Master's Thesis

Title

Virtual Network Topology Control to Achieve Low Energy Consumption in an Optical Data Center Network

Supervisor

Professor Masayuki Murata

Author

Yuya Tarutani

February 14th, 2012

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Virtual Network Topology Control to Achieve Low Energy Consumption in an Optical Data Center Network

Yuya Tarutani

Abstract

In recent years, online services such as cloud computing have become popular, and the amount of data, required to be processed by such online services, is increasing. To handle such a large amount of data, data centers with hundreds of thousands of servers have been built. In a data center, a large amount of data is handled by distributed computing frameworks, which require the communication between servers. Thus, the data center network should provide communication with sufficiently large bandwidth and small delay between all servers so that the data center can handle a large amount of data efficiently. On the other hand, energy consumption of the data center network should be reduced, because energy consumption of the data center network occupies the non-negligible fractions of the total energy consumption in the data center.

In this thesis, we introduce the virtual network configured over the data center network constructed of the optical cross connects (OXCs) and the electronic switches. In this network, the core of the data center network is constructed by using the OXCs and optical fibers. Then, the electronic switches, deployed in each server rack, are connected to the core network by connecting them to OXCs. The virtual network topology (VNT) is constructed by establishing the lightpath between the electronic switches. In our method, we minimize the energy consumption of the data center network by minimizing the number of ports of electronic switches used in the VNT and shutting down the unused ports.

Though the VNT control methods have been proposed in many papers, they are not applicable to a data center network because they do not consider the situation that the traffic changes within a second, or their calculation time is too large for a data center network.

Therefore, we also propose a method to reconfigure the VNT suitable for a data center network. In this method, the traffic changes in a short period are handled by the load balancing over the

VNT. We design the VNT so as to achieve sufficiently large bandwidth and small delay with small energy consumption, considering the load balancing. Then, if the current VNT is not suitable to the current demands considering the load balancing, the VNT is reconfigured. Our method reconfigures the VNT by setting parameters of a topology so as to avoid large calculation time in a data center. As the topology used in the VNT configuration, we propose the topology called *Generalized Flattened Butterfly (GFB)*, and a method to set its parameters so as to suit the current condition.

In our evaluation, we first check whether our method satisfies the requirements of the bandwidth and the maximum number of hops by changing the number of ToR switches in the virtual network, the target traffic volume from each ToR switch and the target maximum number of hops. From the result, we clarify that our method satisfies all the requirements used in our evaluation. In addition, we compare the topology constructed by our method with the existing data center network topologies. We clarify that our method achieves the sufficient bandwidth by using a smaller number of virtual links than the existing data center networks; the number of ports used by our method is about a half of the number of ports used by the flattened butterfly.We also clarify that our method achieves the target maximum number of hops between ToR switches by using the similar number of virtual links to the existing data center networks. Finally, we evaluate our method when the number of ToR switches connected to the virtual network changes. From the results, we clarify that our method keeps the requirements of the bandwidth and the maximum number of hops satisfied even when the changes of the number of ToR switches occur.

Keywords

Data Center

Energy Consumption

Virtual Network Topology

Optical Network

Contents

List of Figures

List of Tables

1 Introduction

In recent years, online services such as cloud computing have become popular, and the amount of data, required to be processed by such online services, is increasing. To handle such a large amount of data, large data centers with hundreds of thousands of servers have been built.

In a data center, data is stored in the memories or storages of multiple servers by using distributed file systems such as Google File System [1]. Then, we handle the data by using the distributed computing frameworks such as MapReduce [2] or Hadoop [3]. The distributed file systems or disrtibuted computing frameworks require the communication between servers within a data center. Thus, the data center network affects the performance of the data center.

Data center network should provide communication with high bandwidth and small delay between all servers so that the data center can handle a large amount of data efficiently. The lack of bandwidth or large delay between servers may prevent the communication between servers, and degrade the performance of the application of the data center handling a large amount of data. However, the traditional data center network, which is constructed as a tree topology, cannot provide communication with sufficiently large bandwidth and small delay between servers; in the traditional data center network, the root of tree topology becomes the bottleneck, and the number of hops between servers becomes large as the number of servers in a data center increases.

Energy consumption is another problem in a data center network. Energy consumption of the data center network occupies a non-negligible fraction of the total energy consumption in the data center [4], and becomes large as the size of the network increases. Thus, to reduce energy consumption of a data center, the energy consumption of the data center network should be reduced.

There are many researches to construct a data center network [5–13]. Al-Fares et al. have proposed the topology called *FatTree* [5], that provides sufficient bandwidth between all server pairs. The FatTree is a tree with multiple root nodes. In this topology, each node uses a half of its ports to connect it to the nodes of the upper layer, and the other half of its ports to connect it to the nodes of the lower layer. Another topology to provide sufficient bandwidth has been proposed by Kim et al. [6]. This topology is called the flattened butterfly. The flattened butterfly provides enough bandwidth between servers, and makes the number of hops between servers small, by using nodes with a large number of ports instead of constructing the tree.

The methods to connect a large number of servers using a small number of switches have also been proposed by C. Guo et al. [7,8], D. Guo et al. [9], and Li et al. [10], and Liao et al. [11]. These topologies use the servers having multiple ports, and are constructed by directly connecting server ports. Though these topologies make their energy consumption small by reducing the number of required switches and links, they may not provide enough bandwidth between all servers.

The topology suitable to the data center network depends on the applications and the current load of the data center. For the application where the servers exchange a large amount of data with each other, the network should provide large bandwidth between all servers related to the application. On the other hand, in the case of the application, where the servers exchange only a small amount of data, the small bandwidth is sufficient, and the topology whose energy consumption is small is preferable.

However, the demands of the data center may change. Servers related to the application, which suddenly become popular, may start exchanging a large amount of data. Additional servers related to the application may be implemented to handle the suddenly increased demand for the application. As a result, the data center network becomes no longer suitable for the current applications and loads. Though we can avoid the lack of bandwidth or large delay by constructing a redundant network, this approach consumes large energy.

In this thesis, we introduce the virtual network configured over the data center network constructed of the optical cross connects(OXCs) and the electronic switches. In this network, the core of the data center network is constructed by using the OXCs and optical fibers. Then, the electronic switches, deployed in each server rack, are connected to the core network by connecting them to OXCs. A lightpath is established between two electronic switches by configuring the OXCs along the route between the electronic switches. A set of the lightpaths forms a virtual network topology (VNT). Traffic between electronic switches is carried over the VNT.

In our method, we minimize the energy consumption of the data center network by minimizing the number of ports of electronic switches used in the VNT and shutting down the unused ports, because energy consumption of electronic switches is much larger than that of OXCs. In the cases of the changes of demands, we keep the sufficiently large bandwidth, small delay between servers and low energy consumption by reconfiguring the VNT.

Dynamic reconfiguration of the VNT constructed over the optical network has also been discussed in many papers [14–18]. However, most of them aim to optimize the VNT for the monitored or estimated current traffic demand, and are not applicable to the data center network, where the traffic changes within a second [19]. Moreover, their calculation time is too large for a data center.

Therefore, we propose a method to reconfigure the VNT suitable for a data center network. In our method, the traffic changes in a short period are handled by the load balancing [20] over the VNT. We design the VNT so as to achieve sufficiently large bandwidth and small delay with low energy consumption, considering the load balancing. Then, if the current VNT is not suitable to the current demands, the VNT is reconfigured. Our method reconfigures the VNT by setting parameters of a topology so as to avoid large calculation time in a data center. As the topology used in the VNT configuration, we propose the topology called *Generalized Flattened Butterfly (GFB)*. We also propose a method to set the parameters so as to suit the current condition.

The rest of this thesis is organized as follows. In Section 2, we explain the overview of the data center network using the OXCs and the electronic switches. In Section 3, we discuss the VNT suitable to optical data center networks, and we propose the GFB. In Section 4, we propose a method to control the VNT by setting the parameters of the GFB. Then, we evaluate our method and clarify that our method can provide enough communication performance between servers with low energy consumption in Section 5. Finally, Section 6 provides a conclusion.

