Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

FleX: A Flex Interconnected HPC System with Stochastic Load Balancing Scheme

MINSU JEON[*], KYUNG-NO JOO[*], TAEWOO KIM, SEONGHWAN KIM, (Member, IEEE), AND CHAN-HYUN YOUN., (Senior Member, IEEE)

School of Electrical Engineering, Korea Advanced Institute of Science and Technology, Daejeon 34141, South Korea

Corresponding author: Chan-Hyun Youn (e-mail: chyoun@kaist.ac.kr).

This work was supported in part by the National Research Council of Science Technology (NST) grant by the Korea government (MSIP) (No. CAP-17-03-KISTI) and Samsung Electronics Co., Ltd. *([*]Minsu Jeon and Kyung-no Joo contributed equally to this work.)*

ABSTRACT We propose a new low-diameter interconnection network called FleX, which offers high flexibility when installing interconnections in a HPC system. FleX consists of multiple layers with only connections between neighboring layers and not within each layer. These structural properties make it easy to achieve a low diameter with regardless of the scale. The cross-like connections between the adjacent layers in FleX impart various alternative minimal paths, allowing FleX to have high resiliency and a wide bisection width. We also discuss the minimal routing scheme and a stochastic load balancing scheme (LBR) for the proposed interconnection network. Through cycle-based simulations, the performance of FleX is evaluated, and the cost and power consumption analyses in comparison with other interconnection networks are also conducted. We verify that FleX has high configuration flexibility with regard to cost and performance, and also provides low latency and high saturation throughput with the same cost over the legacy interconnection networks for the HPC system. Moreover, being synergied with the proposing LBR, we also verify that FleX can expand its saturation throughput further while only sacrificing the latency slightly.

INDEX TERMS Network topology, parallel computing system, routing algorithm.

I. INTRODUCTION

VARIOUS interconnection networks [1]–[7] that interconnect computing resources for the high performance computing (HPC) systems have been introduced to utilize computing resources more effectively with several packet-processing schemes [8]–[10]. Such networks has been adopted in HPC systems globally [11]–[19]. Fat tree [5], [20], for example, has been widely used among Top500 HPC clusters [21], proving its effectiveness given its use in practical HPC systems [11]–[13], [16]. Although fat tree is a classic example, it also has several drawbacks, such as an increased network diameter and exponential growth of the top-level router load, when used in the construction of a large-scale multi-level interconnection network.

Some studies [22]–[25] have found that an equality network can improve latency and throughput performance outcomes over a 2-tier fat tree system, though this type of network still requires the huge number of inter-router links. Dragonfly [1] proposed a virtual-router idea in which serveral routers constitute a single group and are considered as one high-radix virtual router. Although this idea overcomes the issue of an excessive load on a single router, the network becomes more vulnerable to inter-group packet congestion. Slim Fly [4] has proposed what is known as the near Moore Bound (MB) network configuration, which has a specific and complicated algorithm to choose the router radix and terminal concentration for each router. For this reason, Slim Fly topology cannot readily support various numbers of terminals (end nodes). Therefore, although Slim Fly can construct a near MB network with a diameter of only two (or three), full realization or changing the scale of the network is challenging.

In this paper, we propose a novel FleX interconnection network for a HPC system. FleX suggests a layered architecture in which each layer on the z-axis $(z = 0, 1, ..., N_z - 1)$ consists of $N_x \times N_y$ routers. Each routers on a single layer can be coordinated with 3 basis points (x, y, z). Connections between routers are established when two routers on an adjacent layer share only one between the x and y coordinates. This results in cross-like connections. Cross-like connections

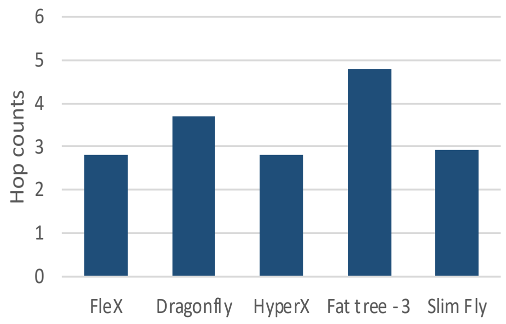


FIGURE 1. Average hop count over interconnection networks on random uniform traffics using minimal routing.

provide many alternative paths between any routers and thereby guarantee a wide bisection width and high fault tolerance. FleX has no limits on the number of layers in general; however, we focus on a three-layered FleX network (diameter 2) in this paper. By doing so, the proposed FleX network has the lowest network diameter of two, making it essentially identical to Slim Fly, though it can be realized more easily in various environments. Fig. 1 explains the motivation behind FleX by comparing the average hop count of interconnection networks with a total terminal size of 1,000 on a random uniform traffic pattern with minimal routing. The configurations for each of the networks in Fig. 1 are fully described in Table 6 and in Section IV-B.

As the proposed FleX contains the various detouring paths without increasing the hop count, it can greatly extend its saturation throughput under adaptive routing. Accordingly, we also propose a new stochastic load balancing routing scheme (LBR) that can aptly utilize the considerable diversity of alternative paths for FleX to extend the saturation throughput.

Throughout this paper, details of the proposed FleX topology are presented, as are the analyses on the structural characteristics of FleX and the minimal routing scheme for FleX. Furthermore, cost, power consumption, and latency performance evaluations are conducted while comparing FleX to four conventional interconnection networks. The experimental results demonstrate that the proposed FleX interconnection system is robust against serious link failures (a nearly 5% enhancement over HyperX at a similar cost) and has the advantages of cost efficiency and greater energy efficiency compared to other interconnection networks. Moreover, FleX can be flexibly configured such that can easily adapt to various cost and performance constraints while also showing lower latency (similar to HyperX with diameter 2) and higher levels of saturation throughput (largely extended by adaptive routing; more than a $\times 2$ extension by the proposed LBR over the minimal routing) compared to other interconnection networks.

In the following section, first we examine the conventional interconnection networks, in Section II. In Section III, details of the proposed interconnection network model and the load balancing scheme are presented. Finally, the simulation-based evaluation of the proposed interconnection network model and load balancing scheme are presented in Section IV, mainly focusing on the configuration cost, power consumption and latency performance.

II. PREVIOUS WORKS

The proposed FleX interconnection system is mainly motivated from two brilliant interconnection networks: Dragonfly and HyperX. Thus, in order fully to deliver the concept of the proposed FleX network, certain aspects of dragonfly and HyperX topologies should be introduced in that they motivated our work. Therefore, in this section, we provide information about some of the key aspects of these two interconnection networks.

A. DRAGONFLY

Dragonfly [1] is a three-level hierarchical network with the *router* at the bottom, *group* in the middle, and *system* on the top. Instead of connecting routers directly with *system*-level global channels, dragonfly suggested a *group*-level, which acts as a high-radix router with interconnections via global channels, in a *system*. The basic configuration of the dragonfly topology is as follows:

Step 1. A system consists of g groups.
Step 2. Groups are connected to each other via single global channel.
Step 3. In each group there are a routers.
Step 4. Routers are fully connected within a group via local channels.
Step 5. Routers are connected to other groups via h global channels.
Step 6. Each router is connected to p terminals.

While physical routers have a radix of $k = p + a + h - 1$, groups in the resulting network act as virtual global routers with a radix of $k' = a(p + h)$. From a system-level perspective, virtual routers are fully connected. Therefore, dragonfly achieves a very low global diameter, allowing this network significantly to reduce the usage of long and expensive global channels compared to a network built directly from radix k routers, thereby reducing the overall construction cost.

Dragonfly essentially has no restrictions on its configuration if the above fundamental requirements are met. However, it is recommended to meet $a = 2p = 2h$ for balancing the channel load. Such recommendation comes from the ratio of the local and global channels through which a packet traverses and can be considered as a mild restriction on the configuration in order to guarantee the maximum performance of the network. Hence, there are limited candidates with regard to the number of end nodes when constructing a full-capacity dragonfly network.

B. HYPERX

HyperX [2] topology is an interconnection network in which routers are directly connected to each other. The basic configuration of the HyperX topology is as follows:

Step 1. Routers are assigned a L-dimensional integer coordinate $\mathbf{I} = (I_1, ..., I_L)$.
Step 2. The k-axis value $I_k(k = 1, ..., L)$ can have values from 0 to upper limit $S_k - 1$, so the entire network consists of $P = \Pi_{k=1}^{L} S_k$ routers.
Step 3. Connection is established when coordinates of two routers I^a, I^b satisfy the following conditions :
- $I_k^a \neq I_j^b \quad if, \quad k = j$
- $I_k^a = I_j^b \quad if, \quad k \neq j$

Step 4. Each router is connected to $\mathbf{T}$ terminals.

As there is no limit on the number of dimensions L, or on the width of each dimension S_k, HyperX topology can be applied technically to any number of terminals or routers. Further, even with the same number of terminals, HyperX can provide numerous configurations.

Although the HyperX topology can be readily adapted to various environments given its flexibility, the performance of each configuration differs. Thus, criteria for finding the best possible configuration are suggested. For a given maximum router radix R, the total number of wire connections for each router should not exceed R, where K_k denotes the number of wires connecting each router in dimension k, with

$$T + \Sigma_{k=1}^{L} K_k(S_k - 1) \leq R. \tag{1}$$

The number of terminal connections should be at least identical to the number of terminals but cannot exceed the maximum number of terminals 2^{R-1}

$$N \leq TP = T(\Pi_{k=1}^{L} S_k) \leq 2^{R-1}. \tag{2}$$

Given the desired relative bisection bandwidth B, the network should have a larger relative bisection bandwidth β

$$\beta \equiv \frac{\min(K_k S_k)}{2T} \geq B. \tag{3}$$

Satisfying the criteria above ((1),(2),(3)), one can find the suitable configuration for the given environment by searching for a feasible configuration with the lowest cost and diameter. However, it should be considered that a trade-off relationship between the cost and diameter, in which the cost increases with an increase in the number of channels as the diameter, or the dimension, decreases, and the cost decreases as the diameter increases.

III. FLEX TOPOLOGY MODEL

As described in Section II, dragonfly tried to reduce the network building cost by reducing the number of global channels. However, as the constraint for constructing the network is added, the number of available network configurations became limited. On the other hand, HyperX can be configured with less constraints, however, as the connections between each router in each dimension are fully connected, it is practically hard to expand the scale from the initially installed interconnect system. Taking these advantages and disadvantages of each other topologies into account, we propose a new network topology that can be flexibly configured with the low cost, and can achieve better performance (*i.e.,* low latency and high saturation throughput) on the same cost.

The detail description and analysis about FleX is presented in this section, and the symbols used in the description are listed in Table 1.

TABLE 1. Symbols for FleX description.

Symbol	Explanation
N	Number of terminals in the network
p	Number of terminals connected to each router
N_x	Number of routers in x-axis on each layer
N_y	Number of routers in y-axis on each layer
N_z	Number of layers
k	Radix of the router

A. TOPOLOGY DESCRIPTION

FleX mainly consists of multiple layers with $N_x \times N_y$ arrays of 2-dimensional logical plane as shown in Fig. 2 (at next page). Each layer in FleX is connected with neighboring layers as a ring. A router in a specific position is denoted as (x, y, z) where $x \in X, y \in Y, z \in Z$ and $X = \{0, 1, ..., N_x - 1\}, Y = \{0, 1, ..., N_y - 1\}, Z = \{0, 1, ..., N_z - 1\}$ as shown in Fig. 2.

