation, the M/M/1 model is too simple to model the packet processing. However, the rational characteristic that a delay monotonically increases as a link load increases does not change. Thus, the MPC-based TE will suppress the queuing delay similarly even in actual situation if the target capacity is set by using realistic delay model.

5. Evaluation in an Actual Network

From the above simulation results, we show that MPC-based TE can reduce the congestion level and end-to-end delay in simple situations where only one link is shared by two OD flows. In actual networks, where multiple OD flows share multiple links, however, the

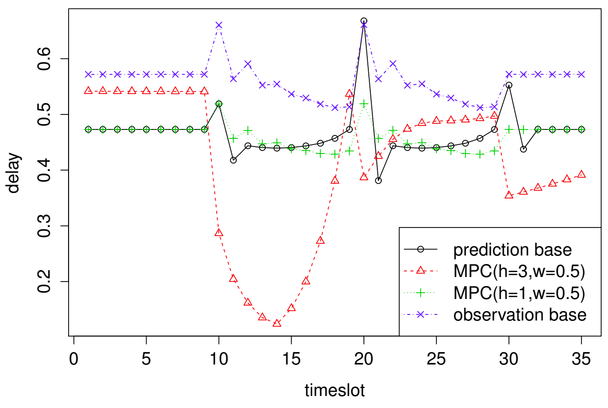


Fig. 8 Maximum end-to-end delay of all OD flows in a simple network

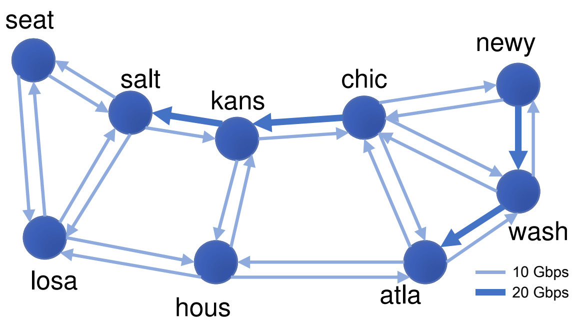


Fig. 9 Internet2 topology

situation is more complex. To demonstrate that MPC-based TE is also effective for actual networks, we evaluate the performance on the Internet2 topology by using actual traffic traces. The evaluation is performed by the similar way to Section 4.

5.1 Simulation Environment

5.1.1 Network Topology

In this subsection, we use an actual Internet2 backbone network, shown in Fig. 9. Each link has a capacity of 10 Gbps except four links (kans $\rightarrow$ salt, chic $\rightarrow$ kans, newy $\rightarrow$ wath, and wash $\rightarrow$ atla), each of which has a capacity of 20Gbps. The link capacities of Internet2 are over-provisioned, so that maximum link utilization is less than 20%. Hence, in our simulation we set the target capacity of the link to 15% of actual capacity.

5.1.2 Network Traffic

Here, we use actual traffic traces [26]. These traffic data are collected by the Netflow protocol at each of the PoP routers. The sampling rate is one out of every 100 packets, and aggregated data are exported every five minutes. The sampling method has two main limitations: it contains sampling errors, and there may be unsampled flows. However, this is not a critical problem

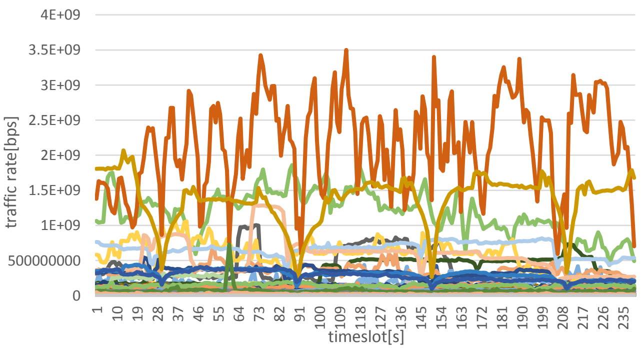


Fig. 10 Internet2 network traffic

for our evaluation because we need only the traffic rate of the aggregated OD flow, which has many samples. We use four minutes of data, avoiding file boundaries by excluding the first and last 30 s of the Netflow data for 12:00 to 12:05 p.m. on 1 November 2011. The traffic data are aggregated into the OD flows between PoP routers by using the BGP information. From the start and end times and the total amount of traffic of each flow in the Netflow data, we obtain the traffic rate every second. The start and end times are recorded with millisecond granularity. When the start and end times of a flow are t_s and t_e, the amount of traffic during a certain period τ is calculated as

