Layout-Conscious Expandable Topology for Low-Degree Interconnection Networks

Thao-Nguyen TRUONG†, Khanh-Van NGUYEN††, Nonmembers, Ikki FUJIWARA†††, and Michihiro KOIBUCHI†††, Members

SUMMARY System expandability becomes a major concern for highly parallel computers and data centers, because their number of nodes gradually increases year by year. In this context we propose a low-degree topology and its floor layout in which a cabinet or node set can be newly inserted by connecting short cables to a single existing cabinet. Our graph analysis shows that the proposed topology has low diameter, average shortest path length and short average cable length comparable to existing topologies with the same degree. When incrementally adding nodes and cabinets to the proposed topology, its diameter and average shortest path length increase modestly. Our discrete-event simulation results show that the proposed topology provides a comparable performance to 2-D Torus for some parallel applications. The network cost and power consumption of DSN-F modestly increase when compared to the counterpart non-random topologies.

key words: Network expandability, network topologies, small-world networks, interconnection networks, high-performance computing

1. Introduction

A large number of data center and supercomputers are gradually increased their size year by year, because of the difficulty in precisely estimating future user demands and financial/political nature. Indeed, a large number of the TOP500 supercomputers have increased their computation power after deploying by increasing the number of processors. For example, K-computer was enlarged its size up to 28% from 2011 to 2012. In 2012, IBM BlueGene/Q, and Cray XK serial systems were also increased number of cores by 100%, and 87.76%, respectively [1]. In addition, commercial data centers have also been expanded. Facebook’s data center server population had been doubled from November 2009 by June 2010 [2].

Survey in [2] indicated that the current topologies such as Fat-tree and De Bruijn-like could not adapt the expansion requirement because of their coarse design size. For example, system with Fat-tree only has size 3456, 8192, 27648 if 24, 32, and 48 ports switches are used, respectively [2]. Using reservation ports for future expansion usage was proposed but it wastes resources and this issue will appear again when all of the free ports are used. In this study, we target on network-topology expandable property with two main points:

- **Arbitrary size**: the network topology could be built up with any required size.
- **Incremental expandability**: the network topology of \( n + x \) nodes could be built up from a topology of \( n \) nodes with any \( x \) nodes.

When increasing the number of nodes, a main design concern is to minimize (i) the number of rewired cables, (ii) the average length of newly introduced cables, and (iii) the degradation of the network efficiency. Each switch usually has a fixed number of ports and all the existing ports are usually used in the existing supercomputers. In such a case, it would thus be difficult to increase the switch degree when a new node is added to the network; re-wiring cables are needed to maintain the switch degree. The network efficiency affects the execution time of communication-sensitive parallel applications, because it may correspond to the diameter and the average shortest path length when their traffic patterns are unpredictable or dynamic depending on input parameters.

In this context we propose a layout-aware expandable topology and its efficient method to incrementally add nodes or cabinets in a machine room. This study is an extended version of the idea of our previous study [3]. In the topology design we use the small-world effect that does not rely on randomness for low path lengths, efficient incremental expansion of nodes, and short average cable length on a floor. In the proposed expansion method, if the switch degree and the number of switches per cabinet are carefully selected, each newly added cabinet can be installed only by connecting some short cables to the switches stored in a single existing cabinet.

The incremental expandability is not frequently considered in recent design of interconnection networks. Other issues including fault tolerance and routing updates still remain when the interconnection network is expanded. To mitigate such issues, however, we can apply existing research outputs, e.g., dynamic or static network reconfiguration methods that update routing tables and topology-agnostic deadlock-free routing [4], which are taken from fault-tolerant or power-aware network studies. Therefore, in this work we focus on the expandable topology and its...
floorplan design. The contribution of this work is as follows.

- We propose a new low-degree topology design, named DSN-F (distributed shortcut network with flexible expansion), which can be easily expanded up to twice the number of nodes. Our incremental expansion method maintains the switch degree and keeps the majority of inter-cabinet cables untouched.
- Graph analysis shows that the diameter and the average shortest path length of DSN-F increases smoothly as the number of nodes increases. The bisection bandwidth of DSN-F is in between that of Torus and that of a random topology.
- Layout analysis shows that the average cable length and the total cost of DSN-F are close to that of the same-degree Torus.
- Discrete-event simulations show that DSN-F has a similar performance tendency to a fully random topology. It could outperform Torus topology up to 70% on average.

The rest of this paper is organized as follows. The related work and background are described in Sect. 2. In Sect. 3 we introduce DSN-F, and present its incremental expansion method. In Sect. 4, we compare DSN-F to existing non-random and random topologies, in terms of path hops, and network bisection. The cable length, total cost, and power consumption of compared topologies when laid out in a machine room are considered in Sect. 5. In Sect. 6, we use discrete-event simulations to compare them. Section 7 concludes our work.

