Distributed Shortcut Networks: Low-Latency Low-Degree Non-Random Topologies Targeting the Diameter and Cable Length Trade-Off

Nguyen T. Truong, Ikki Fujiwara, *Member, IEEE*, Michihiro Koibuchi, *Member, IEEE*, and Khanh-Van Nguyen

Abstract—Low communication latency becomes a main concern in highly parallel computers and supercomputers that reach millions of processing cores. Random network topologies are better suited to achieve low average shortest path length and low diameter in terms of the hop counts between nodes. However, random topologies lead to two problems: (1) increased aggregate cable length on a machine room floor that would become dominant for communication latency in next-generation custom supercomputers, and (2) high routing complexity that typically requires a routing table at each node (e.g., topology-agnostic deadlock-free routing). In this context, we first propose low-degree non-random topologies that exploit the small-world effect, which has been well modeled by some random network models. Our main idea is to carefully design a set of various-length shortcuts that keep the diameter small while maintaining a short cable length for economical passive electric cables. We also propose custom routing that uses the regularity of the various-length shortcuts. Our experimental graph analyses show that our proposed topology has low diameter and low average shortest path length, which are considerably better than those of the counterpart 3-D torus and are near to those of a random topology with the same average degree. The proposed topology has average cable length drastically shorter than that of the counterpart random topology, which leads to low cost of interconnection networks. Our custom routing takes non-minimal paths to provide lower zero-load latency than the minimal custom routings on different counterpart topologies. Our discrete-event simulation results using SimGrid show that our proposed topology is suitable for applications that have irregular communication patterns or non-nearest neighbor collective communication patterns.

Index Terms—Network topologies, small-world networks, interconnection networks, high-performance computing

1 Introduction

Designing low-latency interconnection networks is a main concern in highly parallel computers as they become larger, such as 3 M cores for Tianhe-2 [1]. The switch delay to forward a message becomes dozens or hundreds of nanoseconds, such as 45.3 ns on BlueGene/Q, 40.1 ns on Anton-2 [2] and about 100 ns even in a commodity InfiniBand QDR switch, which uses a forwarding table. The receiving and sending overhead at a host could be 100 ns by enabling intelligent network interfaces [3], [4]. As device technology and its corresponding software overhead continue to improve, an expected MPI-level communication naturally becomes latency sensitive, such as dozens of nanoseconds for custom supercomputers or hundreds for commodity clusters in 2018 [2], [5].

Low-latency network design has two trends in recent supercomputers: high-radix indirect networks, such as the use of the InfiniBand standard, and low-radix direct networks, such as BlueGene/Q, Anton-2 [2] and K-Computer [6]. In the high-radix indirect networks, such as fat trees, a traditional concern is to obtain a good degree-diameter trade-off in graph construction [7], because high-radix switches usually have a delay of around 100 ns, which is relatively larger than the cable delay. There are different types of such trade-off depending on the design priority of a proposal. For example, some important technology-driven topologies such as SlimFly [8] aim at a constant diameter as small as just two or three at the expense of possibly having high degrees.

In contrast, in the low-radix direct networks, which are our target in this study, since the switch delay is becoming as low as dozens of nanoseconds, the cable delay (5 ns per meter) also becomes a crucial concern to reduce communication latency. Low-latency topology design thus becomes complicated.

Although some prior topology designs take into account the cable delay for a low-latency network [9], [10], those designs use random shortcut links that require routing-table implementation by off-chip content-addressable memory (CAM) that may be costly when each switch is embedded in a compute node, i.e., a direct network. These new constraints bring a new challenge in low-latency network design to develop (1) a low-degree non-random network

- N.T. Truong is with the Graduate University for Advanced Studies (SOKENDAI), Hayama, Kanagawa Prefecture 240-0115, Japan. E-mail: nguyen.88.tt@gmail.com.
- I. Fujiwara is with the National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan. E-mail: ikki@nii.ac.jp.
- M. Koibuchi is with the National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, Japan, and the Graduate University for Advanced Studies (SOKENDAI), Hayama, Kanagawa Prefecture 240-0115, Japan. E-mail: koibuchi@nii.ac.jp.
- K.V. Nguyen is with the Ha Noi University of Science and Technology, 1 Dai Co Viet Road, Ha Noi, Viet Nam. E-mail: vannk@soict.hust.edu.vn.

Manuscript received 31 Aug. 2015; revised 11 July 2016; accepted 13 Sept. 2016. Date of publication 26 Sept. 2016; date of current version 15 Mar. 2017. Recommended for acceptance by B. Urgaonkar.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2016.2613043

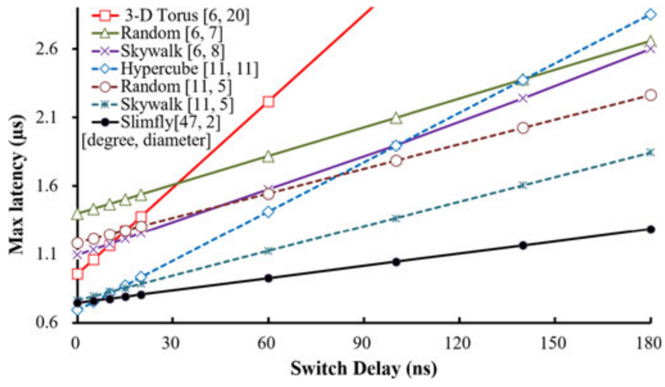


Fig. 1. Maximum latency along the shortest-hop paths versus switch delay in 2,048-switch networks (except for Slimfly, which has 1,922 switches). Links between switches and compute nodes are not considered. The legend indicates the degree and the diameter.

topology that has low-cable delay when mapped onto a floor plan, and (2) a custom routing algorithm that does not require routing-table implementation at switches.

For such direct networks, the ideal topology should have properties of both (1) low average shortest path length (ASPL) and low diameter, and (2) short cable length on a floor plan. Regarding the diameter, it is well known in graph theory that for a degree d and a network size n, the diameter lower bound, or Moore bound, is computed as the lowest value k such that $1 + d\sum_{i=0}^{k-1}(d-1)^i \geq n$. Unfortunately, only a few graphs satisfy this bound [7]. Besides, several topologies are proposed in a low-degree fashion, where the degrees are upper bounded by a rather low constant number. An n-vertex graph with a constant degree have a diameter no less than $\theta(\log n)$ asymptotically [11], [12], [13]. Let us say that such a graph or topology exhibits the constant-degree-and-logarithmic-diameter (hereafter "CDLD") trade-off optimality.

In fact, several traditional regular topologies, such as hypercubic topologies including the cube-connected cycle (CCC) and certain butterfly networks, reach this optimality of constant degree and logarithmic diameter. However, these traditional networks are considered difficult to deploy in a machine room, and are difficult to scale up, because they usually have a complicated, strict structure, especially in using high dimensions. By contrast, the tori (normally with quite a low number of dimensions) have a significantly higher diameter but easier and more cost-efficient to deploy.

Recently, random topologies including random shortcut networks (RSN) [13] have been proposed along this CDLD direction to go even further in reducing the diameter. In RSN, the diameter can be very close to the optimum value (even outperforming traditional CDLD topologies), but it is reported that the cable length of the random topology is about three times larger than that of hypercubes and tori [9]. This fact keeps the cable delay relatively large compared to the switch delay.

Fig. 1 shows the maximum end-to-end latency versus the switch delay for a 2,048-switch network. This graph indicates better topologies for different switch delay values. Data points are shown for the traditional hypercube and torus topologies, and for fully random topologies with different degrees [12], Skywalks which consider the cable delay [10], and Slimfly as a reference of high-degree topologies. The legend in the figure indicates the degrees and the diameters (i.e., the hop count between the furthest two nodes) for each topology. When the switch delay is zero (though this is unrealistic), the torus achieves the lowest maximum zero-load latency.[1] By contrast, when the switch delay is large, such as around 100 ns, random networks are better because the switch delay is a major factor of communication latency.

When the switch delay is set to 30 and 80 ns for degrees 6 and 11, respectively, the tori and the random low-degree topologies have a similar maximum end-to-end latency. This fact illustrates that both low diameter and short cable length are needed for low-latency communications in these switch-delay era. Skywalk, the random-based topology design, hits a good trade-off between diameter and cable length, leading to low network latency (as shown in Fig. 1). However, Skywalk requires routing-table implementation at switches which may be costly. Consequently, these switch-delay era requires a new non-random low-degree topology design targeting the path hops and cable length trade-off with a custom routing algorithm. In this context we propose *Distributed Shortcut Networks (DSNs)*, which are non-random networks that exploit certain aspects of the small-world effect, featuring the CDLD property. That is, our new DSN have strengths somewhat comparable to RSN, but have clear advantages in saving cables and in having a custom routing.

Below are the main contributions of this paper.

1) We propose a new non-random approach for designing cost-effective layout-conscious interconnect topologies with low degree, logarithmic diameter and low cable length.
2) We provide ring-based DSN for the theoretical analysis to the approach, and 2-D grid-based DSN topologies for the competitive use in practice. Both have a natural custom routing, where we consider the regularity of the topologies.
3) With regard to the diameter and cable length trade-off, our 2-D topology sacrifices a little in diameter if compared to the random topologies, i.e., 10.5 percent higher than RSN, but saves a lot in cable length, i.e., 31.2 and 64.5 percent shorter than tori and RSN.
4) The analysis shows that our custom routing has better network latency than 3-D Torus, especially in the low-latency switch era, e.g., 24.7 percent lower maximum latency when the switch delay is 30 ns/hop.
5) Discrete event simulation results show that the performance of network depends on the communication patterns generated by specific applications. Our proposed topology is suitable for applications that have irregular communication patterns or non-nearest neighbor collective communication patterns.

