Master's Thesis

Title

Preconfiguring Robust Logical Topologies with Autonomous Routing for Data Center Networks

Supervisor

Professor Masayuki Murata

Author

Yuta Shimotsuma

February 10th, 2014

Department of Information Networking

Graduate School of Information Science and Technology

Osaka University

Master's Thesis

Preconfiguring Robust Logical Topologies with Autonomous Routing for Data Center Networks

Yuta Shimotsuma

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. In data centers, servers handle a large amount of data by cooperating with each other. Because the lack of the bandwidth between cooperating server pairs degrades the cooperation between servers, sufficient bandwidth between servers should be provided.

However, the sufficient bandwidth between servers cannot be provided when the routes within a data center are fixed. The traffic patterns between servers vary according to the data and change drastically within 1 sec. The routes should be configured dynamically to follow such traffic changes so that a large number of traffic flows of the communicating server pairs do not concentrate on certain links. Moreover, failures are also common in data centers. If failures occur, the routes that do not include the failed links or devices should be configured.

One approach to providing sufficient bandwidth between servers even in such cases is autonomous routing. In this thesis, we propose an autonomous routing method that can handle both of traffic changes and failures. In this method, we use an approach called *multi-topology routing* (MTR). In the MTR, multiple logical topologies, which are the subsets of the physical network topology, are preconfigured. When a node detects a failure, the node autonomously selects the logical topology that does not include the failed link or node, and relays the traffic using the selected logical topology by attaching the information of the selected logical topology to the header of the packets.

In this thesis, we apply the MTR not only to handling the failures but also to avoiding the congestion even when traffic changes occur. In our method, each switch autonomously selects the logical topology based on the availability and the utilizations of the links connected to the switch. By considering the utilizations of the links, we avoid the concentration of the traffic on a certain link. Even if failure or congestion occurs, the switch that detects it relays the packets so as not to pass the unavailable links by eliminating the logical topologies, where unavailable links are on the path to the destination, from the candidate logical topologies. Then, the information on the eliminated logical topologies is attached to the header of the packet to inform the unavailable logical topologies, and the other switches autonomously select the logical topology among the available logical topologies.

In this thesis, we also propose a method to construct the logical topologies for our autonomous routing. In this method, the set of logical topologies is constructed so as to provide sufficient bandwidth between all communicating server pairs.

We evaluate our routing method combined with the method to construct the logical topologies via simulation. The results show that our methods provide larger bandwidth between any server pairs than the ECMP even when failure occurs.

Keywords

data center

logical topologies

autonomous routing

Contents

List of Figures

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. In data centers, servers handle a large amount of data by cooperating with each other. Because the lack of the bandwidth between cooperating server pairs degrades the cooperation between servers, sufficient bandwidth between servers should be provided.

However, the sufficient bandwidth between servers cannot be provided when the routes within a data center is fixed. The traffic patterns between servers vary according to the data and change drastically within 1 sec [1–3]. The routes should be configured dynamically to follow such traffic changes so that a large number of traffic flows of the communicating server pairs do not concentrate on certain links. Moreover, failures are also common in data centers [4, 5]. If failures occur, the routes that do not include the failed links or devices should be configured.

There are many researches on the methods to control routes within a data center dynamically [3, 6]. In these methods, a central controller is deployed. The central controller periodically collects the traffic informationand network availability information. Then, it determines the routes of traffic within a data center. Since traffic pattern changes within 1 sec, the control interval must be less than 1 sec to follow such frequent changes in traffic. However, collecting traffic information in a short-interval takes a large overhead. Moreover it is difficult to calculate optimal routes within a short interval.

One approach to providing sufficient bandwidth between servers even in such cases is autonomous routing. In this approach, each switch determines routes autonomously based on the local traffic information and link availability information observed by it. Because neither collecting a large amount of information of whole network nor calculating the optimal routes of all server pairs is required, this approach can change the routes within a sufficiently small interval to follow the changes in traffic in a data center. A. Greenberg et al. proposed a method to distribute load for a data center topology called FatTree [7]. In this method, each switch randomly selects the next switch to avoid the concentration of traffic on a certain link. However, this method does not consider the failures, and causes traffic concentration on a certain link when some links are unavailable. In this thesis, we propose an autonomous routing method that can handle both of traffic changes and failures. In this method, we use an approach called *multi-topology routing*

(MTR) [8, 9]. In the MTR, multiple logical topologies, which are the subsets of the physical network topology, are preconfigured. When a node detects a failure, the node autonomously selects the logical topology that does not include the failed link or node, and relays the traffic using the selected logical topology by attaching the information of the selected logical topology to the header of the packets.

In this thesis, we apply the MTR not only to handling the failures but also to avoiding the congestion even when traffic changes occur. In our method, each switch autonomously selects the logical topology based on the availability and the utilizations of the links connected to the switch. By considering the utilizations of the links, we avoid the concentration of the traffic on a certain link. Even if failure or congestion occurs, the switch that detects it relays the packets so as not to pass the unavailable links by eliminating the logical topologies, where unavailable links are on the path to the destination, from the candidate logical topologies. Then, the information on the eliminated logical topologies is attached to the header of the packet to inform the unavailable logical topologies, and the other switches autonomously select the logical topology among the available logical topologies.

In this thesis, we also propose a method to construct the logical topologies for our autonomous routing. In this method, the set of logical topologies is constructed so as to provide sufficient bandwidth between all communicating server pairs.

We evaluate our routing method combined with the method to construct the logical topologies via simulation. The results show that our methods provide sufficiently large bandwidth between any server pairs even when failure occurs.

The rest of this thesis is organized as follows. In Section 2, we present related work of data center networks and autonomous routing based on multiple logical topologies. In Section 3, we propose the autonomous routing method based on multiple logical topologies in a data center. In Section 4, we also propose the design method of logical topologies for routing in a data center. Then, we evaluate our method and clarify that our method can provide sufficient bandwidth between all servers in Section 5. Finally, Section 6 provides a conclusion.

2 Related Work

2.1 Data Center Networks

As the online services handing a large amount of data have become popular, data centers handling such a large amount of data becomes important. In a data center, the network plays an important role, because a large amount of data is handled by the cooperation between the servers via the network. Thus, many researches on the data center network, including the network structures, routing strategies, transport protocols and so on, have been done [10–14].

Among the researches on the data center, the routing is important to provide sufficient bandwidth to the communicating server pairs; the routes should be configured so as to avoid the concentration of traffic between communicating server pairs on certain links based on the current traffic. The traditional data center uses the Spanning Tree Protocol to configure the routes within a data center. The Spanning Tree protocol constructs the tree topology by using the subset of the links and switches. By using the spanning tree, the loop-free paths are set between all server pairs. However, the Spanning Tree Protocol does not use the links included in the constructed tree topologies, and does not provide a large bandwidth.

Another famous routing protocol used in the data center is TRILL (Transparent Interconnection of Lots of Links) [15]. In the TRILL, the routes are set by the *equal-cost-multi-path* (ECMP) strategy [16] ; the traffic are distributed among the multiple shortest paths. By using the multiple shortest paths, the TRILL provides much larger bandwidths than the Spanning Tree Protocol. However, even in case of the TRILL using the ECMP strategy, it is difficult to provide a large bandwidth if some links become unavailable due to failures or temporal congestion, because it does not use the routes whose path lengths are larger than the shortest paths.