$$x = \frac{\theta}{t_e - t_s}\tau \tag{26}$$

by assuming that traffic arrives at a constant bitrate with θ the total amount of flow traffic. The traffic amount at the time slot k corresponding to the actual time interval $[t_k, t_{k+1}]$ depends on the active time of the flow in the time slot, so τ is set to the active time as

$$\tau = \begin{cases} t_{k+1} - t_s & (t_k < t_s \wedge t_{k+1} < t_e) \\ t_e - t_s & (t_k < t_s \wedge t_{k+1} \geq t_e) \\ t_{k+1} - t_k & (t_k > t_s \wedge t_{k+1} < t_e) \\ t_e - t_k & (t_k > t_s \wedge t_{k+1} \geq t_e) \\ 0 & (otherwise). \end{cases} \tag{27}$$

Finally, the traffic rate of an OD flow is obtained by summing the traffic amount for all flows in the OD flow. The calculated traffic rates are shown in Fig. 10.

5.1.3 Prediction Method

We use the same prediction method that was used in Section 4.

5.1.4 Cost Function

We use the same cost function that was used in Section 4.

5.1.5 Calculation of Routes

As in Section 4, we use CPLEX [25] to calculate the routes. In this evaluation, the optimization is finished within one second when $h = 3$.

5.1.6 Compared Method

In addition to the simple prediction-based TE and observation-based TE, we also compare MPC-based TE with the following smoothed observation-based TE. The smoothed observation-based TE calculates the next routes $R(t+1)$ by using the smoothed value $\bar{\boldsymbol{x}}(t)$, which reduces the noise of observation value $\boldsymbol{x}(t)$. We use an exponential moving average for smoothing. If $\bar{x}_i(t-1)$ is a previous smoothed value of the flow i, and we observe a current traffic rate $x_i(t)$, then we update the smoothed value to $\bar{x}_i(t) = \eta x_i(t) + (1-\eta)\bar{x}_i(t-1)$, where η represents the degree of weighting decrease of historical data. By comparing the MPC-based TE with the smoothed observation-based TE, we demonstrate that the advantages of MPC-based TE are not due to smoothing the observed traffic rates, though the traffic prediction obtains the average dynamics of traffic and eliminates short-term variation.

5.2 Simulation Results

Fig. 11 shows the amount of traffic exceeding the target link capacity when MPC-based TE is applied to the Internet2 topology with actual traffic traces. For readability, we only show the results at certain time slots around the time when congestion occurs. The label "with smoothing" represents the results of smoothed observation-based TE.

Similar to Fig. 6, only MPC($h = 3, w = 0.5$) avoids congestion at all time slots. The simple prediction-based TE causes congestion due to prediction errors. MPC($h = 1, w = 0.5$) cannot also avoid congestion because we cannot change the routes sufficiently. On the other hand, MPC($h = 3, w = 0.5$) avoids the congestion; MPC($h = 3, w = 0.5$) avoids the overreaction to prediction errors by avoiding the significant route changes, and follows the traffic changes by changing the routes gradually after future congestion is predicted.

By comparing the results of MPC-based TE with smoothed observation-based TE, we can distinguish the effect of smoothing and prediction. From Fig. 11, the TE using simple smoothing cannot avoid the congestion because the smoothing amplifies the difference between expected and actual traffic rates, which slows the response to the traffic change.

5.3 Discussion on Parameter Setting

The MPC-based TE has some parameters such as weight for route change, length of predictive horizon, and cycle length of control and prediction. We investigate the effect of these parameters in detail using the Internet2 topology with actual traffic trace.