2 Distributed Shortcut Networks

2.1 Our Basic Approach

Our approach is learned from observing small-world network model, i.e., the Kleinberg's model [14]. The small-world model have been introduced to explain the structures

1. We assume that torus is not folded in the layout.

of popular real-world large-scale networks [15]. Typically, a small-world network is a simple regular graph such as a ring or torus with additional random links, which can drastically reduce the diameter. For example, in Random Shortcut Networks (RSN) [13], (uniform) random shortcuts are added to a ring of nodes. DLN-2^x [13] and Chord-like topologies [16] have logarithmic-diameter that can be formed by setting up each node with a collection of various-range shortcut links, typically links jumps 2^x nodes when the nodes are placed on a ring.[2] This collection of links with 2^x-ranges turns the routing task on the ring-based topology into a simple routine binary search.

Analysis of Kleinberg's small-world model shows that, if the size of local neighborhood $\theta(\log n)$ is large enough, the graph can have a logarithmic diameter with only one long link per node, because those long links collectively act as various-range shortcuts [11], [17]. This analysis suggests that, if we want to build a logarithmic-diameter graph, we can start with a simple base graph such as a ring or a grid (for providing some local connectivity) and then add shortcuts with the collection of all the 2^x-ranges, which we call the distance-halving links.

To design the constant-degree-and-logarithmic-diameter topologies, beside exploiting small-world model (for logarithmic-diameter), we use the supernode technique (for constant-degree). That is, we start with a base graph of supernodes and then add distance-halving shortcuts, and then we can replace each abstract supernode with a collection of regular nodes (typically $\log n$ or more nodes), and distribute the distance-halving links among these regular nodes. We also need to establish local connectivity among these regular nodes with some additional local links. An option often used for this internal structure of a supernode is a simple cycle.

In summary, our design approach is to start with a base graph of abstract supernodes, then add distance-halving links among them. The base graph can be a ring, a grid or any structure using a natural simple distance metric. The supernodes are then replaced with small subgraphs that act as 'local neighborhoods'. The inner topology of the supernode, i.e., this neighborhood subgraph, should be chosen cleverly so that we can create natural efficient custom routing for the whole network. The distance-halving links could be simply the links with 2^x-ranges, but other options are possible.

Based on this design approach, in Section 3 we describe in detail two specific design proposals with custom routing: one is for a ring and one is for a 2-D grid as the base graph. The reason we include the ring is because it is more straightforward, and looks similar to some traditional topologies, and therefore, easier for theoretical analysis. In other words, this ring-based topology is an intermediate step for aiding the discussion of the grid-based topology, which is more practical for deployment in a machine room (where supernodes can be naturally deployed as cabinets).

2.2 Framework for Comparing Network Topologies

One of the most important interconnect design factors is the correlation between the maximum degree and the graph diameter, or more precisely, the trade-off between these two. Interconnect topologies can be classified into classes of different *degree-and-diameter characteristics* where the degree and the diameter can be expressed in different asymptotic terms, such as a constant, or a logarithmic or polynomial of the size n, i.e., the number of vertices. We are interested in the following classes, to which most popular existing topologies belong.

- *The LDLD class*: topologies that have a logarithmic degree and a logarithmic diameter, i.e., both the degree and the diameter are $\theta(\log n)$. Typical examples are the hypercube and the DLN-2^x topologies. We can also find such topologies from the ones designed for peer-to-peer networks using distributed hash tables such as Chord.
- *The CDLD class*: topologies that have a constant degree and a logarithmic diameter. More precisely, a CDLD topology has a degree upper bounded by a constant and a diameter as $\theta(\log n)$. Some traditional hypercubic topologies such as the CCC topology and the butterfly networks are in this class. The recent random shortcut networks [13] and Skywalk [10] are also in this class. This class can be seen as an ideal trade-off, where interconnect topologies can enjoy both using only low-radix switches and having rather low communication latency.
- *The CDPD class*: topologies that have a constant degree and a polynomial diameter. The diameter of such a topology is not logarithmic (or poly-logarithmic) but a polynomial of n. The torus belongs in this class.
- *The PDCD class*: topologies that have a polynomial degree and a constant diameter. The complete graph (diameter 1), Dragonfly [18] and SlimFly [8] belong in this class. The topologies here are designed with the highest priority of minimizing the communication latency, but it is at the expense of using expensive high-radix switches.

It is well-known that the CDLD topologies achieve an optimal trade-off between degree and diameter asymptotically. Clearly, designing a topology satisfying this CDLD trade-off is a challenging task, but we have observed a technique (previously used but not explicitly stated) for transforming an LDLD topology into a CDLD one, or more precisely, constructing the CDLD topology by modeling the LDLD topology. In such a transformation—namely, having A, a LDLD graph, transformed into B, a CDLD graph—we can observe in B separate neighborhood subsets of nodes, each with size $\theta(\log n)$. We refer to these supersets as virtual supernodes, such that we can obtain A from B by collapsing these supernodes into regular nodes. We name this technique as "using virtual supernodes" after the observation that a single node in A becomes a local group, i.e., a virtual supernode in B. Each supernode has degree $\theta(\log n)$, but each regular node inside it has degree $O(1)$. By using some specific internal structure of each subgraph of B incurred by (the regular nodes within) a supernode, the diameter of B is also $\theta(\log n)$, just as that of A. As a typical example of using this supernode technique, one can augment a hypercube graph of size 2^k to obtain a CCC graph of size $k * 2^k$ by

2. The DLN-x, Distributed Loop Network of degree x [13], consists of n vertices arranged in a ring and additional shortcuts between vertices i and j such that $j = i + \lfloor n/2^k \rfloor \bmod n$ for $k = 1, \ldots, x-2$.

TABLE 1
Topology Comparison

Topologies	Degree-Diameter trade-off	Layout -conscious	Custom Routing
torus	CDPD	yes	yes
hypercube	LDLD	no	yes
CCC	CDLD	no	yes
butterfly	CDLD	no	yes
Dragonfly	PDCD	yes	yes
Slim Fly	PDCD	yes	yes
RSN	CDLD	no	no
DLN-2^x	LDLD	yes	yes
Skywalk	CDLD	yes	no
DSN	CDLD	yes	yes

replacing each node, a vertex of the hypercube, by a cycle of k nodes.

Spatiality exploitation can be seen as a characteristic of many interconnect topologies: typically, nodes are deployed as vertices of a grid (which could be as simple as a 1-D ring), and this grid-based structure can be exploited to provide useful design features. Examples include the torus, DLN (ring-based), and Skywalk (2-D grid based). By exploiting the grid geometry, often the links can be made as short as possible, and hence, the total cable cost of these topologies are among the lowest.

Combining the spatiality exploitation with the supernode technique can help to provide *layout-conscious* designing, which means designing interconnect topologies with features that can be easily and naturally negotiated for creating cost-efficient physical deployment in a machine room. The use of supernodes, which are small groups of regular nodes, makes such a topology lend itself naturally to being deployed in the machine room because each supernode can make a cabinet. The Skywalk topology evolved from the line of RSN [13]), but its design follows the layout-conscious principles, where the supernodes (i.e., cabinets) are placed at the vertices of a 2-D grid and shortcuts are created uniformly and randomly between these supernodes along the vertical or horizontal lines of the grid. Thus, Skywalk is much better in cable saving among the RSN.

While the most mentioned interconnect topologies have their own *custom routing algorithms*, the random shortcut topologies including Skywalk do not, because the random nature of the shortcuts requires constructing and storing full routing tables at each node. This could be a weak point because (1) the number of table entries may become extremely large proportional to the system size, such as over 500 K, and (2) a low-radix direct network usually prefer a compact implementation of the routing because its router chip is sometimes integrated to a processor chip.

Table 1 illustrates our framework of comparison between interconnect topologies for degree-diameter trade-off, layout-consciousness, and availability of custom routing. Our DSN is one of the layout-conscious topologies for low-radix switches. Compared to traditional low-radix topologies, ours offers smaller diameter comparable to recent RSN and its followers. These random topologies, however, do not have custom routing and also have higher cable cost.

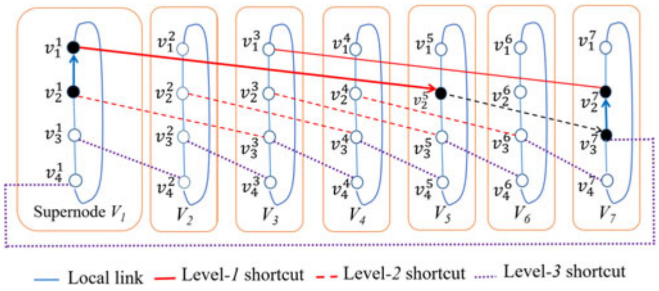


Fig. 2. 1-D DSN with 28 nodes arranged into a ring of seven supernodes. Some links are omitted.

3 Our Distributed Shortcut Networks and Properties

We exploit the non-random topology design based on small-world networks model presented in Section 2, beside existing random-based networks [11], [13], [14], [15], [16].

