Virtual Network Control to Achieve Low Energy Consumption in Large-scale Optical Networks

Yuya Tarutani

Graduate School of Information Science and Technology Osaka University

July 2014

List of publication

Journal papers

1. Yuya Tarutani, Yuichi Ohsita, Shin'ichi Arakawa, and Masayuki Murata, 'Optical-Layer Traffic Engineering with Link Load Estimation for Large-Scale Optical Networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 4, no. 1, pp. 38–52, January 2012.

Refereed Conference Papers

1. Yuya Tarutani, Yuichi Ohsita, Shin'ichi Arakawa, and Masayuki Murata, "Estimation of traffic amounts on all links by using the information from a subset of nodes," in *Proceedings of The Second International Conference on Emerging Network Intelligence (EMERGING 2010)*, pp. 18–23, October 2010.

2. Yuya Tarutani, Yuichi Ohsita, and Masayuki Murata, "A Virtual Network to Achieve Low Energy Consumption in Optical Large-scale Datacenter," in *Proceedings of IEEE International Conference on Communication Systems (ICCS 2012)*, pp. 45–49, November 2012.

3. Yuta Shimotsuma, Yuya Tarutani, Yuichi Ohsita, and Masayuki Murata, "Evaluation of data center network structures considering routing methods," in *Proceedings of The Ninth International Conference on Networking and Services (ICNS 2013)*, pp. 146–152, March 2013.

Non-Refereed Technical Papers

1. Yuya Tarutani, Yuichi Ohsita, Shin'ichi Arakawa, and Masayuki Murata, "Selecting monitored links for estimating all link utilizations," *Student Workshop on IEICE Technical Committee on Photonic Network*, March 2010 (in Japanese)

2. Yuya Tarutani, Yuichi Ohsita, Shin'ichi Arakawa, and Masayuki Murata, "Estimation of Traffic Amounts on all Links by Using the Information From a Subset of Nodes," *Technical Report of IEICE (PN2010-27)*, vol. 110, no. 264, pp. 19–24, November 2010 (in Japanese)

3. Yuya Tarutani, Yuichi Ohsita, Shin'ichi Arakawa, and Masayuki Murata, "Optical layer Traffic Engineering using the traffic information from a subset of Nodes," *Student Workshop on IEICE Technical Committee on Photonic Network*, August 2011 (in Japanese)

4. Yuya Tarutani, Yuichi Ohsita, and Masayuki Murata, "A Virtual Network to Achieve Low Energy Consumption in Large-scale Datacenter," *Technical Report of IEICE (PN2012-98)*, vol. 111, no. 475, pp. 91–96, March 2012 (in Japanese)

5. Yuta Shimotsuma, Yuya Tarutani, Yuichi Ohsita, and Masayuki Murata, "Evaluation of data center network structures considering routing methods," *Technical Report of IEICE (IN2012-31)*, vol. 112, no. 88, pp. 43–48, June 2012 (in Japanese)

6. Yuya Tarutani, Yuichi Ohsita, and Masayuki Murata, "Virtual Network Reconfiguration with Fault-tolerance and Low Energy Consumption for Multi-Tenant Data Centers," *Technical Report of IEICE (IN2013-193)*, vol. 113, no. 473, pp. 293–298, March 2014 (in Japanese)

7. Yuya Tarutani, Yuichi Ohsita, and Masayuki Murata, "Virtual Network Configuration Method for Low Energy Consumption in Data Centers," *IEICE General Conference*, March 2014 (in Japanese)

Preface

In recent years, various new applications, such as peer-to-peer, video on demand, and cloud computing have been deployed over the Internet. These new application leads to sudden and significant changes in traffic volume. Network providers must handle such significant traffic changes. One approach to handling such traffic changes is to construct the network that has the enough bandwidth to accommodate any traffic. However, the network with a large bandwidth consumes a large amount of energy, and the energy consumption of the network has becomes one of the important problem in the Internet.

One approach to reducing the energy consumption of the network is to use optical circuit switches (OCSs) and electronic switches. Electronic switches are connected to the OCSs. In such a network, a virtual network can be configured by setting the OCSs to connect different ports of the electronic packet switches. Thus, the energy consumption of the network can be reduced by configuring the virtual network to minimize the number of ports required by the electronic packet switches under the constraints that the current traffic can be accommodated, and powering down any unused ports.

However, there are several problems when performing the virtual network reconfiguration in a large-scale network. The first problem is the overhead for collecting the current traffic information. The current traffic information in the whole network is required as inputs of the virtual network reconfiguration method. However, as the number of nodes in the network increases, the overhead for collecting the traffic volume information becomes large. As a result, it is difficult to obtain the traffic information within a short time interval. Another problem is the calculation time to obtain the suitable topology of the virtual network. The existing methods require a large calculation time

compared with the interval of the traffic changes.

In this thesis, we propose methods to overcome the above problems. First, we develop a method that reduces the overhead for collecting traffic information by selecting a subset of nodes and by only collecting the traffic information from the selected nodes. Then, we estimate the traffic volumes in the whole network using the information gathered from the selected nodes. According to the simulation results, we clarify that our method can accurately identify the congested links, where the number of traffic demands passing through some links is large; however, the estimation errors of our method become large when the number of traffic demands passing each link is small. Furthermore, the virtual network reconfiguration method can sufficiently mitigate congestion by using the traffic volume estimated by our method from the information of 50% of all nodes in the case of Japan topology and 30% of all nodes in the case of AT&T topology.

Next, we propose a method called VNR-DCN that immediately reconfigures the virtual network so as to reduce the energy consumption under the constraints on the bandwidth and delay between servers in data center networks based on optical communication paths. In VNR-DCN, we configure the virtual network to satisfy the requirements by setting the parameters of the topology, called GFB, instead of solving an optimization problem. In the evaluation, we show that a virtual network configured by VNR-DCN requires a small number of active ports. In addition, we also show the impact of virtual network configuration on energy consumption.

Finally, we also propose a method to place the virtual machines (VMs) in a multi tenant data center that can accommodate multiple services. In a multi tenant data center, each service provider requests the accommodation of multiple VMs required by the service, and the data center provider places the VMs based on the requests. The placement of the VMs has a large impact on the nodes to be powered on. Thus, to save the energy consumption, the placement of the VM is important. In the services of the data center, the robustness against failures is another important problem; no data should be lost even in case of failures. One approach to achieving the robustness is to introduce redundancy; each chunk, which is a fraction of data, is exchanged between the VMs, and is replicated to k VMs out of all placed VMs. However, none of existing work considers the number of chunk replicas. Therefore, we propose a method to place the VMs considering the number of chunk replicas. Through numerical simulation, we show that our method keeps all data available even in the case of any single point of failure with low energy consumption.

Acknowledgments

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

I would like to express my deepest appreciation to my supervisor, Professor Masayuki Murata of Graduate School of Information Science and Technology, Osaka University for his encouragement, valuable discussions, and meaningful advices throughout my Ph.D.

I would like to express my deep appreciation to 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.

Also, I must express my deep appreciation to Assistant Professor Yuichi Ohsita of Graduate School of Information Science and Technology, Osaka University, for his continuous support, appropriate guidance, and invaluable direct advice. His kindness on my behalf were invaluable, and I am forever in debt.

Furthermore, I express my appreciation to Professor Naoki Wakamiya, Associate Professor Shin'ichi Arakawa, Associate Professor Go Hasegawa, Assistant Professor Yoshiaki Taniguchi, Assistant Professor Yuki Koizumi of Graduate School of Information Science and Technology, Osaka University, and Assistant Professor Daichi Kominami of Graduate School of Economics, 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.

Last, but not least, I thank my parents for their invaluable support and constant encouragement during my undergraduate and doctoral studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In recent years, various new applications, such as peer-to-peer, video on demand, and cloud computing have been deployed over the Internet. These new applications lead to sudden and significant changes in traffic volume. Network providers must handle such significant traffic changes. One approach to handling such traffic changes is to construct the network that has the enough bandwidth to accommodate any traffic. However, the network with a large bandwidth consumes a large amount of energy, and the energy consumption of the network has becomes one of the important problem in the Internet [1-3].

Optical network technologies are one of the promising approach to reducing the energy consumption of the network. An optical network is constructed of optical circuit switches (OCSs) and optical fibers connecting the optical switches. Each OCS relays the input optical signal to the predefined output port without any electronic processing by reflecting or refracting the input signal. Therefore, the optical network relays a large amount of traffic with only small energy consumption.

The optical network requires the configuration of the OCSs before relaying the traffic, and the packet level switching is difficult; there is still no commercial optical packet switch though several optical packet switch architectures have been proposed [4, 5]. Therefore, the optical network technology is often combined with the electronic network technologies. Figure 1.1 shows the optical

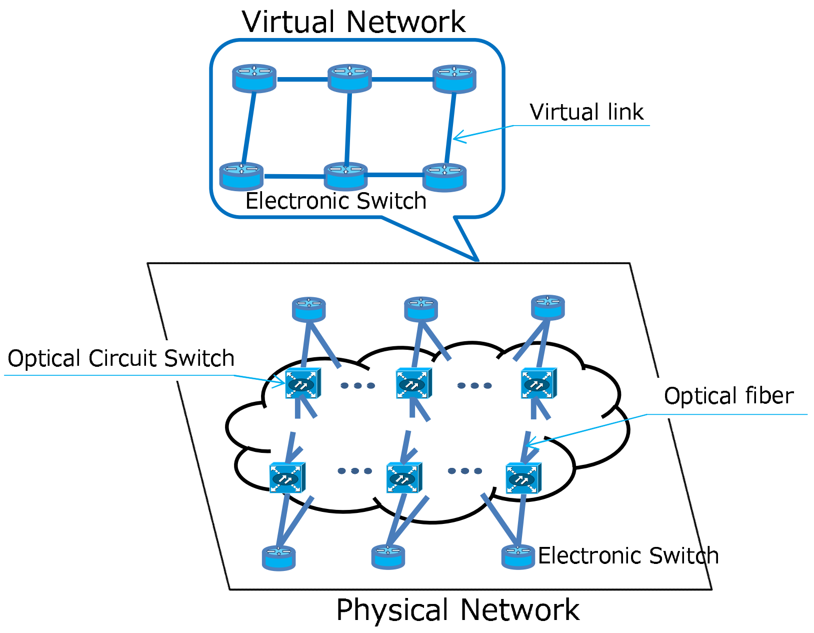


Figure 1.1: Network Architecture Using OCSs

network architecture combined with the electronic network technologies. In this network, optical transceivers are equipped with the ports of the electronic switches, and the optical transceivers are connected to the OCS. In this network, the virtual link can be established by setting the OCSs along the routes between the electronic switches. The virtual link is regarded as a directly connected high-capacity link for the electronic switch. The set of the virtual links forms the virtual network of the electronic switches, and the virtual network can be reconfigured by adding or removing virtual links.

Many methods to reconfigure the virtual network have been proposed [6-8]. These methods reconfigure the virtual network to make the maximum link utilization less than the threshold by adding and deleting virtual links based on current traffic information.

The virtual network reconfiguration can also save the energy consumption. The energy can be saved by minimizing the number of powered-on electronic switches or ports, because the electronic

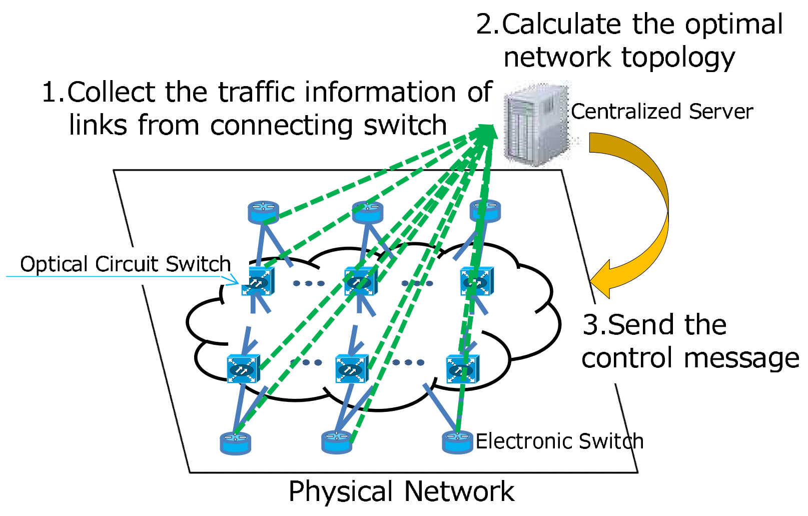

2.Calculate the optimal network topology
1.Collect the traffic information of links from connecting switch
Centralized Server
Optical Circuit Switch
3.Send the control message
Electronic Switch
Physical Network

Figure 1.2: Virtual Network Reconfiguration

switches consume more energy than the OCSs. The dynamic reconfiguration of the virtual network is a promising way to reduce the energy consumption, because the traffic demands change in time; a large amount of traffic arrives in the peak hours, but only a small amount of traffic arrives at midnight. Therefore, several methods [9-11] that dynamically configure the network so as to minimize the number of powered-on network devices under the constraint that the current traffic is accommodated without congestion.

In the virtual network reconfiguration, the virtual network is calculated and controlled with a control server as shown Figure 1.2. The virtual network reconfiguration is performed by performing the following steps periodically. First, the control server collects the traffic information from the network. Then, the control server calculates the suitable virtual network topology based on current traffic information. Finally, the control server configures the OCSs in the network.

The control period of the virtual network reconfiguration for the energy saving should be small, because the network constructed of the minimum number of switches and ports can be easily congested by the traffic changes. Therefore, the control server needs to collect traffic information frequently so as to detect the traffic change. If the traffic change is detected, the virtual network must be reconfigured immediately. However, to perform such a frequent collection of traffic information and immediate reconfiguration, existing virtual network reconfiguration processes have the following problems.

The first problem is the overhead to collect the traffic information. Collecting all traffic demands requires monitoring overheads at each node; i.e., a packet-header inspection is necessary to identify the destination node. In addition, because the centralized server needs to collect information about the traffic demand from every node, a collecting overhead is required. The CPU loads and the bandwidth required for collecting the traffic demands increase with the number of nodes in the network [12]. One reason for this is that the centralized server must query all nodes to retrieve the traffic information.

Another problem is the calculation time to obtain the suitable topology of the virtual network. In the virtual network reconfiguration, the centralized server calculates the optimization problem so as to obtain the optimal virtual network topology. However, the optimization problem of constructing the virtual network requires a long calculation time, because it requires to find the best combination of $N(N-1)$ binary variables, where N is the number of nodes. Thus, the heuristic method is required. However, the heuristic method to obtain the suitable network topology that minimizes the energy consumption has not been discussed sufficiently.

Therefore, this thesis proposes methods toward the frequent virtual network reconfiguration aiming at the reduction of the energy consumption that overcome the above problems.

1.2 Outline of Thesis

Reducing the Collecting Overhead with Link Load Estimation for Large-Scale Optical Networks [13-17]

In chapter 2, we propose a method to reduce the overhead for collecting the necessary traffic information for the virtual network reconfiguration method by estimating the traffic matrix from the traffic information collected from a subset of nodes. In this method, we select the nodes and collect the traffic volume information on each link from the selected nodes without using the past traffic volume information. Then, we estimate the traffic matrix by using the information collected from the selected nodes, the uncollected traffic volume information on each link by using the estimating traffic matrix. In this method, the accuracy of the estimated traffic matrix depends on the selection of nodes. Therefore, we propose a method for selecting the source node to avoid large estimation errors of the traffic matrix. According to the simulation results, we clarify that our method can accurately identify the congested links in real ISP topologies, where the number of traffic demands passing through some links is large; however, the estimation errors of our method become large when the number of traffic demands passing each link is small. Furthermore, the virtual network reconfiguring method can sufficiently mitigate congestion by using the traffic volume estimated by our method from the information of 50% of all nodes in the case of Japan topology and 30% of all nodes in the case of AT&T topology.

Virtual Network Reconfiguration for Reducing Energy Consumption in Optical Data Centers [18-22]

In chapter 3, we focus on the data center network as one of the example of the large network. A data center is constructed of many servers and network connecting the servers. In a data center, the network within the data center plays an important role, because the servers provide the service by cooperating with each other; the lack of bandwidth or large delay degrades the performance of the service. The contribution of the network itself to the total energy consumption of the data center can also be significant, reaching 10-20% of the total energy consumption [23] and increasing as the size and speed of the network increase. Therefore, reducing the energy consumption of data center networks would be essential for constructing energy-efficient data centers [24-26].

There have been many studies on the construction of data center network topologies (e.g., [27-38]) , but no single network topology can achieve both of the small energy consumption and the sufficiently large bandwidth.

The dynamic reconfiguration of the network topology is one of the promising approach to reducing the energy consumption. A method allowing the topology to change dynamically by powering down the ports of switches has been proposed by Heller et al. [11]. In that method, the ports of switches are powered down if sufficient bandwidth can be provided without them. However, that approach powers down only a limited number of ports because it cannot change the network topology drastically, even if there exists a network topology requiring only a small number of ports and able to accommodate the instantaneous traffic demand. Therefore, a notable reduction in energy consumption can be achieved if the network topology can be changed flexibly based on changes in traffic demand.

Therefore, we propose *Virtual Network Reconfiguration for Data Center Networks (VNR-DCN)*, which satisfies the above requirements. The VNR-DCN obtains the suitable virtual network by setting the parameters of the *Generalized Flattened Butterfly (GFB)*, which is a new topology proposed in this thesis where various types of the network topologies can be reproduced by setting only a small number of parameters, instead of solving the optimization problem. By setting a small number of parameters, we obtain the suitable virtual network topology in a short time. The VNR-DCN sets the parameters of the GFB so as to accommodate the total amount of current traffic, considering the load balancing combined with the routing for GFB. By this approach, the changes of the total amount of traffic are handled by reconfiguring the virtual network, while the frequent changes in traffic pattern are handled by the load balancing over the virtual network without frequent reconfiguration of the virtual network. Through numerical simulation, we show that the VNR-DCN achieves lower energy consumption than an energy saving method using only electronic switches.

Virtual Machine Placement Considering the Redundancy of Data in Multi tenant Data Centers [39]

In chapter 4, we focus on the placement of virtual machines (VMs) in the multi tenant data centers. Because constructing a data center for each service would incur an unacceptably large cost, multi tenant data centers, such as Amazon EC2 [40], that can accommodate multiple services based on the requests from service providers have been built and put into operation.

In multi tenant data centers, the placement of the VMs has a large impact on the nodes to be powered on. Thus, to save the energy consumption, the placement of the VM is important. The optimal placement of VMs combined with assigning paths between the VMs has been formulated as the virtual network embedding (VNE) problem, and has been discussed in many papers [41-54].

The robustness against failures is another important problem in data centers; all data should be kept even in case of failures. One of the popular approaches to achieving robustness against failures is to introduce redundancy. Google file system (GFS) [55] or Hadoop file system [56] stores each chunk, which is a fraction of data, is replicated to *k* VMs of all hosted VMs, where *k* is a parameter indicating the number of chunk replicas. In this approach, VMs storing the replicas of each chunk depends on the chunk. By this approach, all data can be obtained from available VMs unless more than *k* VMs fail. Setting *k* to a large value makes the service more robust against failures; all data become available in case of failure of a large number of VMs. However, setting *k* to a large value causes inefficient use of resources; storing a large number chunk replicas requires a large amount of storage of the VMs and a large bandwidth between VMs. The placement of the VMs is closely related to the suitable value of *k*; *k* should be larger than the number of VMs that can be unavailable simultaneously, while the number of VMs that can be unavailable by a failure depends on the placement of the VMs.

Therefore, we propose a method to place the VMs so as to minimize the energy consumption, considering the number of chunk replicas. The placement of the VMs should be obtained in a short time, because after a failure, we should reallocate the VMs before the next failure occurs. Therefore, we propose a heuristic method. In this heuristic method, we define the *critical flow* for each node, which is the flow whose connection becomes unavailable when the node fails. Based on the information of the critical flows for a certain node, the number of VMs that would be disconnected by the failure of the node is calculated. Therefore, our method places the VMs so that any node failure does not disconnect more than *k* VMs by using the information of the critical flows.

By using our heuristic method, we also investigate the cost of considering the number of chunk replicas. If *k* is set to a small value, more servers may be required to make the number of VMs that would be disconnected less than *k*, because the VMs should be placed at the different places. This may cause large energy consumption by powering on more servers. However, our results demonstrate that the cost of considering the number of chunk replicas becomes small as the number of tenants increases, and even when we set *k* to a small value, we can achieve the sufficient robustness without causing a large energy consumption.

Chapter 2

Reducing the Collecting Overhead with Link Load Estimation for Large-Scale Optical Networks

2.1 Traffic matrix estimation

2.1.1 Overview

Traffic matrix is the matrix of $T_{s,d}$ that represents the traffic volume from nodes s to d. Let N be the number of nodes in the network. Then the traffic matrix is represented as follows:

$$T = \begin{bmatrix} T_{1,1} \\ T_{1.2} \\ \vdots \\ T_{N,N} \end{bmatrix}. \tag{2.1}$$

As this equation indicates, for obtaining the traffic matrix, the traffic information between all nodes is required. The overhead to collect the traffic information increases it as the number of nodes. Therefore methods have been investigated to estimate a traffic matrix from the traffic volume on each link, which is determined from routing information A, and traffic matrix T. Routing information A is known to the network administrator and traffic matrix T is unknown. The following equation holds:

$$AT = X, \tag{2.2}$$

where X is a matrix of X_i that represents the traffic volume passing through link i:

$$X = \begin{bmatrix} X_1 \\ \vdots \\ X_L \end{bmatrix}. \tag{2.3}$$

In the above equation, L is the number of links in the networks. A is a routing matrix with element $A_{s,d,l}$ representing the route of traffic between nodes s and d; when the traffic passes through link l, $A_{s,d,l}$ is one, otherwise it is zero. A is represented as follows:

$$A = \begin{bmatrix} A_{1,1,1} & A_{1,2,1} & \cdots & A_{N,N,1} \\ A_{1,1,2} & A_{1,2,2} & \cdots & A_{N,N,2} \\ \vdots & \vdots & \ddots & \vdots \\ A_{1,1,L} & A_{1,2,L} & \cdots & A_{N,N,L} \end{bmatrix}. \tag{2.4}$$

Note that when we consider the splittable flow, $A_{s,d,l}$ is the rate of traffic demand $T_{s,d}$ conveyed through link l.

Traffic matrix estimation is an approach to estimate T satisfying Eq. (2.2) on the basis of matrix X and routing matrix A. Therefore, we collect the traffic volume on each link to create matrix X. We cannot estimate the unique traffic matrix satisfying Eq. (2.2) because the number of equations in Eq. (2.2) is usually lesser than the number of elements in T that is, several traffic matrix candidates satisfy Eq. (2.2). Therefore, the traffic matrix estimation methods [57-62] obtain the estimated traffic matrix that is close to the true traffic matrix from the candidates by using a model of the traffic matrix.

One of the challenges of traffic matrix estimation is to accurately estimate the traffic matrix from as little traffic volume information as possible. We can reduce the collecting overhead by reducing the traffic volume information as much as possible without degrading the performance of the virtual network reconfiguration method using the estimated traffic matrix. However, the traffic volume information used by traffic matrix estimation has rarely been discussed. In this chapter, we develop a method for estimating the traffic matrix from the information of a subset of nodes and discuss the traffic volume information required to estimate the traffic matrix accurately enough to perform the virtual network reconfiguration method.

2.1.2 Related Work

Many approaches have estimated the traffic matrix using a traffic matrix model [57-62]. For example, Zhang et al. [61] developed an estimation scheme called the *tomogravity method* that estimates traffic matrices by following a gravity model in which the traffic volume between two nodes is proportional to the product of their traffic. The tomogravity method works as follows. First, it estimates traffic demand $T^{grav}_{s,d}$ on the basis of the monitored traffic in the ingress and egress links in order to follow the gravity model by the following equations:

$$T^{grav}_{s,d} = X_{l^{in}_s} \frac{X_{l^{out}_d}}{\sum_k X_{l^{out}_k}}, \tag{2.5}$$

where l^{in}_s is the ingress link at node s and l^{out}_d is the egress link at node d. We denote T^{grav} as the matrix in which each entry is $T^{grav}_{s,d}$. Then the tomogravity method estimates traffic matrix $\hat{T}$ by the following equations:

$$\begin{aligned} &\min \|\hat{T} - T^{grav}\|, \\ &s.t.\ A\hat{T} = X. \end{aligned} \tag{2.6}$$

That is, this method calculates $\hat{T}$ satisfying Eq. (2.2) and minimizes the difference between $\hat{T}$ and T^{grav}. Although the traffic information required for the tomogravity method is much smaller than that needed for directly collecting traffic matrix information, L numbers of traffic information must still be collected to estimate traffic matrices.

One approach to reducing the collecting overhead is to divide the collection of traffic information into multiple steps. In this approach, the centralized server collects the traffic information from a subset of nodes in each step and constructs the traffic information of all links from the traffic information collected at all steps. However, this approach cannot follow the traffic changes that occur during the traffic information collection steps.

Zhang et al. [62] developed a more sophisticated method in which they calculated the correlation between each bit of traffic volume information collected at different times or different points. Then they estimated the uncollected traffic volume information using the correlation. However, this method cannot accurately estimate the uncollected traffic volume information when a traffic change that is different from past tendencies occurs.

2.1.3 Our Approach

In this chapter, we develop a method that estimates the traffic matrix from the information collected from a subset of nodes without using the past traffic volume information. Before estimating the traffic matrix, we estimate the uncollected traffic volume on each link and create matrix X' indicating the roughly estimated traffic volume on each link as follows

$$X' = \begin{bmatrix} X'_1 \\ \vdots \\ X'_L \end{bmatrix}, \tag{2.7}$$

where

$$X'_l = \begin{cases} X_l & \text{if } l \text{ is the link connected} \\ & \text{to the source nodes,} \\ U_l & \text{otherwise.} \end{cases} \tag{2.8}$$

In the above equation, X_l is the monitored traffic volume on link l and U_l is the traffic volume on link l estimated from the collected traffic volume information of other links.

Then we estimate the traffic matrix to match X' and A by minimizing the following equation:

$$\min \|A\hat{T} - X'\|. \tag{2.9}$$

There are two challenges in our approach. The first is estimating U_l from the monitored traffic volume on other links without the past traffic volume information. We estimate U_l using the relationship between the number of traffic demands that pass a link and the traffic volume on it. The details of our estimation method are described in Section 2.2. Another challenge is selecting the source node. The accuracy of the estimated traffic matrix depends on the selection of the source node. Thus, in our method, we select the source node to avoid large estimation errors of the traffic matrix. The details of the method to select the source nodes are described in Section 2.3.

2.2 Traffic matrix estimation from subset information of nodes

In this section, we introduce a method that estimates the traffic matrix using the traffic volume monitored at a subset of the nodes. Figure 2.1 shows the steps of our method. In Step 1, we select source nodes and collect the traffic information from them. In this step, the source nodes are randomly selected. The details of Steps 2 and 3 are described below.

2.2.1 Estimation of traffic volume using the number of traffic demand

In this approach, we only use the traffic volume information monitored at the selected source nodes. However, because the lack of traffic volume information complicates the estimation of traffic matrices, we estimate the uncollected traffic volume information before estimating the traffic matrix.

2.2.1.1 Method to estimate traffic volumes

The traffic volume on each link is proportional to the number of traffic demands passing the link unless the variance of the traffic demands is large. Thus, we model the relationship between the

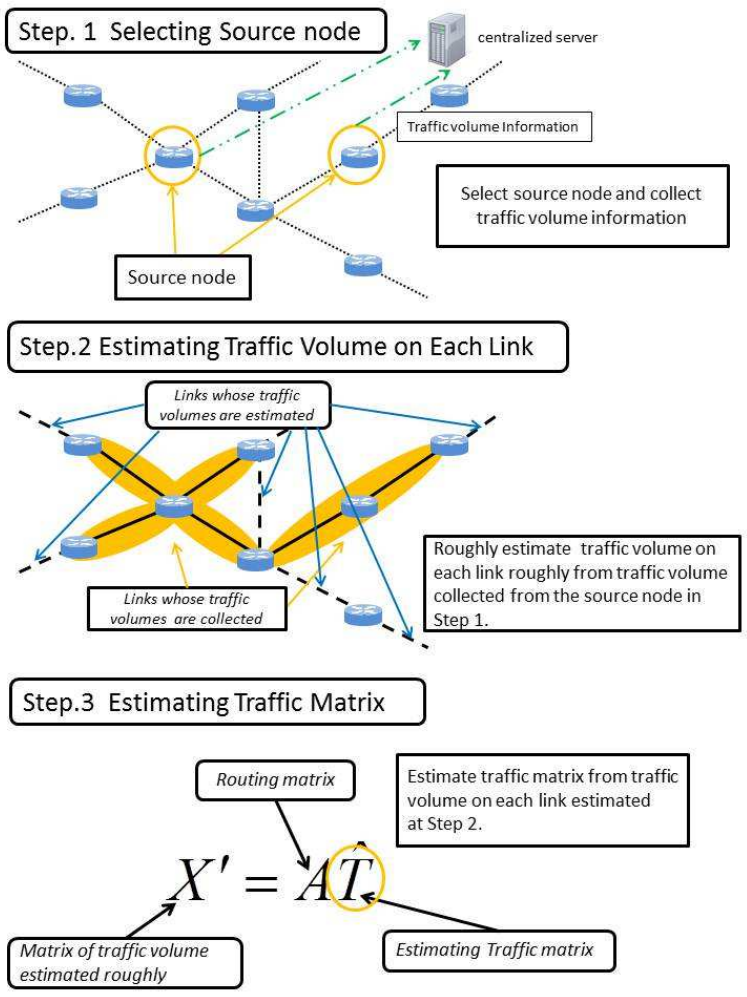

Step. 1 Selecting Source node
centralized server
Traffic volume Information
Select source node and collect traffic volume information
Source node
Step.2 Estimating Traffic Volume on Each Link
Links whose traffic volumes are estimated
Roughly estimate traffic volume on each link roughly from traffic volume collected from the source node in Step 1.
Links whose traffic volumes are collected
Step.3 Estimating Traffic Matrix
Routing matrix
Estimate traffic matrix from traffic volume on each link estimated at Step 2.
X′ = AT̂
Matrix of traffic volume estimated roughly
Estimating Traffic matrix

Figure 2.1: Flowchart for traffic matrix estimation

number of traffic demands passing a link and its traffic volume as follows

$$W_i = \alpha Z_i + \beta, \tag{2.10}$$

where W_i is the traffic volume on link i, Z_i is the number of traffic demands passing link i, and α and β are constant parameters. Z_i for any node i can be calculated from the routing matrix.