2 Virtual Networks on an Optical Data Center Network

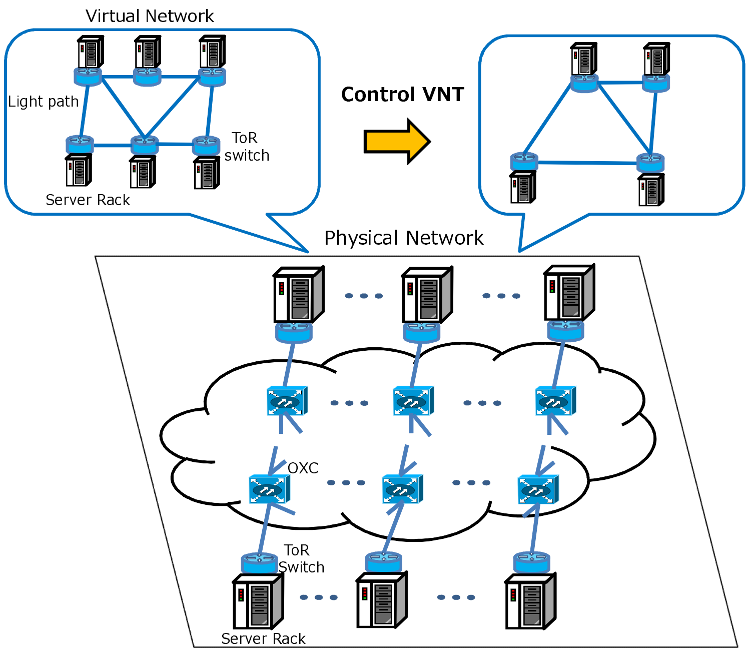

Figure 1: Data Center Network Using OXCs

In this thesis, we construct the data center network using optical cross connects (OXCs) and electronic packet switches as shown in Figure 1. In this network, similar to the traditional data center networks, electronic switches are deployed in the server rack. We call the electronic switches in server racks *top-of rack (ToR) switches*. All servers in a server rack are connected to a ToR switch in the rack with an 1 Gbps link.

In our data center network, we construct the core network by using OXCs. In the core network of our data center, OXCs are connected by the optical fibers, and the wavelength multiplexed signals are sent between OXCs. In each OXC, the input signals are demultiplexed into the signals of each wavelength, and relayed to the destination ports. Then, the signals are multiplexed into wavelength multiplexed signals and sent to the next OXCs. In this network, we can set lightpaths for each wavelength. By setting the lightpaths, the signals of the lightpaths are sent along the defined routes.

In our data center network, each ToR switch has multiple ports that can terminate the lightpath. We connect the ToR switch to the core network by connecting the ports to OXCs. We set the lightpath between the ports of the ToR switches. A set of the lightpath forms a VNT, which is constructed of the ToR switches and virtual links between ToR switches. Traffic between ToR switches is carried over the VNT. The VNT can be easily reconfigured by adding or deleting lightpaths if the current VNT is no longer suitable.

In our data center network, most of energy is consumed by the ToR switches, because energy consumption of OXCs is much smaller than that of ToR switches. Thus, we focus on the energy consumption of the ToR switches. In this thesis, we assume that we can shut down each port of ToR switches. Then, we reduce the energy consumption by minimizing the number of ports of ToR switches used in the VNT and shutting down the unused ports.

3 Virtual Network Topologies Suitable to Optical Data Center Netwoks

Because it is difficult to obtain the optimal topology for a large data center network in a short time, our VNT reconfiguration method constructs the VNT by setting parameters of a topology, which is suitable for data center networks, instead of calculating the optimal topology.

In this section, we discuss the requirements for the topology used by our VNT reconfiguration, and investigate the properties of the existing network topology for the data center network. Then, we propose a new topology called *generalized flattened butterfly*, which can construct various data center networks by setting parameters.

3.1 Requirement

The virtual network should satisfy the following requirements.

Low Energy Consumption Energy consumption of the network occupies a non-negligible fraction of the total energy consumption in the data center. To reduce energy consumption of the data center, energy consumption of the data center network should be reduced.

In the data center network used in this thesis, most of energy is consumed by the ToR switches, and the energy consumption of the ToR switches can be reduced by shutting down their unused ports. Thus, by constructing the virtual network using the smallest number of ports of ToR switches, the energy consumption of the data center network is minimized.

Large Bandwidth between Servers In some applications such as distributed file system, a large amount of data is exchanged between servers. The bandwidth provided between servers is important for such applications; the lack of bandwidth increases the time required to transport data. Therefore, the VNT should provide sufficient bandwidth between servers.

Small Delay between Servers A data center handles a large amount of data by using the distributed computing frameworks. In the distributed computing frameworks, a large number of servers communicate with each other. If the delays between servers are large, it takes time to obtain the required data from other servers, and the performance of the data center is degraded.

Thus, the delay should be kept small enough for the application of the data center.

The delays between servers are difficult to forecast when constructing the VNT, because the delays are affected by traffic load. In our thesis, we keep the small delay between servers by constructing the VNT which can provide sufficient bandwidth and make the number of hops between servers small.

3.2 Existing Data Center Network Topologies

3.2.1 FatTree

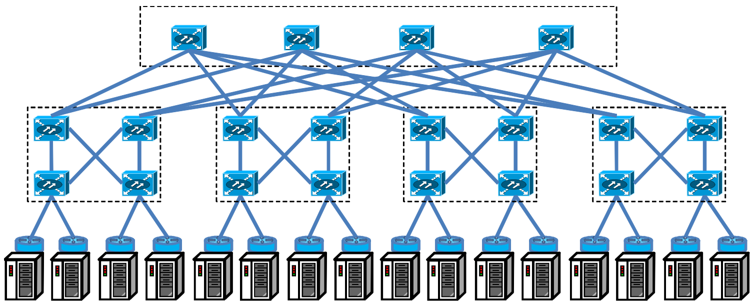

Figure 2: FatTree

Al-Fares et al. have proposed the method to construct the topology called *FatTree* by using switches with a small number of ports [5]. The FatTree is a tree topology constructed of multiple roots and multiple pods containing multi-layer of switches as shown in Figure 2.

Each pod is regarded as the virtual switch having a large number of ports constructed by multiple switches having a small number of ports. Pods are constructed as the butterfly topology, where each switch uses a half of its ports to connect it to the switches of the upper layer, and the other half of its ports to connect it to the switches of the lower layer. The switches at the lowest layer are connected to the servers.

Though the method proposed by Al-Fares et al. [5] constructs the 3-layer FatTree, which is constructed of root switches and the pods containing two layers of switches, we can construct the higher-layer FatTree topologies. The k-layer FatTree constructed of switches with n ports includes $(2k-1)\frac{n}{2}^{k-1}$ switches.

In the FatTree, the number of links from the lower-layer switch equals the number of links to the upper-layer switch at each switch. That is, the sum of bandwidth from a switch to the upper layer equals that from the lower layer to the switch. Therefore, any switch does not become a bottleneck, and we can provide sufficiently large bandwidth between all servers.

However, in the FatTree, a large number of switches or switches with a large number of ports are necessary to construct large data center networks, which leads to large energy consumption.

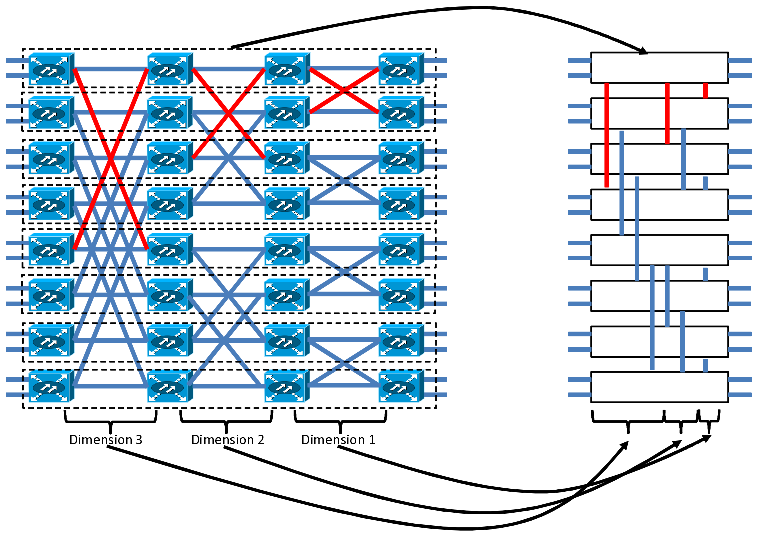

Figure 3: Flattened Butterfly

3.2.2 Flattened Butterfly

Kim et al. [6] have proposed the data center network topology called *flattened butterfly*. The flattened butterfly is constructed by *flattening* the butterfly topology as shown in Figure 3; we combine the switches in each row of the butterfly topology into a single switch.

The flattened butterfly provides sufficiently large bandwidth between all servers with lower energy consumption than the FatTree [5]. However, the energy consumption of the flattened butterfly is still large because the flattened butterfly requires the switches with a large number of ports to construct a large data center network.