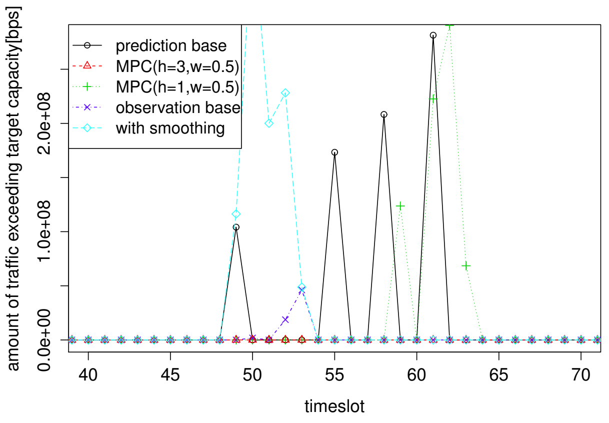


Fig. 11 Traffic exceeding target link capacity with actual Internet2 traces

5.3.1 Weight for Route Change

First, we examine the impact of w, which is the weight of route change. In the above evaluation, we show that our TE method is robust against prediction errors when $w = 0.5$. The value of w, however, represents the sensitivity to not only the prediction error but also the changing traffic. Hence, we may have to consider a trade-off between the robustness and sensitivity to set an appropriate value of w.

Figure 12 shows the maximum amount of traffic exceeding the target link capacity for all time slots when the MPC-based TE is conducted with various values of w. The y-axis is the amount of exceeding traffic, and the x-axis is the value of w. The label h means that the MPC-based TE is conducted with the predictive horizon length of h.

In Fig. 12, the medium value of w such as w=0.1–0.6 is appropriate for avoiding the congestion, which manages to balance the robustness and sensitivity. In addition, the achieved performance of the MPC-based TE is not sensitive to w within the range of w=0.1–0.6.

5.3.2 Length of Predictive Horizon

Second, we investigate the impact of the length of the predictive horizon h. This parameter indicates how long into the future the control server considers calculating the routes. Using the large value of h, the control server can take into account not only the next time slot but also further time slots to change the routes gradually in advance of traffic changes. However, setting too large h may cause wrong route changes because the prediction errors generally become large as the prediction target is far ahead. In addition, the larger h becomes, the longer the calculation of routes takes.

Figure 13 shows the maximum amount of traffic exceeding the target link capacity when the MPC-based

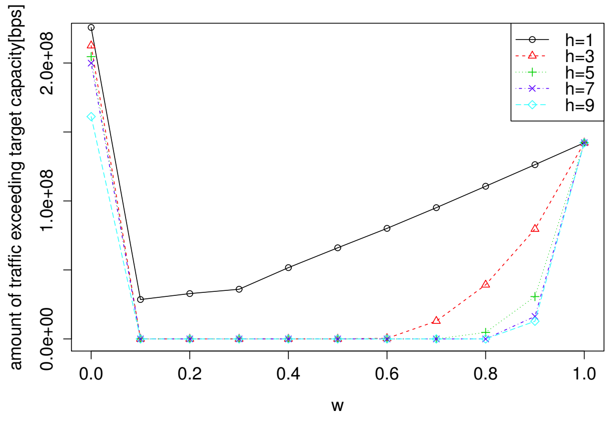


Fig. 12 Maximum amount of traffic exceeding the target link capacity for all time slots when the MPC-based TE is conducted with various values of w

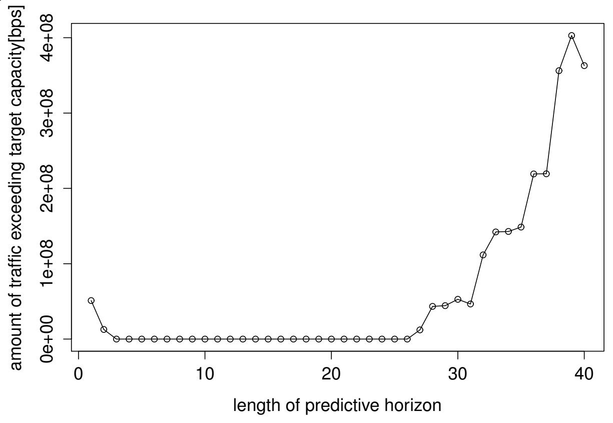


Fig. 13 Maximum amount of traffic exceeding the target link capacity for all time slots when the MPC-based TE is conducted with various values of h ($w = 0.5$)