Therefore, the methods to configure the routes suitable to current traffic have been discussed. In these methods, a central server which calculates the routes in a data center is deployed. The central server collects the information on traffic in a data center and availability of the devices or links in a data center. Then, the server calculates the routes based on the collected information. Finally, the calculated routes are set by the technologies such as OpenFlow.

However, the central control is not sufficient to configure the routes suitable to the current traffic, because the traffic in a data center changes frequently. S. Kandula et al. explored the nature of traffic in data centers [2], and showed that data center traffic is bursty and is difficult to predict. T. Benson et al. also examined the change in traffic exchanged between each pair of server racks and found that the reconfiguration of the routes should be performed at less than 1 second interval to provide sufficient bandwidth between the communicating server pairs. The routing reconfiguration at less than 1 second interval takes a large overhead to collect traffic information of a whole network at short interval. Moreover, it is difficult to calculate the routes in a short time.

One approach to handle such frequent changes in traffic is the autonomous routing. In this approach, each switch determines routes autonomously based on the local traffic information and link availability information observed by it. Because neither collecting a large amount of information of whole network nor calculating the optimal routes of all server pairs is required, this approach can change the routes within a sufficiently small interval to follow the changes in traffic in a data center. A. Greenberg et al. proposed a method to distribute load for a data center topology called FatTree [7]. In this method, each switch randomly selects the next switch to avoid the concentration of traffic on a certain link. However, this method does not consider the failures, and causes traffic concentration on a certain link when some links are unavailable.

Therefore, in this thesis, we propose an autonomous routing method that can handle both of traffic changes and failures.

2.2 Autonomous Routing Based on Multiple Logical Topologies

The multi topology routing (MTR) is one of the autonomous routing strategies which aim to keep the communication even when failure occurs. In the MTR, the multiple logical topologies, which includes all nodes and a subset of links in the physical network, is preconfigured. In case of failures, a node nearest to the failed link or failed node detects the failure. The node selects the logical topology in which the failed link or failed node is not used to relay the traffic. Then, it relays the traffic via the selected logical topology by attaching the ID of the selected logical topology to the header of the packet.

There are several researches on the MTR [8, 9, 17]. F. Hansen et al. presented an approach to constructing a set of a small number of logical topologies that can handle any single failures in the network. In this method, each logical topology is constructed as a tree structure. The set of the logical topologies are configured so that each node becomes a leaf node in at least one of the logical topologies.

S. Steven et al. proposed the method to achieve both of power saving and fault tolerance against any single link failure [17]. This method constructs two logical topologies. First one is constructed so as to minimize the energy consumption under the constraint that all traffic can be accommodated. The other logical topology is constructed so as to keep the routes between any node pairs even when a link in the first logical topology fails.

M. C. Scheffel et al. formulated an optimization problem to design the set of the logical topologies so as to be robust to any single failures without increasing the path length [9].

3 Autonomous Routing Based on Multiple Logical Topologies in a data center

In this thesis, we propose an autonomous routing method that provides a large bandwidth between communicating servers even in case of frequent traffic changes or failures. This method is based on the MTR; we preconfigure multiple logical topologies for routing. The routes are determined by each switch autonomously selecting one of the logical topologies.

3.1 Behavior of Each Switch

In this approach, each switch selects one of the logical topology, which is used to forward the packet, by the following steps.

1. Obtain the set of the available logical topologies.
2. Select the logical topologies where the number of hops to destination node is the shortest among the available logical topologies. We denote the selected logical topologies by *the candidate logical topologies*.
3. Select a logical topology where the utilization of the link connected to the node is the smallest among the candidate logical topologies.

We explain the details of each step below.

3.1.1 Obtaining the Set of the Available Logical Topologies

There are two cases that each switch obtains the information on the unavailability of the logical topologies. The first one is the case that the switch itself detects the unavailability. Each switch detects the failure or congestion that occurs at the links connected to it. When a switch detect such a unavailable link, a logical topology is regarded as unavailable by the switch if the unavailable link is used to relay the traffic to the destination in the logical topology. When there are the logical topologies regarded as unavailable, the switch attaches the information on the unavailable logical topologies to the packet header, and relays the packet. The other case that the switch knows the unavailability is to look the packet header where other switches attach the information on the unavailable logical topologies.

After obtaining the unavailable logical topologies, we obtain the set of available logical topologies by eliminating the unavailable logical topologies from all logical topologies.

3.1.2 Selecting the Candidate Logical Topologies

In this approach, we select the logical topology where the number of hops to the destination is the shortest among the available logical topologies. By selecting the logical topologies where the number of hops is the shortest, we avoid a large number of hops which uses bandwidths of a large number of links.

3.1.3 Selecting the Logical Topology

The switch selects the utilization of the link connected to the node is the smallest among the candidate logical topologies. The utilization of the link is obtained from the traffic information or queue length of the each port of the switch. By selecting the logical topology in this way, we balance the loads among links connected to the switch.

3.2 Embedding the Information on Availability of Logical Topologies

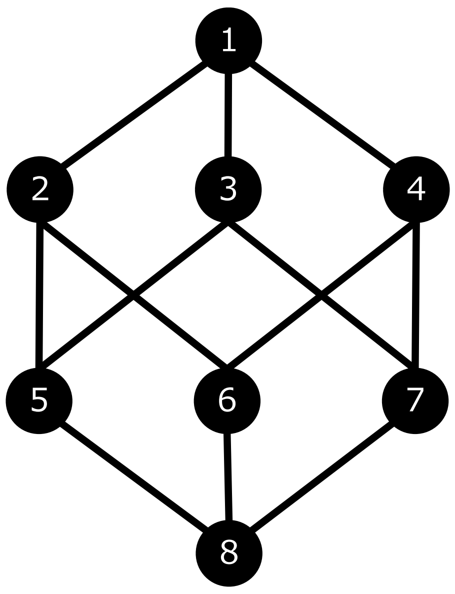
The information on availability of the logical topologies can be represented by n bits, where each bit indicates the availability of each logical topology. By embedding this information to the packet headers, the switches obtain the information on the availability of the logical topologies.

One approach to embedding the information on availability of the logical topologies is to use the destination address field in the IPv6 header; we use n bits of the 128-bit address field to embed the availability information. As shown in Section 5, even a small number of logical topologies increase the bandwidth provided between servers. Therefore, n is small enough to be embedded to the destination address field.

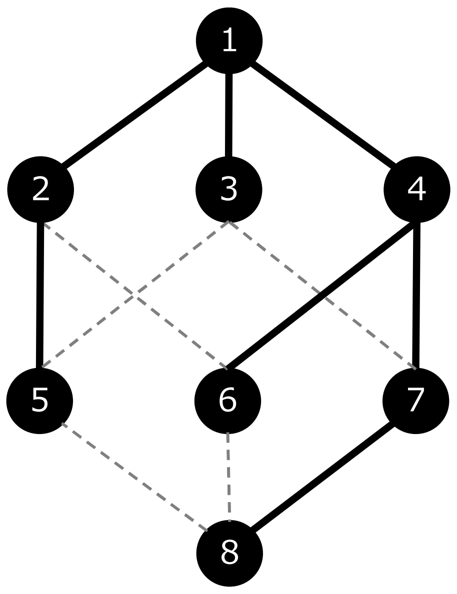
3.3 Example of Routes Set by Our Method

Figure 1 shows an example of a physical topology, consisting of 8 nodes and 12 links, and logical topologies constructed over the physical topology. Fig. 2 shows an example of the routes configured by our method. In Figure 2(a), since the load of any links is low and all links is regarded as available, a packet from node 2 to node 7 is transmitted via the shortest path, $2 \rightarrow 1 \rightarrow 4 \rightarrow 7$.

