Techniques for Enhancing the Rebalancing Algorithm for Folded Clos Networks

Satoru Ohta
Department of Electrical and Computer Engineering, Faculty of Engineering
Toyama Prefectural University
Imizu, Toyama, Japan
e-mail: ohta@pu-toyama.ac.jp

***Abstract*—Folded Clos Networks (FCNs) are switching networks constructed by connecting small switches aligned to ($2p + 1$)-stages ($p = 1, 2, 3, \ldots$). FCNs have often been examined in previous studies on data center networks. To take advantage of the high bandwidth provided by an FCN, it is necessary to establish an adequate routing method that uniformly diffuses flows. A previous study proposed the rebalancing algorithm as such a method, which is executable with locally obtainable information at each switch. By applying the rebalancing algorithm to an FCN, it becomes possible to impose an upper bound on the number of flows passing through a link. This means that the rebalancing algorithm prevents the link load from becoming excessively high. This paper reviews theoretical aspects of the algorithm applied to three- and five-stage FCNs. Then, two techniques are proposed for improving the rebalancing algorithm in terms of the load equality between links. The techniques distribute traffic more uniformly and do not affect the upper bound on the number of flows on a link. The effectiveness of the two techniques is assessed via computer simulation for different traffic and network models. The network models include a three-stage FCN and a five-stage FCN. The simulation results demonstrate the effectiveness of the two proposed techniques.**

Keywords- network; algorithm; routing; data center; packet.

I. INTRODUCTION

Data center networks are becoming increasingly important, as the majority of popular information services are provided via data centers. It is thus essential to establish topologies for high-performance data center networks. To this end, studies on data center networks have been performed based on several topologies, including the Clos network [1], [2] fat-tree [3], DCell [4], and BCube [5]. Among these topologies, the Clos network has often been studied because it functions as a scalable and high-bandwidth network by interconnecting small commodity switches. Various data center networks based on the Clos network topology have been implemented [2], [6]–[8].

A Clos network is a three-stage nonblocking switching network originally developed by Charles Clos in 1953 [9]. On the basis of this three-stage network, it is possible to configure nonblocking switching networks with ($2p + 1$)-stages ($p = 1, 2, 3, \ldots$). In data center network applications, the network appears in the form of a Folded Clos Network (FCN). A three-stage FCN is roughly equivalent to a three-stage network; however, it is constructed by folding the corresponding three-stage Clos network at its center. Similarly, it is also possible to construct a five- or seven-stage FCN.

To apply an FCN to data center networks, the routing of a packet is important. Inadequate routing may cause a load imbalance between the links, which in turn may cause traffic congestion and degrade performance. However, if the load is uniformly distributed among the links, an FCN can achieve high throughput by fully utilizing the bandwidth of every link.

Several previous studies [7], [8], [10] have used a routing method that involves forwarding a packet to a randomly selected route. This method is reasonable, as it uniformly distributes the average number of flows between links. However, using this method, there is a high probability that the load on a given link will become excessively large. Consequently, traffic congestion may occur due to heavily loaded links, thus degrading network performance. As discussed in [11], this problem may be critical for big data applications, which require high-bandwidth transmission. It is therefore important to develop a routing algorithm that diffuses the traffic load more uniformly than random routing.

A routing algorithm for an FCN should be executable in a distributed manner to decrease the processing overhead generated by handling frequent route decisions. In addition, the algorithm should function without global information of the entire network to eliminate the communication overhead associated with gathering information.

Routing can be performed on either a per-packet or per-flow basis. The former method determines a route in a packet-by-packet manner. Thus, packets that belong to the same flow may pass through different routes. Since delays are also different depending on the routes, packet reordering occurs for per-packet routing. Meanwhile, the latter method determines a unique route for a flow. Every packet of the flow goes through that route. This study examines a method based on per-flow routing because packet reordering is unavoidable for per-packet routing.

Ohta [12] presented two distributed algorithms – the rebalancing algorithm and the load sum algorithm – that diffuse flows in FCNs. Using computer simulations, it was demonstrated that these methods diffuse flows more uniformly than random routing. Of the two methods, the rebalancing algorithm uses information that is locally obtainable at the source switch of a flow. Although the load sum algorithm has superior performance with respect to load equality, it is less practical due to the communication overhead between switches. Thus, if the rebalancing

algorithm is improved to diffuse flows more uniformly, a more practical and efficient algorithm can be obtained.

In [1], Ohta proposed techniques for improving the rebalancing algorithm with respect to load equality. These techniques are based on information that is locally obtainable at the source switch of a flow. The first technique modifies the algorithm to distribute uplink loads more evenly, while the second technique utilizes the fact that the algorithm has a process for scanning middle switch indices for routing and rerouting. Therefore, using the second method, the order of scanning the middle switch indices is determined to uniformly diffuse flows.

An advantage of the rebalancing algorithm is that an upper bound is theoretically derived for the number of flows on a link. When using the abovementioned improvement techniques, this upper bound is not affected. Therefore, the worst-case link load is limited, as in the case in which these techniques are not applied. The effectiveness of the two techniques was confirmed via computer simulations.

This paper expands the work of [1] and offers the following contributions:

- A discussion of the presented techniques applied to a five-stage FCN.
- Performance evaluation under traffic conditions not considered in [1].

A theoretical upper bound is derived for the number of flows on a link for a five-stage FCN as well as a three-stage FCN. The performance of the proposed techniques applied to a five-stage FCN is also evaluated through computer simulation. With respect to traffic conditions, this paper examines several different traffic models in simulation. In one traffic model, flows are equally generated for every pair of source and destination switches. Other traffic models are lightly or heavily skewed. For these traffic conditions, flows are destined to a limited number of destination switches from a certain source switch. A traffic model that generates light load is also examined. The simulation results reveal that one of the proposed techniques is not effective for a highly skewed traffic model. However, this technique is more effective for light traffic load. The load equality is effectively improved for every examined traffic model if both proposed methods are used simultaneously.

The remainder of this paper is organized as follows. Section II provides a discussion of the FCN, while Section III reviews related work. Section IV explains the rebalancing algorithm, while Section V presents two modification techniques. Section VI evaluates the effectiveness of the techniques, and Section VII concludes the paper.

II. FOLDED CLOS NETWORK

Figure 1 presents an example of a Clos network. As illustrated, the network is a three-stage switching network that consists of r first-stage switches, m second-stage switches, and r third-stage switches. Each first-stage switch has n input ports whereas each third-stage switch has n output ports. This switching network was originally proposed by Charles Clos [9] and has been comprehensively investigated over a long period of time [13].

It is possible to construct a five-stage Clos network by replacing the second-stage switches with three-stage Clos networks. For example, consider the case where $n = m = 2$ and $r = 6$ in the configuration in Figure 1. Then, by replacing the second-stage switch with a three-stage network of $n = m = 2$ and $r = 3$, a five-stage network is obtained, as illustrated in Figure 2. By repeating this procedure, it is possible to construct a $(2p + 1)$-stage Clos network for any $p > 1$.