With this relation, we can estimate the traffic volume on each link in the following steps. First, we calculate the constant parameters α and β using the traffic volume on each link collected from the selected source nodes with the least-square method:

$$\alpha = \frac{|S| \sum_{i \in S} Z_i W_i - \sum_{i \in S} Z_i \sum_{i \in S} W_i}{|S| \sum_{i \in S} Z_i^2 - (\sum_{i \in S} Z_i)^2}, \tag{2.11}$$

$$\beta = \frac{\sum_{i \in S} Z_i \sum_{i \in S} W_i - \sum_{i \in S} Z_i W_i \sum_{i \in S} Z_i}{|S| \sum_{i \in S} Z_i^2 - (\sum_{i \in S} Z_i)^2}, \tag{2.12}$$

where S is the set of links connected to the source nodes. Then we estimate traffic volume U_j on link j that is not collected from the source nodes as follows:

$$U_j = \alpha Z_j + \beta. \tag{2.13}$$

Finally, we create matrix X' defined by Eq. (2.7) by using the estimated volume of traffic U_j.

2.2.1.2 Validation of the model used to estimate the traffic volume

We validate the relationship modeled by Eq. (2.10) using a simulation. In this simulation, we use AT&T's router-level topology (523 nodes and 1304 links) measured in Reference [63]. We add one ingress link and one egress link for all nodes in the AT&T topology and generate traffic between each pair of ingress and egress links.

According to Reference [64], each element of the actual traffic matrices obeys a lognormal distribution. Thus, in this simulation, we generate traffic matrix T to follow a lognormal distribution as follows:

$$T = \Theta \left(T^{init} + \Delta \right), \tag{2.14}$$

where T^{init} is a traffic matrix generated to follow a lognormal distribution, Δ is a matrix indicating the white Gaussian noise with a mean of 0 and a variance of 1, and Θ is a scale parameter. We generate $T^{init}_{i,j}$ on the basis of the lognormal distribution with $\mu = 16.6$, $\sigma = 1.04$ to match the results described in Reference [64]. The unit of the traffic volume of the traffic demand generated by the above steps is Mbps and Θ is set to 0.1 so that the generated traffic can be accommodated in the topology used in this simulation.

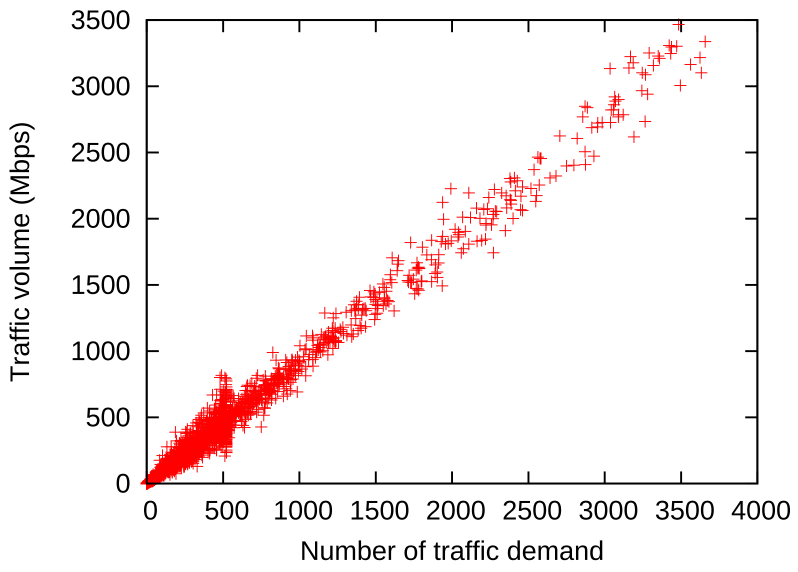


Figure 2.2: Relationship between traffic demand and traffic volume on each link in the case of AT&T topology (μ = 16.6, σ = 1.04)

Figure 2.2 shows the relationship between the number of traffic demands passing a link and the traffic volume on the link obtained by our simulation. According to Figure 2.2, the model described by Eq. (2.10) fits the traffic volume on each link when the traffic demands are generated using the same parameters as the actual traffic demand monitored at the real network in Reference [63]. Therefore, Eq. (2.13) is considered to estimate the traffic volume on each link without the large estimation errors found in real networks.

The difference between the actual traffic volume and the traffic volume modeled by Eq. (2.10) becomes large as the variance of the traffic demands increases. This difference leads to the estimation errors of the traffic volumes on some links. However, if we obtain sufficient traffic volume information from the source nodes, the estimation errors of the traffic volumes on some links do not have large impacts on the accuracy of the traffic matrix estimated by the steps described in subsection 2.2.2. The accuracy of the estimated traffic matrix when the variance of the traffic demands is large is discussed in Section 2.4.2.2.

2.2.2 Estimation of the traffic matrix

We estimate the traffic matrix from the roughly estimated traffic volume on each link. If we apply the tomogravity method to estimate the traffic matrix from the estimated traffic volume on each link, estimation errors may become large. The errors included in the traffic volume on the ingress and egress links cause the inaccurate estimation of T^{grav} and the large estimation errors of the tomogravity method, even when the traffic volumes on other links are estimated accurately. Therefore, we need a traffic matrix estimation method in which the estimation errors included in the traffic volumes on particular links do not significantly affect the estimation results.

Although a more sophisticated estimation method may exist, in our evaluation described in Section 2.4, we use a simple approach to estimate the traffic matrix by minimizing the following equation:

$$\min \|X' - A\hat{T}\|. \tag{2.15}$$

The results shown in Section 2.4 clarify that we can accurately identify the congested links and perform the virtual network reconfiguration method to mitigate the congestion by using the estimated traffic matrix, even with the simple approach to estimate the traffic matrix.

2.2.3 Accuracy of the estimation

We evaluate the accuracy of the estimation by simulation. The source nodes are randomly selected. We use Japan topology (49 nodes and 91 links) and set the routes of packets by the shortest path first (SPF) algorithm.

2.2.3.1 Metric

We use two metrics to evaluate the accuracy of the estimation.

Accuracy of the estimated traffic matrix We use the Root Mean Squared Error (RMSE) to evaluate the accuracy of the estimated traffic matrix. RMSE (T_{RMSE}) of the traffic matrix is defined as follows:

$$T_{RMSE} = \sqrt{\frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\hat{T}_{i,j} - T_{i,j}\right)^2}, \tag{2.16}$$

where N is the number of nodes in the network, $\hat{T}_{i,j}$ is the estimated traffic demand between nodes i and j, and $T_{i,j}$ is the actual traffic demand between nodes i and j. In our evaluation, we show RMSE normalized by the average amount of traffic demand instead of the absolute value of RMSE in order to compare the accuracy of the estimated traffic matrix in the different environments.

Accuracy of the estimated traffic volume on each link The traffic volume on each link is important information for the virtual network reconfiguration method because inaccurate traffic volume on each link may cause misidentification of the congested links. Thus, we also investigated the accuracy of the traffic volume on each link estimated from the estimated traffic matrix $\hat{T}$ and the routing matrix A. The volume on each link $\hat{X}$ is estimated as follows:

$$\hat{X} = A\hat{T}. \tag{2.17}$$

Similar to the accuracy of the estimated traffic matrix, we use the RMSE to evaluate the estimated volume of traffic on each link. RMSE (X_{RMSE}) of the estimated volume of traffic on each link is defined as follows:

$$X_{RMSE} = \sqrt{\frac{1}{L} \sum_{k=1}^{L} \left(\hat{X}_k - X_k\right)^2}, \tag{2.18}$$

where L is the number of links in the network, $\hat{X}_k$ is the estimated traffic volume on link k, and X_k is the actual traffic volume on link k. In our evaluation, we show RMSE normalized by the average amount of traffic volume on each link instead of the absolute value of RMSE for comparing the accuracy of the estimated volume of traffic on each link in different environments.

2.2.3.2 Simulation result

Figures 2.3 and 2.4 show T_{RMSE} normalized by the average amount of traffic demand and X_{RMSE} normalized by the average volume of traffic on each link when we change the number of source

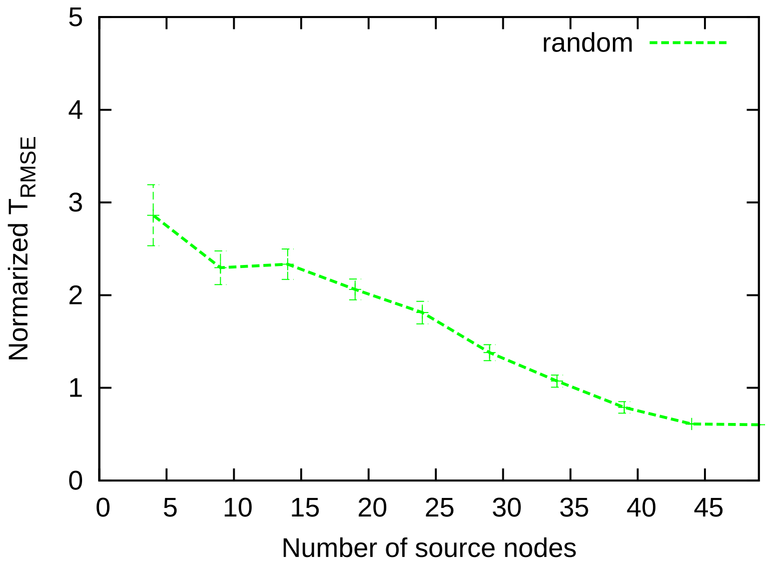


Figure 2.3: RMSE of the amount of traffic demand in the case of Japan topology ($\mu = 16.6$, $\sigma = 1.04$)

nodes, respectively. In these figures, the vertical axes are normalized T_{RMSE} and normalized X_{RMSE}, and the horizontal axis is the number of source nodes. We plot the results for the average of 20 cases randomly selected from the source nodes with the error bar representing the 95% confidence interval. According to Figures 2.3 and 2.4, the estimation errors become large as the number of source nodes reduces. This is because a large number of traffic demands do not pass any randomly selected source nodes. Furthermore, the traffic demand that does not pass any source nodes is difficult to estimate because we cannot obtain any information of the traffic passing no source nodes. Therefore, we should select the source nodes so that maximum possible traffic demands pass at least one of the source nodes in order to accurately estimate the traffic matrix from the small subset of nodes.

2.3 Selecting source nodes

In this section, we propose a method for selecting the source nodes to avoid large estimation errors of the traffic matrix. In our method, we select source nodes so that maximum possible traffic

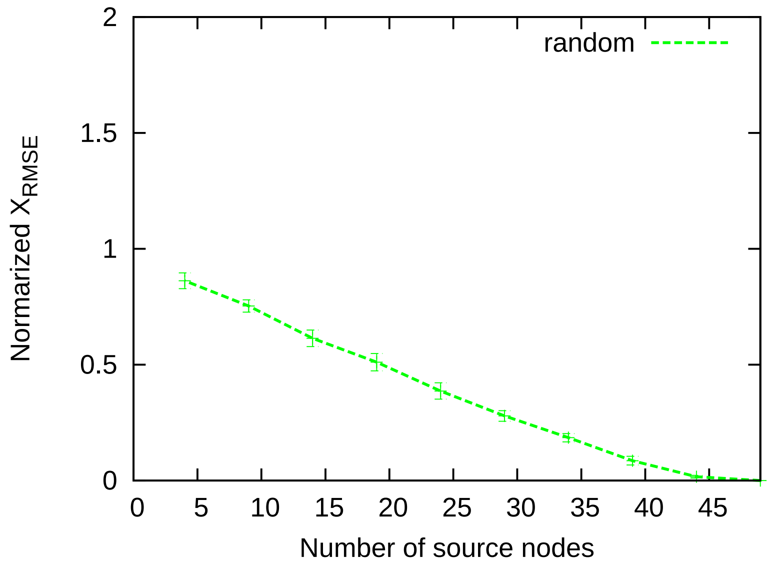


Figure 2.4: RMSE of the traffic volume on each link in the case of Japan topology (μ = 16.6, σ = 1.04)

demands pass at least one of the source nodes.

2.3.1 Method to select source nodes

Our method selects the source nodes by the following steps. First, we set all nodes as candidates for the source nodes. Then we eliminate the nodes passed only by the traffic passing the other candidates from the candidates in the *elimination phase*. Finally, we select the source nodes from the candidates in the *selection phase*.

The details of the elimination phase and the selection phase to select N source nodes are described below.

2.3.1.1 Elimination phase

We eliminate the nodes only passed by the traffic demands passing the other candidates. By eliminating such nodes from the candidates, we can reduce the number of candidates without reducing the number of traffic demands passing at least one of the candidates of the source nodes.

We use the following variables: Q_i is the number of traffic demands that does not pass any other candidates except node i and $L_{n,m}$ is the number of candidates that passed by the traffic demand from node n to node m. We select the nodes to be eliminated on the basis of the number of traffic demands passing node i, P_i. In the elimination phase, our method eliminates the candidates by the following steps:

Step 1.1 Initialize Q_i to 0, and $L_{n,m}$ to the number of nodes that passed by the traffic demand from nodes n to m.

Step 1.2 Eliminate node x whose P_i is the smallest among the candidates whose Q_i is 0 from the candidates.

Step 1.3 Update $L_{n,m}$ by decrementing $L_{n,m}$ whose corresponding traffic demand passes node x.

Step 1.4 Update Q_i for all candidates by counting the elements of $L_{n,m}$ whose value is 1 and whose corresponding traffic passes node i. If a node whose Q_i is 0 exists, go to Step 1.5, otherwise go to Step 1.6.

Step 1.5 If the number of candidates is larger than N, go back to Step 1.2. Otherwise, go to Step 1.6.

Step 1.6 End.

2.3.1.2 Selection phase

In the selection phase, we select the source nodes so that maximum possible traffic demands pass at least one of the source nodes. In our method, we use a greedy approach that iteratively selects the source nodes so as to maximize the number of traffic demands passing the selected source nodes.

In each iteration of the selection phase, we select the source nodes by using the number of traffic demands that passes node i and does not pass the currently selected source nodes, R_i.

In the selection phase, we select the source nodes by the following steps.

Step 2.1 If the number of candidates is less than N, select all nodes in the candidates and go to Step 2.6. Otherwise go to Step 2.2.

Step 2.2 Initialize R_i to P_i.

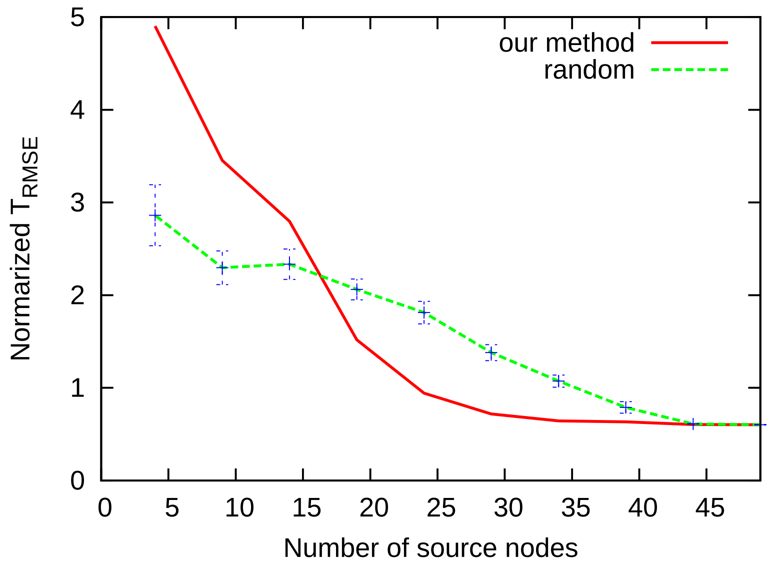


Figure 2.5: RMSE of the traffic matrix in the case of Japan topology (μ = 16.6, σ = 1.04)

Step 2.3 Select node x whose R_x is the largest among the candidates. If there are multiple candidates whose R_x are the largest, select node x with the largest P_i.

Step 2.4 Check the route of each traffic demand passing selected node x. Then update R_i by decrementing its value if a traffic demand passing selected node x also passe node i.

Step 2.5 If the number of the selected source nodes is less than N, go back to Step 2.3. Otherwise go to Step 2.6.

Step 2.6 End.

2.3.2 Accuracy of the estimation using the information from the selected source nodes

We evaluate the accuracy of the estimation using the traffic information from the source nodes selected by our method. In this evaluation, we use the same metrics, topology and traffic demands as in Subsection 2.2.3.

Figure 2.5 shows T_{RMSE} normalized by the average amount of traffic demand when we change the number of source nodes. The vertical axis is normalized T_{RMSE}, and the horizontal axis is the

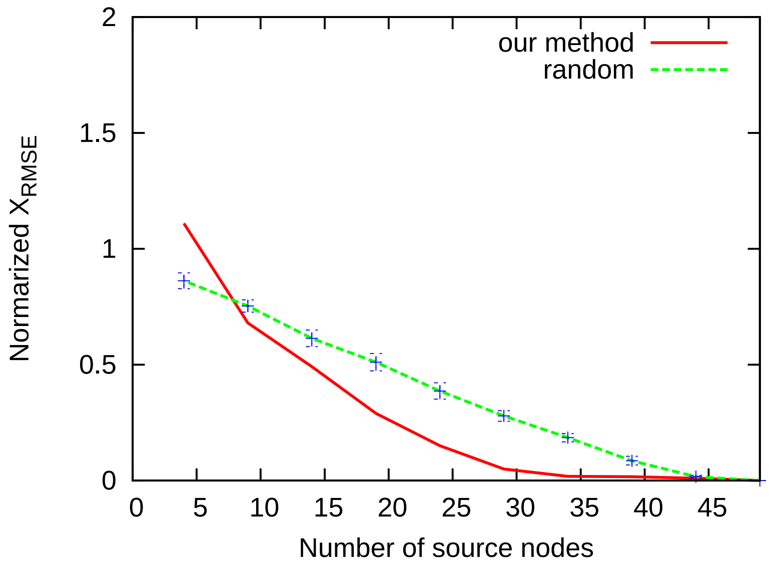


Figure 2.6: RMSE of the traffic volume on each link in the case of Japan topology (μ = 16.6, σ = 1.04)

number of source nodes. “our method” indicates the results for the case that we select the source nodes by our method. We also plot the results for the average of 20 cases of the randomly selected source nodes indicated as ”random” with the error bar representing the 95% confidence interval.

According to Figure 2.5, by selecting more than 24 source nodes, our method can estimate the traffic matrix with an estimation error of less than 0.6 times the average of the actual traffic matrix, while the method that randomly selects source nodes cannot. This is because our method selects source nodes to cover maximum possible traffic demands. Therefore, most traffic demands pass at least one source node selected in our method and can be estimated from the traffic volume information collected from the source nodes.

Even when we select the source nodes by our method, the estimation error shown in Figure 2.5 is larger than the results shown in References [57-62]; this is because since our method uses only the information of the selected source nodes. However we do not have to accurately estimate traffic matrices if we can accurately identify the congested links and perform the virtual network reconfiguration method to mitigate the congestion with the estimated traffic matrix. Figure 2.6 shows X_{RMSE} normalized by the average volume of traffic on each link as a function of the number of source nodes similar to Figure 2.5. According to Figure 2.6, our method accurately estimates the traffic volume on each link by selecting more than 24 source nodes, whereas the method that randomly selects source nodes cannot. That is, we can accurately identify the congested links from the traffic volume on each link estimated by our method.

2.4 Evaluation of the virtual network reconfiguration method using the estimated traffic demands

In this section, we evaluate the virtual network reconfiguration method using the traffic demands estimated by our method through simulations with multiple traffic patterns, topologies and the virtual network reconfiguration methods. Then, we clarify the environment in which our method estimates the traffic matrix and the traffic volume on each link accurately enough to perform the virtual network reconfiguration method.

2.4.1 Simulation settings

2.4.1.1 Topology

In our evaluation, we use Japan topology (49 nodes and 91 links) and a random graph (100 nodes and 200 links). Japan topology is a real ISP topology constructed by considering the geographical distance between each node. Thus, there are no links between the node pairs at a large distance. This causes the concentration of traffic demands on certain links. On the other hand, the random graph is constructed without the geographical distance. Therefore, there are no links where the traffic demands are concentrated.

We set the bandwidth for an optical path to 10 Gbps and the number of transmitters/receivers of node i to $D_i + 8$ where D_i is the degree of node i. In this simulation, we assume that the number of wavelengths on optical fibers will be sufficient. The initial virtual network is configured to suit the traffic matrix generated by Eq. (2.14) with $\mu = 16.6$ and $\sigma = 1.04$.

2.4.1.2 Traffic

The virtual network reconfiguration method reconfigures the virtual network when the traffic demand changes and the virtual network becomes unsuitable for the current traffic. In this simulation, we assume that the traffic changes significantly and that the actual traffic matrix after the traffic change is again generated randomly by Eq. (2.14). We generate two types of actual traffic matrices: *realistic traffic* and *skewed traffic*. Realistic traffic is generated using $\mu = 16.6$ and $\sigma = 1.04$ so that the generated traffic follows the traffic distribution monitored at the real ISP [64]. Skewed traffic is generated using $\mu = 16.6$ and $\sigma = 2.08$ so that the variance of the generated traffic demand is large. For skewed traffic, the traffic volume on each link may be different from the traffic volume modeled by Eq. (2.10) owing to the traffic demand whose volume is significantly larger than the other traffic. We set Θ to 1 in the cases of Japan topology and random graph so that the generated traffic can be accommodated in the topologies used in this simulation

2.4.1.3 Virtual network reconfiguration method

In this evaluation, we use two virtual network reconfiguration methods; One is the method proposed in Reference [6] that we call the Adaptive Reconfiguration based on Link Utilization (ARLU) in this chapter. The other is the Minimum Delay Logical Topology Design Algorithm (MLDA) proposed in Reference [65].

ARLU[6] ARLU adds a new optical path to mitigate congestion and, if possible, deletes currently underutilized optical paths for reclamation. Although the original version of this method adds or deletes only one optical path at a time, such addition or deletion cannot sufficiently mitigate the congestion in a large-scale network. Thus, we use the extended method to add or delete multiple optical paths.

This method uses two thresholds for the utilization of each optical path to define the congested and underutilized states T_H and T_L, respectively. In our evaluation, we set T_H to 0.3 and T_L to 0.2. The general sequence of the algorithm to calculate the virtual network is as follows:

Step 1 Calculate the utilization of all links of the virtual network from the estimated traffic matrix.

If at least one congested link (i.e., a link whose utilization exceeds threshold T_H) is found, go to the optical path addition phase (Step 2). If there is a link whose utilization is less than threshold T_L, go to Step 3.

Step 2 Execute the optical path addition phase described below and go to Step 4.

Step 3 Execute the optical path deletion phase described below and go to Step 4.

Step 4 Calculate the packet routes over the new virtual network and re-calculate the utilization of all links of the new virtual network from the estimated traffic matrix.

Step 5 If the optical path is not added/deleted, go to Step 6. Otherwise, return to Step 1.

Step 6 End.

In the above steps, the routes of the packets over the virtual network are calculated using the SPF algorithm. The following are the details of the optical path addition/deletion phases.

In the optical-path addition phase, if the link utilization of the current virtual network exceeds T_H, a new optical-path is set up to reroute traffic away from the congested link. First, we collect a set of traffic demands that passes the most congested link. Then, we select the busiest set of collected traffic demands. Finally, we add the direct optical path (i.e., a single directly connected link) from the ingress to the egress nodes of the selected traffic demands.

In the optical path deletion phase, if the utilization of an optical path is less than T_L and its deletion doesn't cause congestion, the path is torn down so that the IP router ports and wavelengths can be reclaimed for future use. The optical path is checked for the potential of its deletion to cause congestion by calculating the utilization of the optical paths after deletion using the traffic matrix. If there is more than one deletion candidate, each candidate path is tested in the ascending order of utilization.

MLDA[65] MLDA is a method that aims to minimize the maximum link utilization. It uses the following parameters; $T_{i,j}$ is the element of the traffic matrix corresponding to the traffic from node i to node j, d_i^{out} is the number of transmitters in node i, and d_i^{in} is the number of receivers in node i. MLDA reconfigures the virtual network by the following steps:

Step 1 Select the node pair i-j whose traffic volume is the largest. If the traffic volume $T_{i,j}$ corresponding to the selected node pair is 0, go to Step 6. Otherwise, go to Step 2.

Step 2 If $d_i^{out} > 0$ and $d_j^{in} > 0$, configure the optical path from node i to node j and go to Step 3. Otherwise, go to Step 4.

Step 3 Decrement d_i^{out} and d_j^{in}.

Step 4 Select the node pair k-l whose traffic volume is the second largest.

Step 5 Update $T_{i,j} \leftarrow T_{i,j} - T_{k,l}$, and go to Step 1.

Step 6 End.

2.4.1.4 Metrics

Accuracy of the estimation We evaluate the accuracy of our estimation method by using the same metric described in subsection 2.2.3.1.

Performance of the virtual network reconfiguration method using the estimated traffic matrix The goal of the virtual network reconfiguration method used in this simulation is to mitigate the congestion. Thus, we evaluate the performance of the virtual network reconfiguration method using the estimated traffic matrix by the maximum link utilization achieved by it.

2.4.2 Performance of ARLU in the case of Japan topology

In this subsection, we evaluate ARLU using the traffic demands estimated by our method using Japan topology (49 nodes and 91 links) as a physical topology. In the evaluation in this subsection, the initial virtual network is also constructed by ARLU so as to suit the initial traffic demands.

2.4.2.1 Realistic Traffic

In this subsection, we generate the realistic traffic for the traffic demands.

Before evaluating the performance of ARLU, we investigate the accuracy of our estimation method when the virtual network is constructed using ARLU. Figures 2.7 and 2.8 show T_{RMSE}

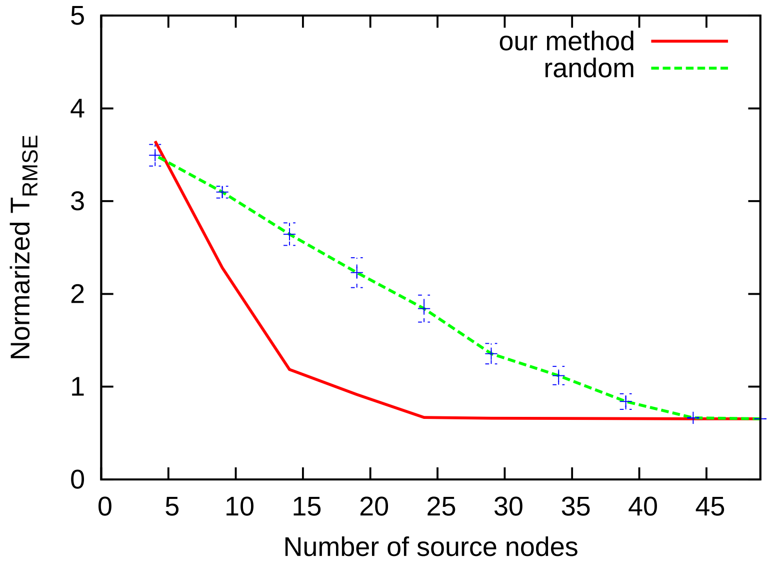


Figure 2.7: RMSE of the traffic matrix using ARLU in the case of Japan topology (μ = 16.6, σ = 1.04)

normalized by the average amount of traffic demand and X_{RMSE} normalized by the average volume of traffic on each link, respectively. In these figures, the vertical axes are the normalized T_{RMSE} and normalized X_{RMSE}, and the horizontal axis is the number of source nodes. "our method" indicates the results for the case that we select the source nodes by our method. We also plot the results for the average of 20 cases randomly selected from the source nodes indicated as "random" with the error bar representing the 95% confidence interval.

According to Figures 2.7 and 2.8, similar to the results described in Subsection 2.3.2, by selecting more than 29 source nodes, our method can estimate the traffic matrix with an estimation error of less than 0.7 times the average of the actual traffic matrix, whereas the method that randomly selects source nodes cannot. Our method can also estimate the traffic volume on each link accurately by selecting more than 29 source nodes. That is, we can accurately identify the congested links from the traffic volume on each link estimated by our method.

Figure 2.9 shows the maximum link utilization achieved by ARLU using the estimated traffic matrix as a function of the number of source nodes. The vertical axis is the maximum link utilization, and the horizontal axis is the number of source nodes. "Our method" indicates the results for

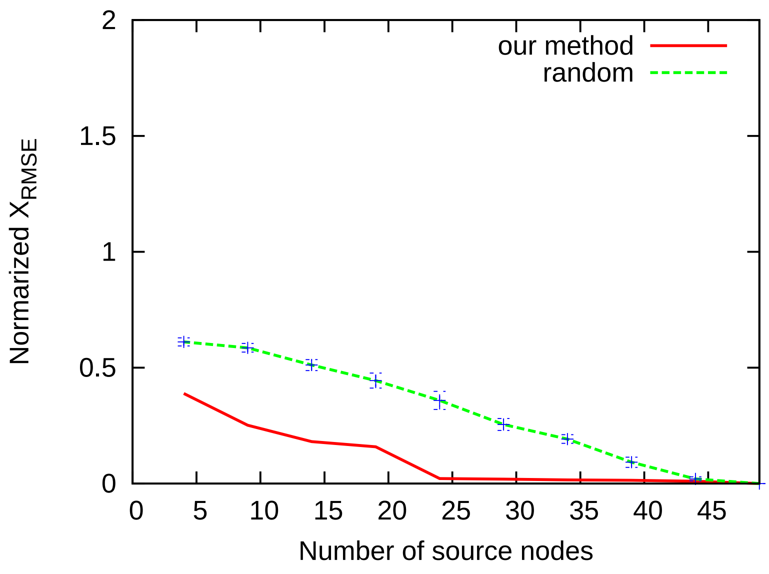


Figure 2.8: RMSE of the traffic volume on each link using ARLU in the case of Japan topology (μ = 16.6, σ = 1.04)

the case that we perform ARLU using the traffic matrix estimated by our method, "actual traffic" indicates the case that we perform ARLU using the actual traffic matrix, and "before reconfiguration" indicates the maximum link utilization before we reconfigure the virtual network. In addition, we plot the results for the average of 20 cases randomly selected from the source nodes indicated as "random" with the error bar representing the 95% confidence interval.

