Layout-aware Expandable Low-degree Topology

Nguyen T. Truong, Van K. Nguyen, Nhat T. X. Le
Ha Noi University of Science and Technology
1 Dai Co Viet Road, Ha Noi, Viet Nam
nguyen.88.tt@gmail.com, vannk@soict.hut.edu.vn,
thongnhat313@gmail.com

Ikki Fujiwara, Fabien Chaix, Michihiro Koibuchi
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, Japan
{ikki, fabien_chaix, koibuchi}@nii.ac.jp

***Abstract*—System expandability becomes a major concern for highly-parallel computers and datacenters, because their number of nodes gradually increases year by year. In this context we propose a low-degree expandable 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, low average shortest path length and short aggregate 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. Flit-level network simulation results show that the proposed topology has lower latency for three synthetic traffic patterns as expected from graph analysis. Our event-driven network simulation results show that the proposed topology provides a comparable performance to 2-D torus even for bandwidth-sensitive parallel applications.**

***Index Terms*—Network expandability, network topologies, small-world networks, interconnection networks, high-performance computing.**

I. Introduction

A large number of datacenters and supercomputers are incrementally expanded 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, e.g. FLOPS (floating operation per second), not only by tuning software/applications but also by increasing the number of processors [1] after a supercomputer is initially deployed. In addition, a large number of commercial datacenters have also been expanded incrementally.

When increasing the number of nodes, a main design concern is to minimize (i) the number of rewired cables, (ii) the aggregate length of newly-introduced cables, and (iii) the degradation of the network efficiency. The network efficiency affects the execution time of communication-sensitive parallel applications, because it may corresponds to the diameter and the average shortest path length when their traffic patterns are unpredictable or dynamic depending on input parameters.

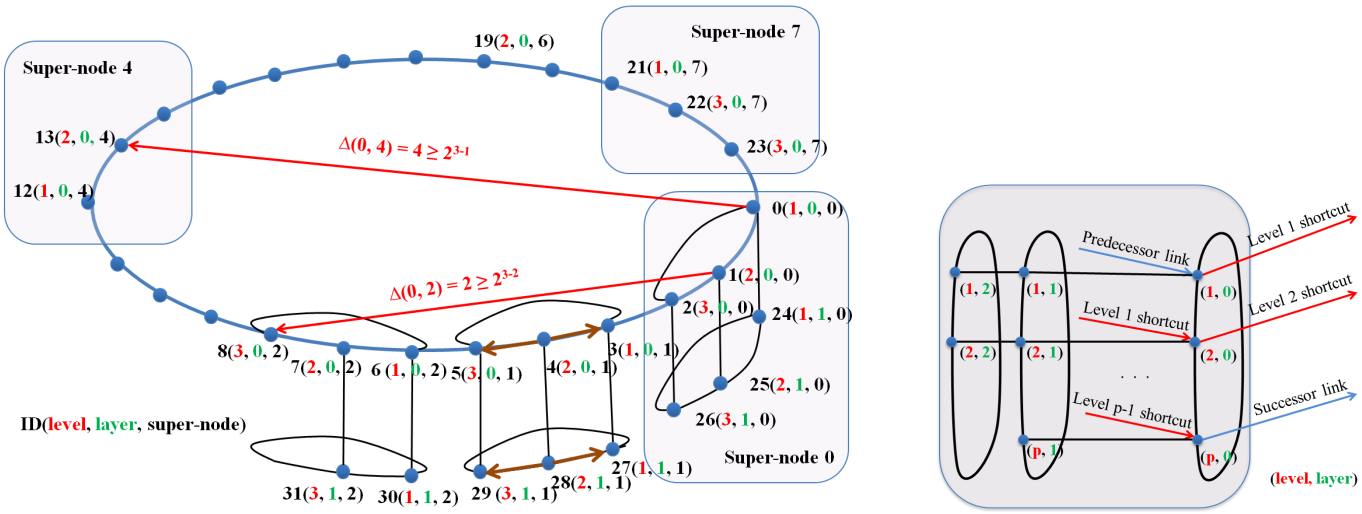
In this context we propose a layout-aware expandable topology and its efficient method to incrementally add nodes or cabinets in a machine room. In the topology design we use the small-world effect that does not rely on randomness for low path lengths, efficient incremental expansion of nodes, and short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

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 [2], 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 easily be 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 gracefully as the number of nodes increases. The average cable length of DSN-F is close to that of the same-degree torus.
- Flit-level simulations show that DSN-F outperforms the same-degree torus in terms of network latency for synthetic traffics, while event-driven simulations show that all the same-degree topologies have similar average performances for bandwidth-sensitive parallel applications.