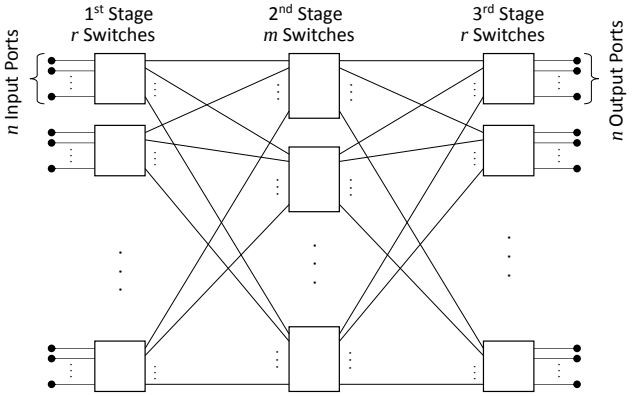

Figure 1. Example of a three-stage Clos network.

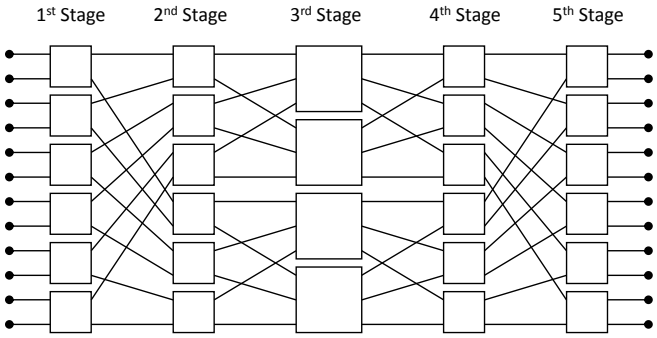

Figure 2. Example of a five-stage Clos network.

The advantages of the Clos network are its small amount of hardware and nonblocking nature. To demonstrate its advantages in hardware amount, let N denote the total number of input or output ports. In the classical switching network theory, the hardware amount of a switching network is often measured by the number of crosspoints, assuming that each small switch is a crossbar. A single crossbar switch with N ports has N^2 crosspoints, and the number of crosspoints is $O(N^{3/2})$ for a three-stage Clos network [9]. Thus, the amount of hardware is much smaller for a Clos network than for a single crossbar switch. In addition, the number of crosspoints is $O(N^{4/3})$ for a five-stage Clos network. Thus, a five-stage configuration can be constructed with less hardware for a larger value of N. It is also known that the amount of hardware for a Clos network is $O(Ne^{2\sqrt{(\log_e 2)\cdot(\log_e N)}})$ when the number of stages is optimized for N [14].

A three-stage FCN is roughly equivalent to a three-stage Clos network. However, an FCN is constructed by folding the three-stage network at its center. An example of a three-stage

FCN is provided in Figure 3. In a three-stage FCN, the first- and third-stage switches are integrated into input/output switches, and the second-stage switches are middle switches that connect the input/output switches.

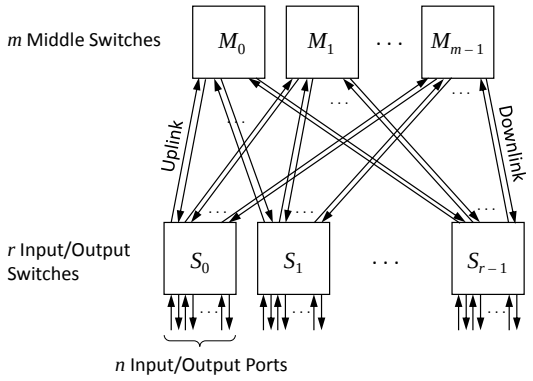

Figure 3. Example of a three-stage FCN.

As illustrated in Figure 3, r input/output switches are labeled S_0, S_1, …, S_{r-1}, while m middle switches are labeled M_0, M_1, …, M_{m-1}. Every middle switch is connected to every input/output switch via an uplink and downlink. An uplink is set from an input/output switch to a middle switch, whereas a downlink is set in the reverse direction. Each input/output switch has n input/output ports.

By implementing each switch as an IP (Internet Protocol) layer 2 or 3 switch, a data center network can be constructed on the basis of an FCN. It is known that a data center network based on Clos topology has the advantages of scalability and high bandwidth.

Because this paper considers applications to data center networks, the information passes through the FCN via packets. Since a middle switch is connected to every input/output switch, a packet can reach its destination switch from an arbitrary middle switch via a downlink. Therefore, the source switch can transmit a packet to the destination switch via any middle switch. However, the traffic load on an uplink or downlink depends on the routing at the source switch. If the routing is inadequate, traffic congestion occurs, which degrades performance. Congestion can be avoided if the traffic is evenly diffused between the uplinks and downlinks in an FCN. It is therefore important to establish a routing method that is executed at the source switch of a packet.

This paper assumes that routing is performed on a flow basis. A flow is a packet stream identified by a set of fields in the packet header [15]. A frequently used field set is {*source address*, *destination address*, *protocol*, *source port*, *destination port*}, which is associated with an IP socket. A different field set can also be used as flow identifiers. If a fixed route is assigned to a flow, packet reordering does not occur. This is advantageous because packet reordering leads to throughput degradation. This paper considers the case in which an FCN connects many hosts and processes via its $N = nr$ input/output ports. In this situation, many concurrent flows exist between ports.

Flows are categorized as elastic or stream [16] depending on the nature of the traffic. In this paper, it is assumed that the network handles elastic flows. This assumption is reasonable because many services are supported by TCP (Transmission Control Protocol), and TCP traffic is elastic. For elastic flows, the throughput of a flow is restricted by the link capacity portion shared with other flows. Therefore, to achieve high throughput, it is indispensable to uniformly diffuse the number of flows among links and decrease the maximum number of flows on a link. Thus, in this paper, the link load is assessed by the number of flows.

Whereas the network illustrated in Figure 3 is based on a three-stage Clos network, it is also possible to construct an FCN from a five-stage Clos network. Figure 4 presents an example of a five-stage FCN. This network is the configuration that results from replacing a middle switch with an FCN in the configuration in Figure 3. This type of five-stage FCN is used as a data center network, as reported in [6].

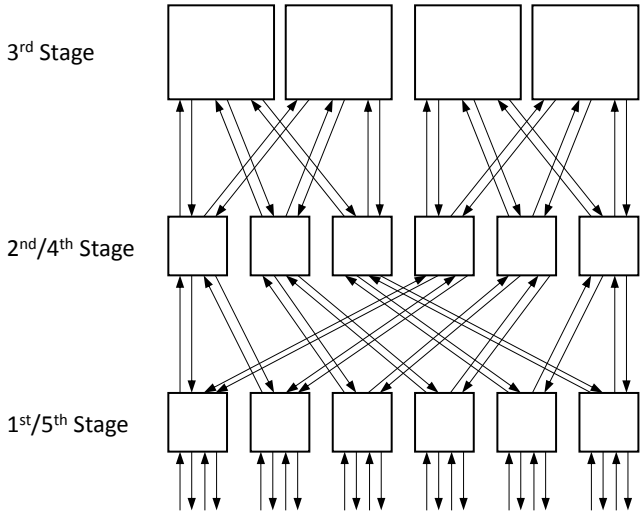

Figure 4. Example of a five-stage FCN.

III. Related Work

An FCN first appeared in the original work by Clos [9] as the triangular array configuration. The configuration has also been referred to as a Clos truncated network [17] or single-sided Clos network [18]. Clos networks or FCNs have been used for various applications including telephone switching [9], [17] cross-connect systems [19]–[21], multi-processor computers [10], network-on-chip [22]–[24], and data center networks [1], [2], [6]–[8], [11], [12], [25]–[28].