According to Figure 2.9, when we select more than 24 source nodes, ARLU using the traffic matrix estimated by our method achieves maximum link utilization similar to ARLU using the actual traffic matrix; on the other hand the maximum link utilization after performing ARLU using the traffic matrix estimated from the information of randomly selected nodes remains large. This is because we can accurately identify the congested links from the traffic matrix estimated by our method, as shown in Figure 2.8.

2.4.2.2 Skewed traffic

In this subsection, we evaluate our method when the variance of the traffic demand is large. In this case, the traffic volume on each link may be different from the traffic volume modeled by Eq.

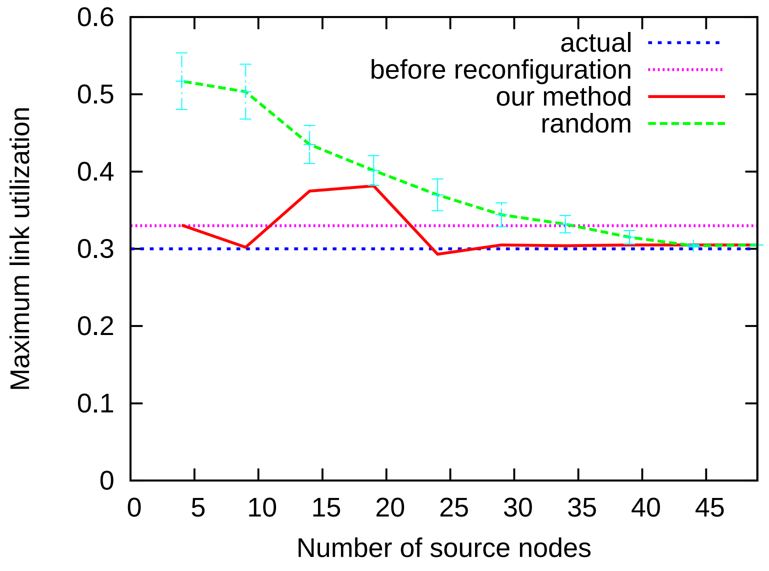


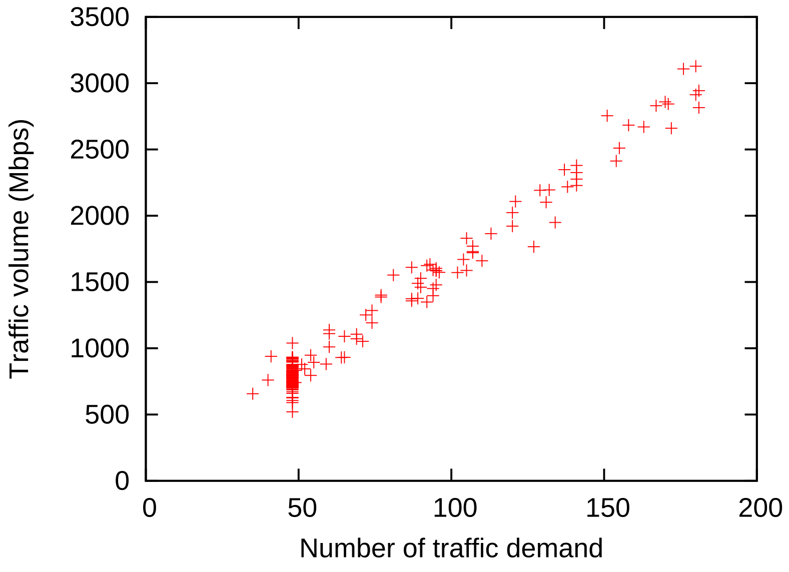
Figure 2.9: Maximum link utilization after the virtual network reconfiguration method using the estimated traffic matrix using ARLU in the case of Japan topology (μ = 16.6, σ = 1.04)

(2.10).

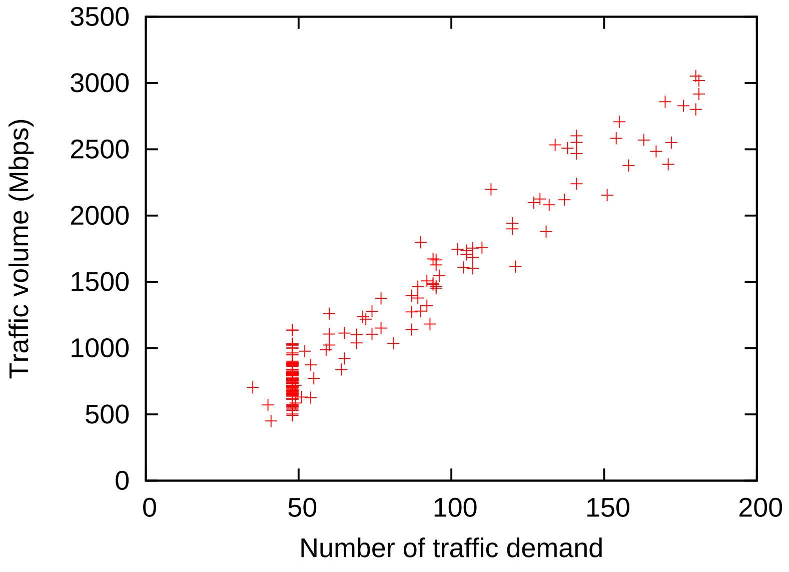
Figure 2.4.2.2 shows the relationship between the number of traffic demands passing a link and the traffic volume on each link in the initial virtual network constructed by ARLU in the case of the skewed traffic. Compared with Figure 2.4.2.2, the difference between the actual traffic volume on each link and the traffic volume modeled by Eq. (2.13) is large because of the large variance of traffic demands.

Figure 2.11 shows X_{RMSE} normalized by the average volume of traffic on each link. According to Figure 2.11, even when the variance of traffic demand is large, our method can accurately estimate the traffic volume on each link by selecting more than 24 source nodes. This is because our method selects source nodes to cover maximum possible traffic demands similar to the case in Figure 2.8. Most traffic demands pass at least one of the source nodes selected in our method and can be estimated from the traffic volume information collected from the source nodes. Therefore, even if the traffic volume estimated by Eq. (2.13) includes a large estimation error, we can accurately identify the congested links.

Figure 2.12 shows the maximum link utilization achieved by ARLU using the estimated traffic



(a) Case of realistic traffic (μ = 16.6, σ = 1.04)



(b) Case of skewed traffic (μ = 16.6, σ = 2.08)

Figure 2.10: Relationship between the number of traffic demands and the traffic volume on each link in the case of Japan topology

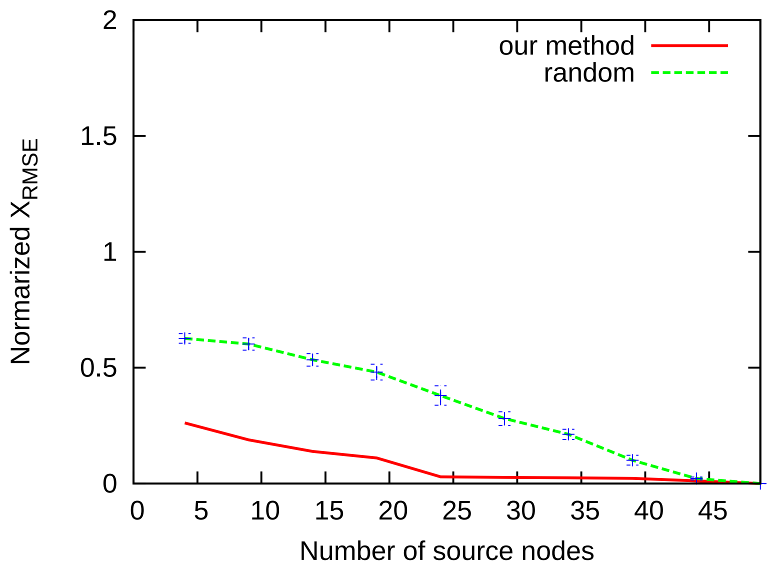


Figure 2.11: RMSE of the traffic volume on each link using ARLU in the case of Japan topology (μ = 16.6, σ = 2.08)

matrix as a function of the number of source nodes. According to Figure 2.12, when the number of source nodes is more than 19, ARLU using the estimated traffic matrix can achieve link utilization similar to ARLU using the actual traffic matrix. This is because our method can accurately estimate the traffic volume on each link by selecting more than 19 source nodes.

2.4.3 Performance of ARLU in the case of the random graph

In this subsection, we evaluate our method using the random graph (100 nodes and 200 links) as a physical topology. The initial virtual network is configured by ARLU. Now, we generate the realistic traffic.

Figures 2.13 and 2.14 show normalized X_{RMSE} and the maximum link utilization achieved by the ARUL using the estimated traffic matrix, respectively. According to Figure 2.13, our method cannot accurately estimate the traffic volume on each link. The estimation errors of our method are larger than the method that randomly selects source nodes. As a result, ARLU using the estimated traffic matrix cannot reduce the maximum link utilization as much as ARLU using the actual traffic matrix unlike the case of Japan topology.

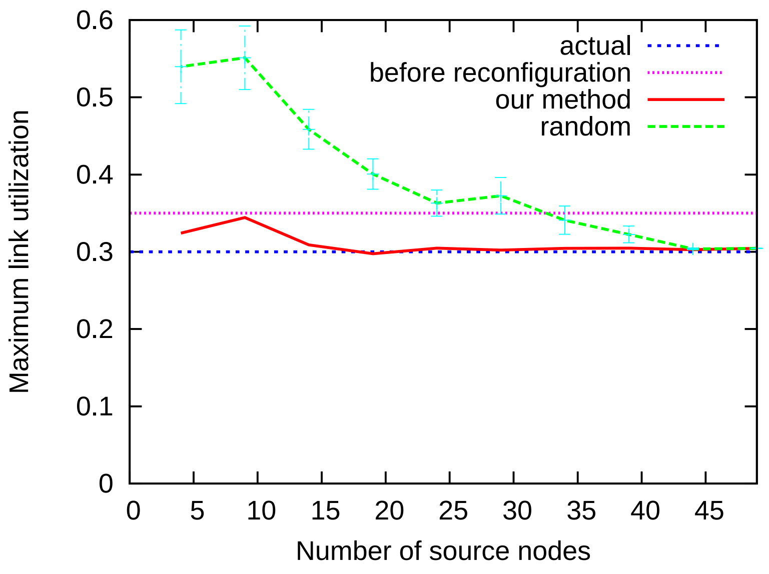


Figure 2.12: Maximum link utilization after the virtual network reconfiguration method using the estimated traffic matrix using ARLU in the case of Japan topology (μ = 16.6, σ = 2.08)

This large estimation error is caused by the small number of traffic demands passing each link. Because the number of traffic demands passing each link is small in the virtual network constructed over the random graph, a large number of traffic demands do not pass any of the selected source nodes and are estimated from the traffic volume estimated by Eq. (2.13).

In addition, the small number of traffic demands passing each link causes the large estimation errors in the traffic volume estimated by Eq. (2.13). As shown in Figure 2.15, a large number of links are passed by only a smaller number of traffic demands than the number of traffic demands passing the ingress/egress link. However, our method does not obtain the traffic information on such links because it selects the nodes passed by a large number of traffic demands. This causes large estimation errors on the parameters, α and β in Eq. (2.13). Because the traffic volume estimated by Eq. (2.13) includes large estimation errors, we cannot accurately estimate the traffic volume.

On the other hand, the number of traffic demands passing each link in the real ISP topology is much larger than the random graph because the real ISP topology is constructed by considering the geographical distance. According to Figure 2.4.2.2, many links are passed by much larger number of traffic demands than the ingress/egress links in the Japan topology. In this case, the

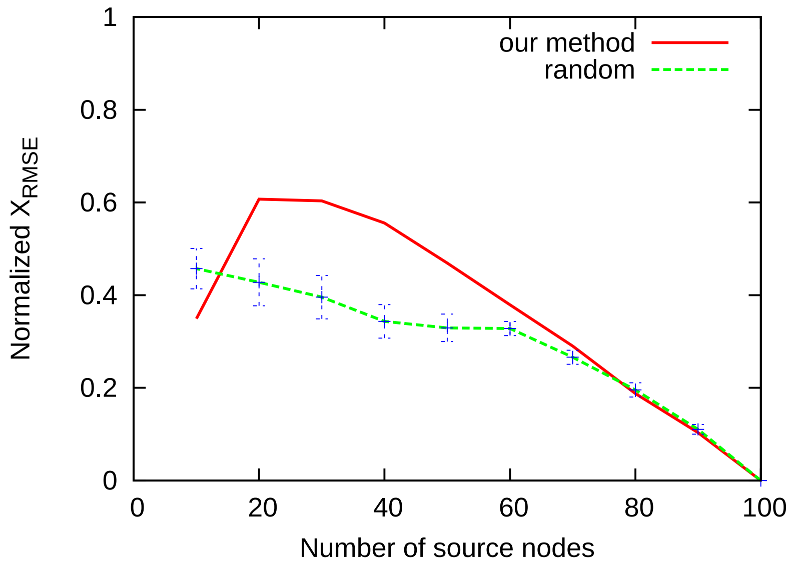


Figure 2.13: RMSE of the traffic volume on each link in the case of the random graph (μ = 16.6, σ = 1.04)

traffic information on the links passed by a small number of traffic demands is also obtained from the selected source nodes; this is because the number of traffic demands passing the ingress/egress link connected to the selected source node is smaller than that passing the other links. Thus, we can accurately estimate parameters, α and β in Eq. (2.13). Therefore, the large estimation errors caused by the small the number of traffic demands passing each link do not occur in the real ISP topology.

2.4.4 Performance of MLDA

We evaluate our method for the case when MLDA is used for reconfiguration. In this evaluation, we configure the initial virtual network by MLDA using the initial traffic matrix. In this subsection, we generate the realistic traffic.

2.4.4.1 In the case that the number of transmitters/receivers are large

In this section, we evaluate MLDA when each node has $D_i + 8$ transmitters/receivers where D_i is the node degree of node i.

Before evaluating the performance of MLDA, we evaluate the accuracy of the estimation when

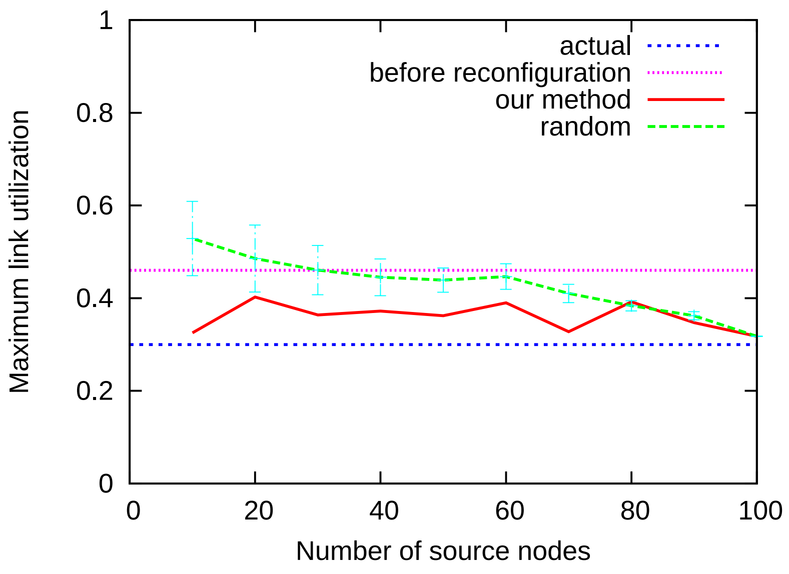


Figure 2.14: Maximum link utilization after the virtual network reconfiguration method using the estimated traffic matrix in the case of random graph (μ = 16.6, σ = 1.04)

the initial virtual network is configured by MLDA. Figures 2.16 and 2.17 show T_{RMSE} normalized by the average amount of traffic demand and X_{RMSE} normalized by the average volume of traffic on each link, respectively. According to Figures 2.16 and 2.17, the estimation errors of our method are significantly larger than the results of the case of ARLU and similar to the estimation errors of the case when the source nodes are randomly selected. This is because of the difference between the initial virtual network constructed by ARLU and that constructed by MLDA. Thus, we compare the virtual network constructed by ARLU and MLDA.

Figure 2.18 shows the relationship between the number of traffic demands passing a link and the traffic volume on the link obtained by our simulation when the initial virtual network configured by MLDA is used. Comparing Figures 2.18 and 2.4.2.2, the number of traffic demands passing each link is much smaller in the virtual network configured by MLDA than that configured by ARLU. This is because MLDA constructs maximum possible optical paths, whereas ARLU constructs only the optical-paths required to mitigate the congestion. This small number of traffic demands passing each link causes the large estimation errors as discussed in subsection 2.4.3,

Figure 2.19 shows the maximum link utilization achieved by MLDA using the estimated traffic

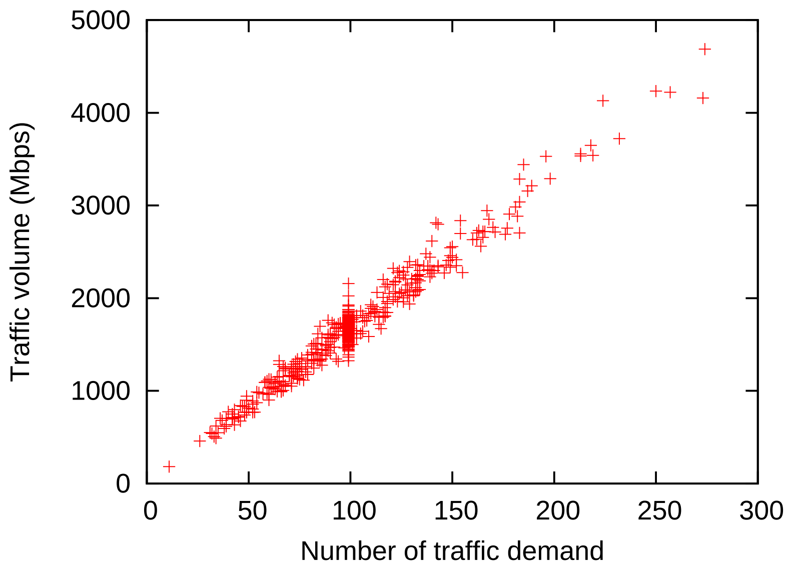


Figure 2.15: Relationship between the number of traffic demands and the traffic volume on each link in the case of the random graph ($\mu = 16.6$, $\sigma = 1.04$)

matrix as a function of the number of source nodes. According to Figure 2.19, the maximum link utilization achieved by MLDA using the estimated traffic matrix is greater than that achieved by MLDA using the actual traffic demand. This is because of the large estimation errors shown in Figures 2.16 and 2.17. However, we can reduce the maximum link utilization compared with that before the virtual network reconfiguration method. This is because we can add many optical paths in the environment used in this evaluation. We add optical paths to the node pairs whose traffic amounts are large despite the large estimation errors.

2.4.4.2 In the case that the numbers of the transmitters/receivers are small

Now, we evaluate our method when the numbers of the transmitters/receivers is small. Each node has $D_i + 2$ transmitters/receivers where D_i is the node degree of the node i. As mentioned earlier, we configure the initial virtual network by MLDA.

Figures 2.20 and 2.21 show normalized T_{RMSE} and X_{RMSE}, respectively. According to the figures, our method can estimate the traffic matrix and the traffic volume on each link more accurately than the case of $D_i + 8$ transmitters/receivers. This is because the number of constructed optical

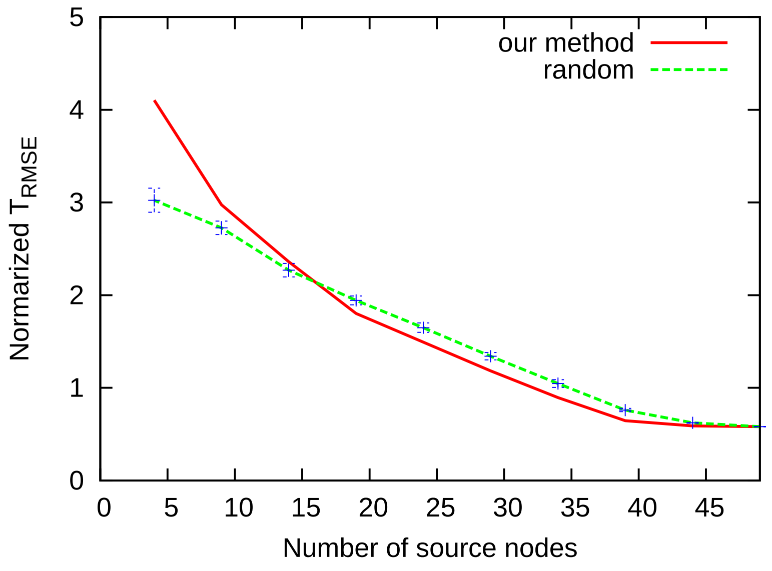


Figure 2.16: RMSE of the traffic matrix using MLDA in the case of Japan topology (μ = 16.6, σ = 1.04)

paths in the case of $D_i + 2$ transmitters/receivers is smaller than that in the case of $D_i + 8$ transmitters/receivers. Because the number of constructed optical paths is small, the number of traffic demands passing a node is larger than for the case of D_i+8 transmitters/receivers. Thus, the number of traffic demands that do not pass any selected source nodes in the case of $D_i + 2$ is smaller than that for the case of $D_i + 8$.

Figure 2.22 shows the maximum link utilization achieved by MLDA using the estimated traffic matrix as a function of the number of source nodes. MLDA using the traffic matrix estimated by our method cannot reduce the maximum link utilization even when we select 39 source nodes, although the estimation error of the traffic matrix (Figure 2.22) is as small as that for the case of ARLU shown in subsection 2.4.2.1. That is, when the numbers of transmitters/receivers is small, MLDA is less robust to the estimation errors of the traffic matrix than ARLU. This is because MLDA uses only the traffic matrix to decide where to add the optical paths, while ARLU uses both the traffic matrix and the traffic volume on each link. As shown in Figures 2.7 and 2.8, the traffic volume on each link can be estimated more accurately than the traffic matrix. Thus, ARLU can configure the adequate virtual network even when using the estimated traffic matrix.

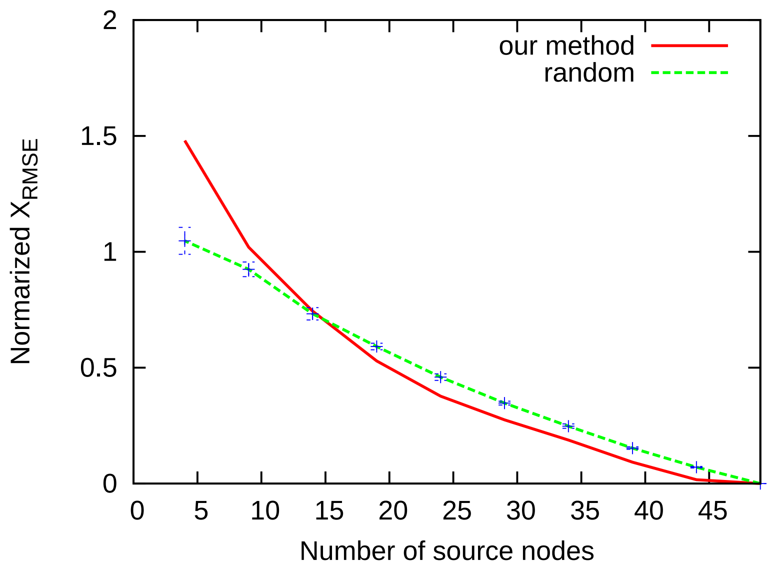


Figure 2.17: RMSE of the traffic volume on each link using MLDA in the case of Japan topology (μ = 16.6, σ = 1.04)

2.4.5 Summary of the results

According to the results in this section, our method can estimate the traffic matrix accurately enough to perform ARLU. Although the estimation errors become large when the number of traffic demands passing each link is small, such large estimation errors do not occur in the real ISP topology. This is because the number of traffic demands passing some links in the real ISP is large since the real ISP topology is constructed by considering the geographical distance between each node.

The evaluation also shows that MLDA is less robust to estimation errors than ARLU. This is because MLDA uses only the traffic matrix to decide where to add the optical paths, while ARLU uses both the traffic matrix and the traffic volume on each link. That is, the virtual network reconfiguration method using both the traffic matrix and the traffic volume on each link is suitable for the case when the estimated traffic information is used.

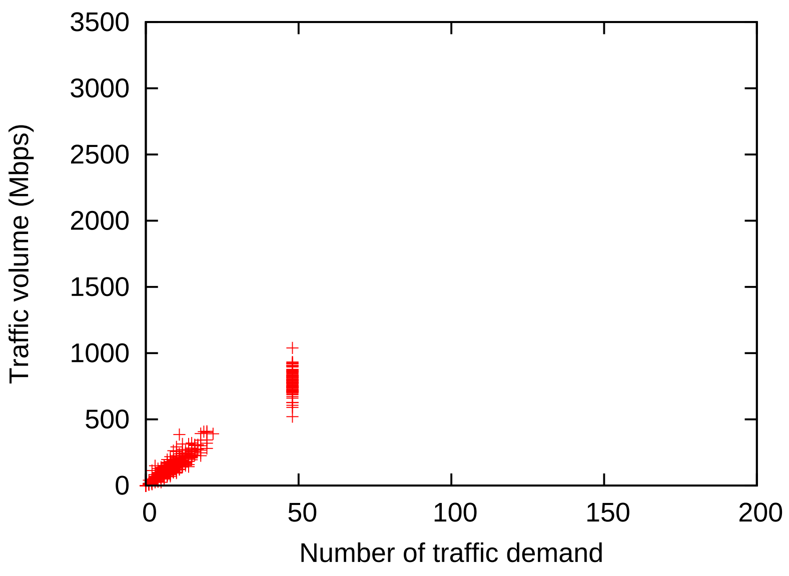


Figure 2.18: Relationship between the number of traffic demands and the traffic volume on each link in the case of Japan topology using MLDA ($\mu = 16.6$, $\sigma = 1.04$)

2.5 Evaluation in a large scale network

Here, we evaluate our method using AT&T topology (523 nodes and 1304 links) measured in Reference [63] as a physical topology. The initial virtual network is configured by ARLU. In addition, we show the results for the case of the skewed traffic that is generated by Eq. (2.14) using $\mu = 16.6$, $\sigma = 2.08$ and $Theta = 0.1$. We omit the results for the case of the realistic traffic because the realistic traffic is easier to estimate than the skewed traffic.

2.5.1 Accuracy of estimated traffic volume

Figure 2.23 shows the normalized X_{RMSE} when we change the number of source nodes. According to Figure 2.23, our method accurately estimates the traffic volume on each link by selecting more than 123 source nodes, while the method that randomly selects source nodes cannot. That is, our method can identify the congested links more accurately than the case of the estimation from the randomly selected nodes in the large scale ISP topology.

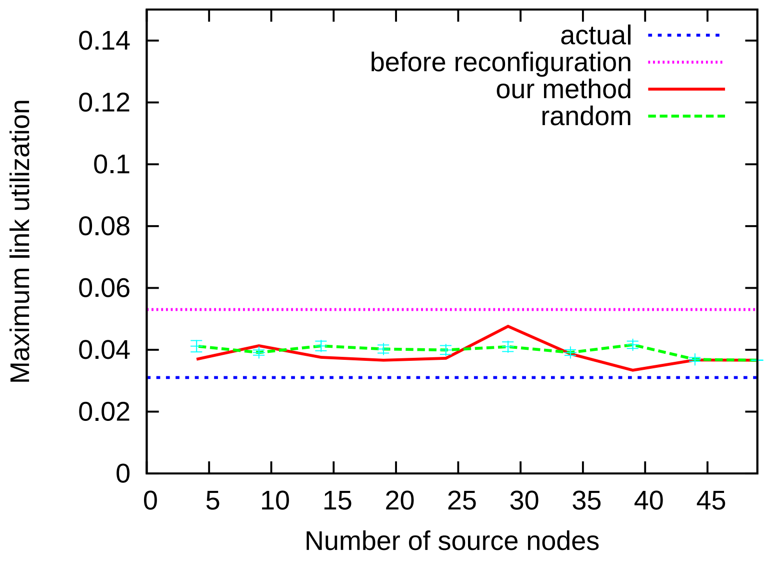


Figure 2.19: Maximum link utilization after the virtual network reconfiguration method using the estimated traffic matrix using MLDA in the case of Japan topology (μ = 16.6, σ = 1.04)

2.5.2 Performance of ARLU

Figure 2.24 shows the maximum link utilization achieved by ARLU using the estimated traffic matrix. According to Figure 2.24, ARLU using the traffic matrix estimated by our method can significantly reduce the maximum link utilization. In addition, it can achieve 1.3 times the maximum link utilization achieved by ARLU using the actual traffic matrix by selecting 123 source nodes in the case of skewed traffic. That is, our method can estimate the traffic volume on each link accurately enough to be used by ARLU by collecting traffic information from 30% of all nodes in AT&T topology.

2.5.3 Calculation time

The calculation time to estimate the traffic volume is important for handling traffic changes that occur in a short period of time. The complexity of our method to select source nodes is O(N^3) and the complexity of our method to estimate the traffic matrix is O(N^2L) where N is the number of nodes and L is the number of links in the topology.