3.1 Basic 1-D Ring Topologies

First, we formulate a basic structure that is a ring with additional shortcuts tailored to support the distance-halving routing technique. We call this structure Ring of Distributed Shortcuts. The basic idea of this ring structure is that we can virtually see it as a (smaller) ring of supernodes, each of which occupies roughly the same number of adjacent regular nodes. Then we add a set of shortcuts with a variety of range to each supernode so that traveling between any two supernodes takes only $\log n$ such shortcuts or less (by using the distance-halving principle). We also have different variants of the ring structure with different ways to create the shortcuts.

We assume a ring of regular nodes are grouped into separate supernodes V_i, $i = 1, \dots, n$, which have roughly the same size m. For $z \le p = \lceil \log n \rceil$, these supernodes form a *type A Ring of Distributed Shortcuts* of degree z, denoted as *Ring-A*$[z, n]$. If we can pick a sequence of $z+1$ adjacent regular nodes $v_1^i, v_2^i, \dots, v_{z+1}^i$ for each supernode V_i such that $j = 1, \dots, z$, v_j^i is connected to v_{j+1}^i (called local link) and to v_{j+1}^k (called level-j shortcut), where $k = i + 2^{p-j} \bmod n$. Clearly, a local link connects nodes inside the same supernode while a level-j shortcut connects two different supernodes V_i and V_k. We also say that a level-j shortcut has a length $l_j = 2^{p-j}$ and z different-length shortcuts can go through V_i, i.e., z is the number of shortcuts of each supernode.

Similarly, we define a *type B Ring of Distributed Shortcuts* of degree z, denoted as *Ring-B*$[z, n]$. In which the destination of the level-j shortcut is $k = i + \lceil n/2^j \rceil \bmod n$, where the length is $l_j = \lceil n/2^j \rceil$. Compared to the type A, the type B provides some cable saving, although it somewhat complicates our custom routing mechanism.

Fig. 2 illustrates an example of our ring-based topology for the case of 28 nodes grouped into rings of $n = 7$ supernodes $V_1, V_2, \dots, V_7$, which have the same size, $m = 4$. Each supernode has $z = p = 3$ types of shortcuts with different lengths. Interestingly, Ring-A and Ring-B are constructed with the same length of shortcuts in this network size. In other words, level-2 shortcuts have length $2^{p-2} = \lceil n/2^2 \rceil = 2$.

Based on the above definitions of Ring-A and Ring-B structures, we define *directed* Ring-A or Ring-B as having all these above mentioned links to be directed with the direction suggested (from node u to v if we say "u is connected to v"). In both Ring-A$[z, n]$ and Ring-B$[p, n]$ where $z = p = \lceil \log n \rceil$, directed or not, we always have a short natural path from supernode V_s to V_t by using proper shortcuts and a few local links. Let $d_{st} = (t - s) \bmod n$ be the distance between these two supernodes. We choose k as the largest integer such that $\frac{d_{st}}{2} \leq l_k \leq d_{st}$, where $k = \lfloor \log \frac{2p}{d_{st}} \rfloor + 1$ in the case of Ring-A.[3] The path between V_s and V_t could start from $u_k^s \in V_s$ by taking the level-k shortcut (the movement that halves the distance to supernode V_{s+l_k}). At this intermediate supernode, we repeatedly find an available level-k' shortcut ($k' = k, \ldots, p$) and then continue jumping by using the distance-halving technique. Note that, the shortcut may use local links to move inside the supernode at each of these intermediate supernodes. For example, we present the routing path between the nodes v_2^1 and v_2^7 shown as the black nodes and the arrow lines in Fig. 2. First, we find the level-k shortcut, which moves half the distance between the source supernode V_1 and the destination supernode V_7. Based on our above description, this shortcut could be a level-1 shortcut, i.e., the path starts from a local movement from source node v_2^1 to v_1^1. Then, messages are routed to v_2^5 by using the level-1 shortcut. At this intermediate node, a suitable shortcut is selected again, e.g., the level-2 shortcut, which jumps to the destination supernode V_7. Finally, messages are sent to the target node by using the local links.

Overall, both our type A and B ring structures provide a basic topology for 1-D Distributed Shortcut Networks. Our custom routing mechanism provides natural short paths between supernodes. Routing between any two regular nodes can be formed by combining the routing between these supernodes and the routing within these super nodes (to reach a target regular node inside the source and the destination supernodes).

Normally, the internal routing in a supernode can simply use the basic ring links and hence takes $m/2$ hops on average. However, we can improve this routing by adding more local links to each supernode. The fact below summarizes our knowledge about the internal routing.

Fact 1. For any given graph G that has been embedded in a (directed) Ring-A or Ring-B$[p, n]$ with $p\lceil \log n \rceil$, the diameter of G is at most $p + 2\Delta$ if each subgraph of G induced by a supernode is connected and has a diameter at most Δ.

3.2 Basic Grid-Based DSN Topology

Below we describe the construction of our grid-based DSN of degree z (an integer parameter) with $n = X \times Y$ supernodes. This grid-based DSN is a 2-D extension of our basic ring structures mentioned above.

Grid: $n = XY$ supernodes are placed at vertices of an $X \times Y$ grid, where each supernode has a size of at least z. Assume that $\log X \geq z, \log Y \geq z$.

3. $k = \lfloor \log \frac{n}{d_{st}} \rfloor + 1$ in the case of Ring-B.

Locality: Per each supernode, arrange local links so that the regular nodes inside connect together to form a subgraph of diameter $\leq$ a given Δ and degrees $\leq$ a given δ. Normally, all these subgraphs are created in the same shape, which we call the *internal structure* of supernodes. The local links are also called internal links.

Rings: Construct shortcuts between supernodes so that all the supernodes form a directed Ring-A or Ring-B$[z, n]$ per each grid row or each column.

Let us now investigate the degrees. Given a grid-based DSN, we split graph G into a subgraph G_S that contains only all the shortcuts and a subgraph G_L that contains the remaining links, which are the local links mostly used to maintain the internal structure inside each supernode. The degrees in G_L are upper bounded by δ. Thus, the fact below is straightforward.

Fact 2. Given a grid-based DSN, each node has at most one outgoing and one incoming shortcut. The average degree of the graph is at most $\delta + 2$. For a supernode of size m, the average degree of the graph is $\leq \delta + 2\frac{z}{m}$.

The concept of our custom routing is simple. The whole routing process is a mix of two types of routing: routing within a supernode using the local links and routing within a ring using mainly the shortcuts. For convenience, we term the former *L-routing* (using the local links) and the latter *S-routing* (using the shortcuts).

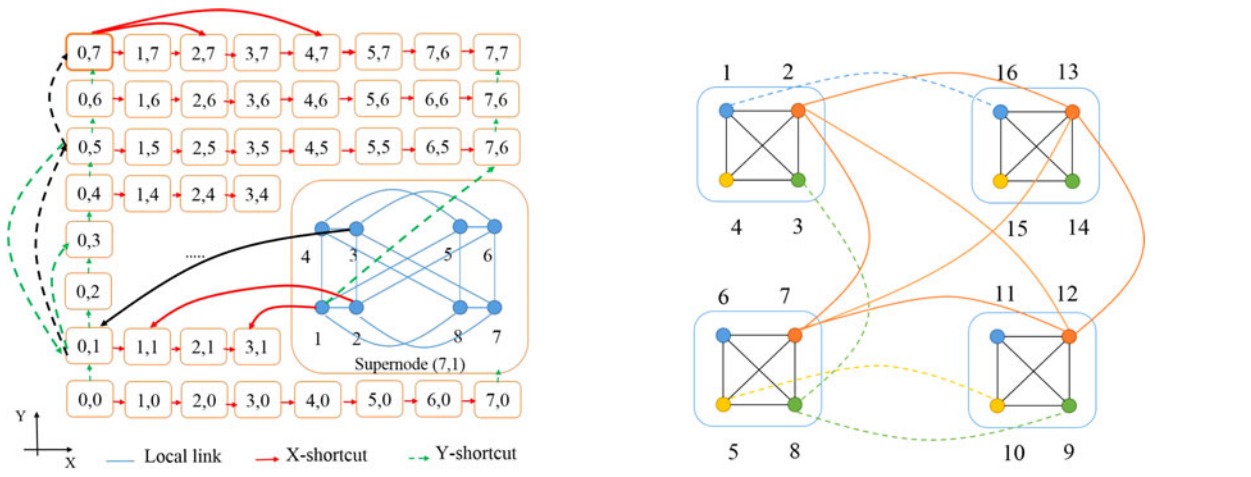
Suppose that we need to create a route from node u in supernode U to node v in supernode V. Without loss of generality, we assume that U and V are not in the same row or column of the grid. Let W denote the supernode that is in the same row as U and in the same column as V. Thus, the route can be formed as follows:

1) In the first phase, find a route from $u \in U$ to node $w \in W$.
2) In the second phase, find a route from $w \in W$ to node $v' \in V$.
3) In the third phase, find a local route to go from v' to v within V.

The routing in the first and second phases consists of L-routing initially and then S-routing. The final third phase is simple L-routing. So there are three stages of L-routing and two of S-routing. The path length is thus upper bounded by $3\Delta + \log X + 1 + \log Y + 1$. This helps to establish the following fact.

Fact 3. For the 2-D DSN defined above with $z = \lfloor \log X \rfloor = \lfloor \log Y \rfloor$, the graph diameter is at most $2z + 3\Delta + 2$ or at most $\log n + 3\Delta + 2$.