For computer network applications, including data center networks, Clos networks are treated as packet switching networks. Various studies have investigated the application of Clos networks as packet switching networks. The routing methods examined in those studies are categorized as per-packet routing and per-flow routing. As an example of per-packet routing, Hassen and Mhamdi [25] investigated a Clos network that had crossbar switches with small-size buffers in each stage. For this configuration, Hassen and Mhamdi proposed distributed and centralized packet scheduling

mechanisms to achieve low packet delay and high throughput. However, their proposed method involves per-packet processing to perform scheduling. Thus, the processing load generated by each packet arrival will become critical for a large network that handles a large number of packets in a unit time. In addition, the advantage of a small-size buffer may not be significant because buffer memory is not expensive.

In another study, Hassen and Mhamdi [26] investigated a modified Clos network configuration that employed a multi-directional network on a chip as a switching module in each stage. This module provides interconnections between middle stage switching modules. For this network configuration, inter-module routing is performed based on global network congestion information. However, it is uncertain whether gathering congestion information for routing is practical.

A quite different per-packet routing method was proposed by Yang et al. [27]. This method determines the switch connections in each stage from a given traffic matrix, where each element of the matrix represents the number of packets to be transmitted from a source to a destination during a certain period. The connections are reconfigured several times to minimize the cost during the transmission period. Unfortunately, it is very unlikely that an exact traffic matrix could be obtained in a real-world data center network. Thus, it may be difficult to apply their method.

To avoid congestion in a Clos network, Ghorbani et al. [28] proposed a method that diffuses traffic on a packet-by-packet basis. This method determines the next hop of a packet according to the queue length of each output buffer in a switch. Consequently, packets of a flow may go through routes with different queueing delays, and this delay variation may cause packet reordering. Ghorbani et al. performed a computer simulation and observed that 0.02 % of packets were delivered out of order. The extent to which this rate of packet reordering affects TCP throughput is unclear.

The studies of [25]–[28] are theoretical, and the proposed methods are evaluated through computer simulations. However, an actual implementation of per-packet routing was reported by Scott et al. [10]. Their Clos network is designed for the interconnection of a multi-processor high performance computer. The routing method employed in their work determines the next hop of a packet depending on the buffer space. The proposed method is based on a custom packet format developed for the multi-processor computer. Consequently, their method will not be directly applicable to datacenter networks.

A more practical approach to congestion avoidance is traffic diffusion based on per-flow routing methods. A common method of flow-based traffic diffusion involves routing a flow to a randomly selected middle switch. This technique, which is referred to as Valiant load balancing, is employed in the system implemented by Greenberg et al. [7] and was originally proposed by Valiant [29] for a binary cube topology. Al-Fares et al. [8] explored a method of computing routing table entries from the indices of switches and host identifiers. This is equivalent to randomly assigning route flows using the output of a hash function fed by switch indices and host identifiers. The architecture reported by Scott et al. employs a per-packet adaptive routing mechanism as well as per-flow deterministic routing [10]. For deterministic routing, flows are diffused to random middle switches via a hash function fed by input ports and destinations.

The idea of randomly routing flows is reasonable because the average number of flows is balanced between links. However, there is a high probability that the worst-case load on a certain link will become excessively large. This can cause traffic congestion and degrade performance, such as packet latency or network throughput. The adaptive routing proposed by Zahavi et al. [11] may reduce this disadvantage of random routing. Initially, this method semi-randomly selects routes for the flows at the source switches. Then, the destination switches identify bad links, which are excessively loaded by the initial routing. Next, the destination switches notify the source switches of the flows, passing the bad links as bad flows. With this notification, the source switches can reroute the bad flows, and the rerouting is repeated until there are no bad links. In [11], the convergence of the rerouting was evaluated using a theoretical analysis based on Markov chain models and computer simulation. However, the number of times the flows must be rerouted to eliminate all bad links in the worst case is not theoretically known. It is also unclear whether bad links are definitively removed by Zahavi et al.'s method. As a result, this method may not be practical.

Ohta [12] presented two flow-based routing methods that diffuse flows more uniformly than random routing. Using these methods, a flow is routed (or rerouted) at its source switch in a distributed manner. One of these methods is the rebalancing algorithm, which runs using locally obtainable information. The other method is the load sum algorithm, which requires communication between source and destination switches. Simulation results demonstrate that these methods both outperform random routing. In addition, the load sum algorithm is shown to diffuse flows more uniformly than the rebalancing algorithm.

The simulation results reported in [12] indicate that all flow equality metrics are smaller for the load sum algorithm than for the rebalancing algorithm. However, the load sum algorithm may be inefficient with respect to the communication overhead between switches, in particular in the case of short-duration flows. Namely, the amount of traffic exchanged by a short-duration flow may become comparable to or smaller than that exchanged by the communication between switches, which is highly inefficient. From this viewpoint, the rebalancing algorithm is likely to be more practical. With improvements to this method in terms of the uniformity of flow diffusion, the rebalancing algorithm can become more effective.

IV. REBALANCING ALGORITHM

This paper focuses on the rebalancing algorithm presented in [12]. This algorithm is in fact a packet stream version of the method described in [30]. It assumes that the route of a newly generated flow is determined when its first packet arrives at the input switch. Implementing such a mechanism with currently available technology would not be simple; however, it is important to investigate potential methods that display higher performance than conventional routing.

This section presents several definitions, specifies local information, and outlines the algorithm.

A. Definitions

Throughout this paper, the following variables are used for a three-stage FCN:

- $F(i, j, k)$: number of flows that pass through a source switch S_i, middle switch M_j, and destination switch S_k $(0 \le i, k \le r-1, 0 \le j \le m-1)$
- $U(i, j)$: number of flows on the uplink set from S_i to M_j
- $D(j, k)$: number of flows on the downlink set from M_j to S_k

In an FCN, if the source switch of a flow is the same as its destination, it is not necessary to route the flow to a middle switch. The flow can be directly routed to the destination within the source/destination switch. In view of this characteristic, $U(i, j)$ and $D(j, k)$ are related to $F(i, j, k)$ as follows:

$$U(i, j) = \sum_{k=0, i \neq k}^{r-1} F(i, j, k), \quad (1)$$

$$D(j, k) = \sum_{i=0, i \neq k}^{r-1} F(i, j, k). \quad (2)$$

The algorithm is described using these variables. Table I summarizes the symbols used in this paper.

TABLE I. TABLE OF SYMBOLS.

Symbol	Definition
m	Number of middle switches in a three-stage FCN
n	Number of input/output ports that an input/output switch has in a three-stage FCN
r	Number of input/output switches in a three-stage FCN
m_1	Number of sub-FCNs in a five-stage FCN
n_1	Number of input/output ports that a 1st/5th stage switch has in a five-stage FCN
r_1	Number of 1st/5th stage switches in a five-stage FCN
m_2	Number of 3rd srage switches in a sub-FCN of a five-stage FCN
n_2	Number of input/output ports that a 2nd/4th stage switch has in a five-stage FCN
r_2	The number of 2nd/4th stage switches in a sub-FCN of a five-stage FCN
S_i	Input/output switch of a three-stage FCN
M_j	Middle switch of a three-stage FCN
$F(i, j, k)$	Number of flows established from S_i to S_k via M_j
$U(i, j)$	Number of flows on the uplink from S_i to M_j
$D(j, k)$	Number of flows on the downlink from M_j to S_k
α	Positive integer used
f_0	Maximum number of flows for an input/output port
f_1	Maximum number of flows on a link between an input/output (1st/5th stage) switch and a middle (2nd/4th stage) switch for a three-stage FCN (five-stage FCN)
f_2	Maximum number of flows on a link between a 2nd/4th stage switch and a 3rd srage switch for a five-stage FCN

B. Locally obtainable information

For data center network applications, flows may be frequently generated and completed in the FCN. In this situation, the routing of a flow should be executed in a distributed manner because the load resulting from frequent route decisions becomes excessively high for concentrated computations. In addition, it is impractical to perform communication between switches because there may be very short flows consisting of only several packets. As described in Section III, it is inefficient to exchange packets between switches for the routing of such short flows. Therefore, the route of a flow should be determined at its source switch using locally obtainable information.