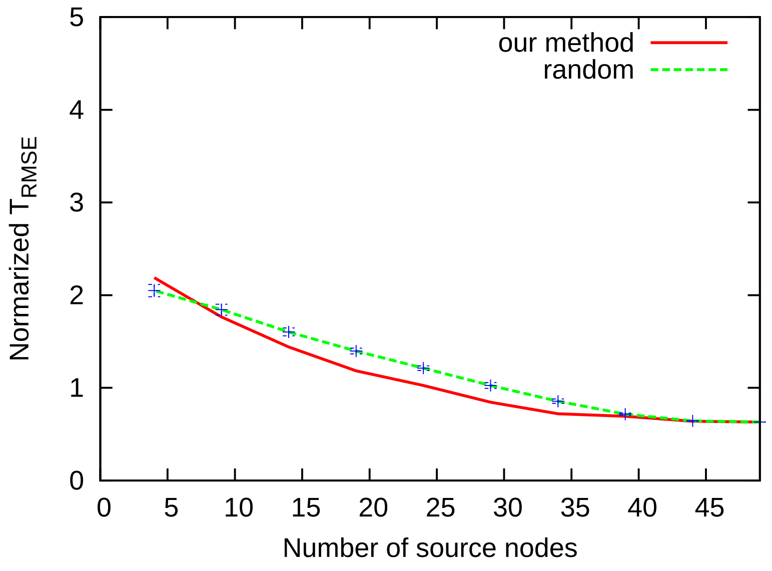


Figure 2.20: RMSE of the traffic matrix using MLDA when the number of the transmitters/receivers is small in the case of Japan topology ($\mu = 16.6$, $\sigma = 1.04$)

We measure the time to estimate the traffic volume by our method. In this simulation, we use a server with a 3.16GHz Intel Xeon 5460 Processor to measure the time to estimate the traffic volume. We implement the method to select source nodes in C++ and the method to estimate the traffic matrix in MATLAB.

In our method, selecting source nodes requires 40 seconds and estimating traffic volume requires only 1.5 seconds for the virtual network constructed over the AT&T topology. We do not have to select the source nodes within a small interval because our method continues to collect the traffic information from the same source nodes unless the virtual network is changed. Therefore, even in a large scale network such as AT&T topology, our method can estimate the traffic volume in a sufficiently short time to handle the traffic changes that occur in a short period of time.

2.6 Conclusion

In this chapter, we developed a method to select the source nodes and estimate the traffic matrix on the basis of the traffic information collected from the selected source nodes. We evaluated

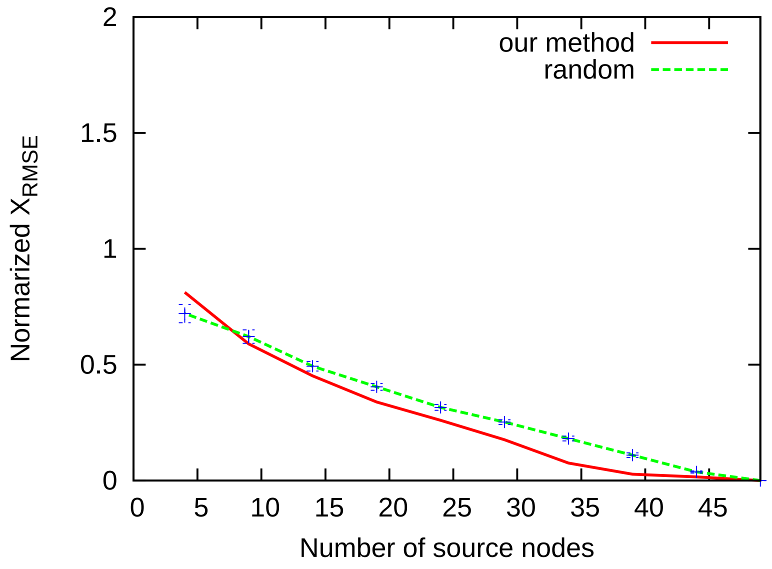


Figure 2.21: RMSE of the traffic volume on each link using MLDA when the number of the transmitters/receivers is small in the case of Japan topology (μ = 16.6, σ = 1.04)

our method through simulations. According to the simulation results, our method estimates the traffic matrix and the traffic volume on each link accurately enough to perform the virtual network reconfiguration method in real ISP topologies; however our method cannot accurately estimate the traffic matrix when the number of traffic demands passing each link is small. The simulation results show that we can mitigate the congestion by using the traffic matrix estimated from 50% of all nodes in the case of Japan topology and 30 % of all nodes in the case of AT&T topology.

Future researches include improving the accuracy of the estimation by developing more sophisticated method to estimate the uncollected traffic volume on each link and the traffic matrix by using the information from the source nodes.

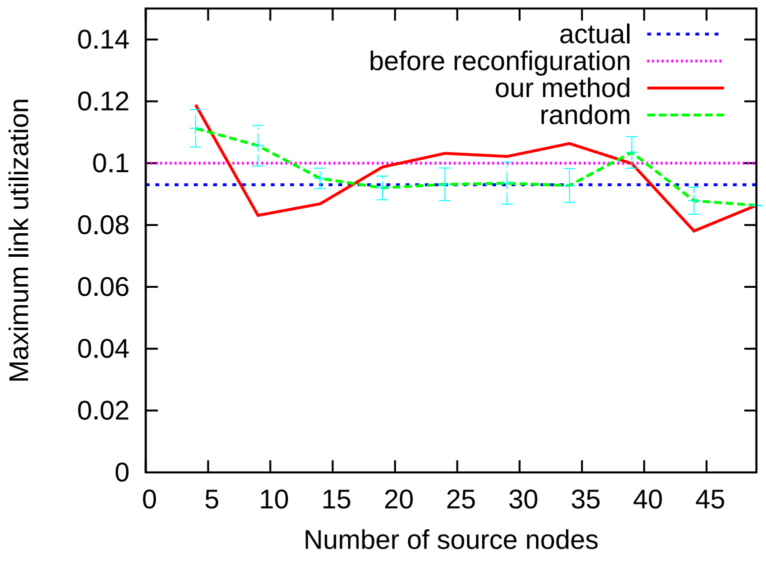


Figure 2.22: Maximum link utilization after the virtual network reconfiguration method using the estimated traffic matrix using MLDA when the resource is reduced in the case of Japan topology (μ = 16.6, σ = 1.04)

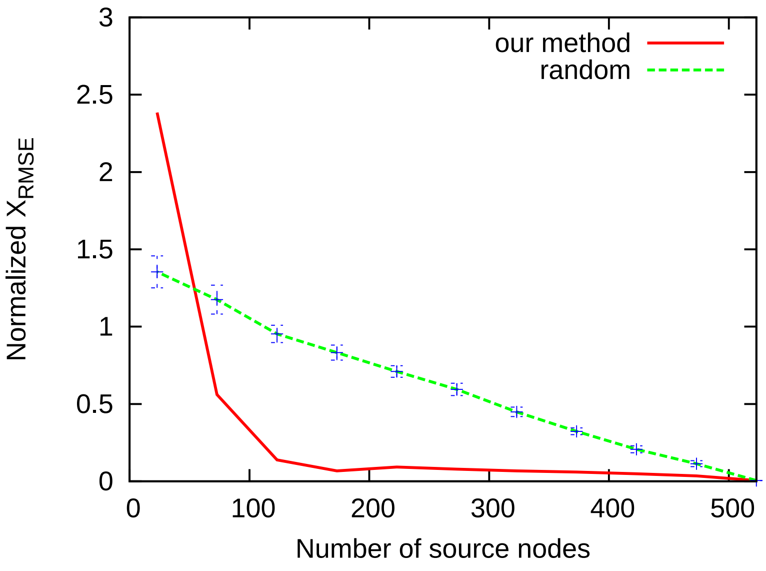


Figure 2.23: RMSE of traffic volume on each link in the case of AT&T topology (μ = 16.6, σ = 2.08)

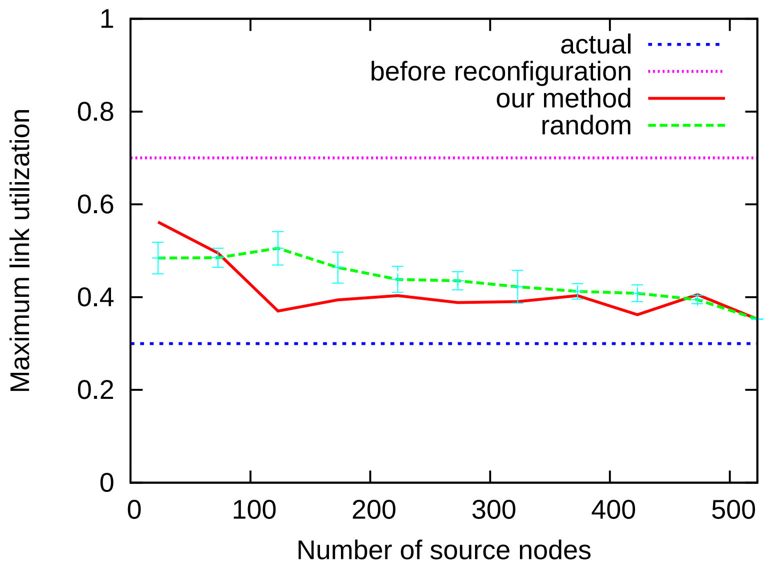


Figure 2.24: Maximum link utilization after the virtual network reconfiguration method using the estimated traffic matrix in the case of AT&T topology (μ = 16.6, σ = 2.08)

Chapter 3

Virtual Network Reconfiguration for Reducing Energy Consumption in Optical Data Centers

3.1 Virtual Networks in Optical Data Center Networks

An optical network architecture that allows for reconfiguration of the virtual network in a data center has been proposed in [10]. Following that proposition, in this section, we briefly introduce a data center network architecture that uses OCSs and electronic switches and describe how the energy consumption in such a datacenter network can be reduced.

3.1.1 Optical Data Center Network Architecture

Figure 3.1 shows a data center network architecture where the core of the data center network is constructed of OCSs. In optical data center network architecture, the core of the data center network is constructed of OCSs. Each ToR switch is connected to all servers within the same server rack, as well as to one of the ports of OCSs in the core network through its ports. A *virtual link*, which is considered a directly connected high-capacity link for ToR switches, is established by configuring the OCSs. One approach using the virtual links is to establish the virtual links

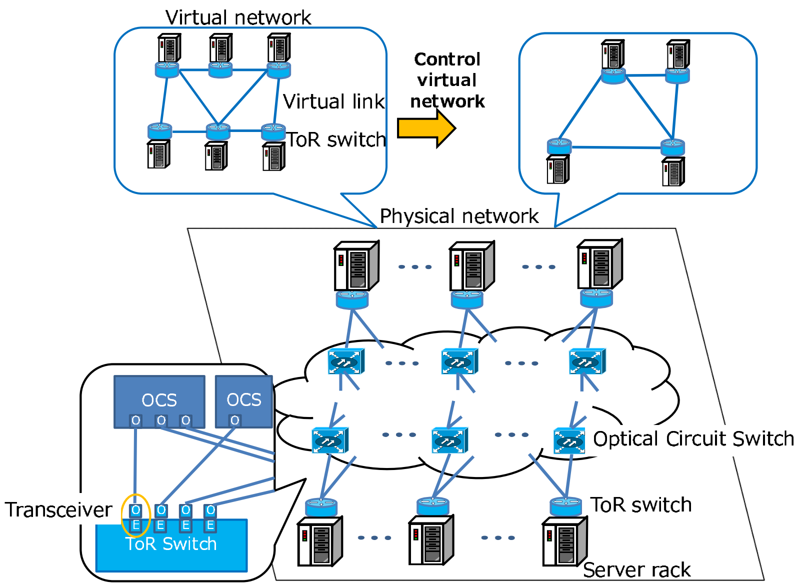


Figure 3.1: Data Center Network Using OCSs and ToR switches

between all communicating ToR switches. However, this approach has difficulty in handling the traffic changes with a short period, because it requires to reconfigure the OCSs every time the communicating ToR switch pair changes.

Instead, we construct the virtual network, which is formed by the set of the ToR switches. The traffic between server racks is relayed over the virtual network; ToR switches may relay packets from one virtual link to another virtual link according to the final destinations of the packets. In this approach, the virtual network reconfiguration is not necessary even if the communicating ToR switch pairs change unless the amount of the traffic changes significantly. When the amount of the traffic changes and the current virtual network becomes no longer suitable, the virtual network is reconfigured by adding or deleting virtual links.

Singla et al. [10] also proposed a method for configuring virtual networks to achieve high throughput. In that method, virtual links are added between connected server racks that handle large amounts of traffic to maximize the throughput. Such networks are also useful for reducing energy consumption; a virtual network that consumes only a small amount of energy can be dynamically reconfigured to accommodate the instantaneous traffic demand. However, existing research has not considered virtual network reconfiguration aimed at minimizing energy consumption for data center networks. Then, we discuss such a virtual network configuration method in Section 3.3.

3.1.2 Virtual Network Reconfiguration Method for Reducing Energy Consumption

The energy consumption of OCSs is much lower than that of ToR switches. For example, the 192-port Glimmerglass OCS [66] consumes less than 85 W of power, while the 48-port Arista 7148SX switch [67] consumes 600 W. Thus, to reduce the energy consumption of the virtual network, the energy consumption of ToR switches should be considered. In this chapter, we assume that the ports of ToR switches can be powered down to save energy. Thus, the energy consumption of the network can be reduced by minimizing the number of open ToR switch ports used in the virtual network, and powering down any unused ports if sufficiently large bandwidth and low delay can be ensured. The virtual network with the minimum required number of open ports can be obtained as a solution to an optimization problem. However, optimization problems require considerable time to be solved for large networks.

Therefore, in this paper, we propose a new method called *VNR-DCN*, which can be used to configure a suitable virtual network that achieves a sufficiently large bandwidth and low latency with a small number of open ports. In VNR-DCN, we configure a suitable virtual network by setting the parameters of the base topology called GFB instead of solving an optimization problem. The complexity of VNR-DCN in setting up the GFB parameters is $O(N|K_{\mathrm{can}}|)$, where N is the number of ToR switches and K_{can} is the set of candidates for the number of layers in GFB (typically two or three, as shown in Subsection 3.3.2.3). In the next section, we discuss the base topology used in our virtual network configuration, before proceeding to explain the method for setting the parameters of the base topology in Section 3.3.

3.2 Virtual Network Topologies Suitable for Optical Data Center Networks

In this section, we discuss the requirements on the base topology used in our virtual network reconfiguration method and investigate the properties of existing network topologies for data center networks. Then, we present GFB, whose parameters can be adjusted to construct various data center network topologies. GFB is an extension of FB [28]. Although FB was originally proposed as a topology for interconnection networks of multiprocessors, Abts et al. [23] found that data center networks constructed using FB provides sufficient bandwidth with lower energy consumption than ones constructed using the FatTree topology because the number of required switches is smaller than that in FatTree. However, FB requires a large number of links, while a topology with a small number of links is suitable for low traffic when considering energy consumption. In this chapter, first we extend FB and introduce the layers and parameters indicating the number of links in each layer. By setting an appropriate number of links for each layer, we construct an energy-efficient network topology that uses only the number of links required to accommodate the instantaneous traffic demand.

3.2.1 Properties of an Optimal Virtual Network for a Data Center

An optimal virtual network for a data center has the following properties.

Low Energy Consumption The energy consumption of the network is responsible for a non-negligible fraction of the total energy consumption in the data center, as mentioned above. In a data center network consisting of optical switches, most of the energy is consumed by ToR switches. Therefore, the number of active ToR switches should be minimized. That is, in an optimal network topology, the only active ToR switches are those connected to servers communicating with other servers. Moreover, the energy consumption of ToR switches can be reduced by powering down any unused ports. Thus, the energy consumption of a data center network can be minimized by constructing a virtual network using the smallest possible number of active ToR switch ports.

Large Bandwidth between Servers In some applications, such as distributed file systems, large amounts of data are exchanged between servers. The bandwidth between servers is thus important for such applications. Therefore, the virtual network should provide sufficient bandwidth for communication between servers. To ensure sufficient bandwidth, virtual links should be added to accommodate the instantaneous traffic demand without congestion.

Short Delay between Servers Data centers handle large amounts of data by using distributed computing frameworks [55, 68], where a large number of servers communicate with each other. If inter-server communication experiences long delays, it takes time to obtain the required data from other servers, which degrades the performance of the entire data center. Thus, the delay should be kept sufficiently low for the purposes of the particular data center.

However, delays in communication between servers are difficult to predict when constructing a virtual network because delays are affected by traffic load. Therefore, in this study we minimize the delay in communication between servers by constructing a virtual network with a small number of hops between servers.

3.2.2 Existing Data Center Network Topologies

Several data center network topologies have already been proposed. Although they apply to physical networks constructed of electronic switches and servers, they may be used as base topologies for virtual networks because the objective here is to change the topology by setting its parameters. In this subsection, we discuss how such existing data center network topologies can be used as a base for our virtual network reconfiguration method.

Al-Fares et al. proposed a topology construction method called *FatTree* by using switches with a small number of ports [27]. FatTree is a tree topology constructed of multiple roots and pods containing aggregation switches. Each pod is regarded as a switch having a large number of ports consisting of multiple switches having a small number of ports. Pods are constructed using the butterfly topology, where each switch uses half of its ports to connect it to switches close to root switches, and the other half to connect it to switches close to leaf switches. Leaf switches are connected to servers. The number of switches in FatTree depends on the depth of the tree and the number of ports on each switch. FatTree with a depth of k constructed of switches with n ports each includes $(2k-1)\frac{n}{2}^{k-1}$ switches. In FatTree, the number of links from a switch close to a leaf switch equals the number of links to a switch close to a root switch; this is true for each switch. That is, the total bandwidth from a switch to switches close to a root switch equals that from switches close to a leaf switch to that switch. Therefore, none of the switches become bottlenecks, and a sufficiently large bandwidth is provided between all servers. However, FatTree is not suitable for virtual networks configuration because switches, except for leaf switches, are not connected to servers. In other words, ToR switches that are not connected to servers must be powered on, which results in high energy consumption.

Kim et al. [28] proposed the FB data center network topology. FB is constructed by *flattening* the butterfly topology, where switches in each row of the butterfly topology are combined into a single switch. FB provides a sufficiently large bandwidth between all servers while reducing the energy consumption compared to FatTree [27]. In addition, all switches in FB are connected to servers. Thus, unlike FatTree, all ToR switches that are not connected to any working servers can be powered down if FB is constructed as a virtual network. However, FB requires switches with a large number of ports to construct a large data center network even if traffic demand is small. Thus, FB is not preferred when there is low traffic demand.

Guo et al. proposed a data center network topology called *DCell*, which is constructed from a small number of switches and servers with multiple ports [29]. DCell uses a recursively defined structure; the level-0 DCell is constructed by connecting one switch with n ports to n servers, and the level-k DCell is constructed by connecting servers belonging to different level-(k-1) DCells. By directly connecting server ports, the DCell topology reduces the number of switches required to construct a large data center network. However, in the optical data centers introduced in Section 3.1, only ToR switches are connected to the OCSs, and virtual links are added between the ToR switches, and virtual links that connect servers directly cannot be added. Therefore, we can extend DCell by replacing a level-0 DCell with a switch. We call this topology *switch-based DCell*. Similar to DCell, switch-based DCell can be used to construct a large data center network by using switches with a small number of ports. That is, switch-based DCell achieves low energy consumption. However, switch-based DCell cannot provide a large bandwidth between all servers, because it has only one link between lower-level DCells.

Table 3.1: Comparison of existing topologies

Topology	Energy consumption	Bandwidth
FatTree [27]	Very high	Sufficiently large
FB [28]	High	Sufficiently large
Switch-based DCell	Low	Small

Table 3.1 summarizes the properties of the data center network topologies outlined above. Switch-based DCell consumes only a small amount of energy and is suitable for situations where the traffic volume between servers is small. However, it may not provide sufficient bandwidth for applications where servers generate a large amount of traffic. Furthermore, FB can provide large bandwidth between servers but consumes a lot of energy. That is, the most suitable network topology depends on the traffic demand. Accordingly, we propose GFB, in which we can reproduce various topologies, including FB and switch-based DCell, by adjusting the GFB parameters. By using GFB, we can set the parameters so as to provide sufficient bandwidth (similarly to FB) when the traffic amount is large and to reduce the number of links to the appropriate minimum when the traffic amount is small.

3.2.3 GFB Topology

In this subsection, we explain GFB in detail. GFB is constructed hierarchically as shown Figure 3.2, where the upper-layer GFB is constructed by connecting multiple lower-layer GFBs. GFB has the following parameters.

- Number of layers: $K_{\max}$
- Number of links per switch used to construct layer-k GFB: L_k
- Number of layer-(k-1) GFBs used to construct layer-k GFB: N_k

We can construct various topologies, including FB and switch-based DCell, by adjusting these parameters. We can construct GFB as a physical network or a virtual network. In this chapter, we

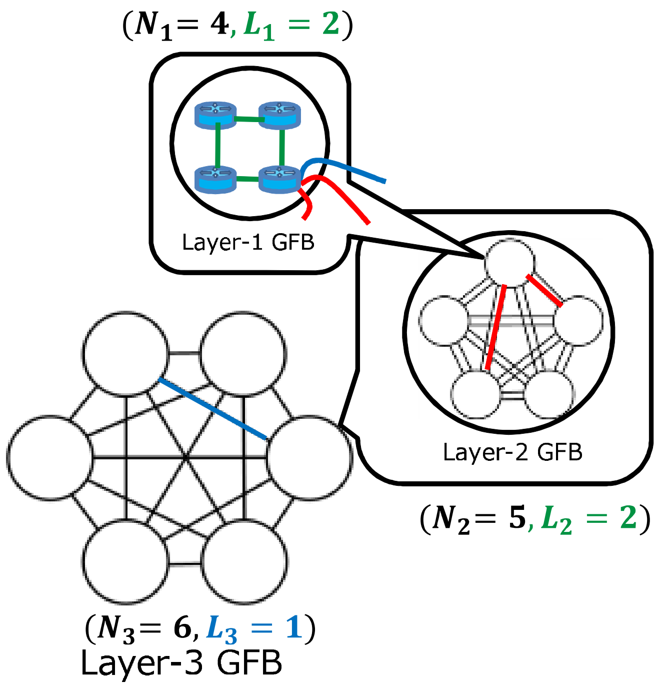


Figure 3.2: GFB topology

propose GFB as the base topology used for our virtual network configuration method. Thus, in this subsection, we describe the generic structure of GFB and use GFB to construct a virtual network.

These parameters are changed periodically so as to provide sufficient bandwidth for the performance of applications and to satisfy the requirements on the network set by the data center administrator. The interval for updating the parameters is not necessarily short, even though the traffic pattern can change rather frequently, e.g., every few seconds. This is because the total amount of traffic changes gradually, and frequent changes in traffic are absorbed by a load balancing technique in the routing method instead of by reconfiguring the virtual network. For example, in the evaluation discussed in Section 3.4, we use a 10-min interval between reconfigurations, following Heller et al. [11].

In the rest of this subsection, we describe the steps for constructing a virtual network using GFB in Paragraph 3.2.3.1. Then, we explain the routing method for GFB, which uses the GFB parameters, in Paragraph 3.2.3.2. Finally, in Paragraph 3.2.3.3, we discuss the properties of GFB based on its parameters.

3.2.3.1 Steps to Construct GFB

GFB is constructed hierarchically by constructing each GFB layer in order from layer-1 GFB to layer-$K_{\max}$ GFB. Layer-k GFB is constructed by the following two steps.

Step I Construct connections between all layer-(k-1) GFBs.

Step II Select switches connected to the links between each two layer-(k-1) GFBs.

In these steps, we use the IDs assigned to GFBs in each layer. A switch can be identified by the ID of the GFB it belongs to. We denote the ID of layer-k GFB to which switch s belongs to as $D_k^{\mathrm{GFB}}(s)$. We also define $D^{\mathrm{sw}}(s)$ for the ID of switch s in layer-k GFB as

$$D^{\mathrm{sw}}(s) = \sum_{1\le i\le K_{\max}} \left(D_i^{\mathrm{GFB}}(s) \prod_{j=1}^{i-1} N_j \right).$$

We provide details about Steps I and II in Paragraphs 3.2.3.1 and 3.2.3.1, respectively.

Connections between layer-(k-1) GFBs In Step I, we select connections between layer-(k-1) GFBs to construct layer-k GFB. We construct the connections between layer-(k-1) GFBs by the following steps.

Step I-1 Compute the number of links L_k^{GFB} necessary to connect one layer-(k-1) GFB to other layer-(k-1) GFBs by

$$L_k^{\mathrm{GFB}} = L_k \prod_{i=1}^{k-1} N_i. \tag{3.1}$$

Step I-2 If L_k^{GFB} is larger than (N_k-1), connect all layer-(k-1) GFBs. Otherwise, construct a ring topology by connecting GFBs with neighboring IDs.

Step I-3 Compute the number of residual links $L_k^{'\mathrm{GFB}}$ which can be used to connect one layer-(k-1)

GFB to other layer-(k-1) GFBs. This number is given as

$$L_k'^{\mathrm{GFB}} = L_k^{\mathrm{GFB}} - \bar{L}_k^{\mathrm{GFB}}, \tag{3.2}$$

where $\bar{L}_k^{\mathrm{GFB}}$ is the number of links per layer-(k-1) GFB constructed at Step I-2.

Step I-4 Check whether layer-(k-1) GFBs have any residual links which can be used to connect layer-(k-1) GFBs. If there are residual links, connect GFB of ID $D_{k-1}^{\mathrm{GFB}}(a)$ to GFB of ID $D_{k-1}^{\mathrm{GFB}}(b)$ when the following equation is satisfied:

$$D_{k-1}^{\mathrm{GFB}}(b) = (D_{k-1}^{\mathrm{GFB}}(a) + \lceil p_k \rceil + C_k^{\mathrm{r}} \lfloor p_k \rfloor) \bmod N_k, \tag{3.3}$$

where C_k^{r} ($C_k^{\mathrm{r}} = 0,1,\cdots,L_k'^{\mathrm{GFB}}$-1) is an integer value that represents the number of residual links used for connections. Furthermore, p_k is a variable that represents the interval containing the IDs of layer-(k-1) GFBs connected to the same layer-(k-1) GFBs. p_k is obtained by

$$p_k = \frac{N_k}{L_k'^{\mathrm{GFB}} + 1}. \tag{3.4}$$

Selection of Switches for Connecting layer-(k-1) GFBs In Step II, we select the switches used for connecting layer-(k-1) GFBs after constructing the connections between layer-(k-1) GFBs. Switch $D_{\mathrm{connect}}^{\mathrm{sw}}(s)$ included in GFB of ID $D_{k-1}^{\mathrm{GFB}}(a)$ is connected to GFB of ID $D_{k-1}^{\mathrm{GFB}}(b)$ when the following condition is satisfied:

$$D_{\mathrm{connect}}^{\mathrm{sw}}(s) = D_{k-1}^{\mathrm{GFB}}(b) + \left\lfloor \frac{C_k(s) n_{D_{k-1}^{\mathrm{GFB}}(a)}}{l_{(D_{k-1}^{\mathrm{GFB}}(a), D_{k-1}^{\mathrm{GFB}}(b))}} \right\rfloor$$

where $C_k(s)$ ($C_k(s)$=0,1,$\cdots$,L_k-1) is an integer value that represents that the number of links used for connections between GFB of ID $D_{k-1}^{\mathrm{GFB}}(a)$ and GFB of ID $D_{k-1}^{\mathrm{GFB}}(b)$, $n_{D_{k-1}^{\mathrm{GFB}}(a)}$ is the number of switches in GFB of ID $D_{k-1}^{\mathrm{GFB}}(a)$, and $l_{(D_{k-1}^{\mathrm{GFB}}(a), D_{k-1}^{\mathrm{GFB}}(b))}$ is the number of links to be constructed between GFBs of IDs $D_{k-1}^{\mathrm{GFB}}(a)$ and $D_{k-1}^{\mathrm{GFB}}(b)$. By connecting switches using the above condition, the intervals containing IDs of switches connected to the same GFB become constant, and we can avoid a large number of hops from any switch belonging to a GFB to other GFBs.

$D^{GFB}(\boldsymbol{d})$	$D^{GFB}(\boldsymbol{c})$
[1,1,1]	[1,1,1]
[1,1,2]	[1,1,2]
[1,1,3]	[1,1,2]
[1,1,4]	[1,1,5]
[1,1,5]	[1,1,5]
[1,2,*]	[1,2,1]
[1,3,*]	[1,1,2]
⋮	⋮
[2,*,*]	[2,1,1]
⋮	⋮
[6,*,*]	[1,3,1]

Figure 3.3: Routing table for a ToR switch belonging to the GFB of ID [1,1,1]

3.2.3.2 GFB Routing

In GFB, routes are established based on GFB IDs. Below, we call routing for GFB *GFB routing*. In GFB routing, packets to a destination ToR switch belonging to a layer-*k* GFBs different from that of the source ToR switch, are sent via the following path.

Step R-1 Select the ToR switch that belongs to the same GFB as the source switch and is connected to the destination GFB. This ToR switch is obtained from the GFB parameters. If multiple ToR switches are connected to the destination GFB, select one of them randomly.

Step R-2 Encapsulate the packet so that its destination becomes the selected ToR switch, and then relay the packet.

Step R-3 The ToR switch with the ID specified in the encapsulated packet decapsulates the packet.

By repeating these steps, we can relay the packet to the destination ToR switch.

The above routing can be implemented as a routing table. Because GFB is constructed in a hierarchical manner, the routing table is also hierarchically aggregated. An example routing table is shown in Figure 3.3. In the figure, $D^{\mathrm{GFB}}(c)$ represents a destination address of an encapsulated packet for a packet to $D^{\mathrm{GFB}}(d)$.

The routing table can be set by the following steps. First, entries in the routing table for ToR switches in the same layer-1 GFB are set up by a shortest-hop-count routing algorithm, such as the Dijkstra algorithm. Because the number of ToR switches in each layer-1 GFB is small, this configuration takes only a short time. Then, the entries for the different layer-k (k=2,3,$\cdots$,$K_{\max}$) GFBs are set as follows: if the switch is connected to a ToR switch in the destination GFB, the entry is set to the ToR switch in the destination GFB. Otherwise, the entry is set to represent the encapsulation of the packet with the ID of a ToR switch connected to the destination GFB. The calculation time for the routing table in GFB routing is $O(\sum_{1<=i<=K_{\max}} N_i)$, which is significantly shorter than $O(N)$.

3.2.3.3 Properties of GFB

In GFB, the maximum number of hops or the number of paths passing through each link can be obtained from the GFB parameters as described below.

