TCP Throughput Achieved by a Folded Clos Network Controlled by Different Flow Diffusion Algorithms

Satoru Ohta

***Abstract*—A folded Clos network (FCN) is the topology often used for data center networks. Its performance depends on the flow diffusion algorithm executed at the input/output switches. Several algorithms that diffuse flows more equally between links than conventional random routing have been proposed in the past. It is expected that those algorithms prevent the TCP (transmission control protocol) throughput from decreasing due to traffic congestion. However, it is unclear whether these flow diffusion algorithms are significantly effective for avoiding throughput degradation of a TCP flow compared with conventional random routing. This paper investigates the effectiveness of flow diffusion algorithms for TCP throughput through packet-level computer simulation. The results confirm that flow diffusion algorithms can efficiently reduce the number of TCP flows, the throughputs of which are degraded.**

***Index Terms*—Computer simulation, data center network, routing, switching network.**

I. Introduction

Many works have studied data center networks in the past [1], [2], which typically consider a folded Clos network (FCN) as their topology [3]–[11] because of its scalability and high bandwidth. An FCN is based on a three-stage switching network originally presented by Charles Clos in 1953 [12]. An FCN comprises input/output and middle switches. Each input/output switch is connected to every middle switch through a point-to-point link. For data center networks, an FCN must have an appropriate routing algorithm to avoid traffic congestion caused by packet streams concentrating on some links.

Two types of routing methods have been used for FCNs: per-packet routing [5]–[7] and flow-based routing [8]–[11]. This paper focuses on flow-based routing, which has the advantage of easily avoiding packet reordering.

One simple flow-based routing method used in several studies [8], [9], [13] is random routing. With this method, the source input/output switches randomly select a middle switch for transmitting a packet as the next hop. Flow-based random routing can be implemented by a hash function, which maps a flow identifier to an index of a middle switch.

Alternative routing methods have been presented [10]. They were developed to diffuse flows more equally between links as compared to random routing. The simulations showed that these alternative algorithms are superior to random routing in terms of load equality, which is measured by the number of flows on a link [10].

It is rational to assess the effectiveness of the abovementioned algorithms via the number of flows if a data center uses TCP (transmission control protocol) based applications. This is because the throughput of a TCP flow decreases due to the overload caused by an unbalanced traffic distribution. However, it is unclear how the improvements made by the abovementioned algorithms on the flow number equality can significantly affect the throughput of a TCP flow.

This paper demonstrates that the algorithms presented in [10] are advantageous in terms of TCP throughput. To clarify this point, the study employs computer simulation that precisely models communication protocols, including TCP and packet queueing process, on the *ns-3* simulation platform [14]. The simulation compares three algorithms: random routing, the balancing algorithm, and the load sum algorithm. The latter two algorithms were presented by [10]. The results show that the average TCP throughput does not greatly differ among the algorithms. However, flows with low throughput are fewer for the balancing and load sum algorithms than that for random routing. Thus, a user will encounter a degraded connection with a smaller probability if these algorithms are employed. This is because the number of overloaded links becomes smaller with the balancing or load sum algorithm.

The rest of the paper is organized as follows. Section II describes background information regarding FCN. Section III reviews related works. Section IV briefly explains the investigated algorithms and describes the problem. Section V details the method and model of computer simulation. Section VI presents the simulation results. Finally, Section VII concludes the paper.

II. Folded Clos Network

Fig. 1 shows an example of an FCN, which is constructed by folding a three-stage switching network presented by Charles Clos in 1953 [12]. An FCN is typically used as the topology in data center networks. As seen in Fig. 1, an FCN consists of r input/output switches and m middle switches. Each input/output switch has n input and output ports. An input/output switch is connected to every middle switch through an uplink, while a middle switch is connected to every input/output switch through a downlink. The middle switches are indexed as 0, 1, …, $m-1$, whereas the input/output switches are indexed as 0, 1, …, $r-1$. This paper assumes that the FCN handles packet streams. A packet stream is connected from an input port of a source input/output switch to an output port of a destination input/output switch via a middle switch.

In an FCN, a packet can reach its destination from any middle switch because a middle switch is connected to every input/output switch. Thus, it can successfully arrive at its

Manuscript received January 31, 2020; revised March 12, 2020. This work was supported by JSPS KAKENHI Grant Number JP19K11928.