Let us consider the case in which the FCN is a three-stage configuration. An input/output switch can obtain the headers of the packets, which arrive from its input port and are forwarded to middle switches. From these headers, the switch can identify the flows to which the packets belong. Because the switch determines the routes for the flows at the source, it can count the number of flows that travel from itself to each middle switch. Therefore, $U(i, j)$ can be managed at the source switch S_i. In addition, the switch can extract the destination switch of the flows from the packet headers. Using this information, the source switch S_i can also count $F(i, j, k)$.

Suppose that a new flow is generated and that its source switch is S_i. Then, assume that S_i can detect the arrival of a new flow. This is possible by comparing the flow identifiers to the routing table. It is also possible for S_i to detect the completion of a flow by a timeout. Therefore, S_i can launch routing or rerouting processes at flow arrival or completion.

C. Algorithm properties

The rebalancing algorithm is detailed in [12]. The algorithm utilizes parameter α, positive integer that controls its behavior. The rebalancing algorithm has the following property:

Property 1: *With the rebalancing algorithm,*

$$F(i, j, k) \le F(i, j', k) + \alpha, \quad (3)$$

for $0 \le j, j' \le m-1,\ 0 \le i, k \le r-1.$

The proof for Property 1 is found in [12]. An advantage of the rebalancing algorithm is that an upper bound exists for the number of flows on an uplink or downlink. Let f_0 denote the maximum number of flows for an input or output port. In addition, let f_1 denote the number of flows on an uplink and downlink. Then, the following property is obtained [12]:

Property 2: *When the rebalancing algorithm is performed on a three-stage FCN characterized by parameters m, n, and r,*

$$f_1 \le \frac{nf_0}{m} + \alpha\left(1 - \frac{1}{m}\right)(r-1) \quad (4)$$

Property 2 is proved from (1), (2), and (3) via the method outlined in [12]. This property ensures that the load on a link does not become very high.

In the rebalancing algorithm, parameter α determines the frequency of rerouting as well as the uniformity of flow diffusion. If α is large, rerouting never occurs. In this case, flows are diffused via route decision when they arrive at their source switches. Simulation results demonstrate that the algorithm works well even without rerouting. Following [12], a rebalancing algorithm that omits the rerouting process is hereafter referred to as *balancing algorithm*.

D. Rebalancing algorithm for five-stage FCNs

The rebalancing or balancing algorithm can be also applied to five-stage FCNs. This subsection shows that the number of flows on a link is upper bounded by the rebalancing algorithm for a five-stage FCN as well as for a three-stage FCN. The characteristic of the rebalancing algorithm applied to a five-stage FCN has never been reported elsewhere. Thus, the analysis on the five-stage FCN case is a main contribution of this paper.

As illustrated in Figure 5, a five-stage FCN can be seen as a combination of three-stage FCNs. In Figure 5, the second/fourth-stage switches and third-stage switches configure m_1 sub-FCNs, 0, 1, …, $m_1 - 1$. By considering these sub-FCNs as m_1 switches, the algorithm can be executed at the first/fifth-stage switches. Within each sub-FCN, it is also possible to run the algorithm at each second/fourth-stage switch. For this scheme, it should be noted that flow rerouting between first/fifth-stage switches causes a new flow generation and flow completion in the affected sub-FCNs. This means that flow rerouting may generate additional flow rerouting in the sub-FCN.

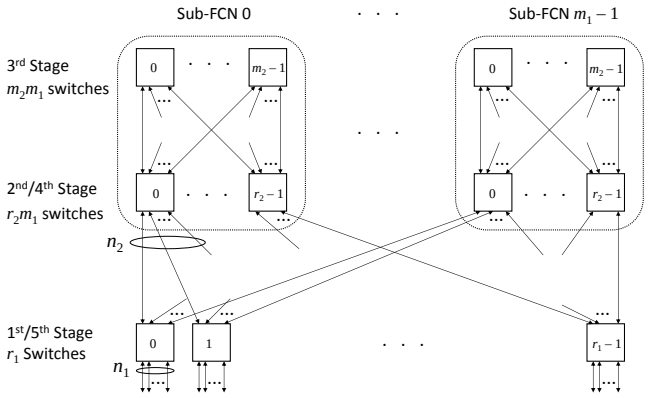

Figure 5. Parameters m_1, m_2, n_1, n_2, r_1, and r_2 for a five-stage FCN.

In the five-stage FCN case, the number of flows is upper bounded by employing the rebalancing algorithm as well as in the three-stage FCN case. Let n_1 denote the number of input/output ports of a first/fifth-stage switch, and let n_2 denote the number of input/output ports of a second/fourth-stage switch. In addition, let r_1 be the total number of first/fifth-stage switches. Assume that each sub-FCN is constructed by m_2 third-stage switches and r_2 second/fourth-stage switches. For this configuration, r_1 must be equal to r_2n_2.

Let f_1 be the maximum number of flows on an uplink or downlink between a first/fifth-stage switch and a second/fourth-stage switch. Additionally, let f_0 denote the maximum number of flows for an input or output port. Then, from Property 2,

$$f_1 \le \frac{n_1 f_0}{m_1} + \alpha\left(1 - \frac{1}{m_1}\right)(r_1 - 1) \quad (5)$$

Next, let f_2 be the maximum number of flows on a link between a second/fourth-stage switch and third-stage switch. Then, because a sub-FCN is also controlled by the rebalancing algorithm, f_2 is bounded by f_1 as follows:

$$f_2 \le \frac{n_2 f_1}{m_2} + \alpha\left(1 - \frac{1}{m_2}\right)(r_2 - 1) \quad (6)$$

Then, from (5) and (6),

$$f_2 \le \frac{n_1 n_2 f_0}{m_1 m_2} + \frac{n_2 \alpha}{m_2}\left(1 - \frac{1}{m_1}\right)(r_1 - 1) + \alpha\left(1 - \frac{1}{m_2}\right)(r_2 - 1) \quad (7)$$

Thus, the load on every link is also upper bounded for five-stage FCNs.

V. Modification Techniques

This section presents two modification techniques to improve the load equality of the rebalancing algorithm. These techniques add criteria for selecting the middle switch index. However, they do not change the conditions that $F(i, j, k)$s must satisfy. Therefore, (3) holds even if these techniques are applied. Consequently, the upper bound expressed by (4) or (7) is unchanged by these techniques.