Maximum Number of Hops The maximum number of hops H_k between switches in layer-k GFB is obtained by

$$H_k = (h_k + 1)H_{k-1} + h_k, \tag{3.5}$$

where h_k is the largest number of links between layer-(k-1) GFBs passed through by the traffic between layer-(k-1) GFBs. h_k is obtained by the following steps. If all layer-(k-1) GFBs are connected directly, then $h_k = 1$. In all other cases, we obtain h_k by computing the largest number of links between layer-(k-1) GFBs passed through by the traffic from the source layer-(k-1) GFB whose ID is 0, because all GFBs are equivalent. From the viewpoint of the source GFB, the topology constructed of layer-(k-1) GFBs is a ring topology where some shortcut links are added directly from the source GFB. To obtain h_k, we divide the set of GFBs which are not directly connected to the source GFB into groups so that the shortcut links from the source GFB become the border of the group. Here, m_j denotes the set of switches within the j-th group, and M denotes the set of all groups. Based on the steps required to construct the connections between layer-(k-1) GFBs, $|m_j|$ is obtained by

$$|m_j| = \begin{cases} \lceil p_k \rceil - 3, & \text{if } j = 1 \text{ or } |M|, \\ \lfloor p_k \rfloor - 2, & \text{otherwise.} \end{cases} \tag{3.6}$$

GFBs in each group form a ring topology. Thus, the maximum number of links passed through by traffic from the source GFB or from the GFB belonging to group m_j is obtained by $\left\lceil \frac{|m_j|+2}{2} \right\rceil$. Since at least one group includes GFBs for which the number of hops from the source GFB is the largest, h_k reaches a maximum of $\left\lceil \frac{|m_j|+2}{2} \right\rceil$ for all groups. That is,

$$h_k = \begin{cases} 1, & \text{if } L_k^{\text{GFB}} \geq (N_k - 1), \\ \lceil \frac{p_k}{2} \rceil, & \text{if } L_k^{\text{GFB}} < (N_k - 1) \text{ and } L_k'^{\text{GFB}} \leq 1, \\ \lceil \frac{\lfloor p_k \rfloor + 2}{2} \rceil, & \text{otherwise.} \end{cases} \tag{3.7}$$

Number of Flows through a Link The number of layer-(k-1) GFB source-destination pairs whose traffic passes through link l between layer-(k-1) GFBs (denoted as x_l^k) is obtained by calculating the number of flows passing through the link in an abstracted topology where layer-(k-1) GFB is regarded as a single node. Multiplying that number by the number of flows passing through layer-(k-1) GFBs, we obtain the number of flows passing through each link. Since all layer-k GFBs are equivalent, the number of flows between a pair of layer-(k-1) GFBs is independent of the GFB ID.

Thus, the number of flows X_l^k passing through link l between layer-(k-1) GFBs is obtained by

$$X_l^k = F_k x_l^k, \tag{3.8}$$

where F_k is the number of flows from a layer-(k-1) GFB to other layer-(k-1) GFBs. x_l^k and F_k can be obtained as follows. In an abstracted topology where lower-layer GFBs are regarded as single nodes, there are two types of links: links in the ring topology (*ring links*), and links added as shortcuts in the ring topology (*shortcut links*). Since all layer-(k-1) GFBs are equivalent in layer-k GFB, the number of flows passing through each ring link is independent of the GFBs connected to the link. Similarly, the number of flows passing through each shortcut link is also independent of GFBs connected to the link. Therefore,

$$x_l^k = \begin{cases} \frac{M_k^{\text{ring}}}{2\prod_{i=1}^{k} N_i}, & \text{if } l \text{ is a ring link,} \\ \frac{M_k^{\text{shortcut}}}{(L_k-2)\prod_{i=1}^{k} N_i}, & \text{if } l \text{ is a shortcut link,} \end{cases} \tag{3.9}$$

where M_k^{ring} is the total number of ring links passed through by traffic between layer-(k-1) GFB source-destination pairs, and M_k^{shortcut} is the total number of shortcut links passed through by traffic between layer-(k-1) GFB source-destination pairs. $2\prod_{i=1}^{k} N_i$ is the number of ring links between layer-(k-1) GFBs, and $(L_k - 2)\prod_{i=1}^{k} N_i$ is the number of shortcut links between layer-(k-1) GFBs.

Traffic between layer-(k-1) GFBs passes through at most one shortcut link because the portion of GFBs connected to a certain GFB is constant. The number of flows that do not pass through a shortcut link is $2h_k\prod_{i=1}^{k} N_i$. Thus,

$$M^{\text{shortcut}} = \prod_{i=1}^{k} N_i(\prod_{i=1}^{k} N_i - 1) - 2h_k\prod_{i=1}^{k} N_i.$$

In addition, M^{ring} is obtained by subtracting M^{shortcut} from the total number of links passed through by traffic between layer-(k-1) GFBs;

$$M^{\text{ring}} = \sum_{i=1}^{h_k} i s_k(i) - M^{\text{shortcut}},$$

where $s_k(i)$ is the number of layer-(k-1) GFB source-destination pairs whose traffic passes through i links in the abstracted topology.

$s_k(i)$ is obtained as follows. $s_k(1)$ has the same value as the number of links in layer-k GFB.

That is,

$$s_k(1) = \begin{cases} N_k(N_k - 1), & \text{if } L_k^{\mathrm{GFB}} \geq (N_k - 1), \\ N_k L_k \prod_{i=1}^{k-1} N_i, & \text{otherwise.} \end{cases} \tag{3.10}$$

$s_k(i)$ for $i > 1$ is obtained by dividing the topology constructed of layer-(k-1) GFBs into groups, similar to the case of calculating h_k. By dividing the topology, $s_k(i)$ is obtained as the sum of the number of layer-(k-1) GFBs located i hops away from the source layer-(k-1) GFB in each group. In other words,

$$s_k(i) = N_k \sum_{m_j \in M} U_{(k,m_j)}(i), \tag{3.11}$$

where $U_{(k,m_j)}(i)$ is the number of layer-(k-1) GFBs located i hops away from the source layer-(k-1) GFB in group m_j. For GFBs in each group, the source GFB and GFBs directly connected to the source GFB form a ring topology,

$$U_{(k,m_j)}(i) = \begin{cases} 0, & \text{if } i > \left\lceil \frac{m_j+2}{2} \right\rceil, \\ 1, & \text{if } i = \left\lceil \frac{m_j+2}{2} \right\rceil \text{ and } |m_j| \text{ is odd,} \\ 2, & \text{otherwise.} \end{cases} \tag{3.12}$$

We determine the number of flows between each two layer-(k-1) GFBs (denoted as F_k), which is independent of the IDs of the source and destination GFBs. Thus, we determine the number of flows between layer-(k-1) GFBs s and d (denoted as $F_k^{s \to d}$) as follows:

$$\begin{aligned} F_k^{s \to d} &= f_k^{s \to s \to d \to d} + \sum_{n \in G} f_k^{n \to s \to d \to d} \\ &+ \sum_{n \in G} f_k^{s \to s \to d \to n} + \sum_{n_1, n_2 \in G} f_k^{n_1 \to s \to d \to n_2}, \end{aligned} \tag{3.13}$$

where $f^{a \to b \to c \to d}$ is the number of flows whose source and destination switches belong to layer-(k-1) GFBs a and d, respectively, and that traverse layer-(k-1) GFBs b and c. G is the set of switches that do not belong to layer-k GFB including layer-(k-1) GFBs s and d. $f_k^{s \to s \to d \to d}$ is obtained as the product of the respective numbers of switches in layer-(k-1) GFBs s and d. That is,

$$f_k^{s\to s\to d\to d} = \prod_{i=1}^{k-1}(N_i)^2. \tag{3.14}$$

$\sum_{n\in G} f_k^{s\to s\to d\to n}$ shows the number of flows from layer-(k-1) GFB s to the outside of layer-k GFB via layer-(k-1) GFB d. Because all layer-(k-1) GFBs are equivalent in GFB, $\sum_{n\in G} f_k^{s\to s\to d\to n}$ is obtained by dividing the number of flows whose source and destination switches belong to layer-(k-1) GFB s and a different layer-k GFB, respectively, by the number of layer-(k-1) GFBs in layer-k GFB.

$$\sum_{n\in G} f_k^{s\to s\to d\to n} = \frac{(\prod_{i=1}^{k-1} N_i)(\prod_{i=1}^{K_{\max}} N_i - \prod_{i=1}^{k} N_i)}{N_k}. \tag{3.15}$$

Similarly, $\sum_{n\in G} f_k^{n\to s\to d\to d}$ is obtained by

$$\sum_{n\in G} f_k^{n\to s\to d\to d} = \frac{(\prod_{i=1}^{k-1} N_i)(\prod_{i=1}^{K_{\max}} N_i - \prod_{i=1}^{k} N_i)}{N_k}. \tag{3.16}$$

$\sum_{n_1,n_2\in G} f_k^{n_1\to s\to d\to n_2}$ shows the number of flows that travel from the outside of layer-k GFB via layer-(k-1) GFB s to the outside of layer-k GFB via layer-(k-1) GFB d. The number of flows traveling from the outside of layer-k GFB via layer-(k-1) GFB s is the sum of flows through links that connect switches in layer-(k-1) GFB s and switches outside layer-k GFBs, which is obtained by

$$\prod_{j=1}^{k-1} N_j \sum_{i=k+1}^{K} (X_l^i L_i). \tag{3.17}$$

We finally obtain the number of flows that travel from the outside of layer-k GFB via layer-(k-1) GFB s to layer-(k-1) GFB d by dividing Eq. (3.17) by the number of layer-(k-1) GFBs in layer-k GFB. The resulting value includes the flows whose destination switches belong to layer-(k-1) GFB

d, whose number is $\sum_{n_1 \in G} f_k^{n_1 \to s \to d \to d}$. Therefore, $\sum_{n_1, n_2 \in G} f_k^{n1 \to s \to d \to n_2}$ is obtained by

$$\sum_{n_1, n_2 \in G} f_k^{n_1 \to s \to d \to n_2} = \frac{\prod_{j=1}^{k-1} N_j \sum_{i=k+1}^{K} (X_l^i L_i)}{N_k} - \sum_{n_1 \in G} f_k^{n_1 \to s \to d \to d}. \tag{3.18}$$

Difference from FB GFB is an extension of FB, and FB is obtained from GFB if we set $L_k = N_k$ - 1 and set N_k the same for all layers. In FB, we cannot set the number of links independently from the number of nodes N_k. Thus, if the number of nodes in the data center is large, N_k should be large, requiring a large number of links even when the traffic amount is small. In GFB, the parameter L_k can be set independently from the other parameters for each layer. Thus, we construct a topology where only links required to accommodate the instantaneous traffic are established.

3.3 Virtual network reconfiguration for data center networks

In this section, we propose a method for adjusting the topology of a virtual network to meet the requirements for minimizing the energy consumption of data center networks. In VNR-DCN, the virtual network is constructed by calculating the GFB parameters to satisfy these requirements.

3.3.1 Overview

The VNR-DCN considers the both types of the traffic changes; frequent changes in traffic pattern and gradual changes of the total amount of traffic. The VNR-DCN reconfigures the virtual network to handle the changes of the total amount of traffic, while the frequent changes of the traffic pattern are handled by the load balancing over the virtual network.

3.3.1.1 Virtual Network Control

In VNR-DCN, the virtual network and the traffic routing are adjusted by a single network controller (NC) and multiple route controllers (RCs). NC ensures that the requirements on the network set by the data center administrator are met, and collects information about the amount of traffic from

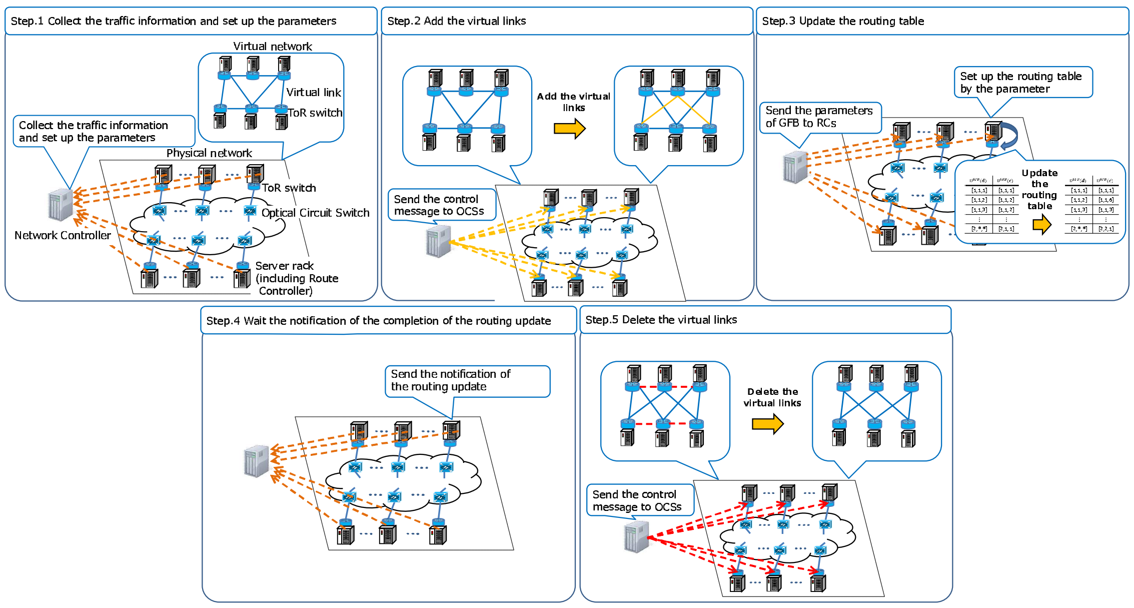

Step.1 Collect the traffic information and set up the parameters
Virtual network
Virtual link
ToR switch
Collect the traffic information and set up the parameters
Physical network
ToR switch
Optical Circuit Switch
Network Controller
Server rack (including Route Controller)
Step.2 Add the virtual links
Add the virtual links
Send the control message to OCSs
Step.3 Update the routing table
Set up the routing table by the parameter
Send the parameters of GFB to RCs
Update the routing table
Step.4 Wait the notification of the completion of the routing update
Send the notification of the routing update
Step.5 Delete the virtual links
Delete the virtual links
Send the control message to OCSs

Figure 3.4: Overview of VNR-DCN

or to ToR switches. Then, NC sets the GFB parameters to satisfy the current requirements on the network and to accommodate the instantaneous traffic demand, after which it configures the OCSs and sends the GFB parameters to RCs. Each server rack hosts a RC. When the virtual network is reconfigured, RCs set the routing rules for the ToR switch in the same rack based on the GFB parameters sent by NC.

In VNR-DCN, NC reconfigures the virtual network by the following steps as shown in Figure3.4.

Step VN-1 Collect traffic information and set the GFB parameters to satisfy the requirements.

Step VN-2 Configure OCSs to add virtual links that are not included in the current virtual network but are included in the virtual network with the new GFB with the parameters set in the previous step.

Step VN-3 Send the GFB parameters to RCs, after which RCs update the routing table for ToR switches.

Step VN-4 Wait for a notification of the completion of routing update from the RCs.

Step VN-5 Configure OCSs to delete unused virtual links.

In the above procedure, the routing tables are updated by the method described in Section 3.2.3.2.

3.3.1.2 Load balancing over the Virtual Network

In VNR-DCN, we use a load balancing technique called valiant load balancing (VLB) [69] combined with the GFB routing. In VLB, we select the intermediate switches randomly, regardless of the destination, in order to avoid the concentration of traffic at any particular links, even when the traffic amount flowing between a certain pair of switches is large. Then, traffic is sent from the source switch to the destination switch via an intermediate switch. The VLB is combined with the GFB routing as follows; (1) Each server encapsulates the packets from it with the addresses of the randomly selected ToR switches, and (2) the encapsulated packets are relayed by the GFB routing.

By applying VLB, the amount of traffic between each ToR switch pair T is obtained by the following equation:

$$T \leq \frac{T^{\mathrm{toSW}} + T^{\mathrm{fromSW}}}{N_{\mathrm{all}}}, \tag{3.19}$$

where T^{toSW} is the maximum traffic amount to a ToR switch, T^{fromSW} is the maximum traffic amount from a ToR switch, and N_{all} is the number of ToR switches in the virtual network. Thus, we can ensure sufficient bandwidth by setting the virtual network so that the number of flows passing a link is smaller than a certain threshold obtained by dividing the capacity of a link by the traffic amount between each switch pair calculated from Eq. (3.19). The rest of this section explains how NC set the GFB parameters considering the load balancing.

3.3.2 GFB Parameter Setup

3.3.2.1 Outline

We propose a method to set GFB parameters so as to minimize the number of used ports by considering two requirements: large bandwidth and short delay between servers. When using VLB, based on Eq. (3.19), the traffic amounts between ToR switch pairs depends only on the total amount of traffic from or to ToR switches. Delays are also difficult to predict when designing a virtual network. In this study, we avoid long delays by providing sufficient bandwidth and ensuring that the maximum number of hops does not exceed a certain threshold.

3.3.2.2 Steps to Set Suitable GFB Parameters

In this subsection, we describe a method to set up the GFB parameters so as to minimize the number of used ports and to satisfy the requirements on bandwidth and maximum number of hops between servers. In VNR-DCN, the GFB parameters are set based on the number of switches connected in the virtual network (N_{all}), the maximum number of hops ($H_{\max}$), the maximum traffic amount from a ToR switch (T^{fromSW}), and the maximum traffic amount to a ToR switch (T^{toSW}) by the following steps.

First, we obtain the candidates for the number of layers. Because the maximum number of hops in GFB cannot be smaller than 1 (Eq. (3.5)) for any layer, to take the maximum number of hops to be no more than $H_{\max}$, and the number of layers ($K_{\max}$) must satisfy the following condition.

$$2^{K_{\max}} - 1 \leq H_{\max}. \tag{3.20}$$

We define K_{can} as the set of numbers of layers satisfying Eq.(3.20). We consider all $K_{\max}$ ($K_{\max} \in K_{\mathrm{can}}$) as candidates of the number of layers. For each candidate, we choose suitable parameters by the following steps.

Step P-1 Set the parameters considering the acceptable number of hops.

Step P-2 Modify the parameters to provide sufficient bandwidth.

Then, we construct the topology that uses the smallest number of virtual links from among the candidates. The details of the above steps are described in the following paragraphs.

Parameter Setting Considering the Acceptable Number of Hops We set parameters N_k and L_k so as to ensure that the maximum number of hops is no more than $H_{\max}$. In this step, to reduce the number of variables, we set N_k to $\prod_{i=1}^{k-1} N_i + 1$ for $1 < k < K_{\max}$. In this way, h_k becomes 1 even when $L_k = 1$. To connect $N_{\rm all}$ switches, $N_{K_{\max}}$ must satisfy the following equation.

$$N_{K_{\max}} = \left\lceil \frac{N_{\rm all}}{\prod_{i=1}^{k-1} N_i} \right\rceil. \tag{3.21}$$

In this step, we also set $L_{K_{\max}}$ so that $h_{K_{\max}}$ becomes 1 to reduce the number of variables. To ensure that $h_{K_{\max}} = 1$, $L_{K_{\max}}$ should satisfy the following equation.

$$L_{K_{\max}} = \left\lceil \frac{N_{K_{\max}}}{\prod_{i=1}^{k-1} N_i} \right\rceil. \tag{3.22}$$

To ensure that the maximum number of hops is no more than $H_{\max}$, h_1 must satisfy the following condition, according to Eq. (3.7).

$$h_1 \leq \left\lceil \frac{H_{\max}+1}{2^{K_{\max}-1}} - 1 \right\rceil. \tag{3.23}$$

To satisfy Eq. (3.23), L_1 should satisfy the following equation.

$$L_1 = \begin{cases} N_1 - 1, & \text{if } h_1 = 1 \\ 2, & \text{if } h_1 \geq \lfloor \frac{N_1}{2} \rfloor \\ \lfloor \frac{N_1}{2h_1} + 1 \rfloor, & \text{otherwise.} \end{cases} \tag{3.24}$$

In the above condition, all N_k $(k > 1)$ and L_k $(k \geq 1)$ are obtained in order from N_1 to $N_{\max}$. The objective of our parameter setting procedure is to minimize the number of used ports of ToR switches. That is, we minimize $\sum_{1 \leq k \leq K_{\max}} L_k$. Since $\sum_{1 \leq k \leq K_{\max}} L_k$ is a convex function of N_1, we find the N_1 that minimizes $\sum_{1 \leq k \leq K_{\max}} L_k$ by incrementing N_1 as long as $\sum_{1 \leq k \leq K_{\max}} L_k$ decreases.

Parameter Modification to Ensure Sufficient Bandwidth If GFB with the parameter set obtained at Step 1 cannot provide sufficient bandwidth, we add links to the layers with insufficient bandwidth. To detect insufficient bandwidth, we check whether the following condition is satisfied for each layer-k.

$$TX_l^k \leq BU, \tag{3.25}$$

where B is the bandwidth of one link, U is the target maximum link utilization, and T is obtained by Eq. (3.19). If Eq. (3.25) is not satisfied, L_k is incremented until Eq. (3.25) is satisfied.

3.3.2.3 Calculation Time

The virtual network control method should be applicable to large data center networks, and the calculation time for the suitable virtual network parameters should be short. The computational complexity for determining the GFB parameters is $O(N|K_{\text{can}}|)$, where N is the number of ToR switches and K_{can} is the set of candidate for the number of layers in GFB. From Eq. 3.20, $|K_{\text{can}}| \sim \log 2H_{\text{max}}$ where H_{max} is the acceptable number of hops which is set by the administrator. Therefore, unless the administrator sets H_{max} to the significantly large value, the calculation time for the determining the GFB parameters is small. In the evaluation discussed in Section 3.4, the parameters can be obtained within few milliseconds for a network with 420 ToR switches, by using a computer with a 3.06 GHz Intel Xeon X5675 processor.

3.3.3 Implementation Issues

In this section, we discuss certain issues related to the implementation of VNR-DCN.

3.3.3.1 Collection of Traffic Information

In VNR-DCN, NC needs to collect traffic information to compute the GFB parameters. NC requires only the amount of traffic from or to ToR switches, which can be obtained by collecting information from the management information base from ToR switches via Simple Network Management Protocol.

3.3.3.2 Configuration of an Optimal Virtual Network

In VNR-DCN, the virtual network is constructed by setting the GFB parameters. Thus, VNR-DCN can be implemented by deploying an NC implementing the method described in Subsection 3.3.1. OCSs accept remote commands that represent the configuration of the OCSs. By sending such commands, NC adds or deletes virtual links.

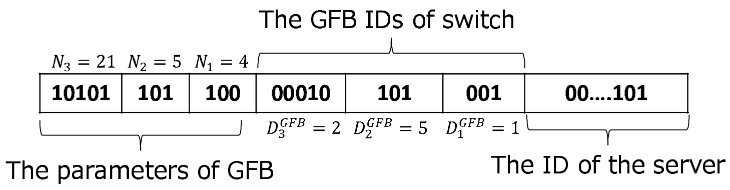


Figure 3.5: Encoding GFB IDs

3.3.3.3 Routing Table Update

Routing in GFB can be implemented by using generic routing encapsulation (GRE) and OpenFlow. In this approach, we use GRE to establish a tunnel from any ToR switch to ToR switches connected to different GFBs. Then, each ToR switch selects the next packet destination from among the directly connected ToR switches and GRE tunnels according to the rules within the ToR switch set by the RC.

One approach to implementing GFB routing in the rules of OpenFlow is encoding GFB IDs into an IPv6 address and using OpenFlow switches. An example of encoding GFB IDs is shown in Figure 3.5, where the GFB parameters are stored in the top 11 bits of the IPv6 address. The GFB ID of the ToR switch is stored in the following 11 bits of the IPv6 address, so that the GFB ID of the upper layer becomes the prefix. The remaining bits represent the ID of the server. By encoding the network in this manner, the GFB routing table can be implemented by using the longest matching prefix. In addition, we can also find packets whose GFB parameters are different from the current parameters. Then, we add rules stating that such packets are relayed to the RC, and RC changes the IPv6 address to fit to the current parameters. In this way, we avoid packet loss even if there are packets in the virtual network during its reconfiguration.

3.4 Evaluation

In this section, we evaluate the performance of a virtual network constructed by VNR-DCN. First, we investigate whether the virtual network constructed by VNR-DCN satisfies the requirements given as input to VNR-DCN. Next, we investigate the number of virtual links required to meet the

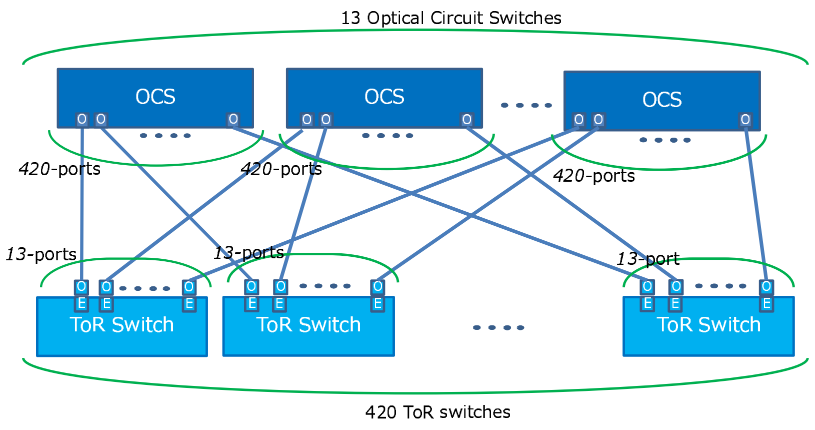


Figure 3.6: Overall structure of physical network used in this evaluation

requirements of VNR-DCN by comparing the results with existing data center networks constructed by parameter setting. Finally, we investigate the impact of reconfiguring the virtual network in terms of reduction in energy consumption by comparing the results with the method based on powering down the ports of ToR switches of a static electronic network.

In the evaluation, we use a physical network shown in Figure 3.6. This network includes 420 server racks and multiple OCSs with 420 ports. Each OCS is connected to all server racks. We assume that the number of ports of the ToR switches and the number of OCSs are 13.

In our evaluation, the traffic amount between ToR switch is generated by the following steps, considering the fact that each server communicates with about 1–10% of other severs in a data center [70]. First, we select the communicating ToR switch pairs randomly so that 5% of all ToR switch pairs communicate with each other, and generate traffic amount based on the uniformly distributed random values. Then, we scale the traffic amount so that the maximum amount of traffic from or to each ToR switch becomes the predefined traffic amount from the ToR switch.

In this section, we discuss the results of the evaluation based on the traffic amount normalized by the bandwidth of one virtual link, because the number of required virtual links depends on the ratio of the generated traffic amount to the bandwidth of each virtual link instead of the traffic amount itself.

3.4.1 Evaluation of the Virtual Network Satisfying the Bandwidth and Delay Requirements

In this subsection, we show that VNR-DCN can construct a virtual network satisfying the requirements that the virtual network accommodate all traffic demand and that the maximum number of hops be no more than a certain threshold.

3.4.1.1 Routing

Routes over the virtual network are established by the following three policies: shortest path (SP), a combination of VLB and shortest path routing (SP-VLB), and a combination of VLB and GFB routing (GFBR-VLB). In the cases of SP-VLB and GFBR-VLB, the traffic from a source ToR switch to a destination ToR switch is distributed by selecting the intermediate ToR switch randomly, as mentioned in Subsection 3.3.2.1.

3.4.1.2 Evaluation Metrics

Although the total amount of traffic in the data center changes gradually, the connections between communicating servers may change significantly even within a few seconds [71, 70]. Therefore, we generate multiple random traffic patterns under the constraint that the total amount of traffic from or to each ToR switch be less than a predefined value, and consider the maximum link utilization among all traffic patterns to check whether the virtual network can accommodate all traffic. In this evaluation, we generate 10 traffic patterns. Each traffic pattern is generated by selecting the communicating ToR switches randomly and generating traffic between the selected ToR switches under the constraint on the total traffic from or to each ToR switch.

We also investigate the number of hops in the virtual network constructed by VNR-DCN to examine whether the constructed virtual network satisfies the requirements on the number of hops. In addition, we investigate the impact of GFB routing compared with SP routing.

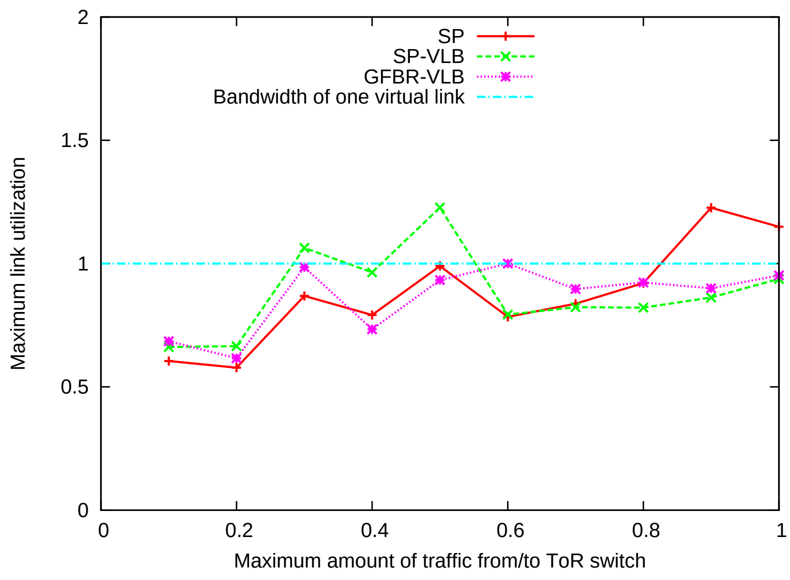


Figure 3.7: Maximum link utilization required to accommodate traffic from/to ToR switches

3.4.1.3 Result

First, we show the maximum link utilization in Figure 3.7, where the horizontal axis denotes the maximum amount of traffic from or to ToR switches, and the vertical axis denotes the maximum link utilization. Note that the maximum link utilization is independent from the maximum amount of traffic from/to ToR switch, because the number of constructed virtual links are different; as the amount of traffic increases, the number of constructed virtual links becomes large. This figure indicates whether the constructed virtual network can accommodate traffic without congestion.

In the cases of SP and SP-VLB, the virtual network cannot accommodate traffic without high link utilization, even when the virtual network is constructed by VNR-DCN. This is caused by insufficient load balancing. In contrast, in the case of GFBR-VLB, the maximum link utilization is always lower than the bandwidth of one virtual link, indicating that VNR-DCN can configure a suitable virtual network that can accommodate any traffic demand. In addition, this result shows that GFBR-VLB is required in order to implement load balancing capable of accommodating any traffic demand under the constraints on the amount of traffic from or to each ToR switch.