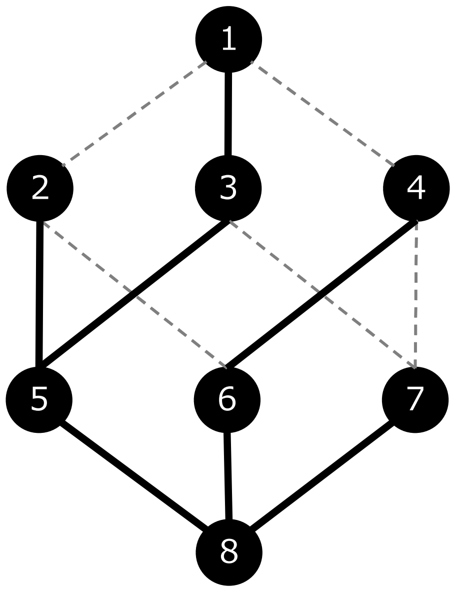
On the other hand, when the link between node 4 and node 7 cannot used because of temporal high load, as shown Fig. 2(b), node 4 attaches the information of the unavailable logical topology A and C to packet, and transmits the packet to node 6 with logical topology B. Node 6 looks the available information of the logical topologies in the packet headers. Then it relays the packet without using the logical topologies A and C. As a result, the packets are sent to the destination without using the unavailable links.



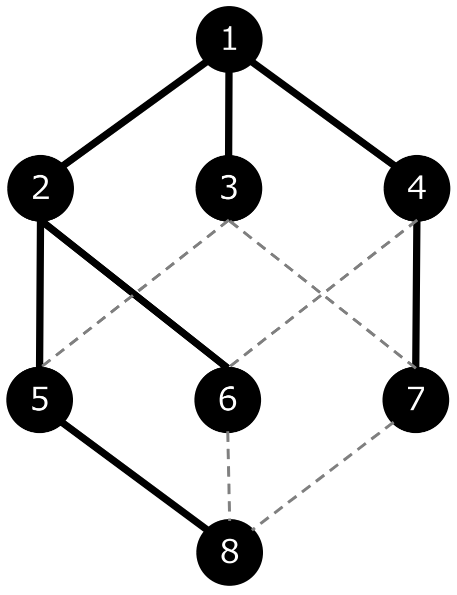
(a) pysical topology



(b) logical A

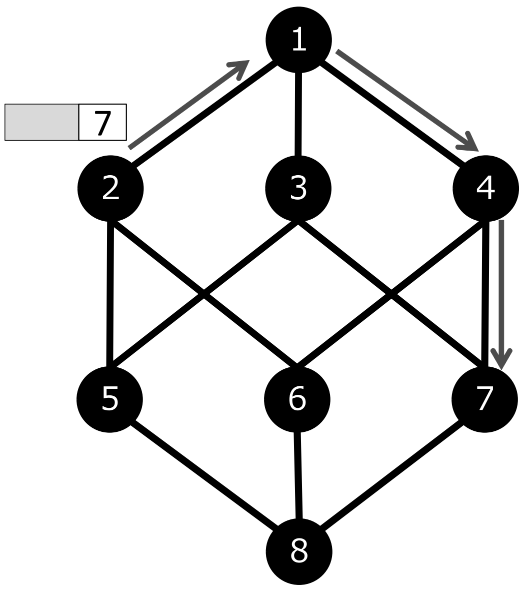


(c) logical topology B

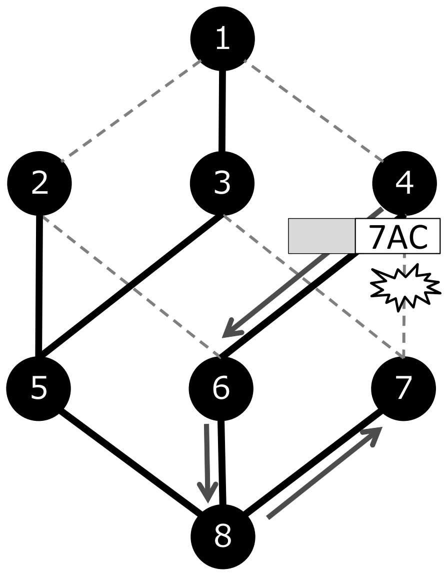


(d) logical topology C

Figure 1: An example physical topology and logical topologies



(a) In case of low load of all links



(b) In case of high load of link between node 4 and node 7

Figure 2: An example of the traffic transmission with the logical topologies

4 Design of Logical Topologies for Routing in a Data Center

In our routing method, the bandwidth provided between server pairs depends on the set of the logical topologies; if the set of logical topologies does not include sufficient number of paths between a server pair, the sufficient bandwidth cannot be provided to the server pair when some links on the path congest or fail. Moreover, if many logical topologies use the same link, the traffic may concentrate on the link.

In this section, we propose a method to design the set of the logical topologies for routing so as to provide a large bandwidth between servers even when failure occurs.

4.1 Requirements of Logical Topologies

To provide a large bandwidth between servers, the set of logical topologies should satisfy the following requirements. (1) Sufficient number of disjoint paths should exist between any server pairs. By using the set of logical topologies with sufficient number of disjoint paths, we can find the route between servers that do not pass the congested or failed links if many links are congested or fail. (2) None of the node should have a large impact on the bandwidth provided between servers. If certain nodes relay a large number of flows in many logical topologies, the failures of the links may significantly decrease the bandwidth provided between servers. (3) The concentrate of traffic on a certain link should be avoided even when failure occurs.

The rest of this subsection, we discuss the metrics indicating the above requirements.

4.1.1 Number of Disjoint Paths

The number of disjoint paths for a server pair is obtained by calculating all routes between the server pair on all logical topologies and counting the number of paths that does not share any links with each other.

4.1.2 Impact of Each Node

The node that relays a large amount of traffic has a large impact on the provided bandwidth between servers if it fails. We use the average of the node betweenness centrality as the metric indicating the impact of the node. We define the average of the node betweenness centrality of node n as

$$B_n^{\mathrm{node}} = \frac{1}{|G|} \sum_{g \in G} \left(\sum_{s,d \in S} \frac{R_g^{\mathrm{node}}(s,n,d)}{R_g(s,d)} \right)$$

where G is the set of logical topologies, S is the set of nodes, $R_g(s,d)$ is the number of shortest paths between nodes s and d on the logical topology g, and $R_g^{\mathrm{node}}(s,n,d)$ is the number of shortest paths between nodes s and d that passes the node n on the logical topology g.

The average of the node betweenness centrality indicates the expected amount of traffic passing the node when the traffic with size 1 are generated between all node pairs and the logical topology used to send the traffic is selected randomly. The node with the large average of the betweenness centrality is likely to relay a large amount of traffic. That is, if the node fails, a large amount of traffic should be relayed via another path that does not include the nodes, which may cause the congestion on the other links. Therefore, we should construct the set of the logical topologies so that the average of the node betweenness centrality is small.