2. Related Work

2.1 Low-Degree Topologies

Topology design has been excitingly discussed for low-radix (e.g., 3-D Torus) vs. high-radix (e.g., Dragonfly metatopology) networks, especially for exascale computing systems. However, a low-radix network, which is our target in this work, is historically used in a large number of supercomputers as discussed in our prior work [5], because of (1) their simple management mechanisms for faults, (2) easy integration of network router/network interface and processors to a single chip or to a board (it can be regarded as a “switchless” network), (3) straightforward layout of switches with relatively short cables in a machine room [6], and (4) easiness in debugging custom communication protocol. There are a large number of low-degree topologies that have good diameter-and-degree properties, such as De Bruijin graph. Since De Bruijin graph requires \( n^k \) nodes for their structure, its expandability is limited.

Our prior works attempted to design an empirical best topology for arbitrary network sizes in terms of low diameter, low average shortest path length and short aggregate cable length in a machine room (e.g., random swapping for high-radix era [7] and distributed shortcut network (DSN) for low-radix era [5]). In this work we use DSN as a baseline for our proposed expandable topology that does not increase switch degree and the number of newly introduced long cables when a network is incrementally expanded.

Below we review the base topology, DSN-x with \( n \) nodes, which is also called DSN for short when the context is clear. The integer parameter \( x \), conditioned to be between 1 and \( p - 1 \) with \( p = \lceil \log n \rceil \), represents the size of the set of the different-length shortcuts.

- **Ring Formation**: \( n \) vertices are arranged in a ring and each node has an ID number from 0 to \( n - 1 \). Each node \( i \) shares two local undirected links with adjacent neighbors \( (i - 1) \mod n \) and \( (i + 1) \mod n \), which are called predecessor (pred for short) and successor (succ for short) links, respectively.
- **Labeling**: Each node also has a numeric label from 1 to \( p \), which is called the level of this node. The levels are assigned to nodes periodically: level \( i = \frac{1}{p} \ldots p \) is assigned to nodes \( k \times p + i - 1 \) where \( k = 0, 1, 2, \ldots, \lceil n/p \rceil \).
- **Shortcut Addition**: Each node that has level \( l \leq x \) has one shortcut link going to node \( j \) that has level \( l + 1 \) and has the minimum clockwise distance to \( i \) but at least \( \lceil n/2^l \rceil \). We call this type of shortcut as level-\( l \) shortcut, which has length at least \( n/2^l \). For a node with level \( l \) we say that it has height \( p + 1 - l \). Thus, the higher a node is, the farther its shortcut goes.

To have a simple view of DSN topology, imagine each group of \( p \) adjacent nodes to be collapsed into one big supernode. Then DSN-x topology could be viewed as a DLN-x topology† of these supernodes.

2.2 Network Expansion

Incremental expandability is commonly required to a commercial HPC and datacenter systems. Low-radix non-random networks, such as 2-D or 3-D tori, can be incrementally expanded in a straightforward manner. For example, \( k \)-ary 2-mesh can be expanded by each \( k \)-node increase with the same custom routing algorithm, e.g., Duta’s protocol or dimension-order routing. In this case the topology is still a two-dimensional mesh. Its short-cable layout in a machine room is also obvious.

By contrast, small-world and random topologies are easy to add nodes while maintaining low diameter and low average shortest path length, but introduce difficulties in achieving a short cable length and a constant switch degree. Koibuchi et al introduced random shortcut network [8] aimed to these properties but did not discuss the expandability. The JellyFish work [2] discussed that total random topologies have high incremental expandability. However the switch degree may increase and the newly introduced long cables may arise when adding nodes to an existing random topology.

†DLN-x, Distributed Loop Network of degree \( x \) [8], consists of \( n \) vertices arranged in a ring and additional shortcuts between vertices \( i \) and \( j \) such that \( j = i + \lceil n/2^k \rceil \mod n \) for \( k = 1, \ldots, x - 2 \).
To our best knowledge, in this context we do not have an efficient method to incrementally add nodes to small-world and random topologies. This is our main challenge in this study.

3. Distributed Shortcut Network with Flexible Expansion (DSN-F)

In the sections below we present our new topology design, named DSN-F, with some refinements to our precedent design, called DSN [5], to make it easier to expand and still maintain low degree and logarithmic diameter properties. We also propose an expansion method that smoothly adds nodes into the new topology. With this method, our new design has arbitrary size and incremental expandability properties.