Figures 3.8 and 3.9 respectively show the maximum and average number of hops in GFB under

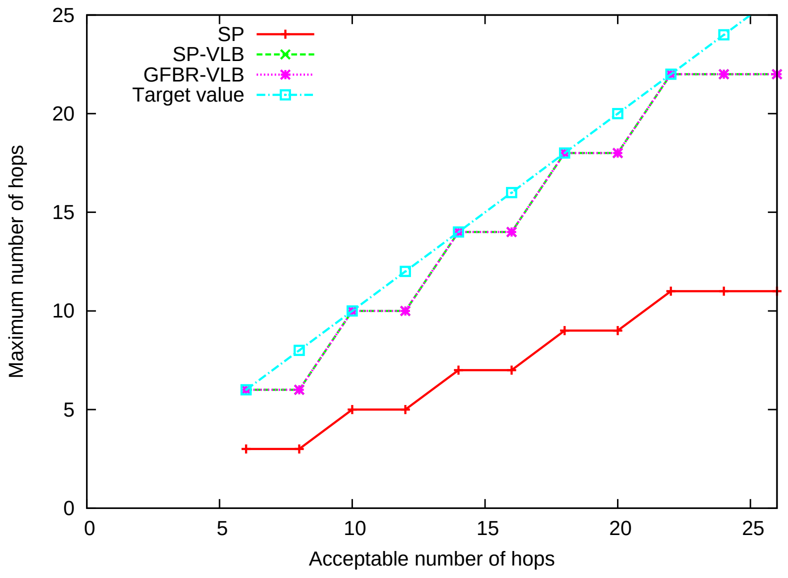


Figure 3.8: Maximum necessary hop count vs. hop limit

the constraint that the maximum number of hops be no more than the acceptable number of hops. The horizontal axes in both figures denote the acceptable number of hops, and the vertical axes denote the maximum and average number of hops, respectively. The virtual network constructed by VNR-DCN meets the requirements on the acceptable number of hops in all cases. GFBR-VLB uses a similar number of hops to SP-VLB. Although the average number of hops in GFBR-VLB is slightly larger than that in SP-VLB, GFB routing does not cause a significant increase in the number of hops.

As discussed above, GFBR-VLB can successfully balance the load and does not use an excessively large number of hops. In addition, the routing table for GFB routing can be constructed immediately after virtual network reconfiguration. Therefore, GFB routing is suitable for application to our virtual network reconfiguration.

3.4.2 Comparison with Existing Data Center Network Topologies

In this subsection, we show that GFB is appropriate for use as the base topology in VNR-DCN. In this evaluation, we investigate the number of virtual links required by GFB under the constraints

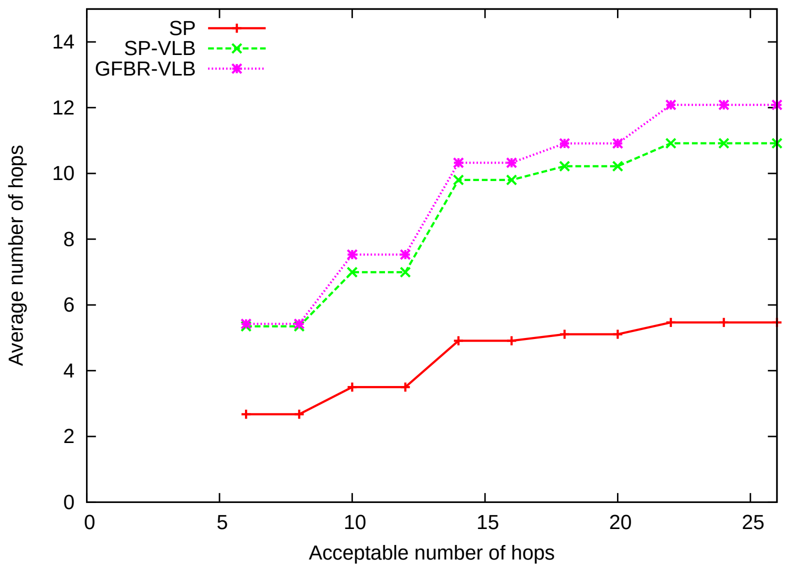


Figure 3.9: Average vs. acceptable number of hops

that it can provide sufficient bandwidth and that the maximum number of hops be no more than $H_{\max}$. We compare the number of virtual links in GFB with existing data center network topologies (FatTree, Torus, Switch-based DCell, and FB). These data center network topologies are configured by setting their corresponding parameters so as to satisfy the same constrains as in the case of GFB. The original FatTree topology is not suitable for implementing a virtual network considering the high energy consumption, as discussed in Subsection 3.2.2. Therefore, in this evaluation, unlike the FatTree topology proposed by Al-Fares et al. [27], we assume that all switches are connected to servers. In VNR-DCN, the GFB parameters are set to minimize the number of virtual links required by the topology under the constraints that it can provide sufficient bandwidth and that the maximum number of hops be no more than $H_{\max}$. Thus, the parameters of the other topologies are also set to minimize the number of virtual links required under these constraints.

First, we investigate the number of required virtual links when the amount of traffic from or to ToR switches is changed. In this evaluation, the amount of traffic from or to each ToR switch is the same. Also, we assume that the traffic from ToR switches is balanced by VLB. The results are shown in Figure 3.10, where the horizontal axis denotes the maximum traffic amount from or to

ToR switches that must be accommodated, and the vertical axis denotes the number of used links required to meet the traffic demand.

Clearly, VNR-DCN uses the smallest number of links to accommodate traffic, regardless of the amount of traffic, while other topologies either require a large number of ports (FB) or cannot accommodate the required amount of traffic with any parameter settings (Switch-based DCell, Fat-Tree, and Torus). This is because in setting the GFB parameters, VNR-DCN adds only links that are necessary to accommodate the traffic. Therefore, the topology constructed by VNR-DCN satisfies the requirement on bandwidth with the lowest energy consumption.

We also compare the number of used links required to meet the requirements on the acceptable number of hops. In this comparison, we assumed that the capacity of each virtual link is sufficient. The results are shown in Figure 3.11, where the horizontal axis denotes the maximum number of hops, and the vertical axis denotes the number of virtual links required to satisfy the requirements. In all cases of the acceptable maximum number of hops, the topology constructed by VNR-DCN uses the smallest number of virtual links to satisfy the requirements. This is again because VNR-DCN adds only links that are necessary, thus maintaining the maximum number of hops at no more than the required value.

3.4.3 Impact of virtual network reconfiguration on energy consumption

In this subsection, we demonstrate that the proposed virtual network reconfiguration reduces the energy consumption, even though the addition of the OCSs is required. In this evaluation, we change the maximum amount of traffic from or to ToR switches from 1 to 0.1 but maintain the same number of physical links.

3.4.3.1 Comparison Method

In this evaluation, we compare VNR-DCN with the following two cases where the network is constructed without OCSs.

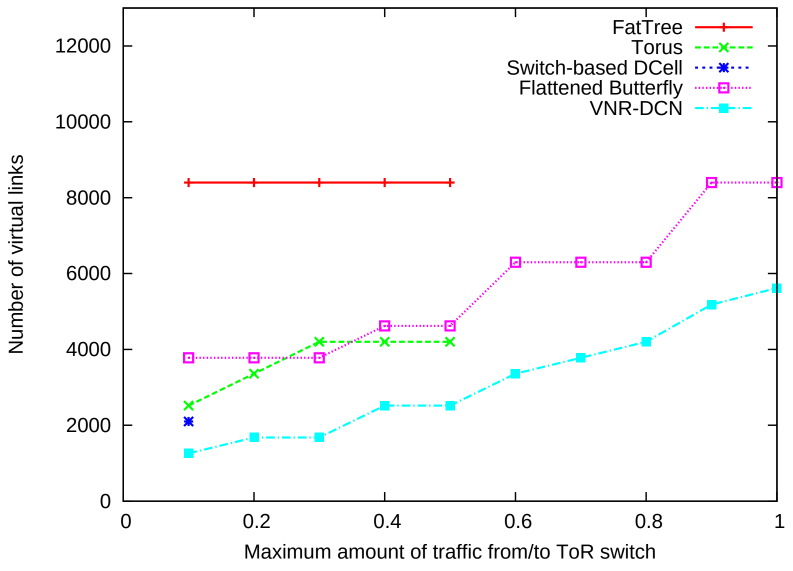


Figure 3.10: Number of virtual links required to accommodate the traffic from ToR switches

Static Electronic Network This network is constructed using only electronic switches, and all ports of the switches are always powered on. In our evaluation, we construct this network to connect the electronic ports of ToR switches. Comparing VNR-DCN with this network construction method, we demonstrate the strong impact of powering down switch ports on the energy consumption. In our evaluation, the topology of the static electronic network is set as in the case of GFB, whose parameters are set so as to accommodate the maximum amount of traffic generated in the evaluation because GFB accommodates the maximum amount of traffic with the smallest number of links.

Elastic Tree The network topology can be reconfigured without using OCSs. Heller et al. proposed a method for reconfiguring the topology by powering down the ports of electronic switches [11]. In our evaluation, we also use this method in the abovementioned electronic network to reduce the energy consumption. Comparing VNR-DCN with this method, we demonstrate the impact of the reconfiguration of the virtual network using OCSs. The method used in Heller et al. is based on FatTree, which requires additional switches, resulting in increased energy consumption compared

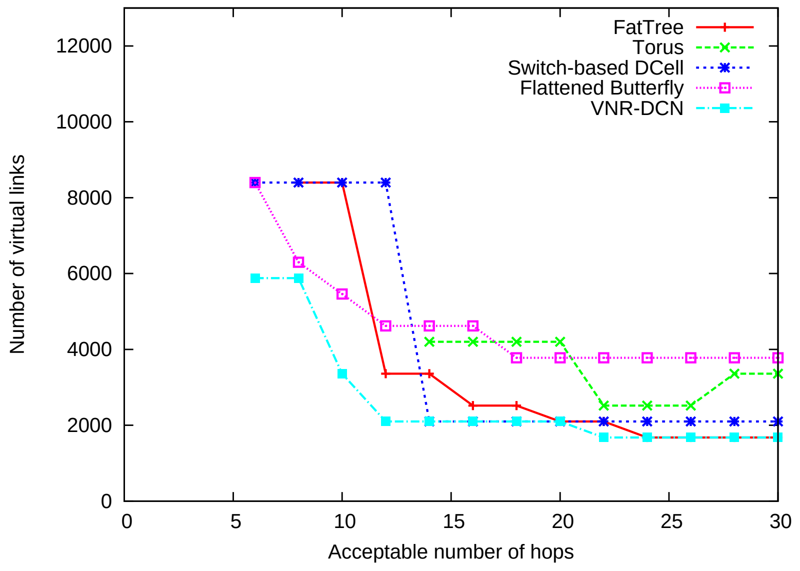


Figure 3.11: Number of virtual links required to ensure that the maximum number of hops is no more than the target value

with the network topology using only ToR switches connected to communicating servers. Thus, in our evaluation, we use GFB as the base topology whose parameters are set so as to accommodate the maximum amount of traffic generated in our evaluation, similarly to the static electronic network.

In the evaluation, we power down the ports of the ToR switches when the following two constraints are satisfied: (1) routes exist between all ToR switches, and (2) the capacity of each link is larger than the traffic passing through the link. When determining the electronic ports to be powered down, we set the routes of the traffic by SP or SP-VLB.

3.4.3.2 Energy Consumption Model

In our evaluation, we models the energy consumption based on the catalogs of the following devices. We use the 48-port Arista 7148 switches [67] as the ToR switches. We need to connect some ports of ToR switches to OCSs via optical transceivers. We use Delta 10 GBASE SR transceivers [72]. In our architecture, we need an OCS with 420 ports, and a OCS with such a large number of

Table 3.2: Energy consumption of data center network elements in this evaluation

Element	Power consumption (W)
ToR (48 ports)	600
OCS (420 ports)	100.8
Transceiver	1

ports has been proposed and implemented [73]. However, we do not have the data of the energy consumption of the OCS with a large number of ports. Thus, we estimate the energy consumption of the OCS by assuming that the energy consumption of each port of the OCS equals to that of a 192-port Glimmerglass OCS [66]. This assumption may overestimate the energy consumption of OCS, because most of the energy of the OCS is consumed by the management function that handles the remote commands from the controller and its energy consumption is independent from the number of ports. Thus, the actual energy consumption of VNR-DCN may be smaller than the following results.

Table 3.2 summarize the energy consumed by the devices used in the network. In the VNR-DCN, only the ports of ToR switches, optical transceivers, and OCSs that are required to construct the current virtual network are powered on. The energy of the ToR switch is consumed by the two kinds of components; one is the component that cannot be shut down even if all ports are shut down, and the other is the component that can be shut down if the corresponding port is not used. However, the ratio of the energy saved by shutting down the ports is not described in the spec sheet of the switches, and depends on the switch architectures [74]. Therefore, we introduce a parameter α indicating the ratio of the energy consumed by the ports to the total energy consumption of the switch. The results by Reviriego et al. [74] indicates that α of the commercial energy efficient Ethernet switches is from 0.5 to 0.7. Thus, in this evaluation, we use two kinds of α, 0.5 and 0.7.

By using this model, the energy consumption of the virtual network P_{logical} constructed over an OCS network is obtained by

$$P_{\mathrm{logical}} = 100.8S_{\mathrm{active}} + (12.5\alpha + 1)L + 252000(1 - \alpha), \tag{3.26}$$

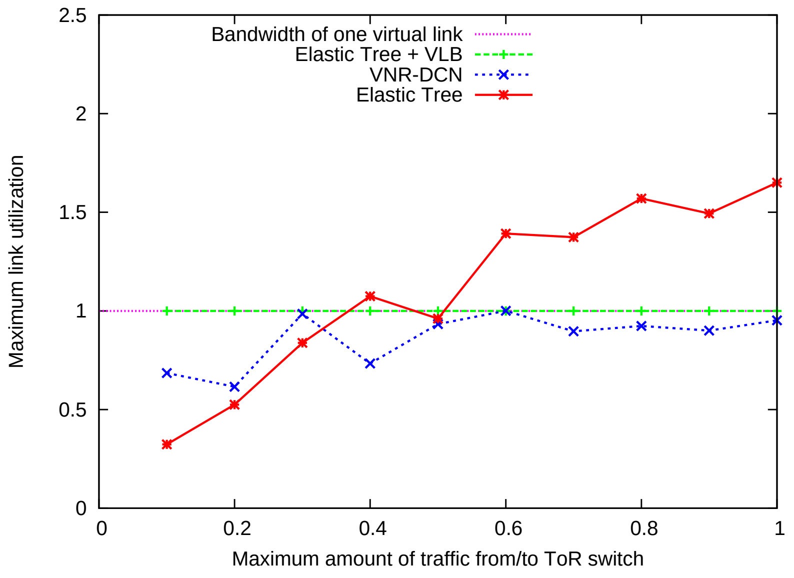


Figure 3.12: Maximum link utilization for different amounts of traffic

where S_{active} is the number of active OCSs, and L is the number of virtual links, respectively. In contrast, when we construct the topology without OCSs, the energy consumption P_{physical} is modeled by

$$P_{\text{physical}} = 12.5\alpha L + 252000(1 - \alpha), \tag{3.27}$$

where L is the number of active links. The energy consumed by the servers or by the ports connected to the servers is ignored in our evaluation because it is the same in all cases.

3.4.3.3 Results

Maximum link utilization for different amounts of traffic Before comparing the energy consumption of virtual networks constructed by each method, we check whether the constructed network can accommodate the required amount of traffic. In a data center, traffic patterns change within a few seconds [71, 70], and the virtual network should accommodate such frequently changing traffic as discussed in Subsection 3.4.1. Thus, we investigate the maximum link utilization when the traffic pattern changes from the initial traffic pattern, while the virtual network is constructed by using the initial traffic pattern. The initial and changed traffic patterns are generated in the same manner as in Subsection 3.4.1, namely by generating 10 traffic patterns after the change and showing the maximum link utilization in all patterns.

The results are shown in Figure 3.12, where the horizontal axis denotes the maximum traffic amount from or to ToR switches that must be accommodated, and the vertical axis shows the maximum link utilization when the traffic demand is changed. In the figure, the labels "VNR-DCN", "Elastic Tree", and "Elastic Tree + VLB" denote the results for the cases where the topology is constructed by using VNR-DCN, the elastic tree method (see paragraph 3.4.3.1), and the elastic tree method with considering VLB, respectively. "Electronic" denotes the results for the static electronic network.

In the case of using an elastic tree, the maximum link utilization is larger than the bandwidth of one virtual link when the maximum amount of traffic is larger than 0.4. The elastic tree method uses a small number of links if they are sufficient to accommodate the initial traffic. However, the link utilization increases after changing the traffic demand. In contrast, the maximum link utilization of the topology constructed using an elastic tree by considering VLB is smaller than the target value. This is because VLB avoids the concentration of traffic at any particular ToR switch pair by balancing the traffic across all ToR switch pairs. Then, the virtual network constructed using an elastic tree and VLB accommodates the traffic balanced by VLB. As a result, the virtual network accommodates all traffic without excessive link utilization, even when traffic demand changes.

Evaluation of the energy consumption We investigate the energy consumption of the topologies. The results are shown in Figures 3.4.3.3 and 3.4.3.3, where the horizontal axis denotes the maximum amount of traffic from or to ToR switches that must be accommodated, and the vertical axis denotes the saved energy consumption compared with the static electronic network.

In this evaluation, we exclude the result for elastic tree because in that case the network cannot accommodate the required amount of traffic, as discussed in the previous paragraph. When the maximum amount of traffic from or to ToR switches is large, the energy consumption of the network constructed by VNR-DCN is larger than that in the case of the static network based on electronic switches alone, and the saved energy consumption becomes lower than zero. This is caused by the additional devices required by VNR-DCN, such as the optical transceiver and the OCS.

Figure 3.13 also shows that in the case of using the elastic tree combined with VLB, the energy consumption can be reduced in comparison to the static electronic network, even when the maximum amount of traffic is generated. This is caused by the fact that all switches in GFB have the same number of links in order to facilitate the calculation of the number of flows passing through each link. However, even though the elastic tree allows switches to have different number of links, the reduction in energy consumption brought by the elastic tree is marginal. When the maximum amount of traffic from or to ToR switches is small, the energy consumption of the network constructed by VNR-DCN is the smallest. This is because VNR-DCN reconfigures the virtual network so as to reduce the number of links based on the instantaneous traffic load. Although the elastic tree also powers down ports to save energy, it cannot reconfigure the network to a topology vastly different from the initial topology because it has to maintain the connectivity between ToR switches, and many links cannot be powered down.

Comparing Figure 3.4.3.3 with Figure 3.4.3.3, the saved energy becomes large in the case of large α, because shutting down each port reduces energy consumption more. However, even in the case of $\alpha = 0.5$, VNR-DCN saves more energy than the elastic tree when the traffic amount becomes small.

Evaluation of energy consumption over 24 h As can be seen in Figure 3.13, the energy consumption of the network constructed by VNR-DCN is lower than others when the maximum amount of traffic from or to ToR switches is small. To clarify the impact of the reduction in energy consumption, we compare the total energy consumed over 24 h. Heller et al. [11] found a clear pattern in the total traffic rate at switches in a data center, where traffic was found to peak during the day and drop at night. In this evaluation, we use the following simple model of variation in traffic over 24 h:

$$T_{\max}(x) = \frac{V_{\text{peak}} - V_{\text{low}}}{2} \sin x + \frac{V_{\text{peak}} + V_{\text{low}}}{2} (0 \leq x \leq 2\pi), \tag{3.28}$$

where $T_{\max}(x)$ is the traffic rate from or to each ToR switch at time x, V_{peak} is the peak traffic rate, and V_{low} is the lowest traffic rate. We investigate the energy consumption in various scenarios by changing V_{peak} and V_{low}. In this evaluation, the elastic tree method and VNR-DCN are used to configure the virtual network 24 times over 24 h.

Figure 3.14 shows the results, where vertical axis denotes the peak rate and horizontal axis denotes the lowest rate. In this figure, each area is colored based on the value of $P^{\text{VNR-DCN}}_{\text{logical}} - P^{\text{elastic}}_{\text{physical}}$ where $P^{\text{VNR-DCN}}_{\text{logical}}$ is the energy consumption when we use the VNR-DCN, and $P^{\text{elastic}}_{\text{physical}}$ is the energy consumption when we use the elastic tree combined with the VLB. The area where VNR-DCN saves more energy than the elastic tree is colored in blue, while the area the elastic tree saves more energy is colored in red. Clearly, VNR-DCN is effective when traffic changes drastically even in the case of $\alpha = 0.5$. This is because VNR-DCN changes the virtual network topology so as to reduce the energy consumption, while the elastic tree cannot change the network topology. In this regard, Kandula et al. showed that traffic rate changes drastically in an actual commercial data center [70]. VNR-DCN is expected to be effective for reducing the energy consumption in such data centers.

3.5 Discussion on the Scalability

In this section, we discuss the scalability of the VNR-DCN, and explain how the VNR-DCN works in a large data center.

3.5.1 Scalability of the controllers

3.5.1.1 NC

In our method, we introduce the centralized controller NC, which collects the traffic information, and sets the parameters of the GFB. As the size of the network becomes large, the number of ToR switches whose traffic information is required to be collected by the NC increases. However, the amount of the collected traffic information is not large, because the VNR-DCN requires only the total traffic amount from/to each ToR switch and does not require the detailed traffic information.

The calculation time of the GFB parameters may increase as the size of the network becomes large. However, the calculation time of the GFB parameters is only $O(N)\log 2H_{\max}$, where N is the number of ToR switches and $H_{\max}$ is the acceptable number of hops set by the administrator, as discussed in Section 3.3.2.3. Therefore, the calculation time does not become large, even in a large network.

In addition, because the short-term traffic changes are handled by the load balancing, the interval of calculation of the parameters of the GFB is not necessarily short. Therefore, the NC can work even in a large data center.

3.5.1.2 RC

We also introduce another kind of the controller, RC. The only task of the RC is to calculate the routing table from the GFB parameters. The calculation time for the routing table in GFB routing is $O(\sum_{1<=i<=K_{\max}} N_i)$, which is significantly shorter than $O(N)$, as discussed in Section 3.2.3.2.

3.5.2 Scalability of the physical network structure

In our architecture, we need the network of the OCSs that can connect any ToR switch pairs. Though we use the OCSs with the same number of ports as the number of ToR switches in the evaluation in Section 3.4, we can construct the network by using OCSs with a small number of ports. For example, by constructing the Clos network using the OCSs, we can construct the network that can connect any ToR switch pairs.

3.5.3 Scalability of the constructed virtual network

Finally, we discuss the virtual network by the VNR-DCN when the number of ToR switches or the amount of the traffic becomes large. Figure 3.15 shows the number of used ports per ToR switch. In this figure, the horizontal axis is the maximum amount of traffic from a ToR switch, and the vertical axis is the number of used ports per ToR switch. We plot three lines; the red line indicates the case with 420 ToR switches, the green line indicates the case with 900 ToR switches, and the blue line indicates the case with 1200 ToR switches.

As shown in this figure, the number of required ports per ToR switch is almost similar even when the number of servers is large. That is, our method is applicable to a large data center.

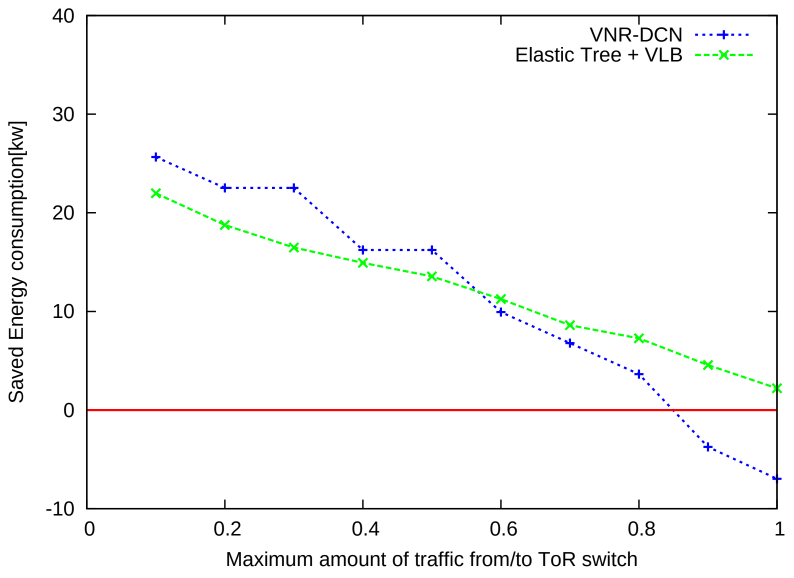
This figure also indicates that the number of used ports per ToR increases linearly as the traffic amount increases. This increase of the number of used ports is not large, because at least a proportional number of virtual links to the amount of traffic is required to accommodate the generated traffic without congestion even if any kind of the network topology is used. Therefore, the VNR-DCN constructs the virtual network with a small number of ports even when the traffic amount becomes large.

3.6 Conclusion

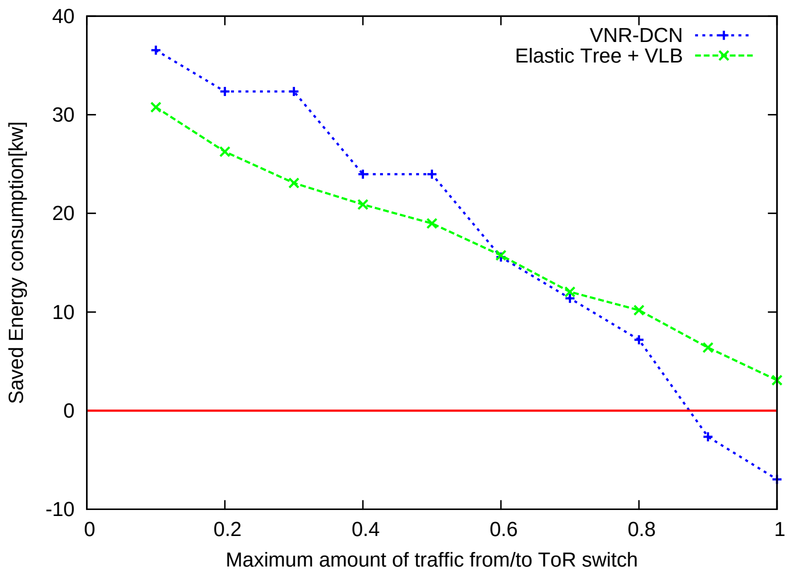
In this chapter, we introduced the concept of a virtual network configured over a data center network consisting of both OCSs and electronic switches. We proposed a method for constructing a virtual network by setting the parameters of its topology as well as a method for reconfiguring the virtual network within a short period of time by adjusting these parameters. In addition, we discussed implementation issues related to VNR-DCN and demonstrated that it can be applied to networks consisting of existing devices.

Through evaluation, we clarified that VNR-DCN constructs a topology satisfying the requirements on bandwidth and delay in the case of using a routing method based on GFB combined with VLB. We demonstrated that VNR-DCN is effective for reducing the energy consumption of the network by comparing it with the method based on powering down the ports of the ToR switches of a static electronic network.

One of our future research topics is to construct the distributed algorithm to further reduce the time required to respond to traffic changes.

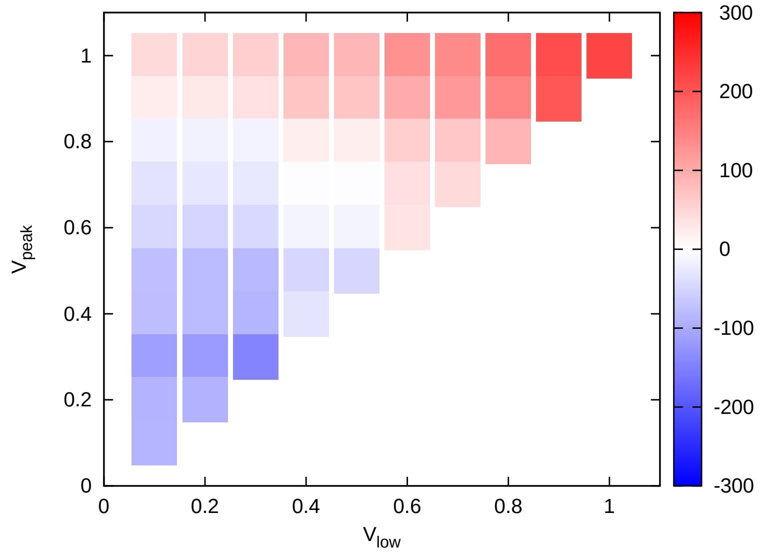


(a) $\alpha = 0.5$

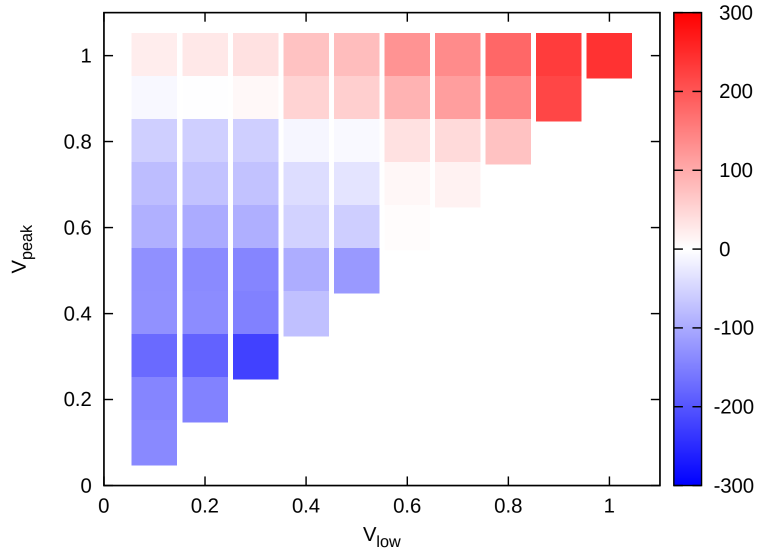


(b) $\alpha = 0.7$

Figure 3.13: Energy consumption required to accommodate the traffic from/to ToR switches



(a) $\alpha = 0.5$



(b) $\alpha = 0.7$

Figure 3.14: Energy consumption required to accommodate the traffic from/to ToR switches

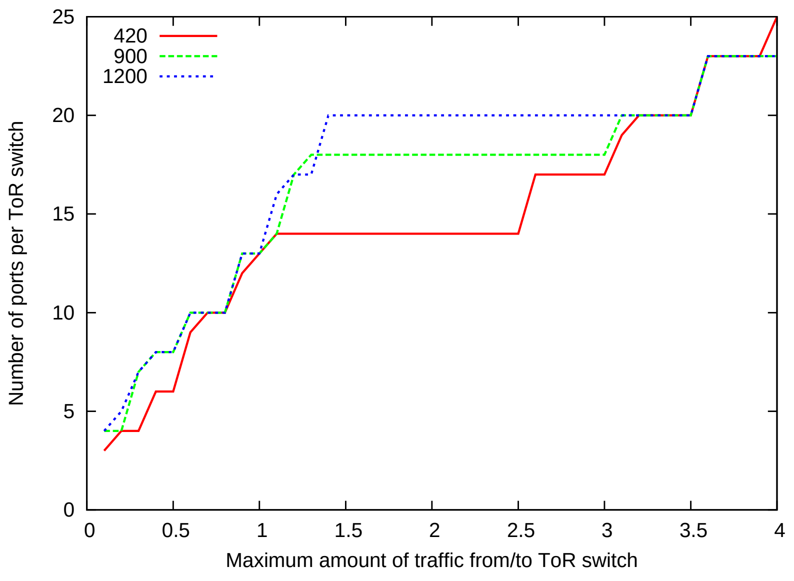


Figure 3.15: The number of ports per ToR switch in the case of increasing number of servers and traffic

Chapter 4

Virtual Machine Placement Considering the Redundancy of Data in Multi tenant Data Centers

4.1 Service Accommodation in Multi tenant Data Centers

4.1.1 Scenario

In this chapter, we focus on a single data center. The data center is managed by a *data center provider*. There are several *service providers*. Each service provider provides one service. Service providers make requests for data center resources to the data center provider and provide their services by using the allocated resources.