The rest of this paper is organized as follows. The related work and background are described in Section II. In Section III we introduce DSN-F and present an incremental expansion method for it. In Section IV, we compare DSN-F to existing non-random and random topologies, based on the cable length when laid out in a machine room. In Sections V and VI, we use flit-level and discrete-event simulations to compare them. Section VII concludes our work.



(a) Topology DSN-F in size of 32

(b) Structure inside a super-node

Figure 1. The DSN-F topology. Red lines are Shortcuts. Black lines are Internal Links. Brown lines are Local_Pred and Local_Succ links. Blue lines are Pred and Succ links.

II. Related Work

A. 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 [3], 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 [4], 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 Bruijn graph. Since De Bruijn 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 [5] and distributed shortcut network (DSN) for low-radix era [3]). In this work we use DSN as baseline for our proposed expandable topology that does not increase switch degree and the number of newly introduced long cables when the network is incrementally expanded. Below, we review the basic 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 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 also called the level of this node. The levels are assigned to nodes periodically: level $i = \overline{1..p}$ is assigned to nodes $k \times p + i - 1$ where $k = 0, 1, 2, \ldots, \lfloor n/p \rfloor$.
- **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 $\lfloor n/2^l \rfloor$. We also call this type of shortcut as the level-l shortcut, which has length at least $n/2^l$. For a node with level l we also say that it has height $p+1-l$. Thus the higher a node is the farther its shortcut goes.

B. 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. Duato'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 short cable length and a constant switch degree. The JellyFish work [6] discussed that random topology has 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.

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.

III. 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 [3], to make it easier to expand and still maintain the low degree and the 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.

A. Basic Approach

Our new design is followed by a few observations.

Firstly, let us discuss about the incomplete super-node in a basic DSN-x topology of n nodes. A super-node I is incomplete if it does not have a complete set of x shortcut links (its number of nodes is less than $p = \lceil \log n \rceil$). Let $r = n \mod p$. The existence of incomplete super-node raises the overshoot issue during routing process. It can lengthen the walk for finding the next k-level node, which I may not have, by the extra r local links inside I. Therefore, the maximum path length from any source node to any destination node is up to $3p + r$ (mentioned in the facts 2 and 3 in [3]). In a simple view, the maximum path length will be shorter if we can avoid the incomplete super-node in terms of designing.

Secondly, 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[1] of super-nodes [3], this method is the same as adding one or more new super-nodes and their shortcuts in DLN-x topology. On the other hand, Section V-C of [3] raises an idea on pushing new nodes into existing super-nodes. The additional nodes only have local links to existing nodes and do not come with shortcuts. In both ways, the expansion methods work well with a small number of additional nodes. But in case of adding many nodes, e.g. when doubling the size of network, it will loosen the small-world effect in designing shortcuts. Therefore, the average shortest path length and the diameter of topology become large as the number of additional nodes grows. Generally, expansion method needs to produce the same type of topology for maintaining the small-world effect.

Following the above observations, we learn the idea of loosening the strict condition in constructing topology by allowing each super-node to have a flexible size [3]. We also standardize the number of super-nodes to be 2^p. Now, topology is a ring of 2^p complete super-nodes that include at least p nodes and p outgoing shortcuts. This design matches exactly to the best case of DLN-p topology. These refinements help to reduce the maximum path length by avoiding the incomplete super-node issue.