4.1.3 Concentration of Traffic on Certain Links

We use the average of the link betweenness centrality as the metric indicating the possibility of the concentration of traffic on certain links. Similar to the average of the node betweenness centrality, the average of the link betweenness centrality of the link l is defined by

$$B_l^{\mathrm{link}} = \frac{1}{|G|} \sum_{g \in G} \left(\sum_{s,d \in S} \frac{R_g^{\mathrm{link}}(s,l,d)}{R_g(s,d)} \right)$$

where G is the set of logical topologies, S is the set of nodes, $R_g(s,d)$ is the number of shortest paths between nodes s and d on the logical topology g, and $R_g^{\mathrm{node}}(s,l,d)$ is the number of shortest paths between nodes s and d that passes the link l on the logical topology g.

The average of the link betweenness centrality indicates the expected amount of traffic on the link when the traffic with size 1 are generated between all node pairs and the logical topology used to send the traffic is selected randomly. The link with the large average of the link betweenness centrality is likely to be passed by a large number of flows and congested. Therefore, we should construct the set of the logical topologies so that the average of the link betweenness centrality is small.

4.2 Steps to Design Logical Topologies

In our method to design the set of logical topologies, we continue adding the suitable logical topology to the set of the logical topologies instead of obtaining the best set of the logical topologies among the all possible set of the logical topologies, because it is difficult to search the best set among the all possible set due to a large calculation time.

In our method, we design the set of the logical topologies by the following steps.

Step 1. Generate candidate logical topologies

Step 2. Select the best logical topology among the candidate logical topologies and add it to the set of the designed logical topologies.

Step 3. If the number of the designed logical topologies is less than the target number, go back to Step 2. Otherwise end.

4.2.1 Generation of the Candidate Logical Topologies

Each logical topology should satisfy the following requirements. (1) The routes between all switch pairs should exist. (2) The probability that the logical topology become unavailable is low. In our method, the logical topology is regarded as unavailable if unavailable links are on the path to the destination node. However, if a large number of logical topologies become unavailable, we cannot provide sufficiently large bandwidth.

In this thesis, we use a tree topology that connects all switches as the candidate logical topologies. Because the number of links included in the tree topology is minimum among the topology that connects all switches. Therefore, if some links become unavailable, the probability that the unavailable link is included in the logical topology constructed as a tree is low.

We construct the candidate logical topologies by the following two phases. In the following explanation, we denote the number of the switches in the data center network by N and the designed set of the logical topologies by C.

Initial Phase We generate N logical topologies where each of the switches is included. We add the generated logical topologies to C.

Increment Phase For each of logical topologies included in C, we add the nodes n satisfying the following condition.

- n is not included in the logical topology
- n has a link to one of the nodes included in the logical topology

If n has multiple links to the nodes included in the logical topology, we adds the logical topologies, where each of the links to n is added, to C. The increment phases is repeated until all logical topologies in C includes all nodes.

4.2.2 Selection from the Candidate Logical Topologies

We calculate the metrics defined in Section 4.1 for the set of the logical topologies where one of the candidate logical topologies is added. Then, we select the best logical topology among the candidate logical topologies.

In our method, we use three metrics, the number of disjoint paths, the average of the node betweenness centrality, and the average of the link betweenness centrality. Among them, we compare the number of disjoint paths first, because the number of disjoint paths has a large impact on the probability to find an alternative route when some links are unavailable, and the alternative routes increase the bandwidth provided between server pairs. Then, if multiple candidate logical topologies have the same number of disjoint paths, we compare the average of the node betweenness centrality. Finally, if multiple candidate logical topologies also have the same average of the node betweenness centrality, we compare the average of the link betweenness centrality.

4.3 Property of Designed Logical Topologies

In this subsection, we investigate the property of the logical topologies constructed by our method. To investigate the property of the designed logical topologies, we use the HyperCube topology constructed of 8 switches with 3 ports as the physical network topology, because the HyperCube has a large number of routes, and many candidate sets of the logical topologies which is the subset of the physical topology ca be constructed over it.

We compare the maximum amount of traffic on the link achieved by our method with that achieved by the optimal routes, which are obtained by solving the optimization problem where the traffic information and available link information are given. In this comparison, the traffic is generated between all server pairs. The traffic rate between each server pair is set to 1, and the bandwidth of each link is set to a sufficiently large value. The maximum amount of traffic on the link is investigated in the following three cases; (1) the case that all links are available, (2) the case of single link failure, and (3) the case of two-link failures. For the cases of the failures, we generate all patterns of the link failures, and calculate the average and maximum of the amount of traffic on the links in all cases.

Figure 3 shows the ratio of the maximum amount of traffic achieved by our method to that achieved by the optimal routes when all links are available. As shown in Fig. 3, the maximum amount of traffic achieved by our method becomes close to that achieved by the optimal routes as the number of logical topologies increase. Our method with more than 5 logical topologies achieves the similar maximum amount of link to the optimal routes.

Figure 6 shows the probability that the routes between all server pair are found in case of single link failure or two-link failure. As shown in this figure, the probability that the routes between all server pairs exist increases as the number of the logical topology increases. Routes between all server pairs can be found by designing more than 3 logical topologies even if any single link failure occurs, and they can be found by designing more than 17 logical topologies even if any two link failure occurs.

Figures 4 and 5 show the ratio of the maximum amount of traffic on the link achieved by our method to that achieved by optimal routes in the cases of single link failures and two-link failures respectively. As shown in these figures, the maximum amount of traffic on the link achieved by our method becomes does not become as small as that achieved by the optimal routes. This is caused by that our method selects the routes autonomously; only the nodes connected to the failed links detect the failure, and the nodes that do not know the failure send the traffic along the routes including the failed links. Though the traffic is relayed via the alternative routes after the traffic arrives the node detecting the failures, this may cause a large number of hops, and increase the amount of the traffic on many links.

Though our autonomous routing may increase the amount of the traffic on the links, these figures show that the average of the ratio of the maximum amount of the traffic achieved by our method to that achieved by the optimal routes is less than 1.2. That is, our autonomous routing using our designed set of logical topologies does not significantly increase the amount of traffic

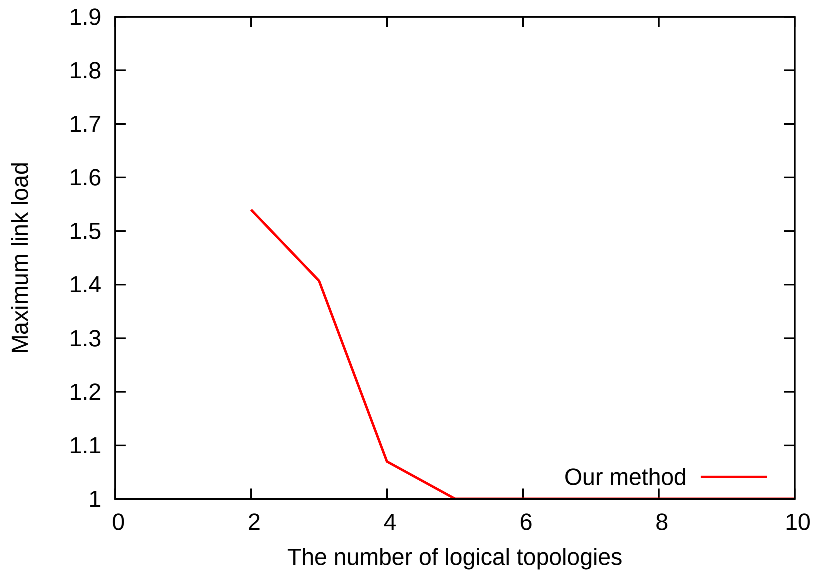


Figure 3: The maximum link load in the case that all links are available

on the link in most of cases.