TE is conducted with various values of h, setting the value of w to 0.5. When h is larger than 27, the congestion level increases as h becomes large. This is because the influence of prediction error becomes large as the predictive horizon becomes long. Too small values of $h = 1, 2$ also cause the congestion because the control server does not consider the traffic change further into the future. The appropriate values of h to avoid the congestion are within the range of 3–26. Hence, it is sufficient for the MPC-based TE to set h to 3 or slightly larger values.

5.3.3 Cycle Length of Control and Prediction

Finally, we discuss the cycle length of control and prediction. In the above simulation, we set the control and prediction cycle length so that they equal the observation cycle length (one second). However, the frequent control increases the loads on the control server. On the other hand, the control server cannot follow the traffic change when the control and prediction cycle is large. Therefore, it is important to clarify which length of cycle is appropriate to avoid the congestion and a large calculation time.

To discuss the impact of cycle lengths of control and prediction, we simply extend the MPC-based TE so as to deal with different cycle lengths of control and prediction. If the prediction cycle is T_p seconds, the control server estimates the future traffic every T_p seconds using the previous average rate in each T_p seconds; that is, we use the average rate $\bar{x}(k) = \frac{1}{T_p}\sum_{i=(k-1)T_p}^{kT_p-1} x(i)$ as input to traffic prediction, and we obtain the future average rates $\hat{\bar{x}}(t+1), \cdots \hat{\bar{x}}(t+h)$. Similarly, if the control cycle is T_c seconds, which is a multiple of T_p, the control server calculates the routes every T_c seconds using the average rate of predicted traffic in each T_c seconds; that is, we use the average predicted value $\hat{x}'(k) = \frac{T_p}{T_c}\sum_{i=(k-1)T_c}^{kT_c-1} \hat{\bar{x}}(i)$ as input to TE in order to calculate the route $R'(t+1)$ during the next T_c seconds. Though the period of control and prediction is changed, the time grain of traffic change is not. That is, traffic rates change every second.

Figure 14 shows the maximum amount of traffic exceeding the target link capacity for all time slots when the MPC-based TE is conducted with various lengths of control and prediction. We set the x-axis to the length of the predictive horizon similar to that in Fig. 13 because the effect of the predictive horizon will change as cycle length changes. The labels T_p and T_c in the figure represent the lengths of the prediction cycle and control cycle, respectively.

From Fig. 14, frequent control and prediction are better for avoiding the congestion. This is simply because the routes are quickly changed corresponding to the traffic change by the frequent control and prediction. However, the control cycle and prediction cycle have different impacts. In Fig. 14, the congestion can be avoided by the frequent prediction ($T_p = 1$) even when the control cycle is 10 seconds. On the other hand, the congestion cannot be avoided when the prediction cycle is 10 seconds. This is because predicting with fine granularity can follow the changing traffic and the control server can accommodate traffic even with fixed routes considering the fluctuation of traffic. Therefore, we can set the length of the control cycle to slightly large while the prediction has to be frequently conducted.

6. Related Work

There is a large body of literature regarding TE. Though we formalized MPC-based TE as a centralized control in which a central control server collects all the information and calculates all the routes for a network, our method is also applicable to distributed schemes. Distributed TE achieves scalability and quick response to the traffic changes using only locally observed traffic information. In TeXCP [6] and MATE [7], each ingress node observes path states such as packet loss rate and delay, and splits the traffic of ingress–egress pairs among the paths on the basis of observed statis-