The detailed routing algorithms are described in Section 4.1. In the following, let us approximately evaluate the total cable length of the shortcuts and compare it with the counterpart in the random model. For simplicity, we assume for now that the distance between two adjacent supernodes is 1 and the distance between two adjacent regular nodes is $\ll 1$. It is easy to see that, per each grid row, the total cable length of the (row) shortcuts outgoing from a given supernode is $\leq 2^{\lceil \log X \rceil}$ for a type A grid DSN and $\leq X$ for a type B grid DSN. Similarly, per each grid column the length is $\leq 2^{\lceil \log Y \rceil}$ for a type A grid DSN and $\leq Y$ for a type B grid DSN. Thus, we have the



(a) 2-D DSN with 512 nodes arranged into grid of 8x8 supernodes. Each supernode includes 8 regular nodes. Some links are omitted.

(b) Internal structure of supernodes with 16 nodes arranged in CoC-16: a *complete* graph of 4 *complete* subgraphs of size 4. Since dashed inter-subgraph links are in the same form as solid links, some links are omitted.

Fig. 3. 2-D distributed shortcut network.

following fact on shortcut length, which also considers a special case (a standard case in practice) that the supernodes are deployed at the nodes of $N \times N$ grid ($X = Y = N$).

Fact 4. Let $X' = 2^{\lceil \log X \rceil}$ and $Y' = 2^{\lceil \log Y \rceil}$.

1) For an $X \times Y$ DSN of type A, the total shortcut length is at most $n(X' + Y')$ and the average shortcut length is at most $\frac{X'+Y'}{2z}$.
2) For an $X \times Y$ DSN of type B, the two factors above are at most $n(X + Y)$ and $\frac{X+Y}{2z}$, respectively.
3) For an $N \times N$ DSN, the average shortcut length is at most $\frac{2N}{z}$ for type A and at most $\frac{N}{z}$ for type B.

In a grid-based random model for creating shortcuts (such as the one in [10]), a shortcut can be created between any two supernodes uniformly and randomly chosen from a row/column of a given grid. As the calculation result of an elementary probability problem, the expected distance between two such nodes is a third of the row/column size. The following fact summarizes the properties of a special case, $2^p \times 2^p$ DSN, where types A and B are identical, and compares this case with the random grid model for cable length. In theory, the two topologies are in the same CDLD class but the grid-based DSN can potentially save much more cable length.

Fact 5. We compare our grid-based DSN with the random grid network discussed above:

(1) For a random network based on a $N \times N$ grid, the average shortcut length is $N/3$.
(2) For an $N \times N$ DSN of type B, the average shortcut length is at most about $\frac{3}{z}$ of the counterpart in the corresponding random grid network.
(3) Consider a $2^p \times 2^p$ DSN (A or B) where all the subgraphs induced by each supernode have constant degrees and a diameter upper bounded by a given constant Δ. The DSN then has constant degrees, a logarithmic diameter ($\leq 2p + 3\Delta$), and the average shortcut length is at most about $\frac{3}{p}$ of the counterpart in the corresponding random grid network.

3.3 Refining for Concrete Topologies and Extensions

We have described the basic ideas of our DSN topologies, which can be seen as a basic framework to create our DSN and achieve nice performance properties. The general DSN topology has three basic elements: 1) a grid for setting the supernode positions, 2) locality for setting the supernode internal structure, 3) rings for configuring the shortcuts. Any of these three can be refined or customized for achieving certain aims. We now use this basic framework to create concrete topologies for certain priority objectives. We also look at a possible extension that can further shorten the network cable length.

3.3.1 Concrete Topology of the 2-D DSN

In our DSN basic framework, the term *internal structure* of a supernode is rather abstract and is obviously open for refinement. Informally speaking, this term refers to a certain kind of shape desired to form the supernodes, that is, the way that the inner nodes (the network nodes that operate inside a given supernode) connect together. In general, each such graph structure can be characterized by three parameters: the size m, the diameter Δ, and the maximum degree δ. Ideally, a good structure should have Δ as low as possible given the size m and the maximum degree δ. In addition, a good structure helps to make the custom routing natural and simple enough. Moreover, under certain circumstances (e.g., in data centers), one may need to add new nodes or remove existing nodes, and so it is preferred to have simple operations for maintaining the basic shape and routing. Below we recommend and discuss a few choices, which we will evaluate in the later sections.

Using a hypercube shape could be a proper choice where dimension routing is a natural and perfect choice for local routing. For the supernode size $m = 8$, a 3-D cube has degree 3 and diameter 3. By adding just one more link, a diagonal in any face, as shown in Fig. 3a, we achieve diameter 2.

However, Fact 3 shows that an increment by 1 in Δ causes an increment by 3 in the diameter of the whole

TABLE 2
Notations of DSN

n	Number of supernodes
V_i	Supernode i
m	Number of nodes inside a supernode
v_k^i	Node k inside supernode V_i
z	Number of different range shortcuts
l_i	Length of level-i shortcuts, $l_i = 2^{p-i}$
d_{st}	Distance between V_s and V_t, $d_{st} = (t - s) \bmod n$

network; ideally, we want Δ to be only 1 or 2. Because using a complete graph as the internal structure can cause too high degrees, we focus on designing for $\Delta = 2$. The following is a good design for a structure of a 2-level hierarchy: the structure is virtually seen as a complete graph of size m_s of subgraphs, each of which is a complete graph of size m_n, where $m = m_s m_n$. Each node of this structure, which we will later refer to as *Complete of Complete* (CoC), has degree $(m_s - 1) + (m_n - 1)$. Fig. 3b illustrates this by giving a concrete case, namely CoC-16, where $m = 16$ and $m_s = m_n = 4$. This CoC-16 has diameter 2 while all the degrees are 6. We think this is good enough and consider further evaluation (numeric analysis and simulation) in later sections.

Another promising approach is as follows: starting with a 2-D grid or torus, add some long links to each node, where these links can be created in a specific pattern or randomly (uniformly). We may consider using this approach in future work.

3.3.2 Dropping the Longest Shortcut

Here we consider an extension where we stretch the concept of rings of shortcuts for producing topologies with an even shorter cable length. We consider dropping the longest shortcuts, the ones that go over a distance that is at least half of a grid row or column size (X or Y, respectively). For convenience, let us call them the *ring-halving links* and call this extension version as 2-D DSN-S (for “short” cable length). By definition it is easy to see that this drop can reduce the total shortcut length by about half. In practice, the cable saving could be up to 50 percent of the total shortcut length. Of course, this will affect the diameter and the average path length of the topology. We evaluate this effect empirically in the Section 5, but here we show that we can still modify our custom routing algorithm to adapt to this reduction.

We change the design so that all of Ring-A (or -B) is no longer directed and hence can accept traffic in both directions, i.e., the traffic flow can run along the rings in both directions. Per each ring, for any s-t pair of source and destination supernodes, we can choose the direction where the s-t hop distance is at most half of the ring size (i.e., grid size). Thus, our custom routing in this direction can still be satisfactory without using the ring-halving links. We describe more detail of this routing algorithm in Section 4.1.

4 Custom Routing Algorithms of DSN Topologies

4.1 Concrete Routing Algorithms

We remind the notations used to describe our topologies and routing algorithms as listed in Table 2. Consider a ring-based DSN topology of n supernodes, each of which is a group of m nodes. Each supernode has z different range shortcuts, e.g., the length of the level-i shortcut is $l_i = 2^{p-i}$ for $i = 1, \dots, z$. Routing between any two regular nodes can be formed by combining the routing between supernodes by using the distance-halving technique and the local routing within a given supernode, as mentioned in Section 3. The route can be shorter with a small refinement by allowing a message to travel along the ring in both directions (we call them FORWARD ROUTING and BACKWARD ROUTING). That is, for any V_s-V_t pair of the source and the destination supernodes, we can choose the direction to have a shorter d_{st} hop distance, which is half of the ring size at most. Fig. 4 shows our ring-based routing algorithms in detail with three main features:

- First, the BACKWARD ROUTING path from v_i^s to v_j^t could be easily found by reverting the order of the FORWARD ROUTING path from v_j^t to v_i^s.
- Second, in FORWARD ROUTING, we choose the level-k shortcut to jump out of the supernode where k is the largest integer such that $\frac{d_{st}}{2} \le l_k \le d_{st}$ (based on the distance-halving technique). Fig. 4 shows the formula to calculate k in the case of Ring-A topology. For Ring-B topology, $k = \lfloor \log \frac{n}{d_{st}} \rfloor + 1$.
- Finally, in the routing algorithm for DSN-S topology, the longest level-1 shortcuts are dropped. Therefore, the routing for a given pair of supernodes of distance $d_{st} = n/2$ is the combination of movement from V_s to intermediate supernode $V_{s'}$ by using the shortcut of the source node first, i.e., $s' = s + l_i$, then moving to V_t by using FORWARD ROUTING.