4.1.1.1 Service Provider

When the service provider starts a new service, the service provider prepares the VM images for the service. The VM images include the mechanism for cooperating with each other so as to keep working without losing any required data even when some VMs become unavailable. One mechanism for this cooperation for robustness against failures is the replication mechanism used by the

GFS. In this mechanism, each fraction of data is replicated to k VMs. By replicating the fraction of data, no fraction of data is ever lost unless more than k VMs become simultaneously unavailable.

After the VM images have been prepared, the service provider requests the accommodation of the VMs by the data center provider. The requests includes not only the VM images but also the parameters, such as the number of chunk replicas k. Finally, after VMs have been accommodated by the data center provider, the service starts.

4.1.1.2 Data Center Provider

When a request is received from a service provider When a request is received from a service provider, the data center provider decides where to accommodate the required VMs. The placement of the VMs is decided under the constraint that no data is lost even in the case of any single point of failure by considering the number of chunk replicas k.

In the case of a failure Because the VMs are placed by considering the number of chunk replicas k, all data can be obtained from available VMs unless more than k VMs fail. After failure occurs, the robustness against next failure can also be achieved by reallocating the VMs with the similar way to the case that a request from a service provider arrives.

4.1.2 Problem Statement

In the above scenario, how to place the VMs considering the number of chunk replicas k is important. In this subsection, we formulate such a problem to place the VMs.

4.1.2.1 Data Center Network

The data center provider manages a data center. The data center includes many servers. Typically, the servers are installed in server racks. Servers in the same server rack are connected to a network switch called the *Top-of-Rack (ToR)* switch, which is also installed in the server rack. The ToR switches are connected to a large switch called the *core switch*. Finally, the data center network, which connects all of the servers, is constructed by connecting the core switches together.

The data center network is represented as a graph G where the nodes indicate devices such as servers and switches, and edges indicate the links between devices. We denote the set of nodes by N and the set of edges by L. A path on the graph G can be defined as a list of nodes. There may be multiple paths between any two nodes. We denote the set of paths between the nodes s and d by $r_{s,d}$, and the set of all paths on the graph G by R. There are two types of nodes, servers and switches. We denote the set of servers by S, where S is a subset of N.

Server nodes have a property indicating the amount of resources such as CPU and memory provided by the server. We denote the amount of resources provided by a server s by U_s. In general, because servers provide multiple types of resources such as CPU, memory, and storage, U_s is a vector. However, to simplify the problem in this paper, we consider the number of CPU cores only, and take U_s to indicate the number of VMs that the server s can accommodate.

4.1.2.2 Request from a Service Provider

Each service is provided by multiple functions cooperating with each other. For example, a web service consists of the web front end, which provides the user interface, and the database that handles the large amount of data required by the service. The relations between the functions in the services of the service provider p are represented as a graph G_p^v, as shown in Figure 4.1. In this graph, each node indicates a function, and each edge indicates a relation between functions. We denote the set of functions included in the request from the service provider p by F_p.

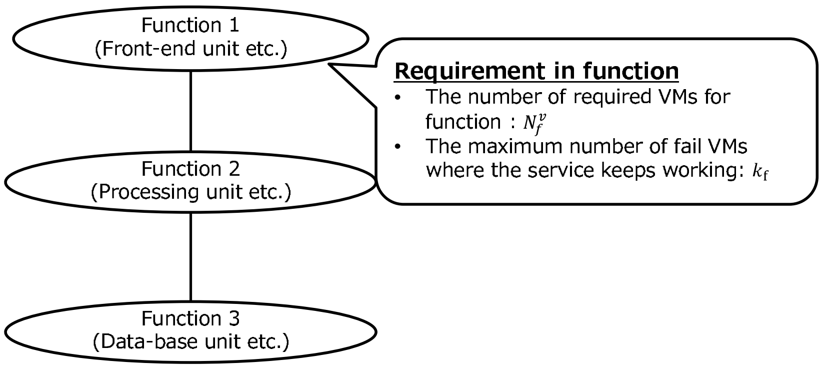

Function 1
(Front-end unit etc.)
Function 2
(Processing unit etc.)
Function 3
(Data-base unit etc.)
Requirement in function
• The number of required VMs for function : N_f^v
• The maximum number of fail VMs where the service keeps working: k_f

Figure 4.1: Request from tenant to providers

Each function is performed by the multiple VMs cooperating with each other. We denote the set of required VMs for the function f by N^v_f. As described above, the VMs employ a mechanism to replicate each fraction of data so as to keep all data available even when some VMs become unavailable. We denote the number of replicas of each fraction of data in the function f by k_f. The above parameters related to each service can be attached as properties of the nodes in the graph G^v_p. The service provider p sends the information for the graph G^v_p to the data center provider with the request for accommodation of the services.

Based on the requests from the service providers, the data center provider accommodates all the VMs included in the requests. We denote the set of service providers by P, and the set of the VMs required to be hosted by N^v. N^v is the union of N^v_f for all functions f of all requests. The data center provider also accommodates the traffic between the VMs by providing connections between the VMs. We call the connections between the VMs the *virtual link*. We denote the virtual link between VMs s and d by $l_{s,d}$. The virtual links are established between the VMs of the same function and between the VMs of related functions. We denote the set of virtual links by L^v.

4.1.2.3 Formulation of the VM Assignment Problem

We aim to allocate the VMs under the constraint that no data is lost even in the case of any single point of failure. This problem can be formulated as the problem of finding the mapping $M_N : N^v \rightarrow S$ and $M_L : L^v \rightarrow R$ where M_N indicates the mapping from the requested VM to the servers, and M_L indicates the mapping from the links included in the service requests to the physical links.

Objective Defining $F(M_N, M_L)$ as the cost required by the placement of the VMs, the objective is

$$\text{minimize } F(M_N, M_L). \tag{4.1}$$

We can use any cost function $F(M_N, M_L)$ based on the objective of the data center provider. In Section 4.3, we use the cost function indicating the energy consumption.

Constraints M_N and M_L must satisfy the following constraints.

All VMs must be hosted by the servers.

$$\forall n^v \in N^v\colon M_N(n^v) \in S.$$

All virtual links must be accommodated on the path between the servers where the corresponding VMs are hosted.

$$\begin{aligned}&\forall n_1^v \in N^v, \forall n_2^v \in N^v\colon\\ &\qquad M_L(l^v_{n_1^v,n_2^v}) \in r_{M_N(n_1^v),M_N(n_2^v)} \textit{ iff } (\forall l^v_{n_1^v,n_2^v} \in L^v).\end{aligned}$$

All the servers must have the sufficient resources to host all of the allocated VMs, including standby VMs. That is,

$$\forall s \in S\colon N_s^{\mathrm{act}}(M_N) \le U_s,$$

where $N_s^{\mathrm{act}}(M_N)$ is the number of VMs hosted by the server s. $N_s^{\mathrm{act}}(M_N)$ is given by

$$N_s^{act}(M_N) = |\{n^v | n^v \in N^v, M_N(n^v) = s\}|.$$

All data must be available even if any single point of failure occurs. We let $G - \{n\}$ indicate the physical network where the node n fails, and $Maxsub(G - \{n\})$ indicate the largest directed acyclic subgraph. To ensure that the service keep working, more than $|N_f^v| - k_f$ VMs should be included in $Maxsub(G - \{n\})$. That is,

$$\forall p \in P, \forall f \in F_p, \forall n \in N\colon N^{avail}_{Maxsub(G-\{n\}),f} \ge |N_f^v| - k_f,$$

where $N^{\mathrm{avail}}_{f,n}$ is the number of VMs hosted by the nodes in $Maxsub$(G-{n}), and given by

$$\begin{aligned}&N^{avail}_{Maxsub(G-\{n\}),f} =\\ &\qquad |\{n^v | n^v \in N_f^v, m \in M_N(n^v), m \in Maxsub(G - \{n\})\}|.\end{aligned}$$

4.2 Heuristic Method to Place the VMs

The optimization problem formulated in Section 4.1.2.3 is difficult to solve when requests from the service provider arrive, because of the following reasons. (1) The optimization problem assumes that all requests are given, but the data center provider should place the VMs as each request arrives and does not have knowledge of future requests. (2) The VMs should be placed soon after each request arrives or reallocation of VMs becomes required, but solving the optimization problem takes a long time.

Therefore, we propose a heuristic method for deciding which servers will host the VMs. In this method, we select the placement of each VM one-by-one to avoid large calculation times. Our method works as follows. Our method finds the critical flows for each node in the data center network in advance. Each time a request is received from the service provider, we then decide the servers to host the VMs by using the information about the critical flows. When deciding the servers to host the VMs, we place the VMs one by one.

The rest of this section explains the details of the above steps.

4.2.1 Method to Find Critical Flows

We first find the critical flows for each node. From the definition, the flow from s to d is a critical flow for the nodes s and d. In addition, assuming that all links in the network G are bidirectional, the following lemma about the critical flow is satisfied.

Lemma *Let $p^{s,d}_i$ denote the i-th node on the shortest path from s to d on the network G. Let the flow from s to d be a critical flow for $p^{s,d}_{i_1}$. Assume that there are multiple disjoint paths from $p^{s,d}_{i_1}$ to $p^{s,d}_{i_2}$ ($i_1 < i_2 < i^{s,d}_{max}$) where $i^{s,d}_{max}$ is the number of nodes on the path from s to d. The flow from s to d is a critical flow for the node $p^{s,d}_{i_2}$ if and only if multiple disjoint paths from $p^{s,d}_{i_1}$ to $p^{s,d}_{i_2+1}$ do not exist.*

Proof *If there is only one path from $p^{s,d}_{i_1}$ to $p^{s,d}_{i_2+1}$, then there exists a node x whose failure disconnects the network into two connected subgraphs: G_1, which contains $p^{s,d}_{i_1}$, and G_2, which contains $p^{s,d}_{i_2+1}$. Because we cannot achieve $p^{s,d}_{i_2+1}$ from $p^{s,d}_{i_1}$ without x, the node x is on the shortest path from*

$p^{s,d}_{i_1}$ to $p^{s,d}_{i_2+1}$. Because the path from $p^{s,d}_{i_1}$ to $p^{s,d}_{i_2+1}$ is a subset of the shortest path from s to d, the node x is on the shortest path from s to d. Because the shortest path does not include the same node multiple times, we have a path from s to $p^{s,d}_{i_1}$ and a path from $p^{s,d}_{i_2+1}$ to d without x. Thus, G_1 contains s and G_2 contains d. That is, there are no paths from s to d without x, and the flow from s to d is a critical flow for the node x. Assuming that there are multiple disjoint path from $p^{s,d}_{i_1}$ to $p^{s,d}_{i_2}$, we can achieve $p^{s,d}_{i_2+1}$ from $p^{s,d}_{i_1}$ even if the node $p^{s,d}_{i_3}$ $(i_1 < i_3 < i_2)$ is removed. Therefore, x must be $p^{s,d}_{i_2}$, and the flow from s to d is a critical flow for $p^{s,d}_{i_2}$.

If there are multiple disjoint paths from $p^{s,d}_{i_1}$ to $p^{s,d}_{i_2+1}$, there exists a path from s to d that does not pass the node $p^{s,d}_{i_2}$. Thus, the flow from s to d is a critical flow for the node $p^{s,d}_{i_2}$ only if multiple disjoint paths from $p^{s,d}_{i_1}$ to $p^{s,d}_{i_2+1}$ do not exist.

By using the above lemma, we find the critical flows for each node by using the number of disjoint paths between the nodes in the network. The number of disjoint paths from the node s to the node d is obtained by the following steps.

1. Set the counter value to 0.
2. Calculate the route from the node s to the node d on the graph G by using the Dijkstra algorithm.
3. If a route is found, increment the counter, remove the intermediate nodes of the routes from the graph G, and go back to step 2. Otherwise, we set the counter value to the number of disjoint paths.

We represent the numbers of the disjoint paths by the matrix D, whose elements $d_{s,d}$ indicate the number of disjoint paths from the node s to the node d.

We then find the critical flow for each node. We denote the i-th node on the shortest path from s to d by $p^{s,d}_i$, and the number of nodes on the shortest path by $i^{s,d}_{max}$. The node s and d is also regarded as a node on the shortest path from s to d. That is, $p^{s,d}_0$ is s, and $p^{s,d}_{i^{s,d}_{max}}$ is d. We let C_n be the set of critical flows for the node n. C_n is obtained by performing the following steps for all of the server pairs s and d. In the following steps, i is a counter, and p^{prev} is the previously found node where the flow from s to d is a critical flow.

1. Add the flow from s to d to C_s and C_d.
2. Initialize i and p^{prev} as $i \leftarrow 1$, and $p^{\mathrm{prev}} \leftarrow s$.
3. If $d_{p^{\mathrm{prev}},p_i^{s,d}}$ is 1, add the flow from s to d to $C_{p_{i-1}^{s,d}}$, and set $p^{\mathrm{prev}} \leftarrow p_{i-1}^{s,d}$
4. If i is less than $i_{max}^{s,d}$, increment i and go back to Step 3. Otherwise, end.

The calculation time to obtain C_n for all nodes is O(hn^2) where h is the average number of hops and n is the number of nodes in the network. Note that the critical flows do not change unless the network topology changes. Therefore, we need to calculate the critical flows only once.

4.2.2 Method to Place the VMs

When a request from the service provider arrives, the data center provider decides the servers hosting the VMs related to the service. In this subsection, we explain how our heuristic method decides the servers to host VMs when a request is received from a service provider p.

The request from the service provider p includes the set of functions F_p, and each function f in F_p requires the set of VMs N_f^v. We select the servers to host these VMs one-by one by performing the following steps for each VM n in N_f^v for each function f in F_p.

1. Select one of the servers with sufficient resources to accommodate the VM n based on the objective function.
2. Check whether the selected server satisfies the requirement that no data is lost in case of any single point failure. If yes, designate the selected server as the server to host the VM n. Otherwise, go back to Step 1 to select another server.

Finally, when all VMs required by the service provider p are assigned to the servers, we place all the VMs.

In the above steps, the method to check whether the selected server satisfies the requirement is important. The rest of this paragraph explains this method.

To ensure all data required by the function f remains available, k_f VMs must be active in the case of any single point of failure. We thus regard the server as the server which does not satisfy the requirement if a single point failure could disconnect more than k_f VMs when the VM n is placed on the server.

The number of VMs that are disconnected from the server s by the failure of the node n, $n^{\text{fail}}_{s,n}$ is given by

$$n^{\text{fail}}_{s,n} = \sum_{v \in N^{v-assign}} a_{s,v,n},$$

where $N^{v\text{-}assign}$ is the set of VMs whose corresponding servers are assigned (i.e., $N^{v\text{-}assign} = N^v \cap \{v | M_N(v) \in N\}$), and $a_{s,v,n}$ is a variable indicating whether server s can access the VM v when node n fails as defined by

$$a_{s,v,n} = \begin{cases} 1 & (s, M_N(v)) \in C_n \\ 0 & \text{otherwise} \end{cases},$$

where C_n is the set of the critical flows for n, and $(s, M_N(v))$ is the flow from s to $M_N(v)$

The number of VMs that cannot be accessed from the server s is at most $\max_{n \in N} n^{\text{fail}}_{s,n}$. Therefore, if there exists a server s hosting the related VM whose $\max_{n \in N} n^{\text{fail}}_{s,n}$ is smaller than k_f, the required data is never lost in any case of a single point of failure. Otherwise, it is possible for data to be lost because of a failure.

Therefore, for the server selected at Step 1, our method calculates $\min_{s \in N^{v\text{-}assign}} \max_{n \in N} n^{\text{fail}}_{s,n}$ for the case that the server is selected as the server to host the VM. If $\min_{s \in N^{v\text{-}assign}} \max_{n \in N} n^{\text{fail}}_{s,n}$ is less than k_f, the server is regarded as the server satisfying the requirement.

4.2.2.1 Calculation Time

In our method, we check the number of critical flows for all nodes between the server hosting the VMs and the candidate servers. Therefore, the computational complexity for selecting the servers to host each VM is $O(HN^2_{\text{vm}}N_{\text{cand}})$, where N_{vm} is the number of servers hosting the related VMs, N_{cand} is the number of servers checked before finding the server satisfying the requirement, and H is the average number of hops between VMs. In most cases, the server satisfying the requirement can be found by checking only a few servers, and N_{cand} is small. In addition, N^2_{vm} and H are much smaller than the number of servers.

Table 4.1: Energy consumption model

Device	Power consumption (W)
ToR switch	150
Core switch	390
Server	122

4.3 Evaluation

In this section, we demonstrate that our method allocates the VMs without losing any data in the case of any single point of failure.

4.3.1 Evaluation Environment

4.3.1.1 Objective Function

In this evaluation, we use the objective function to minimize the energy consumption. Only the servers hosting the VM work, and the other servers can sleep. Assuming that the sleeping servers consumes only a negligible amount of energy, the energy consumption of the data center E is

$$E = \sum_{n \in N^{\text{active}}} E_n,$$

where E_n is the energy consumption of the node n, and N^{active} is the set of the nodes that are required to be powered on.

In our evaluation, we set E_n to the values shown in Table 4.1. In this table, the energy consumption of the ToR switch is set based on the energy consumption of the Cisco Nexus 2148T switch[75], the energy consumption of the core switches is set based on the energy consumption of the Cisco Nexus 5048 switch[76], and the energy consumption of the server is set based on the energy consumption of the Dell R320 server[77].

In our method, we consider this objective function by selecting the node that requires the smallest additional energy consumption as the candidate node to host the VM.

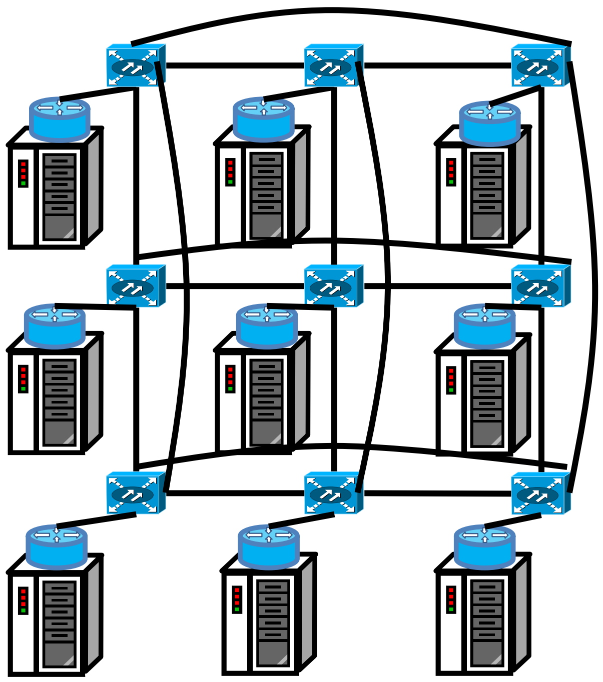


Figure 4.2: Torus topology

4.3.1.2 Network Topology

In this evaluation, we use the torus network as shown in Figure 4.2, because this network topology is often used in data centers and for supercomputers.

Torus is a network topology constructed by locating switches in a multi-dimensional grid as shown in Figure 4.2. The torus is constructed based on the IDs of the switches which are a *k*-dimensional vector. In the *k*-dimensional Torus, switch A is connected to switch B whose ID is next to the ID of switch A in one dimension and equals the ID of switch A in the other dimensions.

In this section, to explain how our method works clearly, we use a 3×3 torus network constructed of core switches. Each core switch is connected to one ToR switch, and each ToR switch is connected to two servers. Each server can host two TMs.

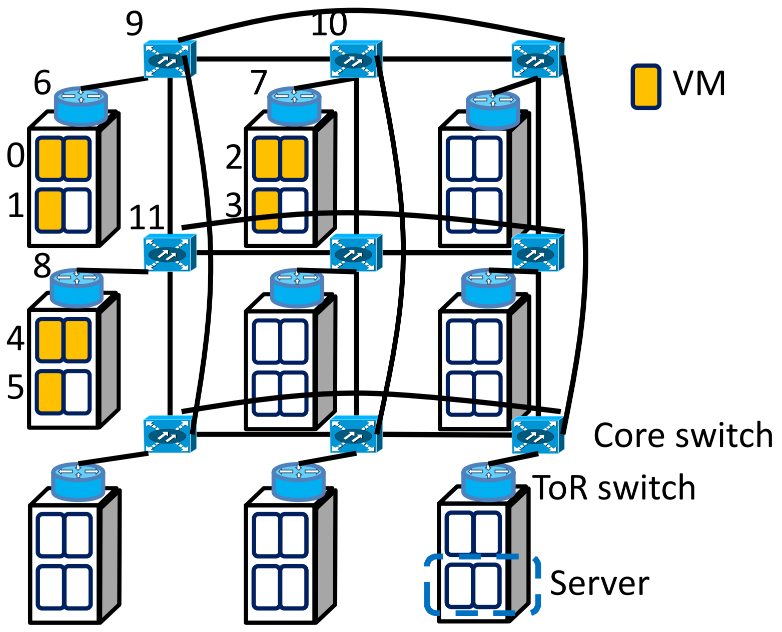


Figure 4.3: How our method places the VMs

4.3.2 Results

In this subsection, we demonstrate how our method places the VMs, and how our method works when a failure occurs.

4.3.2.1 How Our Method Places the VMs

We will now generate one request and show how the requested VMs are placed. In this paragraph, the generated request includes one function requiring 9 VMs, and has k_f of 3.

Figure 4.3 shows how the VMs are accommodated by our method. Here, all VMs are accommodated on near servers so that the server hosting the VM can communicate through at most three core switches. As a result, only a small number of switches are powered on. In each server rack, only three VMs are accommodated. This is because some data is lost when a ToR switch fails if more than three VMs are accommodated in each server rack.

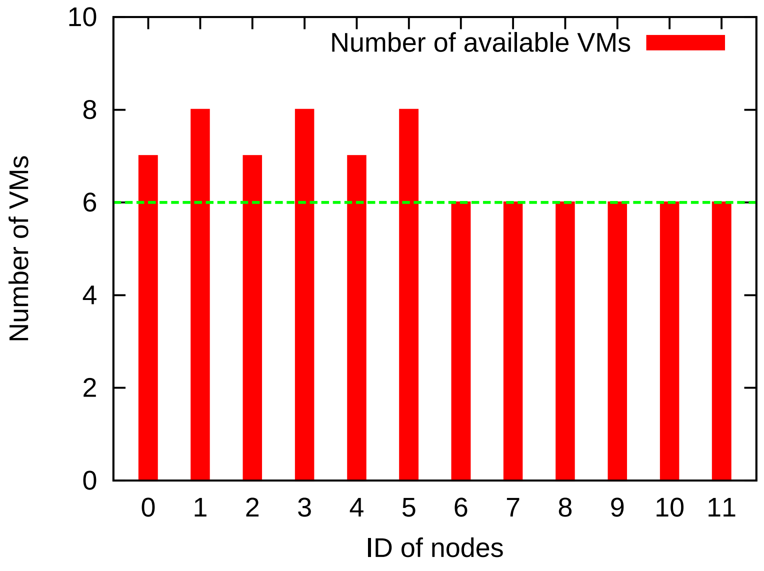


Figure 4.4: The number of available VMs when a failure occurs

4.3.2.2 How Our Method Works When a Failure Occurs

In our method, we keep all data available in the case of any single point of failure by keeping more than $|N_f^v| - k_f$ VMs active. We will now demonstrate this by generating a failure after the service request above has been accommodated.

Figure 4.4 shows the number of available VMs in the case of a failure. The horizontal axis indicates the ID of the failed node, and the vertical axis indicates the number of available VMs. As shown in this figure, the number of unavailable VMs is at most 3. That is, no data is lost, because k_f is set to 3.

4.3.2.3 Energy Consumption Achieved by our Method

Our method accommodates VMs on each service when each request arrives. We will now compare the energy consumption achieved by our method after two services are accommodated one-by-one with that achieved by optimum placement that knows everything about the two requests. In this paragraph, the first request includes one function requiring 9 VMs, and has k_f of 3. The second request includes one function requiring 11 VMs and has k_f of 4.

Table 4.2: Results of our method and the optimal placement

	our method	optimal
Energy consumption	3920	3920
Number of active servers	10	10

The optimum placement is obtained by comparing all possible placement patterns. For each of the possible placement patterns, we check whether the placement satisfies the constraint that more than $|N^v_f| - k_f$ VMs are available even in the case of any single point failure. We then select the placement pattern whose energy consumption is the smallest among the placement patterns that satisfy the above constraint.

Figure 4.3.2.3 and Figure 4.3.2.3 show the VMs as arranged by our method and the optimum placement. As shown in this figure, our method powers on the same number of servers as the optimal placement even though some VMs are hosted by different servers. As a result, our method achieves the optimal energy consumption as shown in Table 4.2.

4.4 Cost of Considering the Number of Replicas

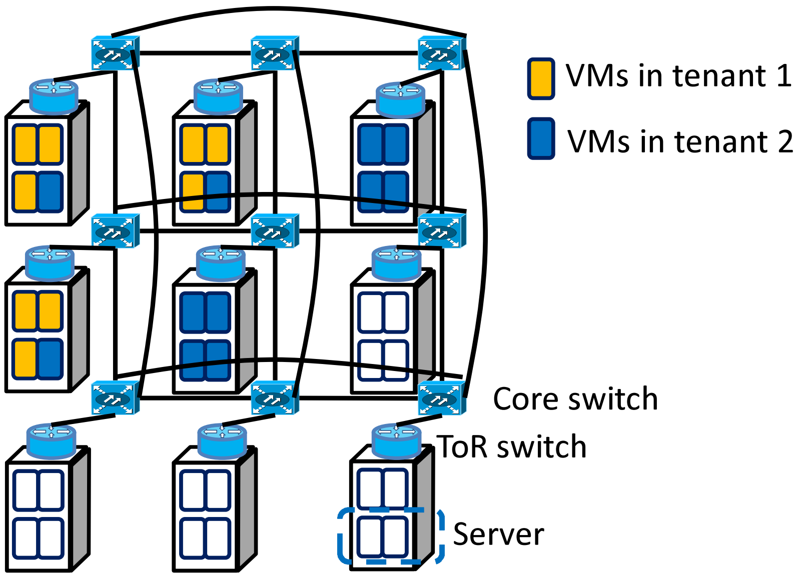
Considering the number of replica k, the VMs related to the same function is placed so that the number of VMs that would be disconnected in case of failures is less than k. As k decreases, more servers are required, because the VM of the same function should be placed at the different places. This may cause a large energy consumption by powering on more servers.

In this section, we discuss this cost of placement of the VMs considering the number of replicas by using the objective function to minimize the energy consumption similar to Section 4.3.

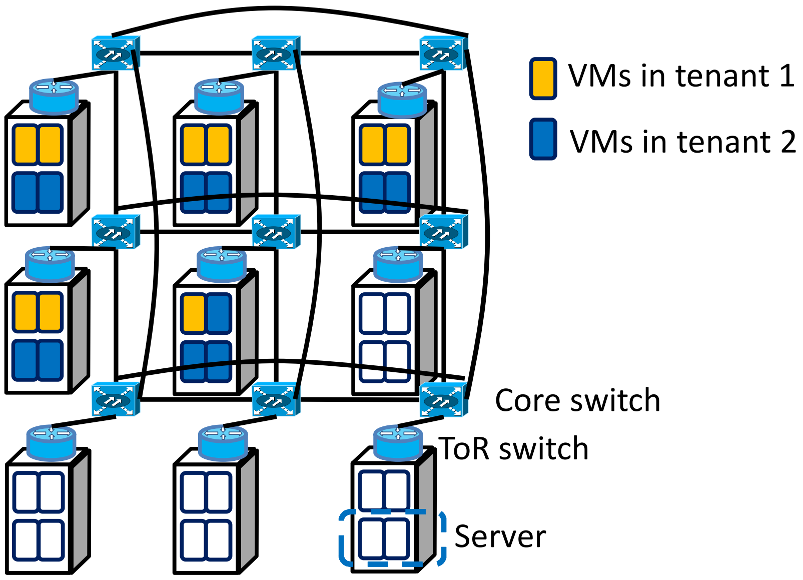
4.4.1 Evaluation Environment

4.4.1.1 Objective Function

We use the same objective function as Section 4.3.



(a) Case of our method



(b) Case of optimizing

Figure 4.5: Placement of VMs

4.4.1.2 Network Topology

In this section, in addition to the torus topology, we also use the FatTree topology, which is also often used in the data centers.