For all each router in (x, y, z) where $\forall x \in X, \forall y \in Y, \forall z \in Z$, connections to all possible routers in (x', y', z') that meet one of the following conditions are established.

$$z' = (z + 1) \bmod N_z, y' = y, \forall x' \in X \text{ s.t. } x' \neq x \tag{4}$$

$$z' = (z + 1) \bmod N_z, x' = x, \forall y' \in Y \text{ s.t. } y' \neq y \tag{5}$$

Equation (4) means that routers in the next layer with having same y-axis value and different x-axis value are connected to make links to next layer's x-axis direction. Likewise, Equation (5) means that routers in the next layer with having same x-axis value and different y-axis value are connected to make links to next layer's y-axis direction. In addition, p terminals are connected to each router.

The following equations present the topology with a graph $G = (V, E)$ as a pair of set V for vertices (routers) and set E for edges (links). The i-th router in position (x, y, z) is denoted as vertex $v_i = (x, y, z)$, and the link between router $v_a = (x, y, z)$ and router $v_b = (x', y', z')$ is denoted as edge $e_j = (v_a, v_b) = ((x, y, z), (x', y', z'))$. L_z is 2 if $N_z = 2$, and otherwise $L_z = N_z$.

$$\begin{aligned} V &= \{v_i | 0 \leq i \leq (N_x N_y N_z - 1)\} \\ &= \{v_i = (x, y, z) | i = N_x N_y \cdot z + N_x \cdot y + x, \\ &\quad x \in X, y \in Y, z \in Z\} \end{aligned} \tag{6}$$

$$\begin{aligned} E &= \{e_j | 0 \leq j \leq (N_x N_y L_z (N_x + N_y - 2) - 1)\} \\ &= \{((x, y, z), (x', y', z')) | x \in X, y \in Y, z \in Z, \\ &\quad z' = (z + 1) \bmod N_z, \\ &\quad (x' \in X \text{ s.t. } x' \neq x, y' = y) \vee (y' \in Y \text{ s.t. } y' \neq y, x' = x)\} \end{aligned} \tag{7}$$

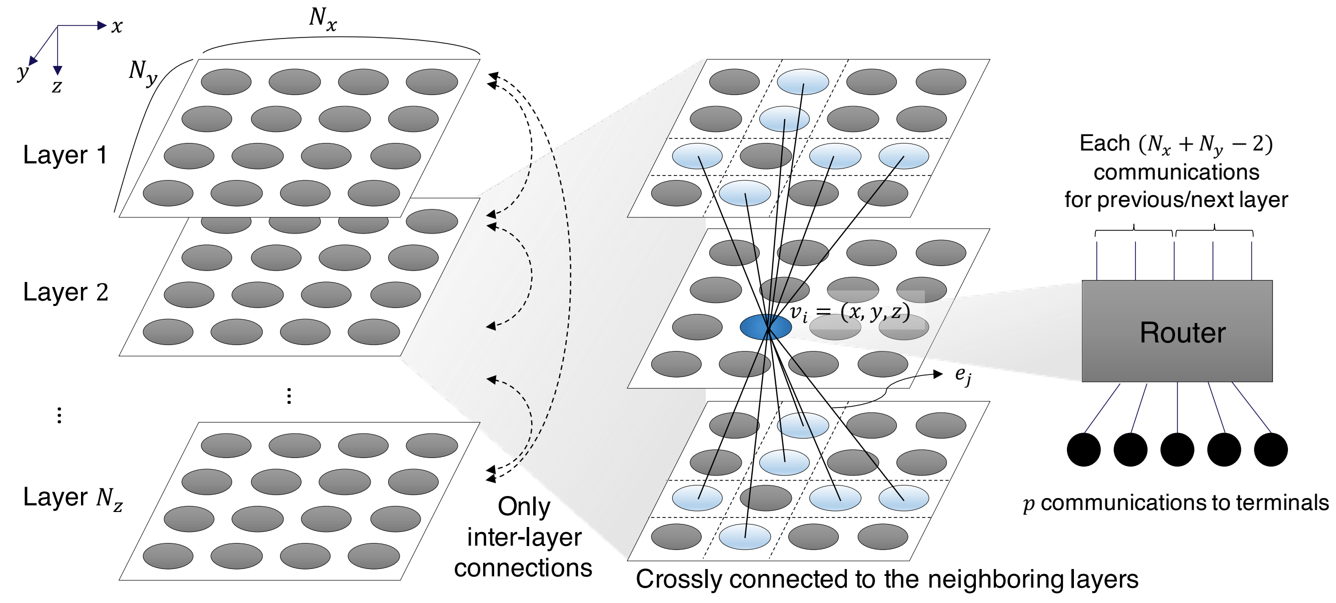

x
y
z
N_x
N_y
Layer 1
Layer 2
Layer N_z
Only inter-layer connections
$v_i = (x, y, z)$
e_j
Crossly connected to the neighboring layers
Each $(N_x + N_y - 2)$ communications for previous/next layer
Router
p communications to terminals

FIGURE 2. General structure of the proposed FleX interconnecting network and its detail description for connections between inter layers.

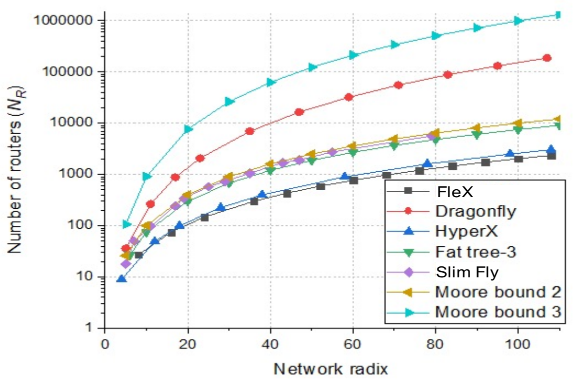


FIGURE 3. Scalability comparison with regards to the network radix.

The topology has the $(N_x \times N_y)N_z$ routers in total and $N_xN_yN_zp$ terminals in total. For each router, it has p links with terminals, $(N_x - 1)$ links with next layer's x-axis direction, and $(N_y - 1)$ links with next layer's y-axis direction. In case of $N_z \geq 3$, each router also has $(N_x - 1)$ links with previous layer's x-axis direction and $(N_y - 1)$ links with previous layer's y-axis direction. Therefore, in 2-layer structure, radix k for the router is $(N_x + N_y + p - 2)$, and otherwise $(2N_x + 2N_y + p - 4)$. In addition, as each router has $(k - p)$ links with other routers and there are $(N_xN_yN_z)$ routers in total, the number of links between each other router is $N_xN_yN_z(k - p)/2$.

Fig. 3 shows the scalability comparisons over the various topologies and Moore bounds. As a metric of the scalability, we observed the maximum number of configurable routers with regards to the network radix which corresponds to the number of ports connected to only the other routers. In Fig. 3, the number of layers in FleX is fixed to 3, and the 2-dimensional HyperX is denoted as HyperX. FleX can achieve 20.28% of diameter-2 Moore bound when the network radix is 100. Dragonfly achieves 15.38% of diameter-3 Moore bound, and Slim Fly can achieve 90.00% of diameter-2 Moore bound. However, due to the structural characteristics of only layer-to-layer connections of FleX without intra-layer connectivity, the actual network configuration cost for FleX is relatively small and performance shows low latency by low-diameter structure. The detail cost and performance for FleX is evaluated in Section IV.

B. ANALYSIS ON STRUCTURAL PROPERTIES

FleX has different network diameter with regards to the number of layers, and diameter increases as the number of layers increases larger than three. We mainly analyze 3-layer FleX structure that has the minimum diameter and examine the structure. For the structural analysis, comparisons with other topologies were conducted mainly over dragonfly, 2-dimensional HyperX, 3-level fat tree, and Slim Fly. Each topology was set as the configuration that allowed each topology to have a full global bandwidth. Dragonfly was configured to satisfy $a = 2p = 2h$, 2-dimensional HyperX had same size on each dimension and was configured to satisfy $p = \lfloor k/3 \rfloor$. 3-level fat tree was set to have $p = \lfloor k/2 \rfloor$, Slim Fly was configured to meet $p = \lceil k/3 \rceil$ as specified in [4]. For FleX, we analyzed on the settings that satisfy $N_x = N_y$ and the $p = \lfloor k/3 \rfloor$ that is equal to the number of links from each router to the routers on the next layer for full global bandwidth.

TABLE 2. Network diameter of various interconnection networks.

Topology	Use Case	Diameter
Fat tree (3-level)	Tianhe-2A [16]	4
Dragonfly	Cray XC50 [17]	3
HyperX (2-dimensional)	HPE Lab [27]	2
Slim Fly	-	2
FleX (3-layer)	-	2

1) Network Diameter

For 3-layer FleX, as any destination can be reached by passing whole 2 neighboring layers, the network diameter of 3-layer FleX becomes 2. Compared to the conventional topologies as shown in Table 2, 3-layer FleX has the lowest diameter along with the Slim Fly and 2D HyperX. This low-diameter characteristics contribute to low latency performance for the network. We also analyze the whole possible minimal paths in FleX where the distance of the maximum shortest path is 2 (fully described in Section III-C1). Such characteristic can also be empirically evaluated as conducted in conventional studies [1], [2], [4], using Booksim [26] cycle accurate simulator to verify in a hop count level. Accordingly, we further verified that FleX ensures diameter of 2 from empirical evaluation which is described on Section IV-B in detail.

2) Distance

For each router in 3-layer FleX, $(k - p) = (2N_x + 2N_y - 1)$ routers are connected with distance one, and rest $(3N_xNy - (k - p) - 1)$ routers are connected with distance two. Therefore, for all ${}_{3N_xN_y}C_2$ minimal path, $3N_xN_y(2N_x + 2N_y - 4)/2$ paths has distance one, and $3N_xN_y(3N_xN_y - (k-p) - 1)/2$ paths has distance two in total. Fig. 1 shows the average hop count over various topologies using minimal routing, which is obtained from the cycle accurate simulation. The detail configurations for topologies are described in Table 6. In the figure, the average hop count for end terminal-to-terminal communication shows less than 3 in the 3-layer FleX topology, and it means that the distance between the routers does not exceed 2. In addition, due to the structural characteristics of FleX, various detouring minimal paths exist which is described on Section III-C1 in detail.

3) Bisection Width

As a metric for true bandwidth of entire network system, we analyze the bisection width which is the minimum number of edges to be removed to partition the topology into two equal partitions. In case that N_x and N_y are even, bisection cut of 3-layer FleX is sliced at the smallest axis among x and y axes. For each router, the number of edges that are connected to the next layer and meet the bisection cut is $min(N_x, N_y)/2$. As the number of routers in each layer is N_xN_y, and the number of connections with next layer is 3, the bisection width results in (8) where N_R denotes the number of total routers in the topology.

TABLE 3. Partitioning results for verifying the bisection width of FleX.