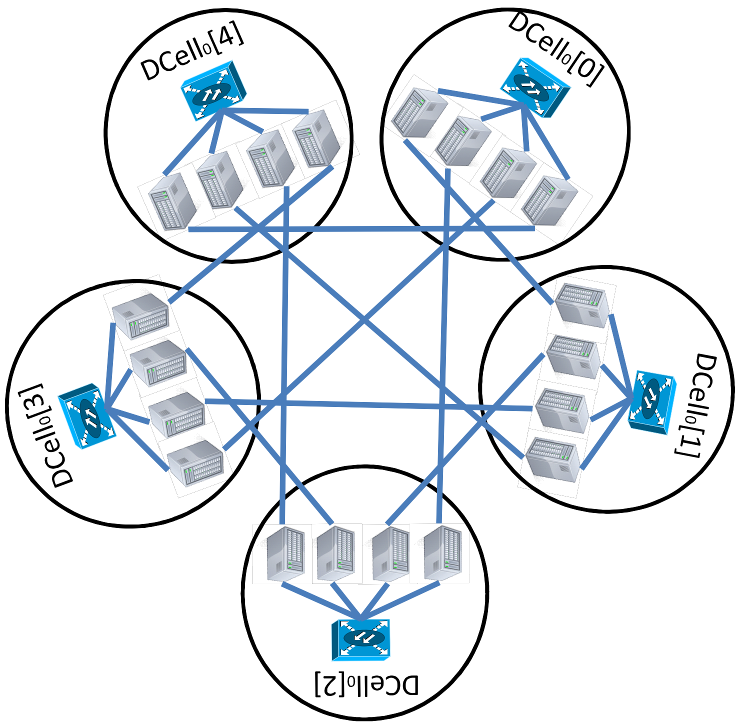

Figure 4: DCell

3.2.3 DCell

Guo et al. have proposed a data center network called *DCell*, which is constructed from a small number of switches and servers with multiple ports as shown in Figure 4 [7]. 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, DCell reduces the number of switches required to construct a large data center network. However, DCell is not used as the topology of the virtual network introduced in this thesis, which is constructed of ToR switches.

Therefore, we introduce the topology called *switch-based DCell*, where the level-0 DCell is replaced with the fully-connected network constructed of switches as shown in Figure 5. Similar to the DCell, the switch-based DCell can construct a large data center network by using the switches with a small number of ports. That is, the switch-based DCell achieves low energy consumption.

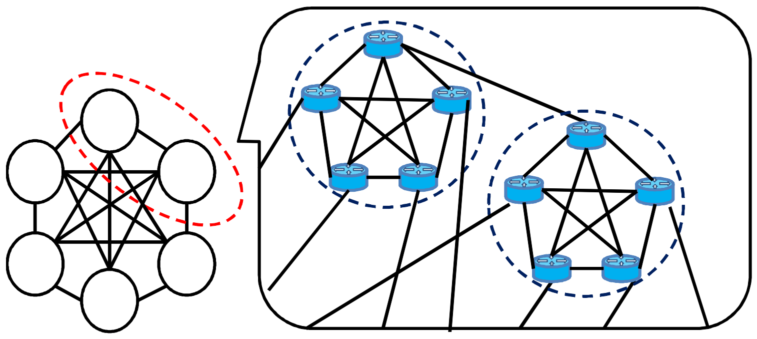

Figure 5: Switch-based DCell

However, the switch-based DCell cannot provide large bandwidth between all servers, because the switch-based DCell has only one link between each lower-level DCells.

3.3 Generalized Flattened Butterfly

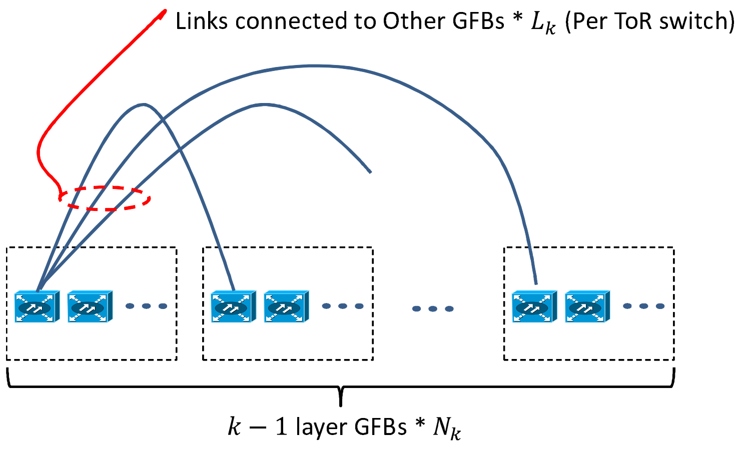

Figure 6: Generalized Flattened Butterfly

As discussed in the previous subsection, the switch-based DCell consumes only small energy, and is suitable when the traffic volume between servers is small. However, it may not provide sufficient bandwidth for the application where servers generate a large amount of traffic. On the other hand, the flattened butterfly can provide large bandwidth between servers but consumes large energy. That is, the topology suitable to the data center network depends on the applications and current loads.

In this subsection, we propose a topology called *Generalized Flattened Butterfly (GFB)*. The GFB is constructed hierarchically as shown in Figure 6; the upper-layer GFB is constructed by connecting multiple lower-layer GFBs. The GFB has the following parameters.

- Number of layers: k
- Number of links per node used to construct layer-k GFB: L_k
- Number of layer-$k-1$ GFBs used to construct layer-k GFB: N_k
- Minimum number of links constructed between layer-$k-1$ GFBs: M_k

By setting these parameters, we can construct various topologies; the switch-based DCell is constructed by setting $L_0 = N_0$ and $L_k = 1$ for $k > 1$, and the flattened butterfly is constructed by setting $L_k = N_k$.

In the GFB, the number of required ports, the maximum number of hops and the bandwidth provided between servers can be changed by setting the parameters. Therefore, we construct the topology suitable for the requirements of the application and current traffic loads by setting the parameters of the GFB.

3.3.1 Steps to Construct the Generalized Flattened Butterfly

The layer-k GFB is constructed by the following two steps.

- Construct the connections between the layer-$k-1$ GFBs.
- Select the switches connected to the links between each layer-$k-1$ GFB pair

In each step, we use the ID assigned for the GFBs of each layer. The switch can be identified by the set of IDs of the GFBs the switch belongs to. We denote the ID of the layer-k GFB the switch s belongs to as $D_k^{gfb}(s)$. We also define the ID of the switch s in the layer-k GFB by

$$D_k^{sw}(s) = (\sum_{1 \le i < k} D_i^{gfb}(s)) \prod_{j=1}^{k-1} N_j.$$

Connections between layer-$k-1$ GFBs We construct the connections between the layer-$k-1$ GFBs by the following steps.

Step 1 Calculate the number of links used to connect one layer-$k-1$ GFB to the other layer-$k-1$ GFBs, L_k^{gfb}, by

$$L_k^{gfb} = L_k \prod_{i=1}^{k-1} N_i \tag{1}$$

Step 2 If L_k^{gfb} is larger than $M_k(N_k-1)$, we can connect all layer-$k-1$ GFB pairs with more than M_k links. Thus, add $\lfloor \frac{L_k \prod_{i=1}^{k-1} N_i}{M_k(N_k-1)} \rfloor M_k$ links between all layer-$k-1$ GFB pairs.

Step 3 If L_k^{gfb} is smaller than $M_k(N_k-1)$, construct the ring topology by connecting the GFBs having the nearest ID with M_k links.

Step 4 Calculate the number of the residual links $L_k^{'gfb}$ which can be used to connect one layer-$k-1$ GFB to the other layer-$k-1$ GFBs by

$$L_k^{'gfb} = L_k^{gfb} - \bar{L}_k^{gfb} \tag{2}$$

where $\bar{L}_k^{gfb}$ is the number of links per layer-$k-1$ GFB constructed at Steps 2 and 3.

Step 5 Connect the GFB of ID $D_{k-1}^{gfb}(a)$ to the GFB of ID $D_{k-1}^{gfb}(b)$ when the following equation is satisfied;

$$D_{k-1}^{gfb}(b) = (D_{k-1}^{gfb}(a) + Cp_k) \text{ mod } N_k, \tag{3}$$

where C is a integer value and

$$p_k = \frac{N_k}{L_k^{'gfb} + 1}. \tag{4}$$

Selection of the switches used to connect layer-$k-1$ GFBs After constructing the connections between layer-$k-1$ GFBs, we select the switches that are used to connect the layer-$k-1$ GFB pair. We select the switch used to connect the GFB of ID $D_{k-1}^{gfb}(a)$ and the GFB of ID $D_{k-1}^{gfb}(b)$ from the switches belonging to the GFB of ID $D_{k-1}^{gfb}(a)$ by the following steps.

Step 1 Calculate the rank of ID of the GFB, $R(D_{k-1}^{gfb}(b))$ indicating that $D_{k-1}^{gfb}(b)$ is the $R(D_{k-1}^{gfb}(b))$-th smallest ID among the IDs of the GFBs connected to the GFB of ID $D_{k-1}^{gfb}(a)$.