3.1 Basic Approach

We consider the expandability properties of DSN topology. A simple expansion method is to add nodes sequentially into the ring of nodes (between nodes 0 and n − 1). Since a basic DSN topology can be viewed as a DLN-x topology of supernodes [5], this method is the same as adding one or more new supernodes and their shortcuts in DLN-x topology. On the other hand, Sect. V-C of [5] raises an idea on loosen the strict condition in constructing a DSN topology. On the other hand, Sect. V-C of [5] raises an idea on loosen the strict condition in constructing a DSN topology.

Hereafter we refer to a node as \([i, k, s]\). There are two types of links inside a supernode:

- Nodes in the same layer are arranged in a ring. Each node \([l, k, s]\) has two links, called Local_Pred and Local_Succ, which are connected to nodes \((l − 1 \mod p), k, s\) and \((l + 1 \mod p), k, s\), respectively.
- Each node \([l, k, s]\) with \(k \geq 1\) has another link, called Layer_Link, which is connected to the node \([l, k − 1, s]\), i.e., the same-level node in the upper layer.

Supernodes are arranged in a ring. In each supernode \(s\), the node \((p, 0, s)\) is connected to the node \((1, 0, (s + 1 \mod 2^p))\) by Succ link, while the node \((1, 0, s)\) is connected to the node \((p, 0, (s − 1 \mod 2^p))\) by Pred link.

In each supernode \(s\), each node with level \(l < p\) has one shortcut link going to another node \(j\) that has level \(l + 1\) in supernode \(u\) with the minimum clockwise distance of \(2^{p-l}\) to \(s\). This kind of links is called Level-l outgoing Shortcut of supernode \(s\) or level-[\(l-1\)] incoming Shortcut of the destination supernode \(u\). Note that only the nodes in layer-0 have shortcuts.

Figure 1 illustrates our topology construction in detail. Figure 1 (a) presents a full network for the case of \(n = 32\) and \(p = 5\). In this case, the topology is a ring of \(2^5 = 32\) supernodes. Each supernode is constructed of two layers. The nodes 0 to 23 are arranged in layer-0 and the rest are in layer-1. Each node with level \(l < 3\) has one Shortcut link. The node 0 in the supernode 0 has a shortcut to the node 13 in the supernode 4 with the clockwise distance \(\Delta = 4\). We denote the distance between the two supernodes by \(\Delta_{ij} = s_j − s_i\). By definition, this shortcut goes a distance that is greater than or equal to \(2^{p-1} = 2^{3-1}\). The Succ and Pred

is still a DSN, we can use the former method to add more nodes. This idea gains the incremental expandability properties for our design.

3.2 Topology Description for DSN-F: New Design for Expansion

Let us describe our new topology design in detail. Hereafter \(n\) denotes the total number of nodes. Each node has a node ID \(i\) with \(0 \leq i \leq (n − 1)\), which is determined by three numbers, namely \(l, k,\) and \(s\):

- **Node level** \(l\) with \(1 \leq l \leq p\), where an integer \(p\) with \(p \times 2^p \leq n < (p + 1) \times 2^{(p+1)}\) denotes the maximum level of all the nodes. The level of the node \(i\) is \(l = i\) mod \(p + 1\).
- **Node layer** \(k\) with \(0 \leq k \leq \lceil n/N \rceil\), where \(N = p \times 2^p\) denotes the maximum number of nodes in each layer (where each supernode has \(p\) nodes). We say the node \(i\) is in layer-\(k\) if \([i/N] = k\).
- **Supernode ID** \(s\) with \(0 \leq s \leq 2^p − 1\). Nodes are grouped into \(2^p\) supernodes identified by the supernode ID \(s\). A supernode \(s\) is a group of nodes \(i\) with \(i/p \mod 2^p = s\). Since \(0 \leq i \leq n − 1\) and \(n \geq p \times 2^p\), each supernode has at least \(p\) adjacent nodes.

Let us describe our new topology design in detail. Hereafter \(n\) denotes the total number of nodes. Each node has a node ID \(i\) with \(0 \leq i \leq (n − 1)\), which is determined by three numbers, namely \(l, k,\) and \(s\):

- **Node level** \(l\) with \(1 \leq l \leq p\), where an integer \(p\) with \(p \times 2^p \leq n < (p + 1) \times 2^{(p+1)}\) denotes the maximum level of all the nodes. The level of the node \(i\) is \(l = i\) mod \(p + 1\).
- **Node layer** \(k\) with \(0 \leq k \leq \lceil n/N \rceil\), where \(N = p \times 2^p\) denotes the maximum number of nodes in each layer (where each supernode has \(p\) nodes). We say the node \(i\) is in layer-\(k\) if \([i/N] = k\).
- **Supernode ID** \(s\) with \(0 \leq s \leq 2^p − 1\). Nodes are grouped into \(2^p\) supernodes identified by the supernode ID \(s\). A supernode \(s\) is a group of nodes \(i\) with \(i/p \mod 2^p = s\). Since \(0 \leq i \leq n − 1\) and \(n \geq p \times 2^p\), each supernode has at least \(p\) adjacent nodes.