N_x	N_y	N_R	Bisection Width (eq. 8)	Partitioner Results
4	4	48	96	96
5	4	60	120	120
15	20	900	6,750	9,000
20	20	1,200	12,000	12,000
4	100	1,200	2,400	2,400

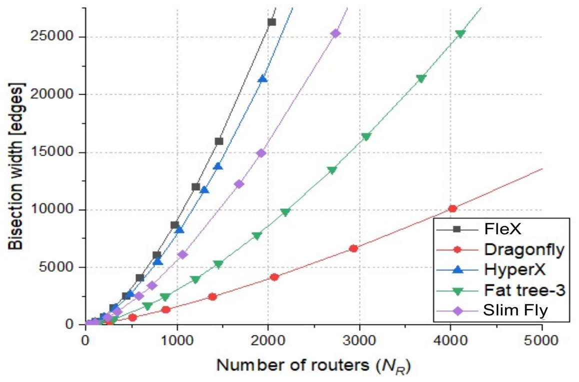


FIGURE 4. Bisection width comparison over the topologies with regards to the number of routers.

$$\begin{aligned} BisectionWidth &= 3N_xN_ymin(N_x, N_y)/2 \\ &= N_Rmin(N_x, N_y)/2 \end{aligned} \tag{8}$$

In Slim Fly [4], the bisection width is approximately verified using the METIS [28] partitioner. The METIS partitioner conducts graph partitioning based on a multi-level k-way partitioning algorithm [29]. Although the graph partitioning problem is NP-complete and the METIS partitioner does not necessarily derive the optimal partition, it is used extensively due to its simplicity and its ability to produce high-quality partitions practically [28], [29]. Likewise, we verified the obtained bisection width for 3-layer FleX using the METIS [28] partitioner. Table 3 shows the bisection width results of the partitioner on various 3-layer FleX configurations. We could verify that the results of the bisection width from the partitioner is identical to the value obtained from (8), though it is slightly different when the smallest length of the axis is an odd number and cannot be divided exactly.

In addition, in order to look at the true bandwidth on inter-router connections, we first looked at the bisection width according to the number of routers. Fig. 4 shows the bisection width comparison over other topologies with regards to the number of routers. Bisection widths for 3-level fat tree, dragonfly and HyperX is derived analytically [1], [2], [5], and bisection width for Slim Fly is obtained using the METIS partitioner. FleX shows the higher bisection width than other topologies, and it means that FleX can accommodate higher amount of communication than the other topologies when all cutting edges are used efficiently. As the bisection width of FleX exceeds $N_R/2$ load, FleX has detouring links, and it can also have advantage on fault tolerance.

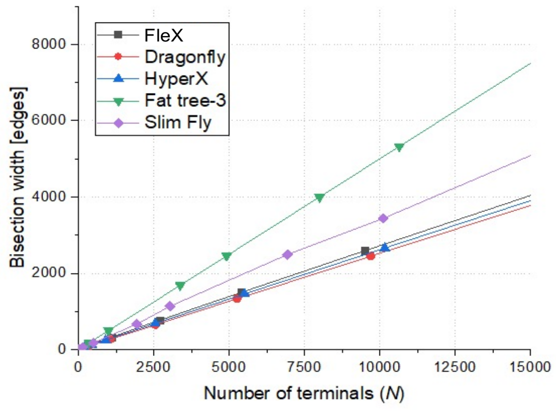


FIGURE 5. Bisection width comparison over the topologies with regards to the number of terminals.

TABLE 4. Bisection ratio comparison over the topologies with regards to the scale of the system.

$\approx N$	Fat tree	Dragonfly	HyperX	Slim Fly	FleX
1,000	0.250	0.187	0.278	0.326	0.300
2,000	0.250	0.182	0.269	0.354	0.286
5,000	0.250	0.179	0.265	0.345	0.278
10,000	0.250	0.177	0.262	0.328	0.273

To look at the true bandwidth in the actual environment of constructing the topology, we also looked at the bisection width according to the number of terminals over topologies. The concentration (p) for each topology was configured to have a full global bandwidth as described earlier in Section III-B, and the bisection width results with regards to the number of terminals were shown in Fig. 5. Given the number of terminals, the bisection width of FleX offers half of the full bisection width ($N/2$), which is similar to HyperX and dragonfly.

As a further analysis, we also observe the bisection ratio (*i.e.,* bisection width divided by the total inter-router width). The bisection ratio indicates how much the endurable load (*i.e.,* bisection width) is distributed to the whole network, which can represent the balance between local (non-bisection links) and global (bisection links) connections of the network, not the performance of the network. As shown in Table 4, FleX shows similar bisection ratio to HyperX, which shows the balanced position between Slim Fly (high bisection ratio) and dragonfly (low bisection ratio). However, different from HyperX, FleX does not contain any intra-layer links, and only contains inter-layer links, while maintaining the low diameter. Accordingly, unlike HyperX, which can often require to accommodate both inter-dimensional and intra-dimensional loads in a certain link, it is more advantageous for FleX to evenly distribute loads to the various links by providing much more diverse routing candidates. The more detailed descriptions about routing in FleX is addressed in Section III-C.

TABLE 5. Fault tolerance of topologies over various network size

$\approx N$	Fat tree	Dragonfly	HyperX	Slim Fly	FleX
500	45%	50%	70%	60%	70%
1,000	45%	55%	75%	-	75%
2,000	55%	60%	80%	70%	80%
5,000	60%	65%	80%	70%	80%
10,000	65%	65%	80%	75%	85%

4) Fault Tolerance

To analyze the fault tolerance performance of the proposed topology network, we measured how many links can be removed randomly without disconnecting the entire network. We gave random failure of links with 5% increments and observed whether the network is disconnected or not. Table 5 shows the maximum number of links that can be removed without disconnecting the networks on various network sizes and topologies. We conducted 100 iterations for each cases and observed the maximum link failure that can endure disconnection with more than half of trials. FleX is shown to endure the higher link failure than other topologies (near 5% enhancement to HyperX with similar cost), and can endure higher link failure for the more number of terminals (N) as it has various detouring links by structural characteristics.

C. STOCHASTIC LOAD BALANCING SCHEMES WITH ROUTING ON FLEX

In this section, we firstly introduce the minimal routing scheme determining the forwarding path of injected packets for 3-layer FleX. As FleX can provide various detouring paths, we also propose a stochastic load balancing scheme that can evenly distribute input traffic load to the such various paths according to the congestion status of each path. We consider routing from a source router $v_s = (x_s, y_s, z_s)$ to a destination router $v_d = (x_d, y_d, z_d)$, and assume that both N_x and N_y is greater than or equal to 3.

1) Minimal Routing

The minimal routing scheme for the 3-layer FleX is based on the dimension order routing algorithm which is used in the multi-dimensional networks such as mesh or hypercube. However, as each router is not directly connected to the routers that have same x and y axis value, but have different z value, dimension order routing can not be directly applied to the FleX network. Therefore, we introduce a minimal routing scheme for FleX network, and it can be briefly described in three main steps:

- Step 1: If $x_s = x_d$ and $y_s = y_d$, route to the router that has different value on one of x or y axis and is on the other neighboring layer.
- Step 2: If $x_s \neq x_d$ and $y_s \neq y_d$, route to the router that has same y-axis value with the destination router and is on the other neighboring layer.
- Step 3: Either $x_s \neq x_d$ or $y_s \neq y_d$, if only one of them is satisfied, route directly to the destination when $z_s \neq$

z_d, otherwise route to the router that has the other value on different x or y axis and is on the other neighboring layer.

For the further detail, Algorithm 1 describes how to route to the minimal path for all possible cases that can occur in the 3-layer FleX network. In Algorithm 1, $x_{intm}, y_{intm}, z_{intm}$ denotes each intermediate value on x,y,z axes for selecting available intermediate routers to route.

When all the x, y, z-axes values of source and destination router are different, routing is conducted to first align the y-axis, then the x-axis and finally the z-axis. Different from the dimension order routing, due to the structural characteristics of the inter-layer connection in FleX, a constraint is added that it should not be routed to the layer of the destination router before the z-axis is finally aligned.

If the destination and source routers are in the same layer, when the both x and y axis values are different, the y-axis in the neighboring layer is aligned firstly, and then the x-axis and the z-axis is aligned simultaneously to reach the destination. In this case, looking at other possible routing paths, there are 4 different minimal paths considering that there are 2 cases for selecting the neighboring layer (previous or next layer) and 2 cases for selecting the axis to align firstly (x-axis or y-axis).

In case that the destination and source routers are on the same layer, and only one of x or y axis value is the same, the packet is routed to the any connected router that is on the axis of different value in the neighboring layer firstly, and then reaches to the destination directly. For this case, likewise, looking at other possible minimal routing paths, there are 2 cases for selecting the neighboring layer, (N_x-2) or (N_y-2) cases for selecting the connected routers that is on the axis having different value, and $2(N_x - 2)$ or $2(N_y - 2)$ different minimal paths in total.

When the destination and source routers are on the different layers, and both x and y axis values are the same, the packet is routed to the any connected router on the x-axis in the neighboring layer that is different from layer of source and destination firstly, and then reaches to the destination directly. Looking for the other possible cases of minimal path, there are $(N_x - 2)$ or $(N_y - 2)$ different minimal paths, as there are $(N_x - 2)$ or $(N_y - 2)$ cases for selecting the arbitrary connected router on x or y axis and only one layer to choose.

If the destination and source routers are on the different layers, and only one of x or y axis have different values, the packet can reach to the destination directly, since the source and destination routers are connected directly. Even though a failure occurs in that connection, the packet can be routed to the any connected router on the axis having different value in the neighboring layer that is different from layer of source and destination firstly, and then can reach to the destination with total traveling distance 2. There are $(N_x - 2)$ or $(N_y - 2)$ different detouring paths that have distance 2, and these detouring path does not have influence on overall diameter of the network.

Algorithm 1: Minimal routing algorithm for 3-layer FleX interconnection network.

```
source router: $v_s = (x_s, y_s, z_s)$
destination router: $v_d = (x_d, y_d, z_d)$
current routed node: $v_c = (x_c, y_c, z_c)$
$v_c \leftarrow v_s$
while ($v_c = v_d$) do
    if ($z_c = z_d$) then
        if ($x_c = x_d$) then
            select one $y_{intm}$ on $\forall y_{intm} \in Y$
                s.t. $y_{intm} \neq y_c$ and $y_{intm} \neq y_d$;
            $v_c \leftarrow (x_c, y_{intm}, (z_c + N_z - 1) \bmod N_z)$
        else if ($y_c = y_d$) then
            select one $x_{intm}$ on $\forall x_{intm} \in X$
                s.t. $x_{intm} \neq x_c$ and $x_{intm} \neq x_d$;
            $v_c \leftarrow (x_{intm}, y_c, (z_c + N_z - 1) \bmod N_z)$
        else
            $v_c \leftarrow (x_c, y_d, (z_c + N_z - 1) \bmod N_z)$
    else
        if (($x_c = x_d$) and ($y_c = y_d$)) then
            select one $x_{intm}$ on $\forall x_{intm} \in X$
                s.t. $x_{intm} \neq x_c$ and $x_{intm} \neq x_d$;
            select one $z_{intm}$ on $\forall z_{intm} \in Z$
                s.t. $z_{intm} \neq z_c$ and $z_{intm} \neq z_d$;
            $v_c \leftarrow (x_{intm}, y_d, z_{intm})$
        else if (($x_c = x_d$) or ($y_c = y_d$)) then
            $v_c \leftarrow (x_d, y_d, z_d)$
        else
            select one $z_{intm}$ on $\forall z_{intm} \in Z$
                s.t. $z_{intm} \neq z_c$ and $z_{intm} \neq z_d$;
            $v_c \leftarrow (x_c, y_d, z_{intm})$
```

Therefore, as we looked all possible cases on minimal routing, FleX can be robust to the link failure, since there are several detouring paths with keeping diameter 2. In addition, it can be expected to use the network efficiently, if we use the various detouring minimal paths according to the load.