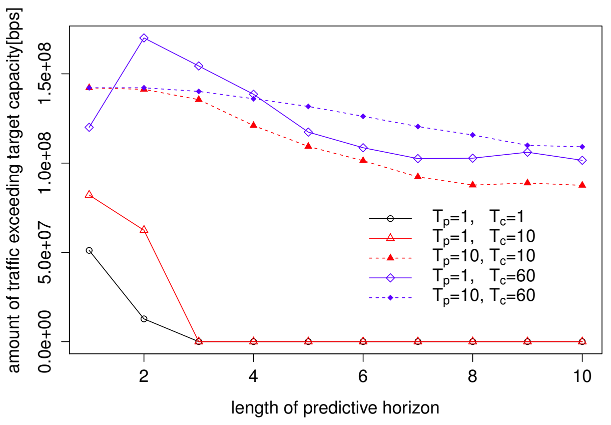


Fig. 14 Maximum amount of traffic exceeding the target link capacity for all time slots when the MPC-based TE is conducted with various lengths of control and prediction ($w = 0.5$)

tics.

In another application of MPC to TE, Rétvári and Németh [27] applied MPC to rate-adaptive multipath routing, in which a central controller adjusts sending rates of each user. Their method preliminarily sets explicit rate control rules corresponding to each set of user demands. The control server then periodically observes user demands, searches the appropriate control rules, and adjusts the sending rate according to the control rules. In setting the rules, they use a traffic model called the *zero-buffer path flow* (ZBPF) model instead of traffic prediction. In the ZBPF model, they assume that no further traffic arrives within the predictive horizon. Hence, their method also works as an observation-based TE.

The predictability of Internet traffic has received significant interest in many domains, such as capacity planning, anomaly detection, admission control, and traffic engineering. Many prediction models have been proposed to predict network traffic, such as ARMA, ARIMA [11,17], ARCH [14], GARCH [12], and neural networks [15,16] . Although we use only a simple prediction method in our evaluation, our TE method can select prediction models according to characteristics of the target variation, such as time granularity and periodicity.

7. Conclusion

We proposed a TE method that uses predicted traffic rates instead of observed values. According to prediction-based control theory, our TE method calculates routes while correcting predictions and avoiding large route changes to mitigate the impact of prediction errors. Through simulation with actual backbone network traffic traces, we demonstrated that our TE method can avoid congestion that observation-based TE cannot. In addition, comparison with MPC-based TE with a smoothing method showed that the advantage of MPC-based TE does not come from the smoothing effect of the traffic prediction. Moreover, we discussed the parameter setting such as the weight for route change w, the length of predictive horizon h, and the cycle length of control and prediction. Then, we clarified that the performance of our method is not sensitive to the parameters w and h in a certain range and that we can select safe values of w and h from the range. Furthermore, we showed that changing routes even in 10-second intervals is sufficient to respond to the change in traffic rates every one second while the prediction has to be conducted in one second.

Future work will include clarification of the robustness of MPC-based TE through theoretical analyses. Additionally, to guarantee its scalability, we will adapt MPC-based TE to distributed control that determines routes using only local traffic information. Through experimental evaluation with MPC-based TE implemented in hardware, we will investigate the effect of interaction with other network controls such as TCP.

Acknowledgement

This work was supported in part by the Strategic Information and Communications R&D Promotion Programme of the Ministry of Internal Affairs and Communications, Japan.

References

[1] C. Graleigh, F. Tobagi, and C. Diot, “Provisioning IP backbone networks to support latency sensitive traffic,” in *Proceedings of IEEE INFOCOM 2003*, Apr. 2003, pp. 375–385.
[2] M. Roughan, “Robust network planning,” *Guide to Reliable Internet Services and Applications Conputer Communications and Networks*, pp. 137–177, 2010.
[3] M. Hajiaghayi, J. H. Kim, T. Leighton, and H. Räcke, “Oblivious routing in directed graphs with random demands,” in *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, May 2005, pp. 193–201.
[4] H. Räcke, “Optimal hierarchical decompositions for congestion minimization in networks,” in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, May 2008, pp. 255–264.
[5] G. Németh, “On a new competitve measure for oblibious routing,” *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 1, pp. 117–123, Feb. 2014.
[6] S. Kandula, D. Katabi, B. Davie, and A. Charny, “Walking the tightrope: responsive yet stable traffic engineering,” in *Proceedings of ACM SIGCOMM 2005*, Aug. 2005, pp. 253–264.
[7] E. Anwar, C. Jin, L. Steven, and W. Indra, “MATE: MPLS adaptive traffic engineering,” in *Proceedings of IEEE INFOCOM 2001*, Apr. 2001, pp. 1300–1309.
[8] H. Wang, H. Xie, L. Qiu, Y. R. Yang, Y. Zhang, and A. Greenberg, “COPE: traffic engineering in dynamic networks,” in *Proceedings of ACM SIGCOMM 2006*, vol. 36, no. 4, Aug. 2006, pp. 99–110.
[9] N. Wang, K. H. Ho, G. Pavlou, and M. Howarth, “An overview of routing optimization for Internet traffic engineering,” *IEEE Communications Survey & Tutorials*, vol. 10, no. 1, pp. 36–56, first quarter 2008.
[10] K. Srijith, L. Jacob, and A. Ananda, “TCP Vegas-A: Im-