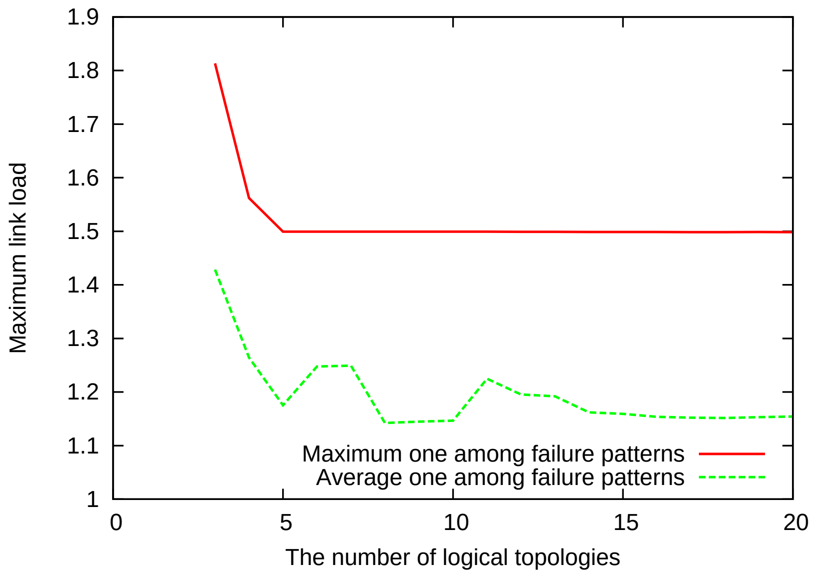


Figure 4: The maximum link load in case of single link failure

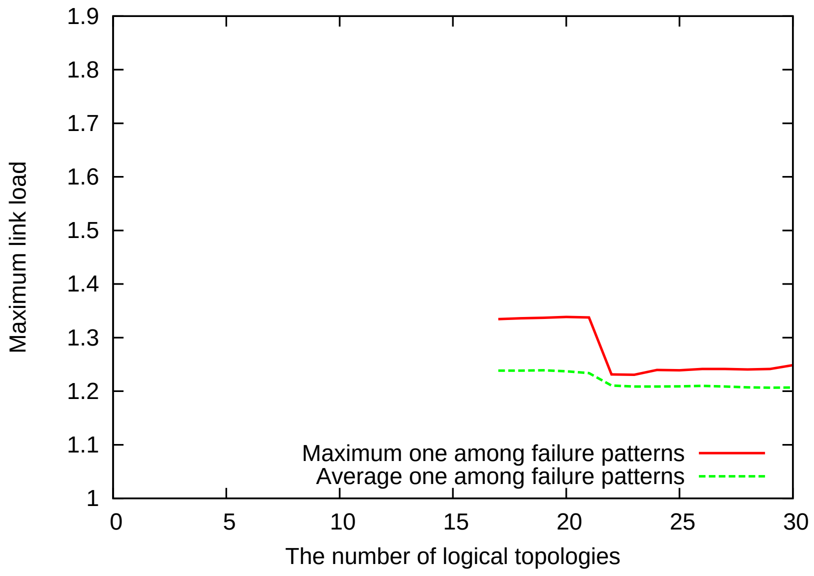


Figure 5: The maximum link load in case of two-link failures

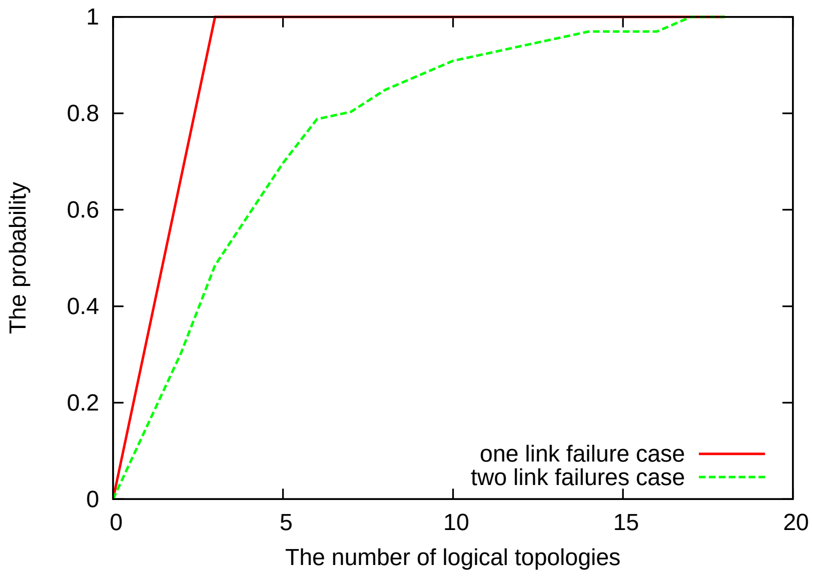


Figure 6: the probability that the routes between all server pair are found in case of single link failure or two-link failure

5 Evaluation

In this evaluation, we compare the autonomous routing using the logical topologies designed by our method with ECMP generally used in data centers.

5.1 Simulation Environment

In this evaluation, we perform each routing method in each network topology to accommodate traffic between servers. The traffic is generated between the selected server pairs. The generated traffic is incrementally accommodated along the route calculated by each method as long as the routes with sufficient bandwidth are found. Finally, we investigate the amount of traffic accommodated in the network. The amount of traffic between a server pair accommodated by the above steps indicates the bandwidth provided between the server pair.

5.1.1 Topologies Used in Our Evaluation

In our evaluation, we use two kinds of the topologies, which are popular for a data center network, FatTree [18] and Torus, as shown in Figure 8 and Figure 7.

In our evaluation, we construct the FatTree topologies by using 20 switches with four ports, and the only switches at the leaf of the FatTree are connected to the server racks. We construct the Torus topology by using the 16 switches with 4 ports, and connect all switches to the server racks. We set the bandwidth between switches to 10 Gbps.

5.1.2 Traffic

In our evaluation, we select the communicating server rack pair randomly. We set the number of communicating server pair to 20% and 10 % of the server rack pairs in the FatTree and the Torus, respectively. We generate 100 pattern of traffic.

5.1.3 Metrics

In our evaluation, we compare the smallest bandwidth among the bandwidth provided between server rack pairs. We also compare the total bandwidth which is the sum of the accommodated traffic.