2) Stochastic Load Balanced Routing

In order to highly utilize various detouring paths (minimal or non-minimal) on FleX practically, we propose a stochastic Load Balanced Routing (LBR) algorithm. LBR stochastically selects a path to route from the source to the destination by modeling the probability to distribute the flits based on the quantitative loads of several candidate paths. The final goal of LBR is identical to those of other adaptive routing schemes, such as UGAL [8], which aims to route each flit to minimize congestion throughout the whole network where the finite numbers of both minimal and non-minimal candidate paths are predefined. Specifically, the goal of LBR targets is evenly to distribute the input loads to all predefined candidate paths in a stochastic manner by quantifying the load of each path based on the queue length information of each router.

Let $G = (V, E)$ be a graph corresponding to the given interconnection network, with the set of nodes V and the set of edges E. Here, let $P(G)$ be the set of all the paths established by G. The length of a path $p \in P$ is defined by the number of edges in p, denoted by $|p|$. If a source node v_s and a destination node v_d are a pair of vertices in a graph $G = (V, E)$, we represent a *minimal path* p^* from v_s to v_d satisfying the length of p^* is not larger than that of any other paths in $P(G)$, that is, $|p^*| \leq |p|$, $\forall p \in P(G)$. Since there might be more than one minimal paths from source node v_s to the destination node v_d, we can consider a function to generate the set of minimal paths from v_s to v_d as follows. A function $f : V^2 \to 2^{P(G)}$ defined on the graph $G = (V, E)$ is the minimal path function, if for a pair of vertices (v_s, v_d) in G, $f(v_s, v_d)$ is a set of all the minimal paths from v_s to v_d. Now, we also consider a *non-minimal path* p' from v_s to v_d which the length is larger than $|p^*|$, as the routing candidates when packet is forwarded. A function $g : V^2 \to 2^{P(G)}$ defined by the graph $G = (V, E)$ is given to the non-minimal path function, if for a pair of vertices (v_s, v_d) in G, $g(v_s, v_d)$ is a set of all the non-minimal paths from v_s to v_d. In this situation, let $q : V \to \mathbb{R}$ defined on a graph $G = (V, E)$ be the real-valued function from a vertex in G to the queue length of the packets to be processed by the node of the corresponding interconnection network, then q is called by *the queue-length function*. Also, for convenience, override the function q from a path to the queue length of all channels in the path, described as $q(p) = \sum_{v \in p} q(v)$. The queue occupancy of local and global channels in the interconnection indicate how much data traffic has been congested.

In LBR environment, we assume that there are N minimal paths and M non-minimal paths from source to destination. In practical, non-minimal paths can be determined through Valiant random routing (VAL) [30]. The minimal paths have the same hop count $n = |p^*|$ from source to destination, while non-minimal paths can formulate $m_i = |p'_i|$. The procedure of updating forwarding rule in interval T can be described as follows. Assume that the source node generates λ_s packets per a second, and the queue-length of each node is updated by the time interval T seconds. Then, LBR stochastically select a path to route from v_s to v_d as shown in the following steps.

Step 1. Take minimal paths $P_n = \{p_1^*, p_2^* \cdots, p_N^*\}$ with $|P_n| = N$ and non-minimal paths $P_{nm} = \{p'_{N+1}, p'_{N+2}, \cdots, p'_{N+M}\}$ with $|P_{nm}| = M$ for all source to destination pairs (v_s, v_d).

Step 2. In P_n and P_{nm} remove busy path p_k meeting the following conditions which the congestion occurs during T seconds

$$q(p_k) \geq \frac{q_s T + \sum_{j=1}^{N+M} q(p_j)}{N+M}. \tag{9}$$

Let $P'_n = \{p_1^*, \cdots, p_{N'}^*\}$ and $P'_{nm} = \{p'_{N'+1}, \cdots, p'_{N'+M'}\}$ be a set of the minimal and near-minimal paths without satisfying above condition.

Step 3. Stochastically select the path $p_k \in P'_n \cup P'_{nm}$ to route with the probability α_k, where

$$\alpha_k = \frac{1}{N'+M'} + \frac{\sum_{j=1}^{N'+M'} q(p_j)}{\lambda_s T(N'+M')} - \frac{q(p_k)}{\lambda_s T}. \tag{10}$$

After updating interval T, each source v_s determines available routing paths including N minimal and M non-minimal paths for all destination v_d, and it also profiles the packet injection rate λ_s. Then it excludes the paths that are expected to congestion during the next interval using (9). This implies that the current queue length $q(p_k)$ is higher than the average queue length of $q(p_i), \forall p_i \in P_n \cup P_{nm}$ during the next interval. Finally according to every (v_s, v_d) the forwarding probability of p_k, α_k can be obtained by maintaining the load balancing of candidate paths if λ_s is constant during period T.

Herein, the sum of forwarding probability (α_k) among all routable candidate paths $(\forall p_k)$ is guaranteed to be 1, which means that all packets can be distributed to the predefined candidate paths as described in *Lemma 1*.

Lemma 1 (Load Distribution)**.** *For given all routable candidate paths (P_n and P_{nm}), all packets can be stochastically distributed to each routable path with probability α_k, which satisfies following:*

$$\sum_{k=1}^{N+M} \alpha_k = \sum_{k=1}^{N'+M'} \frac{1}{N'+M'} + \frac{\sum_{j=1}^{N'+M'} q(p_j)}{\lambda_s T(N'+M')} - \frac{q(p_k)}{\lambda_s T} \tag{11}$$

$$= 1 + \frac{\sum_{j=1}^{N'+M'} q(p_j)}{\lambda_s T} - \frac{\sum_{k=1}^{N'+M'} q(p_k)}{\lambda_s T} \tag{12}$$

$$= 1. \tag{13}$$

Moreover, if the number of packets $(\lambda_s T)$ accumulated during updating interval T is sufficient that case for (9) does not occur at the certain router, the packets are distributed to make the expected queue length of each candidate path be same as follows:

$$q(p_k) + (\lambda_s T)\alpha_k \tag{14}$$

$$= q(p_k) + \lambda_s T \left[\frac{1}{N+M} + \frac{\sum_{j=1}^{N+M} q(p_j)}{\lambda_s T(N+M)} - \frac{q(p_k)}{\lambda_s T} \right] \tag{15}$$

$$= \frac{\lambda_s T + \sum_{j=1}^{N+M} q(p_j)}{N+M}, \tag{16}$$

where (16) shows that expected queue length after stochastically load balancing is same over all candidate paths $\forall p_k$ (*i.e.*, traffic load for each path becomes evenly).

The proposed routing scheme can dynamically forward packets to candidate paths preserving the load balancing in the interconnection. In addition, it can also be a feasible routing solution with global queue information for a large-scale system by controlling the time interval T.

Likewise, in order to verify the performance of the LBR of global adaptive routing version in FleX network, we also

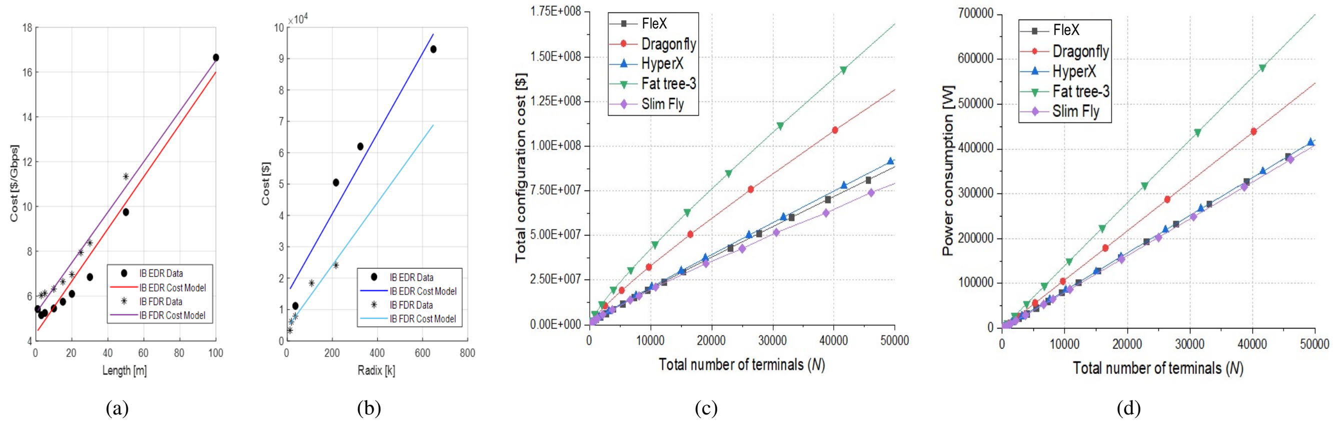


FIGURE 6. (a) Obtained cost model for cable in terms of length. (b) Obtained cost model for router with respect to radix. (c) Comparison of the total cost for constructing the network. (d) Comparison of the power consumption of routers in the network.

implemented the simulation, where 1 minimal was obtained by the previously described minimal routing method and the other 3 non-minimal path was obtained through VAL [30]. Detailed evaluations and results are discussed in Section IV-B.

IV. EVALUATION

In this section, FleX is evaluated in terms of its cost, power and performance aspects over other topologies, and the results in each case are discussed. In addition, the cost and performance of FleX settings on the various numbers of layers are briefly evaluated and discussed.

A. COST AND POWER CONSUMPTION

In supercomputing or datacenter environments, the cost of hardware for constructing the system and the energy consumption cost account for more than 70% of the Total Cost of Ownership (TCO) [31], [32]. Therefore, we mainly evaluated the configuration cost and power consumption for FleX. We compared results for FleX, dragonfly, 2-dimensional HyperX, 3-level fat tree, and Slim Fly. Each topology is configured to have a full global bandwidth, as fully described in Section III-B.

1) Configuration Cost Model

To compare the costs of constructing the network over various topologies, we looked at the costs of the routers and interconnection cables, which constitutes a large portion of the overall network configuration. First of all, as the cost of cable per unit bandwidth found to be proportional to the length of the cable, as indicated in general studies [3], [4], it can be modelled through linear regression. We assumed the InfiniBand EDR environment, which is commonly used in Top500 HPC systems [21], and selected the Mellanox InfiniBand EDR 100Gbps active optical cable model. The cost model for the cable in this case is obtained as $f(x) = 0.1167x + 4.3410$ [$/Gbps] which can be shown in Fig. 6(a). We used commercially available selling price data [33] to establish the cable cost model. The cost model for the Mellanox

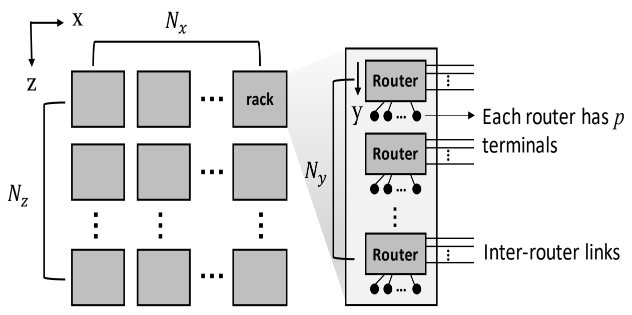


FIGURE 7. Physical layout for constructing FleX interconnection network.