In the following, we look into the details of the concrete custom routing algorithm for the grid-based DSN topology,

The ring-based DSN custom routing algorithm

```
 1: procedure DSN-ROUTING($v_i^s, v_j^t$)
 2:     if $d_{st} \le n/2$ then
 3:         DSN-FORWARD-ROUTING($v_i^s, v_j^t$)
 4:     else
 5:         DSN-BACKWARD-ROUTING($v_i^s, v_j^t$)
 6:     end if
 7: end procedure

 8: procedure DSN-BACKWARD-ROUTING($v_i^s, v_j^t$)
 9:     $path \leftarrow$ DSN-FORWARD-ROUTING($v_j^t, v_i^s$)
10:     Revert order of $path$
11: end procedure

12: $u$ is an intermediate node in the routing path
13: $V_x$ is supernode that $u$ belongs to
14: procedure DSN-FORWARD-ROUTING($v_i^s, v_j^t$)
15:     $u \leftarrow v_i^s$                          ▷ Start at source node
16:     repeat
17:         $k \leftarrow \lfloor \log \frac{2^p}{d_{xt}} \rfloor + 1$
18:         if level of $u$ is not $k$ then
19:             $u \leftarrow v_k^s$          ▷ i.e., L-Routing from $u$ to $v_k^s$
20:         else
21:             $u \leftarrow u.Shortcut$                 ▷ i.e., S-Routing
22:         end if
23:     until $u$ in $V_t$                  ▷ i.e., reach supernode $V_t$
24:     $u \leftarrow v_j^t$                              ▷ i.e., L-Routing
25: end procedure
```

Fig. 4. Concrete custom routing for Ring-A DSN.

which is our final target in this study, by using the above ring-based routing algorithm. Given n supernodes placed in a grid of $X \times Y$, a supernode V_i now is identified by a tuple (x_i, y_i), where $1 \leq x \leq X$ and $1 \leq y \leq Y$. As the topology definition, each supernode is a group of m nodes with z different-length shortcuts in each dimension (X and Y). Fig. 3a is an example of 512 nodes in the 2-D DSN topology where the supernodes of eight nodes are arranged into an 8×8 grid. Here, each supernode has $z = 3$ shortcuts in the X dimension (solid red lines) and z other shortcuts in the Y dimension (dashed green lines). For example, the level-1 shortcut from node 1 in supernode (7,1) comes to supernode (x_k, y_k), where $x_k = (7+4) \bmod 8 = 3$ and y_k is the same as the source (also called the level-1 X-shortcut). Clearly, the X-shortcuts build up the ring-based DSN topology for each row and, similarly, the Y-shortcuts build up the ring-based DSN topology for each column.

Based on this observation, the routing algorithm of 2-D DSN topologies becomes simple by using the ring-based routing in two small steps. The routing path from node i in supernode (x_s, y_s) to node j in supernode (x_t, y_t) could be a concatenation of the ring-based routing path from the source node to an intermediate node u in supernode (x_t, y_s) and the ring-based routing path from u to the destination node. In the former routing path, called X-dimension routing, we use only X-shortcuts in the distance-halving technique. Similarly, in the later routing path, called Y-dimension routing, only Y-shortcuts are used. This simple routing mechanism looks similar to the routing of the 2-D torus in terms of dimension. Therefore, we can say that it is easy on the implementation and scalability perspective, e.g., when they have more dimension.

4.2 Achieving Deadlock-Free Routing

In this section we discuss how to refine our DSN topologies for achieving deadlock-free routing (when using cut-through or wormhole switching mechanisms). The idea is to refine the design of the internal structure of the supernodes as well as to customize the custom routing accordingly.

Let us briefly review our custom routing in this way:

- Phase A: L-routing within the source super node U (to reach a node with a proper horizontal shortcut)
- Phase B: S-routing horizontally along a row ring (to reach an intermediate supernode W, which is the intersection between the row of U and the column of V, to the destination supernode)
- Phase C: L-routing within supernode W (to reach a node with a proper vertical shortcut)
- Phase D: S-routing vertically along a column ring to reach V
- Phase E: S-routing within V to reach destination node t.

From this we observe that the two S-routing phases (B and D) naturally use separate links from the ones that can be used by the other phases (A, C and E) in most cases. In fact, if the internal structure is dense enough, we can easily separate the links being used between all the phases by elaborating on which links are used for which phases. Thus, the only deadlock issue remaining is to guarantee that no loop can be formed by the links used in the same phase by several routing tasks; that is, no loop can be formed among all the partial routes created inside a given particular phase.

This can also be done by carefully designing the supernode's internal structure and customizing our custom routing accordingly. Note that, since we want the diameter of structure inside supernode Δ to be as small as 2 or less, the local links are usually abundant for division in separate groups, each of which is fixed for a distinct responsibility.

4.3 Fault Tolerance

This section illustrates that our routing algorithm can be extended to have fault tolerance by enabling alternative paths to the destination. We assume that a node u has information about the neighbor failures, i.e., a faulty link (u, v) where v is an adjacent node of u. Note that a node has two type of links that include local-link and shortcut link. When a local-link fails, the routing can bypass it by taking advantages of the structure inside supernode in our L-Routing phase. Similarly, the routing can avoid faulty shortcut links in S-Routing phase by using another available shortcut link.

In Ring-based DSN, the routing path between a given pair of supernodes s and t is an ordered sequence of movements with different length, i.e., 2^{p-i}. This sequence can be modeled by $d_{st} = \sum_{i=1}^{p} a_i * 2^{p-i}$, $a_i \in \{-1, 0, 1\}$. We use 2^{p-i}-length shortcut to make a forward (or backward) S-routing movement if a_i equals to 1 (or -1), and L-routing otherwise. Without loss of generality, we assume that a 2^k-distance shortcut fails. We now look for the possible alternative paths for the path $d_{st} = 2^k + \sum_{i=p-k+1}^{p} a_i * 2^{p-i}$ by finding the alter movement of distance 2^k. We present two alternative paths as follows:

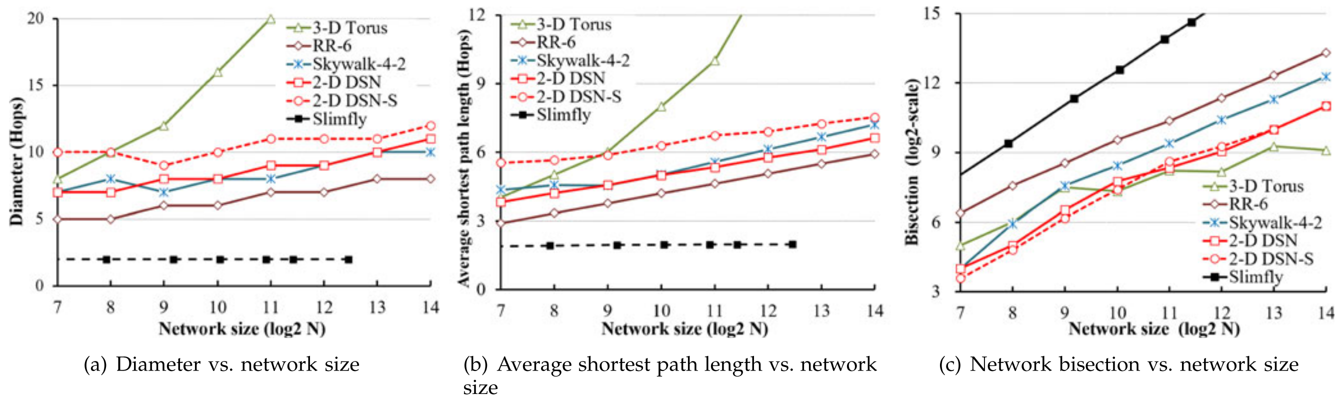
1) Use the shorter shortcuts to alter the faulty link, i.e., $d_{st} = 2^{k-1} + 2^{k-1} + \sum_{i=k+1}^{p} a_i * 2^{p-i}$.
2) Use the longer shortcuts to bypass the faulty link and then go backward to the destination if needed, i.e., $d_{st} = 2^{k+1} - 2^k + \sum_{i=p-k+1}^{p} a_i * 2^{p-i}$.

Clearly, both of two above alternative paths need two hops for S-Routing and at most $\Delta + 1$ hops for local traveling to reach the proper shortcut link where Δ is the diameter of structure inside supernodes. Overall, our path diversity increases at most $\Delta + 2$ hops when compared to the baseline that all links and nodes are healthy.

In Grid-based DSN, the custom routing for Grid-based DSN is the combination of routing in X-dimension and Y-dimension sequentially (denoted by XY-routing). Because the routing in each dimension is similar to Ring-based DSN routing mechanism, the routing method to avoid faults for Ring-based DSN can be applied for Grid-based DSN. Then, we can select the routing order, i.e., X-dimension after Y-dimension and vice versa, for taking shorter path hops avoiding faulty links.

5 Topology Analysis

In this section, we first use graph analyses to compare our newly proposed topologies with a typical non-random topology (torus) and random topologies such as RSN [13] and Skywalk [10] in terms of the diameter, the average shortest path length, and the network bisection. Next, we compute the average cable length, the network cost, and the network



(a) Diameter vs. network size (b) Average shortest path length vs. network size (c) Network bisection vs. network size

Fig. 5. Diameter, average shortest path length, and network bisection versus network size.

latency by considering their floor plans in a machine room to evaluate them from a practical perspective.

In this paper, we focus not only on our basic proposed topology (2-D DSN) but also on the extended version, which is refined for shortening the cable length (2-D DSN-S). Fact 2 showed that the average degree of our proposed topologies is around 6 when the network has 128 to 16,384 nodes. Thus, we compare them with the same-degree, same-sized counterparts. We choose 3-D Torus, RR-6 [13],[4] and Skywalk-4-2 [10][5] to be the representative of torus, RSN, and Skywalk, respectively. Also, Slimfly is considered as a reference of high-degree topologies in this comparison. We use the Slimfly topologies provided in [8] with different sizes (from 98 to 5,618 nodes) and degree (from 11 to 79).

5.1 Graph Analysis

5.1.1 Diameter and Average Shortest Path Length

We consider the topology scalability, i.e., how the diameter and the average shortest path length increase with the number of nodes, by means of graph analysis. Figs. 5a and 5b show the diameter and the average shortest path length of each topology. Smaller values are considered to be better.