A. Uplink flow diffusion

The rebalancing algorithm uses $F(i, j, k)$ and flow arrival and completion events from the local information. Therefore, of the available local information, $U(i, j)$ remains unused. Although the rebalancing algorithm decreases the difference between the $F(i, j, k)$s for a particular pair of i and k, the uplink load $U(i, j)$ is not necessarily uniformly distributed. For a new flow arrival, the middle switch M_J is selected such that $F(i, J, k)$ is the minimum of the $F(i, j, k)$s. In this process, there may be two or more candidates for J. Let us select J from the candidates so that $U(i, J)$ is the minimum of the candidates. Then, flows are more uniformly distributed between the uplinks. This does not necessarily improve the load equality between the downlinks; however, the performance is improved for the uplinks.

Similarly, flow diffusion via rerouting can also be modified using $U(i, j)$. For rerouting, middle switch M_J is selected such that $F(i, J, k)$ is the maximum of the $F(i, j, k)$s. Suppose that there are two or more such indices J. Then, it is possible to use the index that maximizes $U(i, J)$. We refer to this modification using $U(i, j)$ as *modification 1*.

B. Start index for scanning the middle switches

The order of searching for middle switch index J also affects the performance of the rebalancing algorithm. Assume that J is scanned in the order of 0, 1, …, $m - 1$ for a new flow

arrival. Then, a smaller index is more likely to be selected as J. Therefore, $F(i, j, k)$ has a high probability of being larger for a smaller index j even though the differences between the $F(i, j, k)$s are bounded by α for fixed i and k. According to (1) and (2), this implies that $U(i, j)$ and $D(j, k)$ also tend to be larger for a smaller value of j. To avoid this imbalance between $U(i, j)$ and $D(j, k)$, the scanning of middle switches should start from a different index depending on k for a fixed value of i. Similarly, the start index should differ depending on i for a fixed value of k. In addition, the start index should be evenly distributed between 0, 1, …, $m-1$ for different values of i or k. To satisfy this requirement, let us examine the following start index j_s:

$$j_s = (i + k)\lceil m / r \rceil . \quad (8)$$

If j_s is greater than $m-1$, j_s is replaced by j_s mod m. In (8), the term $\lceil m / r \rceil$ is necessary for evenly distributing j_s between 0, 1, …, $m-1$ for the case of $m \geq 2r$. For a new flow arrival, the index is scanned in the order of j_s, $j_s + 1$, $j_s + 2$, …; if the index reaches m, it wraps to 0.

For rerouting, simulation results reveal that the index should be started from $(j_s + m)$ mod m and then decreased. If the index reaches -1, it wraps to $m-1$. The rationale for this scheme is as follows. The scheme aims to generate a situation in which $F(i, j_s, k) \geq F(i, j_s + 1, k) \geq F(i, j_s + 2, k) \geq \ldots$. To maintain this situation, it is preferable to select J from later elements of the sequence j_s, $j_s + 1$, $j_s + 2$, …, $(j_s + m)$ mod m because $F(i, J, k)$ decreases due to rerouting. The use of the abovementioned start index is hereafter referred to as *modification 2*.

VI. EVALUATION

The effectiveness of the improvements was evaluated using computer simulations. The simulations examined the rebalancing and balancing algorithms to which modifications 1 and 2 were applied. For comparison, the original rebalancing and balancing algorithms reported in [12] were also evaluated. In the rebalancing algorithm, parameter α was set to 1.

A. Load equality metric

The degree of load equality was estimated using the following metrics, which were also used in [12]:

- Maximum: the maximum number of flows in the links at a certain measurement time
- Variance: the variance in the flow numbers in the links at a certain measurement time
- Bad links: the number of links in which the number of flows exceeds threshold C at a certain measurement time

By using three different metrics, it is possible to reliably evaluate load equality. Of the metrics, it is evident that variance is an adequate measure for load equality. However, even for the same variance or standard deviation value, the degree of traffic congestion may differ. For example, consider two cases that exhibit the same variance value. For these cases, suppose that the maximum number of flows passing through a link is greater for one case than the other. Then, because the flows share the link capacity, the throughput of a flow will decrease and the performance will be more strongly degraded in the former case. This suggests that the maximum metric is necessary as well.

It is also evident that the employment of the variance and maximum metrics are insufficient. Consider two cases with an identical maximum metric value. The first case is a situation in which the number of flows takes the maximum value for only one link but is not large for any other links. In contrast, in the second case, the number of flows is equal or close to the maximum value for many links. It is clear that a greater number of flows will be degraded in the latter case. That is, the range of flows affected by congestion differs for these cases. The bad links metric is thus essential for distinguishing this difference.

B. Simulation model

In the simulations, the following two network models were employed:

- Three-stage FCN with $r = 48$, $m = n = 24$, and
- Five-stage FCN: $r_1 = 144$, $m_1 = n_1 = 8$, and $m_2 = n_2 = r_2 = 12$.

The parameters used for the three-stage FCN were the same as those used in the model examined in [11]. Thus, the parameters were adequate for simulating a realistic network. The parameters for the five-stage FCN model were determined so that the same number of links as in the three-stage model were generated between the stages. Thus, for both models, there were $m \times r = m_1 \times r_1 = m_1 \times m_2 \times r_2 = 1{,}152$ uplinks and downlinks between stages. The total number of input or output ports was also 1,152 ($= r \times n = r_1 \times n_1$).

A flow was generated by opening a socket between hosts a and z, which were connected to two different input/output switches. By opening a socket, two flows were generated in the direction from a to z as well as in the reverse direction.

The simulation examined five traffic models. These models were constructed as follows. A previous study [7] reported that, in a real-world data center, an average machine has 10 concurrent flows. By aggregating the traffic from 10 such machines, the average number of flows is 100 for a port on each input/output switch. Four traffic models simulated this situation, and one traffic model simulated a lighter load, i.e., 25 flows on average for each input/output port.

The average number of flows was set to 100 or 25 for a port as follows. The duration of a socket was a random value according to an exponential distribution with an average of 57.6 s. This value is realistic to some extent because 10's of gigabytes of data are transmitted for some "big-data" applications [11]. The transfer time will reach some ten seconds for such applications even if the flow throughput reaches some Gb/s. To set the average number of flows to 100, the interval of opening sockets was randomly determined by an exponential distribution at an average of 0.001 s. Now, let N flows exist at a certain time in the FCN. Then, on average, N/57.6 flows are completed in 1 s whereas 2/0.001 flows are generated in 1 s. In equilibrium, these flow completion, and

generation rates are balanced. This suggest that N equals $2 \times 57.6/0.001$. Thus, the average number of flows provided to one of 1152 input or an output port is 100. Similarly, by setting the average interval of opening sockets to 0.004 s, the average number of flows to a port is 25.

The threshold for bad links, C, was set to 105 for the models, where the average number of flows given to a port is 100. When the average number of flows is 25, C was set to 30. These values were slightly larger than the average number of flows for the given traffic condition. The values of the above metrics would have been smaller if the flows were more uniformly diffused.

Four traffic models are the same in point that the average number of flows given to a port is 100. Of the models, three are intended for the three-stage FCN, while one model is intended for the five-stage FCN. The models intended for the three-stage FCN differ in their selection of source-destination pairs for flows. The models are defined as follows.

Traffic #1: For this model, the source-destination switch pair is uniformly distributed. To generate this model, a pair of different source and destination switches is randomly selected with equal probability. Then, input and output ports are randomly selected for the source and destination switches.