IB FDR 56Gbps active optical cable was also obtained and compared, as shown in Fig. 6(a).

In addition, we obtained the cost model for routers in the same way. For the routers, the cost is proportional to the radix. We selected the Mellanox EDR InfiniBand switch, and the cost model for the router is derived as $f(k) = 128k + 15029$ [$] through linear regression. Likewise, the cost model for the Mellanox IB FDR router was also obtained and compared, as shown in Fig. 6(b).

To calculate the approximate cost of configuring the network, we assumed that the routers and the terminals were grouped in the same rack. All of the connections between the routers and terminals were set to $1m$, and the distance between the two different racks was based on the Manhattan distance, adding $2m$ overhead, as shown in [3].

To calculate the configuration cost of FleX, the physical layout of FleX is applied in a naive fashion. As shown in Fig. 7, each y-axis in each layer of FleX is firstly packaged into one rack. The detailed layout was packaged first for each y-axis of each layer of FleX. Next, the racks corresponding to each layer are arranged in a row according to the x-axis order, and the links are connected as presented in the FleX topology description. The Manhattan distances on the x, y, and z axes in all cases in the topology are applied to calculate the lengths of the inter-router links.

TABLE 6. Configurations of topologies for latency performance evaluation with their configuration cost and power consumption.

	Fat tree	Dragonfly	HyperX	Slim Fly	FleX
Total terminals (N)	1,000	1,056	1,000	972	1,080
Total routers (N_R)	300	264	100	162	108
Radix (k)	20	15	28	20	30
Total links	2,000	1,452	900	1,053	1,080
Total configuration cost [$]	6.88×10^6	5.16×10^6	2.75×10^6	3.48×10^6	3.05×10^6
Total power consumption [W]	14,336	11,088	7,840	8,618	9,072

For the other topologies, dragonfly, 3-level fat tree, and Slim Fly were arranged with their respective packaging rules, as described in [4], and 2-dimensional HyperX was also arranged using its own packaging rule [2].

In these settings, the total network configuration cost for each topology was calculated using the obtained cost model of the cable and router as shown in Fig. 6(c).

As shown in Fig. 6(c), the cost of configuring the network for FleX is lower than those of HyperX and dragonfly but slightly higher than that of Slim Fly at the various total terminal sizes (N). In the InfiniBand EDR environment as assumed here, the switch costs are much higher than the cable costs. Accordingly, the fewer routers we use, the lower the cost to construct the network.

To examine these results more closely, as shown in Fig. 6(a) with regard to the cable cost, the cost per unit length on the unit bandwidth decreased as the technology matured from FDR to EDR, but the bandwidth grew to 56Gbps for FDR and 100Gbps for EDR. On the other hand, for the router, the cost per unit radix increased as the FDR evolved into EDR, as shown in Fig. 6(b). However, the rate of increment for the router cost with regard to the radix has decreased over the years. For example, earlier researches [4] showed that the gradient value of the obtained router cost model for the Mellanox IB FDR10 router is 350.4, much higher than the gradient value (=128) of the recent cost model for the Mellanox IB EDR router. Due to these development tendencies of interconnection hardware technology, it can be inferred that using fewer cables and using fewer routers with higher radix values would be more efficient with regard to configuration costs.

Therefore, as shown in the topology-specific configurations for constructing nearly 1,000 terminals with having a nearly full global bandwidth (Table 6), FleX had nearly 1,000 links and used relatively few routers with a high radix to result in a relatively low configuration cost. In Table 6, as the concentration (p) of HyperX is set to 10 instead of $p = \lfloor k/3 \rfloor = 9$ in order to match the closest possible number of terminals for a latency performance comparison, HyperX showed a slightly lower configuration cost than FleX. Details of latency performance results are covered in Section IV-B.

In addition, as shown in Fig. 15 (will be discussed in Section IV-B), the configuration cost increases significantly as the dimension of HyperX increases from 2 to 3, as 3D HyperX has a higher configuration cost than dragonfly. However, for FleX, increasing the number of layers results in a smaller increase in the configuration cost and can provide a range of various configurations depending on the number of layers without changing the radix.

2) Power Consumption Model

The power consumed by the interconnect system accounts for a significant portion of that used by the overall computing system [32], [34]. We assumed that each router port has 4 lanes consuming approximately 0.7 watts for each lane individually, as shown in [32]. Fig. 6(d) shows the total power consumption results of the routers over the topologies. The power consumption by FleX is lower than those of 3-level fat tree, Dragon fly but slightly higher than that by Slim Fly. These results show that FleX is relatively more energy efficient than the other topologies.

B. LATENCY PERFORMANCE

We evaluated the minimal routing and adaptive routing performance outcomes with the Universal Globally Adaptive Load-balanced (UGAL) [8] and LBR routing schemes on the 3-layer FleX network. In UGAL routing, we consider local version routing operated by selecting the path with the smallest value upon multiplication of the local queue length and the hop count for each packet, using only partial router information among all routers in a path. We used the Booksim [26] cycle accurate simulator, in which packets are injected by the Bernoulli process in input-queued router environments. This cycle accurate simulator is widely used to evaluate latency performances of target interconnection networks with the specific routing schemes, as has been conducted in several studies [1], [2], [4], [22]–[25]. We added the network configuration and routing function modules for FleX in the simulator. We used single flit (flow control unit) packets and set the router delay for credit processing such that it used 2 cycles. The buffer size for each virtual channel (VC) was set to 256, and the speedup of the internals of the routers over the channel transmission rate was set to 2.

First, we evaluated the performance of FleX against other topologies, with each topology having the configuration settings shown in Table 6. For a fair comparison, we set each topology with similar terminal size (N) and evaluated the latency performance of various routing methods in each network on 10 different traffic patterns (random uniform, bit complement, bit reversal, shuffle, transpose, hot spot, random permutation, asymmetric, neighbor, and tornado). Each topology was configured to have a full global bandwidth and

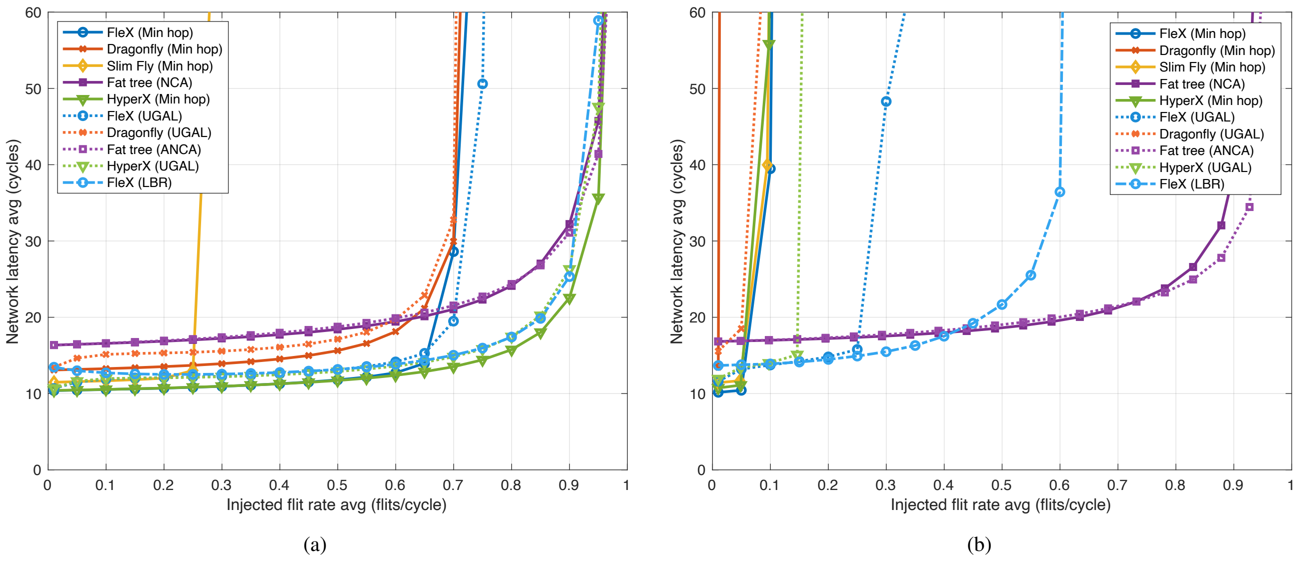


FIGURE 8. Latency performance evaluation on (a) random uniform traffic, and (b) bit complement traffic pattern.

near 1,000 terminals in total. The 3-level fat tree used a structure of $k = 20$, $p = 10$, and dragonfly applied $a = 2p = 2h$, $p = 4$ configuration. For HyperX, a 2-dimensional structure with a network diameter of 2 is used; the size of each dimension is set to 10, and the concentration is set as $p = 10$ to match both a nearly full global bandwidth ($p = \lfloor k/3 \rfloor$) and nearly 1,000 terminals. In Slim Fly, q was set to 9 considering the full global bandwidth structure with a network diameter of 2 and nearly 1,000 terminals, and p was set to 6 and decreased by 1 from the concentration (p=$\lceil k/3 \rceil$=7) with the full global bandwidth specified in [4] to have nearly 1,000 terminals among structures where the network radix (k-p) is 13. 3-layer FlexbleX with a network diameter of 2 was configured with $N_x = N_y = 6$, $p = \lfloor k/3 \rfloor = 10$ for a full global bandwidth. With regard to the configuration cost considering the topologies, 3-level fat tree and dragonfly show higher configuration costs, and 2-dimensional HyperX and FleX show the lowest total cost, though only differing by 3% taking into account the difference in the total size of terminals. HyperX shows a lower configuration cost compared to FleX because the concentration of HyperX is increased by 1 from the full global bandwidth condition.

As the bit complement, bit reversal, shuffle, transpose traffic patterns require the total number of terminals to be exactly the power of 2 (1024 in this case), we disabled the extra terminals, and the patterns for the missing terminals were not given.

Fig. 8 shows the latency results with regards to the offered load on each network, the routing method, and the traffic pattern. In Fig. 8(a), packets are injected into each terminals with a random uniform probability. FleX yields lower latency and higher saturation throughput than Slim Fly and dragonfly, with results similar to those of HyperX. It is clear that FleX yields lower saturation throughput than fat tree and HyperX with minimal routing and UGAL scheme because there are certain paths for which packets go back to the previous layer according to the destination with minimal routing. However, FleX shows saturation throughput comparable to the results of HyperX and 3-level fat tree using LBR to address this overhead adaptively.