Hereafter we refer to a node as \([i, k, s]\).

There are two types of links inside a supernode:

- Nodes in the same layer are arranged in a ring. Each node \([l, k, s]\) has two links, called Local_Pred and Local_Succ, which are connected to nodes \((l − 1 \mod p), k, s\) and \((l + 1 \mod p), k, s\), respectively.
- Each node \([l, k, s]\) with \(k \geq 1\) has another link, called Layer_Link, which is connected to the node \([l, k − 1, s]\), i.e., the same-level node in the upper layer.

Supernodes are arranged in a ring. In each supernode \(s\), the node \((p, 0, s)\) is connected to the node \((1, 0, (s + 1 \mod 2^p))\) by Succ link, while the node \((1, 0, s)\) is connected to the node \((p, 0, (s − 1 \mod 2^p))\) by Pred link.

In each supernode \(s\), each node with level \(l < p\) has one shortcut link going to another node \(j\) that has level \(l + 1\) in supernode \(u\) with the minimum clockwise distance of \(2^{p-l}\) to \(s\). This kind of links is called Level-l outgoing Shortcut of supernode \(s\) or level-[\(l-1\)] incoming Shortcut of the destination supernode \(u\). Note that only the nodes in layer-0 have shortcuts.

Figure 1 illustrates our topology construction in detail. Figure 1 (a) presents a full network for the case of \(n = 32\) and \(p = 5\). In this case, the topology is a ring of \(2^5 = 32\) supernodes. Each supernode is constructed of two layers. The nodes 0 to 23 are arranged in layer-0 and the rest are in layer-1. Each node with level \(l < 3\) has one Shortcut link. The node 0 in the supernode 0 has a shortcut to the node 13 in the supernode 4 with the clockwise distance \(\Delta = 4\). We denote the distance between the two supernodes by \(\Delta_{ij} = s_j − s_i\). By definition, this shortcut goes a distance that is greater than or equal to \(2^{p-1} = 2^{3-1}\). The Succ and Pred
3.3 Incremental Expansion Method

The following section provides an insight into its arbitrary size and incremental expandability properties.

The size of a shortcut set \( p \) is an important parameter for labeling and topology construction. For any number of nodes \( n \), we can always find a corresponding integer \( p \). Therefore, we can say that DSN-F has an arbitrary size property. Moreover, the inside structure of supernodes makes it easy to add new nodes while maintaining the advantages on degree and diameter.

As an example, consider adding one node to an \( n \)-node DSN-F topology with \( p \) levels and \( K \) layers. Generally, we add a new node into the layer-(\( K - 1 \)) which has \( r = n \mod N \) nodes in it (\( r < p \)). If the layer-(\( K - 1 \)) has full \( p \) nodes, we will firstly create a new layer-\( K \), and then add the new node to it. In both cases, since nodes are added into a supernode based on the labeling scheme of the construction method, the structure inside the supernode is maintained. Hence, the maximum/average degree does not change, and the diameter increases at most 1 hop (for the new layer) when compared to the original topology. From the cabling point of view, this method has an advantage that it avoids rewiring, i.e., it only adds new cables for the new node.

Additionally, in case of the number of additional nodes grows high, we propose a method to transform the structure of supernodes from DLN-\( p \) to DLN-(\( p + 1 \)), with an acceptable number of cables rewired. The main idea is (i) we add nodes to the topology using the above-mentioned method; and (ii) whenever the numbers of nodes inside all the supernodes reach \( 2p + 2 \), we firstly split each supernode into two new smaller ones with size of \( p + 1 \), then add shortcuts to the new supernodes to ensure that each of them has a full set of \( p + 1 \) shortcuts. As a result, the topology produced by the transformation method is still DSN-F but its argument integer \( p \) changes to \( p + 1 \). We say that DSN-F has an incremental expandability property since we can continue adding more nodes into the new topology by applying the transformation method repeatedly.