Traffic #2: This model simulates lightly skewed traffic. Namely, for a randomly selected source switch index i, the destination switch index k is selected from a certain range of switch indices with equal probability. The difference between k and i is kept greater than $r/4$. Specifically, k is set to a value $R(i + x)$ where

$$R(x) \triangleq x \bmod r \tag{9}$$

and $r/4 < x < 3r/4$.

Traffic #3: This model simulates heavily skewed traffic. For a randomly selected source switch index i, the destination switch index is selected from the following three numbers: $R(i + r/2 - 1)$, $R(i + r/2)$, and $R(i + r/2 + 1)$.

Traffic #4: For this model, flows are generated similarly as for Traffic #1 except that a flow passes through different second/fourth-stage switches for the source and destination sides. Thus, every flow passes through a third-stage switch. Using this rule, an equal average load is provided to a link between first/fifth-stage and second/fourth-stage switches as well as to a link between second/fourth-stage and third-stage switches.

Additionally, light traffic load was also examined for the three-stage FCN by the following model.

Traffic #5: This model is the same as Traffic #1 except that the average number of flows given to a port is 25.

The sockets were opened 2×10^6 times for Traffic #1–#4 and 5×10^5 times for Traffic #5. The metrics were measured every 1 s in the period from 401–1,900 s. The system was considered to be in equilibrium during this period. At each measurement time, the metrics were obtained from 2,304 links (1,152 uplinks and 1,152 downlinks) for the three-stage FCN, and 4,608 links for the five-stage FCN. By executing this metric calculation from 401 s to 1,900 s, 1,500 samples were obtained for one execution of the simulation program. This process was repeated 10 times with different initial values for the random function to obtain reliable results. The averages of the metrics were computed from the measured data.

Hereafter, the term *maximum* signifies the average of the sampled maxima. Thus, values labeled as the maximum are real numbers, although each sample value of the maximum metric is an integer. Similarly, the average value of the bad link metric is also a real number, although its sample is an integer.

The simulation was performed by a custom event-driven simulation program that listed events, including flow generations and flow completions, in a table. Then, the program executed the process associated with the event according to the scheduled time. Because the proposed methods were evaluated for flow characteristics, it was not necessary to consider packet behaviors or protocols that are precisely modeled by existing simulation platforms (for example, ns-3 [19]). For this purpose, the use of a custom program was more efficient. The program was built using the C language and compiled using GCC 4.8.5. The simulation was performed on a Core i3/16GB RAM PC running on CentOS 7.

C. Simulation results

Table II summarizes the simulation results for the rebalancing algorithm and the three-stage FCN fed with Traffic #1, while Table III presents the results for the balancing algorithm, the three-stage FCN, and Traffic #1.

TABLE II. RESULTS FOR THE REBALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #1 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	111.701	11.154	122.816
Modification 1	111.102	7.713	66.366
Modification 2	109.276	8.710	75.086
Modifications 1 & 2	110.370	7.163	55.734

TABLE III. RESULTS FOR THE BALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #1 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	113.429	15.121	186.507
Modification 1	112.686	9.996	101.800
Modification 2	110.846	11.781	127.464
Modifications 1 & 2	112.420	9.734	97.372

Tables II and III demonstrate that the load equality was successfully improved by modifications 1 and 2. As illustrated in the tables, every metric decreased when the modifications were applied. In particular, modification 1 effectively

improved the variance and bad links metrics. Therefore, this modification is effective even though it does not affect the equality between the $D(j, k)$s. The improvements due to modification 2 were not as large as those due to modification 1. However, all metrics also decreased when modification 2 was applied. The best results were obtained for the variance and bad links metrics when both modifications 1 and 2 were applied. The improvement in the bad links metric was particularly notable. This implies that the number of flows is concentrated in a narrow range for most links. For the examined network, the bound expressed by (4) is 145 when f_0 is assumed to be 100. Thus, from Table II, the actual maximum appears much smaller than that upper bound.

In a comparison between the rebalancing and balancing algorithms, it was found that the former was always superior to the latter for any case. However, the rerouting performed by the rebalancing algorithm may cause packet reordering, which may decrease the throughput. Meanwhile, the proposed modifications considerably improved the load equality of the balancing algorithm, which does not perform rerouting. When the modifications were applied, the load equality was better for the balancing algorithm than that for the original version of the rebalancing algorithm. Therefore, a practical solution is to use the balancing algorithm with the proposed modifications.

Tables IV and V list the results for the three-stage FCN and Traffic #2, lightly skewed traffic. Table IV pertains to the case of the rebalancing algorithm, while Table V pertains to the case of the balancing algorithm.

TABLE IV. RESULTS FOR THE REBALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #2 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	109.545	8.127	61.666
Modification 1	109.108	6.224	35.673
Modification 2	107.823	6.653	36.653
Modifications 1 & 2	108.570	5.865	29.412

TABLE V. RESULTS FOR THE BALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #2 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	110.703	10.365	96.802
Modification 1	110.209	7.596	55.571
Modification 2	109.038	8.521	63.876
Modifications 1 & 2	110.018	7.450	52.879

Tables IV and V reveal that every metric also decreased for the case of Traffic #2 when the modifications were applied. Similarly to the case of Traffic #1, the best result was obtained by applying both modifications 1 and 2.

Despite the results presented in Tables II–V, it cannot be concluded that the modifications are always effective for all traffic models. This is illustrated in Tables VI and VII, which display the results for Traffic #3. Table VI displays the results for the rebalancing algorithm, while Table VII displays the results for the balancing algorithm.

TABLE VI. RESULTS FOR THE REBALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #3 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	105.863	4.535	12.716
Modification 1	105.762	4.383	11.224
Modification 2	105.920	4.801	15.700
Modifications 1 & 2	105.817	4.407	11.503

TABLE VII. RESULTS FOR THE BALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #3 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	106.317	4.858	15.490
Modification 1	106.157	4.633	13.310
Modification 2	106.411	5.197	19.384
Modifications 1 & 2	106.194	4.649	13.475

For Traffic #3, Tables VI and VII indicate that modification 2 is not particularly effective. As illustrated in the tables, when modification 2 was applied, every metric increased. This result can be explained by the definition of the search start index j_s used in modification 2. As seen in (8), j_s is determined by the indices of the source and destination switches. Meanwhile, a source-destination pair is selected from very few (namely, three) candidates for Traffic #3. Due to this heterogeneity in source-destination pairs, the start index j_s is not efficiently distributed over $0, 1, \ldots, m - 1$, thus leading to a less uniform load on the links.

Tables VIII and IX present the results for the five-stage FCN and Traffic #4. The results for the rebalancing algorithm are presented in Table VIII, while the results for the balancing algorithm are presented in Table IX. The tables demonstrate that each modification efficiently improved the metrics for the five-stage FCN. Similarly to the case of the three-stage FCN, the best result was obtained by applying modifications 1 and 2. However, the advantage of applying modifications 1 and 2 is not great in comparison with the case of applying only modification 1.

TABLE VIII. RESULTS FOR THE REBALANCING ALGORITHM, FIVE-STAGE FCN, AND TRAFFIC #4 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	120.116	19.729	417.707
Modification 1	119.098	13.414	283.588
Modification 2	118.053	17.403	378.419
Modifications 1 & 2	118.958	13.249	280.890

TABLE IX. RESULTS FOR THE BALANCING ALGORITHM, FIVE-STAGE FCN, AND TRAFFIC #4 MODEL.