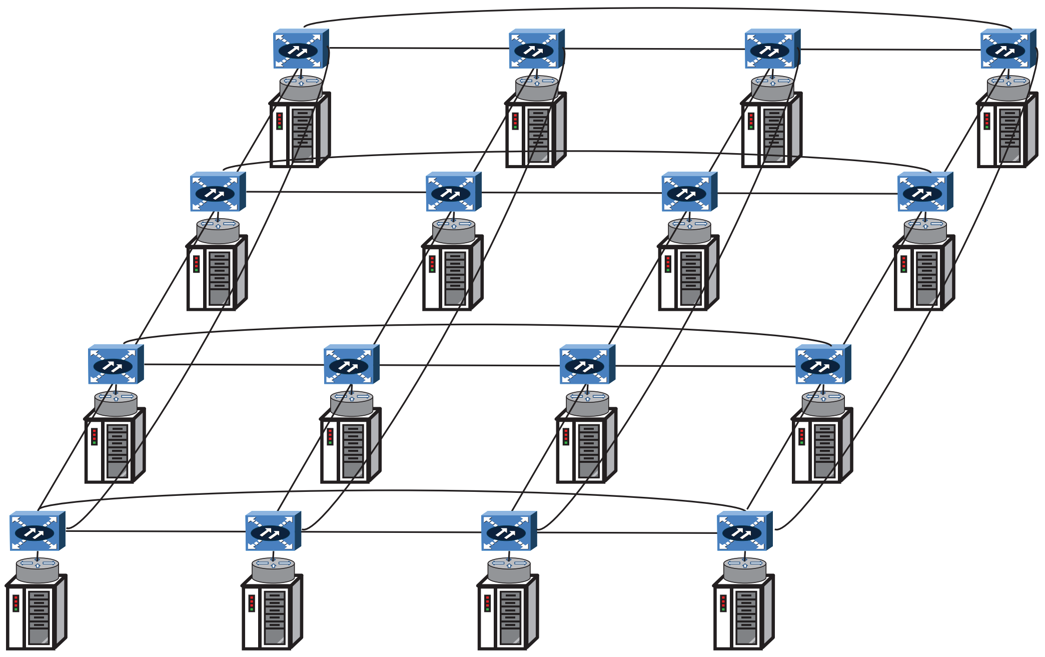


Figure 7: Torus

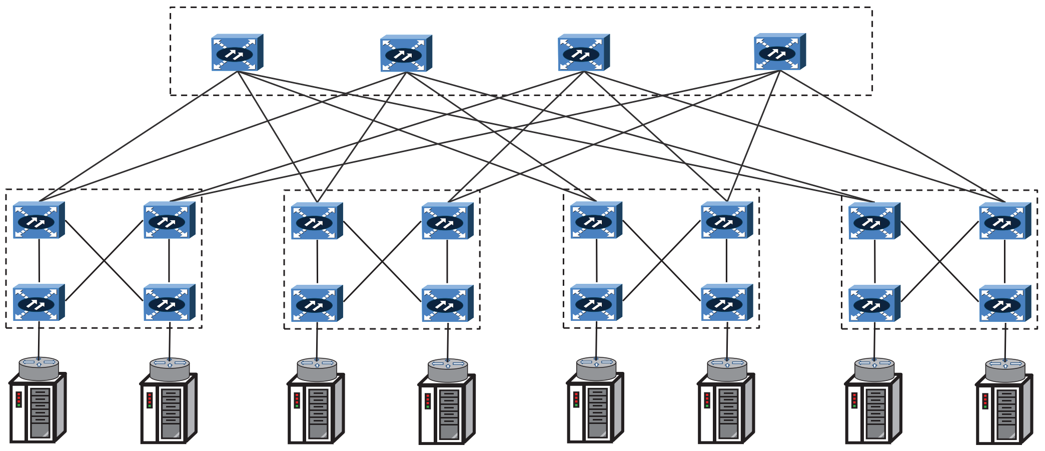


Figure 8: FatTree

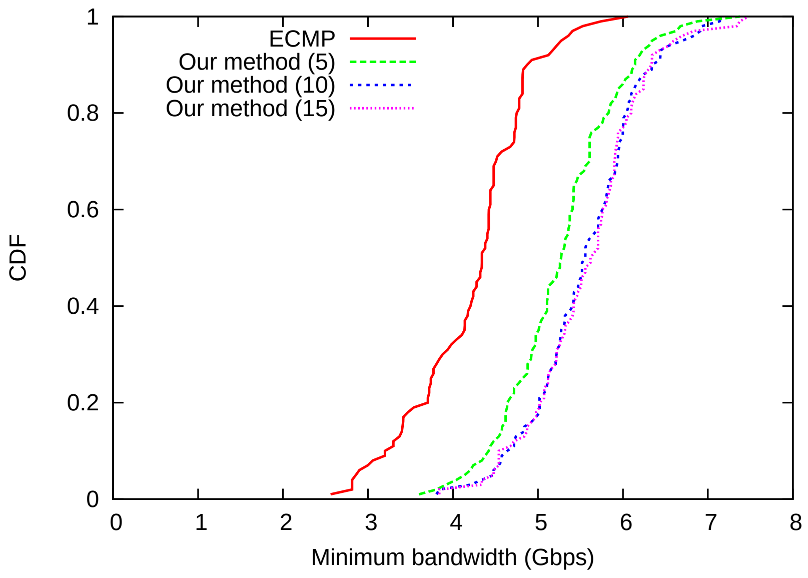


Figure 9: The minimum bandwidth between sever rack pairs in Torus

5.2 Evaluation without Failures

5.2.1 Case of Torus

Figure 9 shows the cumulative distribution function (CDF) of the minimum bandwidth provided between server racks for 100 patterns of randomly generated traffic. Figure 10 shows the CDF of the total bandwidth.

Fig. 9 also shows that the minimum bandwidth provided between serer pairs increases as the number of logical topologies becomes large. However, even if the number of the logical topologies is 5, our method provides the large bandwidth; even in the case of preconfiguring 5 logical topologies, our method provides larger bandwidth than the ECMP. This is because in addition to balancing the loads among the shortest paths, our method also uses the paths whose lengths are larger than the shortest path if the links on the shortest path do not have the sufficient bandwidth.

Fig. 10 shows that the total bandwidth also becomes large as the number of the logical topologies increases. This figure indicates that by preconfiguring more than 10 logical topologies, our method can provide larger total bandwidth than ECMP.

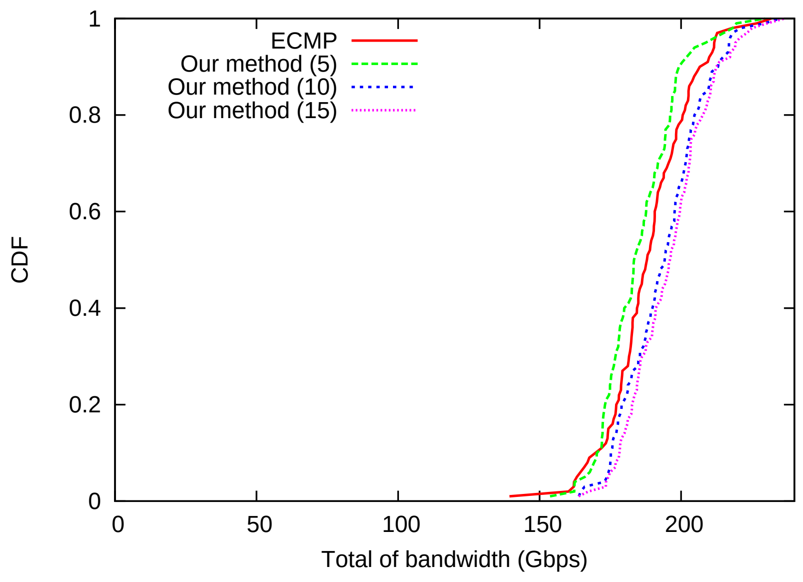


Figure 10: The total bandwidth between all sever rack pairs in Torus

5.3 Evaluation in Case of Failure

In this subsection, we compare the bandwidth provided between server racks when failures occur.

First, we evaluate the bandwidth provided between server racks when single link failure occurs. In this evaluation, all single link failures are generated. For each of the case of the failures, we generate 100 patterns of the set of the communicating server pairs.

Figures 13 and 15 show the cumulative distribution function (CDF) of the minimum bandwidth provided between server racks in Torus and FatTree, respectively. In these figures, the bandwidth provided between server rack pair is regarded as 0 if the routes between the server rack pair cannot be found.