For the bit complement traffic patterns, FleX shows lower latency and higher saturation throughput over dragonfly, Slim Fly, and HyperX on each routing method, but it has lower saturation throughput than 3-level fat tree, as shown in Fig. 8(b). Although FleX incurs a lower cost for configuration and consumes less power consumption than fat tree, Flex can further endure an injection load of nearly 0.6 (flits/cycle) with the help of evenly distributing to detouring paths by LBR. However, as we set all detouring paths to non-minimal paths for convenience of the implementation, the latencies of UGAL and LBR increase slightly compared to that of the minimal routing scheme. For the FleX network, as it contains several minimal detouring paths, we can expect lower latency also with UGAL and LBR schemes.

In the bit reversal traffic patterns, as shown in Fig. 9(a) at next page, FleX achieves the lowest latency, similar to HyperX, and higher saturation throughput than Slim fly. Analogous to random uniform packets, FleX yields lower saturation throughput due to the doubled congestion of the paths that go back to the previous layer and go forward to the next layer with minimal routing. However, with adaptive routing considering this overhead, as in the UGAL and LBR cases, FleX achieves higher saturation throughput than dragonfly, HyperX, nearly matching that of the expensive fat tree network.

With regard to the shuffle traffic patterns (Fig. 9(b)), although FleX yields lower saturation throughput than fat tree with minimal routing, for the UGAL and LBR cases, by distributing traffics to various detouring paths adaptively, it achieves higher saturation throughput than the other intercon-

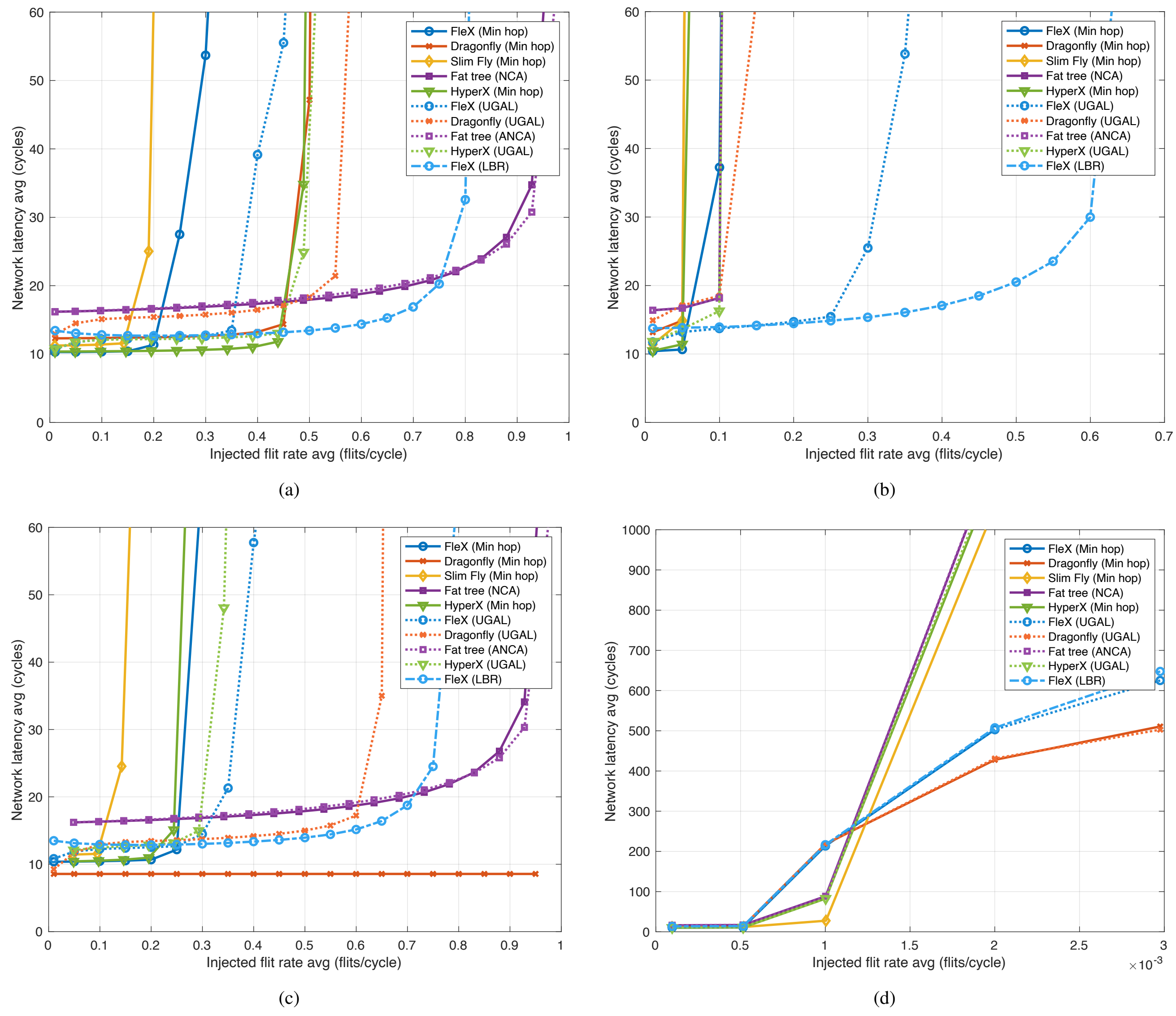


FIGURE 9. Latency performance evaluation on (a) bit reversal traffic, (b) shuffle traffic, (c) transpose traffic, and (d) hotspot traffic pattern.

nection networks, and the lowest latency, similar to HyperX.

In Fig. 9(c), dragonfly shows the lowest latency with minimal routing and UGAL routing. Unlike the other interconnection networks, dragonfly has $p = 4$ with 8 routers per group and has 1,024 terminals in total. Therefore, the destination terminal of each source terminal is regularly aligned (as the index of each source increases by 1, the destination index increases by 32, which corresponds to the number of terminals in one group), resulting in fewer inter-router traffic overlaps compared to the other networks and lower latency. In the case of UGAL on dragonfly, as each router routes packets with local judging, unnecessary routing on a non-minimal path can occur, resulting in higher latency and lower saturation throughput than those with minimal routing on dragonfly. FleX shows higher saturation throughput than Slim Fly and HyperX and shows the second lowest latency outcome, similar to HyperX with the minimal routing scheme. Likewise, with the synergy gained by containing various detouring paths in FleX and evenly distributing the traffic load by LBR, FleX can extend the saturation throughput in the LBR scheme considerably.

For the hot spot traffic pattern, as shown in Fig. 9(d), FleX has the lowest latency with HyperX and Slim Fly until encountering saturation throughput.

In the random permutation traffic patterns, FleX shows the lowest latency performance, similar to HyperX, as shown in Fig. 10(a) at next page. Except for fat tree, which is the most expensive network, the other networks can only accommodate an injected flit rate of lower than 0.4 flits/cycle with minimal routing and UGAL. However, with the help of the various balanced detouring paths in FleX and the evenly load distributing scheme, FleX highly extends the saturation throughput, close to the result of fat tree when applying the LBR scheme.

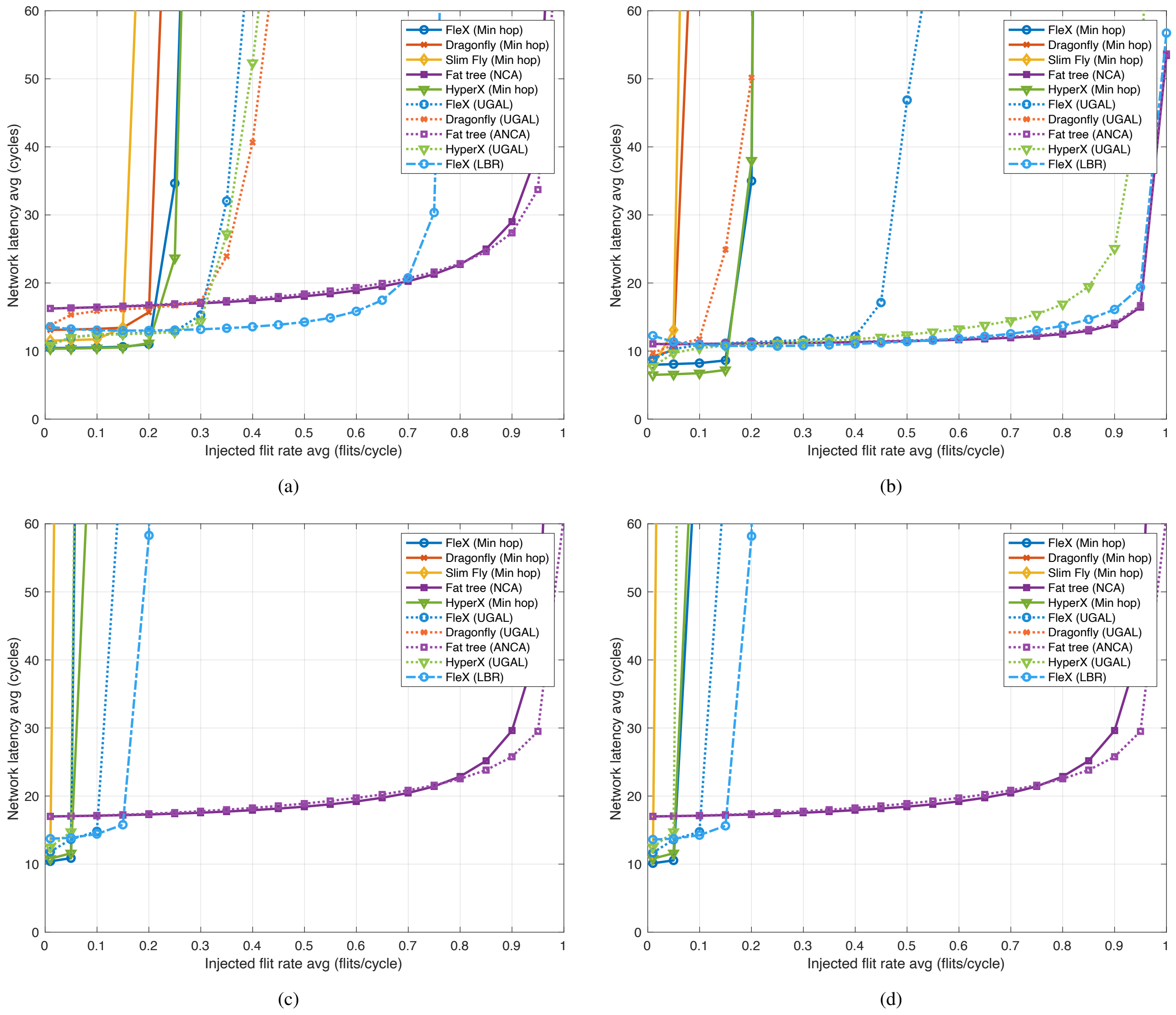


FIGURE 10. Latency performance evaluation on (a) random permutation traffic, (b) asymmetric traffic, (c) neighbor traffic, and (d) tornado traffic pattern.

The results with asymmetric traffic patterns also show similar trends, as shown in Fig. 10(b). In this traffic pattern, HyperX shows higher saturation throughput than Flex on UGAL. Nevertheless, by fully utilizing the balanced detouring paths in Flex through the use of the LBR scheme, FleX can extend the saturation throughput further, similarly to the fat tree network, while only incurring close to half of the configuration cost of fat tree.