S. Ohta is with Toyama Prefectural University, Kurokawa 5180, Imizu-shi, Toyama, 939-0398 Japan (e-mail: ohta@pu-toyama.ac.jp).

doi: 10.18178/ijiee.2020.10.1.714

destination without depending on the middle switch used as the next hop of the source switch. However, if source switches inappropriately forward packets to middle switches, uplinks or downlinks may be offered excessively heavy load. Such a heavy load may lead to traffic congestion, which can degrade the performance of the network. To avoid congestion, each source switch must run a routing algorithm that diffuses traffic equally. Thus, it is necessary to establish a traffic diffusion algorithm to be executed at a source switch.

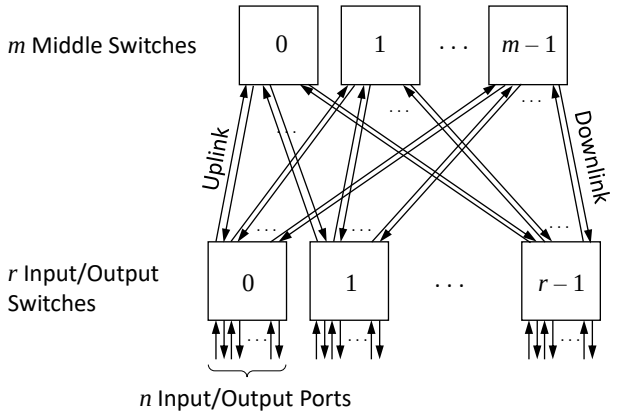

Fig. 1. Example of a folded Clos network (FCN).

This paper assumes that a packet is routed on a flow basis. A flow is defined as a stream of packets identified by a set of fields in the packet header [15]. A field set frequently used includes a 5-tuple of source address, destination address, protocol, source port, and destination port, which is associated with an IP (internet protocol) socket. For TCP-based applications, a flow is associated with a TCP connection. By routing packets on a flow basis, packet reordering is avoided, which is favorable as packet reordering degrades the TCP's performance.

III. Related Work

Various studies on FCNs have been done for the data center network application [4]–[11]. From the aspect of routing techniques, some studies examined per-packet routing methods [4]–[6]. Per-packet routing determines the next hop of a packet on a packet-by-packet basis. Thus, packets of a flow may pass through routes with different delays in an FCN. This causes packet reordering. Packet reordering will degrade the network performance, particularly when applications run on the TCP protocol. It is reported that 0.02 % of packets are delivered out of order by the method of [6]. However, it is uncertain whether this reordering rate is sufficiently low to avoid performance degradation.

Packet reordering does not occur for per-flow routing, for which every packet of a flow passes through the same route. By equally diffusing flows in an FCN, traffic congestion is avoided with preventing packets from being delivered out of order. A simple flow diffusion method is random routing, which randomly selects the middle switch used as the next hop for a flow. Examples of random routing can be found in [8], [9], [13]. This method can be implemented by using a hash function, which maps a number associated with a flow to an index of a middle switch. This approach equally distributes the average number of flows that pass through a link. However, the number of flows may become large for some links in the network with a substantial probability [10]. Flows that pass through a heavily loaded link suffer from performance degradation, including a decrease of throughput or an increase of latency.

Reference [10] presented algorithms that allow more equal diffusion of flows between links than random routing; these are the rebalancing algorithm and the load sum algorithm. A notable feature of these algorithms is the existence of a theoretical upper bound on the number of flows passing through a link, which ensures that the number of flows does not become excessively large. Reference [10] also investigated a method called the balancing algorithm, which is a version of the rebalancing algorithm and does not reroute existing flows. Through computer simulation, it was shown that these algorithms outperform conventional random routing in terms of several statistical metrics associated with the equality of the number of flows.

Reference [11] showed that the rebalancing and balancing algorithms in [10] can be further improved in terms of load equality by employing two modifications: uplink load balancing and an appropriate choice of the start index for scanning middle switches. These modifications do not affect the theoretical upper bound of the number of flows passing through a link. The improvement was confirmed via computer simulation.

In [10] and [11], the algorithms were assessed in terms of the number of flows without considering the individual bandwidth of each flow. This is rational to some extent for frequently used TCP-based applications because a TCP flow is elastic [16]. For elastic flows, the average flow throughput is limited by the link bandwidth and the number of flows that commonly share a link. Thus, it is considered that the TCP throughput will be improved by equally diffusing the number of flows between links.

