Prediction Based Traffic Engineering under Uncertainty

Submitted to
Graduate School of Information Science and Technology
Osaka University

January 2017

Tatsuya OTOSHI

List of publication

Journal papers

1. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto, “Traffic Engineering Based on Model Predictive Control,” *IEICE Transactions on Communications*, vol. E98-B, no. 6, pp. 996–1007 June 2015.

2. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, and Kohei Shiomoto, “Traffic Prediction for Dynamic Traffic Engineering,” *Computer Networks*, vol. 85, pp. 36–50, July 2015.

Refereed Conference Papers

1. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, and Kohei Shiomoto, “Traffic Prediction for Dynamic Traffic Engineering Considering Traffic Variation,” in *Proceedings of IEEE GLOBECOM*, December 2013.

2. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto, “Traffic Engineering Based on Stochastic Model Predictive Control for Uncertain Traffic Change,” in *Proceedings of IFIP/IEEE International Workshop on Management of the Future Internet*, May 2015.

3. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto, "Hierarchical Traffic Engineering Based on Model Predictive Control," in *Proceedings of International Conference on Computing, Networking and Communications*, February 2016.

4. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto, " Framework for Traffic Engineering under Uncertain Traffic Information," in *Proceedings of International Conference on ICT Convergence*, October 2016.

Non-Refereed Technical Papers

1. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, and Kohei Shiomoto, "Traffic Prediction for Dynamic Traffic Engineering Considering Traffic Variation," *Technical Reports of IEICE(IN2012-115)*, pp. 65–70, November 2012 (in Japanese).

2. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Noriaki Kamiyama, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto, "Evaluation of Traffic Engineering Considering Traffic Prediction," *Technical Reports of IEICE(IN2013-78)*, pp. 7–12, October 2013 (in Japanese).

3. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Noriaki Kamiyama, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto, "Evaluation of Traffic Engineering Based on Model Predictive Control Using Traffic Trace in Actual Network", *Technical Reports of IEICE(IN2013-194)*, pp. 299–304, March 2014 (in Japanese).

4. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto, "Evaluation of Traffic Engineering Based on Model Predictive Control Using Traffic Trace in Actual Network," *Technical Reports of IEICE(IN2014-81)*, pp. 1–6, November 2014 (in Japanese).

5. Tatsuya Otoshi, Yuichi Ohsita, Masayuki Murata, Yousuke Takahashi, Keisuke Ishibashi, Kohei Shiomoto, and Tomoaki Hashimoto, “Hierarchical Traffic Engineering Based on Model predictive Control,” *Technical Reports of IEICE(IN2014-136)*, pp. 91 – 96, March 2015 (in Japanese).

Preface

A variety of services have been deployed over the Internet in recent years, such as streaming and cloud services. Due to advancements in Internet-related technologies and the subsequent growth in the number of online users, online traffic and its variation in time have increased drastically. Backbone networks need to accommodate such fluctuating traffic without congestion. Thus far, the most common approach adopted to solve this problem has involved preparing an excess of resources to accommodate traffic surges. This approach, however, leads the low utilization of network resources most of the time.

One promising approach to accommodate large volumes of traffic with limited resources is dynamic traffic engineering (TE), where a controller monitors the traffic pattern at any given time and dynamically changes routes. However, prevalent TE methods typically set routes only for observed traffic. This renders the configured routes unsuitable when the volume of traffic along a route changes drastically, since routes are not updated until the next control cycle. Although the controller can quickly respond after such changes in traffic by setting a short-control cycle interval, frequent route changes lead to other problems, such as the degradation of TCP throughput. Thus, existing TE methods find it difficult to adapt to traffic changes and generate stable routes at the same time.

In this thesis, we study prediction-based TE methods to solve the above problem. In this method, a controller not only monitors the traffic pattern, but also predicts changes in the pattern to determine routes. If the controller knows that traffic flow along a certain route is going to increase, it can gradually make changes to routes before the actual increase occurs. Of course, the prediction incurs errors due to the uncertainty of changes in traffic, although many models for traffic prediction have been investigated in the literature. If traffic prediction is simply applied to TE, the resulting uncertainty leads to the setting of inappropriate routes, and leads to congestion or unnecessary route changes. Thus, the main problem is handling uncertainty concerning future traffic.

One approach to handling the above uncertainty involves setting a safe-side route that covers the upper bound of the traffic variation. For this purpose, we first propose a traffic prediction method to estimate the upper bound of future traffic variation. In this method, the observed traffic variation is first divided through preprocessing into a predictable variation and a noisy variation. With regard to the former, we predict future variation in traffic and estimate the error bound of the prediction. For the latter, we estimate the bounds of the variation. We thus obtain the upper bound of the future variation by adding the bounds of prediction error and noisy variation to the predicted variation. By applying this method to an actual traffic trace, we investigate the predictive accuracy and effectiveness of the proposed method for TE. We show that prediction-based TE with our prediction method can reduce the bandwidth required to accommodate traffic by 18.9% over a TE method that uses only observed traffic, though prediction errors were over 40% on average.

Predicting variations in traffic in the distant future is very useful for avoiding the consequences of drastic route changes due to fluctuation in the volume of traffic along routes of interest. However, predicting the distant future in this context incurs large prediction errors and leads to inappropriate decision concerning route changes. Thus, prediction-based control is required for robustness against prediction errors, especially in case of predictions related to the distant future. To this end, we introduce a control-theoretic method called model predictive control (MPC) to TE control. In our method, a controller calculates a route series to accommodate future traffic variation without drastic route changes. The controller then sets the route for the next time slot and monitors traffic after the route changes. By obtaining new data, the controller corrects its prediction results and recalculates the routes. Thus, it avoids the impact of prediction errors. Using a simulation, we showed that the proposed method can avoid congestion incurred by a prediction-based TE method that does not use the MPC mechanism. Moreover, we investigate the impact of such parameters as how far into the future traffic is predicted, the extent to which routes changes are avoided by the prediction, and the frequency with which control and prediction are executed.

Although MP-TE is robust against prediction errors, it offers no clear guarantee against prediction uncertainty. In order to effect more reliable control, we improve MP-TE to directly handle prediction errors. In this method, the controller calculates routes so that the probability of congestion is kept lower than a target probability. Because of the magnitude of the prediction error in the distant future, such a probabilistic constraint can become too severe and cause unnecessary route changes. Thus, we also propose a constraint-relaxation method where the guaranteed probability gradually increases from the target probability. In the simulation, we showed that the proposed method can achieve lower queuing delay than the original MP-TE. Furthermore, we showed that only a small number of additional route changes was required to accommodate the prediction error.

Finally, we address the hierarchical control of MP-TE to achieve scalability. In hierarchical control, multiple controllers are deployed over the network, which hierarchically determines routes. The controller in the bottom layer decides the specific routes in a small area, whereas the controller in the upper layer determines inter-area routes using abstract information concerning the lower layer. Since routes change in an area affects other areas through changes in the states of the network, cooperation among areas is required. The common approach to handling interaction among areas is to set a long control interval at the upper layer. This approach, however, causes another problem whereby the reaction at the upper layer delays to the environmental changes. To solve this problem, we again use the MPC in the hierarchical TE. Our method deploys an MP-TE controller to determine routes in each area. To cooperate with other controllers, each controller predicts their behavior to decide the routes. Through the simulation, we showed that our method achieves routing convergence more quickly than the prevalent hierarchical TE method. We also investigated sensitivity to prediction error, and found that our method works well even when the prediction error constitutes 76% of average traffic.

Acknowledgments

This thesis could not have been accomplished without the assistance of many people, and I would like to acknowledge all of them.

Foremost, I would like to express my deepest gratitude to my supervisor, Professor Masayuki Murata of Graduate School of Information Science and Technology, Osaka University, for his exact guidance, encouragement, and insightful comments.

I am heartily grateful to the members of my thesis committee, Professor Takashi Watanabe. Professor Toru Hasegawa, and Professor Teruo Higashino of Graduate School of Information Science and Technology, Osaka University, and Professor Morito Matsuoka of Cyber Media Center, Osaka University, for their multilateral reviews and perceptive comments.

Furthermore, I would like to show my sincere appreciation to Assistant Professor Yuichi Ohsita of Graduate School of Information Science and Technology, Osaka University, for continuous support, helpful discussion, and insightful advices.

Also, I would like to express my sincere appreciation to Dr. Kohei Shiomoto, Dr Keisuke Ishibashi, Dr. Noriaki Kamiyama, and Mr. Yousuke Takahashi of NTT network Technology Laboratory, for their helpful comments and fruitful discussions.

Moreover, I would like to show my appreciation to Assistant Professor Professor Tomoaki Hashimoto of Osaka Institute of Technology and Associate Professor Kenji Kashima of Kyoto University for support in the theoretic aspect.

Additionally, I must acknowledge Professor Naoki Wakamiya, Associate Professor Shin'ichi Arakawa, Assistant Professor Daichi Kominami, Specially Appointed Assistant Professor Naomi Kuze, Associate Professor Go Hasegawa, Assistant Professor Yuki Koizumi of Osaka University, for their valuable comments and suggestions on my study.

I express my appreciation to all of past and present colleagues, friends, and secretaries of the Advanced Network Architecture Research Laboratory, Graduate School of Information Science and Technology, Osaka University.

Finally, I conclude my acknowledgment with expressing my thanks to my parents and family. Thank you for giving me invaluable supports throughout my life.

Contents

List of publication **i**

Preface **v**

Acknowledgments **ix**

1 Introduction **1**

1.1 Background 1

1.2 Outline of Thesis 3

2 Traffic Prediction Method for Traffic Engineering **7**

2.1 Introduction 7

2.2 Related Work 10

2.3 Fixed Route Design Using Traffic Prediction 12

2.4 Traffic Prediction Process 14

2.4.1 Overview 14

2.4.2 Prediction 15

2.4.3 Prediction Preprocessing 19

2.4.4 Range of Excluded Variation 21

2.5 Evaluation 22

2.5.1 Datasets 22

2.5.2 Characteristics of the Traffic Prediction 23

2.5.3 Performance of the Traffic Engineering 29

2.6 Conclusion 34

3 Traffic Engineering Method Based on Model Predictive Control 41

3.1 Introduction 41

3.2 Dynamic Traffic Engineering and Traffic Prediction 43

3.2.1 Dynamic Traffic Engineering 43

3.2.2 Dynamic TE with Traffic Prediction 45

3.3 Traffic Engineering Based on MPC 47

3.3.1 Overview of MPC 47

3.3.2 Applying MPC to TE 48

3.4 Evaluation of Basic Behavior of MPC-based TE 50

3.4.1 Simulation Environment 51

3.4.2 Simulation Results 55

3.5 Evaluation in an Actual Network 57

3.5.1 Simulation Environment 58

3.5.2 Simulation Results 62

3.5.3 Discussion on Parameter Setting 62

3.6 Related Work 66

3.7 Conclusion 68

4 Traffic Engineering Guaranteeing Risk Probability against Prediction Uncertainty 69

4.1 Introduction 69

4.2 SMP-TE: Stochastic Model Predictive Traffic Engineering 71

4.2.1 Stochastic Model Predictive Control 71

4.2.2 Applying SMPC to TE 74

4.3 Evaluation 76

4.3.1 Simulation Environment 76

4.3.2 Effect of Stochastic Constraint 80

4.3.3 Multi-step Prediction Effect 80
4.3.4 Probability Relaxation Effect 81
4.3.5 Scalability 81
4.4 Conclusion 82

5 Scalable Traffic Engineering by Hierarchical Model Predictive Control 87

5.1 Introduction 87
5.2 Related work 90
5.3 Hierarchical Network Model 91
5.4 Hierarchical Network Control based on MPC 93
5.5 Hierarchical MP-TE 95
5.5.1 Hierarchical TE 95
5.5.2 Hierarchical MP-TE 100
5.6 Evaluation 102
5.6.1 Stationary Traffic Case 102
5.6.2 Dynamic Traffic Case 107
5.7 Conclusion 114

6 Conclusion 117

Bibliography 121

List of Figures

2.1 Prediction process 15

2.2 Internet2 topology 22

2.3 MAPE of each prediction method 25

2.4 Example of the predicted traffic time series using each preprocessing method . . . 27

2.5 Example of failure prediction with SARIMA 28

2.6 Example of the prediction for the sudden traffic change 35

2.7 Complement Cumulative distribution of maximum peak link load normalized by InvCap routing with different confidence levels in the SARIMA model prediction with the trend component 36

2.8 Complement Cumulative distribution of maximum peak link load normalized by InvCap routing with different confidence levels in the ARIMA model prediction with the trend component 37

2.9 Complement Cumulative distribution of maximum peak link load normalized by InvCap routing when the SARIMA model prediction with the different preprocessing is used in traffic engineering 38

2.10 Complement Cumulative distribution of required maximum peak link load normalized by InvCap routing when the ARIMA model prediction with the different preprocessing is used in traffic engineering 38

2.11 Box plot of gain of prediction-based traffic engineering in reduction of maximum peak link load compared with the observation-based traffic engineering (Control period: 12slots) 39
2.12 Complement Cumulative distribution of maximum peak link load normalized by InvCap routing when the ARIMA and SARIMA model predictions with the different preprocessing are used in traffic engineering 39

3.1 Overview of traffic engineering with traffic prediction 46
3.2 Overview of MPC 47
3.3 Overview of traffic engineering based on MPC 49
3.4 Simple network topology 51
3.5 Network traffic for simple network topology 52
3.6 Traffic exceeding target link capacity in a simple network 56
3.7 Average end-to-end delay of all OD flows in a simple network 58
3.8 Maximum end-to-end delay of all OD flows in a simple network 59
3.9 Internet2 topology 60
3.10 Internet2 network traffic 61
3.11 Traffic exceeding target link capacity with actual Internet2 traces 63
3.12 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 64
3.13 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 ($w = 0.5$) 65
3.14 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 ($w = 0.5$) 67

4.1 Overview of SMP-TE 74
4.2 Internet2 topology 77
4.3 Internet2 network traffic (time slot granularity: 2 h) 78
4.4 99.9% Queuing Delay on the Bottleneck Link at Each Time 84

4.5 Maximum Difference of Path Fraction at Each Time 85

5.1 The hierarchical network model. 96
5.2 Lattice topology with 64 nodes. 103
5.3 Time series of traffic causing inner-area congestion. 104
5.4 Time series for maximum link load and average route changes with increasing inner-area traffic. 105
5.5 Time series for maximum link load and average route changes with increasing inter-area traffic. 106
5.6 Time series of maximum link load and average route changes of MP-TE with various weights of route changes ($h = 3, \sigma = 0$). 109
5.7 Time series of maximum link load and average route changes of MP-TE with various weights of routes changes ($h = 3, \sigma = 2.3 \times 10^6$) 110
5.8 Time series of maximum link load and average route changes of MP-TE with various lengths of prediction ($w = 0.8, \sigma = 0$). 111
5.9 Time series of maximum link load and average route changes of MP-TE with various lengths of prediction ($w = 0.8, \sigma = 2.3 \times 10^6$). 112
5.10 Time series of maximum link load and average route changes of simple TE ($\sigma = 0$). 112

List of Tables

2.1 Number of Data Used in Training and Test series 23

2.2 Values of Parameters for Confidence Levels (α,β) 31

4.1 Route Changes Caused in Each TE Method 82

5.1 σ values and standard deviations of prediction error. 108

Chapter 1

Introduction

1.1 Background

The Internet has come to play an important role in our society, not only as a novel means of communication, but also as an indispensable infrastructure to a variety of industries today. In the early years, its main application was text-based exchanges, such as e-mail, chat, and simple text-based webpages. In recent years, various services have been deployed over the Internet, such as streaming and cloud services. Due to the enhancement of Internet services and the increase in the number of users over the years, fluctuation in online traffic has increased. This situation is expected to persist over the next few decades with the emergence of new Internet services and technologies, including the Internet of Things (IoT). The *backbone network*, which is the core network of the Internet, needs to accommodate this variable traffic without becoming congested.

The most common current approach to address the above problem of online traffic is *overprovisioning* [1, 2]. That is, the network manager prepares an excess of resources in order to avoid congestion even when traffic surges. Overprovisioning requires a large network capacity, and leads to poor use of resources, approximately 17%–29% in case of Google backbone [3], less than 50% in the Sprint backbone [4], and less than 20% in Internet2 [5]. This increases waste of equipment and management. Thus, online traffic needs to be accommodated with a limited amount of resources to save cost due to overprovisioning.

Traffic engineering (TE) is a promising approach to accommodate traffic with limited resources. TE implements load balancing over network resources by changing the routes of traffic patterns. A number of TE methods have been proposed in the literature [6–19]. They can be roughly divided into two types, static TE and dynamic TE, depending on whether the routes are altered according to traffic variation.

Static TE involves methods that set a fixed route to accommodate traffic. A common method is *oblivious routing* [6–8], which calculates a fixed route without prior knowledge of traffic statistics. In oblivious routing, the route is calculated by minimizing a metric called the *oblivious ratio*, which represents the worst ratio of the maximum link load to its optimal value. Although this method guarantees the performance in the worst case, it worsens in normal situations. In [8], Németh showed that oblivious routing involves the greatest expenditure of time in a congested state with a number of source-destination pairs, even though the oblivious ratio is kept low. Thus, static TE is not suitable to accommodate traffic that significantly varies over time.

Dynamic TE involves changing routes in accordance with changes in traffic. In dynamic TE, a controller periodically collects traffic information from network monitors and changes routes to accommodate the observed traffic pattern [12–14]. However, these methods typically set routes only for the observed traffic. This renders the configured routes unsuitable once traffic changes drastically because the routes are not updated until the next control cycle. Although the controller can quickly respond to such traffic changes by setting a short control cycle interval, frequent route changes cause routing oscillation, which degrades TCP throughput. Since the packets of a TCP session are transferred through different paths during routing oscillation, the order of received packets is different from that of sent packets. Such packet reordering reduces the window size of the TCP session and, hence, its throughput. Routing oscillation also causes fluctuation in the round-trip time (RTT) of a TCP session, which degrades the throughput of delay-based TCP [20]. Thus, avoiding significant routes changes is important for dynamic TE.

A promising solution to the above problem involves predicting future traffic variation to calculate the routes change schedule. If the TE controller knows in advance that traffic drastically changes at certain points of time and in certain places in the future, it can gradually change routes before the traffic change occurs. Thus, drastic routes change can be avoided in prediction-based

TE.

Of course, prediction is uncertain, although many models have been studied for traffic prediction, such as ARMA, ARIMA [21, 22], ARCH [23], GARCH [24], and neural networks [25–27]. When applying traffic prediction to TE, such uncertainty leads the controller to set inappropriate routes, which degrades TE performance. For instance, the controller allocates fewer resources to a traffic flow than actually required if traffic is underpredicted. Conversely, the controller allocates excessive resources to a traffic flow, which causes resource shortage in other flows, if traffic is overpredicted.

In this thesis, we address the problem of handling uncertainty concerning future traffic to establish prediction-based TE, and propose methods to this effect.

1.2 Outline of Thesis

Traffic Prediction Method for Traffic Engineering [28–30]

To avoid congestion due to uncertainty in traffic changes, TE should set a safe-side route to accommodate the upper bound of traffic variation. In Chapter 2, we first propose a traffic prediction methodology to estimate the upper bound of future traffic variation. In this method, observed traffic variation is separated into predictable, longer-term variation and noisy, short-term variation by preprocessing. A time-series model is then fitted to only predictable variation. By using the fitted model, future variation and estimated prediction error are calculated. For noisy variation, our method estimates the range of noisy variation by a standard deviation. Finally, the predicted upper bound of future traffic is calculated by summing up the predicted traffic, and the bounds of the prediction errors and noisy variation. Since the preprocessing method determines the target variation for prediction, the accuracy of prediction and the upper bound depends on the preprocessing method. Thus, we assess the prediction method with three preprocessing methods—*lowpass filter*, *trend component*, and an *envelope*—to investigate the preprocessing suitable for TE. By applying the prediction method to a traffic trace in a backbone network, we first investigate the characteristic of the prediction results with two common prediction models, ARIMA and SARIMA. From the aspect of prediction accuracy, the prediction result contained errors for all preprocessing methods tested. Although the lowpass filter achieved the lowest prediction error among other preprocessing methods as well as in the case of prediction without preprocessing, the average prediction error was over 40%. We also investigated the characteristics of the prediction results in detail, and found that preprocessing improved prediction accuracy in case of daily variation. and upper bound estimation for a sudden traffic increases. Following this, we evaluate TE performance using our prediction method to investigate the effectiveness of the predicted traffic in the TE. We used a simple TE method for comparison, where the controller calculated fixed routes for multiple time slots, such as 12 hours, by minimizing the peak of link utilization using the predicted traffic. We were able to reduce the required bandwidth to accommodate traffic by 18.9% using the trend component and SARIMA over the simple TE, which used only observed traffic.

Traffic Engineering Based on Model Predictive Control [31–33]

To alter routes according to traffic variation without drastic changes, we need to predict traffic not only in the near future, but also in the distant future. The prediction of the distant future, however, induces a large prediction error in general. Thus, In Chapter 3, we propose a prediction-based TE method that avoids the impact of prediction errors, especially in the far future. To achieve this, we focus on a methodology in system control called model predictive control (MPC) [34–37]. In MPC, a system controller calculates an input series to avoid drastic input changes by predicting how the future output of the system will change. Once the controller calculates the optimal input series for the future, the controller enters the first input to the system. The controller then obtains a new system output as feedback, and corrects the prediction and the input at a later time. To apply this MPC mechanism to TE, we first model the network as a system where the input is the route and the output is link congestion. We then propose a prediction-based TE method called *Model Predictive Traffic Engineering (MP-TE)*. We evaluate the performance of our method using an actual traffic trace, and found that the MP-TE can avoid congestion that simple prediction-based TE cannot. Moreover, we investigate the impact of such parameters as the extent in the future to which traffic needs to be predicted, the extent to which route changes should be avoided, and the frequency with which control and prediction should be executed. We found that performance was not sensitive to the length of prediction and the weight of the route changes. Furthermore, we found route changes at 10-second intervals was sufficient to accommodate traffic changes every second, whereas the prediction was required every second.

Traffic Engineering Guaranteeing Risk Probability against Prediction Uncertainty [38, 39]

Although we propose a prediction-based TE method, there is no clear guarantee against prediction uncertainty in our method. In Chapter 4, we improve the MP-TE to handle the prediction error more directly in the search for highly reliable control. Fortunately, such uncertainty-aware control has also been considered in the MPC, such as stochastic MPC [40]. In stochastic MPC, the controller predicts system behavior with a probability distribution and calculates the probability that the system's states violate system constraints. By applying stochastic MPC to TE, we propose the *Stochastic MP-TE (SMP-TE)*, which guarantees that the probability of congestion will remain below a certain target probability. Such probability constraints, however, can become too severe, especially in case of predictions of the distant future, because prediction errors generally increase when the prediction target is distant in the future. This leads to frequent and unnecessary routes changes because the controller changes routes to compensate for large prediction errors, even if the relevant traffic pattern never emerges. Thus, we also propose a constraint-relaxation method to avoid unnecessary route changes, where the guaranteed probability gradually increases from the target probability for the distant future. Through a simulation involving a real network traffic, we showed that the SMP-TE maintains a lower queuing delay than the original MP-TE by directly handling prediction error. We also showed that the constraint-relaxation method reduces the frequency of route changes by reducing unnecessary ones.

Scalable Traffic Engineering by Hierarchical Model Predictive Control [41, 42]

In Chapter 5, we propose a hierarchical TE method based on the MP-TE to achieve scalable network control. Hierarchical control is a common approach to controlling large-scale networks, and many past studies have proposed various methods [43–48]. In hierarchical network control, the network is divided into multiple areas with multiple layers. In the bottom layer, each area has only a small number of nodes and links. In the upper layer, each area is constructed by multiple lower areas whose original topology is abstracted from. A controller is then deployed in each area at each layer. The controller collects network information in the area to determine the operation there, and sends abstracted information to other controllers in the upper layer. Thus, each controller only manages a small topology to calculate operation, even if the entire network becomes large. One difficulty in hierarchical network control is the oscillation of operation due to interactions among controllers in other layers. A common approach to avoid oscillation is to set a long control interval in the upper layer [49, 50]. This approach, however, causes a response delay in the upper layer, and the changes in the environment across the relevant areas leads to severe congestion, which is ever-more pronounced nowadays because of the content distribution network (CDN), user mobility, and so on. Thus, a new approach is required to avoid oscillation without setting a long interval in the upper layer. Our approach here involves predicting the behavior of other controllers to achieve cooperation among them. More specifically, we deploy MP-TE controllers in the hierarchical TE, where each predicts how much traffic is sent to a given area from other areas, and how much can be sent to the other areas to determine the routes. Through a simulation, we showed that our method more quickly responds to traffic changes and reduces congestion in comparison with the existing hierarchical TE approach. We also investigated sensitivity to the prediction error and the sensitivity of parameters, i.e., the length of prediction and the weight of route changes. We found that our method works well even if the prediction error is as high as 76% of average traffic, which is large enough to be an actual prediction error. Moreover, we found that the upper layer should set a large weight for route changes, whereas performance is not very sensitive to other parameters.