Algorithms	Maximum (flows)	Variance (flows^2)	Bad Links (links)
Original Version	122.126	23.903	486.419
Modification 1	120.941	15.679	319.221
Modification 2	119.295	20.266	424.823
Modifications 1 & 2	120.899	15.492	315.647

For Traffic #5, Table X shows the results for the rebalancing algorithm, and Table XI shows the results for the balancing algorithm. As shown in the tables, the characteristic for Traffic #5 differs from those for other traffic models. For the case of Traffic #5, modification 2 is more effective for the maximum and bad links metrics than modification 1. The variance metric is smaller for modification 1 than for modification 2, although the difference is almost negligible. When both modifications 1 and 2 were applied, the maximum and bad links metrics are greater than those for the case of applying modification 2. However, every metric becomes smaller by employing both modifications than that by the original version.

TABLE X. RESULTS FOR THE REBALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #5 MODEL.

Algorithms	Maximum (flows)	Variance ($flows^2$)	Bad Links (links)
Original Version	36.486	11.666	122.128
Modification 1	35.885	6.495	60.932
Modification 2	33.308	6.825	37.355
Modifications 1 & 2	35.399	5.973	51.601

TABLE XI. RESULTS FOR THE BALANCING ALGORITHM, THREE-STAGE FCN, AND TRAFFIC #5 MODEL.

Algorithms	Maximum (flows)	Variance ($flows^2$)	Bad Links (links)
Original Version	36.558	11.737	123.221
Modification 1	35.906	6.539	61.762
Modification 2	33.303	6.832	37.560
Modifications 1 & 2	35.498	5.995	51.809

As illustrated in Tables I–XI, the effectiveness of each technique depends on the traffic model. For Traffic #1–#4, modification 1 is more effective. For Traffic #3, modification 2 does not improve the metrics. However, for Traffic #5, modification 2 works very well. When both modifications 1 and 2 were applied for Traffic #1 and #2, most metric values became smaller than those for the case of applying either one of modification 1 or 2. However, even if both modifications were applied, the metric values almost equaled those for the case of applying modification 1 for Traffic #3. Furthermore, when both modifications were applied for Traffic #5, some metric values became greater than those for the case of applying modification 2.

These results suggest that the best performance is obtained by selecting the technique depending on the characteristic of traffic load. For heavy and skewed loads, modification 1 should be used. If the load is light, modification 2 will be a better choice. However, if predicting the load characteristic is difficult, both modifications 1 and 2 should be applied. By applying both modifications, the metrics were considerably improved compared to the original version for every examined traffic model. Thus, applying both modifications is an effective way to improve load equality, although it does not always yield the best results.

D. Discussion

In Section VI.C, the proposed techniques are evaluated through the number of flows that pass through a link. However, it is uncertain whether this advantage in flow number equality directly leads to the improvement of the performance experienced by users. For the clarification of this point, it is necessary to perform additional computer simulation, which precisely models the packet-level behaviors including protocol and queueing processes. Through this simulation, the performance improvement will be confirmed through the metrics such as throughputs and response time, which are experienced by users. Actually, the packet level simulation of the flow diffusion algorithms has been partly done and reported in [32]. In [32], the TCP throughput is measured for the bulk data transfer application. The result shows that the number of flows with small throughputs successfully decreases through equal flow diffusion obtained by the balancing algorithm. This characteristic implies that the proposed techniques will effectively reduce the probability of throughput degradation because the techniques successfully improve the flow number equality.

As a future study, implementation of the rebalancing or balancing algorithm with the proposed techniques may be required to assess the feasibility and advantage of the approach. For implementation, it will be necessary to employ a mechanism that enables flow-based routing, for example, OpenFlow [33]. Additionally, the start and end of a flow must be detected to run the rebalancing algorithm. This detection of flow start/end will be achieved by the techniques presented in [34]. However, further study is necessary to clarify if it is possible to set the routing table entry from a flow detection in a practical processing time.

VII. CONCLUSION AND FUTURE WORK

This paper investigates two techniques to improve the rebalancing algorithm [12], which diffuses flows in an FCN. The first technique decreases the difference between the uplink loads by adding a criterion for determining the middle switch used in the routing or rerouting processes. In addition, the load equality depends on the scanning order of the middle switch indices. Based on this, the second technique determines the start index for scanning to balance the loads. The two techniques were applied to the rebalancing and balancing algorithms and were evaluated using computer simulations. The balancing algorithm is a version of the rebalancing algorithm that is modified to omit the rerouting process. The results demonstrated that the proposed techniques successfully improved load equality.

By expanding the work of a previous study [1], this study examined the application of the techniques to a five-stage FCN. For a five-stage FCN, the upper bound was analyzed for the number of flows on a link on the basis of the upper bound for the three-stage FCN case. To the best of the author's knowledge, this bound has never been reported in the literature. Thus, the derivation of that bound is an important contribution. In addition, computer simulations confirmed that the proposed techniques were effective for the five-stage

FCN case. This effectiveness has also not been reported in previous studies.

As another expansion of [1], performance against a broader range of traffic models was tested via computer simulations. Several of the models employed skewed traffic matrices, for which the source-destination pair of a generated flow was selected from a limited number of candidates. In addition, one model simulated a lighter traffic load than other models. The results demonstrated that the second technique was not effective for a highly skewed traffic matrix. However, the second technique is more effective for light traffic loads. In addition, load equality was improved for every tested traffic model when both proposed techniques were applied. Thus, as an important result, it was found that both techniques should be used independent of traffic loads.

Further study is necessary to determine how the load equality enhanced by the proposed techniques affects packet-level performance, such as packet latency and throughput. To achieve this, a more precise packet-level computer simulation is required. The implementation of the two proposed techniques is also important for future work.

ACKNOWLEDGMENT

This work was supported by JSPS KAKENHI Grant Number JP19K11928. The author would like to thank Enago (www.enago.jp) for the English language review.

REFERENCES

[1] S. Ohta, "Techniques to improve a flow diffusion algorithm for folded Clos networks," The Eighteenth International Conference on Networks (ICN 2019), Valencia, Spain, Mar. 2019, pp. 68–73, ISBN: 978-1-61208-695-8.

[2] A. Singh et al., "Jupiter rising: a decade of Clos topologies and centralized control in Google's datacenter network," The 2015 ACM Conference on Special Interest Group on Data Communication, London, United Kingdom, Aug. 2015, pp. 183–197, ISBN: 978-1-4503-3542-3, doi: 10.1145/2785956.2787508.

[3] Z. Guo and Y. Yang, "On nonblocking multicast fat-tree data center networks with server redundancy," IEEE Trans. on Computers, vol. 64, no. 4, pp. 1058–1073, Apr. 2014.

[4] C. Guo et al., "DCell: a scalable and fault-tolerant network structure for data centers," ACM SIGCOMM 2008 Conference on Data communication, Seattle, WA, USA, Aug. 2008, pp. 75–86, ISBN: 978-1-60558-175-0, doi: 10.1145/1402958.1402968.

[5] C. Guo et al., "BCube: a high performance, server-centric network architecture for modular data centers," ACM SIGCOMM 2009 Conference on Data communication, Barcelona, Spain, Aug. 2009, pp. 63–74, ISBN: 978-1-60558-594-9, doi:10.1145/1592568.1592577.