While [10] and [11] conducted flow-level computer simulation, [17] performed packet-level simulation to compare different topologies employed in data center networks. In [17], packet latency and throughput were evaluated with using the *ns-2* [18] simulation platform. It is expected that this approach is also effective to evaluate the routing algorithms presented in [10] and [11].

IV. Investigated Methods and Problem Description

This study compares the following flow diffusion algorithms.

- Random routing
- Balancing algorithm
- Load sum algorithm

Random routing randomly selects the middle switch used by a flow, which is the method commonly used in previous studies.

The balancing algorithm is outlined as follows. Let variable $F(i, j, k)$ denote the number of flows set from the source switch i $(0 \le i \le r-1)$ to the destination switch k $(0 \le k \le r-1)$ via the middle switch j $(0 \le j \le m-1)$. Then, suppose that a flow is newly generated between the source switch i and the destination switch k. For this situation, the

algorithm selects the middle switch J $(0 \le J \le m-1)$ for the new flow such that $F(i, J, k)$ is the minimum among $F(i, 0, k)$, $F(i, 1, k)$, …, $F(i, m-1, k)$. Meanwhile, the load sum algorithm utilizes the uplink load $U_{i,j}$ and the downlink load $D_{j,k}$. $U_{i,j}$ is defined as the number of flows from the source switch i to the middle switch j, while $D_{j,k}$ is defined as the number of flows from the middle switch j to the destination switch k. Then, the middle switch J for a newly generated flow is determined such that the sum $U_{i,J} + D_{J,k}$ is the minimum among $U_{i,j} + D_{j,k}$'s.

A flow-level assessment of these algorithms was previously performed [10], showing that the balancing and load sum algorithms outperform random routing. Additionally, the load sum algorithm diffuses traffic more equally than the balancing algorithm.

Among data streams exchanged in a data center network, TCP flows are crucial because many important application-layer protocols rely on TCP. As mentioned in Section III, equality in the number of flows will increase TCP throughput. However, it is unclear how the difference in load equality estimated by the flow number affects TCP throughput. Therefore, this paper focuses on the throughput of a TCP flow (or connection) in an FCN and examines its dependence on flow diffusion algorithms. This is achieved by computer simulation, which precisely models the packet queueing mechanism and TCP/IP protocol.

V. Computer Simulations

Packet-level computer simulation was performed to evaluate the effect of flow diffusion algorithms on TCP throughput using the software products developed in [10] and [11]. That is, in those studies, the processes of generating flows and determining the routes of flows were simulated through the custom programs. This study employed slightly modified versions of these programs to generate flows and make a decision of their routes. The simulation program considers each generated flow as a TCP connection. Through the connection, bulk data are transmitted with the control by TCP. Specifically, the simulation is performed through the following three phases.

Phase 1: In this phase, flows are randomly generated. The program determines the hosts, between which a TCP connection is set up. It also decides the start and end time of the connection.

Phase 2: This phase decides which middle switch is used for each connection. This decision is made according to the evaluated flow diffusion algorithms.

Phase 3: For each connection generated in Phase 1, bulk data are transmitted between the hosts. Packets are routed to the middle switch determined in Phase 2. Then, the amount of transmitted data and the throughput are measured for each TCP connection.

As stated above, Phases 1 and 2 were performed using modified versions of the custom programs developed in [10] and [11]. Outputs from the Phase 1 and 2 programs were fed to the Phase 3 program via a text file. The file created by the Phase 1 program lists the following flow information: flow identifier, start time, end time, and two hosts. It also includes the dimension of the investigated FCN denoted by m, n, and r. Meanwhile, the Phase 2 program outputs the file that lists the following: flow identifier, middle switch used by the flow, and time of setting the route.

The Phase 2 program was written in three versions, each of which runs an evaluated algorithm. For the balancing algorithm, the modifications presented in [11] were applied, which include uplink load balancing and an appropriate choice of the start index for scanning middle switches. For the load sum algorithm, the start index of scanning middle switches was determined similarly to the balancing algorithm.

The simulation program of Phase 3 was newly developed in this study. The program was built on the *ns-3* network simulation platform [14] that enables precise simulations of the protocols, including TCP and the packet queueing process.