Chapter 2

Traffic Prediction Method for Traffic Engineering

2.1 Introduction

In recent years, time variation of Internet traffic has increased due to wide deployments of streaming and/or cloud services. Backbone networks are expected to accommodate such time-varying traffic without congestion. So far, backbone networks have addressed this problem by preparing redundant link capacity by considering not only average traffic but also traffic surges [1, 2]. However, such an approach requires overly large capacity in accordance with the level of traffic change increases and causes low bandwidth utilization. For the last dozen years, the literature has reported that average link utilization of backbone networks has been very low, such as 17–29% in Google backbone [3], less than 50% in Sprint backbone [4], and 20% utilization is targeted in Internet2 [5]. This not only causes the waste of the bandwidth due to poor utilization of the network resource but also incurs unnecessary energy consumption. Henceforth, the traffic congestion must be avoided with limited resources, which will definitely reduce the over-provisioning cost and power consumption.

Adaptive traffic engineering is a promising approach for accommodating time-varying traffic by appropriately setting up the Origin–Destination (OD) routes [9–13]. In such traffic engineering methods, a control server periodically measures the traffic load in the network (typically every hour) and dynamically changes the routes so as to minimize the network congestion. However, traffic engineering using the measured traffic only mitigates the observed congestion and never avoids the future congestion. The currently congested links are resolved by changing routes at the next control epoch. By making the control interval shorter (say, in a unit of minutes), the control server may respond quickly to such traffic changes. However, it obviously causes the heavy load at the control server and affects the performance of the upper-layer protocol TCP due to frequent route changes. Such routing oscillation degrades the throughput of TCP sessions because of packet reordering and changes of RTT [20]. Our solution here is to execute traffic engineering by predicting the future traffic changes. That is, the control server should set up routes by considering the future traffic demands, not past ones. More exactly, the control server predicts the traffic variation in the next control cycle and then determines routes that can accommodate the predicted traffic without causing congestion in the next control cycle. For deciding the traffic variation, we again have the "time-scale" problem: if we want to have stable operation, we need set up a larger control cycle, but in that case, we cannot react to the temporal changes of traffic variation within the control interval. The shorter control cycle has exactly the same problem described above.

So far, various prediction methods have been studied on the basis of traffic predictive models such as ARMA, ARIMA [21,22], ARCH [23], GARCH [24], and Neural Network [25–27]. However, to the best of our knowledge, existing prediction methods do not solve the above problem because they can predict the traffic variation accurately only for its target time scale. For example, the method proposed by Guang et al. [26] targets prediction in time scale of several hours. Therefore, it cannot obtain information about shorter-term variations because they are removed as noise before the prediction. On the other hand, a prediction method targeting a small time scale such as milliseconds or minutes [21,23,24,51] is only effective for very near future prediction because of the significant degradation of prediction accuracy in the far future.

In this chapter, we propose a traffic prediction procedure intended for application to traffic engineering by separating the short-term (non-periodical or temporal) and longer-term (hour or day) variations. We directly predict the longer-term variation as existing methods and estimate the short-term variation instead of predicting it. We then obtain the range of traffic variation including short-term variation during the next several hours, which is used as a basis for calculating the necessary capacities of each route in the next control interval. That is, our key contribution here is that we investigate how to handle the prediction uncertainty in order to apply our method to traffic engineering. As described before, the prediction uncertainty stems from two factors (prediction error for periodical pattern and noisy short-term variation), and we take account of such prediction uncertainty in determining the necessary resources for each route. In this chapter, we focus on the results of traffic engineering instead of the accuracy of prediction, because prediction methods with small error are not always suited to traffic engineering. Even when mean prediction error is low, congestion cannot be avoided by traffic engineering using the predicted traffic if the temporal increase of the traffic causing congestion is not predicted. On the other hand, a prediction method responsive to the traffic increase that may cause the congestion can avoid congestion even if the method's mean prediction error is large. Therefore, we evaluate our prediction procedure by investigating the influence of prediction method on traffic engineering performance.

In our earlier work [29], we only compared the effectiveness of traffic engineering using predicted traffic with observation-based traffic engineering. This chapter also investigates details of the impact of traffic prediction on traffic engineering. We first investigate the impact of two parameters in our prediction procedure having a large impact on traffic engineering, the confidence level of prediction errors and the confidence level of short-term variation. We find that the confidence level of the short-term variation should be set to a large value, while a small confidence level for prediction errors is generally sufficient.

We then investigate the impact of considering periodicity, and find that even prediction without considering periodicity is sufficient if the control period is a few hours, while traffic prediction considering periodicity improves the worst-link utilization achieved by traffic engineering if the control period is larger than 24 hours.

The rest of this chapter is organized as follows. Section 2.2 surveys related work of traffic prediction and traffic engineering. Section 2.3 introduces the traffic engineering method using the predicted traffic. Section 2.4 describes the prediction procedure. Section 2.5 presents an evaluation of our prediction procedure. Section 2.6 mentions the conclusion and future work.

2.2 Related Work

traffic engineering

There is a large body of literature regarding TE [9–13]. The most of existing traffic engineering methods are observation-based approach in which the control server collects the current traffic information and then sets the routes so as to accommodate the observed traffic. However, such observation-based method may not be able to accommodate the future traffic because the traffic pattern will change from the observed pattern.

One approach to handling such uncertainty of the future traffic is to allocate sufficient resources to accommodate worst-case traffic patterns. For example, a static routing method called *oblivious routing* [6–8] sets a fixed route to accommodate worst-case traffic. Instead of observing current traffic, this method tries to accommodate all possible traffic patterns by minimizing the maximum link load. Wang *et al.* proposed a robust traffic engineering method by introducing the oblivious routing concept [9]. Their method considers the *convex hull* of a set of historical traffic patterns, namely the set of arbitrary weighted average of observed traffic. It handles uncertain future traffic dynamics by optimizing routes for this convex hull under constraints where the worst-case performance is not degraded. However, the approach requires large resources to accommodate worst-case traffic.

To accommodate the future traffic variation with a small resources, it is important to know the future traffic. Thus, our traffic engineering approach uses the prediction of time series of traffic to decide the routes.

traffic prediction

The predictability of Internet traffic has received significant interest in various domains, such as capacity planning, anomaly detection, admission control, and traffic engineering. The prediction methods of the network traffic have been studied for various time scales such as milliseconds, seconds or minutes order [21, 23, 24, 51], daily [25–27], and even monthly variation [22].

The prediction based traffic engineering requires the traffic prediction for the control period.

The control period may be a few hours or more. Thus, the traffic prediction should follow the daily variation. On the other hand, the traffic variation during the control period includes the temporal changes, which should also be considered by the traffic engineering so as to avoid the congestion.

Some of existing prediction methods focus on the daily traffic variation [25–27]. However, they exclude the short-term variation, which is also important for the traffic engineering. For instance, the method in [26] eliminates the values which is too far from average traffic value, and then removes the white noise from the data by Fourier analysis before inputting the data to the prediction process. If these eliminated data is not considered in traffic engineering, the calculated routes cannot accommodate the temporal traffic change and may cause the congestion.

One simple approach to consider these removed variation is to use the short-term prediction method [21, 23, 24, 27, 51]. However, the short-term prediction method causes a large prediction error when it is used to predict the traffic during the control period, which may be a few hours or more. To predict the daily variation with a small time granularity, a number of iterations of one-step ahead prediction is required, which causes inaccurate prediction for the distant future due to the accumulation of errors. For instance, in [27], the error of the iterative prediction with 5 minutes of granularity monotonically increases as the prediction target becomes long.

Therefore, in this chapter, we clarify how to handle the long-term and short-term variation for the prediction based traffic engineering. In our approach, we decompose the traffic variation into long-term and short-term variation. Then, in addition to the prediction of the long-term variation, we also estimate the range of the short-term variation. Finally, we obtain the predicted upper bound of traffic variation by summing the predicted long-term variation and estimated range of the short-term variation.

In addition, we evaluate the prediction method combined with the traffic engineering. Though most of the existing work on the traffic prediction discuss their prediction accuracy by comparing the predicted values with the actual values. However, prediction errors of some flows may have only a small impact on the performance on the traffic engineering, while other flows may have a large impact; the large flows affect the link utilization significantly than the small flows and may be required to be predicted accurately. Therefore, in this chapter, we discuss the suitable prediction method considering the results of the traffic engineering using the predicted traffic.

2.3 Fixed Route Design Using Traffic Prediction

In this chapter, we deploy a central control server that controls the network. The central control server observes and predicts the traffic rate and calculates routes on the basis of the predicted traffic.

The control server observes the traffic rate at each flow in fixed intervals (e.g. 10 minutes, 30 minutes, or one hour) called *time slots*. The observed traffic rates of all flows in the t-th time slot are represented as a vector. We denote this vector as $\boldsymbol{x}_t$. The aggregation of a number of flows is useful for reducing the observing cost and prediction time. In this chapter, we aggregate the flow as OD flow that traverses from the ingress Point-of-Presence (PoP) router to the egress PoP router. This flow grain is sufficient to decide the routing in a backbone network, and the existing observation based traffic engineering methods often use OD flow [9, 12].

Using the observed traffic rates until the t-th time slot, the control server predicts the future traffic rates in the next f time slots. The prediction of future traffic is formulated as

$$\hat{\boldsymbol{x}}_{t+1..t+f} = F\left(\boldsymbol{x}_{t-h+1..t}\right), \tag{2.1}$$

where $\boldsymbol{x}_{a..b} = (\boldsymbol{x}_a, \boldsymbol{x}_{a+1}, \cdots, \boldsymbol{x}_b)$ is a matrix in which each column corresponds to each vector, $\hat{\boldsymbol{x}}_k$ is the predicted traffic in the k-th time slot, f is the number of time slots where the traffic rate is predicted, h is the length of observed time slots used in the prediction, and F is a prediction function defined by a prediction method.

In traffic engineering, the control server calculates the routes so as to avoid congestion for f time slots. We define these f time slots as the *control period*. In this chapter, we consider the case in which the control period is 3–24 hours. The calculated routes are represented as a matrix A called *routing matrix*. The (i, j)-element $a_{i,j}$ in the routing matrix A represents the ratio of the traffic over the OD flow j mapped onto the link i. Corresponding to the routing matrix, the predicted traffic mapped onto each link in the control period is represented as

$$\hat{\boldsymbol{y}}_{t+1..t+f} = A\hat{\boldsymbol{x}}_{t+1..t+f}, \tag{2.2}$$

where $\hat{\boldsymbol{y}}_k$ is the vector indicating the predicted traffic on all links in the k-th time slot. Traffic engineering is the process to adjust A so as to control $\hat{\boldsymbol{y}}_{t+1..t+f}$ in some desirable way.

In traffic engineering, the most widely used metric of congestion is maximum link utilization [9, 12], i.e. the utilization of the most congested link. In this chapter, we use a simple optimization approach that minimizes the maximum utilization among all links for all time slots within the control period, though there may be a more sophisticated approach using the predicted traffic. Using this simple approach, we can clarify the impact of the prediction on the traffic engineering performance by simply observing the achieved maximum link utilization. If the traffic engineering method using the traffic information predicted by a method keeps the small link utilization for a long time, the prediction method is suitable for the traffic engineering intended to stabilize traffic accommodation.

The optimization problem is formulated as the following linear programming problem:

$$\begin{aligned}
minimize \quad &: \quad U && (2.3)\\
subject\ to \quad &: \quad \forall s,d,\ \sum_{p(l)=s} A^{s,d}(l) = 1 && (2.4)\\
&\quad\ \ \forall s,d,\ \sum_{f(l)=d} A^{s,d}(l) = 1 && (2.5)\\
&\quad\ \ \forall s,d,n,\ \sum_{p(l)=n} A^{s,d}(l) = \sum_{f(l)=n} A^{s,d}(l) && (2.6)\\
&\quad\ \ \forall l,k \in [t+1,t+f], \sum_{s,d} \frac{A^{s,d}(l)\hat{x}_k^{s,d}}{C(l)} < U, && (2.7)
\end{aligned}$$

where U is the maximum link utilization, $A^{s,d}(l)$ is the ratio of traffic from s to d routed over the link l, and $p(l)$ and $f(l)$ are the start and end nodes of the link l, respectively, $\hat{x}_k^{s,d}$ is the predicted traffic rate of the flow from s to d at the k-th time slot and $C(l)$ is the capacity of the link l. $\hat{x}_k^{s,d}$ and $C(l)$ are given in this problem, and $A^{s,d}(l)$ and U are the variables to be obtained. Eqs. (2.4–2.6) are the constraints for flow conservation. Eq. (2.7) ensures that U is the maximum link utilization of all the links for all the time slots within the given control period. By solving the above problem, we obtain routing matrix A, which is used for the control period $[t+1, t+f]$, and is not changed before $t+f+1$. Setting f to a large value avoids frequent route changes, but to do so the traffic prediction should be response to traffic variation occurring in the control period $[t+1, t+f]$; if the predicted traffic cannot respond to the temporal traffic variation that occurs in some time slots, congestion may occur. In Section 2.4 we therefore discuss a traffic prediction procedure that considers traffic variation in each time slot.

To map the routing matrix A to actual network, we assume the paths between an OD pair of routers are determined by MPLS Label Switched Paths (LSPs). According to the link-based routing determined by A, the control server can calculate the path-based routing, i.e. defining the link set used by each LSP and split ratio among the LSPs. Each PoP router splits traffic among the LSPs corresponding to an OD flow using the hashing method described by Anwar et al. [13]. In this method, each fine-grain flow (e.g TCP flow) is routed to only one LSP to avoid the packet reordering that degrades TCP throughput.

2.4 Traffic Prediction Process

2.4.1 Overview

In the network, traffic variation has a daily pattern in longer-term (hour or day) variation, and the traffic changes every few hours. The traffic prediction needs to follow longer-term variation so that the traffic engineering calculates the routes suitable for the next few hours. However, the actual traffic variation includes noisy variation (short-term variation), and the longer-term tendency is polluted. Such polluted data cause a large prediction error. Therefore, we use preprocessing that extracts the daily periodical variation excluding the noisy variation to improve the prediction accuracy.

On the other hand, the short-term traffic variation excluded by the preprocessing may cause the congestion. The short-term traffic variation is hard to predict, but it can be considered as a noisy fluctuation whose mean and variance are stable if the preprocessing extracts the longer-term traffic variation accurately. Thus, we consider the short-term traffic variation by calculating the variance of the traffic variation excluded by the preprocessing. Then, by adding the confidence interval of the calculated variance to the predicted longer-term traffic variation, we avoid the underprediction caused by the short-term traffic variation.

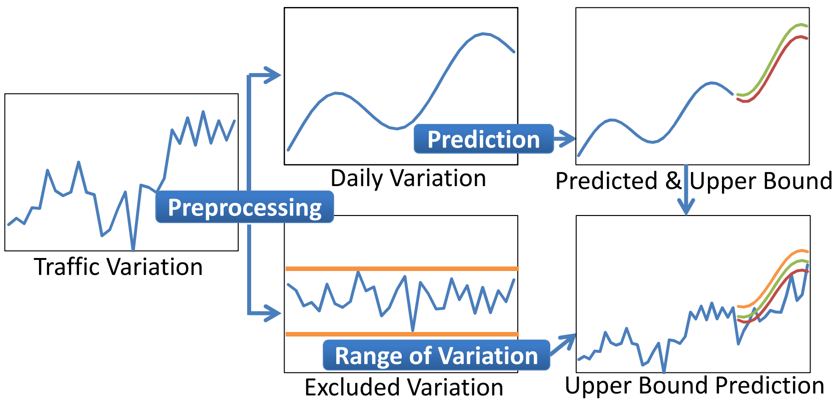


Figure 2.1: Prediction process

Moreover, we also consider the confidence interval of the prediction error to avoid the impact of the prediction error on the traffic engineering. The confidence interval causes the overprediction. However, the overprediction has a smaller impact than the underprediction. This is because the underprediction causes the lack of allocated resources and congestion while the overprediction does not affect the communication performance until the overpredicted flow blocks resources to be allocated to other flows.

Our approach is summarized in Fig. 2.1. First, we extract the longer-term variation from the actual traffic variation by the preprocessing. Second, we predict the future traffic variation using the extracted variation and estimate the variance of excluded variation. Finally, we obtain the upper bound of traffic variation summing up the predicted upper bound of longer-term variation and the confidence interval of the excluded variation. The obtained upper bound is used as input of the traffic engineering.

2.4.2 Prediction

After each preprocessing, the traffic prediction process calculates the future traffic and its confidence interval. To predict the traffic, many prediction methods have been proposed such as methods based on time series models and neural network. The model-based prediction method fits the model parameters from inputted data and then predicts the future values and the confidence interval of the prediction. Although other approach such as neural network does not require to assume a certain model for the original traffic variation, they have difficulty in obtaining the exact interval of the prediction and only approximations are available [52]. Though our prediction process is not limited by a certain prediction method, we use the model-based approach since we focus on the effect of considering the confidence interval. We use two traffic prediction models ARIMA and SARIMA as examples to discuss the effect of considering the periodicity of traffic variation. The rest of this section gives an overview of prediction with the ARIMA and SARIMA models.

2.4.2.1 Prediction models

ARMA model Before describing the ARIMA and SARIMA models, we briefly explain the ARMA model, which is the base model for the ARIMA and SARIMA models.

The ARMA model represents data at each time slot using the previous data and errors as

$$\begin{aligned} x_n &= \sum_{i=1}^{p} a_i x_{n-i} + \sum_{i=0}^{q} b_i \epsilon_{n-i} + c \qquad (2.8) \\ b_0 &= 1, \end{aligned}$$

where p and q respectively denote the numbers of past data and errors on which the current data depends. a_i and b_i are the coefficients, ϵ_i is the error at the i-th time slot and c is a constant.

ARIMA model The ARIMA model is an extension of the ARMA model so as to model the non-stationary data, such as the data whose mean value fluctuates over time. To apply the ARMA model to such data, the non-stationarity is removed. When the variation of the mean has a linear characteristic, the differenced data $\Delta x_n = x_n - x_{n-1}$ exclude the variation of the mean. In this manner, d times differencing operation Δ^d can remove the mean variation following a polynomial of degree d. In the ARIMA model, ARMA model in Eq. (2.8) is applied to the differenced data $\Delta^d x_n$.

SARIMA model The SARIMA model is a generalization of the ARIMA model. Considering the periodicity, the SARIMA model applies a periodical differencing to the data as $\Delta_s x_n = x_n - x_{n-s}$, where s is a period length. After the D times of the periodical differencing $\Delta_s^D x_n$ are applied, the differencing method in the ARIMA model is also applied. Therefore, differenced data are finally denoted as $\Delta^d \Delta_s^D x_n$. Considering the daily periodicity and the weekday/weekend difference, we set s to the weekly length.

The differenced data are fitted to the following model, which expands the ARMA model by adding the data and errors in previous periods as

$$\begin{aligned} x_n &= \sum_{i=1}^{p} a_i x_{n-i} + \sum_{i=0}^{q} b_i \epsilon_{n-i} + c \\ &\quad + \sum_{j=1}^{P} A_j \sum_{i=1}^{p} a_i x_{n-sj-i} + \sum_{j=1}^{Q} B_j \sum_{i=0}^{q} b_i \epsilon_{n-sj-i} \\ b_0 &= 1, \end{aligned} \tag{2.9}$$

where P and Q denote the numbers of previous periods for depended data and errors, respectively. A_i and B_i are the coefficients that indicate how the previous i-th period affects the current time slot.

2.4.2.2 Model Fitting

An ARIMA or SARIMA model is fitted to the data by the following steps.

First, the differencing parameter is determined by differencing the data until the data become stationary. A stationarity test is performed by examining whether the data follow a non-stationary process $x_t = x_{t-1} + \epsilon$ called *unit root process*. We use the KPSS test [53] for determining d. The KPSS test examines the null hypothesis $\epsilon = 0$, which means the data are stationary. For determining D in the SARIMA model, we use the Canova-Hansen test [54]. The Canova-Hansen test applies the null hypothesis test to the Fourier coefficients variation of each period.

Second, the coefficients and the number of the terms in a model are determined. To determine the number of the terms in a model, we determine the coefficients by the MLE for each case of the number of terms. Then, we determine the model by selecting the model with the highest goodness among the models calculated by the MLE. The goodness of a model is defined by the Akaike

Information Criterion (AIC) [55], which is defined by

$$\mathrm{AIC} = -2\log L + 2k, \tag{2.10}$$

where L is the maximized likelihood with the MLE and k is the number of parameters. $k = p + q + P + Q$ in the SARIMA model, and $k = p + q$ in the ARIMA model. A model with a large number of parameters can fit the data well but may fit the incidental variation such as noise. By penalizing k, AIC can select the best model while avoiding overfitting the data. To search for the model with the highest goodness, we use the method proposed by Hyndman *et al.* [56]. In this method, the model with the highest goodness is searched for by changing p, q, P and Q by one until no new model can improve AIC.

2.4.2.3 Prediction with Fitted Model

After the fitting of a model, the future traffic is predicted in accordance with the obtained model. The predicted traffic in the next k-th time slot is calculated as following conditional expectation of x_{t+k} given the previous observation values:

$$\bar{x}_{t+k} = E[x_{t+k}|x_{t-h+1..t}]. \tag{2.11}$$

According to the prediction model (2.8 or 2.9), the traffic rate of the next one time slot is directly calculated with observation values. The next two or more time slots are iteratively predicted by using the former predicted value instead of the observation value.

2.4.2.4 Confidence Interval

The model-based prediction can calculate the confidence interval for the prediction error. The upper confidence bound for the prediction can be calculated by $\bar{x}_{t+k} + \alpha\hat{\sigma}_{t+k}$, where $\bar{x}_{t+k}$ is the predicted traffic rate at the next k-th time slot, α is a parameter indicating the considered confidence level, and $\hat{\sigma}_{t+k} = \sqrt{V[x_{t+k}|x_{t-h+1..t}]}$ is the estimated standard deviation of prediction error where $V[x_{t+k}|x_{t-h+1..t}]$ is the conditional variance of predicted value given the observed values.

2.4.3 Prediction Preprocessing

In the preprocessing, we extract the daily periodical variation from the observed traffic. The object of preprocessing is to filter out the short-term traffic variation that is hard to predict. This increases the accuracy of the prediction of the longer-term traffic variation.

In this chapter, we investigate the following preprocessing methods: the lowpass filter, the trend component, and the envelope. The rest of this subsection details the preprocessing methods.

2.4.3.1 Lowpass Filter

One approach to extract the longer-term variation of the traffic variation is to use the lowpass filter, which extracts the longer-term variation using the Fourier transform.

By using the Fourier transform, the time series of the traffic data can be represented as

$$x_k = \sum_{n=0}^{h-1} f_n \exp\left(2\pi i \frac{nk}{h}\right), \tag{2.12}$$

where f_n is Fourier coefficient corresponding the frequency n/h and i is the imaginary unit. Eq. (2.12) also includes high frequency variations such as noise. To reduce these noisy variations, the lowpass filter removes the terms with large n and extracts the longer-term variation as

$$l_k = \sum_{n=0}^{L} f_n \exp\left(2\pi i \frac{nk}{h}\right), \tag{2.13}$$

where L is the threshold to remove the high frequency variations. In this chapter, we set L so as to remove the variation of frequency higher than the daily variation, i.e. the lowpass filter extracts the daily pattern of traffic variation.

2.4.3.2 Trend Component

In the second approach, we extract the longer-term variation by using a time series model. One approach to model the longer-term variation is the trend model [57]. We call the traffic variation extracted by using the trend model *trend component*. The trend component includes the daily traffic variation and longer-term traffic variation. The trend model is denoted as

$$x_k = t_k + \epsilon_k \tag{2.14}$$

$$\Delta t_k = \Delta t_{k-1} + w_k, \tag{2.15}$$

where x_k is the traffic rate of a flow in the k-th time slot, t_k is the trend component, $\Delta t_k = t_k - t_{k-1}$, $\epsilon_k \overset{\text{i.i.d.}}{\sim} N(0, \theta^2)$ is the noise of observation, and $w_k \overset{\text{i.i.d.}}{\sim} N(0, \lambda^2)$ is the noise in the trend component.