In the rest of this section we provide a detailed description of our transformation method. Let us consider a transformation of DSN-F topology from argument \( p \) to \( p + 1 \). Before transforming, the topology is a ring of \( 2^p \) supernodes, each with a full set of \( p \) shortcuts. Inside each supernode, \( 2p + 2 \) nodes are arranged into 3 layers. Without loss of generality, we use ID numbers from 0 to \( 2^p + 1 \) to identify those nodes, as shown in Fig. 2 (a). After transforming, a supernode \( S \) is split into two smaller supernodes \( S_A \) and \( S_B \). Each of them has \( p + 1 \) nodes, and is constructed of only one layer. The supernode \( S_A \) is a combination of \( p \) layer-0 nodes and one layer-3 node, i.e., node 2p at layer-3, level-1. Clearly, now this node can be considered as a level-(\( p + 1 \)) node of \( S_A \). Similarly, the rest of the nodes are pushed to the supernode \( S_B \). In the view of links, we remove/add links following the steps below:

- All the Layer Links (\( p \) links between layer-0 and layer-0 nodes) are removed.
- New internal links are added in the new supernodes from layer-1 to layer-0 and layer-3.
- New Layer Links are added between the new supernodes.
- New Succ Links are connected to the new supernodes from DLN-(\( p + 1 \)).
1, and two links between layer-1 and layer-2) are removed.
• In each supernode, most of the local links are not affected. We add the node level \((p + 1)\) into the ring of layer-0 in the position between the node level 1 and the node level \(p\). This action removes one link and adds two links inside each supernode.

In the supernode \(S_A\):
- Keep all the shortcuts (both incoming and outgoing) from node level 1 to nodes level \(p - 1\).
- Remove the Succ link of the node level \(p\) and then add new outgoing shortcuts from it to the node level \((p + 1)\) in the supernode \((S + 1)\).
- Add a new Succ link from the node level \(p + 1\) of \(S_A\) to the node level 1 of \(S_B\).

In the supernode \(S_B\):
- Add \(p\) outgoing shortcuts. From the node level \(l \leq p\) of \(S_B\), add one shortcut to the node level \(l + 1\) in the supernode \((S + p + 1)\).
- Add a new Succ link from the node level \(p + 1\) of \(S_B\) to the node level 1 of \((S + 1)\).

Figure 2 illustrates our transformation method. Figure 2 (a) shows a supernode \(S\) before splitting, and Fig. 2 (b) presents two smaller supernodes after splitting. Clearly, the length of a typical shortcut from the node \(i\) to \(j\) is \(2^{(p+1)-l}\). Therefore, the maximum degree of the network is 5. Most of the nodes in layer-1 or higher has a minimum degree 3, namely Local_Pred, Local_Succ, and Layer_Link. In fact, for any node with degree 3, we can find a corresponding layer-0 node with degree 5, which is in the same supernode and the same level. In other words, the number of nodes with degree 3 is equal to the number of nodes with degree 5. Thus, the average degree is 4.

4. Graph Analysis

In this section, we analyze the graph properties of the DSN-F topology generated by the incremental network expansion. We compare DSN-F with some typical topologies that have the same average degree, namely Torus, RR-4 \([8]\), and DSN \([5]\). First, we compare them in terms of diameter and average shortest path length, which affect the network latency. Next, we compute the network bisection in order to look insight into the network throughput.

4.1 Graph Properties of DSN-F

We consider the worst case of DSN-F in terms of graph properties by the theorem below. Remind that we arrange an \(n\)-node topology into \(2^p\) supernodes. Each supernode is constructed of some layers. The number of layers in the topology is denoted by \(K = \lceil n/N \rceil\) where \(N = p \times 2^p\).

**Theorem 1.** On the properties of DSN-F:

a. The average degree of vertices is 4, and the maximum degree is 5.

b. The diameter of topology is at most \(3p/2 + 2K - 2\) and is logarithmic.

c. The bisection bandwidth of topology is \(N\).p.

**Proof of Theorem 1.**

a. By definition, a node can have four types of links, namely Pred/Succ, Shortcut, Local_Pred/Local_Succ, and Layer_Link. Figure 1(b) illustrates the links. Clearly, the maximum degree of the layer-0 nodes is 5, while it is 4 for the nodes in layer-1 or higher. Therefore, the maximum degree of the network is 5. Most of the nodes in layer-1 or higher has a minimum degree 3, namely Local_Pred, Local_Succ, and Layer_Link. In fact, for any node with degree 3, we can find a corresponding layer-0 node with degree 5, which is in the same supernode and the same level. In other words, the number of nodes with degree 3 is equal to the number of nodes with degree 5. Thus, the average degree is 4.

b. We estimated the upper bound of hop counts when using a simple custom routing. Consider the routing task from the node \(i\) with \([l_i, k_i, s_i]\) to the node \(j\) with \([l_j, k_j, s_j]\). We assume that \(0 \leq s_i < s_j\) without loss of generality. A simple routing algorithm proceeds in three phases:

- **PRE-WORK** phase: route a packet to the node \([l_i, 0, s_i]\), which is in layer-0 in the same supernode and the same level with the source node \(i\).
- **MAIN-PROCESS** phase: route the packet to the node \([l_i, 0, s_j]\) through two small steps. Firstly, the packet reaches the node \([l_j, 0, s_j]\). Secondly, it goes through properly-ordered set of links, which can be determined based on the distance from the source supernode \(s_i\) to the destination supernode \(s_j\), i.e., \(\Delta_{ij} = s_j - s_i\). This distance could be represented by a binary string \(b_1b_2...b_{l-1}\) where \(k = 1...p\). The link set includes level-k shortcut if \(b_k = 1\), and Local_Succ link if \(b_k = 0\). The order of links starts from \(b_1\) and ends with \(b_{l-1}\) in the direction of increasing node level.
- **FINISH** phase: route the packet to the destination node \(j\) following the Layer_Links inside the destination supernode.

In the PRE-WORK phase, we route a packet from \(i\) to \(j\) in the corresponding layer-0 node using Layer_Links. It takes at most \(K - 1\) hops. In the MAIN-PROCESS phase, we first route the packet to the node \([l_i, 0, s_j]\) by successively taking Local_Succ or Local_Pred links. This action needs at most \(p/2\) hops (along the local ring of layer-0). Then the MAIN-PROCESS phase takes at most \(p\) hops to move the packet from the source supernode to the destination supernode, because the packet always goes up in terms of the level. In conclusion, the MAIN-PROCESS phase needs at most \(p_2 = 3p/2\) hops. In the FINISH phase, the packet is routed...
to the destination node \( j \) using Layer Link. The packet is at a node in the same supernode and the same level with \( j \) when the MAIN-PROCESS phase finishes. Therefore, the maximum path length for this action is at most \( K - 1 \) hops. Overall, using our DSN-F routing algorithm, the path from any source node \( i \) to any destination node \( j \) takes at most \( 3p^2 + 2K - 2 \) hops.

It is possible to find a path shorter than ones found by the above routing algorithm. Hence, the maximum diameter of topology could be \( \frac{3p}{2} + 2K - 2 \). By definition, \( p \) is an integer number that satisfies \( p \times 2^p \leq n < (p + 1) \times 2^{p+1} \), and thus \( \log n \leq (p + 1) + \log p + 1 \). In other words, we can say \( p = \log n + \text{Constant} \), which means that the maximum routing diameter is logarithmic.

c. Consider a bisection cut which breaks the network into two approximately equal halves where the first half includes all the supernodes with even ID numbers, and the second half contains the rest. We now estimate the number of links crossing the cut. Since the level-\( x \) shortcuts from supernode \( i \) come to the supernode \( i + 2^x \) where \( 1 \leq x \leq p - 1 \), shortcuts from a supernode with an even ID number always go to other supernodes with an even ID number. There is no shortcut from supernodes with even ID numbers that crosses the cut. Similarly, shortcuts from supernodes with odd ID numbers do not cross the cut. Moreover, no Internal link but all the Succ/Pred links crosses the cut. Therefore, the total number of links crossing the cut is \( 2p^2 \), and the bisection is \( N/p \).

From this theorem, the degree-diameter factor of DSN-F is closely the same as the basic DSN topology. The average degree of DSN-F is a bit higher but the maximum degree is still 5. Thus, we can say that DSN-F is a low-degree topology. In terms of diameter, DSN-F is significantly better. Note that the diameter of DSN is about \( 2.5 \log n + r \) with \( r < \log n \), whereas that of DSN-F is \( \frac{3p}{2} + 2K - 2 \). By the definition of \( p \), we can prove that \( \frac{3p}{2} + 2K - 2 \) is approximately equal to \( \frac{3 \log n}{2} \). Therefore, the diameter of DSN-F is about a half of the basic DSN. This value is equal to the improved diameter version mentioned in Sect. V-B of [5].

4.2 Diameter and Average Shortest Path Length vs. Number of Added Nodes

Figure 3 shows the diameter and the average shortest path length of the 1,024-node DSN-F topology and its expanded variations. The x axis indicates the number of added nodes, i.e., \( x = 0 \) means the baseline 1,024-node network and \( x = 1024 \) means the expanded 2,048-node network. Not surprisingly, the diameter and the average shortest path length slightly increase as the number of nodes increases. When the number of added nodes is greater than 768, i.e., \( x = 896 \), one more layer is added into each supernode following our expansion method (3 layers per supernode). Therefore, the diameter at \( x > 768 \) increases 1 hop compared to the diameter at \( x = 768 \). When 1,024 nodes are added, i.e., the network grows twice as large as the baseline, supernodes are split into smaller ones which have only 1 layer. Due to this regularity, the diameter at \( x = 1024 \) is slightly lower than at \( x = 768 \).