The Phase 3 program was built on the standard API (application program interface) provided by *ns-3*. Firstly, it creates the specified number of switches and hosts by using the node model. Then, the FCN is constructed by connecting the nodes using the point-to-point link model.

To simulate the TCP-based application, the bulk transfer model of *ns-3* was employed. According to the Phase 1 output, the application server (sink) and client (sender) are set up in the hosts. The client application is set to begin and stop data transmission at the start and end time of the flow, respectively. The amount of data transferred is obtained as the statistic of the server. Furthermore, the flow throughput is obtained by dividing the data amount by the duration of the flow.

The route of a flow was set by using the fixed routing model of *ns-3*. However, the *ns-3* model routes a packet according to its destination address, which means that it is difficult to route packets using the flow identifier and 5-tuple of protocol, source/destination addresses, and source/destination ports. This difficulty can be avoided by assigning distinct destination hosts to simultaneously existing flows. Then, a host becomes the destination of a unique flow during its existing time period. Thus, the route associated with a flow is successfully simulated by the *ns-3* standard routing mechanism, which is based on the destination address. This setting was achieved by modifying the Phase 1 program such that a host will not handle two or more concurrent flows.

To assign distinct hosts to simultaneously existing flows, there must be a sufficiently large number of hosts. If more flows simultaneously exist, then more hosts are needed. Since a host is connected to an input/output switch via a point-to-point link, the value of parameter n must be large to supply more hosts. Thus, the parameter n was chosen to be sufficiently large to provide distinct hosts for simultaneously existing flows.

The throughput was measured for flows that exist during the equilibrium by the following steps. Suppose that the system is in a stable state from time t_0 to t_1. Then, the throughput of a flow was measured if its start time is earlier than t_1 and its end time is later than t_0.

In the simulations, the parameters m and r were both set to 8. The parameter n was set to 20 for Traffic #1 (light traffic) and was set to 50 for Traffic #2 (heavy traffic). Thus, the scale of the FCN is not very large. This is the case because simulations on *ns-3* are very time-consuming and evaluation

on a large FCN is not practical. To decrease the computational time of *ns-3*, the number of packet-related events must be kept low. Hence, the link capacity between switches was set to 10 Mb/s and the MTU (maximum transmission unit) size was set to 10000 Bytes; note, however, that these values may not be very realistic for data centers. The link capacity was 100 Mb/s between hosts and input/output switches. The capacity was set higher for this link because the purpose of the simulation is to assess the effect of traffic congestion on links between input/output and middle switches. Thus, the link capacity was set higher between hosts and input/output switches to not influence the performance determined by the load between input/output and middle switches. The simulation model is illustrated in Fig. 2.

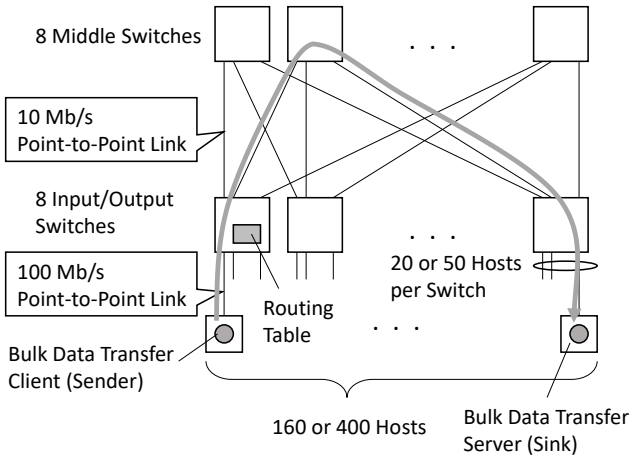

Fig. 2. Simulation setting.

In Phase 1 of the simulation, flows were generated with a random interval according to an exponential distribution with an average of T_0. The duration of a flow was also randomly determined according to an exponential distribution with an average of T_1. For a flow, two different source/destination switches were randomly selected, and the hosts connected to the switches were chosen. Let *a* and *z* denote the selected hosts. For these hosts, bulk data are transmitted from *a* to *z* as well as from *z* to *a*. Therefore, two TCP connections are established between hosts *a* and *z*. This process was repeated 8,000 times. Thus, the number of generated TCP connections is 16,000 for each data set.