Eq. (2.14) indicates that the original data are composed of the trend component and the noise, and Eq. (2.15) indicates that the trend component is perturbed by Gaussian noise.

At the first step to calculate the trend component, the variances θ^2 and λ^2 are found by the Maximum Likelihood Estimation (MLE). Then, the trend component $t_i (i = t - h + 1, \cdots, t)$ is determined by the conditional expectation $E\left[t_i | x_{t-h+1..t}\right]$ with the probability of transition in Eqs. (2.14) and (2.15).

In terms of extracting the daily variation, the trend component approach is the same as the lowpass filter. However, the trend component extracts the main tendency of the traffic variation, while the lowpass filter extracts the targeted frequency component. Therefore, the trend component also extracts the variation mismatched to the frequency component when the variation can be taken as the main tendency.

2.4.3.3 Envelope

Extracting the variation of traffic upper bounds may be useful to predict the bandwidth required to accommodate the short-term traffic variation. In the third approach, we extract the upper bound variation by tracing the peak value in the fixed time interval. We divide the observed values $x_{t-h+1}, \cdots, x_t$ into $l = \frac{h}{\tau}$ intervals, where τ denotes the length of the intervals. The set of the time slots in the k-th interval is denoted as

$$I_k = \left\{(k-1)\tau + t - h + 1, \cdots, k\tau + t - h\right\}. \tag{2.16}$$

We set the interval length τ to 12 hours considering the daily variation.

The peak value in I_k is represented by x_{p_k}, where p_k represents the peak time slot denoted as

$$p_k = \arg\max_{j \in I_k} x_j. \tag{2.17}$$

In this chapter, we extract the envelope by connecting the peak values $x_{p_1}, \cdots, x_{p_l}$ and the latest value $x_{p_{l+1}} = x_t$ with lines. By including the latest value x_t, the prediction can reflect the latest data. We simply perform the linear interpretation for points between $x_{p_{k-1}}$ and x_{p_k}, and each point is interpreted as

$$\begin{aligned} x_j &= x_{p_k} + \frac{x_{p_{k+1}} - x_{p_k}}{p_{k+1} - p_k}(j - p_k) \\ j &= p_k, p_k + 1, \cdots, p_{k+1}, \quad k = 1, \cdots, l. \end{aligned} \tag{2.18}$$

2.4.4 Range of Excluded Variation

The traffic variation excluded by the preprocessing should also be considered because it may cause the congestion. In this chapter, we consider the excluded traffic variation by using the standard deviation of the excluded traffic variation. The standard deviation is calculated as

$$\sigma = \sqrt{\frac{1}{h} \sum_{k=t-h+1}^{t} (x_k - x'_k)^2}, \tag{2.19}$$

where x_k is the original traffic rate on a flow at k-th time slot and x'_k is the extracted variation by preprocessing. Using σ, we compensate for the excluded variation in the predicted traffic with $\bar{x}_t + \beta\sigma$ where β is a parameter indicating the confidence level for the upper bound prediction of the excluded variations.

Finally, the upper bound prediction including both the prediction error and the excluded variation in the preprocessing can be calculated as $\hat{x}_i = \bar{x}_i + \alpha\hat{\sigma}_i + \beta\sigma$.

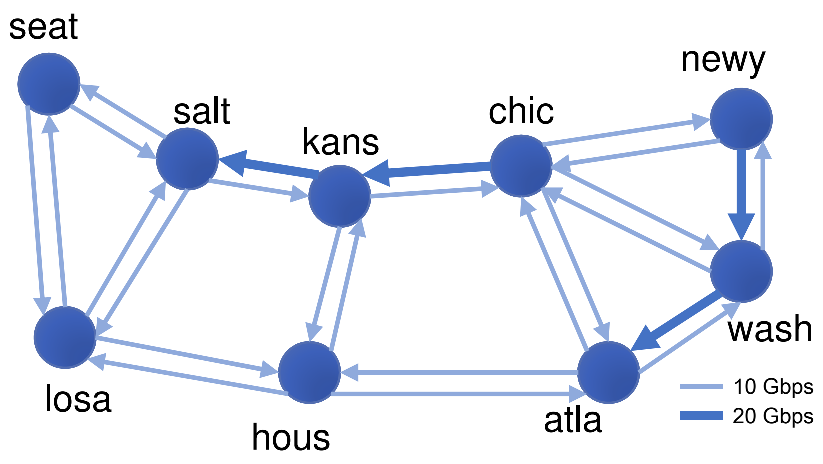


Figure 2.2: Internet2 topology

2.5 Evaluation

2.5.1 Datasets

We use actual traffic traces from the backbone network of Internet2 [58], a research and education network in the United States. Figure 2.2 shows its topology, and the capacity of each link is described in [59]. The traffic data are collected by a Netflow protocol at each of the nine PoP routers. The sampling rate is one packet in every 100 packets, and aggregated data are exported every five minutes. The sampling method has two main problems: it causes sampling errors, and there may be unsampled flows. However, it is not a critical problem for our evaluation because we only need the traffic rate of aggregated OD flow, which has a large number of samples. The large daily variation between day and night is mainly observed in the traffic variation. Focusing on such traffic variation over several hours, we set the length of the observation time slot to one hour and aggregate the observed data into the time slots.

We use four week's worth of data (11/28/2011 to 12/25/2011) aggregated into the flows between PoP routers using the BGP information. Table 2.1 summarizes the number of time slots used to train the traffic model, and number of time slots used to test the prediction accuracy or traffic engineering performance. We use the data from the previous two weeks as the observed data. We

Table 2.1: Number of Data Used in Training and Test series

control period [hours]	training series [hours]	test series [hours]
3	336	3
12	336	12
24	336	24

perform the preprocessing and prediction processes using these data, then compute optimal routes for the targeted control period using the predicted traffic. Finally, we evaluate these routes with actual traffic traces during the control period. We perform the above process 24 times, changing the start time of the prediction because traffic variation at the start of the prediction greatly affects its accuracy. Due to an over-provisioning policy [5], link utilization on the Internet2 network is less than 20%. Congestion rarely occurs in such situations, but this means that most of equipped capacity is redundant and unnecessary energy consumption is incurred. Our interest here is how to deal with congestion under limited resources in a way that reduces over-provisioning and power consumption costs, so we multiplied actual traffic amounts by 5 in the following evaluation.

2.5.2 Characteristics of the Traffic Prediction

2.5.2.1 Prediction Error

Before the evaluation of prediction based traffic engineering, we investigate the characteristics of the prediction method. First, to investigate accuracies of the prediction methods, we compare the *mean absolute percentage error (MAPE)*, defined as $\mathrm{MAPE}_k = \frac{1}{k}\sum_{i=t+1}^{t+k} \frac{|\bar{x}_i - x_i|}{x_i}$, where $\bar{x}_i$ is the predicted traffic rate, x_i is the actual traffic rate, and k is the length of the test series. This is one of the most frequently used metrics of prediction performance in previous work (e.g. [25, 27]).

Fig. 2.3 compares the MAPE corresponding to the length of the prediction target. In Fig. 2.3, “non-preprocess” means prediction using original data without preprocessing; “trend,” “envelope,” and “lowpass” mean prediction with each corresponding preprocessing; and “arima” and “sarima” mean prediction by the ARIMA and SARIMA models, respectively. Fig. 2.3 indicates that any traffic prediction includes prediction errors (e.g. at least around 40% in the case of “lowpass”). Fig. 2.3 also indicates that the MAPE generally increases as the prediction target becomes far from the current time slot except the case of “envelope”. This increase is caused by the accumulation of one step prediction errors. In the SARIMA and ARIMA models, the future traffic value is predicted by continuing the one step prediction. As a result, even if the prediction errors included in each step is small, the prediction errors in the far future become large by accumulating the prediction errors included in each step.

From Fig. 2.3, prediction with the envelope has the largest prediction error. This is because the “envelope” includes the large short-term fluctuation, since the upper bound of traffic is frequently changed by temporal traffic changes. It is difficult for SARIMA or ARIMA model to fit to the traffic pattern which includes such a large fluctuation. As a result, the prediction result with the envelope has large error even in one-step prediction. This large prediction errors also makes the MAPE of envelope independent from the time slot, while the MAPE of the other prediction methods increases as the time slot becomes far from the current time slot.

In Fig. 2.3, prediction with the lowpass filter achieves the lowest prediction error, because the lowpass filter effectively improves the prediction accuracy by excluding noisy variation. However, this result does not necessarily mean that prediction methods using the lowpass filter are best suited to traffic engineering. The MAPE indicates the overall accuracy of the prediction of all flows. However, for the traffic engineering, the importance of the prediction may depend on the flows; the prediction of the large flows may be important since the large flows have a large impact on the link utilization. We demonstrate the impact of the prediction on traffic engineering in Subsection 2.5.3.

2.5.2.2 Predicted Traffic Variation in Case of Daily Traffic Pattern

To investigate the detailed characteristic of each prediction method, we show the predicted traffic time series. As an example, Fig. 2.4 shows the prediction results of a flow using each preprocessing method without a confidence interval. In Fig. 2.4, “SARIMA” and “ARIMA” mean the prediction methods using the SARIMA model and the ARIMA model, respectively. Additionally, “real” means the actual traffic rate. Figs. 2.4(a)–(c) show the prediction results using the trend component, the lowpass filter, and the envelope. Fig. 2.4(d) shows the prediction results using original data without preprocessing.

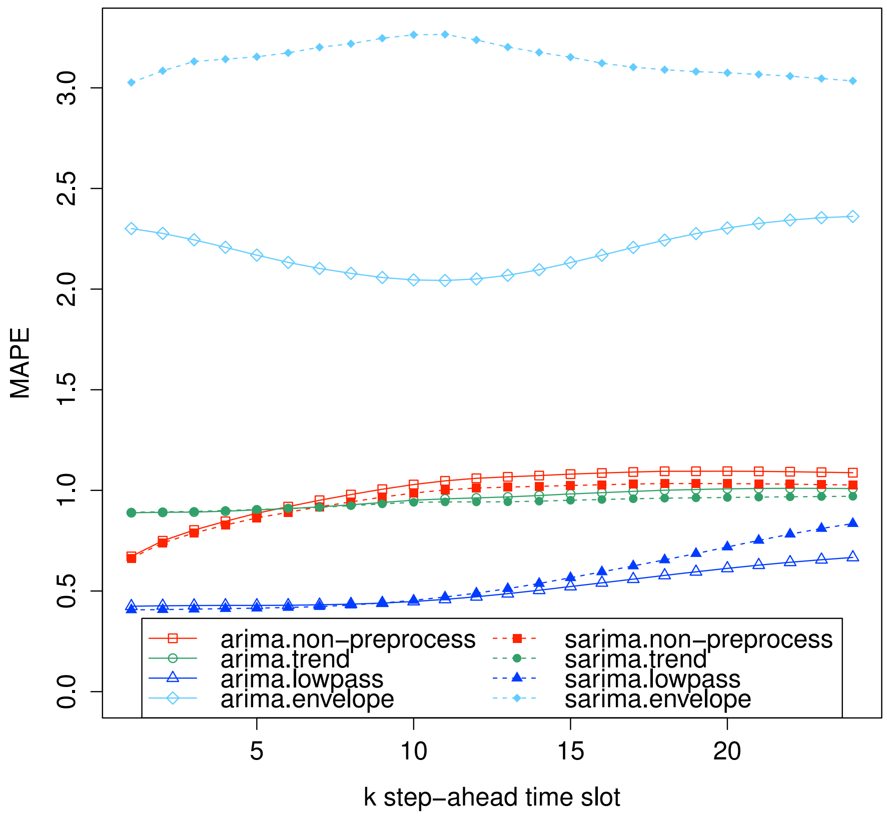


Figure 2.3: MAPE of each prediction method

Fig. 2.4 indicates that the preprocessing methods “trend” and “lowpass” improve the accuracy of the prediction of the daily variation. This is because the preprocessing excludes the noisy variation and clarifies the longer-term traffic variation, which enables accurate modeling of the daily traffic variation. Fig. 2.4 also indicates that the SARIMA model predicts the daily variation more accurately than the ARIMA model. This is because considering the periodicity in the prediction model is effective for predicting the daily variation. The results shown in Fig. 2.4 is different from those in Fig. 2.3. This is caused by that the prediction errors in the small flows; the MAPE is average of the prediction errors normalized by their actual values, and the prediction errors in the flows whose actual traffic amounts are small have significantly large impacts on the MAPE. In addition, the large prediction errors occur in the small flows, especially in the flows whose average traffic amounts are small but that have some spikes. Figure 2.5 shows an example of such small flows with spikes. In this figure, the vertical dotted line indicates the start point of the prediction. In this figure, there is a spike before the start point of the prediction. Such spikes cannot completely extracted by the trend or lowpass filter, and have a impact on the extracted long-term tendency. In the case of Fig. 2.3, the spike causes the sudden increase and decrease in the extracted tendency just before the start point of the prediction. As a result, the SARIMA model whose parameters are set to fit such sudden changes becomes different from the long-term trend of the traffic, and causes a large prediction error.

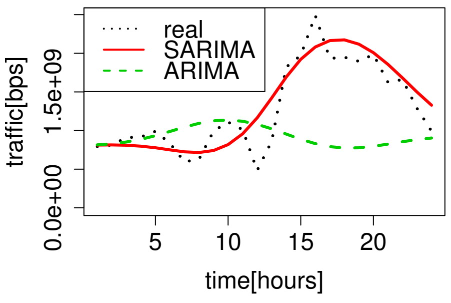
However, such spikes causing the large prediction errors are not found in the large flows. This is because the large flow includes a numerous number of user flows. The spikes in the flows are caused by the spikey behavior of the user flows. However, even if the flow includes the user flows whose behaviors are spikey, the spikey flows have only small impacts on the total traffic amounts of the flow, when the flow includes a large number of user flows.

Considering the traffic engineering, the prediction of the large flows such as a flow shown in Fig. 2.4 are important, compared with the small flows, since the large flows have a large impact on the link utilizations. Thus, the evaluation of the accuracy of the prediction is not sufficient, and we need the evaluation of the performance of the traffic engineering using the predicted traffic, which is discussed in Section 2.5.3.

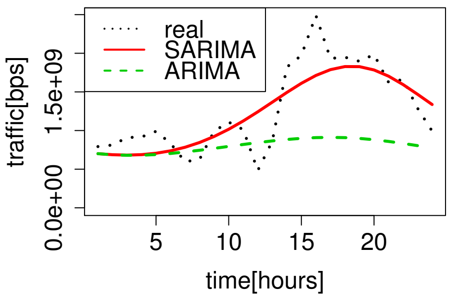
2.5.2.3 Predicted Traffic Variation in Case of Sudden Traffic Change

Though we do not need the accurate prediction on the spikey flows with small average traffic rates, the large flow may have the traffic variation which suddenly deviate from the longer-term pattern. Since the large flow affects the performance of the traffic engineering, the prediction should follow the main pattern of variation even in this case.

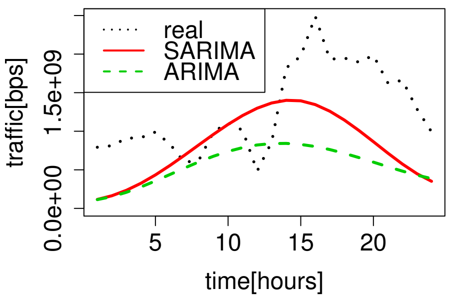
In this subsection, we investigate the accuracy of the prediction with a lowpass filter and trend component when such sudden change occurs in the large flows. Fig. 2.6 shows the prediction results of the SARIMA and ARIMA of a flow when the sudden traffic change is included. Fig. 2.6 plots the actual traffic variation and the predicted variation. In this figure, we plot the prediction results



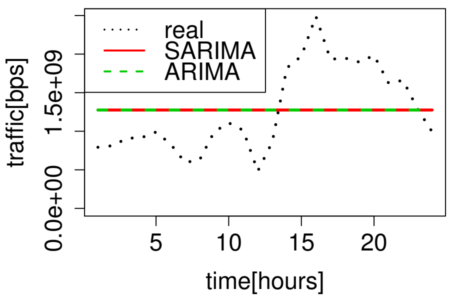
(a) Trend component



(b) Lowpass filter



(c) Envelope



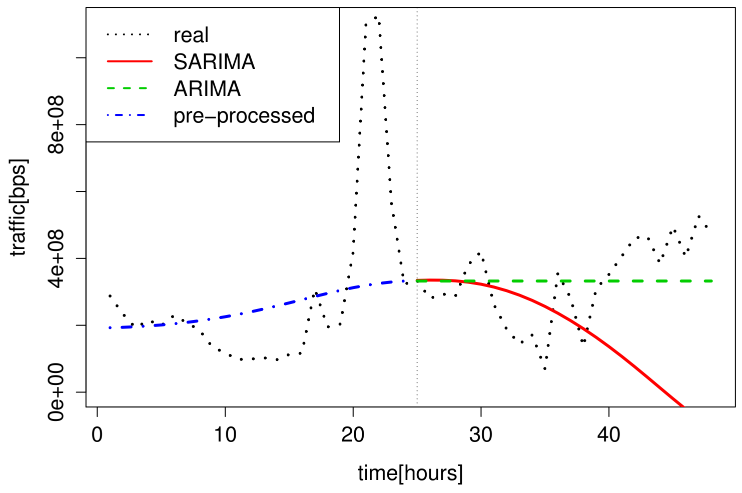
(d) Non-preprocessing

Figure 2.4: Example of the predicted traffic time series using each preprocessing method

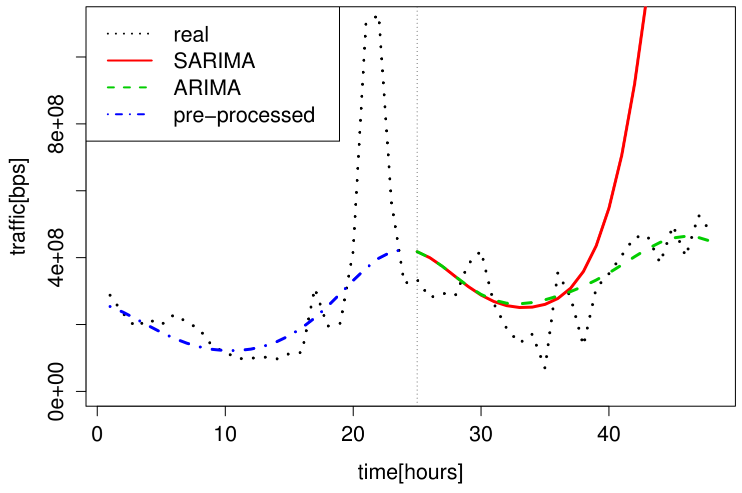
of two prediction methods (lowpass and trend) that can accurately predict the daily traffic variation as discussed in the previous subsection. The vertical dotted line indicates the start point of the prediction. “upper lowpass” and “upper trend” indicates the upper bound calculated by setting α and β to 0.84, which correspond to the confidence level of 80% for prediction error and short-term variation, respectively.

Unlike the spikey flows with small average traffic rates, the large flow, whose traffic rates suddenly increase, increases over multiple time slots as shown in Fig. 2.6. Thus, we can obtain the information used for the prediction of the sudden increase.

In Fig. 2.6, the method using the trend component follows the main variation under sudden traffic change more accurately than the method using the lowpass filter. This is because the trend



(a) Trend component



(b) Lowpass filter

Figure 2.5: Example of failure prediction with SARIMA

component extracts the tendency to increase, while the lowpass filter removes all the variation shorter-term than daily variation. As a result, the lowpass filter removes the increasing tendency and underpredicts the sudden increase in traffic.

Fig. 2.6 also indicates that the predicted traffic of the ARIMA and SARIMA are almost the same. This is because the periodicity of the traffic variation is not effective for such sudden variation.

2.5.3 Performance of the Traffic Engineering

In this subsection, we investigate the performance of the traffic engineering using the predicted traffic. In this evaluation, we compute the optimum routes by solving the linear programming problem in Eqs. (2.3–2.7) using the predicted traffic. The linear programming problem is solved by CPLEX [60]. After the calculated routes are set, we investigate the performance of traffic engineering using actual traffic with the calculated routes.

To evaluate the performance of traffic engineering, we investigate the link load of each link at each slot, which are the sum of traffic passing the link. In this evaluation, we focus on the peak link loads during the control period, because the network operator should set the bandwidth of each link so as to accommodate peak traffic without congestion. Among all links, we also focus on the most congested link, because the reduction of the link load on the most congested link is one of important objectives in the traffic engineering. Since the most congested link is passed by a large number of flows, the mitigation of the congestion of such a link improves the performance of a large number of flows. In addiction, the reduction of the link load on the most congested link avoids the concentration of traffic on a certain link, which may cause the necessity of enhancement of the link capacities. Thus, we use the maximum peak link loads defined by

$$r = \max_{l,k\in[t+1,t+f]} y_k(l) \tag{2.20}$$

where f is the length of control period, and $y_k(l)$ is the traffic rate on the link l at the time slot k. The small r indicates that we do not require a large bandwidth to accommodate traffic without congestion.

In the evaluation, we normalize the value of r by that of *InvCap routing*, which is the most commonly used method for load balance routing. InvCap routing calculates the shortest path using the inverse of link capacities as weights, splitting the traffic equally among equal weighted paths. The normalized maximum peak link load r' is defined as

$$r' = \frac{r}{r_{\text{InvCap}}} \tag{2.21}$$

where $r_{\text{InvCap}} = \max_{l,k} y_k^{\text{InvCap}}(l)$ is the maximum peak link load under the InvCap routing, and $y_k^{\text{InvCap}}(l)$ is the traffic rate on the link l at the time slot k under the routes determined by InvCap routing. In our evaluation, we focus on the largest value of r' to clarify the reduction in the required bandwidths to avoid congestion.

2.5.3.1 Impact of Considering the Short-Term Variation and Prediction Errors

We compare the maximum peak link load by the traffic engineering using the predicted traffic with various α and β. Figures 2.7–2.8 show the complement cumulative distribution function (CCDF) of the normalized maximum peak link load. Figures 2.7–2.8 show the cases of SARIMA and ARIMA with the trend component for various control periods. In this comparison, when changing α, β is set to 0. On the other hand, when changing β, α is set to 0. Here, "mean" indicates the result using mean prediction without confidence interval, and "k %" means that confidence level corresponds to $k\%$. The confidence interval corresponding to a confidence level is calculated under the assumption that predictive error follows a Gaussian distribution. This assumption can be examined by a Kolmogorv-Smirov (KS) test, and the null hypothesis of Gaussian distribution cannot be rejected at a significance level of 5% for more than 85% of OD flows.

In most cases in Figs. 2.7–2.8, the largest link load of prediction-based traffic engineering is improved by considering the confidence level. This is because by considering the range of the short-term variation and prediction errors, the congestion occurred by temporal traffic variation can be avoided. When these ranges are not considered, temporal traffic variation sometimes causes congestion. Moreover, the difference between considering confidence intervals or not becomes large when the control period is large. This is because a large control period has a higher possibility

Table 2.2: Values of Parameters for Confidence Levels (α,β)

control period [slots]	3	12	24
trend	(0.5,0.6)	(0.5,0.8)	(0.9,0.8)
lowpass	(0.8,0.5)	(0.7,0.8)	(0.8,0.9)
envelope	(0.8,-)	(0.9,-)	(0.8,-)
non-preprocess	(0.7,-)	(0.7,-)	(0.8,-)

of temporal traffic changes which may causes the congestion.

From Figs. 2.7–2.8, an overly large α sometimes requires large capacity, while the maximum peak link load is kept small even when β is set to a large value. When α is set to a large value, the predicted traffic rate of the distant future time slots becomes large because the traffic of the distant future time slot is difficult to predict and the variance of the prediction becomes large. As a result, too many resources are allocated to the traffic whose variance of the prediction is large. On the other hand, the variance of the short-term traffic variation is constant for all time slots in our prediction procedure. Thus, even when β is set to a large value, no traffic is predicted as a too large value. Therefore, setting β to a large value and α to 0 is sufficient to avoid future congestion caused by short-term variation.

2.5.3.2 Comparison of the Preprocessing Methods

We compare the impacts of the preprocessing methods on the traffic engineering using the predicted traffic. Hereafter, we configure the confidence levels (α and β) of each prediction method in traffic engineering so that the maximum link load at the peak time slot is minimized. Table 2.2 shows the configured values of (α,β). For “envelope” and “non-preprocess”, the value of β is not valid because it makes no sense to consider the removed variance in preprocessing in these methods.