Step 2 Select the candidate switches whose switch ID $D_k^{sw}(s)$ satisfy the following equation.

$$D_k^{sw}(s) = R(D_{k-1}^{gfb}(b)) + \left\lfloor \frac{Cn_{D_{k-1}^{gfb}(a)}}{l_{(D_{k-1}^{gfb}(a), D_{k-1}^{gfb}(b))}} \right\rfloor$$

where C is a integer value, $n_{D_{k-1}^{gfb}(a)}$ is the number of switches in the GFB of ID $D_{k-1}^{gfb}(a)$, and $l_{(D_{k-1}^{gfb}(a), D_{k-1}^{gfb}(b))}$ is the number of links to be constructed between GFBs of IDs $D_{k-1}^{gfb}(a)$ and $D_{k-1}^{gfb}(b)$.

Step 3 Select the switch with the smallest number of ports used to connect to the GFB of ID $D_{k-1}^{gfb}(b)$ among the candidate switches.

Step 4 Check whether the switch selected at Step 3 has residual ports to be used to connect layer-$k-1$ GFBs. If yes, designate the switch selected at Step 3 as the switch used to connect to the GFB of ID $D_{k-1}^{gfb}(b)$. Otherwise, designate the switch having the nearest switch ID to the switch selected at Steps 3 and having the residual ports used to connect layer-$k-1$ GFBs as the switch used to connect the GFB.

3.3.2 Properties of the Generalized Flattened Butterfly

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

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

$$H_k = (h_k + 1)H_{k-1} + h_k, \tag{5}$$

where h_k is the largest number of links between layer-$k-1$ GFBs passed by the traffic between layer-$k-1$ GFBs. H_k is obtained by calculating h_k. In the rest of this paragraph, we discuss how to calculate h_k from the parameters of the GFB.

If L_k^{gfb} defined by Eq. (1) is larger than $M_k(N_k - 1)$, we add links between all pairs of layer-$k-1$ GFBs. Thus, $h_k = 1$.

If L_k^{gfb} is smaller than $M_k(N_k - 1)$ and $L_k^{'gfb}$ defined by Eq. (2) is zero, the connections between layer-$k-1$ GFBs form a ring topology. In this case, h_k is $\lfloor \frac{N_k}{2} \rfloor$.

If L_k^{gfb} is smaller than $M_k(N_k - 1)$ and $L_k^{'gfb}$ is a positive value, we add links to the GFBs satisfying Eq. (3). In this case, we discuss the calculation of h_k by dividing the topology constructed of layer-$k-1$ GFBs into modules so that each module includes the GFBs whose IDs are within the range from Cp_k to $(C+1)p_k$ where C is a integer variable and p_k is defined by Eq. (4). Then, we calculate the maximum number of hops from the source GFB whose ID is zero. Since all low-layer GFBs play the same role in the high-layer GFB, h_k is calculated by calculating the maximum number of hops from the GFB whose ID is zero.

The maximum number of hops to the GFBs in each module depends on whether p_k is odd or even. Figures 7 and 8 show the examples of the cases that p_k is even or p_k is odd. In these figures, each circle indicates a layer-$k-1$ GFB, and numbers in the circles indicate the number of hops from the source GFB. As shown in these figures, if the source GFB does not belong to the module of the destination GFB, the maximum number of hop counts is $\lfloor \frac{p_k}{2} \rfloor + 1$. If the source GFB belongs to the module of the destination GFB and p_k is even, the maximum number of hops is $\lfloor \frac{p_k}{2} \rfloor$. Unless the number of modules is two, the module which does not include the source GFB exists. The number of modules becomes two when $L_k^{'gfb}$ is 1. Thus, h_k is $\lfloor \frac{p_k}{2} \rfloor$ only when $L_k^{'gfb}$ is 1 and p_k is even. Otherwise, h_k is $\lfloor \frac{p_k}{2} \rfloor + 1$.

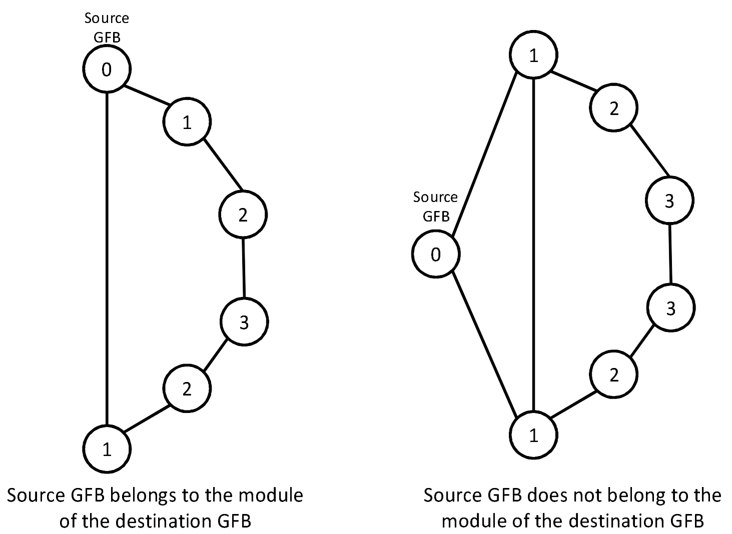
Source GFB
0
1
2
3
2
1
Source GFB belongs to the module of the destination GFB
1
2
3
Source GFB
0
3
2
1
Source GFB does not belong to the module of the destination GFB

Figure 7: Example of number of hops in the topology constructed of low-layer GFBs (p_k is odd)

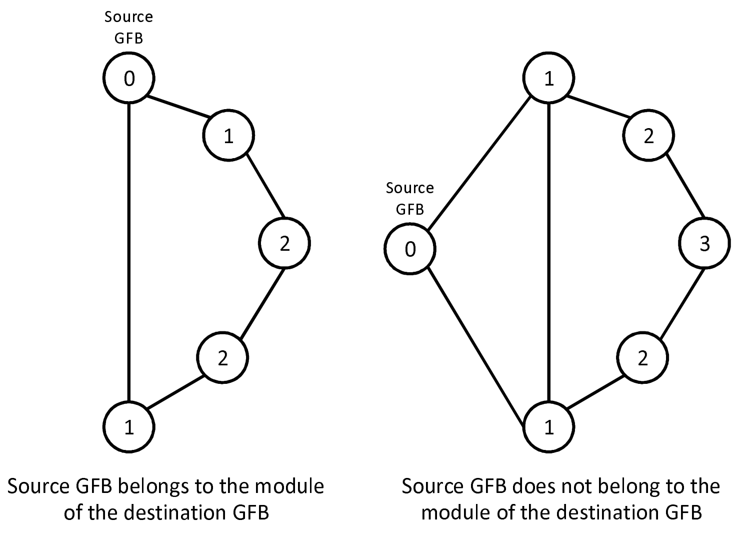
Source GFB
0
1
2
2
1
Source GFB belongs to the module of the destination GFB
Source GFB
0
1
2
3
2
1
Source GFB does not belong to the module of the destination GFB

Figure 8: Example of number of hops in the topology constructed of low-layer GFBs (p_k is even)

Summarizing the above discussion, h_k is calculated by

$$h_k = \begin{cases} 1 & (L_k^{gfb} \geq M_k(N_k - 1)) \\ \lfloor \frac{N_k}{2} \rfloor & (L_k^{gfb} < M_k(N_k - 1) \text{ and } L_k^{'gfb} = 0) \\ \lfloor \frac{p_k}{2} \rfloor & (L_k^{gfb} < M_k(N_k - 1) \text{ and } L_k^{'gfb} = 1 \text{ and } p_k \text{is even}) \\ \lfloor \frac{p_k}{2} \rfloor + 1 & (Otherwise) \end{cases} \tag{6}$$

Eq. (5) assumes that the paths with the maximum number of hops pass the routes with the maximum number of hops in each layer. If there are multiple links between a certain GFB pairs, there may be routes from a switch to the outside of the GFB that do not pass the routes with the maximum number of hops in a layer. Thus, the calculated H_k may be larger than the actual maximum number of hops. However, H_k is useful because we can construct the topology where the maximum number of hops is less than a threshold by making H_k less than the threshold.

Number of Flows through a Link We calculate the number of flows passing each link when one flow is generated between each switch pair. In the GFB, all switches play the same role. Thus, the number of flows passing a link depends only on the layer of the GFBs the link connects. X_k denotes the number of flows passing the link between layer-$k-1$ GFBs.

The flows between layer-$k-1$ GFBs are balanced among all links between layer-$k-1$ GFBs.

Thus, X_k is the sum of the number of flows passing the links between layer-$k-1$ GFBs divided by the number of the links. That is,

$$X_k = T_k \frac{\sum_{i=1}^{h_k} i s_k(i)}{L_k \prod_{i=1}^{k} N_i} \quad (7)$$

where T_k is the number of flows between each layer-$k-1$ GFB pair, $s_k(i)$ is the number of layer-$k-1$ GFB pairs whose flow passes i links between layer-$k-1$ GFBs. Thus, X_k is obtained by calculating $s_k(i)$ and T_k.