To assess the dependence of the performance of the algorithms on traffic volume, two traffic models Traffic #1 and #2 were used for testing, which represent light and heavy traffic load cases, respectively. The parameter T_0 was set to 0.5 s for both models, while T_1 was set to 160 s for Traffic #1 and 320 s for Traffic #2. For Traffic #1, the expected average number of bulk data streams in equilibrium is 640 ($=2 \times 160/0.5$). Since the number of uplinks (or downlinks) is $mr = 64$, there is an average of 10 bulk data streams on a 10 Mb/s link. Similarly, the average number of bulk data streams is 20 on a link for Traffic #2.

For the above traffic models, it was found that the system is in a stable state from 500 to 4000 s. This is demonstrated in Fig. 3 that plots the average number of flows on a link versus simulation time for Traffic #1. Note that every measured flow should start and stop within the stable state period. Thus, the measurement start time t_0 was set to 2000 s, while the stop time t_1 was set to 3000 s.

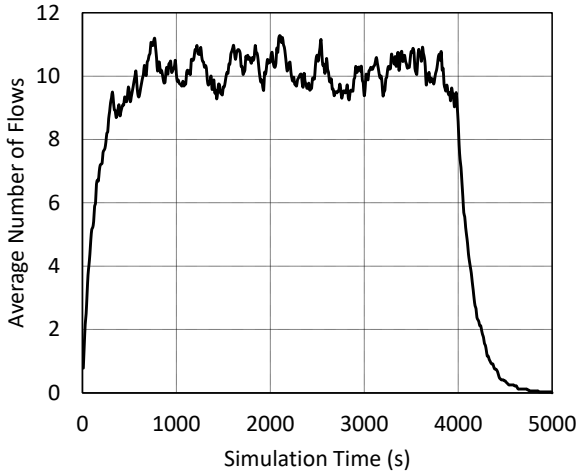

Fig. 3. Example of the average number of flows passing a link versus simulation time for Traffic #1.

For each traffic model, four sets of 16,000 TCP connections were generated by feeding different random seeds to the Phase 1 program. The statistics of flows were calculated from the outputs of the Phase 3 program for the four flow sets.

VI. Evaluation Results

Firstly, the average throughput of a flow is compared between the algorithms for Traffic #1 and #2 in Table I. As shown in Table I, the average throughput for the load sum algorithm is the largest, while that for random routing is the smallest for Traffic #1. However, the difference between the throughputs for both algorithms is not large. The difference in the average throughput between the load sum algorithm and the random routing is as small as 37.4 kb/s for Traffic #1. For Traffic #2, the difference in the average throughput among all algorithms is smaller and almost negligible. Therefore, it is concluded that the difference in the average throughput is not significant.

TABLE I: Average Throughput

Traffic Model	Random Routing	Balancing Algorithm	Load Sum Algorithm
Traffic #1	883.5 kb/s	901.1 kb/s	920.9 kb/s
Traffic #2	470.6 kb/s	475.0 kb/s	474.6 kb/s

Although the balancing and load sum algorithms are not necessarily advantageous in terms of the average throughput, their superiority becomes clear in terms of the distribution of the flow throughput, as shown in Figs. 4 and 5. In both figures, the *x*-axis shows the flow throughput, while the *y*-axis shows the cumulative percentage of the number of flows; i.e., the throughputs are equal to or less than *x* kb/s for *y*% of flows. Fig. 4 shows the result for Traffic #1, while Fig. 5 shows the result for Traffic #2.

Figs. 4 and 5 show that the balancing and load sum algorithms decrease the number of flows with smaller throughputs in comparison with random routing. This characteristic is observed in Fig. 6, which shows a zoomed-in version of Fig. 4 for small throughputs.

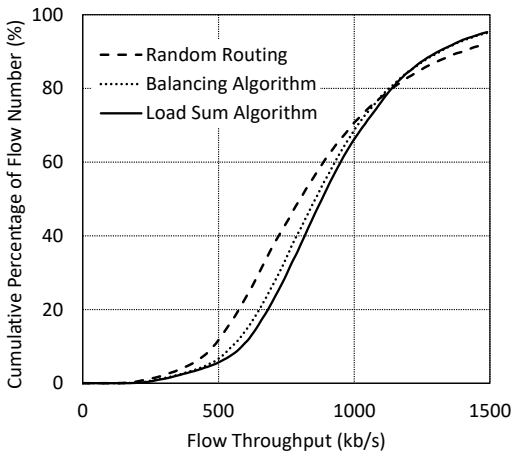

Fig. 4. Cumulative percentage of the number of flows versus throughput for Traffic #1.