Fig. 13 shows that our method find the routes between all communicating server pairs, while the ECMP cannot find the routes between server pairs in 25% of all cases in the Torus. This is because the ECMP uses only the shortest path. As a result, even if the available routes whose number of hops is larger than the shortest paths exist, the ECMP cannot use the routes. On the other hand, in the case of the FatTree shown in Fig. 15, the ECMP can also find the routes between all servers even if any single link failure occurs. This is because the FatTree has multiple shortest paths between all server rack pairs.

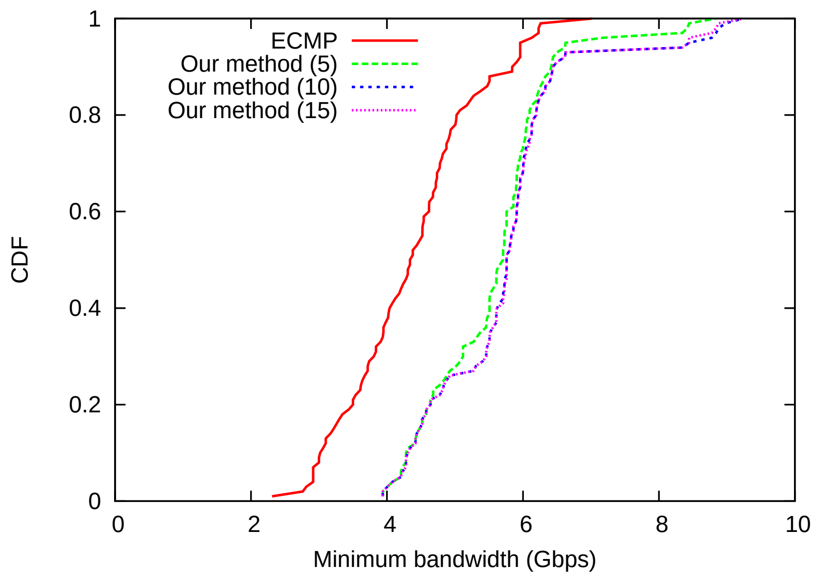


Figure 11: The minimum bandwidth between sever rack pairs in FatTree

Comparing the minimum bandwidths provided between server racks, our method provides a larger bandwidth than the ECMP as shown in Figs. 13 and 15. That is, our method provides the sufficiently large bandwidth between any communicating server pairs, even if any single link failure occur regardless of the network topoligies.

Figures 14 and 16 show the cumulative distribution function (CDF) of the total bandwidth provided between server racks in Torus and FatTree, respectively. In these figures, the ECMP provides larger bandwidth than our method. However, these results does not indicate that the ECMP provides larger bandwidth between any server pairs; as discussed above, the ECMP cannot provide sufficient bandwidth to any server rack pairs. Providing small bandwidth to certain server rack pairs makes the total bandwidth large; the bandwidths of many links remains large after accommodating the traffic between such server rack pairs, because such server rack pair cannot send the large amount o traffic. The remaining bandwidths are used by the other server rack pairs, and the bandwidths provided by the server rack pairs become large, which makes the total provided bandwidth large.

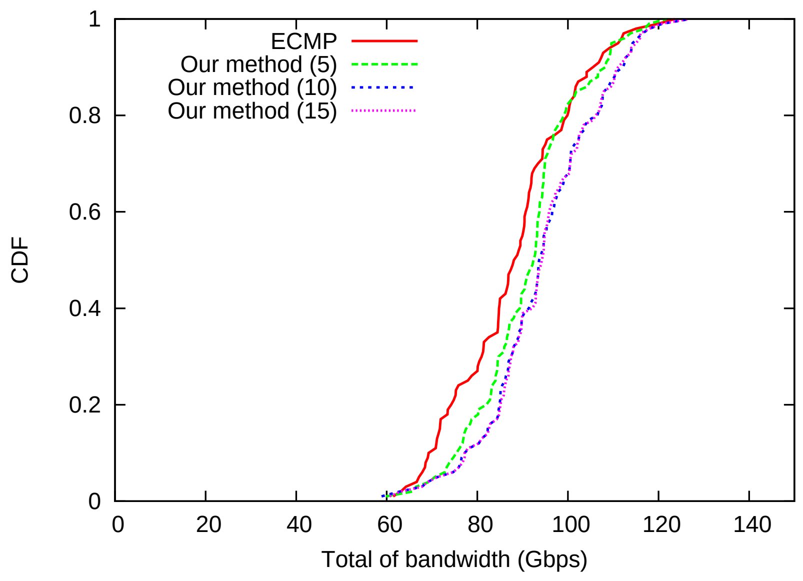


Figure 12: The total bandwidth between all sever rack pairs in FatTree

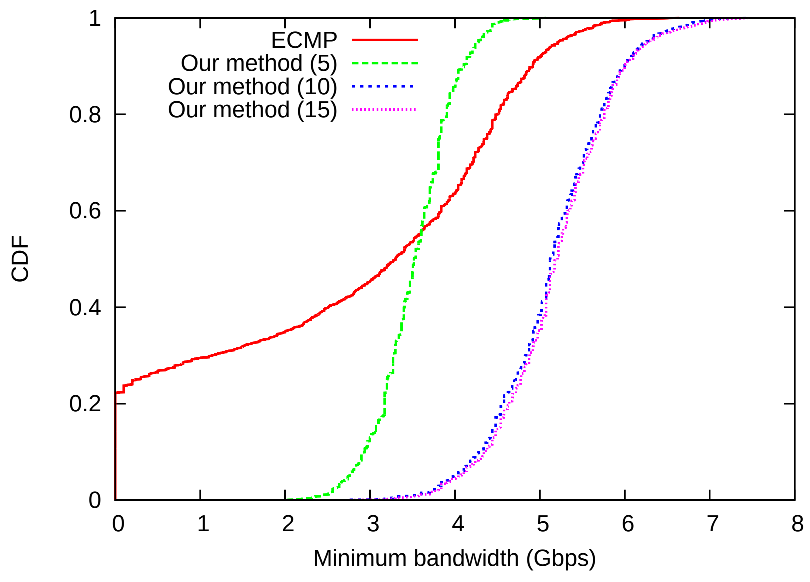


Figure 13: The minimum bandwidth between sever rack pairs in Torus in case of single link failure

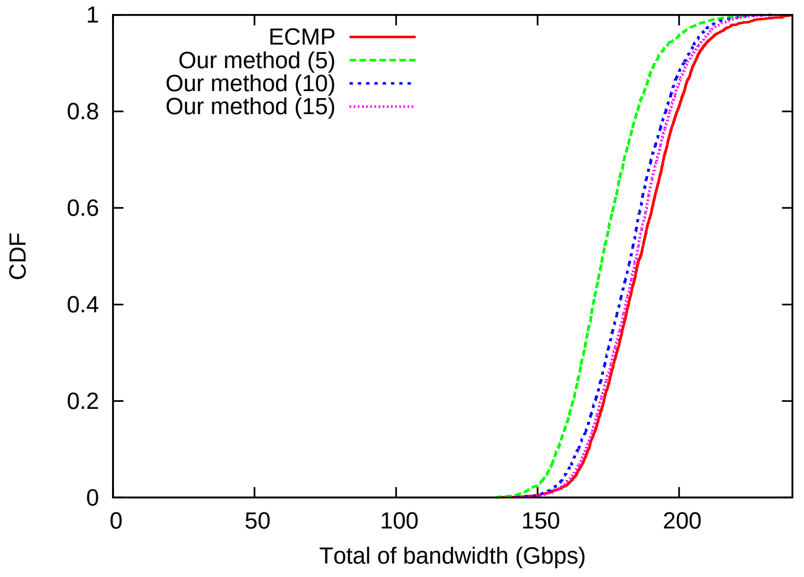


Figure 14: The total bandwidth between all sever rack pairs in Torus in case of single link failure