In most network sizes, RR-6 achieves the smallest diameter and ASPL whereas 3-D Torus leads to the largest. 2-D DSN has the same result as Skywalk-4-2, which is much smaller than that of 3-D Torus, and is considered as a good achievement. As expected, the result of 2-D DSN-S is larger than 2-D DSN, since the longest shortcuts of 2-D DSN are removed in 2-D DSN-S. Specifically, when compared to 3-D Torus, the diameter and the ASPL of 2-D DSN decreases up to 78.8 and 74.5 percent, respectively. These values are not far from the smallest value, e.g., 10.5 percent larger ASPL than that of RR-6 in the 2^{14}-node networks.

In addition, the path lengths of all the compared topologies except for 3-D Torus increase slightly as the number of nodes increases. Therefore, we can say that these topologies have good scalability. The growth rate of 2-D DSN is lower than that of Skywalk-4-2 and similar to that of RR-6. This result implies that DSN may have better performance than Skywalk with the same degree when the network becomes even larger.

4. An RR-x topology is constructed of a ring and $x-2$ additional random shortcuts at each node.

5. A Skywalk-d_i-d_o topology of degree $d_i + d_o$ is tailored to map onto the physical layout of cabinets, as described in [10]. Each node has d_i intra-cabinet links and d_o inter-cabinet links.

5.1.2 Network Bisection

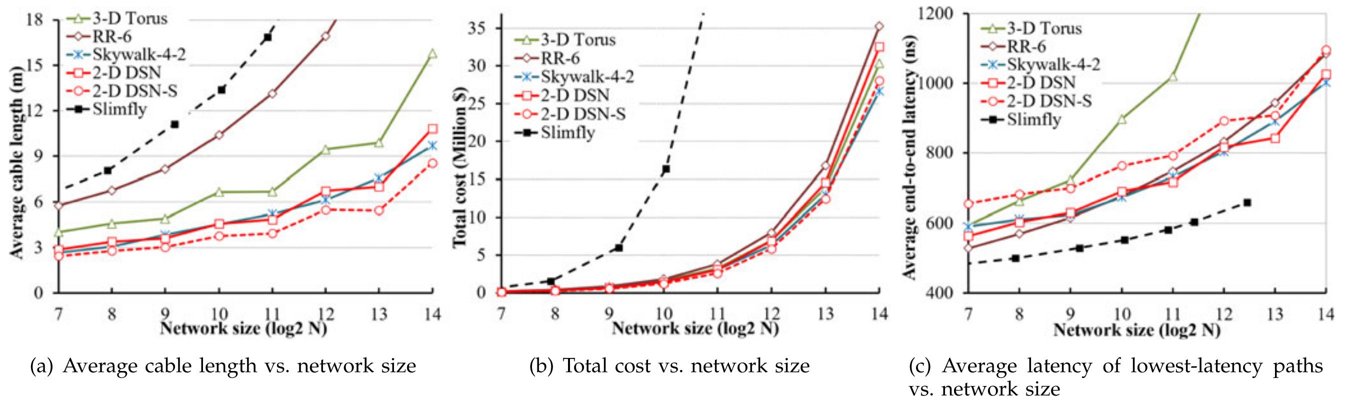
The bisection is one of the important characteristics of a network and is usually used to predict the network throughput from a theoretical viewpoint. We consider a p-way partition of a graph, where the graph is partitioned into p components that have nearly the same size in terms of the number of vertices as well as edges. We use *min-cut* to refer to the minimum number of edges that connect any two of the p components. A network bisection in a common sense is the min-cut computed with $p = 2$. In this study, we compute the cuts of the compared topologies by using METIS, which uses an efficient multi-level iterative approach [19].

Fig. 5c presents the min-cuts of the compared topologies with 2,048 nodes. Higher values are considered better. Clearly, the min-cuts of our proposed topologies are narrower than those of both RR-6 and Skywalk-4-2 but are wider than that of 3-D Torus. The gap between 2-D DSN and 3-D Torus becomes larger when the number of nodes increases. This result indicates that 2-D DSN has a higher bisection bandwidth compared to 3-D Torus. Therefore, we say that the DSN topologies have good performance in terms of throughput.

5.2 Layout Analysis

In this section, we estimate the cable length, the network cost and the latency when deploying the topologies onto a machine room floor. We assume that the compute nodes and the switches are enclosed in cabinets and the physical floor plan is large enough to organize these cabinets into a 2-D grid layout. Each cabinet occupies space 0.6 m wide and 2.1 m deep, including space for the aisle, as in the recommendations in [20]. We consider c cabinets arranged into an $x \times y$ grid, where $x = \lceil \sqrt{c}\ \rceil$ is the number of cabinet rows, and $y = \lceil c/x \rceil$ is the number of cabinets per row.

When deploying the topologies of switches into cabinets, how to assign a switch to a particular cabinet is an important question (we call this a mapping problem). We assign the switches to the cabinets sequentially by considering the order of these switches in the logical topologies. Interestingly, RSN and Skywalk are constructed in any mapping method. For 2-D DSN, nodes in the same supernode are deployed in the same cabinet. That is, a supernode in the logical design are mapped exactly to a cabinet in a physical deployment. For 3-D Torus, switches that have the same pair of x-index and y-index are put in the same cabinet. That is, a cabinet at physical location (x, y) includes



(a) Average cable length vs. network size (b) Total cost vs. network size (c) Average latency of lowest-latency paths vs. network size

Fig. 6. Average cable length, total cost, and average latency of lowest-latency paths versus network size in 8-switches/cabinet networks of 40-ns switches and 40-Gbps 5-ns/m switch-to-switch cables.

switches with logical index $(x, y, *)$. Since a supernode of our proposed topologies includes eight nodes, we assume that each cabinet has eight switches.

5.2.1 Average Cable Length and Layout

Cable length is one of the drivers of topology deployment cost and is also an important factor of low communication latency in the low-switch-delay era. For a practical perspective, we first measure the average cable length based on the method in [21]. In this method, cables between compute nodes and switches are ignored, since their lengths are constant regardless of the layout. Cables between switches in the same cabinet (called intra-cabinet cables) are 2 m long. Cables between switches in different cabinets (called inter-cabinet cables) are computed by using the Manhattan distance of cabinets and additional overhead (2 m at each cabinet).

The average cable length of each topology is shown in Fig. 6a. Lower values are considered better. As expected, Skywalk, which is a random topology optimized for the cable length, achieves a short average cable length. Our proposed topologies also have a short average cable length by using the above mapping method, where the switches in the same supernode are mapped into the same cabinet. Therefore, most of the links are implemented by short intra-cabinet cables, i.e., four intra-cabinet and two inter-cabinet cables per each switch. As a result, 2-D DSN has similar values compared to Skywalk-4-2. The average cable length is reduced up to 31.2 and 64.5 percent when compared to 3-D Torus and RR-6, respectively. It is reasonable that 2-D DSN-S, our extended topology refined for shortening the cable length, achieves the best result. Its cable length decreases from 14 to 20 percent than that of 2-D DSN when the network size increases. Therefore, DSNs are non-random topology optimized for short cable length, which lead to low cost and low end-to-end latency.

5.2.2 Network Cost

We now provide a further cost comparison of the above topologies. We use the cost model mentioned in [8], which considers both the cost of switches and the cost of interconnection cables. In this model, the switch cost, based on the Mellanox IB FDR10 switches, is a linear function of the switch degree or the radix. On the other hand, the cost of both electrical and optical cables depends on the cable length and the link bandwidth [8]. In this study, we set the cable bandwidth to 40 Gbps.

Fig. 6b plots the total cost of networks versus the network size. Although 2-D DSN has a shorter average cable length, its cost is higher than 3-D Torus when the network size is larger than 4,096. This result indicates that the switch cost of 2-D DSN is higher than that of 3-D Torus. When looking into detail, it is reasonable because the switch cost depends on the switch radix (or the node degree). As we mentioned before, our proposed topologies have an average degree around 6 (specifically, the networks of 1,024, 2,048, and 4,096 nodes have average degrees 5.75, 6, and 6.25, respectively). The larger the network size, the higher the average degree of 2-D DSN, which leads to higher cost of both the cables and the switches. The increment of the degree also affects the cost of 2-D DSN-S. However, the results are approximately the same as those of Skywalk-4-2 and lower than 3-D Torus and RR-6. We say that 2-D DSN-S achieves low cost in the low-degree topology design.

5.2.3 End-to-End Latency

In this section, we measure the end-to-end latency of the lowest-latency paths in order to examine the lowest performance that the compared topologies could support. Remember that our DSN topologies focus directly on the low-latency switch era. Thus, we set the switch delay to 40 ns/hop and the cable delay is set to 5 ns/m.

The network latency versus network size is shown in Fig. 6c. Lower values are considered better. These results show that 2-D DSN achieves lower latency than 3-D Torus with the same degree. For example, the average latencies of 2-D DSN are 29.70 and 57.83 percent lower than those of the 3-D Torus in network of 2,048 and 16,384 nodes, respectively. When compared to RR-6, we see that our 2-D DSN almost always lead to lower latencies (up to 10 percent) with a few exceptions. The exceptions correspond to the cases of small networks. In these cases, the good effect of the shorter cable length on the cable delay is not large enough to overcome the bad effect of the larger hop counts on the switch delay. It is remarkable that 2-D DSN outperforms RR-6 and is similar to Skywalk-4-2, although it is well known in previous work that topologies exploiting randomness gain a drastic reduction in latency [10], [13]. Therefore, we say that our 2-D DSN are low-latency low-degree non-random topologies.