4.3 Diameter and Average Shortest Path Length vs. Network Size

The average degree of DSN-F is 4. We compare it with the same-degree topologies with the same number of nodes, namely 2-D Torus, RR, and DSN. Figures 4 and 5 show the diameter and the average shortest path length of each topology. Lower values are considered better.
In all the network sizes, RR achieves the lowest while 2-D Torus leads to the highest. DSN-F maintains the main characteristics of DSN. Specifically, the diameter and the average shortest path length of DSN-F are at most 16.7% and 4.6% lower than those of DSN. Therefore, we expect that DSN-F leads to almost the same performance with DSN.

4.4 Network Bisection

An important characteristic of a network is its bisection. Consider a $p$-way partition of a graph, which means a graph partitioned into $p$ components each with nearly the same size. We use “minimum cut” to refer to the minimum number of edges that connect any two of those $p$ components. A network bisection in a common sense is the minimum cut computed with $p = 2$. For generality we compute cuts for all $p$-way partitions with $2 \leq p \leq 6$. We perform this computation using METIS, which uses an efficient multi-level iterative approach [9].

Figure 6 presents the minimum cuts of the compared topologies with 1024 nodes. Higher values are considered better. DSN and DSN-F have similar minimum cuts except in the cases of $p = 2$ and $p = 4$. In these cases, almost all the shortcuts connect nodes in the same subgraph together. Hence, the cuts include only the Succ and Pred links. This result shows that the proof of Theorem 1.c is correct. In addition, the minimum cuts of DSN and DSN-F are wider than Torus and narrower than RR. These results indicate that DSN-F has good properties not only in terms of latency but also in terms of bisection (throughput).

5. Layout Analysis

5.1 Average Cable Length and Layout

We estimate the cable length required to deploy the topologies over a physical layout of cabinets. The parameters and the optimization method are the same as those in our previous work [7]. We assume a physical floor that is sufficiently large to align all cabinets on a 2-D grid. Formally, assuming $c$ cabinets, the number of cabinet rows is $x = \lceil \sqrt{c} \rceil$ and the number of cabinets per row is $y = \lceil c/x \rceil$. We assume that each cabinet is 0.6m wide and 2.1m deep including space for the aisle, following the recommendations in [10]. The distance between the cabinets is computed using the Manhattan distance. We estimate average cable length in a more conservative way than in [11]: 2m intra-cabinet cables and a 2m wiring overhead added to the length of inter-cabinet cables at each cabinet. We ignore cables between compute nodes and switches, since their lengths are constant. We assume that each cabinet stores 16 switches.

The average cable length of each topology† is computed and shown in Fig. 7. Lower values are considered better. Our DSN-F topology features an average cable length similar to DSN and 2-D Torus topologies in most of the network sizes (except for $2^{10}$ and $2^{11}$). It reduces the average cable length by up to 17.72% and 24.82% where the network size is $2^7$ and $2^{12}$, respectively. This improvement illustrates the best cases where we carefully select the number of switches per cabinet. In this case, all the switches in the same layers are in the same cabinet and all the local links in logical design are installed inside only one cabinet. As the network size become larger, the cable length of DSN-F with 16 switches per cabinets becomes closer to the best case, where one supernode is mapped to exactly one cabinet.

5.2 Network Cost and Power Consumption

In this section we provide the cost comparison between networks with DSN-F and other topologies. We also estimate the cost per node of DSN-F network as the number of nodes increases. In additional, we measure the power consumption of those networks those networks deployed into the physical layout of cabinets that we used to estimate the cable length in the previous section.

We use the cost modal mentioned in [12], which con-

†Notice that the layout of 2-D Torus is well studied, e.g., folded method for uniform link length. However, the average cable length of folded Torus is the same as that of the corresponding original Torus in which only the wraparound links are long. Thus we fairly compare 2-D Torus, DSN and DSN-F in terms of cable length.
undersiders both the cost of switches and the cost of interconnection cables. In this model, the switch cost is a linear function of radix, while the costs of both electrical and optical cables depend not only on cable length but also on link bandwidth. We set the link bandwidth to 40 Gbps.

Figure 8 plots the cost per node vs. the number of nodes $N$ of networks with DSN-F, DSN, 2-D Torus, and RR topologies. The cost of 2-D Torus is always the smallest and the cost of RR is the highest. DSN and DSN-F have almost the same cost except for $N = 2^7$ and $N = 2^{12}$, where DSN-F is the best. This observation shows that the result in Fig. 8 is similar to Fig. 7. Spare ports are generally needed for network expansion purposes. We thus evaluate the cost per node of network with DSN-F topology for the case with 25, 50, and 100% expandability support by preparing the number of the corresponding spare ports. The results show that the cost increases by up to 13, 27, and 54%, respectively. These are expected results because the switch cost which is almost proportional to the number of ports dominates the overall network cost.