Figs. 2.9–2.10 show the CCDF of normalized maximum peak link load at each control period when the traffic is predicted by the SARIMA or ARIMA with each preprocessing. Here, “observation-based TE” means calculating routes using the previous one hour’s worth of data instead of the predicted traffic.

Figs. 2.9–2.10 show that the traffic engineering with the prediction keeps maximum peak link load low for the worst or almost worst case than “observation-based TE”. This is because the traffic engineering using the predicted traffic variation sets the routes so as to avoid the future congestion by considering the future traffic variation. On the other hand, the “observation-based TE” sets the routes on the basis of the observed traffic, which sometimes differs from the future traffic significantly. As a result, the “observation-based TE” causes the congestion on a certain link.

Comparing the results of the different control periods, the maximum peak link loads increases as the control period becomes large. This is because the traffic changes included in the control period increases as the control period becomes large. As a result, more bandwidth is required to accommodate the traffic fluctuation during the control period. However, the short control period causes frequent route changes. In addition, the routing optimization may take a long time, the control period cannot be set to a small value especially in a large network. Even in the case of the long control period, the prediction based TE does not required a large bandwidth, compared with the observation based TE, which is one of the important advantage of the prediction based TE.

In Fig. 2.9, the SARIMA method with the trend keeps the worst value of link load small compared with the other methods. This is caused by that the SARIMA with the trend follows both of the long-term variation and sudden changes. As a result, traffic engineering using SARIMA with the trends allocates sufficient resources considering the long-term variation and sudden changes.

We also investigate the gain of the prediction based TE compared with “observation-based TE”, defined by

$$1 - \frac{r}{r_{\text{observation}}}$$

where r is the maximum peak link load of the prediction based TE, $r_{\text{observation}} = \max_{l,k} y_k^{\text{observation}}(l)$ is that of the observation based TE, and $y_k^{\text{observation}}(l)$ is the traffic rate on the link l at the time slot k under the routes determined by observation based TE. Figure 2.11 shows the performance gain of each control period when the control period is set to 12 slots. In Fig. 2.11, the maximum, third quartile, median, first quartile, and minimum values are plotted as horizontal line from top to bottom, and the average value is plotted as a crossed point. Similar to the previous results, we focus on the worst case to evaluate the reduction of capacity which must be prepared. Although there is difference among the prediction method, the worst case of gain is positive in all methods. Especially, the gain of the prediction based traffic engineering using SARIMA with the trend component is at least 18.9%. That is, the prediction based traffic engineering using SARIMA with trend component reduces the required bandwidth by 18.9% compared with the observation based traffic engineering.

2.5.3.3 Comparison of the ARIMA and SARIMA Models

We also compare the performance of the traffic engineering using the traffic predicted by the ARIMA and SARIMA models. Fig. 2.12 compares the CCDF of maximum peak link load normalized by InvCap routing. In Fig. 2.12, we present the results for the traffic engineering using the traffic predicted by the ARIMA/SARIMA with the trend component or lowpass filter, and the observation-based traffic engineering.

Fig. 2.12 indicates that the traffic engineering using the traffic predicted by the ARIMA keeps maximum peak link load similar in size to that of the traffic engineering using the traffic predicted by the SARIMA when the control period is small. This is because the traffic variation of the short control period can be predicted even without considering the periodicity of the traffic variation.

On the other hand, the SARIMA method achieves lower maximum peak link load than the ARIMA when the control period is 24 slots. Because the longer-term traffic variation cannot be predicted without considering the periodicity, the prediction errors of the ARIMA become large. On the other hand, the SARIMA predicts the longer-term variation accurately by considering the periodicity. As a result, the traffic engineering using the traffic predicted by the SARIMA allocates the resources to the traffic properly.

We also compare the computational complexity of the ARIMA and SARIMA. The computational complexity of the prediction with ARIMA and SARIMA is $O(m^3t)$ where m is the oldest time slot in the model and t is the length of the data used to learn the parameters of the model. $m = max(sP, sQ) + max(p, q)$ in the SARIMA model and $m = max(p, q)$ in the ARIMA model. In the case of datasets used in our evaluation, the period length s equals 168, the number of period terms P or Q usually equals 1, and p or q usually equals 3–5. Therefore, the value of m in the SARIMA model is about 30–60 times larger than the ARIMA model. Thus, the ARIMA predicts the traffic about 38,000–180,000 times faster than the SARIMA.

Therefore, the ARIMA prediction is useful for the traffic engineering targeting the short time scale. This kind of the traffic engineering may require the future traffic variation to be frequently recalculated, and the ARIMA predicts the traffic quickly. In addition, the ARIMA predicts the traffic variation with sufficient accuracy to avoid the congestion during the short control periods.

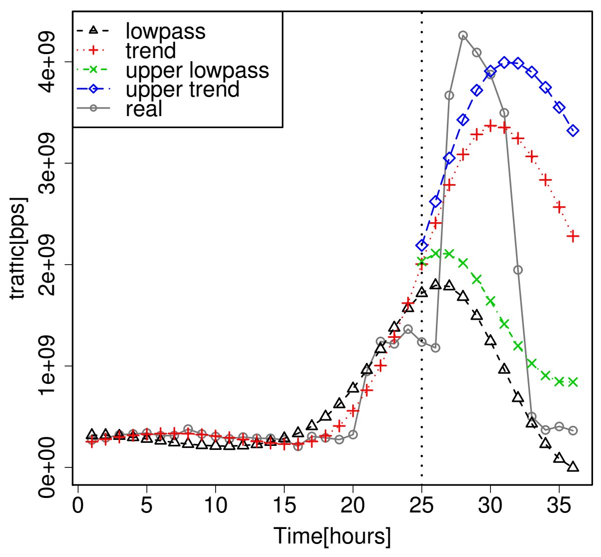
On the other hand, the SARIMA is required when we aim to calculate the stable routes for a long control period. By considering the longer-term traffic variation, we may handle even unexpected traffic changes by changing routes of only a small amount of traffic corresponding to the unexpected changes. As a result, we keep the network stable.

To achieve this, we must predict both the longer- and short-term traffic variations accurately. The SARIMA predicts the longer-term variation accurately, while the ARIMA cannot. Though the SARIMA takes a long time to predict the traffic, the future traffic does not need to be frequently recalculated when the target control period is large.

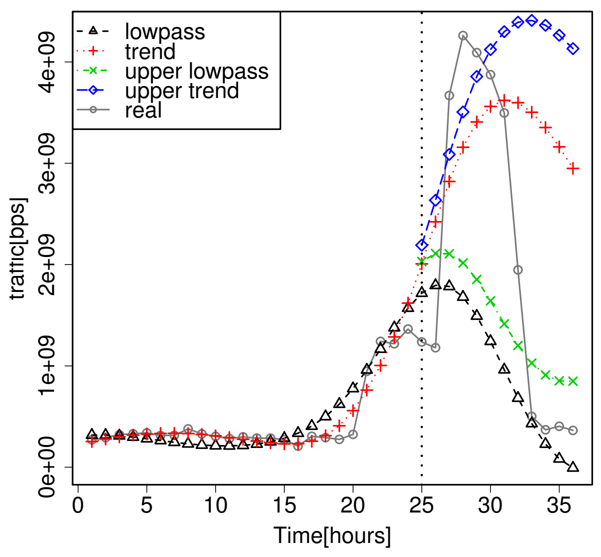
2.6 Conclusion

In this chapter, we proposed a traffic prediction procedure that obtains all the information required for traffic engineering. In our prediction procedure, we extract the longer-term variation before the prediction so as to improve the prediction accuracy of the daily traffic variation. The short-term traffic variation is also handled by calculating the variance of the traffic variation excluded by the preprocessing. Through the simulation, we clarified that the results of traffic engineering using the predicted traffic show that considering the short-term variation and prediction errors avoids the congestion caused by the prediction uncertainty. The results also indicate that the ARIMA model is suitable for the traffic engineering method targeting the short-term control period and the SARIMA model is suitable for the longer-term control period.

Our future work will include further investigation of more sophisticated prediction models such as neural networks, and developing traffic engineering methods suitable for use with predicted traffic.

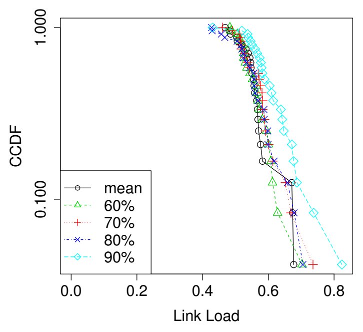


(a) SARIMA prediction

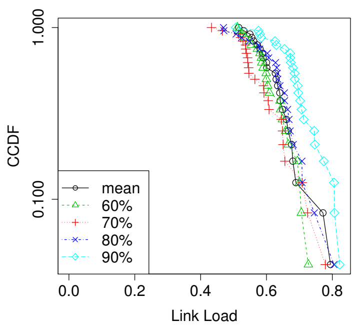


(b) ARIMA prediction

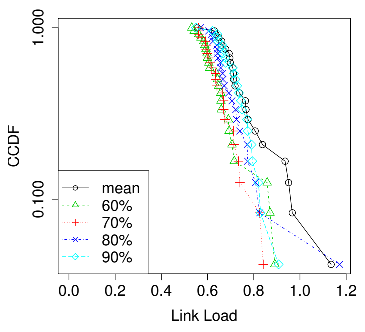
Figure 2.6: Example of the prediction for the sudden traffic change



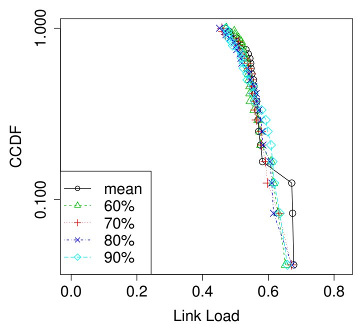
(a) Different α levels (control period: 3 slots)



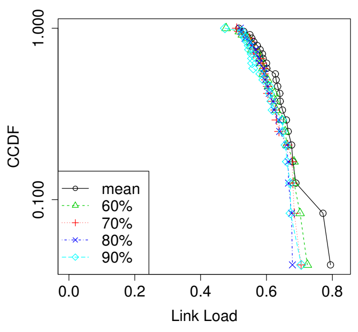
(b) Different α levels (control period: 12 slots)



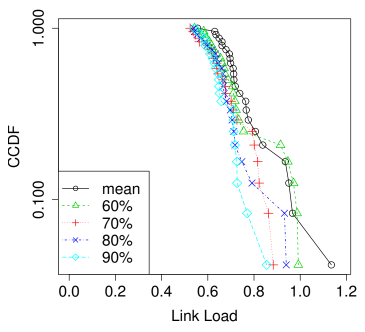
(c) Different α levels (control period: 24 slots)



(d) Different β levels (control period: 3 slots)

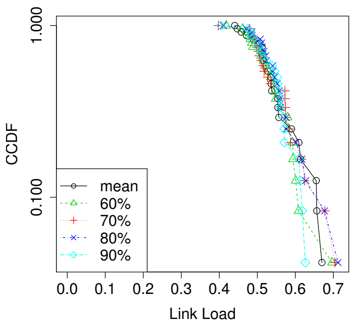


(e) Different β levels (control period: 12 slots)

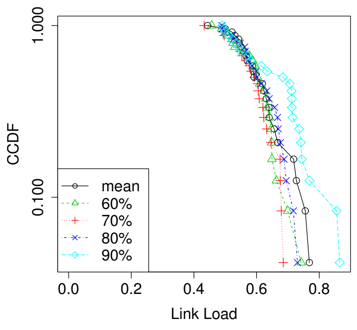


(f) Different β levels (control period: 24 slots)

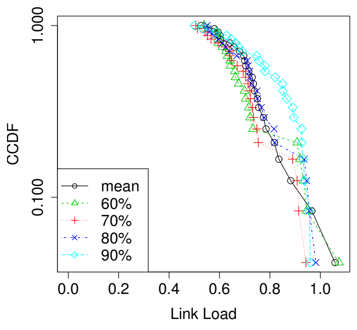
Figure 2.7: Complement Cumulative distribution of maximum peak link load normalized by InvCap routing with different confidence levels in the SARIMA model prediction with the trend component



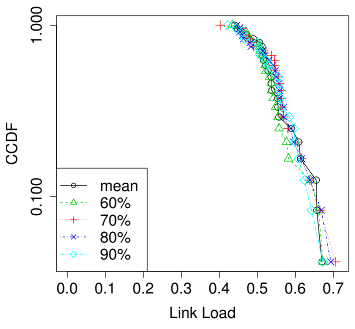
(a) Different α levels (control period: 3 slots)



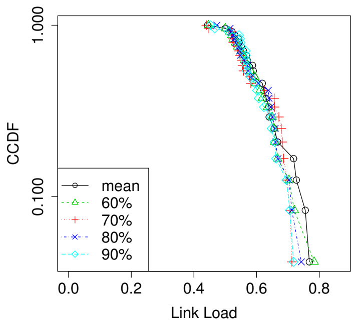
(b) Different α levels (control period: 12 slots)



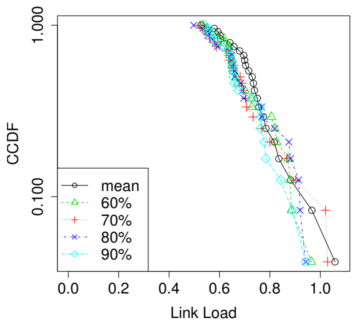
(c) Different α levels (control period: 24 slots)



(d) Different β levels (control period: 3 slots)

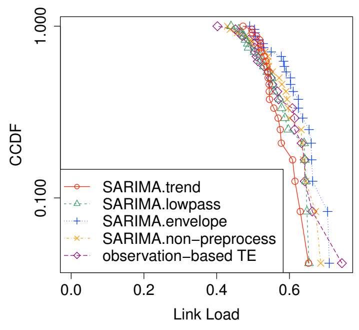


(e) Different β levels (control period: 12 slots)

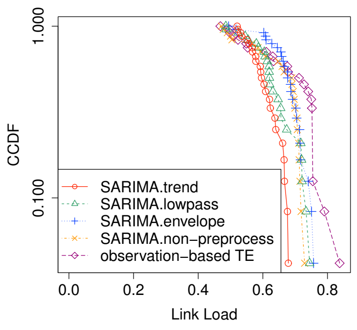


(f) Different β levels (control period: 24 slots)

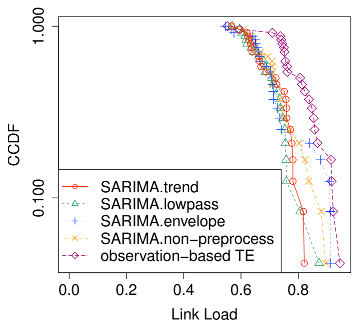
Figure 2.8: Complement Cumulative distribution of maximum peak link load normalized by InvCap routing with different confidence levels in the ARIMA model prediction with the trend component



(a) Control period: 3 slots

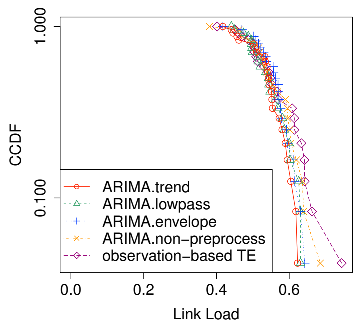


(b) Control period: 12 slots

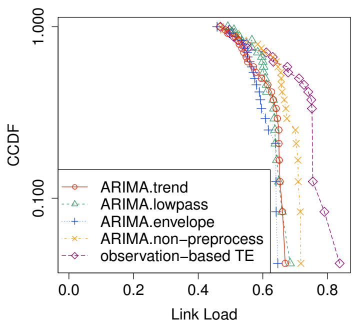


(c) Control period: 24 slots

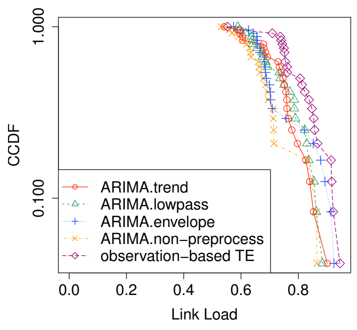
Figure 2.9: Complement Cumulative distribution of maximum peak link load normalized by InvCap routing when the SARIMA model prediction with the different preprocessing is used in traffic engineering



(a) Control period: 3 slots



(b) Control period: 12 slots



(c) Control period: 24 slots

Figure 2.10: Complement Cumulative distribution of required maximum peak link load normalized by InvCap routing when the ARIMA model prediction with the different preprocessing is used in traffic engineering

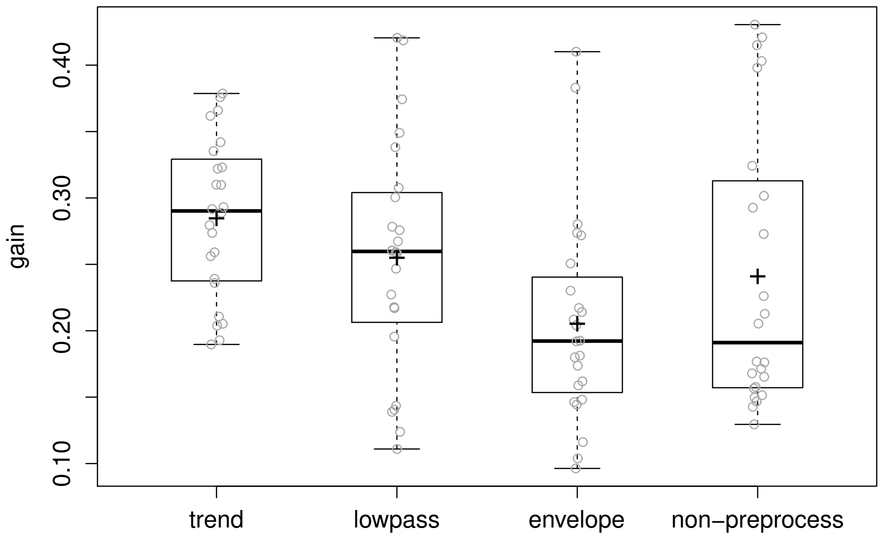
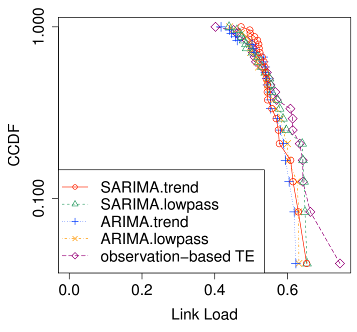
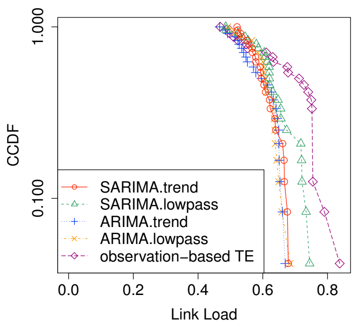


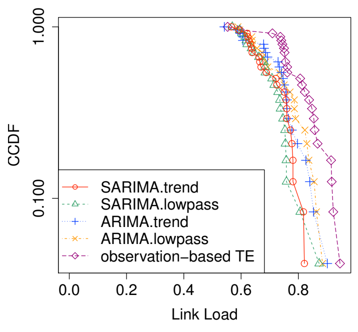
Figure 2.11: Box plot of gain of prediction-based traffic engineering in reduction of maximum peak link load compared with the observation-based traffic engineering (Control period: 12slots)



(a) Control period: 3 slots



(b) Control period: 12 slots



(c) Control period: 24 slots

Figure 2.12: Complement Cumulative distribution of maximum peak link load normalized by InvCap routing when the ARIMA and SARIMA model predictions with the different preprocessing are used in traffic engineering

Chapter 3

Traffic Engineering Method Based on Model Predictive Control

3.1 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* [6–8] 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 [8], 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 [9, 11–13]. 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 [20]. 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 calculated 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 [21, 23, 24, 51] up to daily [25, 26] and even monthly or yearly long-term variation [22, 61]. Traffic prediction that considers both daily and short-term variation has also been proposed for TE [29]. 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 chapter, 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) [34, 35], 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 chapter, 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 chapter as follows. (i) This chapter proposes a new prediction-based TE method, which is robust to the prediction errors by applying the idea of MPC. (ii) This chapter demonstrates the advantage of our TE method by simulation using the actual traffic trace. (iii) This chapter discusses the suitable parameter setting of our TE method.

The rest of this chapter is organized as follows. Section 3.2 describes TE and TE with traffic prediction. Section 3.3 describes our TE method, to which we apply MPC. Section 3.4 presents an evaluation of basic behavior in our TE method. Section 3.5 gives an evaluation of our TE method as applied to an actual backbone network. Section 3.6 surveys related work. Section 3.7 presents our conclusions.

3.2 Dynamic Traffic Engineering and Traffic Prediction

3.2.1 Dynamic 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 [9, 12], 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

$$\hat{\boldsymbol{y}}(t+1) = G \cdot R(t+1) \cdot \boldsymbol{x}(t) \tag{3.1}$$

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:

$$minimize \quad : \quad f(\hat{\boldsymbol{y}}(t+1)) \tag{3.2}$$

$$subject\ to \quad : \quad \hat{\boldsymbol{y}}(t+1) = G \cdot R(t+1) \cdot \boldsymbol{x}(t). \tag{3.3}$$

The most used cost function is maximum link utilization [9, 12] 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 [10, 13, 15]. 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.

3.2.2 Dynamic TE with Traffic Prediction

Traffic prediction is useful for TE to prevent route change delays due to differences between actual and observed traffic. Fig. 3.1 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

$$\hat{x}(t+1) = F(x(1), \cdots, x(t)), \tag{3.4}$$

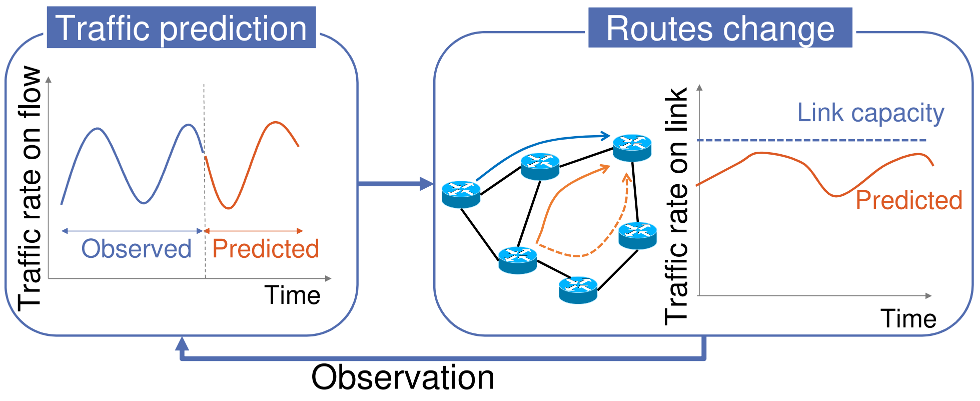


Figure 3.1: Overview of traffic engineering with traffic prediction

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

$$\hat{\boldsymbol{y}}(t+1) = R(t+1)\hat{\boldsymbol{x}}(t+1). \tag{3.5}$$

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:

$$minimize \quad : \quad f(\hat{\boldsymbol{y}}(t+1), \cdots, \hat{\boldsymbol{y}}(t+h)) \tag{3.6}$$

$$subject\ to \quad : \quad \hat{\boldsymbol{y}}(k) = R\hat{\boldsymbol{x}}(k), k = t+1, \cdots, t+h. \tag{3.7}$$

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

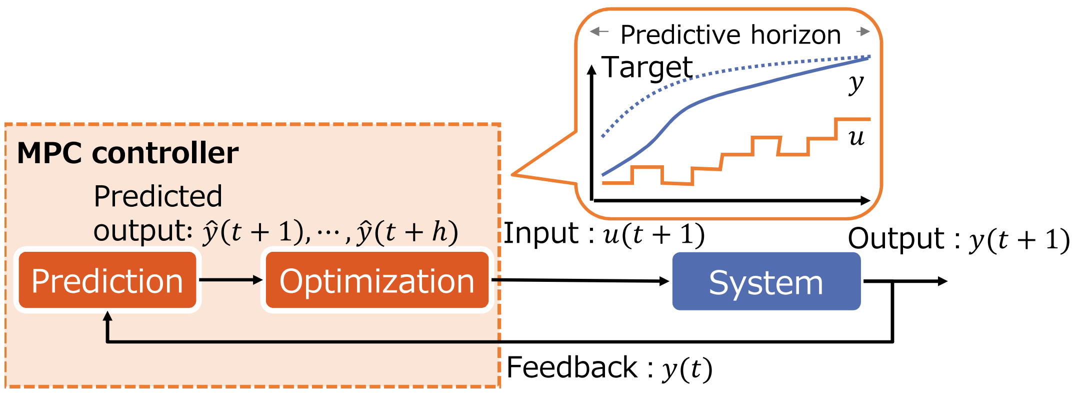


Figure 3.2: Overview of MPC

3.3 Traffic Engineering Based on MPC

3.3.1 Overview of MPC

MPC is a method of system control based on predictions of system dynamics; this has been studied in recent years. Fig. 3.2 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:

$$(u(t+1), \cdots, u(t+h)) = \underset{(u(t+1),\cdots,u(t+h))}{\arg\min} \; J_1. \tag{3.8}$$

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

$$z(k+1) = \phi(k, z(k), u(k)) \tag{3.9}$$

$$y(k) = \psi(k, z(k), u(k)), \tag{3.10}$$

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. (3.8), the controller calculates the input values by the following optimization problem:

$$(u(t+1), \cdots, u(t+h)) = \underset{(u(t+1),\cdots,u(t+h))}{\arg\min}\ (1-w)J_1 + wJ_2 \tag{3.11}$$

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

3.3.2 Applying MPC to TE

We apply MPC to TE to achieve a prediction-based TE that is robust against prediction errors. Fig. 3.3 shows an overview of our TE method, to which MPC is applied. We assume that a control

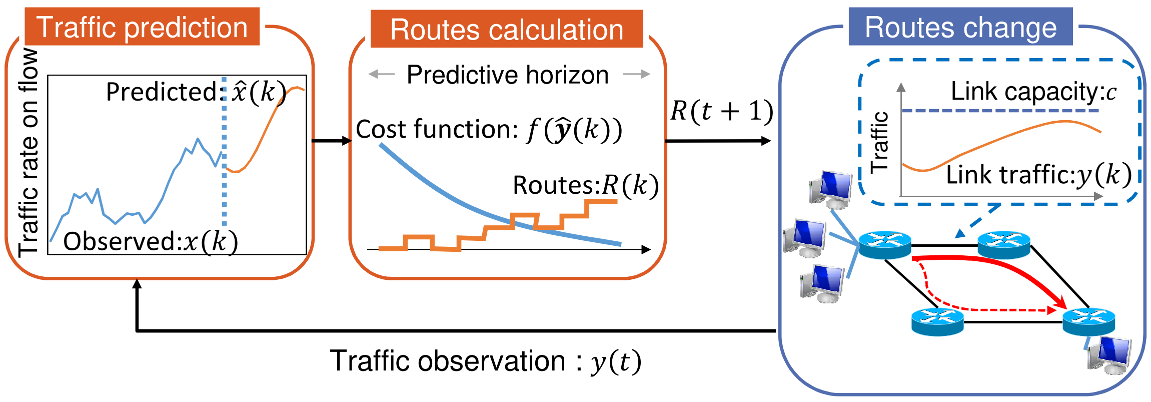


Figure 3.3: Overview of traffic engineering based on MPC

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.