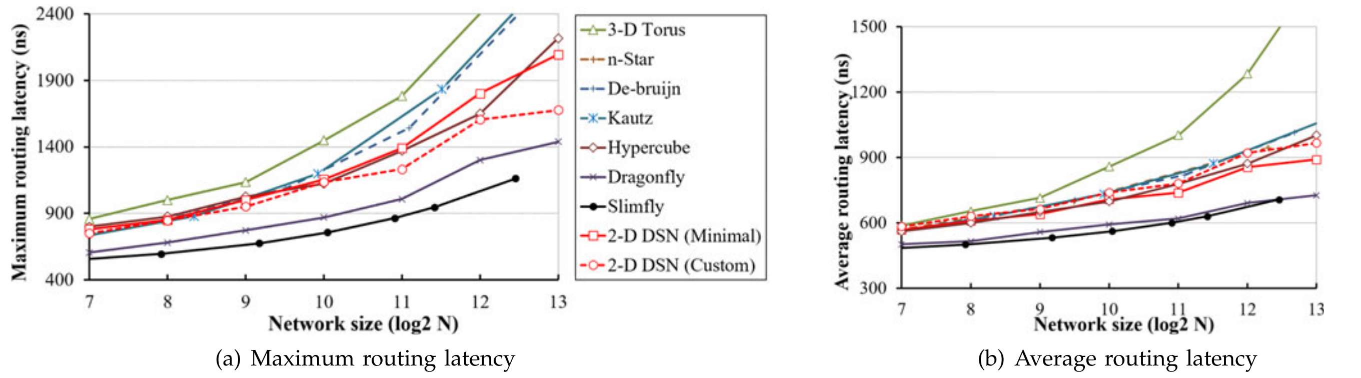


Fig. 7. Maximum and average routing latency versus network size. All the topologies have degree of 6 except for Hypercube (from 7 to 13), Dragonfly (from 9 to 135), and Slimfly (from 11 to 79).

6 Custom Routing Analysis

We now measure the end-to-end latency of our custom routing algorithm described in Section 4.1 and compare the latency to that of some well-known low-radix traditional topologies that support minimal custom routing, i.e., shortest-hop paths. We choose 3-D Torus, n-Star, De-bruijin, and Kautz as the low-degree counterparts with degree of 6. We also choose Hypercube, Dragonfly [18], and Slimfly as high-degree competitive topologies.

In this study, the routing latency depends not only on the routing path hops (switch delay) but also on the physical layout in a machine room (cable delay). We assume that each of the compared networks is optimally deployed in a floor plan. We do this by using the mapping and the layout optimization method in [22].

6.1 Maximum and Average Routing Latency

Fig. 7 represents the maximum and the average end-to-end routing latency when the switch delay is set to 40 ns/hop. The lower value is considered better. Not surprisingly, Slimfly leads to the best result regardless of the network size, since it has a high degree compared to the other topologies, i.e., degree 35 compared to 6 of the 2-D DSN in the case of 1,024 nodes.

For our proposed topologies, we estimate two routing algorithms, which are the ideal minimal routing and our custom routing. 2-D DSN with our custom routing achieves lower latency than the other low-degree topologies with the minimal routing. Its performance is comparable to that of Hypercube with the minimal routing, since both topologies achieve logarithmic routing path hops. In addition, for our 2-D DSN topology, the custom routing gains a better performance than the minimal routing in terms of maximum routing latency. This result shows that the cable delay becomes an important factor of the end-to-end latency in the low-latency switch era.

6.2 Routing Latency versus Switch Delay

To measure the range of switch delay in which our custom routing could achieve good performance, we estimate the routing latency of 1,024-switch networks while changing the switch delay. Fig. 8 shows the maximum and the average routing latency versus the switch delay. For each topology, the legend in the figure indicates the network size, the maximum degree and the routing diameter, i.e., the longest routing path length. In most cases, when the switch delay is increased from 0 ns (the ideal case) to 100 ns, of course, Slimfly leads to the lowest latency while 3-D Torus leads to the highest. 2-D DSN with our custom routing always achieves much lower latency than 3-D Torus, e.g., 24.7 and 14.5 percent lower maximum and average latency, respectively, in the case that the switch delay is set to 30 ns/hop. Especially, when compared to Hypercube (which has a higher degree), although the custom routing algorithm of 2-D DSN has higher average latency, it has a similar result in terms of maximum routing latency.

6.3 Fault Tolerance

We evaluate the fault tolerance capability of our proposed routing by randomly removing 0 to 20 percent of the links from the network of 1,024 switches. Beside our two proposed mechanisms, namely XY-routing and XY-YX-routing (which

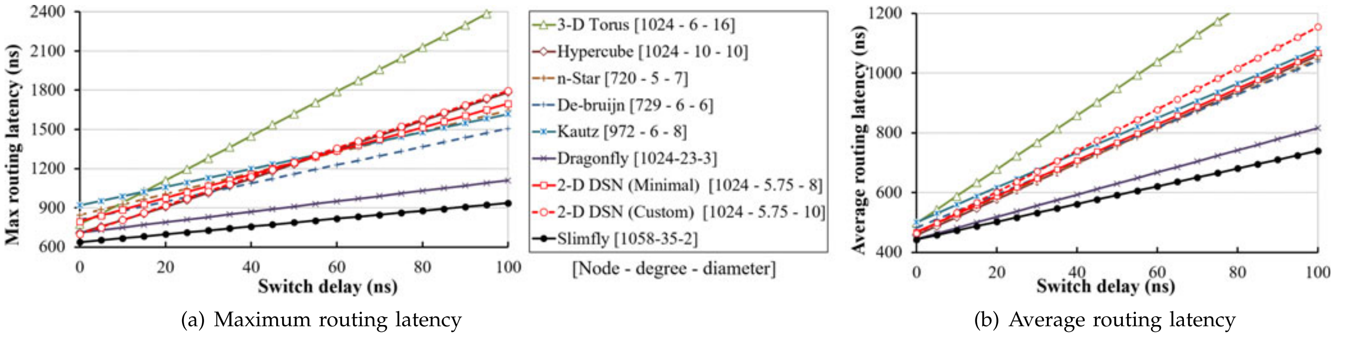


Fig. 8. Maximum and average routing latency versus switch delay in network of 128 cabinets, 1,024 switches.

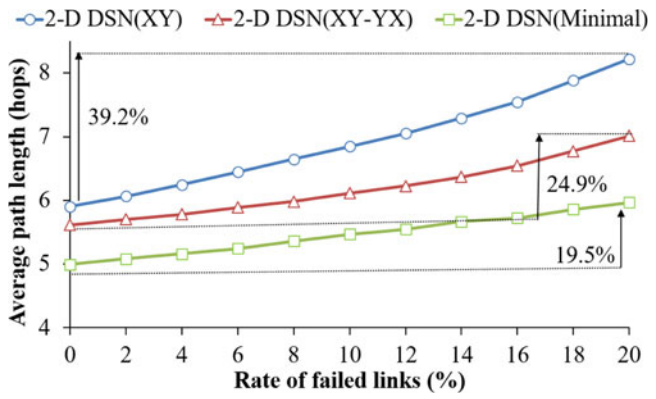


Fig. 9. Average routing path length versus the rate of faulty link in network of 1,024 switches.

chooses the shorter path between XY-routing and YX-routing), we consider the minimal routing as the base line.

Fig. 9 represents the average routing path length versus the ratio of faulty links against the number of total links. In most cases, as expected, the XY-routing has the longest routing paths while the minimum routing has the shortest. The XY-YX has similar shape of curve when compared to that for the minimal routing. For example, the increasing-rates of path length are 8.82 and 24.86 percent for XY-YX while these are 9.45 and 19.49 percent for minimal routing at the faulty link rate of 10 and 20 percent, respectively.

7 Simulation Results

We evaluate the performance of parallel applications and benchmarks when executed on our proposed topologies and existing same-degree topologies by means of an event-discrete simulation.

7.1 Parameters

We use the SimGrid simulation framework (v3.12) [23]. SimGrid implements validated simulation models, is scalable, and simulates the execution of unmodified parallel applications that use the Message Passing Interface (MPI) [23]. We use NAS Parallel Benchmarks (version 3.3.1, MPI versions) [24] (Class B for BT, CG, DT, LU, MG and SP, and Class A for FT benchmarks), the Himeno Benchmark [25], and the BigDataBench (version 3.1) [26] (K-means data clustering). It is reported that the network topology strongly affects the application performance due to the inter-node communication overhead that usually has spatial and temporal access locality [27]. We simulate various topologies of 256 nodes with 40-ns switch delay. Each host has 100 GFlops in computation speed. We configure SimGrid to utilize its built-in version of MVAPICH2 for MPI collective communications [28]. We measure the performance of the parallel applications executed on our proposed topology, where the custom routing is implemented for the communication between switches. The counterpart topologies, namely 3-D Torus, RR-6 and Skywalk-4-2, implements the minimal routing taking the shortest-hop path.