$s_k(i)$ is calculated by the similar manner to h_k as follows.

If L_k^{gfb} defined by Eq. (1) is larger than $M_k(N_k-1)$, links are added between all pairs of layer-$k-1$ GFBs, and the numbers of hops between all GFB pairs are 1. In this case, $s_k(1) = N_k(N_k-1)$ and $s_k(i) = 0$ for $i > 1$.

If L_k^{gfb} is smaller than $M_k(N_k-1)$ and $L_k^{'gfb}$ is zero, the connections between layer-$k-1$ GFBs form a ring topology. In this case, the number of layer-$k-1$ GFBs i hops away from a certain GFB is two for $i < h_k$. The number of layer-$k-1$ GFBs h_k hop away from a certain layer-$k-1$ GFB depends on whether N_k is odd or even; the number of layer-$k-1$ GFBs h_k hop away from a certain layer-$k-1$ GFB is two if N_k is odd, or one if N_k is even.

If L_k^{gfb} is smaller than $M_k(N_k-1)$ and $L_k^{'gfb}$ is a positive value, we add links to the GFBs satisfying Eq. (3). In this case, each layer-$k-1$ GFB has $L_k \prod_{i=1}^{k-1} N_i$ links. Thus, $s_k(1)$ is $N_k L_k \prod_{i=1}^{k-1} N_i$.

To calculate $s_k(i)$ for $i > 1$ in the case that links are added between GFBs satisfying Eq. (3), we divide the topology constructed of layer-$k-1$ GFBs into modules similar to the steps to calculate h_k, and calculate the number of GFBs i hops away from the source GFB whose ID is zero.

Each module has two GFBs i hops away from the source GFB for $1 < i < h_k$. Thus, $N_k(\frac{N_k}{p_k} - 2)$ for $1 < i < h_k$.

The number of GFBs h_k hops away from the source GFB depends on whether p_k defined by Eq. (4) is even or odd, and the number of modules. When the number of modules is two, each module includes the GFBs h_k hops away from the source GFB. The number of GFBs h_k hops away from the source GFB in each module is 2 if p_k is odd, or 1 if p_k is even.

When the number of modules is more than two, two modules include the source GFB, and the other modules do not include the source GFB. As shown in Fig. 7, if p_k is odd, the modules including the source GFB do not have the GFB h_k hops away from the source node, and each module that does not include the source GFB has one GFB h_k hops away from the source GFB. In this case, $s_k(h_k) = N_k(\frac{N_k}{p_k} - 2)$. If p_k is even, the modules including the source GFB have one GFB h_k hop away from the source GFB, and each module that does not include the source GFB has two GFB h_k hops away from the source GFB. In this case, $s_k(h_k) = N_k(2(\frac{N_k}{p_k} - 2) + 2)$.

Summarizing the above discussions, $s_k(i)$ is calculated by

$$\begin{aligned}
s_k(1) &= \begin{cases} N_k(N_k - 1) & (L_k^{gfb} \geq M_k(N_k - 1)) \\ N_k L_k \prod_{i=1}^{k-1} N_i & (\text{otherwise}) \end{cases}, \\
s_k(i) &= 2N_k \frac{N_k}{p_k} \text{ for } 1 < i < h_k, \\
s_k(h_k) &= \begin{cases} N_k(N_k - 1) & (L_k^{gfb} \geq M_k(N_k - 1)) \\ N_k(\frac{N_k}{p_k} - 2) & (L_k^{gfb} < M_k(N_k - 1) \text{ and } L_k^{'gfb} > 1 \text{ and } p_k \text{ is even}) \\ N_k(2(\frac{N_k}{p_k} - 2) + 2) & (L_k^{gfb} < M_k(N_k - 1) \text{ and } L_k^{'gfb} > 0 \text{ and} p_k \text{is odd}) \\ 2N_k \frac{N_k}{p_k} & (\text{otherwise}) \end{cases}.
\end{aligned}$$

We calculate the number of flows between each layer-$k-1$ GFB pair, T_k. Hereafter, we discuss the calculation of the number of flows from the source layer-$k-1$ GFB s to the destination layer-$k-1$ GFB d in the layer-k GFB A. Since all layer-$k-1$ GFB play the same role in the layer-k GFB, the number of flows between each layer-$k-1$ GFB pair depends only on the layer of the GFB. Thus, T_k is calculated by calculating the number of flows from the source layer-$k-1$ GFB s to the destination layer-$k-1$ GFB d in the layer-k GFB A.

Flows from the layer-$k-1$ GFB s to the layer-$k-1$ GFB d include three types of the flows; (1) the flows whose source switches belong to the layer-$k-1$ GFB s and destination switches belong to the layer-$k-1$ GFB d, (2) the flows whose destination switches belong to the layer-$k-1$ GFB d but source switches do not belong to the layer-k GFB A, (3) the flows whose source switches belong to the layer-$k-1$ GFB s but destination switches do not belong to the layer-k GFB A, and (4) the flows whose source and destination switches do not belong to the layer-k GFB A.

T_k is calculated by

$$T_k = T_k^{in} + T_k^{inward} + T_k^{outward} + T_k^{through} \tag{8}$$

where T_k^{in} is the number of flows passing between each layer$k-1$ GFB pair whose source switch belong to the layer-$k-1$ GFB s and destination switches belong to the layer-$k-1$ GFB d, T_k^{inward}

is that whose destination switches belong to the layer-$k-1$ GFB d but source switches do not belong to the layer-k GFB A, $T_k^{outward}$ is that whose source switches belong to the layer-$k-1$ GFB s but destination switches do not belong to the layer-k GFB A, and $T_k^{through}$ is that whose source and destination switches do not belong to the layer-k GFB A.

T_k^{in} is calculated by the product of the number of switches included in the layer-$k-1$ GFB s and that included in the layer-$k-1$ GFB d. That is,

$$T_k^{in} = \prod_{i=1}^{k-1}(N_i)^2. \tag{9}$$

The number of flows whose source switches belong to the layer-$k-1$ GFB s and destination switches do not belong to the layer-k GFB A is calculated by the product of the number of switches in the layer-$k-1$ GFB s and that outside the layer-k GFB A. $T_k^{outward}$ is the number of flows that go out of GFB A through the layer-$k-1$ GFB d. Thus, we calculate $T_k^{outward}$ by dividing the number of flows whose source switches belong to the layer-$k-1$ GFB s destination switches do not belong to the layer-k GFB A by the number of layer-$k-1$ GFBs in the layer-k GFB A.

$$T_k^{outward} = \frac{(\prod_{i=1}^{k-1} N_i)(\prod_{i=1}^{K} N_i - \prod_{i=1}^{k} N_i)}{N_k}. \tag{10}$$

Similarly, T_k^{inward} is calculated by

$$T_k^{inward} = \frac{(\prod_{i=1}^{k-1} N_i)(\prod_{i=1}^{K} N_i - \prod_{i=1}^{k} N_i)}{N_k}. \tag{11}$$

$T_k^{through}$ is the number of flows that come from the outside of the layer-k GFB A via the layer-$k-1$ GFB s and go to the outside of the layer-k GFB A via the layer-$k-1$ GFB d. The number of flows coming from the outside of the layer-k GFB A via the layer-$k-1$ GFB s is the sum of flows on the links that connect switches in the layer-$k-1$ GFB and the switches outside the layer-k GFB A, which is calculated by $\prod_{j=1}^{k-1} N_j \sum_{i=k+1}^{K}(X_i L_i)$. We obtain the number of flows that come from the outside of the layer-k GFB A via the layer-$k-1$ GFB s and are sent to the layer-$k-1$ GFB d by dividing $\prod_{j=1}^{k-1} N_j \sum_{i=k+1}^{K}(X_i L_i)$ by the number of the layer-$k-1$ GFBs in the layer-k GFB A. $\frac{\prod_{j=1}^{k-1} N_j \sum_{i=k+1}^{K}(X_i L_i)}{N_k}$ includes the flows whose destination switches belong to the layer-$k-1$ GFB d, whose number is T_k^{inward}. Therefore, $T_k^{through}$ is calculated by

$$T_k^{through} = \frac{\prod_{j=1}^{k-1} N_j \sum_{i=k+1}^{K}(X_i L_i)}{N_k} - T_k^{inward}. \tag{12}$$

Because all switches belong to the top-layer GFB, T_K^{inward}, $T_K^{outward}$, and $T_K^{through}$ are 0, and $T_K = T_K^{in}$ where K is the number of layers of the constructed GFB. In other layers, T_k is calculated by using Eq. (8) through (12).