3.3.2.1 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 chapter. 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.

3.3.2.2 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:

$$\begin{aligned} &minimize &:& \sum_{k=t+1}^{t+h} \Big((1-w)f(\hat{\boldsymbol{y}}(k)) + w\|\Delta R(k)\|^2\Big) && (3.12)\\ &subject\ to &:& \forall k, \hat{\boldsymbol{y}}(k) = G \cdot R(k) \cdot \hat{\boldsymbol{x}}(k) && (3.13)\\ &&& \forall k, \forall i, \forall j, R_{i,j}(k) \in [0,1] && (3.14)\\ &&& \forall k, \forall j, \sum_{i \in \wp(j)} R_{i,j}(k) = 1. && (3.15) \end{aligned}$$

Here, $\hat{\boldsymbol{x}}(k), G$ are given variables and $R(k), \hat{\boldsymbol{y}}(k)$ are the variables to be optimized. Eq. (3.13) represents the relation between the traffic rates of the OD flows and links. Eqs. (3.14) and (3.15) 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.

3.4 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

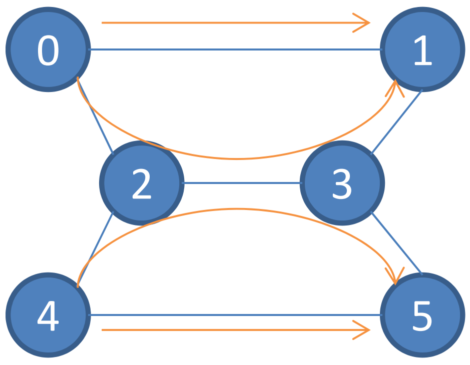


Figure 3.4: Simple network topology

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.

3.4.1 Simulation Environment

3.4.1.1 Network Topology

We use the simple network topology shown in Fig. 3.4. 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. 3.4, 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.

3.4.1.2 Network Traffic

We use the artificial traffic shown in Fig. 3.5. This artificial traffic includes traffic increases and decreases, which will cause congestion unless the routes are appropriately changed.

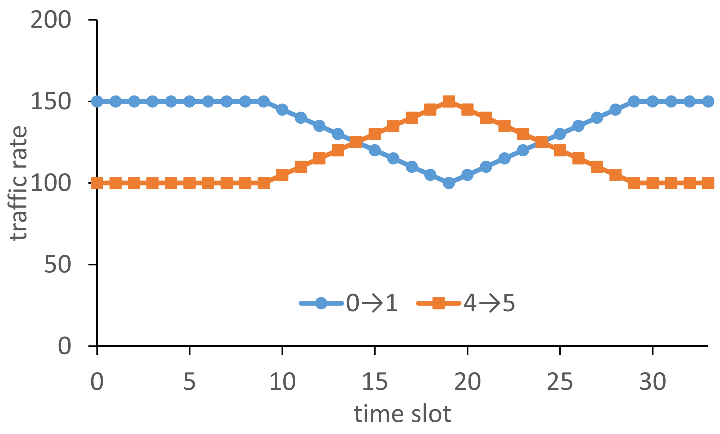


Figure 3.5: Network traffic for simple network topology

3.4.1.3 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 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.

3.4.1.4 Cost Function

In this evaluation, we use a cost function that is based on link utilization, which is similar to existing work [62]. 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

$$\zeta_j(k) = \max_{i \in \mathscr{P}(j)} [y_i(k) - c_i]^+ / n_i, \tag{3.16}$$

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

$$\zeta'_j(k) = \zeta_j(k) / \max_i c_i \tag{3.17}$$

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.

3.4.1.5 Route Calculation

Though the congestion level defined by Eq. (3.17) is non-linear, 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:

$$\begin{aligned}
minimize \quad &: \quad \sum_{k=t+1}^{t+h} ((1-w)\|\boldsymbol{\zeta}'(k)\| + w\|\Delta R(k)\|) && (3.18)\\
subject\ to \quad &: \quad \forall k, \forall p, \forall l \in \mathscr{P}(p), n_l \zeta_p'(k) \geq \frac{\alpha_l(k)}{\max c_l} && (3.19)\\
& \quad \forall k, \forall l, \alpha_l(k) = \hat{y}_l(k) - c_l(k) + S_l(k) && (3.20)\\
& \quad \forall k, \forall l, \alpha_l(k) \geq 0 && (3.21)\\
& \quad \forall k, \forall l, S_l(k) \geq 0 && (3.22)\\
& \quad \forall k, \hat{\boldsymbol{y}}(k) = G \cdot R(k) \cdot \hat{\boldsymbol{x}}(k) && (3.23)\\
& \quad \forall k, \forall i, \forall j, R_{i,j}(k) \in [0,1] && (3.24)\\
& \quad \forall k, \forall j, \sum_{i \in \wp(j)} R_{i,j}(k) = 1. && (3.25)
\end{aligned}$$

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. (3.18)–(3.25), we use the CPLEX [60] 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.

3.4.1.6 Compared Method

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.

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.

3.4.2 Simulation Results

3.4.2.1 Congestion Level

Fig. 3.6 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. 3.6 indicates the advantages of MPC-based TE. In Fig 3.6, 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, 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

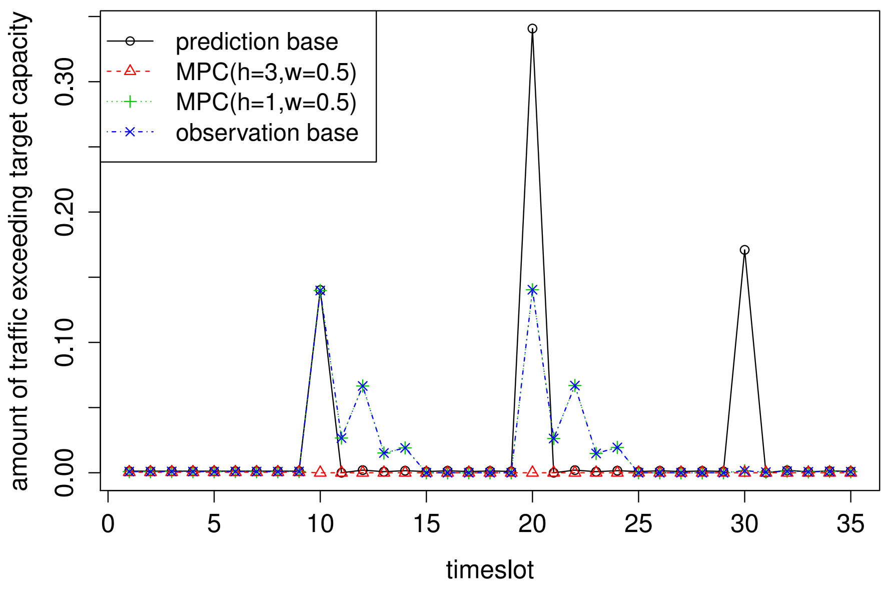


Figure 3.6: Traffic exceeding target link capacity in a simple network

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.

3.4.2.2 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. 3.7 and 3.8 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. 3.7 and 3.8, 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 3.5, 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.

3.5 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 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

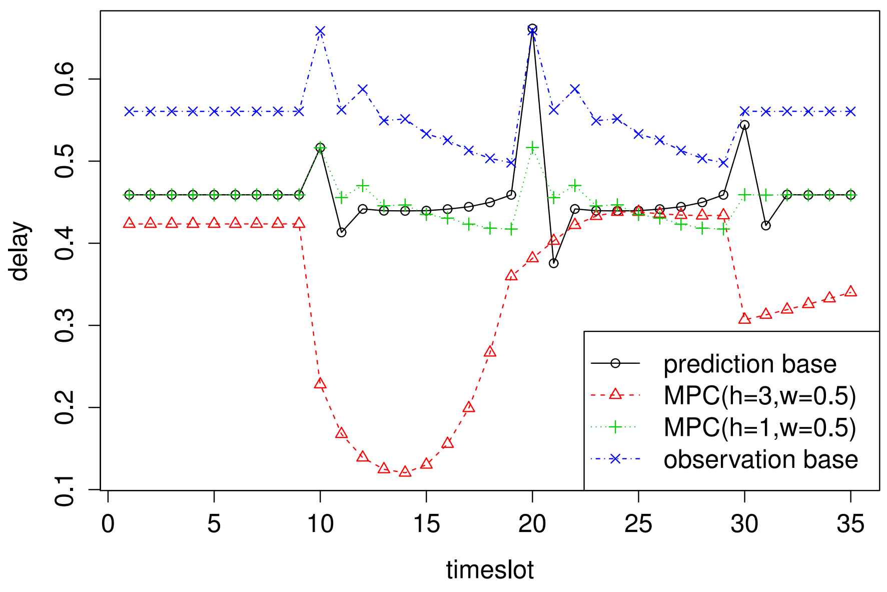


Figure 3.7: Average end-to-end delay of all OD flows in a simple network

way to Section 3.4.

3.5.1 Simulation Environment

3.5.1.1 Network Topology

In this subsection, we use an actual Internet2 backbone network, shown in Fig. 3.9. 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.

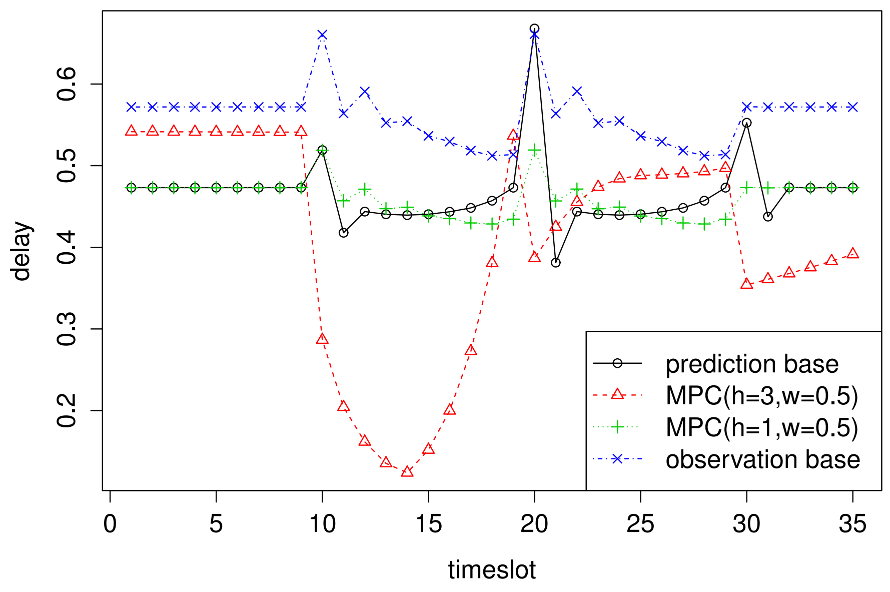


Figure 3.8: Maximum end-to-end delay of all OD flows in a simple network

3.5.1.2 Network Traffic

Here, we use actual traffic traces [58]. 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 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

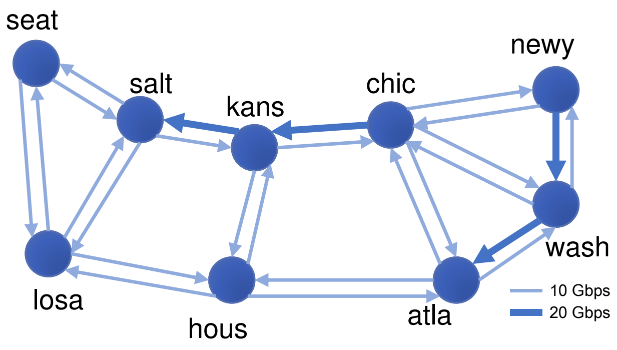


Figure 3.9: Internet2 topology

flow are t_s and t_e, the amount of traffic during a certain period τ is calculated as

$$x = \frac{\theta}{t_e - t_s}\tau \tag{3.26}$$

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

$$\tau = \begin{cases} t_{k+1} - t_s & (t_k < t_s \wedge t_{k+1} < t_e) \\ t_e - t_s & (t_k < t_s \wedge t_{k+1} \geq t_e) \\ t_{k+1} - t_k & (t_k > t_s \wedge t_{k+1} < t_e) \\ t_e - t_k & (t_k > t_s \wedge t_{k+1} \geq t_e) \\ 0 & (otherwise). \end{cases} \tag{3.27}$$

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. 3.10.

3.5.1.3 Prediction Method

We use the same prediction method that was used in Section 3.4.

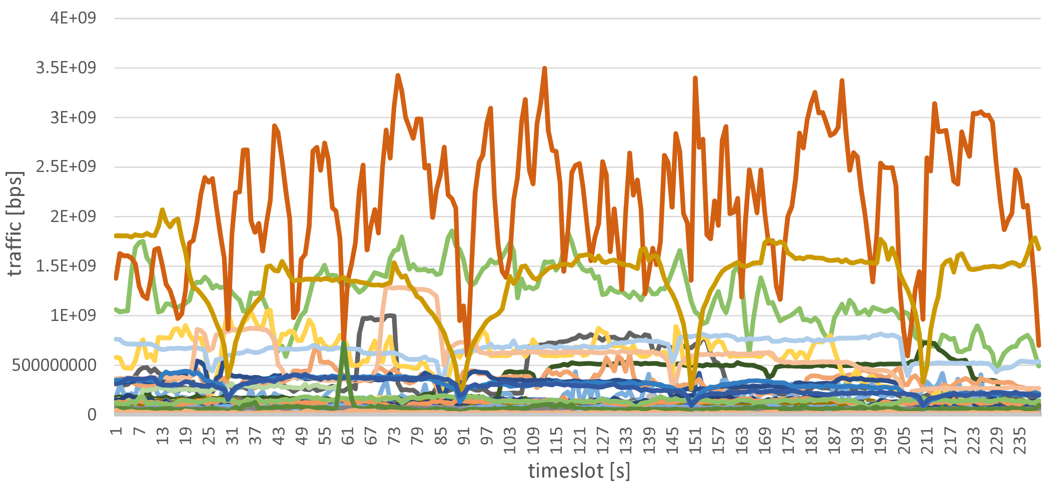


Figure 3.10: Internet2 network traffic

3.5.1.4 Cost Function

We use the same cost function that was used in Section 3.4.

3.5.1.5 Calculation of Routes

As in Section 3.4, we use CPLEX [60] to calculate the routes. In this evaluation, the optimization is finished within one second when $h = 3$.

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.

3.5.2 Simulation Results

Fig. 3.11 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. 3.6, 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. 3.11, 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.

3.5.3 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.

3.5.3.1 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,

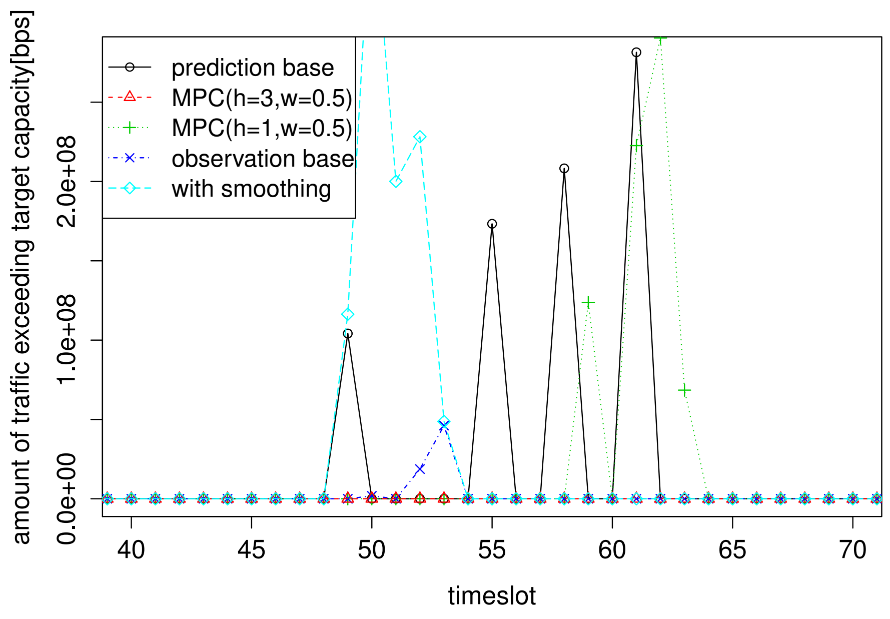


Figure 3.11: Traffic exceeding target link capacity with actual Internet2 traces

we may have to consider a trade-off between the robustness and sensitivity to set an appropriate value of w.

Figure 3.12 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. 3.12, 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.

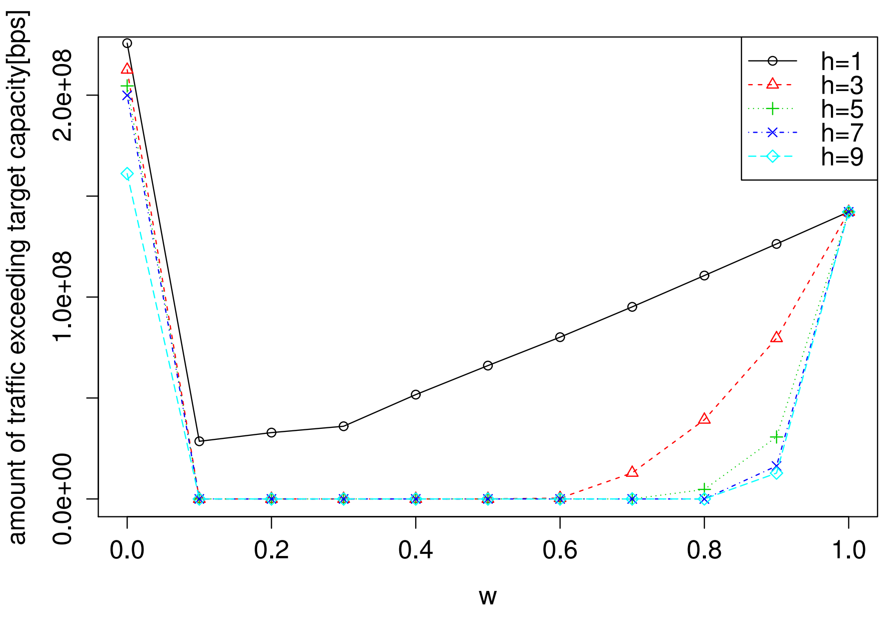


Figure 3.12: Maximum amount of traffic exceeding the target link capacity for all time slots when the MPC-based TE is conducted with various values of w

3.5.3.2 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 3.13 shows the maximum amount of traffic exceeding the target link capacity when the MPC-based 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,

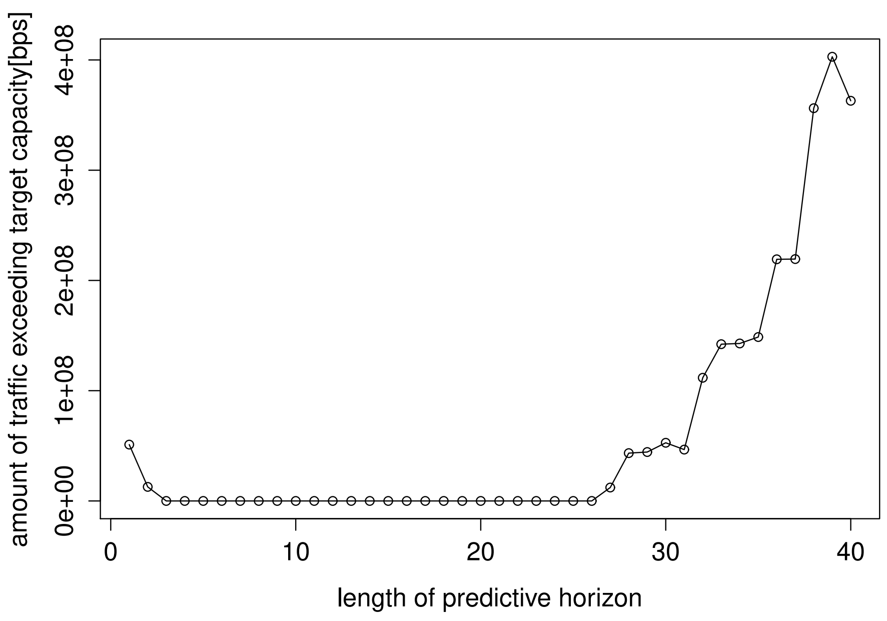


Figure 3.13: Maximum amount of traffic exceeding the target link capacity for all time slots when the MPC-based TE is conducted with various values of h $(w = 0.5)$

it is sufficient for the MPC-based TE to set h to 3 or slightly larger values.

3.5.3.3 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 $\hat{\hat{x}}(t+1), \cdots \hat{\hat{x}}(t+h)$. 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 $\hat{x}'(k) = \frac{T_p}{T_c} \sum_{i=(k-1)T_c}^{kT_c-1} \hat{\hat{x}}(i)$ 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 3.14 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. 3.13 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. 3.14, 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. 3.14, 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.