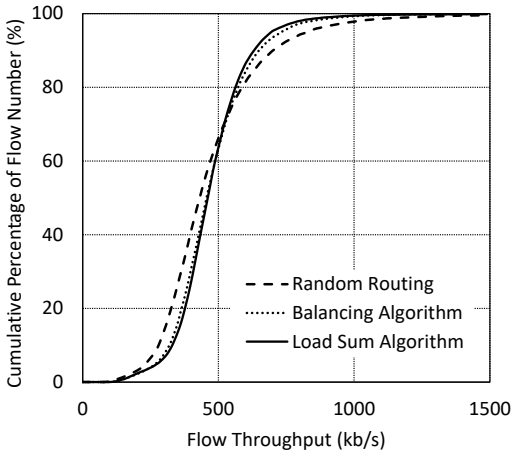

Fig. 5. Cumulative percentage of the number of flows versus throughput for Traffic #2.

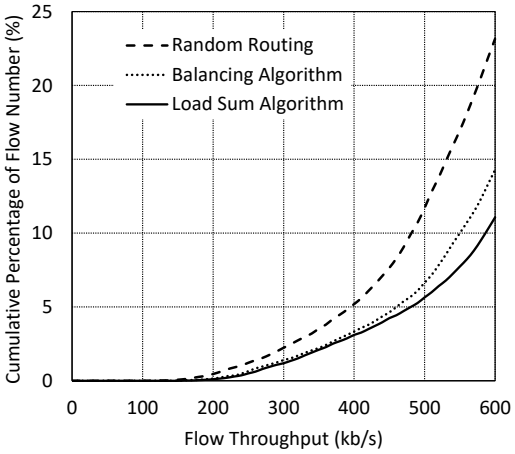

Fig. 6. Cumulative percentage of the number of flows versus throughput for Traffic #1 for small throughputs.

Assume that the threshold of unacceptable throughput is 600 kb/s for Traffic #1. Hence, from Fig. 6, the ratio of flows with unacceptably low throughput is 23% for random routing and decreases to 14% for the balancing algorithm. However, when the load sum algorithm is employed, the ratio decreases further to 11%, which is less than half of that for random routing. Similarly, assume that the threshold of unacceptable throughput is 300 kb/s for Traffic #2 because the load is twice as heavy as Traffic #1. Then, the ratios of flows with unacceptably low throughput are 14% for random routing, 7.6% for the balancing algorithm, and 6.6% for the load sum algorithm. Thus, the tendency of flows with low throughput does not greatly change with traffic volume.

From the above results, it is concluded that the balancing and load sum algorithms can effectively decrease TCP flows with small throughputs; note that the load sum algorithm is slightly superior. However, as pointed out in [10], the implementation of the load sum algorithm is more difficult because of the communication required between switches. Thus, the employment of the balancing algorithm is a more practical solution.

VII. Conclusion

This paper assessed the TCP connection throughput of flow diffusion algorithms from [10] for FCNs. The investigated algorithms were the balancing algorithm, the load sum algorithm, and conventional random routing. They were tested using packet-level computer simulation performed on the *ns-3* simulator to precisely model the protocols and the packet queueing process. Flow throughput was measured in terms of the duration and byte amount of bulk data transmission. The simulation results showed that the average throughput of a TCP connection does not significantly differ among the algorithms. However, it was clarified that the balancing and load some algorithms can effectively reduce the number of degraded TCP connections, the throughputs of which are unacceptably decreased by traffic congestion. Thus, it is concluded that the flow number equality provided by these flow diffusion algorithms can lead to less degradation and high-quality TCP services.

As future work, assessment for broader conditions will be necessary, including simulations for larger FCNs, heavier/lighter traffic load, and different traffic matrices.

Conflict of Interest

The author declares no conflict of interest.

Author Contributions

S. Ohta did all the tasks related to this paper.

Acknowledgment

The author would like to thank Kengo Maeda for his help in the computer simulation. The author would like to thank Enago (www.enago.jp) for the English language review.