Finally, the number of flows on links X_k is obtained by substituting the calculated value of T_k and $s_k(i)$ to Eq. (7).

4 Virtual Network Topology Control to Achieve Low Energy Consumption

4.1 Outline

In our method, the VNT is constructed so as to minimize the number of used ports considering two kinds of requirements; bandwidths and delay between servers.

One approach to provide sufficient bandwidths between servers is to construct the VNT that can accommodate the current traffic demands between servers. However, in a data center, traffic may change within a second [19]. Thus, if the VNT is optimized for the current traffic demands, the VNT may be required to be reconfigured every second.

In our method, to avoid too frequent reconfiguration, the traffic changes in a short period are handled by the load balancing [20] over the VNT. And, we design the VNT so as to achieve sufficiently large bandwidth and small delay with small energy consumption, considering the load balancing.

In this thesis, we use one of the load balancing technique called *Valiant Load Balancing (VLB)* [20]. In the VLB, we select the intermediate nodes randomly regardless of the destination to avoid the concentration of traffic on certain links even when traffic volume of a certain node pair is large. Then, traffic is sent from the source node to the intermediate node and from the intermediate node to the destination node. By applying the VLB, the amount of traffic between each switch pair is the sum of traffic amount from and to a switch divided by the number of switches in the network. Thus, we provide sufficient bandwidth by making the number of flows passing a link less than a threshold, which is calculated by dividing the capacity of the link by the traffic amount between each switch pair calculated considering the VLB.

The delay is also hard to forecast when designing the virtual network. In this thesis, we avoid too large delay by providing enough bandwidth and making the maximum number of hops less than a threshold.

In our method, the VNT is designed by setting the parameters of the GFB so as to satisfy the above requirements. If the demands change and new switches are added in the virtual network or some of switches currently belonging to the virtual network disappear, we check whether the current VNT satisfies the requirements described above. If the current VNT does not satisfy the requirements, we reconfigure the VNT suitable to the current environment by setting the parameters of the GFB again.

In subsection 4.2, we propose a method to set the parameters of the GFB so as to satisfy the requirements. Then, in subsection 4.3, we explain a method to keep the requirements satisfied when some switches are added to the virtual network or deleted from the virtual network.

4.2 Topology Control to Satisfy the Requirements

In this subsection, we propose a method to set the parameters of the GFB so as to minimize the number of used ports and satisfy the requirements of the bandwidth and the maximum number of hops between servers. In this method, the parameters of the GFB are set by the following steps. In the following steps, the number of switches connected in the virtual network N^{all}, the threshold to the number of hops H_{max}, the sum of the maximum traffic volume from a ToR switch T^{SWfrom}, the sum of the maximum traffic volume to a ToR switch T^{SWto}, and the capacity of a virtual link B are given.

Step 1 Set the maximum number of layers K. The numbers of layers less than K are regarded as the candidates of the number of layers.

Step 2 Set the number of the GFB in each layer N_k for all candidates of the number of layers.

Step 3 Set the number of virtual links between GFBs in each layer L_k so as to make the number of hops between switches less than the threshold for all candidates.

Step 4 Check whether the GFB with the parameters set at Steps 2 and 3 can provide the sufficient bandwidth for all candidates. If all candidates can provide the sufficient bandwidth go to Step 6. Otherwise, go to Step 5.

Step 5 Modify L_k so as to provide the sufficient bandwidth for the candidates that cannot provide the sufficient bandwidth.

Step 6 Construct the topology which uses the smallest number of virtual links among the candidates.

Step 7 End.

Setting the maximum number of layers We set the maximum number of layers so as to make the maximum number of hops less than a threshold. The maximum number of hops of the layer-k GFB is minimized when $h_k = 1$ in Eq. (5) for all layers. Thus, the maximum number of layers K must satisfy the following condition to make the maximum number of hops less than a threshold $H_{\max}$.

$$H_{\max} \geq 2^K - 1 \tag{13}$$

In our method, all numbers of layers K satisfying the above condition are regarded as the candidates of the number of layers of the GFB.

Setting N_k N_k should satisfy the following conditions.

- N_k of the low-layer GFB is smaller than N_k of the high-layer GFB.

  Each layer-$k-1$ GFB has $\prod_{i=1}^{k-1} N_i$ switches and each switch has L_k links for the connections between layer-$k-1$ GFBs. Each layer-$k-1$ GFB can use $L_k \prod_{i=1}^{k-1} N_i$ links to connect it to other layer-$k-1$ GFBs. Considering the case that L_k is minimized to achieve low energy consumption, the low-layer GFB can use only a small number of virtual links to connect it to other GFBs. Thus, if we set N_k to a large value in the low layer, the number of hops between the GFBs becomes large. On the other hand, the high-layer GFB can use more links than the low-layer GFB, and we can keep the number of hops small even when we set N_k to a large value.

- The GFB can connect more than N^{all} switches.

  The number of switches that can be connected in the layer-K GFB $\prod_{i=1}^{K} N_i$ should be larger than N^{all} to connect all switches required to be connected.

One approach to set N_k so as to satisfy the above conditions is to set N_k by

$$N_k = \prod_{i=1}^{k-1} N_i + 1, \tag{14}$$

and set N_1 so as to satisfy

$$N_{\text{all}} \leq \prod_{i=1}^{K} N_i. \tag{15}$$

Eq. (14) sets the number of GFBs in each layer to the similar manner to the DCell [7].

Setting N_k by Eq. (14), the increment of N_1 by 1 may dramatically increase the number of switches that can be connected in the layer-K GFB. Thus, it is possible that the number of switches that can be connected in the GFB whose N_1 is set to n_1 is much larger than N^{all} though the GFB whose N_1 is set to a value $n_1 - 1$ cannot connect N^{all} switches.

In our method, N_k is set by the following equation instead of Eq. (14).

$$N_k = \frac{\prod_{i=1}^{k-1} N_i}{d} + 1, \tag{16}$$

where d is one of the divisors of $\prod_{i=1}^{k-1} N_i$. By setting N_k by Eq. (16), we avoid the number of switches to be connected in the GFB becoming too large.

Setting L_k to satisfy the requirements of the maximum number of hops In our method, we initially set L_1 to 2 and L_k to 1 for $k > 1$, which are the minimum values of L_k required to connect all switches. Then, we check whether the maximum number of hops calculated by Eq. (5) is less than H_{max}. If the maximum number of hops calculated by Eq. (5) is larger than H_{max}, we modify L_k to make the maximum number of hops less than H_{max}.

Setting N_k by Eq. (16), all layer-$k-1$ GFB pairs in the layer-k GFB are connected if $k > 1$ even when $L_k = 1$. The increment of L_k for $k > 1$ cannot reduce the maximum number of hops. Therefore, if the maximum number of hops calculated by Eq. (5) is larger than H_{max}, we increment L_1 until the maximum number of hops calculated by Eq. (5) becomes less than H_{max}.

Setting L_k to provide sufficient bandwidth In our method, to avoid too frequent reconfiguration, the traffic changes in a short period are handled by the VLB [20] over the VNT. In the VLB, the traffic is sent via the intermediate switch that is randomly selected regardless of the destination. That is, the amount of traffic between each switch pair T^{SWpair} satisfies the following equation.

$$T^{SWpair} \leq \frac{T^{SWfrom} + T^{SWto}}{N^{all}} \tag{17}$$

We define $\hat{T}^{SWpair} = \frac{T^{SWfrom} + T^{SWto}}{N^{all}}$. We provide the sufficient bandwidth by making the number of flows passing each link less than $\frac{B}{\hat{T}^{SWpair}}$.

The number of flows passing a link in each layer is calculated by Eq. (7). In our method, we check whether X_k calculated by Eq. (7) is less than $\frac{B}{\hat{T}^{SWpair}}$ for each layer k. If X_k is larger than $\frac{B}{\hat{T}^{SWpair}}$, we increment L_k until X_k becomes smaller than $\frac{B}{\hat{T}^{SWpair}}$ to provide sufficient bandwidth.

4.3 Topology Control against Server Addition or Delition

When the demand for the data center increases, the additional servers may be implemented to handle the increased demand. When the demand decreases, some of the servers related to the application may be shut down to reduce energy consumption. In such cases, the number of ToR switches required to be connected to the virtual network may also change.

The VNT should satisfy the requirements even when the number of ToR switch connected to the virtual network changes. In this subsection, we describe the method to control the parameters of the GFB when the number of ToR switches connected to the virtual network changes.

Control against Server Addition We denote the number of ToR switches which can be connected to the GFB of the current parameter as $N_{\max}$. $N_{\max}$ is calculated by

$$N_{\max} = \prod_{i=1}^{K} N_i \tag{18}$$

If the number of ToR switches added to the virtual network N_{add} satisfies the following equation,

$$N_{\mathrm{add}} \leq N_{\max} - N_{\mathrm{current}}, \tag{19}$$

where N_{current} is the number of ToR switches connected to the current virtual network, we connect all additional ToR switches without changing any parameter of the GFB. In this case, we connect the additional ToR switches to the layer-$k-1$ GFB including the smallest number of ToR switches.