proving the performance of TCP Vegas," *Computer Communications*, vol. 28, no. 4, pp. 429–440, Mar. 2005.
[11] 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.
[12] 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.
[13] L. Xiang, "A new hybrid network traffic prediction method," in *Proceedings of IEEE GLOBECOM 2010*, Dec. 2010, pp. 1–5.
[14] 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.
[15] 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.
[16] 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.
[17] 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.
[18] N. K. Groshwitz and G. C. Polyzos, "A time series model of long-term NSFNET backbone traffic," in *Proceedings of IEEE ICC 1994*, May 1994, pp. 1400–1404.
[19] 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.
[20] 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.
[21] 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.
[22] D. O. Awduche, "MPLS and traffic engineering in IP networks," *IEEE Communications Magazine*, vol. 37, pp. 42–47, Dec. 1999.
[23] E.-S. M. El-Alfy, S. N. Mujahid, and S. Z. Selim, "A pareto-based hybrid multiobjective evolutionaty approach for constrained multipath traffic engineering optimization in MPLS/GMPLS networks," *Journal of Network and Computer Applications*, vol. 36, no. 4, pp. 1196–1207, Jul. 2013.
[24] J. Knowles, M. Oates, and D. Corne, "Advanced multi-objective evolutionary algorithms applied to two problems in telecommunications," *BT Technology Journal*, vol. 18, no. 4, pp. 51–65, 2000.
[25] "IBM ILOG CPLEX Optimizer," optimization software : http://www-01.ibm.com/software/integration/optimization/cplex-optimizer.
[26] "Internet2 data," available from http://internet2.edu/observatory/archive/data-collections.html.
[27] G. Rétvári and G. Németh, "On optimal multipath rate-adaptive routing," in *Proceedings of the Fifteenth IEEE Symposium on Computers and Communications*, Jun. 2010, pp. 605–610.

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 traf?c engineering and traffic prediction. He is a student member of 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).

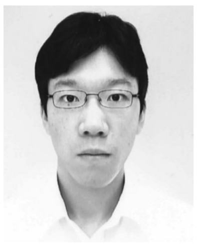
Masayuki Murata received M.E. and Ph.D. degrees in information science and technology from Osaka University in 1984 and 1988. In April 1984, he joined the Tokyo Research Laboratory IBM Japan as a researcher. From September 1987 to January 1989, he was an assistant professor with the Computation Center, 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.

Noriaki Kamiyama received M.E and Ph.D degrees in communications engineering from Osaka University in 1994 and 1996, respectively. From 1996 to 1997, he was with the University of Southern Calfornia as a visiting researcher. He joined NTT Multimedia Networks Laboratories in 1997. Since then, he has been engaged in research concerning pricing methods for IP networks, network topology design, and IP traffic measurement. He is currently in NTT Network Technology Laboratories. He is a member of the IEEE, IEICE, and the Operations Research Society of Japan.

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, 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, 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 Tokyo Metropolitan Institute of Technology in 2003, 2004, and 2007, ewspectively, all in aerospace engineering. He was a Research Assistant at the RIKEN Brain Science Institute from 2007 to 2008, and an Assistant Professor at the Shinshu University. His research interests are in the area of dynamical and control systems.