References

[1] W. Xia, P. Zhao, Y. Wen, and H. Xie, "A survey on data center networking (DCN): Infrastructure and operations," *IEEE Communication Surveys & Tutorials*, vol. 19, no. 1, pp. 640–656, First Quarter 2017.

[2] A. Akella, T. Benson, B. Chandrasekaran, C. Huang, B. Maggs, and D. Maltz, "A universal approach to data center network design," in *Proc. 2015 International Conference on Distributed Computing and Networking (ICDCN '15)*, Jan. 2015, paper 41.

[3] N. Farrington and A. Andreyev, "Facebook's data center network architecture," in *Proc. 2013 Optical Interconnects Conference (OI 2013)*, May 2013, pp. 49–50.

[4] A. Singh *et al.*, "Jupiter rising: A decade of Clos topologies and centralized control in Google's datacenter network," in *Proc. the 2015 ACM Conference on Special Interest Group on Data Communication*, Aug. 2015, pp. 183–197.

[5] F. Hassen and L. Mhamdi, "A Clos-network switch architecture based on partially buffered crossbar fabrics," in *Proc. 2016 IEEE 24th Annual Symposium on High-Performance Interconnects (HOTI)*, Aug. 2016, pp. 45–52.
[6] S. Yang, S. Xin, Z. Zhao, and B. Wu, "Minimizing packet delay via load balancing in Clos switching networks for datacenters," in *Proc. 2016 International Conference on Networking and Network Applications (NaNA 2016)*, July 2016, pp. 23–28.
[7] S. Ghorbani, B. Godfrey, Y. Ganjali, and A. Firoozshahian, "Micro load balancing in data centers with DRILL," in *Proc. the 14th ACM Workshop on Hot Topics in Networks (HotNets-XIV)*, Nov. 2015, paper 17.
[8] 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.
[9] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," in *Proc. ACM SIGCOMM 2008 Conference on Data Communication*, Aug. 2008, pp. 63–74.
[10] S. Ohta, "Flow diffusion algorithms based on local and semi-local information for folded Clos networks," in *Proc. 4th International Conference on Electronics and Software Science (ICESS2018)*, Nov. 2018, pp. 46–54.
[11] S. Ohta, "Techniques to improve a flow diffusion algorithm for folded Clos networks," in *Proc. the 18th International Conference on Networks (ICN 2019)*, Mar. 2019, pp. 68–73.
[12] C. Clos, "A study of nonblocking switching networks," *Bell System Technical Journal*, vol. 32, no. 2, pp. 406–424, Mar. 1953.
[13] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The BlackWidow high-radix Clos network," in *Proc. the 33rd Annual International Symposium on Computer Architecture (ISCA '06)*, June 2006, pp. 16–28.
[14] Ns developers, ns-3 | a discrete-event network simulator for internet systems. [Online]. Available: https://www.nsnam.org/
[15] H. A. Kim and D. R. O'Hallaron, "Counting network flows in real time," in *Proc. IEEE Global Telecommunications Conference (GLOBECOM 2003)*, Dec. 2003, pp. 3888–3893.
[16] J. W. Roberts, "Traffic theory and the Internet," *IEEE Communications Magazine*, vol. 39, no. 1, pp. 94–99, Jan. 2001.
[17] A. F. Abdulhameed and R. E. Ahmed, "Performance evaluation of data center network topologies via NS-2 simulations," *International Journal of Computing and Digital Systems*, vol. 8, no. 6, pp. 625–635, Nov. 2019.
[18] Ns developers. The Network Simulator - ns-2. [Online]. Available https://www.isi.edu/nsnam/ns/.

Satoru Ohta was born in Yokosuka, Japan on June 11, 1958. He received the B.E., M.E., and Dr. Eng. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 1981, 1983, and 1996, respectively.

In 1983, he joined NTT, where he worked on the research and development of cross-connect systems, broadband ISDN, network management, and telecommunication network planning. Since 2006, he has been a professor in the Faculty of Engineering at Toyama Prefectural University, Imizu-shi, Japan. His current research interests include network performance evaluation, ad hoc networks, and optimization of data center facilities.

Prof. Ohta is a member of the IEEE, IEICE, and ITE. He received the Excellent Paper Award in 1991 from IEICE. He also received the Best Paper Award at the 2019 International Conference on Advanced Technologies for Communications. He served as a technical program committee member for several international conferences.