When N_{add} does not satisfy Eq. (21), we set the new parameters of the GFB by the method described in Section 4.2.

Control against Server Deletion If the number of ToR switches required to be connected becomes small, we remove some ToR switches from the layer-1 GFB including the largest number of ToR switches. By removing the ToR switches from the layer-1 GFB including the largest number of ToR switches, we avoid the significant difference between the numbers of ToR switches included in the layer-1 GFBs, since the difference between the numbers of ToR switches included in the layer-1 GFBs may cause the concentration of traffic on certain links.

If more than one ToR switches are removed from all layer-1 GFBs, the GFB after removing the ToR switch becomes the GFB whose N_1 is decremented. In this case, since the number of virtual links between the layer-1 GFBs is also reduced, we may not provide sufficient bandwidth between servers. Thus, the parameters of the GFB should be newly set by the method described in Section 4.2.

Unless more than one ToR switches are removed from all layer-1 GFBs, we can keep the required bandwidth and number of hops. Hereafter, we discuss the condition that we can keep the requirements satisfied without changing the parameters of the GFB.

The number of the layer-1 GFBs in the virtual network N_{GFB_1} is calculated by

$$N_{\mathrm{GFB}_1} = \prod_{i=2}^{K} N_i. \tag{20}$$

Among the layer-1 GFBs, $N_{\max} - N_{\mathrm{current}}$ layer-1 GFBs includes only $N_1 - 1$ ToR switches. Thus, if the number of deleted ToR switches N_{del} satisfies

$$N_{\mathrm{del}} \leq N_{\mathrm{GFB}_1} - (N_{\max} - N_{\mathrm{current}}), \tag{21}$$

some of the layer-1 GFBs still include N_1 ToR switches after the deletion. In this case, the parameter of N_1 of the GFB is not changed, and we also do not have to change the other parameters.

5 Evaluation

5.1 Outline of Our Evaluation

Our method connects the target number of ToR switches considering two kinds of requirements for the VNT, the bandwidth required between ToR switches and the maximum number of hops between ToR switches. The bandwidths required between ToR switches are given by the sum of the maximum volume of traffic from each ToR switch, T^{SWfrom}, and the sum of the maximum volume of traffic to each ToR switch, T^{SWto}. In all of our evaluation, we set $T^{SWfrom} = T^{SWto}$.

In the first evaluation, we check whether our method can construct the VNT satisfying the above requirements, by changing the target number of ToR switches N^{all}, the sum of traffic volume from or to each ToR switches T^{SWfrom} or T^{SWto}, and the acceptable maximum number of hops H_{max}.

Then, we investigate the number of ports of ToR switches required to achieve the requirements. In this comparison, we compare the topology constructed by our method with four topologies, FatTree, Torus, Switch-based DCell and Flattened Butterfly. Unlike the FatTree topology proposed by Al-Fares et al. [5] , we assume that the traffic is generated not only from the switches at the lowest layer but also from the switches at the upper layer in the FatTree used in this evaluation, since powering up additional switches consumes more energy. In our evaluation, the parameters of each topology are set so as to minimize the number of ports required by the topology under the constraint that it can provide the sufficient bandwidth and the maximum number of hops is less than H_{max}.

Finally, we evaluate our VNT control method under the condition that the number of ToR switches changes. In this evaluation, we add or delete a randomly selected number of ToR switches at each step. Then, we check whether our method can keep the requirements satisfied even in this case.

In all of these evaluations, we assume that the number of wavelengths on optical fibers is sufficient. We set the bandwidth of one link to 10 Gbps.

5.2 Properties of the Constructed Topology

In this subsection, we first check whether our method can construct the VNT satisfying the requirements. In this evaluation, we change the requirements for the VNT by changing the target number of ToR switches N^{all}, the sum of traffic volume from or to each ToR switches T^{SWfrom} or T^{SWto}, and the acceptable maximum number of hops H_{max}. Table 1 shows the requirements for the VNT used in our evaluation and the parameters of the GFB set by our method for each requirement.

Table 2 shows the performance of the VNT constructed by our method. As shown in this table, the VNT constructed by our method satisfies the requirements in all cases; the maximum number of hops in the constructed VNT is smaller than the required value, and the maximum link utilization is less than 1.0.

Table 1: The requirements for the VNT and parameters of the GFB set by our method for the requirements

Requirements			The parameter of the GFB calculated by our method						
Maximum number of hops	Maximum volume of traffic from each ToR switch	Number of ToR switches	Number of layers	N_1	N_2	N_3	L_1	L_2	L_3
11	1Gbps	930	3	5	6	31	2	1	1
11	3Gbps	930	3	5	6	31	4	1	1
5	3Gbps	930	2	30	31	-	11	1	-
11	1Gbps	420	2	20	21	-	3	1	-
11	3Gbps	420	3	4	5	21	3	1	1
7	5Gbps	420	2	20	21	-	5	1	-
5	5Gbps	420	2	20	21	-	11	1	-

5.3 Comparison with Existing Data Center Network Topologies

We compare the VNT constructed by our method with the existing data center network topologies. In this comparison, all topologies include 420 ToR switches. We compare the number of virtual inks per ToR switch required to achieve the requirements by changing the sum of traffic volume from or to each ToR switches. In this comparison, we set the acceptable maximum number of hops to a sufficiently large value. That is, only the bandwidth provided for each ToR switch is the only requirement for the VNT.

Figure 9 shows the results. In this figure, the horizontal axis indicates the sum of traffic

Table 2: Properties of the constructed GFB

Requirements			Performance of the constructed GFB			
Maximum number of hops	Maximum volume of traffic from each ToR switch	Number of ToR switches	Maximum number of hops	Maximum link utilization	Number of used ports per ToR switch	Number of used links
11	1Gbps	930	11	0.443	4	3720
11	3Gbps	930	7	0.915	6	5780
5	3Gbps	930	5	0.581	12	11160
11	1Gbps	420	11	0.390	4	1680
11	3Gbps	420	7	0.937	5	2100
7	5Gbps	420	7	0.952	6	2520
5	5Gbps	420	5	0.952	12	5040

volume from or to each ToR switches that is required to be accommodated, and the vertical axis indicates the number of virtual links per ToR switch required to satisfy the requirement. As shown in this figure, the switch-based DCell cannot accommodate traffic more than 1 Gbps, and the FatTree and the torus cannot accommodate traffic more than 6 Gbps per ToR switch. In the switch-based DCell, the link between level-0 DCells becomes bottleneck, which cannot be solved by the parameter settings. In the FatTree topologies, we cannot construct the topology having more links than the 3-layer FatTree. Thus, the FatTree cannot accommodate more traffic that cannot be accommodated in the 3-layer FatTree. Similarly, the torus cannot accommodate more traffic that cannot be accommodated in the torus whose dimension is the largest among the torus constructed by 420 switches.

Though the flattened butterfly can accommodate a large amount of traffic, it requires a large number of virtual links. This figure indicates that our method uses the smallest number of virtual links to accommodate traffic regardless of the amount of traffic. This is because our method to set parameters of the GFB adds only links that are necessary to accommodate the traffic. Therefore, the topology constructed by our method satisfies the requirement of the bandwidth with the smallest energy consumption.

We also compare the number of virtual links per ToR switch required to achieve the requirements by changing the acceptable maximum number of hops. In this comparison, we assume that the capacity of each virtual link is sufficient. That is, only the acceptable maximum number of hops is the only requirement for the VNT.

Figure 10 shows the results. In this figure, the horizontal axis indicates the maximum number of hops, and the vertical axis indicates the number of virtual links per ToR switch required to satisfy the requirement. As shown in this figure, in most of cases of the acceptable maximum number of hops, the topology constructed by our method uses the smallest number of virtual links to satisfy the requirements.

However, in some cases, the numbers of links used by the FatTree or the flattened butterfly are smaller than that of our method. In our method, if the acceptable maximum number of hops cannot be achieved by the layer-k GFB, we construct the layer-$k-1$ GFB whose N_k is set by Eq. (16). In some cases, we cannot construct the layer-$k-1$ GFB whose maximum number of hops equal the required value. Thus, the constructed layer-$k-1$ GFB achieves the maximum number of hops much smaller than the required value, but may require more links. However, we may be able to modify our method so as to minimize the number of required links even in such cases, which is one of our future research topics, since the flattened butterfly can also be constructed by setting the parameters of the GFB.