7.2 Evaluation

Fig. 10 shows simulation results for 3-D Torus, RR-6, Skywalk-4-2, and 2-D DSN topologies. The y-axis indicates the relative performance in Mop/s of each topology,

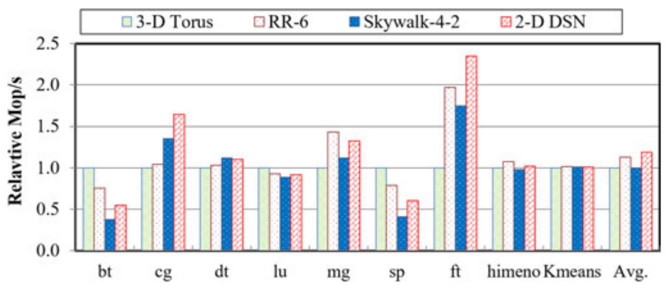


Fig. 10. Application performance evaluated using NAS Parallel, Himeno, and BigDataBench benchmarks. “Avg” means the average over all the benchmarks. Values are normalized to those of 3-D Torus.

normalized to that of 3-D Torus. Higher values are considered better. Overall, our proposed topology outperforms 3-D Torus by 19 percent on average. The random topologies (RR-6 and Skywalk-4-2) and our proposed non-random topology (2-D DSN) are better for applications that have collective communications across the system (e.g., CG, MG, FT and Himeno) or irregular communications (e.g., DT and K-means) because these topologies have lower end-to-end average latency than the torus. However, the torus topology has better performance than the others for benchmarks that generate a large number of neighboring communications (e.g., LU, SP and BT). We can see that the performance of a network topology depends on benchmarks, specifically on communication patterns. Our proposed topology is suitable for applications that have irregular communication patterns or non-nearest neighbor collective communication patterns.

8 Conclusion

In this study, we proposed a new non-random approach, named Distributed Shortcut Networks, for designing cost-effective layout-conscious interconnect topologies with low degree, logarithmic diameter and short cable length. Our approach is based on a common technique seen in traditional regular topology designs, e.g., the use of “virtual super-nodes”, combined with a different design philosophy learned from observing small-world networks, i.e., the Kleinberg’s model [14]. We discussed both the ring-based DSN and the 2-D grid-based DSN topologies, where the ring-based DSN provides insight and theoretical analysis to the approach while the 2-D grid-based DSN provides competitive and practical use. Both support a custom routing mechanism that exploits the regularity of the proposed topologies.

Our graph and layout analyses showed that the proposed topologies have low diameter and low average shortest path length, which are considerably better than those of 3-D Torus and near to those of random topologies (RSN [9] and Skywalk [10]) with the same average degree. Moreover, the average cable length and network cost of the proposed topologies are drastically shorter than those of RSN. In terms of end-to-end latency, it is remarkable that the DSNs outperform the RSN and is similar to Skywalk, although it is well known in previous work that topologies exploiting randomness have drastic reductions in latency [10].

As with a typical non-random topology, it is possible to exploit the structure of our DSN topologies to create a custom routing algorithm with a natural routing logic. We also proposed a custom routing algorithm, by which the routing logic at each switch is expected to be simple and small. Our latency analysis showed that our custom routing algorithm achieves a good result in the low switch-latency era. It is significantly better than the same-degree traditional topology such as 3-D Torus and is comparable to high-degree topologies such as Hypercube. The discrete event simulation showed that our proposed topology is suitable for applications that have irregular communication patterns or non-nearest neighbor collective communication patterns.

Acknowledgments

This work was partially supported by JST CREST, MIC/SCOPE(152103004), KAKENHI #16H02816, and the Vietnamese Ministry of Sci. and Tech (PI: Van-Khanh Nguyen).

References

[1] Top 500 Supercomputer Sites. (2015). [Online]. Available: http://www.top500.org/

[2] B. Towles, J. P. Grossman, B. Greskamp, and D. E. Shaw, "Unifying on-chip and inter-node switching within the Anton 2 network," in *Proc. ACM/IEEE Int. Symp. Comput. Archit.*, Jun. 2014, pp. 1–12.

[3] N. Tanabe, et al., "Low latency high bandwidth message transfer mechanisms for a network interface plugged into a memory slot," *Cluster Comput.*, vol. 5, no. 1, pp. 7–17, 2002.

[4] P. Kogge and J. Shalf, "Exascale computing trends: Adjusting to the "New Normal" for computer architecture," *Comput. Sci. Eng.*, vol. 15, no. 6, pp. 16–26, 2013.

[5] J. Tomkins, "Interconnects: A buyers point of view," presented at the ACS Workshop, Baltimore, MD, USA, 2007.

[6] Y. Ajima, S. Sumimoto, and T. Shimizu, "Tofu: A 6D mesh/torus interconnect for exascale computers," *IEEE Comput.*, vol. 42, no. 11, pp. 36–40, Nov. 2009.

[7] Combinatorics Wiki, "The degree diameter problem for general graphs," (2015). [Online]. Available: http://combinatoricswiki.org/wiki/The_Degree_Diameter_Problem_for_General_Graphs

[8] M. Besta and T. Hoefler, "Slim fly: A cost effective low-diameter network topology," in *Proc. IEEE/ACM Int. Conf. High Perform. Comput. Netw. Storage Anal.*, Nov. 2014, pp. 348–359.

[9] M. Koibuchi, I. Fujiwara, H. Matsutani, and H. Casanova, "Layout-conscious random topologies for HPC off-chip interconnects," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit.*, Feb. 2013, pp. 484–495.

[10] I. Fujiwara, M. Koibuchi, H. Matsutani, and H. Casanova, "Skywalk: A topology for HPC networks with low-delay switches," in *Proc. IEEE Int. Parallel Distrib. Process. Symp.*, 2014, pp. 263–272.

[11] J.-Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *Proc. 2nd ACM Symp. Cloud Comput.*, 2011, pp. 2:1–2:13.

[12] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking data centers randomly," in *Proc. USENIX Symp. Netw. Des. Implementation*, 2012, pp. 225–238.

[13] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, and H. Casanova, "A case for random shortcut topologies for HPC interconnects," in *Proc. Int. Symp. Comput. Archit.*, 2012, pp. 177–188.

[14] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. 32nd Annu. ACM Symp. Theory Comput.*, 2000, pp. 163–170.

[15] D. Watts and S. Strogatz, "Collective dynamics of small-world networks," *Nature*, vol. 393, pp. 440–32, 1998.

[16] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proc. Conf. Appl. Technol. Archit. Protocols Comput. Commun.*, 2001, pp. 149–160.

[17] C. Martel and V. Nguyen, "Analyzing Kleinberg's (and other) smallworld models," in *Proc. 23rd Annu. ACM Symp. Principles Distrib. Comput.*, Jul. 2004, pp. 179–188.

[18] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *Proc. Int. Symp. Comput. Archit.*, 2008, pp. 77–88.

[19] METIS—Serial Graph Partitioning and Fill-reducing Matrix Ordering. (2014). [Online]. Available: http://glaros.dtc.umn.edu/gkhome/metis/metis/overview

[20] HP, "Optimizing facility operation in high density data center environments, technoloogy brief," 2007. [Online]. Available: http://h18004.www1.hp.com/products/servers/technology/whitepapers/datacenter.html

[21] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: A cost-efficient topology for high-radix networks," in *Proc. Int. Symp. Comput. Archit.*, 2007, pp. 126–137.

[22] I. Fujiwara, M. Koibuchi, and H. Casanova, "Cabinet layout optimization of supercomputer topologies for shorter cable length," in *Proc. Int. Conf. Parallel Distrib. Comput. Appl. Technol.*, Dec. 2012, pp. 227–232.

[23] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, scalable, and accurate simulation of distributed applications and platforms," *J. Parallel Distrib. Comput.*, vol. 74, no. 10, pp. 2899–2917, 2014.

[24] The NAS Parallel Benchmarks. (2014). [Online]. Available: http://www.nas.nasa.gov/Software/ NPB/

[25] T. H. B. Himeno, "RIKEN," 2014. [Online]. Available: http://accc.riken.jp/2444.htm

[26] BigDataBench, 2016. [Online]. Available: http://prof.ict.ac.cn/BigDataBench

[27] F. Chaix, I. Fujiwara, and M. Koibuchi, "Suitability of the random topology for HPC applications," in *Proc. 24th Euromicro Int. Conf. Parallel Distrib. Netw.-Based Process.*, Feb. 2016, pp. 301–304.

[28] MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE, (2013). [Online]. Available: http://mvapich.cse.ohio-state.edu/

Nguyen T. Truong received the BE and ME degrees from Hanoi University of Science and Technology, Hanoi, Vietnam, in 2011 and 2014, respectively. He is currently working toward the PhD degree at the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan. His research interests include the areas of high-performance computing and interconnection networks.

Ikki Fujiwara received the ME degree from Tokyo Institute of Technology, Tokyo, Japan, in 2004, and the PhD degree from the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan, in 2012. He is currently a project assistant professor in the Information Systems Architecture Research Division, National Institute of Informatics, Tokyo, Japan. His research interests include the areas of high-performance computing and optimization. He is a member of the IEEE.

Michihiro Koibuchi received the BE, ME, and PhD degrees from Keio University, Yokohama, Kanagawa, Japan, in 2000, 2002, and 2003, respectively. He is currently an associate professor in the Information Systems Architecture Research Division, National Institute of Informatics, and the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan. His research interests include the areas of high-performance computing and interconnection networks. He is a member of the IEEE.

Khanh-Van Nguyen received the BE degree from Hanoi University of Science and Technology (HUST), Vietnam, in 1992, the MS degree from the University of Wollongong, Australia, in 1995, and the PhD degree from the University of California—Davis, in 2006. He is currently an associate professor in the School of Information and Communication Technologies, HUST. His main research domain is algorithms and theoretical models for computer networks and distributed computing.