3.6 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 [12] and MATE [13], 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

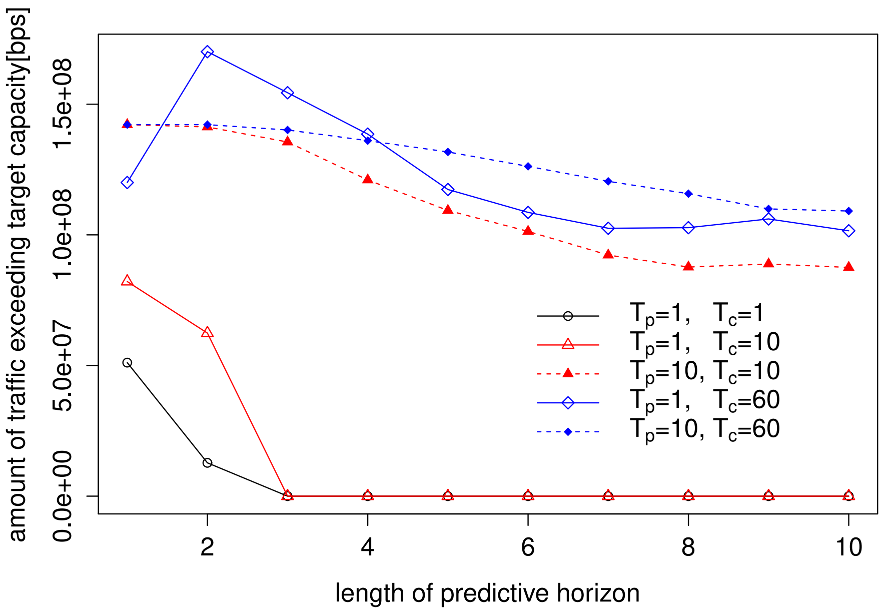


Figure 3.14: Maximum amount of traffic exceeding the target link capacity for all time slots when the MPC-based TE is conducted with various lengths of control and prediction ($w = 0.5$)

statistics.

In another application of MPC to TE, Rétvári and Németh [63] 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 [21, 22], ARCH [23], GARCH [24], and neural networks [25,26] . 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.

3.7 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.

Chapter 4

Traffic Engineering Guaranteeing Risk Probability against Prediction Uncertainty

4.1 Introduction

Traffic engineering (TE) plays an essential role in deciding routes that effectively use network resources. This is particularly important when one considers increasing time variation of Internet traffic such as streaming and cloud services. In the past, backbone networks have addressed this problem by reserving redundant link capacity to accommodate traffic surges [2]. However, doing so incurs significantly higher costs as the average and variance of traffic increases; poor network resource utility tends to reserve more than double the capacity required to accommodate the actual traffic. Dynamic TE methods have been studied for treating time-varying traffic in a way that effectively utilizes limited resources [9, 11, 12]. In the dynamic TE method, a control server periodically observes network traffic and dynamically reroutes flows to accommodate the observed traffic. However, such methods set routes for only observed traffic. This renders configured routes unsuitable if traffic changes happen, because routes are not changed during the next control cycle. One might think that the control server should quickly respond to traffic changes by shortening the control cycle interval, but doing so can induce frequent route changes resulting in route oscillation that degrades 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. Route oscillations also cause overly frequent changes in round-trip time (RTT), which decreases the throughput of delay-based TCP [20]. Hence, a dynamic TE that avoids congestion without significant route changes is required.

One approach to solving such problems is to use the predicted traffic. In [33], we have presented such a TE method, which is based on a control theory of predictive control called *model predictive control (MPC)*. In this method, the control server calculates the routes of several future time slots (or control cycles), considering predicted traffic variation. The control server then sets up only the next-step routes, and corrects the traffic prediction by reflecting traffic changes newly observed. It can then follow traffic variation without significantly changing the routes. Of course, traffic prediction errors cannot be wholly avoided even in the above method, and prediction errors may cause congestion in the next step.

In this chapter, we discuss the TE method that is robust to prediction errors. One approach to handling prediction uncertainty is to utilize a probability distribution of prediction errors. Fortunately, MPC can consider the probability distribution, by introducing the constraints that the risk of control failure should be less than a predefined threshold. MPC considering the probabilistic distribution is called *stochastic MPC(SMPC)* [40].

In this chapter, we apply SMPC to TE. We call this *stochastic model predictive traffic engineering (SMP-TE)*. We first model the problem of TE as an optimization problem that maximizes the network performance under the constraints that the probability of congestion should be less than a predefined threshold. In SMP-TE, this optimization problem is solved by the controller so as to obtain the routes for future time slots. Then, similar to the TE based on MPC [33], the controller sets up only the next-step routes, and recalculates the routes for the future time slot after the correction of the traffic prediction by reflecting newly observed traffic data. SMP-TE avoids congestion without requiring a large amount of excess network resources even if prediction causes errors, because the optimization problem considers the distribution of prediction errors.

In SMP-TE, one of the important problems is the threshold for the probability of congestion.

Generally, the prediction errors of the traffic in the far future become large. Such a large prediction error may cause the unnecessary route changes. To solve this problem, we also propose a relaxation of probability constraints.

The rest of this chapter is organized as follows. Section 4.2 describes our TE method, SMP-TE, in which we apply SMPC to the TE method. Section 4.3 presents an evaluation of our TE method comparing with the simple prediction-based TE. Section 4.4 presents our concluding remarks.

4.2 SMP-TE: Stochastic Model Predictive Traffic Engineering

Before describing our SMP-TE, we first briefly introduce SMPC in Subsection 4.2.1. We then move to our proposed SMP-TE in Subsection 4.2.2.

4.2.1 Stochastic Model Predictive Control

4.2.1.1 Model Predictive Control

First, we briefly show the concept of MPC. MPC is a method of system control based on predictions of system dynamics that has been studied in recent years. In MPC, a controller sets an input parameter so as to maintain system performance at close to an operator-specified target. Unlike traditional system control, the MPC controller predicts changes in the output value to calculate 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 using the objective function $J_1 = \sum_{k=t+1}^{t+h} \|y(k) - r_y(k)\|^2$, where $\| \cdot \|$ represents the Euclidean norm:

$$(u(t+1), \cdots, u(t+h)) = \underset{(u(t+1),\cdots,u(t+h))}{\arg\min} J_1. \tag{4.1}$$

To solve the above optimization problem, future outputs $y(t+1), \cdots, y(t+h)$ 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

$$z(k+1) = \phi(k, z(k), u(k)) \tag{4.2}$$

$$y(k) = \psi(k, z(k), u(k)), \tag{4.3}$$

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. We use $\hat{y}(k)$ as the predicted value of $y(k)$.

Modeling the system by a mathematical formula, however, may entail modeling errors. The prediction of system output will entail error under such an incorrect model, and prediction errors become increasingly large with more distant predictive horizons. One approach to solve this increasing error is to use feedback of actual system output. That is, the MPC controller implements only the first of the calculated inputs $u(t+1)$. Then, the MPC controller observes the output and corrects the prediction, using the output value. 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. (4.1), the controller calculates the input values by the following optimization problem:

$$(u(t+1), \cdots, u(t+h)) = \underset{(u(t+1),\cdots,u(t+h))}{\arg\min} \quad J_1 + wJ_2, \tag{4.4}$$

where w is a parameter for weighting the two objective functions J_1 and J_2.

4.2.1.2 Probability Constraints

Realistic systems entail input or output constraints such as physical constraints and boundary conditions. Here, if the system has an upper bound on output, the MPC controller needs to search the optimal input under the constraint

$$y(t+k) \leq y_u,\ k = 1, \cdots, h, \tag{4.5}$$

where y_u is the upper bound of the output value.

If a modeling error exists, the calculated input may be infeasible, which violates the constraint (4.5). Given the exact upper bount $\epsilon > y(t+k) - \hat{y}(t+k)$, the controller determines the input without the constraint violation by maintaining the constraint

$$\hat{y}(t+k) + \epsilon \leq y_u,\ k = 1, \cdots, h. \tag{4.6}$$

However this hard constraint causes overly conservative control, and guaranteeing applicability to the worst-case scenario will considerably degrade control performance. Additionally, exact upper bounds are rarely available in actual situations.

A probability distribution of error can determine a soft bound to use in place of an exact bound on modeling error. Assuming that the k-th ahead modeling error $\epsilon_y(t+k)$ is a random variable following a certain probability distribution, the output value $y(t+k) = \hat{y}(t+k) + \epsilon_y(t+k)$ is also a random variable. Then, the probability that $y(t+k)$ violates the upper bound can be defined, and we denote the probability as $P[y(t+k) > y_u]$. SMPC is a control method that deals with such random variables of output. To avoid constraint violations, the violating probability should be at a certain level p. Then, the controller calculates safe inputs by the probability constraint

$$P[y(t+k) > y_u] < p,\ k = 1, \cdots, h. \tag{4.7}$$

Eq. (4.7) becomes a harder constraint when p is small, and Eq. (4.7) is equivalent to Eq. (4.6) when $p = 0$. Even allowing for the rare case of constraint violation according to p, the controller can still robustly avoid performance degradation to model errors.

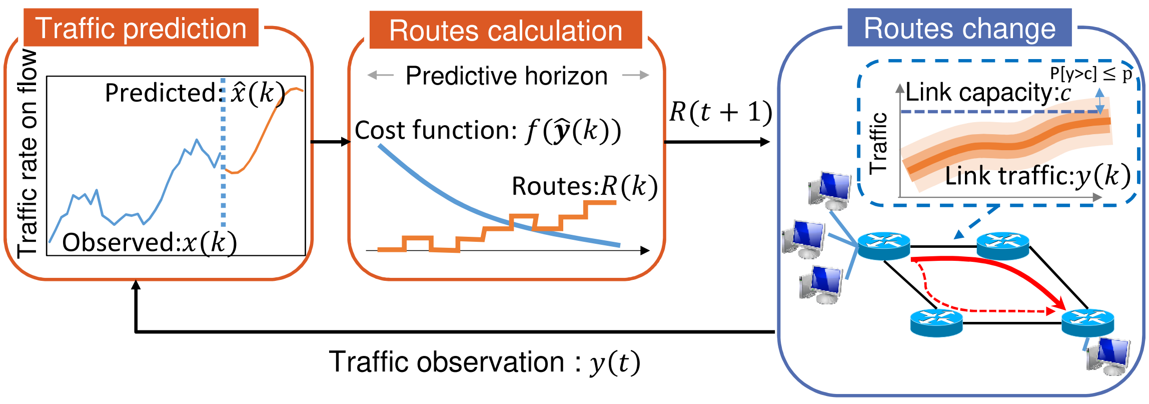


Figure 4.1: Overview of SMP-TE

4.2.2 Applying SMPC to TE

4.2.2.1 TE Model for SMPC

We apply SMPC to TE, and realize a prediction-based TE that is robust to prediction errors. Figure 4.1 shows an overview of our TE method, to which SMPC 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 routes $R(k)$ and measures network outputs and the traffic rates on links $\boldsymbol{y}(k)$. The control server periodically changes 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. 2) The control server calculates routes from the prediction so as to minimize a cost function $f(R(k))$ while maintaining a low congestion occurrence probability.

4.2.2.2 Formulation of the Optimization Problem

To avoid congestion caused by prediction errors, we use probabilistic constraints as capacity constraints. Given target capacities C_l and probability p, the controller maintains the occurrence probability of capacity overshooting $P[y_l(k) > C_l]$ under p. With this constraint, the control server computes routes by considering objective functions $J_1 = \sum_{k=t+1}^{t+h} f(R(k))$, which indicates a summation of the cost function at each time slot, and $J_2 = \sum_{k=t+1}^{t+h} \|\Delta R(k)\|^2$, which indicates the sum of squares of the amount of route changes. 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 route changes.

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

$$\begin{aligned} &\mathit{minimize} && \sum_{k=t+1}^{t+h} \Big((1-w)f(R(k)) + w\|\Delta R(k)\|^2\Big) && (4.8) \\ &\mathit{subject\ to} && \forall k, \hat{\boldsymbol{y}}(k) = G \cdot R(k) \cdot \hat{\boldsymbol{x}}(k) && (4.9) \\ &&& \forall k, l, P[y_l(k) > C_l] \leq p && (4.10) \\ &&& \forall k, \forall i, \forall j, R_{i,j}(k) \in [0,1] && (4.11) \\ &&& \forall k, \sum_{i \in \wp(j)} R_{i,j}(k) = 1. && (4.12) \end{aligned}$$

where $\hat{\boldsymbol{x}}(k)$ is predicted value of $\boldsymbol{x}(k)$, $wp(j)$ is the set of available paths of OD flow j, and G is a matrix whose (i,j) element $G_{i,j}$ is 1 if the path j traverses the link i and 0 otherwise. Here, $\hat{\boldsymbol{x}}(k), G$ are given variables and $R(k), \hat{\boldsymbol{y}}(k)$ are the variables to be optimized. Eq. (4.9) represents the relation between the traffic rates of the OD flows and links. Eq. (4.10) is the probabilistic constraint that the probability of congestion occurrence is lower than p. Eqs. (4.11) and (4.12) 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 $\hat{\boldsymbol{x}}(k)$ using the newly observed traffic rate, and recalculates the next routes by solving the optimization problem again.

4.2.2.3 Relaxation of Future Probabilistic Constraint

In the above formulation of SMP-TE, the probability p was constant for all time slots within the predictive horizon. However, prediction accuracy for the next time slot is more important in the current model setting. Also, further future prediction is less reliable. Accordingly, forcing the same level of probabilistic constraint for unreliable far-future predictions incurs unnecessary routes changes. This is because the probability $P[y_l(k) > C_l]$ becomes large when the $y_l(k)$ has large variance, and constraints Eq. (4.10) becomes more active.

To solve this problem, we introduce the increasing probability $p(k)$ for the capacity constraints. By replacing the p in Eq. (4.10) with $p(k)$, the probabilistic constraint is gradually relaxed as time slots advance. There are many possible approaches to relaxing the probability. In this chapter, we exponentially decrease the complement probability $q(k) = 1 - p(k)$ as

$$q(k) = (1 - p) \exp(-\frac{k - t - 1}{\tau}), \tag{4.13}$$

where τ is the time constant that determines the decreasing speed. If $q(k)$ is less than 0.5, even the expected traffic rates are not accommodated by the calculated routes. In this case, calculated routes no longer avoid the congestion, hence it does not make sense to consider the case of $q(k) < 0.5$. We thus limit the minimum value of $q(k)$ as 0.5.

4.3 Evaluation

4.3.1 Simulation Environment

4.3.1.1 Network Topology

In the following evaluation, we use the Internet2 backbone network (Fig. 4.2). The Internet2 backbone network has 9 PoP routers and 13 bidirectional links.

4.3.1.2 Traffic

We use actual traffic traces [58] monitored from 00:00 on 6 Feb 2014 to 23:59 on 12 Feb 2014. These traffic data were 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 5 min. Though sampled data may contain sampling errors, those errors do not have a large impact on our evaluation because a huge number of flows are aggregated into OD flows, in which the aggregated error of each flow is much smaller than the total traffic amounts.

Our interest lies on how the SMP-TE can avoid the congestion under limited resources and the

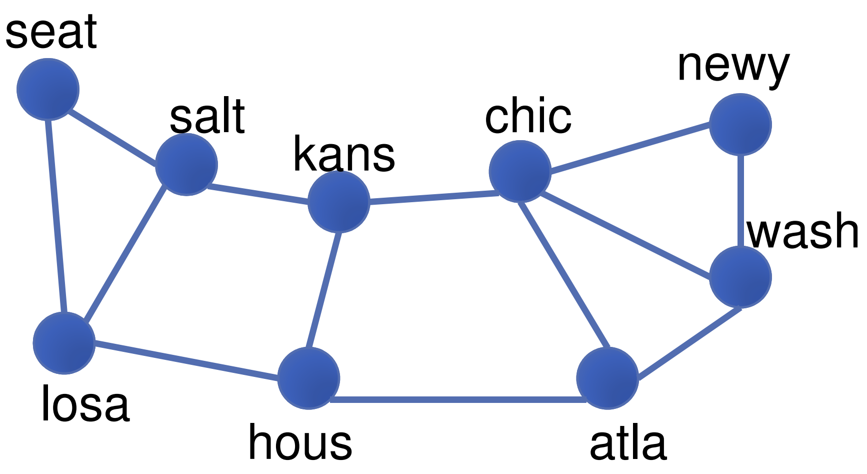


Figure 4.2: Internet2 topology

existence of prediction error. However, in fact, the Internet2 network is not congested due to an over-provisioning of link capacity; the maximum link utilization is less than 20%. Accodingly, we artifically set up a congested environment by multiplying actual traffic amounts by 5, and setting the target link utilization to 95% in the following evaluation. The traffic data used in our evaluation is shown in Fig. 4.3, where the time slot length is set to 2 h.

4.3.1.3 Prediction Error Model

In our evaluation, the predicted traffic rates are given by adding prediction errors to the actual traffic rates to evaluate the SMP-TE without impact of the specific prediction model. The prediction error is generated so as to follow a zero-mean Gaussian distribution, based on the existing chapters on prediction methods [21, 64].

Assuming that the prediction error of each time slot is independent, the variance of prediction error in the kth ahead time slot is $\sigma^2 k$ where σ^2 is the variance of one-step prediction error. We set the variance of one-step prediction error on flow j based on a normalized prediction error metric called *normalized mean squared error (NMSE)*:

$$\text{NMSE} = \frac{\sigma_j^2}{V[x_j(t)]}, \tag{4.14}$$

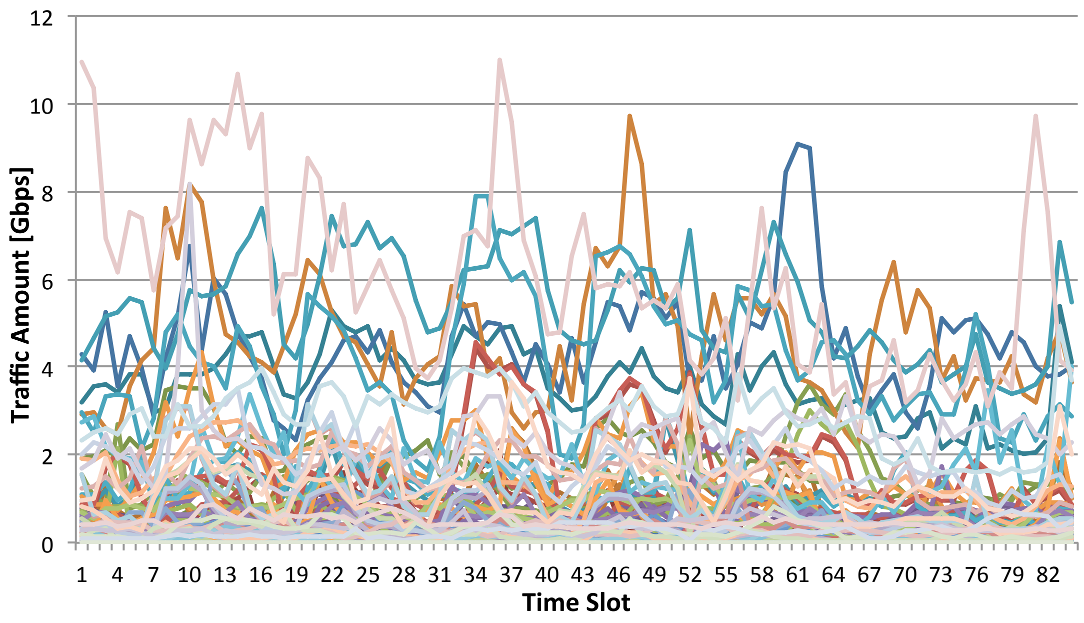


Figure 4.3: Internet2 network traffic (time slot granularity: 2 h)

where σ_j^2 is the variance of prediction error on flow j, and $V[x_j(t)]$ is the variance of actual traffic. Therefore, we set σ_j^2 to $0.3V[x_j(t)]$ based on typical NMSE values [65].

4.3.1.4 Cost Function

In our evaluation we use the average hop length as a cost function, because reducing hop length lowers propagation delay in the calculated routes. Because the queuing delay is negligible when the link load is under a certain targeted capacity, we minimize the end-to-end delay by minimizing the propagation delay. The average hop length D is defined using R; $D = \frac{1}{F}\sum_j \sum_{i\in\wp(j)} R_{i,j}d_i$, where d_i is the path length of i. We use the normalized hop length $\frac{D}{\max_j d_j}$ as a cost function. Though we conduct the simulation with changing the weighting parameter w from 0 to 1 by 0.1, we show only the result with $w = 0.5$ because the similar result is obtained with any $0 < w < 1$.

4.3.1.5 Routing Calculation

To solve the probabilistic optimization Eqs. (4.8)–(4.12), we transform the probabilistic constraint Eq. (4.10) into a deterministic constraint. Similaly to [66], the probabilistic constraint Eq. (4.10) is equally replaced by following deterministic constraint

$$\forall k, \forall l, \hat{y}_l(k) + \Phi^{-1}(1-p)\sqrt{\sum_j A_{l,j}(k)^2 \sigma_j^2 k} \leq C_l(k), \tag{4.15}$$

where Φ^{-1} is the quantile function of the Gaussian distribution, and $A_{i,j}(k)$ is the (i,j)-element of the matrix $A(k) = G \cdot R(k)$, which indicates the fraction of OD flow j traversing link i.

As a result of the transformation, our optimization problem Eqs. (4.8)–(4.12) is equally replaced by a convex optimization program called *second-order cone programming (SOCP)*. We use the optimization problem solver CPLEX [60] package to solve the SOCP. We ran CPLEX on a computer with four Intel Xeon E7-4870 processors. The calculation of each time slot is finished within a few seconds for Internet2 topology.

4.3.1.6 Compared Methods

We compare our SMP-TE method with two prediction-based TE methods. The first is the simplest TE method, which uses only one-step ahead prediction without considering the probability distribution of prediction error. This is a special case of our method when parameters are set to $h = 1$ and $p = 0.5$. Comparison with this method demonstrates the effect of multi-step prediction.

The second method is simple MPC-based TE similar to [33], which uses multi-step-ahead prediction without considering the probability distribution. This is also a special case of SMP-TE with parameter setting $p = 0.5$. Hereafter, we call this MPC-based TE as MP-TE. Comparison with this method confirms that the stochastic constraint is effective for avoiding the impact of prediction error without causing significant route changes.

4.3.2 Effect of Stochastic Constraint

First, we show how the stochastic constraint is effective for avoiding congestion caused by prediction error. Figure 4.4 shows the queuing delay of the bottleneck link. Shown is a 99.9% delay, which means that 99.9% of packets will experience delay caused by queuing on a link lower than this value. We calculated the link delay from link utilization by approximating packet processing on the Internet by the M/M/1 queuing model. According to queuing theory, 99.9% delay is calculated as $-\log(1-0.999)\frac{\bar{L}}{C_l - y_l}$, where $\bar{L}$ is the average packet length, and C_l is the actual capacity of link l

In Fig. 4.4, SMP-TE achieves lower delay in both cases of one-step prediction and multi-step prediction. This is because SMP-TE sets a safer route that accommodates the traffic without congestion, even when the prediction error occurs. On the other hand, the non-stochastic approaches of simple prediction-based TE and MP-TE cause higher delay, because the routes calculated using only expected traffic no longer deal with unexpected traffic arrival. Of course, congestion may occur with even SMP-TE if the prediction error is significantly outside of the expected range. However, this case only occurs with probability p, which the network operator can set to an appropriate value.

4.3.3 Multi-step Prediction Effect

The prediction-based TE can gradually change routes by predicting future congestion in advance. Although this is not an effect of the stochastic approach, our interest is in how this prediction effect is reproduced in our SMP-TE. Figure 4.5 shows the maximum difference of the path fraction, which is defined as $\max_p |\Delta R_p(t)|$ in each case of the TE method. From this figure, TE with one-step prediction requires a significant route change at time slot 31, but that is avoided by using the multi-step prediction. This is because gradual route changes proactively proceeded before the actual traffic change by considering the multi-step prediction. This indicates that far-future prediction is also effective toward avoiding significant route changes in SMP-TE.

However, the frequency of route changes increases in both SMP-TE and MP-TE. This is because wasteful routes changes occur when the predicted future congestion does not actually occur. SMP-TE in particular causes more route changes, because the control server becomes more conservative at expanding future prediction error. One solution of this too-conservative to far-future congestion is relaxation of the future probabilistic constraint, as mentioned in subsection 4.2.2. The effect of constraint relaxation is discussed in the next subsection.

4.3.4 Probability Relaxation Effect

In this subsection, we discuss the effect of relaxing future constraints. Figure 4.5 shows that relaxation avoids frequent route changes, while Fig. 4.4 shows that congestion is avoided even in the case of the relaxation. To discuss the effect of the probability in more detail, we focus on the route changes performed by the TE methods. Table 4.1 shows a summary of routes changes performed by each TE method. In this table, "average" means the average difference of path fraction $|\Delta R_p(t)|$ for all times and all paths, "max" means the maximum difference of the path fractions, and "frequency" means the ratio of time slots in which more than 1% of traffic is moved from a path to other paths. As previously mentioned, the maximum route changes become small in the TE-method using multi-step prediction. On the other hand, more frequent route changes occurred in multi-step prediction, though the average routes changes are the lowest among TE methods.