The total power consumption of the network depends on the total number of switches and the numbers of ports per switch (or radix of a switch). Following [12], we assume that a switch port has 4 lanes, each lane consumes $\approx 0.7$ watts, and each port consumes $4 \times 0.7 = 2.8$ watts. Because all of the four topologies we use in this experiment have the same average radix, i.e., 4, the total power consumption of these topologies is exactly the same and is a linear function of the number of nodes. We thus omit the plot.

6. Parallel Application Performance

6.1 Discrete-Event Simulation

We use SimGrid as a parallel-computer simulator [13]. The network size is set to 64 or 256 switches. Four processes are assigned to hosts attached to a single switch. The network link bandwidth is set to 40 Gbps, switch delay is set to 200 nsec, and the processor power is set to 100 GFLOPS. We take a minimal routing for all the topologies. We simulate the execution of the NAS Parallel Benchmarks (version 3.3.1, MPI versions) [14] (Class B for BT, CG, DT, LU, and SP, and Class A for FT benchmarks).

6.2 Simulation Results

Figure 9 plots the results for DSN-F, DSN, 2-D Torus, and RR topologies. The $y$ axis indicates the Mop/s for NAS Parallel Benchmarks relative to 2-D Torus. Higher values are considered better. As expected, as the network size becomes large, the performance improvement of DSN-F, DSN and RR topologies over 2-D Torus becomes larger. For example, DSN-F has a similar performance tendency with RR. It outperforms Torus by 23% and 70% on average, for 64 and 256 switches, respectively. Although these benchmarks have each different traffic patterns, no empirical performance tendency of each topology are found. There is no universal topology for a variety of parallel applications.

7. Conclusions

In this study we proposed an expandable low-degree topology, named DSN-F, for supercomputers and datacenter networks where the number of nodes gradually increases year by year. Since their interconnection networks recently become latency-sensitive to support various massively parallel applications [15], the network topology that exploits
small-world effect, e.g., DSN [5], is attractive for those systems. Unlike conventional “regular” topologies (e.g., k-ary n-cubes), those “small-world” topologies have no intuitive way to add nodes once the topology is deployed. In this context we extended the precedent DSN topology design so as to easily be expanded while maintaining the switch degree and keeping the majority of cables untouched. We theoretically illustrated that a cabinet or a node set can be newly inserted by connecting some short cables to a single existing cabinet. We evaluated DSN-F in comparison with the same-degree Torus, RR, and DSN topologies in terms of diameter, average shortest path length, bisection, cable length, and parallel application performance. Our evaluation results demonstrated that DSN-F has similar diameter, average shortest path length, and average cable length to those of DSN and RR. The cable length is close to that of 2-D Torus, whereas the diameter and the average shortest path length are close to those of RR. The discrete-event simulation showed that there is no universal topology for bandwidth-sensitive parallel applications. From various practical quantitative aspects we conclude that DSN-F is a promising alternative to make future supercomputers and datacenter networks flexibly expandable.

Acknowledgments

This work was partially supported by JST CREST and KAKENHI #25280018. We thank Dr. Fabien Chaix, Institute of Computer Science, Foundation for Research and Technology - Hellas, Greece, for his assistance to use the parallel-computer simulator SimGrid.

References


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

Khanh-Van Nguyen is a Senior Lecturer at the School of Information and Communication Technologies (SoICT) of Hanoi University of Science and Technology (HUST), where he leads the Software Engineering and Distributed Computing Lab (SEDIC Lab). He received his Ph.D. degree in Computer Science from the University of California, Davis (2006) and also spent one post-doctoral year (2007–08) as a CNRS researcher at University Paris 7 & 11. He was head of the Department of Software Engineering (of SoICT) from 2008 to 2012. His main research domain is algorithms and theoretical models for computer networks and distributed computing.

Ikki Fujiwara received the BE and ME degrees from Tokyo Institute of Technology, Tokyo, Japan in 2002 and 2004, respectively, and received the PhD degree from the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan, in 2012. He is currently a Project Associate 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 IPSJ, IEICE and IEEE.
Michihiro Koibuchi received the BE, ME, and PhD degrees from Keio University, Yokohama, Japan, in 2000, 2002 and 2003, respectively. Currently, he is an associate professor in the Information Systems Architecture Research Division, National Institute of Informatics and the Graduate University of Advanced Studies, Tokyo, Japan. His research interests include the area of high-performance computing and interconnection networks. He is a member of the IEEE and IPSJ and a senior member of IEICE.