[6] N. Farrington and A. Andreyev, "Facebook's data center network architecture," 2013 Optical Interconnects Conference, Santa Fe, NM, USA, May 2013, pp. 49–50, ISSN: 2376-8665, doi: 10.1109/OIC.2013.6552917.

[7] A. Greenberg et al., "VL2: a scalable and flexible data center network," Communications of the ACM, vol. 54, no. 3, pp. 95–104, Mar. 2011.

[8] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," ACM SIGCOMM 2008 conference on Data communication, Seattle, WA, USA, Aug. 2008, pp. 63–74, ISBN: 978-1-60558-175-0, doi:10.1145/1402958.1402967.

[9] C. Clos, "A study of nonblocking switching networks," Bell System Technical Journal, vol. 32, no. 2, pp. 406–424, Mar. 1953.

[10] S. Scott, D. Abts, J. Kim, and W.J. Dally, "The BlackWidow high-radix Clos network," The 33rd Annual International Symposium on Computer Architecture (ISCA '06), Boston, MA, USA, June 2006, pp. 16–28, ISBN:0-7695-2608-X, ISSN: 1063-6897, doi: 10.1109/ISCA.2006.40.

[11] E. Zahavi, I. Keslassy, and A. Kolodny, "Distributed adaptive routing for big-data applications running on data center networks," 2012 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS '12), Austin, Tx, USA, Oct. 2012, pp. 99–110, ISBN: 978-1-4503-1685-9.

[12] S. Ohta, "Flow diffusion algorithms based on local and semi-local information for folded Clos networks," The Fourth International Conference on Electronics and Software Science (ICESS 2018), Takamatsu, Japan, Nov. 2018, pp. 46–54, ISBN: 978-1-941968-52-9.

[13] A.Jajszczyk, "Nonblocking, repackable, and rearrangeable Clos networks: fifty years of theory evolution," IEEE Communications Magazine, vol. 41, no. 10, pp.28–33, Oct. 2003.

[14] D. G. Cantor, "On nonblocking switching networks," Networks, vol. 1, no. 4, pp. 367–377, winter 1971.

[15] H.-A. Kim and D. R. O'Hallaron, "Counting network flows in real time," IEEE Global Telecommunications Conference (GLOBECOM 2003), San Francisco, CA, USA, Dec. 2003, pp. 3888–3893, ISBN: 0-7803-7974-8, doi: 10.1109/GLOCOM.2003.1258959.

[16] J. W. Roberts, "Traffic theory and the Internet," IEEE Communications Magazine, vol. 39, no. 1, pp.94–99, Jan. 2001.

[17] L. A. Bassalygo, I. I. Grushko, and V. I. Neiman, "The Structures of One-Sided Connecting Networks," The Sixth International Teletraffic Congress (ITC 6), Munich, Germany, Sept. 1970. Available from https://itc-conference.org/en/itc-library/itc6.html, 2019.11.18.

[18] G. Broomel and J. R. Heath, "Classification categories and historical development of circuit switching topologies," Computing Surveys, vol. 15, no. 2, pp. 95–133, June 1983.

[19] N. Fujii, "Application of a rearrangement algorithm for digital cross-connect system control," The Eighth Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE INFOCOM '89), Ottawa, Canada, Apr. 1989, pp. 228–233, ISBN: 0-8186-1920-1, doi: 10.1109/INFCOM.1989.101458.

[20] M. K. Panda, T. Venkatesh, V. Sridhar, and Y. N. Singh, "Architecture for a class of scalable optical cross-connects," First International Conference on Broadband Networks, San Jose, CA, USA, Oct. 2004, pp. 233–242, ISBN: 0-7695-2221-1, doi: 10.1109/BROADNETS.2004.16.

[21] Y. –K. Chen and C. –C. Lee, "Fiber Bragg grating-based large nonblocking multiwavelength cross-connects," Journal of Lightwave Technology, vol. 16, no 10, pp. 1746–1756, Oct. 1998.

[22] Y. –H. Kao, N. Alfaraj, M. Yang and H. J. Chao, "Design of high-radix Clos network-on-chip," The 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip (NOCS '10), Grenoble, France, May 2010, pp. 181–188, ISBN: 978-0-7695-4053-5 doi: 10.1109/NOCS.2010.27.

[23] A. Zia, S. Kannan, G. Rose and H. J. Chao, "Highly-scalable 3D Clos NOC for many-core CMPs," The 8th IEEE International NEWCAS Conference 2010 (NEWCAS2010), Montreal, QC, Canada, June 2010, pp. 229–232, doi: 10.1109/NEWCAS.2010.5603776.

[24] A. Joshi et al., "Silicon-photonic Clos networks for global on-chip communication," The 2009 Third ACM/IEEE International Symposium on Networks-on-Chip (NOCS '09), , San Diego, CA, USA, May 2009, pp. 124 – 133, ISBN: 978-1-4244-4142-6, doi: 10.1109/NOCS.2009.5071460.

[25] F. Hassen and L. Mhamdi, "A Clos-network switch architecture based on partially buffered crossbar fabrics," 2016 IEEE 24th Annual Symposium on High-Performance Interconnects (HOTI), Santa Clara, CA, USA, Aug. 2016, pp. 45–52, ISSN: 2332-5569, doi: 10.1109/HOTI.2016.020.

[26] F. Hassen and L. Mhamdi, "High-capacity Clos-network switch for data center networks," 2017 IEEE International Conference on Communications (ICC 2017), Paris, France, May 2017, paper NGNI07-1, pp. 1–7, ISSN: 1938-1883, ISBN: 978-1-4673-8999-0, doi: 10.1109/ICC.2017.7997147.

[27] S. Yang, S. Xin, Z. Zhao, and B. Wu, "Minimizing packet delay via load balancing in Clos switching networks for datacenters," 2016 International Conference on Networking and Network Applications (NaNA 2016), Hakodate, Japan, July 2016, pp. 23–28, doi: 10.1109/NaNA.2016.14.

[28] S. Ghorbani, B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Micro load balancing in data centers with DRILL," The 14th ACM Workshop on Hot Topics in Networks (HotNets-XIV), Philadelphia, PA, USA, Nov. 2015, paper 17, ISBN: 978-1-4503-4047-2, doi:10.1145/2834050.2834107.

[29] L. G. Valiant, "A scheme for fast parallel communication," SIAM J. Computing, vol. 11, no. 2, pp. 350–361, May 1982.

[30] S. Ohta, "A simple control algorithm for rearrangeable switching networks with time division multiplexed links," IEEE J. on Selected Areas in Communications, vol. SAC-5, no. 8, pp.1302–1308, Oct. 1987.

[31] ns developers, *ns-3, Network Simulator* [Online], Available from https://www.nsnam.org/, 2019.11.09.

[32] S. Ohta, "TCP throughput achieved by a folded Clos network controlled by different flow diffusion algorithms," International Journal of Information and Electronics Engineering, in press..

[33] O. Cocker and S. Azodolmolky, Software-Defined Networking with OpenFlow - Second Edition: Deliver innovative business solutions, Packt, Birmingham, UK, 2017.

[34] S. Zhu and S. Ohta, "Real-time flow counting in IP networks: strict analysis and design issues," Cyber Journals: Multidisciplinary Journals in Science and Technology, Journal of Selected Areas in Telecommunications, vol. 2, no. 2, pp. 7–17, Feb. 2012.