For the neighbor (Fig. 10(c)) and tornado (Fig. 10(d)) traffic patterns, the results show similar aspects to each other. FleX shows the lowest latency performance, similar to HyperX, on each routing scheme, and the saturation throughput of FleX on the LBR scheme is largely expanded compared to the results of dragonfly, Slim Fly and HyperX. However, as the neighbor and tornado traffic patterns mainly generate local communications in the dimension of the network radix, fat tree, which has a huge number of links over local communications in each dimension, inevitably shows the highest saturation throughput, while the other networks wither fewer links cannot overcome the structural limitations in such traffic patterns showing the low saturation throughput.

The results overall show that FleX has lower latency than the 3-level fat tree, which shows relatively similar performance regardless of the traffic pattern. FleX can also achieve higher saturation throughput than HyperX, dragonfly, and Slim Fly by applying LBR scheme to distribute the input loads evenly to the various balanced detouring paths in Flex. As LBR allows non-minimal paths to be candidate routing paths, the evaluation results show that LBR sacrifices latency performance over minimal routing (about 16% increase), whereas LBR largely expands the saturation throughput (nearly $\times 2$ improvement over minimal routing).

In addition, Fig. 11 shows a hop count histogram of packets in 3-layer FleX topology with minimal routing. In the

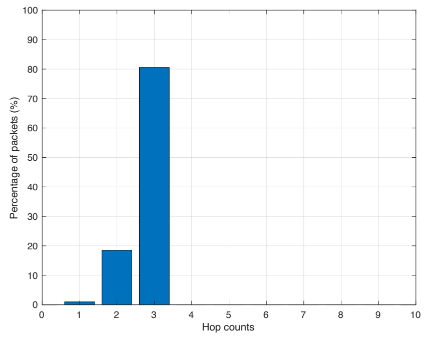


FIGURE 11. Histogram for hop count of packets at 3-layer FleX topology with minimal routing.

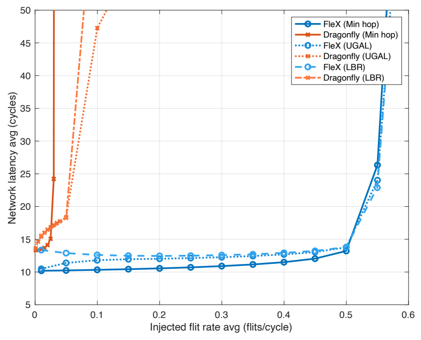


FIGURE 12. Latency performance on the next group (layer) traffic pattern in FleX and dragonfly.

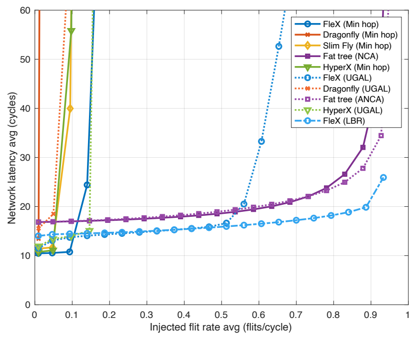


FIGURE 13. Latency performance comparison for the higher cost configuration of FleX.

figure, we can verify that the network meets the diameter-2 at maximum 3 hop count, with this low-diameter characteristic resulting in low latency performance. Moreover, we can verify that the proposed minimal routing algorithm for FleX routes the packets along the minimal path correctly.

We also compared FleX on the worst case traffic pattern of dragonfly, as shown in Fig. 12. Although it is unfair to compare FleX, for which all links are global links, to dragonfly, where a small percentage of links are global links, we attempted to verify the advantages of constructing all links as global links in FleX. In the worst case traffic pattern of dragonfly, each router sends packets only to the routers in the next group. As the layer in FleX has concept similar to that of a group in dragonfly, we applied a traffic pattern in which each router sends packets only to the next layer to FleX. With only the inter-layer connections and not intra-layer connections in which the layer corresponds to the group in dragonfly, we find that FleX achieves lower latency and higher throughput on the worst case traffic pattern (next group or layer traffic pattern).

In order to demonstrate the configuration flexibility of FleX, we also set a new configuration of a FleX network with a higher cost than the configuration in Table 6 and compared the latency outcomes with those of other topologies with bit complement traffic, which showed the worst performance in the previous configuration. The newly configured FleX network has a total of 960 terminals, 192 routers with 33 radix, and 2,688 links between each router, with a total configuration costs of 5.49×10^6\$ and power consumption of 17,740W. As shown in Fig. 13, there is a performance enhancement compared to the previous FleX configuration. FleX has the lowest latency and higher saturation throughput than dragonfly, Slim Fly, and HyperX. For LBR scheme, FleX achieves higher saturation throughput than the 3-level fat tree, which still has a higher configuration cost.

Finally, in order to observe the impact of the updating period (T) for LBR on the network, we evaluated the latency performances on the various values of T where LBR is applied to Flex in the random uniform traffic pattern. As shown in Fig. 14, the results show that the average network

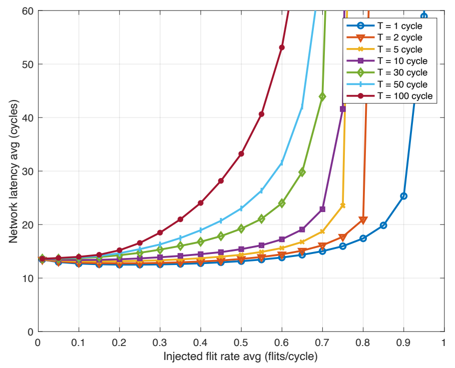


FIGURE 14. Latency performance on the various settings of the updating period (T) for LBR at FleX.

latency increases and the saturation throughput decreases as the updating period (T) grows up. As the status of the queue length for each port dynamically changes in the random traffic, the derived load balancing policy from LBR can lose its optimality as the time goes on from the recent updated time, which can contribute to the degradation on the latency and saturation throughput performance under the longer routing table updating period. Accordingly, setting the updating period (T) for LBR as small as possible can result in the highest latency performance. However, in the practical HPC system, the available minimum updating period (T) for LBR is limited depending on the hardware specifications, the total scale of the system, the implementational methods of interconnection management software, etc. Therefore, preliminary profiling about the available updating period is required to be conducted before operating the actual system for expecting the best performance via LBR.

C. OVER 3-LAYER IN FLEX

In this section, we evaluate cost and performance while increasing the number of layers in FleX to more than 3 to explore the impact of changing the number of layers on the cost and performance, as this is one of the flexibility factors in FleX, which can increase the network size under limited

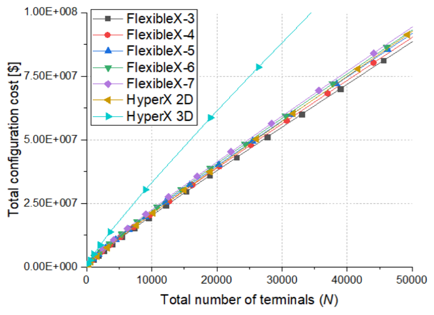


FIGURE 15. Cost of configuring FleX over the number of layers.

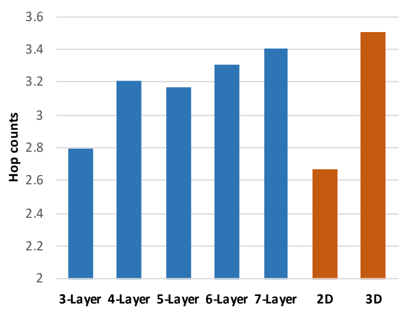


FIGURE 16. Average hop counts in FleX over the number of layers comparing to HyperX 2D and 3D.

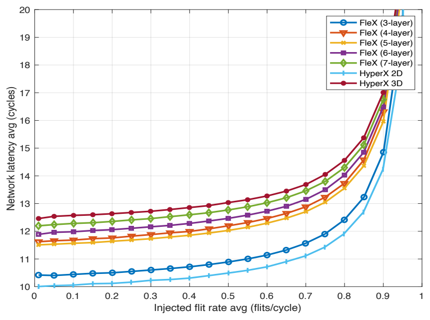


FIGURE 17. Latency performance of FleX over the number of layers.

radix conditions. We firstly calculated the total configuration cost of FleX with the various numbers of layers and compared these outcomes to 2-dimensional HyperX and 3-dimensional HyperX, as shown in Fig. 15. The cost calculation models and packaging rule for each topology were applied identically to the methods used earlier in Section IV-A. As shown in Fig. 15, the configuration cost of FleX increases by a smaller margin compared to HyperX (2D to 3D) as the number of layers (*i.e.,* network dimension of FleX) is increased, and the configuration costs for FleX with 4~7 layers show variations in the range between the cost of 3-layer FleX and 3-dimensional HyperX. Moreover, network diameters of FleX with 4~7 layers are all 3.

For a performance comparison, FleX is configured to have structures such that the number of layers is increased from 3 to 7 while $N_x = N_y = 6$ is fixed. HyperX is set to have 2-dimensional and 3-dimensional structures with dimension sizes of 6 in each dimension. To examine the tendency toward increased numbers of layers in FleX, the concentration (p) was unified to $p = 1$ on all compared structures. With these topology settings, we evaluated the results of minimal routing in the random uniform traffic pattern using the Booksim [26] simulator, as described in Section IV-B.

Fig. 16 shows the average hop count results at the offered load of 0.5 flits/cycle for each topology structure. As the number of layers increases in FleX, the ratio of the minimal routing path with a distance of 3 increases in the structure where only links between adjacent layers are established, resulting in an increased average hop count. In the case of 4-layer structure, the hop count exceeds that of the 5-layer structure, as the ratio of the minimum path with a distance of 3 in total was higher than that of the 5-layer structure due to the structural characteristics. In addition, FleX with 4$\sim$7 layers shows the average hop count variations between those of 3-layer FleX with a network diameter of 2 and 3D HyperX with a network diameter of 3, demonstrating that FleX can provide the flexibility on increasing the network size while maintaining the radix value.

Fig. 17 shows the latency performance for each topology structure. These results show a tendency similar to that in Fig. 16, and as the number of layers in FleX increases, the latency increases with an increase in the average hop count. Similarly, in the 4-layer case, the latency of 4-layer FleX exceeds that of 5-layer FleX, as the ratio of the minimal path with a distance of 3 is larger in the 4-layer structure. In addition, for 3-dimensional HyperX, the network size is identical to that of 6-layer FleX, but there are fewer inter-router links, showing higher latency and a higher average hop count. On the other hand, for 2-dimensional HyperX, as the network size is smaller than that of 3-layer FleX, 2-dimensional HyperX shows lower latency and a lower average hop count than 3-layer FleX.

These results show that FleX can provide more flexible options when configuring an interconnection network by offering an adjustable number of layers. Specifically, it is shown that FleX can expand the network size of existing systems by easily adding layers without expanding the radix, with only small latency degradation.

V. CONCLUSION

We introduced FleX topology, which can be configured easily, is cost-effective, and can produce low latency performance with a low-diameter structure. By removing all connections within the layers and connecting only the neighboring layers as a ring, a low-diameter network could be established in a 3-layer structure. In addition, due to the absence of comlex constraints for configuring the network in FleX, several variations of configurations can be created easily to find suitable configurations for a given fixed budget.

In addition, we analyzed the various structural properties of FleX and proposed a minimal routing algorithm for the proposed topology. The proposed topology was evaluated in terms of the configuration cost and power consumption. The evaluation results showed that FleX is more efficient in terms of cost and power consumption compared to other topologies.