Comparing the relaxed SMP-TE with original SMP-TE, relaxation of constraints can reduce the frequency of route changes. However, the maximum route changes in relaxed SMP-TE are larger than in the original SMP-TE. This is because constraint relaxation delays the control server reaction to future congestion. With larger τ, the relaxation is gradually conducted over time, avoiding such response delay. However, the result causes frequent route change during ordinary traffic. Therefore, τ should be appropriately tuned to an optimal balance between following traffic variation and ignoring prediction error. Our future work will include developing a method for tuning this parameter.

4.3.5 Scalability

Theoretically, the worst-case time complexity of each iteration to solve SOCP is $O(N^2 \sum_i N_i)$ [67] where N is the number of variables and N_i is the dimension of second-order cone constraints. In the SMP-TE, the number of variables is $O(mn^2)$ and the dimension of second-order cone constraints

Table 4.1: Route Changes Caused in Each TE Method

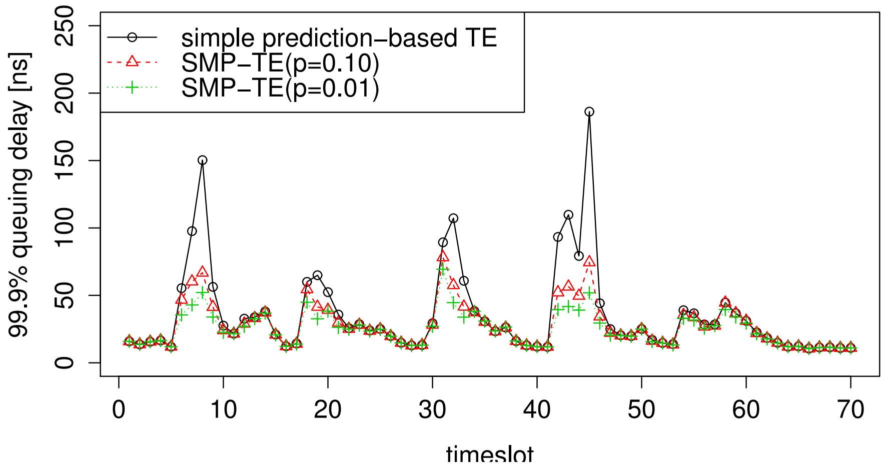
TE using one-step prediction			
	average	max	frequency
simple prediction-based TE	0.083%	10%	16%
SMP-TE($p = 0.1$)	0.094%	14%	23%
SMP-TE($p = 0.01$)	0.11%	16%	36%
TE using multi-step prediction($h = 5$)			
	average	max	frequency
MP-TE	0.074%	6.3%	33%
SMP-TE($p = 0.1$)-relaxed($\tau = 5$)	0.088%	8.9%	42%
SMP-TE($p = 0.1$)-relaxed($\tau = 20$)	0.089%	7.1%	46%
SMP-TE($p = 0.1$)	0.090%	6.0%	51%
SMP-TE($p = 0.01$)-relaxed($\tau = 5$)	0.10%	12%	52%
SMP-TE($p = 0.01$)-relaxed($\tau = 20$)	0.11%	8.9%	62%
SMP-TE($p = 0.01$)	0.012%	6.0%	78%

is $O(n^2)$ where m, n and are the number of links and nodes, respectively. Therefore, the computational complexity of SMP-TE is $O(m^2 n^6)$. To use SMP-TE in a large network, we should reduce the calculation time. One approach is decomposing a network into multiple ranges and applying the SMP-TE to each range, which is one of our future research topics.

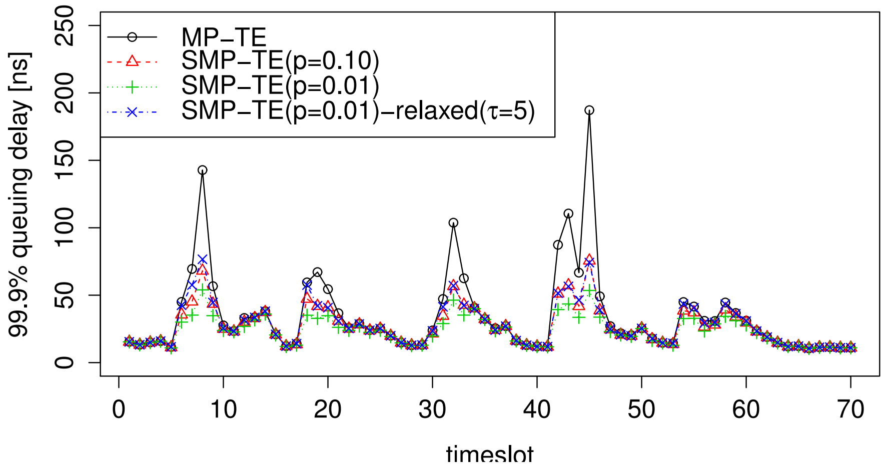
4.4 Conclusion

We have proposed a TE method called SMP-TE that follows predicted traffic in a way that avoids the impact of prediction error. According to the basic idea of SMPC, our SMP-TE calculates routes while considering the probability distribution of prediction error. Through simulation using actual backbone network traffic traces, we demonstrated that SMP-TE can avoid congestion in cases where simple prediction-based TE cannot. Additional route changes required to accommodate the prediction error remained small.

Future work will include further verification of our method using larger networks with more realistic traffic. Furthermore, we will reduce calculation times by adapting SMP-TE to distributed control that determines routes using only local traffic information. We are now implementing SMP-TE, and will report the results in a forthcoming paper.

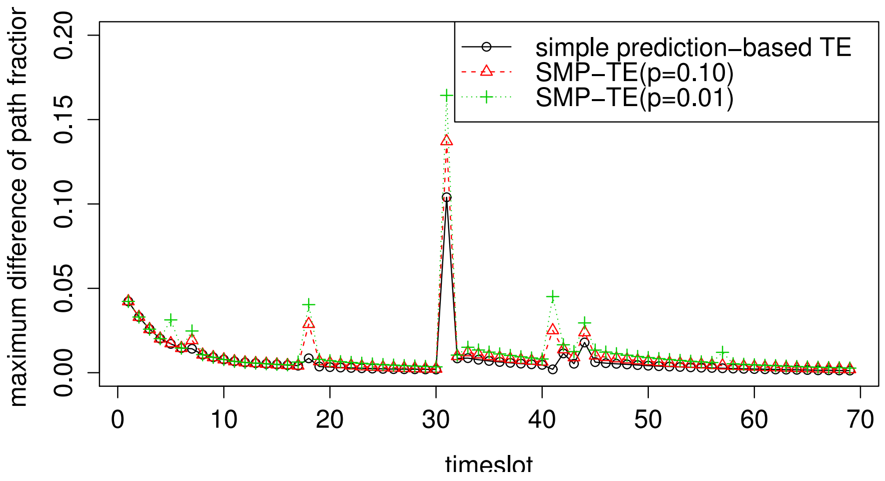


(a) With One-step Prediction($h = 1$)

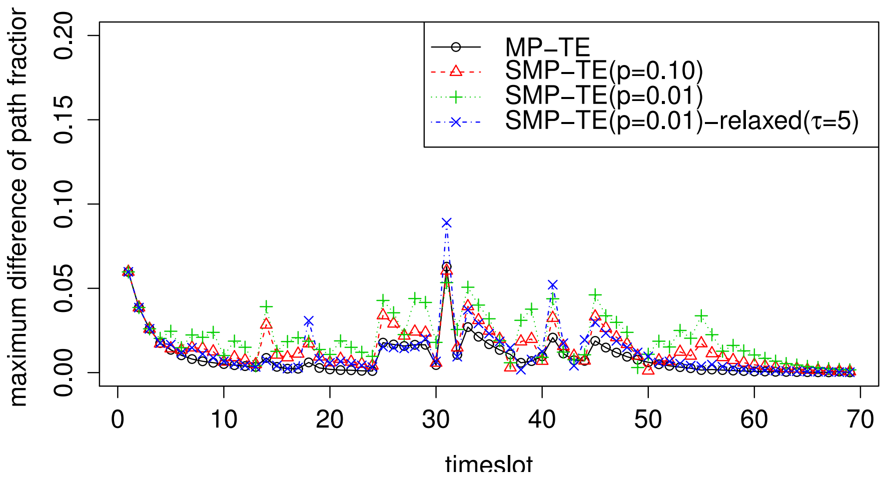


(b) With Multi-step Prediction($h = 5$)

Figure 4.4: 99.9% Queuing Delay on the Bottleneck Link at Each Time



(a) With One-step Prediction($h = 1$)



(b) With Multi-step Prediction($h = 5$)

Figure 4.5: Maximum Difference of Path Fraction at Each Time

Chapter 5

Scalable Traffic Engineering by Hierarchical Model Predictive Control

5.1 Introduction

The hierarchical control scheme is a promising approach to improving the scalability of network controls [43–48]. 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.

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 [49, 50]. 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)* [34, 35], 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 this chapter. To develop an effective MPC-based hierarchical control method, we utilize the idea of *hierarchical traffic engineering (TE)* [45, 47, 48]. 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 [33, 39]. 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 chapter is organized as follows. Section 5.2 surveys related work. Section 5.3 explains the overview of hierarchical network control. Section 5.4 explains the framework of MPC-based hierarchical network control. In Section 5.5, we propose a new hierarchical TE method called hierarchical MP-TE based on an MPC-based hierarchical network framework. Section 5.6 evaluates hierarchical MP-TE. Section 5.7 presents our concluding remarks.

5.2 Related work

Hierarchical Network Control

Hierarchical network control [43–46] 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* [48, 49] 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 [49,68]. For instance, Chang *et al.* used multiple policies for updating routing information to avoid route oscillation [49], 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.

MPC and Its Application in Network Control

MPC is a method of system control based on predictions of system dynamics [34, 35]. MPC effectively handles environmental changes by combining the feedback and feedforward controls, whose detail is given in Chapter 3. 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 [33], 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* [39] 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 chapter, 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.

5.3 Hierarchical Network Model

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 $\boldsymbol{z}(k)$.

$$\boldsymbol{X}(k) = g(\boldsymbol{z}(k)), \tag{5.1}$$

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,

$$\boldsymbol{z}(k) = f(\boldsymbol{z}(k-1), \boldsymbol{u}(k)) + \boldsymbol{\epsilon}(k), \tag{5.2}$$

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 $\hat{z}(k) = g^{-1}(\boldsymbol{X}(k))$, 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 $\boldsymbol{u}^{2;b}(k+1)$.

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.

5.4 Hierarchical Network Control based on MPC

In this section, 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 5.3, 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, the control server minimizes the objective functions J_1 and J_2, which are determined as follows:

$$J_1 = \sum_{k=t+1}^{t+h} \|\hat{\boldsymbol{z}}^{m;a}(k) - r_z(k)\|^2 \tag{5.3}$$

$$J_2 = \sum_{k=t+1}^{t+h} \|\Delta \boldsymbol{u}^{m;a}(k)\|^2 \tag{5.4}$$

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:

$$(\boldsymbol{u}^{m;a}(t+1), \cdots, \boldsymbol{u}^{m;a}(t+h)) = \underset{(\boldsymbol{u}^{m;a}(t+1), \cdots, \boldsymbol{u}^{m;a}(t+h))}{\arg\min} (1-w)J_1 + wJ_2. \tag{5.5}$$

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.

5.5 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 5.4. In this section, we first formulate hierarchical TE; then we propose a new TE method.

5.5.1 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.

5.5.1.1 Construction of the Hierarchical Network

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

Area Partitioning Area partitioning divides the network into multiple areas so that each area includes the connected subnetwork of the original network. Similarly to [43,47,48], 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 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., [43], we manually divide the network into areas in the evaluation.

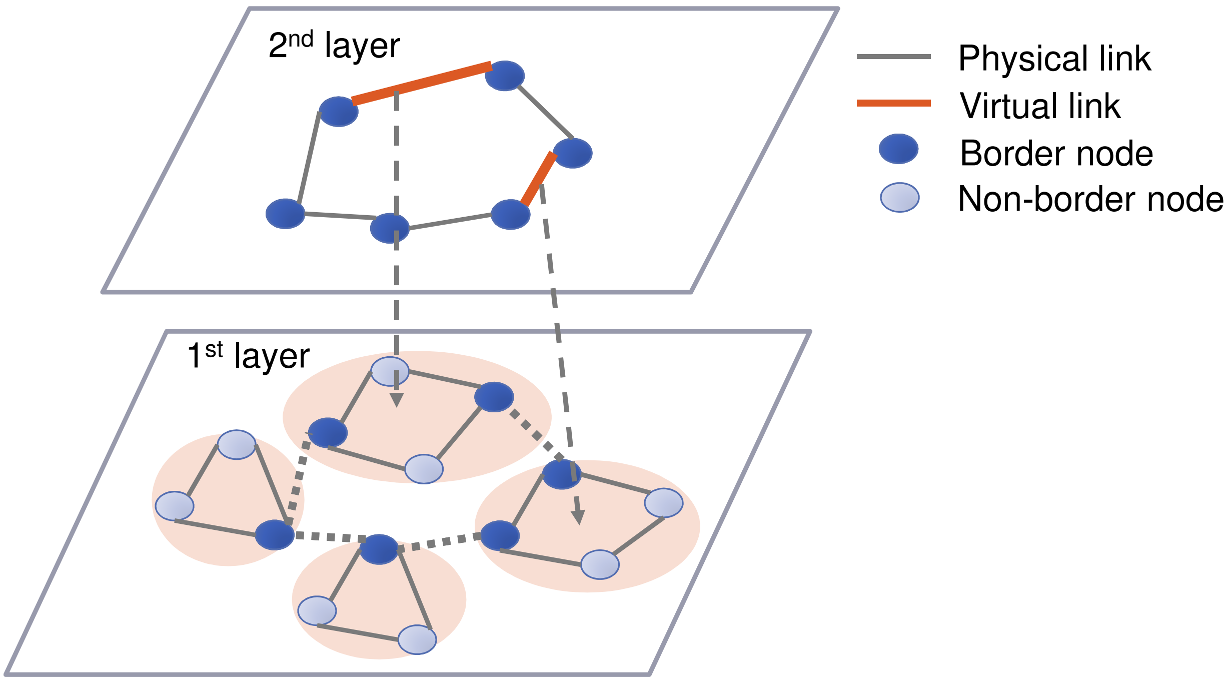


Figure 5.1: The hierarchical network model.

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 [47, 68], and there is a trade-off between information accuracy and topology complexity.

In this chapter, 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 5.1 shows an example of the hierarchical network. In this network, the upper layer includes the virtual links and the physical links between different areas.

5.5.1.2 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.

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 5.5.1.1, 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

$$C_l^{m;a}(k) = c_l - y_l^{m;a}(k) \tag{5.6}$$

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 chapter, 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

$$C_l^{m;a}(k) = \sum_{p \in P(i)} \min_{i \in L(p)} (C_i^{m-1;b}(k) - y_i^{m-1;b}(k)) \tag{5.7}$$

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.

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

$$\zeta_l^{m;a}(k) = [\Delta y_l^{m;a}(k) - C_l^{m;a}(k)]^+ \tag{5.8}$$

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. (5.8), 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:

$$\boldsymbol{y}^{m;a}(k) = G^{m;a} \cdot R^{m;a}(k) \cdot \boldsymbol{x}^{m;a}(k) \tag{5.9}$$

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:

$$\begin{aligned}
&minimize &:\quad & \left|\left|\hat{\boldsymbol{\zeta}}^{m;a}(t+1)\right|\right|^2 && (5.10)\\
&subject\ to &:\quad & \boldsymbol{y}^{m;a}(t+1) = G^{m;a} \cdot R^{m;a}(t+1) \cdot \boldsymbol{x}^{m;a}(t) && (5.11)\\
& & & \forall l, \zeta_l^{m;a}(t+1) = [\Delta y_l^{m;a}(t+1) - C_l^{m;a}(t)]^+ && (5.12)\\
& & & \forall f, p, R_{p;f}^{m;a}(l) \in [0,1] && (5.13)\\
& & & \forall f, \sum_{p \in \wp^{m;a}(f)} R_{p;f}^{m;a}(t+1) = 1 && (5.14)
\end{aligned}$$

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. (5.11) represents the relation between the traffic rates of the flows and links. Eq. (5.12) is the definition of ζ. Eqs. (5.13) and (5.14) 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.

5.5.2 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 5.5.1, 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 5.5.1.2. 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.

5.5.2.1 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 [21, 22], ARCH [23], GARCH [24], or neural networks [25, 26], 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 $\tau = 1$.

5.5.2.2 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 5.5.1.2, 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 $\boldsymbol{\zeta}^{m;a}(k) = \mathbf{0}$. Then, according to Section 5.4, the control server calculates the routes by solving the following optimization problem:

$$\begin{aligned}
minimize \quad &: \sum_{k=t+1}^{t+h}\left(\frac{1-w}{N_L^{m;a}}\left\|\frac{\hat{\boldsymbol{\zeta}}^{m;a}(k)}{Z^{m;a}}\right\|^2 + \frac{w}{N_P^{m,a}}\|\Delta R^{m;a}(k)\|^2\right) && (5.15)\\
subject\ to \quad &: \forall k, \hat{\boldsymbol{y}}^{m;a}(k) = G^{m;a}\cdot R^{m;a}(k)\cdot \hat{\boldsymbol{x}}^{m;a}(k) && (5.16)\\
&\quad \forall k, l, \hat{\zeta}_l^{m;a}(k) = [\Delta\hat{y}_l^{m;a}(k) - \hat{C}_l^{m;a}(k)]^+ && (5.17)\\
&\quad \forall k, f, p, R_{p;f}^{m;a}(l) \in [0,1] && (5.18)\\
&\quad \forall k, f, \sum_{p\in\wp^{m;a}(f)} R_{p;f}^{m;a}(k) = 1 && (5.19)
\end{aligned}$$

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. (5.15) 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 chapter we calculate the routes using the above optimization problem only when $Z^{m;a} \neq 0$.

5.6 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.

5.6.1 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.

5.6.1.1 Simulation Environment

Network Topology In the following evaluation, we use the lattice topology shown in Figure 5.2. 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.

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 5.3. 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

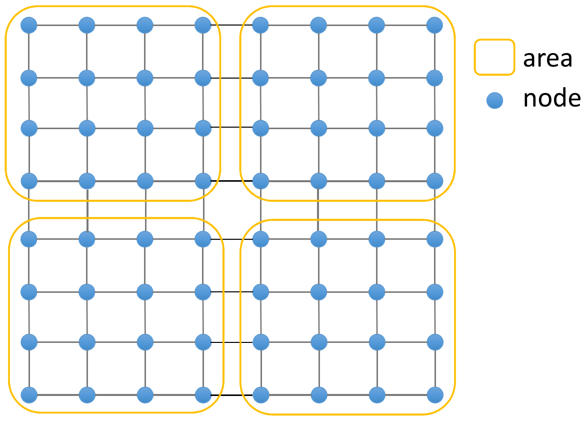


Figure 5.2: Lattice topology with 64 nodes.

change. In this situation, the links between the areas become congested.

Routing Calculation The routes in MP-TE are determined as a solution of the optimization problem (5.15)–(5.19). In a similar way to [33], the optimization problem is equally transformed as a convex quadratic programming problem which can be solved by common solvers. We use the CPLEX [60], 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.

Comparison Method For comparison, we use a basic hierarchical TE method without the MPC concept described in Section 5.5.1, 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.

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

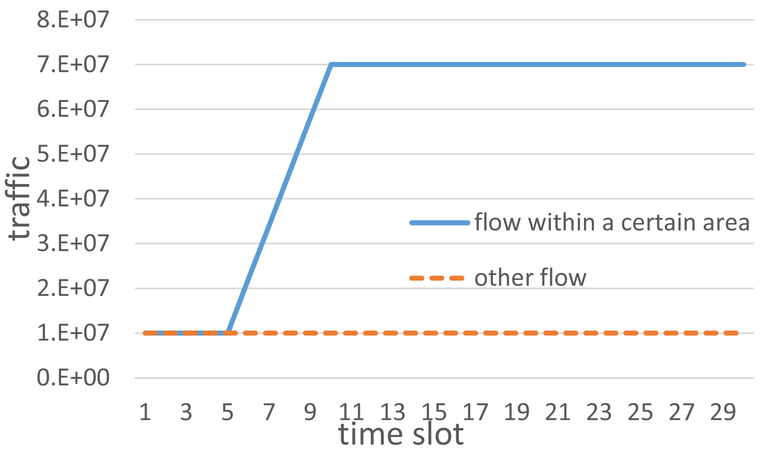


Figure 5.3: Time series of traffic causing inner-area congestion.

the routing has converged.

5.6.1.2 Results

Figure 5.4 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

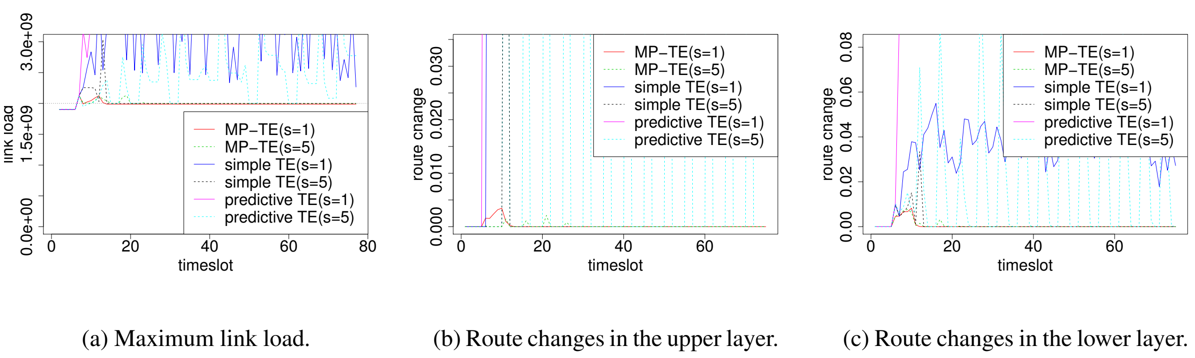


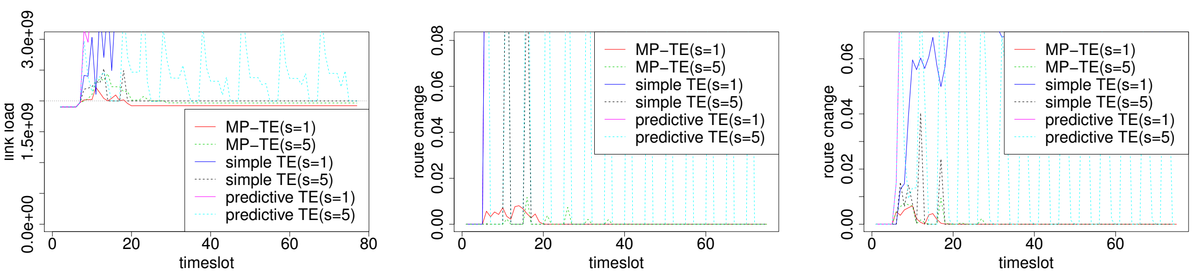
Figure 5.4: Time series for maximum link load and average route changes with increasing inner-area traffic.

parameter setting for MP-TE is given in Section 5.6.2. Additionally, Figure 5.5 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 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



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

Figure 5.5: Time series for maximum link load and average route changes with increasing inter-area traffic.

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.

5.6.2 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.

5.6.2.1 Simulation Environment

The simulation environment is almost the same as that mentioned in subsection 5.6.1.1, the main difference being the traffic pattern. In this simulation, we generate traffic which includes cyclic variation and noisy variation as [69]. The traffic rate from node i to node j at time step k is given as

$$x_{i,j}(k) = \mu \left(1 + \sin\left(\frac{2\pi}{T}k + \theta_{i,j}\right)\right) + W(k) \tag{5.20}$$

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 5.1 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

Table 5.1: 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

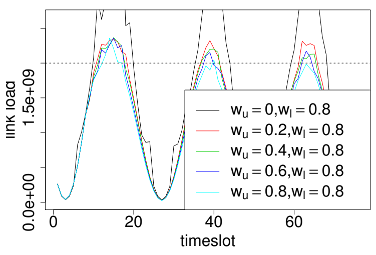
large. When $\sigma = 2.3 \times 10^6$, the standard deviation of prediction error is 5.3×10^6, which is about 76 % of average traffic. Since the actual error of one-step-ahead prediction is about 30% [29], our simulation covers the case where prediction error is much larger than the error actually expected.