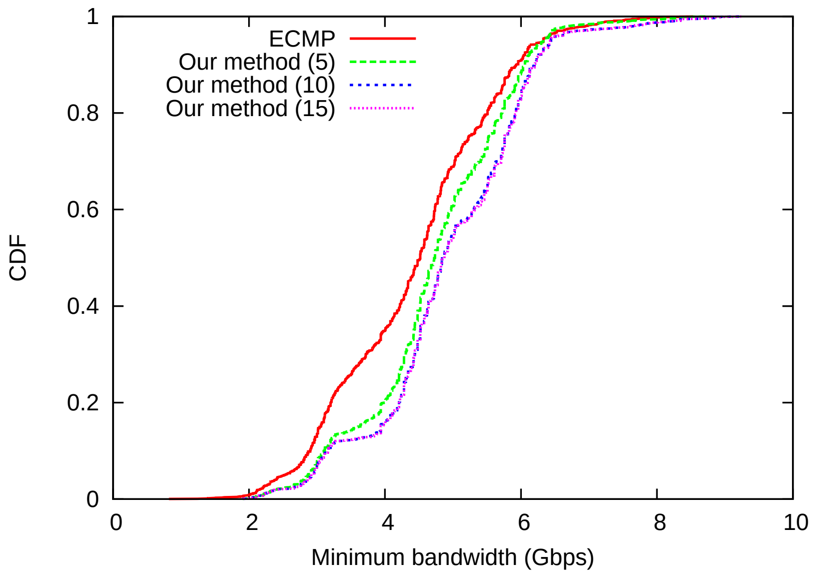


Figure 15: The minimum bandwidth between sever rack pairs in FatTree in case of single link failure

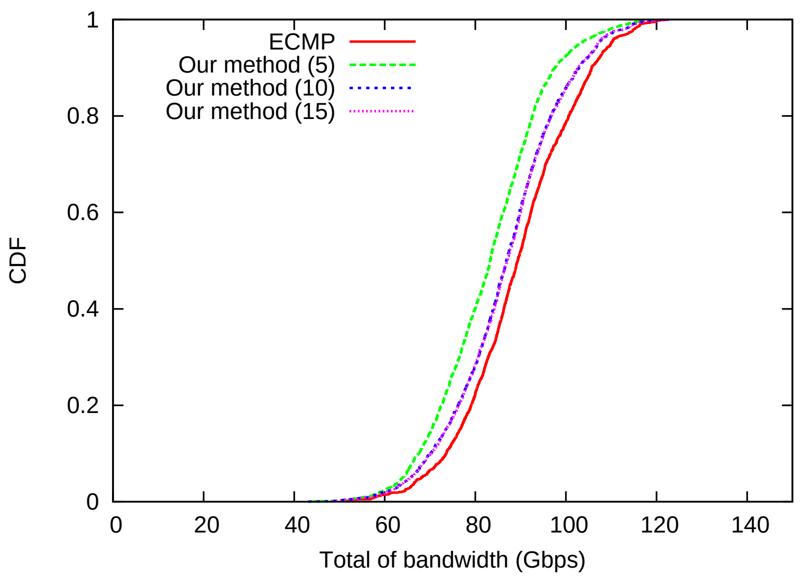


Figure 16: The total bandwidth between all sever rack pairs in FatTree in case of single link failure

6 Conclusion

Servers in a data center cooperate with each other to handle a large data, and the traffic pattern between servers changes significantly within a second. To keep the performance of the data center, sufficiently large bandwidth should be provided between any server pairs even in such frequent and significant traffic changes. Because of too frequent traffic changes, the central control cannot be applied to handle such traffic changes. In this thesis, we proposed a method to provide a sufficiently large bandwidth without central control. In this method, we preconfigure the multiple logical topologies which are the subset of the physical network. Then, each node selects the logical topologies used to send the traffic based on the traffic information monitored directly by it. In this thesis, we also proposed a method to configure the logical topologies for our routing method. Through the numerical simulation, we demonstrated that our method can provide a larger bandwidth than ECMP both cases with and without failures.

Acknowledgements

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. Without his guidance and support, I cannot complete this thesis. And also I would like to express my sincere appreciation to Associate Professor Shin ’ ichi Arakawa and Assistant Professor Yuki Koizumi for appropriate advices. In addition, I would like to express sincere appreciation to Mr. Yuya Tarutani, who is my senior associate. I received a lot of advices from him and he kindly provided consultation for me. 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] T. Benson, A. Anand, A. Akella, and M. Zhang, “The Case for Fine-Grained Traffic Engineering in Data Centers,” in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, pp. 1–6, Apr. 2010.

[2] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The Nature of Datacenter Traffic: Measurement and Analysis,” in *Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement*, pp. 202–208, Nov. 2009.

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

[4] L. A. Barroso, J. Dean, and U. Holzle, “Web Search for a Planet: The Google Cluster Architecture,” *IEEE Micro*, vol. 23, pp. 22–28, Mar. 2003.

[5] P. Gill, N. Jain, and N. Nagappan, “Understanding network failures in data centers: measurement, analysis, and implications,” *ACM SIGCOMM Computer Communication Review*, vol. 41, pp. 350–361, Aug. 2011.

[6] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdats, “Hedera: Dynamic Flow Scheduling for Data Center Networks,” in *Proceedings of the 7th USENIX conference on Networked systems design and implementation*, pp. 281–295, Apr. 2010.

[7] 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.

[8] A. K. et al, “Resilient Routing Layers for Recovery in Packet Networks,” in *Proceedings of Dependable Systems and Networks*, pp. 238–247, June 2005.

[9] M. C. Scheffel, C. G. Gruber, T. SchwabeCorresponding, and R. G. Prinz, “Optimal multi-topology routing for IP resilience,” *International Journal of Electronics and Communications*, vol. 60, pp. 35–39, Jan. 2006.

[10] 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.

[11] 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.

[12] Y. Liao, J. Yin, D. Yin, and L. Gao, "DPillar: Dual-port server interconnection network for large scale data centers," *Computer Networks*, vol. 56, pp. 2132–2147, May 2012.

[13] 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.

[14] 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 2011*, pp. 61–65, Apr. 2011.

[15] J. TOUCH and R. PERLMAN, "Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement," RFC 5556, Internet Engineering Task Force, May 2009. `http://tools.ietf.org/html/rfc5556`.

[16] C. HOPPS, "Analysis of an Equal-Cost Multi-Path Algorithm," RFC 2992, Internet Engineering Task Force, Nov. 2000. `http://tools.ietf.org/html/rfc2992`.

[17] S. S. W. Lee, K.-Y. Li, and A. Chen, "Energy efficient multi-topology routing configurations for fast failure reroute in IP networks," in *Proceedings of Global Communications Conference (GLOBECOM), 2012 IEEE*, pp. 2785–2790, Dec. 2012.

[18] 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, Oct. 2008.