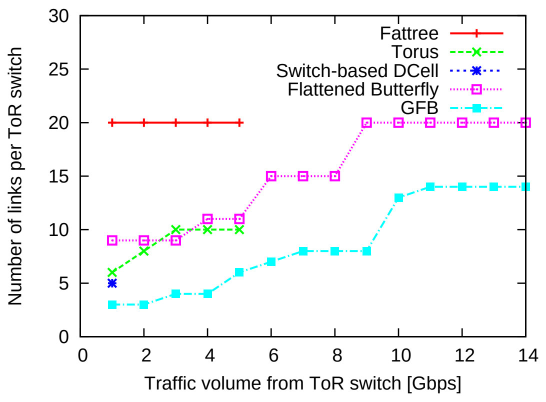

Figure 9: Number of virtual links required to accommodate the traffic from ToR switches

5.4 Performance against Addition or Deletion of Switches

The demand may change, and the new additional ToR switches may become required to be connected to the virtual network, or some of the ToR switches may be removed from the virtual

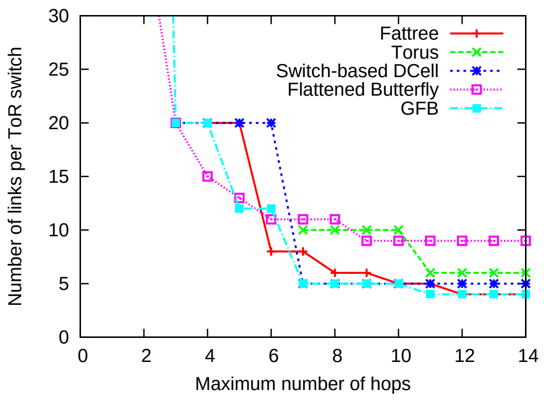

Figure 10: Number of virutal links required to make the maximum number of hops less than the target value

network. In this evaluation, we check whether our method can keep the requirements satisfied in such cases.

In this evaluation, we set the number of ToR switches initially connected to the virtual network to 930. The acceptable maximum number of hops is set to 11, and the sum of the traffic volume from each ToR switch to 3 Gbps. At each step, we add or delete a randomly selected number of ToR switches.

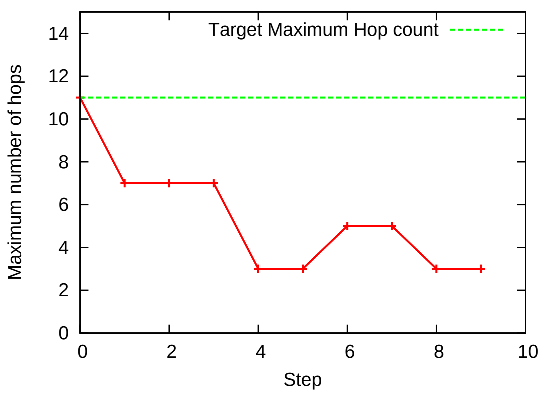

Figure 11: Number of used virtual links per ToR switch to keep the maximum number of hops less than the target value

Figures 11 and 12 show the maximum number of hops and the maximum link utilization at

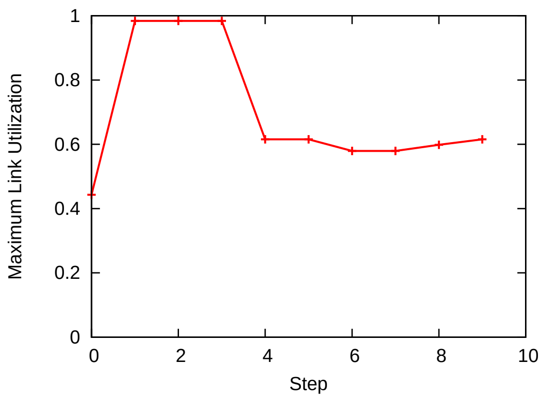

Figure 12: Number of used virtual links per ToR switch to keep sufficient bandwidth

each step respectively. In these figures, the horizontal axis is the number of steps, and the vertical axis is the maximum number of hops or the maximum link utilization. As shown in these figures, our method keeps the requirements satisfied even when some ToR switches are added to or deleted from the virtual network.

As shown in these figures, the maximum number of hops becomes much smaller than the acceptable number of hops as steps pass. The maximum link utilization is also much smaller than 1. This is because we do not change the parameters of the GFB if the requirements of the provided bandwidth and the maximum number of hops are satisfied. However, re-calculating the parameters of the GFB may decrease the energy consumption. A method to determine whether the re-calculation of the parameters of the GFB is required considering the energy consumption is one of our future research topics.

6 Conclusion

In this thesis, we introduced the virtual network configured over the data center network constructed of the OXCs and the electronic switches. In this network, the core of the data center network is constructed by using the OXCs and optical fibers. Then, the electronic switches, deployed in each server rack, are connected to the core network by connecting them to OXCs. We construct the VNT by establishing lightpaths between the electronic switches. In our method, we minimize the energy consumption of the data center network by minimizing the number of ports of electronic switches used in the VNT and shutting down the unused ports.

In this thesis, we also proposed a method to reconfigure the VNT suitable for a data center network. In this method, the traffic changes in a short period are handled by the load balancing over the VNT. We design the VNT so as to achieve sufficiently large bandwidth and small delay with small energy consumption, considering the load balancing. Then, if the current VNT is not suitable for the current demands, the VNT is reconfigured. Our method reconfigures the VNT by setting parameters of a topology so as to avoid large calculation time in a data center. As the topology used in the VNT configuration, we proposed the topology called *Generalized Flattened Butterfly (GFB)*, and a method to set its parameters so as to suit the current condition.

We evaluated our method, and clarified that our method can construct the VNT satisfying the requirements of the bandwidth and the maximum number of hops between the switches, by setting parameters of the GFB. In addition, the evaluation results indicate that the topology constructed by our method uses a smaller number of links to satisfy the requirements than the existing data center network topologies; the number of links required by our method is about a half of the number of links required by the flattened butterfly to provide sufficient bandwidth.

One of our future research topics is to modify the method to set the parameters of the GFB. As discussed in Section 5, there are some cases that the topology constructed by our method uses more links than the flattened butterfly to make the maximum number of hops less than a required value. Because the flattened butterfly can also be constructed by setting the parameters of the GFB, we may be able to modify our method so that the topology constructed by our method uses a smaller number of links in all cases.

Another future research topic is a method to determine whether the re-calculation of the parameters of the GFB is required considering the energy consumption. In the method proposed in this thesis, we re-calculate the parameters only when the requirements of the bandwidths and the maximum number of hops become unsatisfied. By determining whether the re-calculation of the parameters of the GFB is required considering the energy consumption, we can reduce further energy consumption.

Acknowledgments

I sincerely would like to appreciate to my supervisor, Professor Masayuki Murata of Osaka University. for his valuable comments, insights, and continuous encouragement.

I sincerely would like to appreciate supports of assistant professor Yuichi Ohsita of Osaka University. My works would not have been possible without his supports.

And also I would like to express my sincere appreciation to Professor Naoki Wakamiya and Associate professor Shin'ichi Arakawa for many advices.

Finally, I much thank the members in Murata laboratory. Though I cannot write here about all people who support me, I thank them very much.

References

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

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

[3] “Apache hadoop project.” `http://hadoop.apache.org/`.

[4] 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.

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

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

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

[8] 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,” in *Proceedings of ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 63–74, Aug. 2009.

[9] 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.

[10] 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.

[11] Y. Liao, D. Yin, and L. Gao, “Dpillar: scalable dual-port server interconnection for data center networks,” in *Proceedings of ICCCN*, pp. 1–6, Aug. 2010.

[12] 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,” in *Proceedings of ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 51–62, Aug. 2009.

[13] 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,” in *Proceedings of ACM SIGCOMM Computer Communication Review*, vol. 39, pp. 39–50, Aug. 2009.

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

[15] Y. Ohsita, T. Miyamura, S. Arakawa, S. Ata, E. Oki, K. Shiomoto, and M. Murata, “Gradually reconfiguring virtual network topologies based on estimated traffic matrices,” *IEEE/ACM Transactions on Networking*, vol. 18, pp. 177–189, Feb. 2010.

[16] M. Zhang, C. Yi, B. Liu, and B. Zhang, “GreenTE: power-aware traffic engineering,” in *Proceedings of ICNP*, pp. 21–30, Oct. 2010.

[17] 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.

[18] N. Farrington, G. Porter, S. Radhakrishnan, H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat, “Helios: a hybrid electrical/optical switch architecture for modular data centers,” in *Proceedings of ACM SIGCOMM Computer Communication Review*, pp. 339–350, Oct. 2010.

[19] T. Benson, A. Anand, A. Akella, and M. Zhang, “MicroTE: Fine Grained Traffic Engineering for Data Centers,” in *Proceedings of ACM CoNEXT*, pp. 1–12, Dec. 2011.

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