5.6.2.2 Results

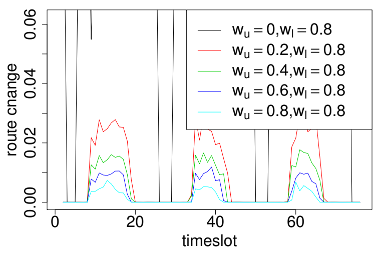
Figures 5.6–5.9 show the results of MP-TE. Figs. 5.6 and 5.7 show the cases of $\sigma = 0$ and $\sigma = 2.3 \times 10^6$, respectively, with various weights of route changes. In these figures, we change the value of w at lower and upper layers separately. We denote the value of w at the lower layer as w_l and that of the upper layer as w_u.

Similarly, Figs. 5.8 and 5.9 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.

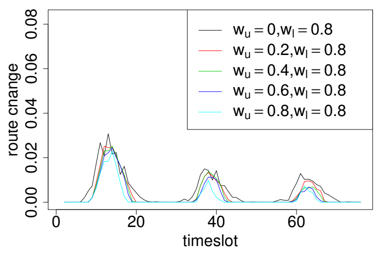
In addition, we show the result of simple TE in Figure 5.10 setting various control intervals at the upper layer. The figure shows that the maximum link load largely exceeds the targeted capacity around the peak time of all three cycles for any s. The reason for this is different for small and large 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



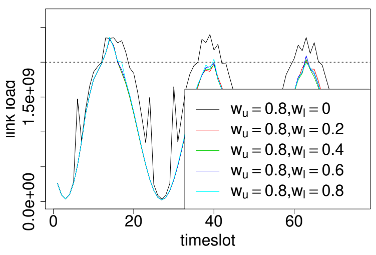
(a) Maximum link load with changing w_u.



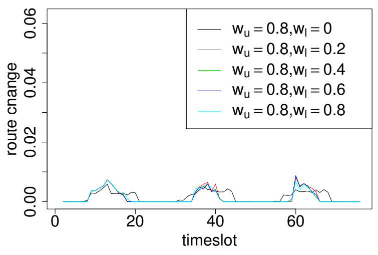
(b) Route changes in the upper layer with changing w_u.



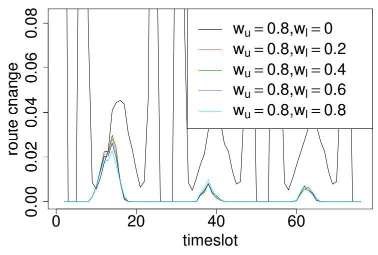
(c) Route changes in the lower layer with changing w_u.



(d) Maximum link load with changing w_l.



(e) Route changes in the upper layer with changing w_l.

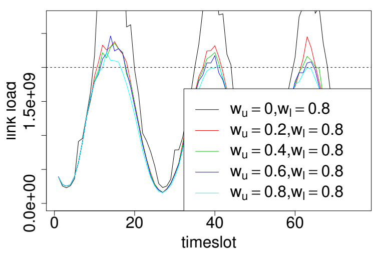


(f) Route changes in the lower layer with changing w_l.

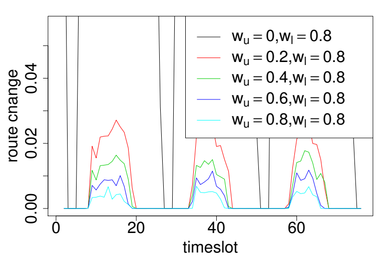
Figure 5.6: Time series of maximum link load and average route changes of MP-TE with various weights of route changes ($h = 3, \sigma = 0$).

traffic rates and virtual link capacities become inappropriate when s becomes large. Thus, simple TE causes large excess traffic for any s.

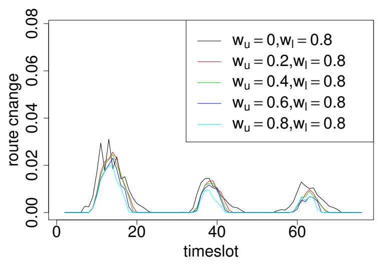
On the other hand, MP-TE keeps excess traffic to nearly zero with appropriate setting of parameters in Figs 5.6–5.9. As mentioned in 5.6.1.2, MP-TE can follow traffic changes quickly with prediction and setting a small control interval while routing oscillation is avoided by restricting route changes. 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.



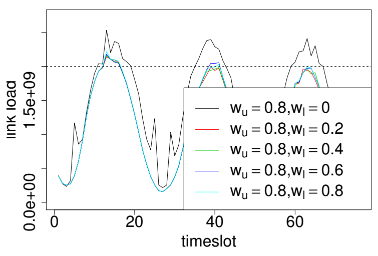
(a) Maximum link load with changing w_u.



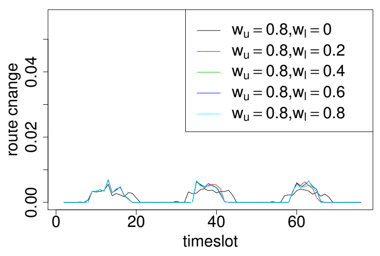
(b) Routes changes in the upper layer with changing w_u.



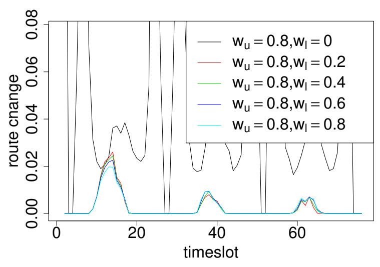
(c) Route changes in the lower layer with changing w_u.



(d) Maximum link load with changing w_l.

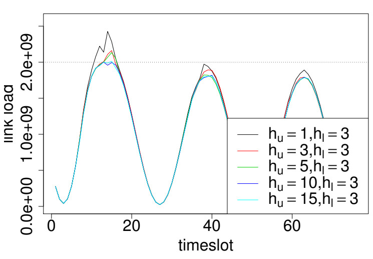


(e) Route changes in the upper layer with changing w_l.

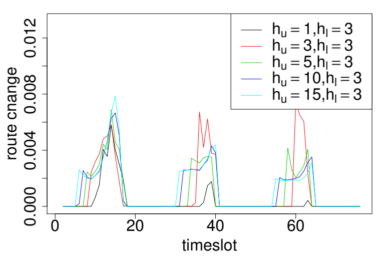


(f) Route changes in the lower layer with changing w_l.

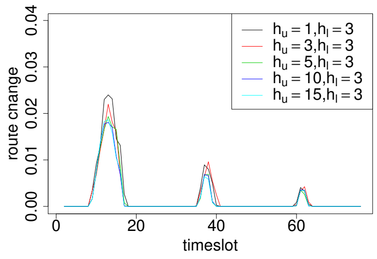
Figure 5.7: Time series of maximum link load and average route changes of MP-TE with various weights of routes changes ($h = 3, \sigma = 2.3 \times 10^6$)



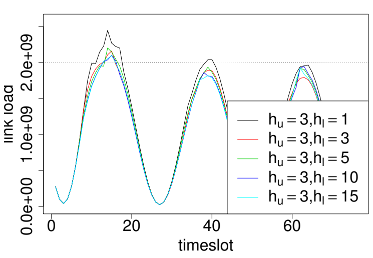
(a) Maximum link load with changing h_u.



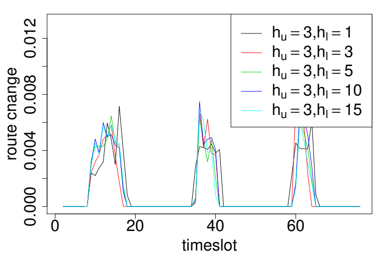
(b) Route changes in the upper layer with changing h_u.



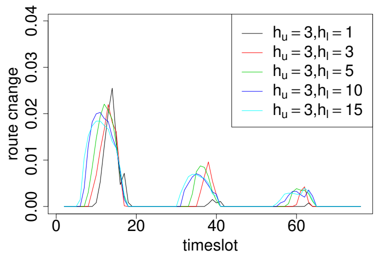
(c) Route changes in the lower layer with changing h_u.



(d) Maximum link load with changing h_l.

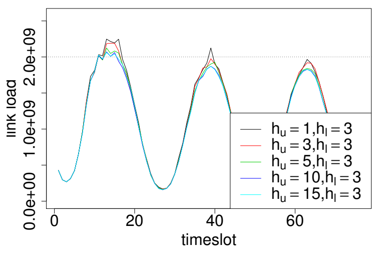


(e) Route changes in the upper layer with changing h_l.

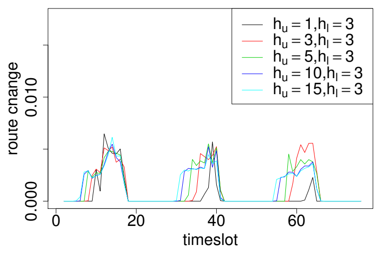


(f) Route changes in the lower layer with changing h_l.

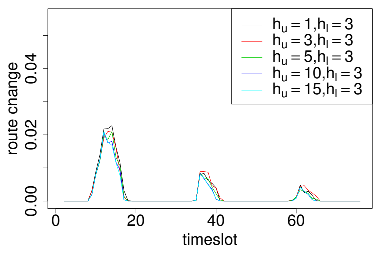
Figure 5.8: Time series of maximum link load and average route changes of MP-TE with various lengths of prediction ($w = 0.8, \sigma = 0$).



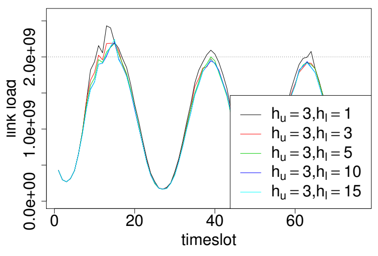
(a) Maximum link load with changing h_u.



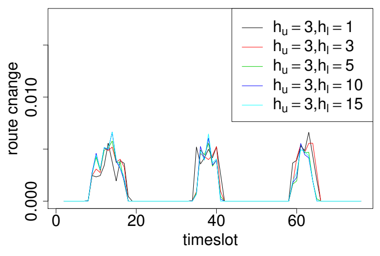
(b) Route changes in the upper layer with changing h_u.



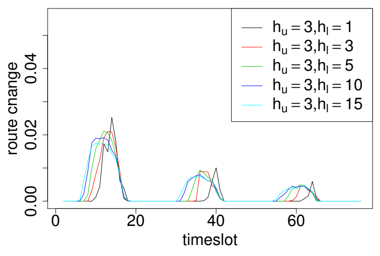
(c) Route changes in the lower layer with changing h_u.



(d) Maximum link load with changing h_l.

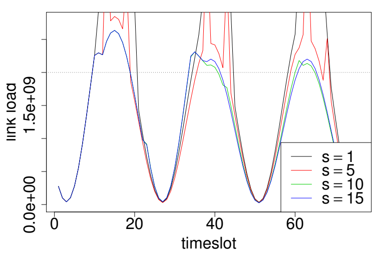


(e) Route changes in the upper layer with changing h_l.

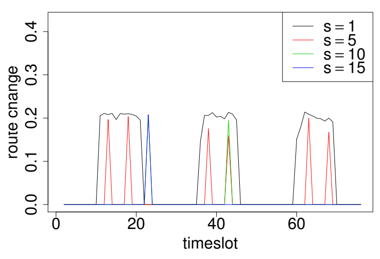


(f) Route changes in the lower layer with changing h_l.

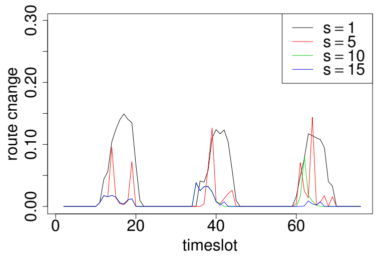
Figure 5.9: Time series of maximum link load and average route changes of MP-TE with various lengths of prediction ($w = 0.8, \sigma = 2.3 \times 10^6$).



(a) Maximum link load.



(b) Route changes in the upper layer.



(c) Route changes in the lower layer.

Figure 5.10: Time series of maximum link load and average route changes of simple TE ($\sigma = 0$).

Impact of Unpredictable Traffic Fluctuation First, we discuss the impact of unpredictable traffic fluctuation. Comparing the cases of $\sigma = 0$ (Figs. 5.6 and 5.8) and $\sigma = 2.3 \times 10^6$ (Figs. 5.7 and 5.9), 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 [66]; 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 5.1 and did not observe a clear difference among these cases.

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. 5.6 and 5.7 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. 5.6 and 5.7 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 $w_l > 0$.

Finally, we investigate the appropriate value of h in hierarchical control. Figs. 5.8 and 5.9 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. 5.8 and 5.9 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.

5.7 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.

Chapter 6

Conclusion

As the Internet and its applications continue to grow, network traffic variation continues to increase. In this thesis, we proposed a prediction-based TE to effectively accommodate such traffic variation. We focused on handling uncertainty in future traffic as it pertained to prediction.

In Chapter 2, we proposed a traffic prediction method to estimate the upper bound of future traffic for TE by setting a safe-side route against prediction uncertainty. In our method, preprocessing separates monitored traffic variation into predictable longer-term variation and noisy short-term variation. A prediction model is then constructed using only predictable variation to improve the predictive accuracy of daily variation. Instead of prediction, the noisy variation is estimated its range. Finally, we obtain the predicted variation and its upper bound, which covers prediction errors and the eliminated noisy variation. Through a simulation involving a real traffic trace, we showed that traffic engineering reduces required link capacity by using predicted traffic. We also clarified that considering the prediction error and noisy variation can avoid congestion due to prediction uncertainty. Moreover, we discussed effectiveness in order to consider periodicity in the prediction model, and found that periodicity should be considered for traffic engineering targeting longer control periods.

It is effective to follow changes in traffic without drastically changing routes to predict traffic trends in the distant future. Such future prediction, however, incurs large prediction errors. In

Chapter 3, we addressed the prediction-based TE method MP-TE that is robust against errors, especially in predictions regarding the distant future. Our approach involved applying the idea of the MPC from system control theory to TE. In this method, the controller calculates the schedule of route changes to avoid large route changes in each time slot. It then sets the calculated routes of the first time slot for the network. Following the route change, the controller monitors traffic information from the network as feedback, and corrects the prediction as well as the consequent future routes. Thus, it reduces the impact of the prediction error. We also evaluated our method using the simulation. The results showed that our TE method can avoid congestion, unlike the simple prediction-based TE method, because of the prediction error. Moreover, we discussed parameter setting, such as the weight of routes change w, the length of the predictive horizon h, and the cycle length of control and prediction. We found that the performance of our method is not very sensitive to parameters w and h. Further, we showed that changing a route at 10-second intervals is sufficient to accommodate traffic changes at every second.

In Chapter 4, we proposed a risk-averse guarantee method by improving the MP-TE. In this method, routes are calculated with a stochastic constraint whereby the probability of congestion should be lower than a designated probability. Due to the characteristic of the prediction error whereby it increases in the distant future, the stochastic constraint increases the number of unnecessary routes changes. Therefore, we also proposed a constraint-relaxation method, in which the guaranteed probability is gradually reduced for the distant future. Through the evaluation involving the actual traffic trace, we showed that the proposed method guarantees lower queuing delays than the original MP-TE. We also showed that the constraint relaxation reduces the frequency of unnecessary routes changes.

In Chapter 5, we proposed the hierarchical MPC-based TE method to achieve scalable control. In hierarchical network control, setting a long control interval in the upper layer is a common approach to induce cooperation among layers. This method, however, delays responses to environmental changes in the upper layer. To solve this problem, we proposed introducing the MPC idea in hierarchical TE once again to predict the behavior of other layers. In this method, the network is hierarchically divided into multiple areas, and multiple MPC controllers are deployed to calculate the routes. To avoid affecting the other areas, each controller gradually changes routes based on the predicted traffic and residual link capacity. Through the simulation, we showed that our proposal achieves routing convergence even by setting the control interval short at the upper layer. We also showed that our proposal reduces the convergence time of hierarchical routing over the existing approach. Moreover, we investigated sensitivity to prediction error, and found that our method works well, even with a large error greater than that incurred in practice. Furthermore, we clarified the appropriate parameter setting of the MPC controller in the hierarchical TE.

In this thesis, we confirmed that the prediction-based TE outperforms observation-based TE, even if there is uncertainty concerning future traffic, by handling uncertainty in a proper manner. Although we focused on the uncertainty of the traffic dynamics in this thesis, there is also uncertainty in the sensing information due to packet loss, faulty monitoring, and so on. One of our future research topics is to handle such incomplete data in prediction-based control. To tackle this topic, an insight into certain neurological mechanisms can provide inspiration, since the brain uses partial information to make decisions in everyday life. Another future research topic is improving the accuracy of prediction about the network state by using not only the traffic information but also other information such as real-world events.

Bibliography

[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 Computer Communications and Networks*, pp. 137–177, 2010.

[3] A. Hassidim, D. Raz, M. Segalov, and A. Shaqed, “Network utilization: The flow view,” in *Proceedings of IEEE INFOCOM 2013*, Apr. 2013, pp. 1429–1437.

[4] S. Iyer, S. Bhattacharyya, N. Taft, and C. Diot, “An approach to alleviate link overload as observed on an IP backbone,” in *Proceedings of IEEE INFOCOM 2003*, Apr. 2003, pp. 406–416.

[5] “Internet2 headroom practice,” available from https://wiki.internet2.edu/confluence/download/attachments/17383/Internet2+Headroom+Practice+8-14-08.pdf?version=1&modificationDate=1239396646935.

[6] 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.

[7] 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.

[8] G. Németh, “On a new competitive measure for oblivious routing,” *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 1, pp. 117–123, Feb. 2014.

[9] 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.

[10] D. O. Awduche, “MPLS and traffic engineering in IP networks,” *IEEE Communications Magazine*, vol. 37, pp. 42–47, Dec. 1999.

[11] 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.

[12] 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.

[13] 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.

[14] S. Agarwal, M. Kodialam, and T. V. Lakshman, “Traffic engineering in software defined networks,” in *Proceedings of IEEE INFOCOM 2013*, Apr. 2013, pp. 2211–2219.

[15] 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.

[16] D. Mitra and Q. Wang, “Stochastic traffic engineering for demand uncertainty and risk-aware network revenue management,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 221–233, Apr. 2005.

[17] lan F. Akyildiz, A. Lee, P. Wang, M. Luo, and W. Chou, “A roadmap for traffic engineering in SDN-OpenFlow netwoks,” *Computer Networks*, vol. 71, pp. 1–30, Oct. 2014.

[18] S. Agarwal, M. Kodialam, and T. V. Lakshman, “Traffic engineering in software defined networks,” in *Proceedings of IEEE INFOCOM 2013*, Apr. 2013, pp. 2211–2219.

[19] M. Roughan, M. Thorup, and Y. Zhang, “Traffic engineering with estimated traffic matrices,” in *Proceedings of ACM SIGCOMM*, Aug. 2003, pp. 248–258.

[20] K. Srijith, L. Jacob, and A. Ananda, “TCP Vegas-A: Improving the performance of TCP Vegas,” *Computer Communications*, vol. 28, no. 4, pp. 429–440, Mar. 2005.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] P. Cortez, M. Rio, and P. Sousa, “Multi-scale Internet traffic forecasting using neural networks and time series methods,” *Expert Systems*, vol. 29, no. 2, pp. 143–155, May 2012.

[28] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, K. Ishibashi, and K. Shiomoto, “Traffic prediction for dynamic traffic engineering considering traffic variation,” *Technical Reports of IEICE(NS2012-115)*, vol. 112, no. 287, pp. 65–70, Nov. 2012.

[29] ——, “Traffic prediction for dynamic traffic engineering considering traffic variation,” in *Proceedings of IEEE GLOBECOM 2013*, Dec. 2013, pp. 1592–1598.

[30] ——, “Traffic prediction for dynamic traffic engineering,” *Computer Networks*, vol. 85, pp. 36–50, July 2015.

[31] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, N. Kamiyama, K. Ishibashi, K. Shiomoto, and T. Hashimoto, “Evaluation of traffic engineering considering traffic prediction,” *Technical Reports of IEICE(IN2013-78)*, vol. 113, no. 245, pp. 7–12, Oct. 2013.

[32] ——, “Evaluation of traffic engineering based on model predictive control using traffic trace in actual network,” *Technical Reports of IEICE(IN2013-194)*, vol. 113, no. 473, pp. 299 – 304, Mar. 2013.

[33] ——, “Traffic engineering based on model predictive control,” *IEICE Transactions on Communications*, vol. E09-B, no. 6, pp. 996–1007, Jun. 2015.

[34] 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.

[35] 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, pp. 683–694, Mar. 2014.

[36] R. Scattolini, “Architectures for distributed and hierarchical model predictive control - a review,” *Journal of Process Control*, vol. 19, no. 5, pp. 723–731, May 2009.

[37] D. Jia and B. Krogh, “Min-max feedback model predictive control for distributed control with communication,” in *Proceedings of 2002 American Control Conference*, May 2002, pp. 4507–4512.

[38] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, K. Ishibashi, K. Shiomoto, and T. Hashimoto, “Evaluation of traffic engineering based on model predictive control using traffic trace in actual network,” *Technical Reports of IEICE(IN2014-81)*, vol. 114, no. 307, pp. 1–6, Nov. 2014.

[39] ——, “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.

[40] T. Hashimoto, “Probabilistic constrained model predictive control for linear discrete-time systems with additive stochastic disturbances,” in *Proceedings of the 52nd IEEE Conference on Decision and Control*, Dec. 2013, pp. 6434–6439.

[41] T. Otoshi, Y. Ohsita, M. Murata, Y. Takahashi, K. Ishibashi, K. Shiomoto, and T. Hashimoto, “Hierarchical traffic engineering based on model predictive control,” *Technical Reports of IEICE(IN2014-136)*, vol. 114, no. 478, pp. 91 – 96, Mar. 2015.

[42] ——, “Hierarchical traffic engineering based on model predictive control,” in *Proceedings of International Conference on Computing, Networking and Communications*, Hawaii, February 2016, pp. 515–521.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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.

[50] 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.

[51] L. Xiang, "A new hybrid network traffic prediction method," in *Proceedings of IEEE GLOBECOM 2010*, Dec. 2010, pp. 1–5.

[52] A. Khosravi, S. Nahavandi, D. Creighton, and A. F. Atiya, "A comprehensive review of neural network-based prediction intervals and new advances," *IEEE Transactions on Neural Networks*, vol. 22, no. 9, pp. 1341–1356, 2011.

[53] D. Kwiatkowski, P. C. Philips, and P. Schmidt, "Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root?" *Journal of Econometrics*, vol. 54, no. 1–3, pp. 159–178, Oct–Dec 1992.

[54] F. Canova and B. E. Hansen, “Are seasonal patterns constant over time? A test for seasonal stability,” *Journal of Business & Economic Statistics*, vol. 13, no. 3, Jul. 1995.

[55] H. Akaike, “Information theory and an extension of the maximum likelihood principle,” in *Proceedings of the Second International Symposium on Information Theory*, 1973, pp. 267–281.

[56] R. J. Hyndman and Y. Khandakar, “Automatic time series forecasting: The forecast package for R,” *Journal of Statistical Software*, vol. 27, no. 3, pp. 1–22, Jul. 2008.

[57] G. Kitagawa and W. Gersch, “A smoothness priors-state space modeling of time series with trend and seasonality,” *Journal of the American Statistical Association*, vol. 79, no. 386, pp. 378–389, Jun. 1984.

[58] “Internet2 data,” available from http://internet2.edu/observatory/archive/data-collections.html.

[59] S. Stoev, G. Michailidis, and J. Vaughan, “On global modeling of backbone network traffic,” in *Proceedings of IEEE INFOCOM 2010*, Mar. 2010, pp. 1–5.

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

[61] 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.

[62] 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.

[63] 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.

[64] K. Papagiannaki, N. Taft, Z.-L. Zhang, and C. Diot, “Long-term forecasting of Internet backbone traffic,” *IEEE Transactions on Neural Network*, vol. 16, no. 5, pp. 1110–1124, Sep. 2005.

[65] S. Han-Lin, J. Yue-Hui, C. Yi-Dong, and C. Shi-Duan, “Network traffic prediction by a wavelet-based combined model,” *Chinese Physics B*, vol. 18, no. 11, pp. 4760–4768, Nov. 2009.

[66] M. Johnston, H.-W. Lee, and E. Modiano, “Robust network design for stochastic traffic demands,” *Journal of Lightwave Technology*, vol. 31, no. 18, pp. 3104–3116, Sep. 2013.

[67] M. S. Lobo, L. Vendenbeghe, S. Boyd, and H. Lebret, “Applications of second-order cone programming,” *Linear Algebra and its Applications*, vol. 284, no. 1, pp. 193–228, Nov. 1998.

[68] 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.

[69] 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.