FatTree [27] is a tree topology constructed of multiple roots and pods as shown in Figure 4.6. Each pod is regarded as a switch having a large number of ports consisting of multiple switches having a small number of ports. Pods are constructed using the butterfly topology, where each switch uses half of its ports to connect to switches close to the root switches, and the other half to connect to switches close to the leaf switches. Leaf switches are connected to servers.

In both topologies used in this section, we use 36 core switches, and connect each core switch to 2 ToR switches, and each ToR switch to 4 servers. Each server can host up to 4 VMs.

4.4.1.3 Request from the Service Providers

In this section, we change the number of requests from 1 to 40. The number of functions in each request is set to 2 or 3 randomly, and the number of VMs required by each function is set to a uniformly distributed random value between 10 and 30. We assume that each function is connected by full-mesh links.

k_f is one of the important parameter. To set k_f, we should consider the tradeoff between the energy consumption and the robustness against failure. A large k_f causes inefficient use of resources; storing a large number of chunk replicas requires a large disk space and a large bandwidth between the VMs. On the other hand, setting k_f to a small value may cause a large energy consumption by placing the VMs of the same function at the different places. In this section, we change the number of chunk replica k_f form 0.1 to 0.5 times the number of the VMs so as to check the tradeoff between the energy consumption and robustness against failure.

4.4.2 Compared method

To investigate the cost of considering the number of replicas, we compare our method with the method that accommodates the same number of VMs but minimizes the energy consumption even if the placement cannot keep all data available in the case of a failure. To obtain the results for

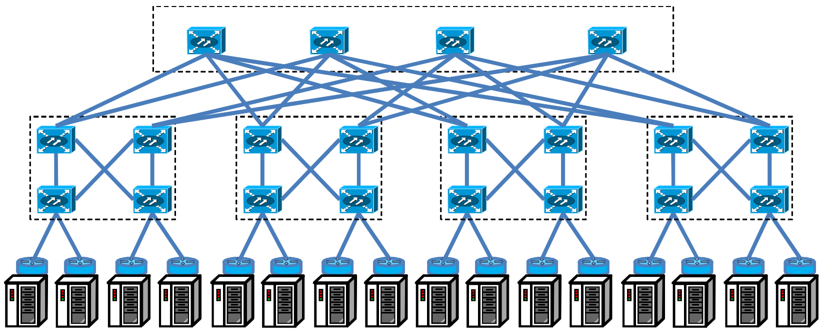


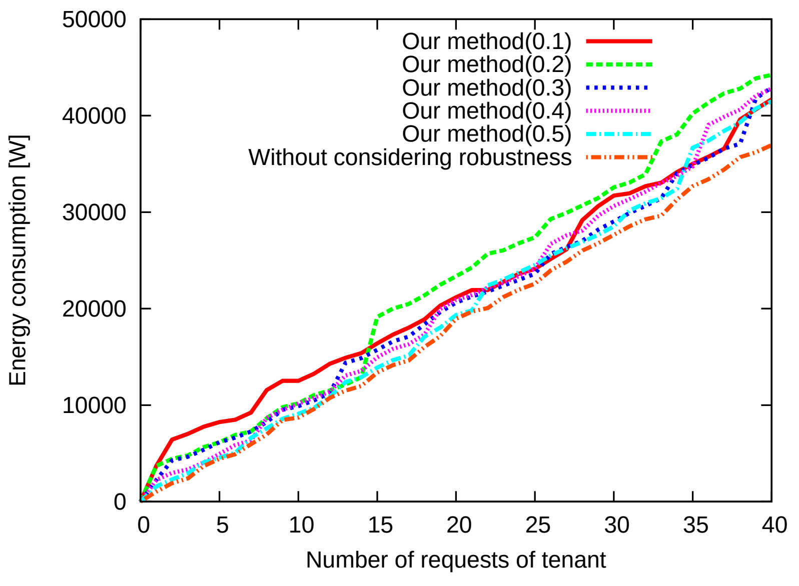
Figure 4.6: FatTree topology

the method that minimizes the energy, we select the server with the smallest additional energy per resource among all the servers when selecting the server hosting each VM.

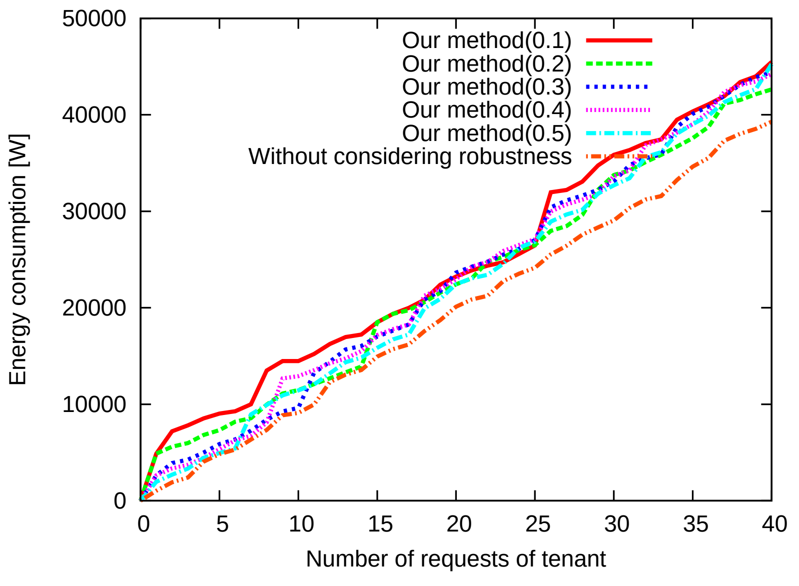
4.4.3 Results

Figures 4.4.3 and 4.4.3 show the energy consumption of the data center. In these figures, the horizontal axes are the number of requests, and the vertical axes are the energy consumption. In addition, to investigate the cost of considering the number of replicas clearly, we also plot the ratio of the energy consumption achieved by our method to that achieved by the method that minimizes the energy consumption without considering the robustness against failures, as shown in Figure 4.4.3 and 4.4.3. In these figures, the horizontal axes are the number of requests, and the vertical axes are the ratio of the energy consumption achieved by our method to the energy consumption achieved by the comparison method.

Figures 4.4.3 and 4.4.3 show that the energy consumption increases as the number of services becomes large. This is because the accommodation of more services requires more servers and switches. In addition, these figures indicate that our method consumes more energy than the method that does not consider the robustness against failures. This is because our method cannot make a server host all VMs related to a certain function since some data may be lost when the server fails, while the method that does not consider the robustness can do this. As a result, our method requires more servers.

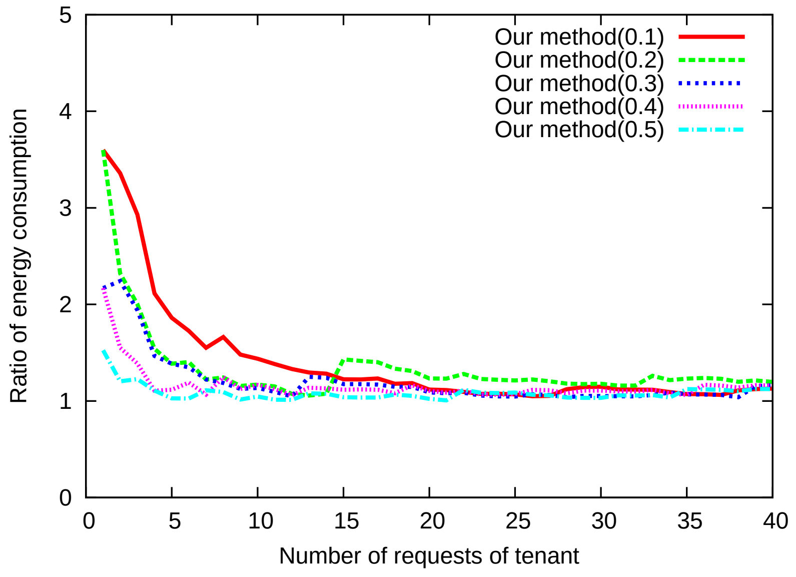


(a) Torus

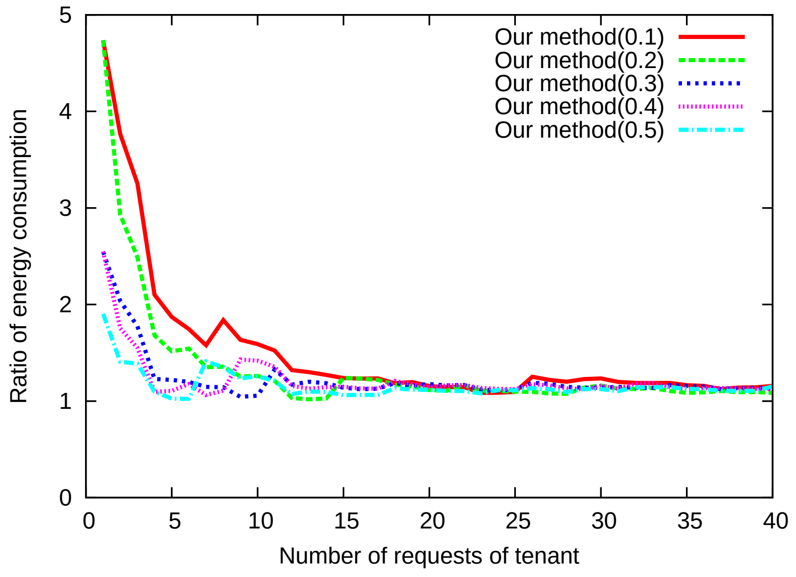


(b) FatTree

Figure 4.7: Energy consumption of a multi tenant data center



(a) Torus



(b) FatTree

Figure 4.8: Ratio of energy consumption

However, Figures 4.4.3 and 4.4.3 show that the ratio of the energy consumption achieved by our method to that achieved by the method that does not consider the robustness becomes small as the number of services becomes large. This indicates that the cost of the placement of the VMs considering the number of replicas is small in multi tenant environments. This is because a server that is powered on to satisfy the requirements of robustness of a service may also be used by other services. As a result, by sharing servers between multiple services, we can satisfy the requirement that no data is lost even in the case of a single point of failure with low energy consumption.

Finally, we investigate the energy consumption for different k_f. If k_f is small, each service is hosted by more servers than the case where k_f is large. Thus, there is the tradeoff between the energy consumption and the robustness against failure. However, as shown in this figure, the difference in the energy consumption caused by the different k_f becomes small as the number of services becomes large. This is because a server powered on to satisfy the requirement of robustness of a service may also be used by other services. The servers are shared by more services as the number of services becomes larger. Therefore, we can keep all data available with low energy consumption even if k_f is set to a small value in a multi tenant data center.

4.5 Conclusion

In this chapter, we introduced a method to place the VMs considering the number of chunk replicas. The placement of the VMs should be obtained in a short time, because after failure, we reallocate the VMs before next failure occurs. Therefore, we proposed a heuristic method. In this method, we calculate the critical flow for each node, which is the flow whose connection becomes unavailable when the node fails. In our method, we place the VMs so that any node failure does not disconnect more than k VMs by using the information of the critical flows. Through numerical simulation, this paper demonstrated that our method keeps all data available even in the case of any single point of failure.

In addition, by using this method, we also investigated the cost of considering the number of chunk replicas. If k is set to a small value, more servers may be required, because the VMs should be placed at the different places to make the number of VMs that would be disconnected less than k. This may cause a large energy consumption by powering on more servers. However, our results demonstrate that the cost of considering the number of chunk replicas becomes small as the number of tenants increases, and even when we set k to a small value, we can achieve the sufficient robustness without causing a large energy consumption.

Chapter 5

Conclusion

In this thesis, we discussed the methods to construct the energy-efficient large-scale networks. We introduced the approach to reduce the energy consumption of network by changing the network topology dynamically, and the architecture to allow that the flexible reconfiguration of virtual network by using the OCSs. In this architecture, we configure the virtual network by using the minimum number of active devices based on current traffic demand, and power down so as to reduce the energy consumption. However, the existing virtual network reconfiguration methods cannot be suitable in this architecture, because the overhead of collecting the traffic information and the calculation time of optimizing the virtual network have been large.

First, we introduced the approach to reduce the overhead for collecting the traffic volume information by selecting a subset of nodes and by only collecting the traffic volume information from the selected nodes. We proposed the method to select the node collecting the traffic volume information, and estimate the traffic matrix on the basis of the traffic information collected from the selected nodes.

We evaluated our method through simulations. According to the simulation results, our method estimates the traffic matrix and the traffic volume on each link accurately enough to perform TE in real ISP topologies; however our method cannot accurately estimate the traffic matrix when the number of traffic demands passing each link is small. The simulation results showed that we can mitigate the congestion by using the traffic matrix estimated from 50% of all nodes in the case of

Japan topology and 30 % of all nodes in the case of AT&T topology.

Next, we introduced the concept of a virtual network configured over a data center network consisting of both OCSs and electronic switches. We proposed a method for constructing a virtual network by setting the parameters of its topology as well as a method for reconfiguring the virtual network within a short period of time by adjusting these parameters. In addition, we discussed implementation issues related to VNR-DCN and demonstrated that it can be applied to networks consisting of existing devices.

Through evaluation, we clarified that VNR-DCN constructs a topology satisfying the requirements on bandwidth and delay in the case of using a routing method based on GFB combined with VLB. We demonstrated that VNR-DCN is effective for reducing the energy consumption of the network by comparing it with the method based on powering down the ports of the ToR switches of a static electronic network. In addition, we also discussed the scalability of the VNR-DCN, and explain how the VNR-DCN works in a large data center.

Finally, we focused on the placement of VMs in multi tenant data centers so as to investigate the impact of energy consumption by considering the robustness against failure. We introduced a method to place the VMs considering the number of chunk replicas. The placement of the VMs should be obtained in a short time, because after failure, we reallocate the VMs before next failure occurs. Therefore, we proposed a heuristic method. In this method, we calculate the critical flow for each node, which is the flow whose connection becomes unavailable when the node fails. In our method, we place the VMs so that any node failure does not disconnect more than k VMs by using the information of the critical flows. Through numerical simulation, we demonstrated that our method keeps all data available even in the case of any single point of failure.

In addition, by using this method, we also investigated the cost of considering the number of chunk replicas. If k is set to a small value, more servers may be required, because the VMs should be placed at the different places to make the number of VMs that would be disconnected less than k. This may cause large energy consumption by powering on more servers. However, our results demonstrated that the cost of considering the number of chunk replicas becomes small as the number of tenants increases, and even when we set k to a small value, we can achieve the sufficient robustness without causing a large energy consumption.

Through the researches, we discussed about the method to construct the energy-efficient data center network. One of our future research topics is to reduce the overhead for collecting traffic information considering collecting interval. Traffic amount of some flows may change frequently, while traffic amount of the other flows may be stable. The information of the frequent changing traffic should be collected in a small time interval, while a large collecting interval is sufficient for the information of the stable traffic. Another topic is the virtual network reconfiguration considering the characteristics of accommodating applications. If some applications allow a large number of hops while the other applications are delay sensitive, the energy saving should be achieved without increasing the number of hops passed by the traffic of the delay-sensitive applications We conclude this thesis that the virtual network reconfiguration method in large optical network is one of the important topics to reduce the energy consumption, handle the traffic changes, and ensure robustness against failure in the future network. We believe that discussion in this thesis will contribute to the design of future networks.

Bibliography

[1] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, “Energy efficiency in the future internet: a survey of existing approaches and trends in energy-aware fixed network infrastructures,” *IEEE Communications Surveys & Tutorials*, vol. 13, pp. 223–244, May 2011.

[2] H. Ikebe, N. Yamashita, and R. Nishii, “Green energy for telecommunications,” in *Proceedings of IEEE 29th International Telecommunications Energy Conference(INTELEVC 2007)*, pp. 750–755, Oct. 2007.

[3] C. Bianco, F. Cucchietti, and G. Griffa, “Energy consumption trends in the next generation access network?a telco perspective,” in *Proceedings of IEEE 29th International Telecommunications Energy Conference(INTELEVC 2007)*, pp. 737–742, Oct. 2007.

[4] H. J. Chao and K. Xi, “Bufferless optical clos switches for data centers,” in *Proceedings of Optical Fiber Communication Conference*, Mar. 2011.

[5] H. Dorren, M. Hill, Y. Liu, N. Calabretta, A. Srivatsa, F. Huijskens, H. De Waardt, and G. Khoe, “Optical packet switching and buffering by using all-optical signal processing methods,” *Journal of Lightwave Technology*, vol. 21, pp. 2–12, Jan. 2003.

[6] A. Gencata and B. Mukherjee, “Virtual-topology adaptation for WDM mesh networks under dynamic traffic,” *IEEE/ACM Transactions on networking*, vol. 11, pp. 236–247, Oct. 2003.

[7] K. Shiomoto, E. Oki, W. Imajuku, S. Okamoto, and N. Yamanaka, “Distributed virtual network topology control mechanism in GMPLS-based multiregion networks,” *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 8, pp. 1254–1262, 2003.

[8] Y. Zhu and M. H. Ammar, “Algorithms for assigning substrate network resources to virtual network components.,” in *Proceedings of International Conference on Computer Communications (INFOCOM)*, pp. 1–12, Apr. 2006.

[9] G. Shen and R. Tucker, “Energy-minimized design for IP over WDM networks,” *IEEE/OSA Journal of Optical Communications and Networking*, vol. 1, pp. 176–186, June 2009.

[10] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, “Proteus: a topology malleable data center network,” in *Proceedings of ACM SIGCOMM Workshop on Hot Topics in Networks*, pp. 8–13, Oct. 2010.

[11] B. Heller, S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee, and N. McKeown, “ElasticTree: saving energy in data center networks,” in *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, pp. 1–16, Apr. 2010.

[12] G. Kuthethoor, P. Sesha, J. Strohm, P. O ’ Neal, J. DelMedico, N. Rome, D. Kiwior, D. Dunbrack, M. Bedford, D. Parker, *et al.*, “Performance analysis of SNMP in airborne tactical networks,” in *Proceedings of IEEE Military Communications Conference*, pp. 1–7, Nov. 2008.

[13] Yuya Tarutani, Yuichi Ohsita, Shin’ichi Arakawa and Masayuki Murata, “Selecting monitored links for estimating all link utilizations,” *Student Workshop on IEICE Technical Committe on Photonic Network*, Mar. 2010 (in Japanese).

[14] Yuya Tarutani, Yuichi Ohsita, Shin’ichi Arakawa and Masayuki Murata, “Estimation of traffic amounts on all links by using the information from a subset of nodes,” in *Proceedings of The Second International Conference on Emerging Network Intelligence (EMERGING 2010)*, pp. 18–23, Oct. 2010.

[15] Yuya Tarutani, Yuichi Ohsita, Shin’ichi Arakawa and Masayuki Murata, “Estimation of traffic amounts on all links by using the information from a subset of nodes,” *Technical Report of IEICE* (PN2010-27), vol. 110, pp. 19–24, Nov. 2010 (in Japanese).

[16] Yuya Tarutani, Yuichi Ohsita, Shin'ichi Arakawa and Masayuki Murata, "Optical-layer traffic engineering with link load estimation for large-scale optical networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 4, pp. 38–52, Jan. 2012.

[17] Yuya Tarutani, Yuichi Ohsita, Shin'ichi Arakawa and Masayuki Murata, "Optical layer traffic engineering using the traffic information from a subset of nodes," *Student Workshop on IEICE Technical Committe on Photonic Network*, Aug. 2011 (in Japanese).

[18] Yuya Tarutani, Yuichi Ohsita and Masayuki Murata, "A virtual network to achieve low energy consumption in large-scale datacenter," *Technical Report of IEICE* (PN2010-98), vol. 111, pp. 91–96, Mar. 2012 (in Japanese).

[19] Yuta Shimotsuma, Yuya Tarutani, Yuichi Ohsita and Masayuki Murata, "Evaluation of data center network structures considering routing methods," *Technical Report of IEICE* (IN2012-31), vol. 113, pp. 43–48, June 2012 (in Japanese).

[20] Yuya Tarutani, Yuichi Ohsita and Masayuki Murata, "A virtual network to achieve low energy consumption in optical large-scale datacenter," in *Proceedings of IEEE International Conference on Communication Systems (ICCS 2012)*, pp. 45–49, Nov. 2012.

[21] Yuya Tarutani, Yuichi Ohsita and Masayuki Murata, "Evaluation of data center network structures considering routing methods," in *Proceedings of The Ninth International Conference on Networking and Services (ICNS 2013)*, pp. 146–152, Mar. 2013.

[22] Yuya Tarutani, Yuichi Ohsita and Masayuki Murata, "Virtual network configuration method for low energy consumption in data centers," *IEICE General Conference*, Mar. 2014 (in Japanese).

[23] D. Abts, M. Marty, P. Wells, P. Klausler, and H. Liu, "Energy proportional datacenter networks," *ACM SIGARCH Computer Architecture News*, vol. 38, pp. 338–347, June 2010.

[24] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 68–73, Jan. 2008.

[25] L. Barroso and U. Holzle, “The case for energy-proportional computing,” *Computer*, vol. 40, pp. 33–37, Dec. 2007.

[26] C. Patel, C. Bash, R. Sharma, M. Beitelmal, and R. Friedrich, “Smart cooling of data centers,” in *Proceeding of International Electronic Packaging Technical Conference and Exhibition*, pp. 129–137, July 2003.

[27] M. Al-Fares, A. Loukissas, and A. Vahdat, “A scalable, commodity data center network architecture,” *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 63–74, Aug. 2008.

[28] J. Kim, W. Dally, and D. Abts, “Flattened butterfly: a cost-efficient topology for high-radix networks,” *ACM SIGARCH Computer Architecture News*, vol. 35, pp. 126–137, May 2007.

[29] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, and S. Lu, “DCell: a scalable and fault-tolerant network structure for data centers,” *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 75–86, Aug. 2008.

[30] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, and S. Lu, “BCube: a high performance, server-centric network architecture for modular data centers,” *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 63–74, Aug. 2009.

[31] D. Guo, T. Chen, D. Li, Y. Liu, X. Liu, and G. Chen, “BCN: expansible network structures for data centers using hierarchical compound graphs,” in *Proceedings of IEEE INFOCOM*, pp. 61–65, Apr. 2011.

[32] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, and J. Wu, “Scalable and cost-effective interconnection of data-center servers using dual server ports,” *IEEE/ACM Transactions on Networking*, vol. 19, pp. 102–114, Feb. 2011.

[33] A. Greenberg, J. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. Maltz, P. Patel, and S. Sengupta, “VL2: a scalable and flexible data center network,” *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 51–62, Aug. 2009.

[34] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, "PortLand: a scalable fault-tolerant layer 2 data center network fabric," *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 39–50, Aug. 2009.

[35] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Proceedings of USENIX Symposium on Networked Systems Design and Implementation*, pp. 1–14, Apr. 2012.

[36] A. R. Curtis, T. Carpenter, M. Elsheikh, A. López-Ortiz, and S. Keshav, "REWIRE: an optimization-based framework for unstructured data center network design," in *Proceedings of IEEE INFOCOM*, pp. 1116–1124, Mar. 2012.

[37] X. Wang, M. Chen, C. Lefurgy, and T. W. Keller, "Ship: A scalable hierarchical power control architecture for large-scale data centers," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, pp. 168–176, Jan. 2012.

[38] Y. Zhang and N. Ansari, "On architecture design, congestion notification, TCP incast and power consumption in data centers," pp. 39–64, Feb. 2013.

[39] Yuya Tarutani, Yuichi Ohsita and Masayuki Murata, "Virtual network reconfiguration with fault-tolerance and low energy consumption for multi-tenant data centers," *Technical Report of IEICE (PN2013-193)*, vol. 113, pp. 293–298, Mar. 2014 (in Japanese).

[40] "Amazon ec2." `http://aws.amazon.com/`.

[41] J. Mudigonda, P. Yalagandula, J. Mogul, B. Stiekes, and Y. Pouffary, "Netlord: a scalable multi-tenant network architecture for virtualized datacenters," *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 62–73, Aug. 2011.

[42] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 38–47, Apr. 2011.

[43] T. Guo, N. Wang, K. Moessner, and R. Tafazolli, "Shared backup network provision for virtual network embedding," in *Proceedings of ICC*, pp. 1–5, June 2011.

[44] H. Yu, V. Anand, C. Qiao, and G. Sun, "Cost efficient design of survivable virtual infrastructure to recover from facility node failures," in *Proceedings of IEEE ICC*, pp. 1–6, June 2011.

[45] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun, "Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures," in *Proceedings of IEEE GLOBECOM*, pp. 1–6, Dec. 2010.

[46] I. Houidi, W. Louati, D. Zeghlache, P. Papadimitriou, and L. Mathy, "Adaptive virtual network provisioning," in *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures*, pp. 41–48, Sept. 2010.

[47] P. Bodík, I. Menache, M. Chowdhury, P. Mani, D. A. Maltz, and I. Stoica, "Surviving failures in bandwidth-constrained datacenters," in *Proceedings of the ACM SIGCOMM*, pp. 431–442, Aug. 2012.

[48] A. Fischer, J. Botero, M. Beck, H. De Meer, and X. Hesselbach, "Virtual network embedding: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, pp. 1888–1906, Nov. 2013.

[49] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, pp. 17–29, Apr. 2008.

[50] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, pp. 206–219, Feb. 2012.

[51] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures*, pp. 81–88, Aug. 2009.

[52] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. De Meer, "Energy efficient virtual network embedding," *IEEE Communications Letters*, vol. 16, pp. 756–759, May 2012.

[53] S. Hong, J. P Jue, Q. Zhang, X. Wang, H. C. Cankaya, Q. She, and M. Sekiya, "Effective virtual optical network embedding based on topology aggregation in multi-domain optical networks," in *Proceedings of Optical Fiber Communication Conference*, pp. 1–3, Mar. 2014.

[54] M. Rahman and R. Boutaba, "Svne: Survivable virtual network embedding algorithms for network virtualization," *IEEE Transactions on Network and Service Management*.

[55] S. Ghemawat, H. Gobioff, and S. Leung, "The google file system," in *Proceeding of ACM SIGOPS Operating Systems Review*, vol. 37, pp. 29–43, Dec. 2003.

[56] "Apache hadoop project." `http://hadoop.apache.org/`.

[57] Y. Vardi, "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American Statistical Association*, vol. 91, pp. 365–377, Mar. 1996.

[58] J. Cao, D. Davis, S. Wiel, and B. Yu, "Time-varying network tomography: Router link data," *Journal of the American Statistical Association*, vol. 95, no. 452, pp. 1063–1075, 2000.

[59] I. Juva, S. Vaton, and J. Virtamo, "Quick traffic matrix estimation based on link count covariances," in *Proceedings of IEEE ICC*, pp. 603–608, June 2006.

[60] A. Soule, A. Nucci, R. Cruz, E. Leonardi, and N. Taft, "Estimating dynamic traffic matrices by using viable routing changes," *IEEE/ACM Transactions on Networking*, vol. 15, pp. 485–498, June 2007.

[61] Y. Zhang, M. Roughan, N. Duffield, and A. Greenberg, "Fast accurate computation of large-scale IP traffic matrices from link loads," *ACM SIGMETRICS Performance Evaluation Review*, vol. 31, pp. 206–217, June 2003.

[62] Y. Zhang, M. Roughan, W. Willinger, and L. Qiu, “Spatio-temporal compressive sensing and Internet traffic matrices,” *ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 267–278, Aug. 2009.

[63] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, “Measuring ISP topologies with Rocketfuel,” *IEEE/ACM Transactions on networking*, vol. 12, pp. 2–16, Feb. 2004.

[64] A. Nucci, A. Sridharan, and N. Taft, “The problem of synthetically generating IP traffic matrices: Initial recommendations,” *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 19–32, July 2005.

[65] R. Ramaswami, K. Sivarajan, I. Center, and Y. Heights, “Design of logical topologies for wavelength-routed optical networks,” *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 840–851, June 1996.

[66] “Glimmerglass intelligent optical system 600.” `http://www.glimmerglass.com/products/intelligent-optical-system-600/`.

[67] “Arista 7148sx switch.” `http://www.aristanetworks.com/media/system/pdf/Datasheets/7100_Datasheet.pdf`.

[68] J. Dean and S. Ghemawat, “Mapreduce: Simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[69] M. Kodialam, T. Lakshman, and S. Sengupta, “Efficient and robust routing of highly variable traffic,” in *Proceedings of HotNets*, Nov. 2004.

[70] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The nature of data center traffic: measurements & analysis,” in *Proceedings of ACM SIGCOMM conference on Internet measurement conference*, pp. 202–208, Nov. 2009.

[71] T. Benson, A. Akella, and D. A. Maltz, “Network traffic characteristics of data centers in the wild,” in *Proceedings of ACM SIGCOMM conference on Internet measurement*, pp. 267–280, Nov. 2010.

[72] "Delta 10gbase-sr sfp+ optical transceiver." `http://www.deltaww.com/filecenter/Products/download/04/0405/DataCenter/LCP-10G3A4EDRx-G_S2.pdf`.

[73] Y.-K. Yeo, Z. Xu, C.-Y. Liaw, D. Wang, Y. Wang, and T.-H. Cheng, "A 448× 448 optical cross-connect for high-performance computers and multi-terabit/s routers," in *Proceeding of Optical Fiber Communication, collocated National Fiber Optic Engineers Conference*, pp. 1–3, Mar. 2010.

[74] P. Reviriego, V. Sivaraman, Z. Zhao, J. A. Maestro, A. Vishwanath, A. Sánchez-Macián, and C. Russell, "An energy consumption model for energy efficient ethernet switches," in *Proceeding of International Conference on High Performance Computing and Simulation (HPCS)*, pp. 98–104, July 2012.

[75] "Cisco Nexus 2000 Series Fabric Extenders Data Sheet." `http://www.cisco.com/c/en/us/products/collateral/switches/nexus-2000-series-fabric-extenders/data_sheet_c78-507093.html`.

[76] "Cisco Nexus 5548P, 5548UP, 5596UP, and 5596T Switches Data Sheet." `http://www.cisco.com/c/en/us/products/collateral/switches/nexus-5000-series-switches/data_sheet_c78-618603.html`.

[77] "PowerEdge R320 rack server." `http://i.dell.com/sites/doccontent/shared-content/data-sheets/en/Documents/Dell_PowerEdge_R320_Spec_Sheet.pdf`.