We also evaluated the latency performance of the minimal, UGAL, and LBR routing in the proposed topology through a cycle accurate simulator. The latency performance results showed that the FleX achieves consistent performance against various traffic patterns as well as lower latency and higher saturation throughput at the same cost, as all links in FleX have a symmetrical structure that allows them to have the same load. Moreover, we identified that the proposed minimal routing algorithms works correctly by examining a hop count histogram of the packets, also finding that LBR can largely extends the saturation throughput in FleX though it sacrifices latency in small portion.

Finally, by conducting a brief evaluation of the cost and performance with regard to the number of layers, we verified that FleX can provide a various alternatives to suit a range of various configuration environments by adjusting the number of layers, which is a key flexibility factor of FleX. As future work, it is expected to find a more efficient way to use the high bisection width of FleX by adaptively selecting the multiple minimal paths available in FleX according to the load.

In conclusion, FleX is demonstrated to be highly applicable, as it shows low latency performance and incurs only a low cost when configuring or managing an interconnection network. Being synergized with LBR, FleX can expand its saturation throughput further while only sacrificing the latency slightly. Moreover, FleX offers a strong advantage in that it can flexibly changes the scale of the physical interconnect system by easily adding/removing the layers. As FleX has only the inter-layer links on neighboring layers, it is easy to add or remove the layers physically. Therefore, it can be easily and rapidly coped with changing the scale of the HPC system (a case in which expanding the scale would be predominant).

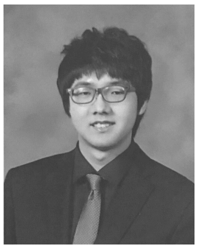
However, as FleX does not contain any intra-layer links, Flex has a structural weakness in how it accommodates local traffic patterns, which is a common limitation of other cost-effective networks as well (*e.g.,* dragonfly, Slim Fly, HyperX). Our LBR shows that it can mitigate this limitation but the limitation also cannot be largely resolved in certain traffic patterns, such as neighbor or tornado traffic patterns. The development of topological solutions or other routing schemes that can endure such local traffic patterns are remained as future works.

REFERENCES

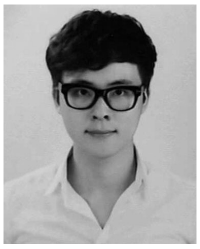
[1] John Kim, Wiliam J Dally, Steve Scott, and Dennis Abts. Technology-driven, highly-scalable dragonfly topology. In 2008 International Symposium on Computer Architecture, pages 77–88. IEEE, 2008.
[2] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S Schreiber. Hyperx: topology, routing, and packaging of efficient large-scale networks. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, page 41. ACM, 2009.
[3] John Kim, William J Dally, and Dennis Abts. Flattened butterfly: a cost-efficient topology for high-radix networks. ACM SIGARCH Computer Architecture News, 35(2):126–137, 2007.
[4] Maciej Besta and Torsten Hoefler. Slim fly: A cost effective low-diameter network topology. In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pages 348–359. IEEE Press, 2014.
[5] Charles E Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. IEEE transactions on Computers, 100(10):892–901, 1985.
[6] Steven L Scott et al. The cray t3e network: adaptive routing in a high performance 3d torus. 1996.
[7] Youcef Saad and Martin H Schultz. Topological properties of hypercubes. IEEE Transactions on computers, 37(7):867–872, 1988.
[8] Arjun Singh. Load-balanced routing in interconnection networks. PhD thesis, Stanford University, 2005.
[9] Pasquale De Luca, Stefano Fiscale, Luca Landolfi, and Annabella Di Mauro. Distributed genomic compression in mapreduce paradigm. In International conference on internet and distributed computing systems, pages 369–378. Springer, 2019.
[10] Long Chen, Oreste Villa, Sriram Krishnamoorthy, and Guang R Gao. Dynamic load balancing on single-and multi-gpu systems. In 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), pages 1–12. IEEE, 2010.
[11] SUMMIT Supercomputer. Available: https://www.olcf.ornl.gov/olcf-resources/compute-systems/summit/.
[12] SIERRA Supercomputer. Available: https://computing.llnl.gov/tutorials/sierra/.
[13] Heng Lin, Xiongchao Tang, Bowen Yu, Youwei Zhuo, Wenguang Chen, Jidong Zhai, Wanwang Yin, and Weimin Zheng. Scalable graph traversal on sunway taihulight with ten million cores. In 2017 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 635–645. IEEE, 2017.
[14] Andrew Komornicki, Gary Mullen-Schulz, and Deb Landon. Roadrunner: Hardware and software overview. IBM Redpaper, 2009.
[15] Greg Faanes, Abdulla Bataineh, Duncan Roweth, Edwin Froese, Bob Alverson, Tim Johnson, Joe Kopnick, Mike Higgins, James Reinhard, et al. Cray cascade: a scalable hpc system based on a dragonfly network. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, page 103. IEEE Computer Society Press, 2012.
[16] Jack Dongarra. Visit to the national university for defense technology changsha, china. Oak Ridge National Laboratory, Tech. Rep., June, 2013.
[17] Cray XC series. Available: http://www.cray.com/products/computing/xc-series.
[18] Piz Daint Supercomputer. Available: https://www.cscs.ch/fileadmin/user_upload/contents_userLab/building_software_piz_daint.pdf.
[19] Q. Koziol. High Performance Parallel I/O. Chapman & Hall/CRC Computational Science. Taylor & Francis, 2014.
[20] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. In ACM SIGCOMM Computer Communication Review, volume 38, pages 63–74. ACM, 2008.
[21] Top500 Supercomputers. Available: http://www.top500.org/.
[22] Chane-Yuan Yang, Chi-Hsiu Liang, Hong-Lin Wu, Chun-Ho Cheng, Chao-Chin Li, Chun-Ming Chen, Po-Lin Huang, and Chi-Chuan Hwang. Exceeding the performance of two-tier fat-tree: Equality network topology. In Future of Information and Communication Conference, pages 1187–1199. Springer, 2019.
[23] Chi-Hsiu Liang, Chun-Ho Cheng, Hong-Lin Wu, Chao-Chin Li, Chun-Ming Chen, Po-Lin Huang, Sang-Lin Huang, and Chi-Chuan Hwang. Beyond the performance of three-tier fat-tree: Equality topology with low diameter. In 2018 International Symposium on Computer, Consumer and Control (IS3C), pages 22–29. IEEE, 2018.
[24] Hong-Lin Wu, Chun-Ho Cheng, Chi-Hsiu Liang, Chao-Chin Li, Chun-Ming Chen, Po-Lin Huang, Sang-Lin Huang, and Chi-Chuan Hwang. Beyond the performance of 3d-torus: Equality topology with low radix. In 2020 International Symposium on Computer, Consumer and Control (IS3C), pages 319–322. IEEE, 2020.
[25] Chun-Ho Cheng, Hong-Lin Wu, Chi-Hsiu Liang, Chao-Chin Li, Chun-Ming Chen, Po-Lin Huang, Sang-Lin Huang, and Chi-Chuan Hwang. Equality noc: A novel noc topology for high performance and energy efficiency. In 2020 International Symposium on Computer, Consumer and Control (IS3C), pages 83–86. IEEE, 2020.
[26] Nan Jiang, Daniel U Becker, George Michelogiannakis, James Balfour, Brian Towles, David E Shaw, John Kim, and William J Dally. A detailed and flexible cycle-accurate network-on-chip simulator. In 2013 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pages 86–96. IEEE, 2013.
[27] HPE lab. Available: http://www.labs.hpe.com/next-next/exascale.
[28] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on scientific Computing, 20(1):359–392, 1998.
[29] George Karypis and Vipin Kumar. Multilevelk-way partitioning scheme for irregular graphs. Journal of Parallel and Distributed computing, 48(1):96–129, 1998.
[30] Leslie G. Valiant. A scheme for fast parallel communication. SIAM journal on computing, 11(2):350–361, 1982.

[31] Willi Homberg. Ways to decrease tco and infrastructure related costs. Prace white paper, Jülich Supercomputing Centre (JSC), 2017.
[32] Dennis Abts, Michael R Marty, Philip M Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. ACM SIGARCH Computer Architecture News, 38(3):338–347, 2010.
[33] Price of HPC and Data Center Gear. Available: http://colfaxdirect.com.
[34] Hang-Sheng Wang, Xinping Zhu, Li-Shiuan Peh, and Sharad Malik. Orion: a power-performance simulator for interconnection networks. In Proceedings of the 35th annual ACM/IEEE international symposium on Microarchitecture, pages 294–305. IEEE Computer Society Press, 2002.

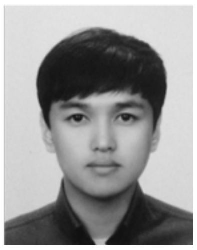
MINSU JEON received the B.S. degree in electronic engineering from Sogang University in 2016. He recieved the M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 2017. He is currently pursuing the Ph.D degree in electrical engineering at Korea Advanced Institute of Science and Techonolgy (KAIST). His research interests include deep learning (DL) application/model, DL serving and high performance computing system.



KYUNG-NO JOO received the B.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST). He recieved the M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 2014. He is currently pursuing the Ph.D degree in electrical engineering at Korea Advanced Institute of Science and Techonolgy (KAIST). His research interests include graph scheduling, mobile edge computing, and distributed training acceleration.



TAEWOO KIM received the B.S. degree in electrical engineering from Kyungpook National University, Deagu, South Korea in 2015, and M.S. degree in electrical engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea in 2017. He is currently pursuing the Ph.D degree in electrical engineering at Korea Advanced Institute of Science and Techonolgy (KAIST). His research interests include the Deep Learning(DL) framework, GPU computing, and interactive learning.



SEONGHWAN KIM received the B.S. degree in media and communications engineering from Hanyang University, Seoul, South Korea, in 2012, and the integrated master's and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2020. He is currently a Postdoctoral Researcher with the Network and Computing Laboratory, KAIST. His research interests include cloud brokering systems, deep learning accelerator, and high performance computing and others.

CHAN-HYUN YOUN (S'84–M'87–SM'2019) received the B.Sc and M.Sc degrees in Electronics Engineering from Kyungpook National University, Daegu, Korea, in 1981 and 1985, respectively, and the Ph.D. degree in Electrical and Communications Engineering from Tohoku University, Japan, in 1994. Before joining the University, from 1986 to 1997, he was a Head of High-Speed Networking Team at KT Telecommunications Network Research Laboratories, where he had been involved in the research and developments of centralized switching maintenance system, high-speed networking, and ATM network. Since 2009, he has been a Professor at the School of Electrical Engineering in Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea. He was an Associate Vice-President of office of planning and budgets in KAIST from 2013 to 2017. He also is a Director of Grid Middleware Research Center and XAI Acceleration Technology Research Center at KAIST, where he is developing core technologies that are in the areas of high performance computing, edge-cloud computing, AI acceleration system and others. He was a general chair for the 6th EAI International Conference on Cloud Computing (Cloud Comp 2015), KAIST, in 2015. He wrote a book on Cloud Broker and Cloudlet for Workflow Scheduling, Springer, in 2017. Dr. Youn also was a Guest Editor IEEE Wireless Communications in 2016, and served many international conferences as TPC member.