Inside a super-node, we arrange nodes into layers named layer-$\{0, 1, 2, \ldots\}$. Layer-0 includes p different-level nodes that come with shortcuts. Each layer also has at most p nodes without shortcuts. With this inside structure of a super-node, we can easily add nodes to an existing topology without rewiring the shortcuts (long and complicated cables) by pushing nodes into the layer of super-nodes. Furthermore, if the number of additional nodes grows high, we propose a method to transform the DSN topology from the view of DLN-p to DLN-p+1, with an acceptable number of rewired cables. Since the topology after transformation is still a DSN, we can use the former method to add more nodes. This idea gains the incremental expandability properties for our design.

B. 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.

1) Labeling: 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 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 (p nodes in each super-node). We say the node i is in layer-k if $\lfloor i/N \rfloor = k$.
- **Super-node ID** s with $0 \leq s \leq 2^p - 1$. Nodes are grouped into 2^p super-nodes identified by the super-node ID s. A super-node 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 super-node has at least p adjacent nodes.

2) Internal Links of Super-node: There are two types of links inside a super-node.

- 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.

3) Succ and Pred Links: Super-nodes are also arranged in a ring. In each super-node 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.

4) Shortcuts: In each super-node s, each node with level $l < p$ has one Shortcut link going to another node j that has level $l + 1$ in the super-node with the minimum clockwise

[1] A DLN-x topology [7] is constructed of a ring and $x-2$ additional random shortcuts at each node.

distance of $2^p/2^l = 2^{p-l}$ to s. Note that only nodes in layer-0 have shortcuts.

Figure 1 illustrates our topology construction in detail. Fig.1(a) presents a full network for the case of $n = 32$ and $p = 3$. In this case, the topology is a ring of $2^3 = 8$ super-nodes. Each super-node 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 super-node 0 has a shortcut to the node 13 in the super-node 4 with the clockwise distance $\Delta = 4$. We denote the distance between the two super-nodes by $\Delta_{ij} = s_j - s_i$. By definition, this shortcut goes a distance that is greater than or equal to $2^{p-l} = 2^{3-1}$. The *Succ* and *Pred* links connect super-nodes together. Node $0\{l = 1, k = 0, s = 0\}$ and $23\{3, 0, 7\}$ are connected by the link that called *Pred* of node 0 or *Succ* of node 23. Fig.1(b) illustrates the structure inside a full super-node with $2p + 2$ nodes arranged into 3 layers. The node $0\{1, 0, 0\}$ has two links *Local_Pred* and *Local_Succ* which are connected to nodes $2\{3, 0, 0\}$ and $1\{2, 0, 0\}$. It also connects to the node $24\{1, 1, 0\}$ by *Layer_Link*.

Our DSN-F topology uses a custom routing algorithm presented in [3] with some small additional steps. 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. Our new routing algorithm proceeds in three steps:

1) Route to the node $\{l_i, 0, s_i\}$, which is in layer-0 in the same super-node and the same level with the source i.
2) Route to the node $\{l_j, 0, s_j\}$ using the algorithm mentioned in [3], which repeatedly finds a proper shortcut to go a half of distance from the source super-node s_i to the destination super-node s_j.
3) Reach the destination node j following the Layer_Links inside the destination super-node s_j.

The detailed algorithm is presented in Fig.2. For example, consider a path from node $24\{1, 1, 0\}$ to node $7\{2, 0, 2\}$ in the network represented in Fig.1(a). We need firstly route to node $0\{1, 0, 0\}$ that is in the same super-node and level with 24 but in layer-0. Then, we route to node $1\{2, 0, 0\}$ that has proper shortcut to go to destination super-node (PRE-WORK). We reach destination super-node 2 by using this shortcut (MAIN-PROCESS) and route to node $7\{2, 0, 2\}$ by *Local_Pred* link in FINISH. The routing path from node 24 to node 7 is (24, 0, 1, 8, 7).

DSN-F routing algorithm pseudo-code

```
1:  procedure DSN-F-ROUTING(i, j)
2:      u ← i
3:      l ← ⌊log 2^p/Δ_uj⌋ + 1
4:
5:  PRE-WORK PHASE ————————————————
6:      while k_u > 0 do                    ▷ k_u = layer of u
7:          u ← u.Layer_Link
8:      end while
9:      while l_u ≠ l do                    ▷ l_u = level of u
10:         u ← u.Local_Succ or u.Local_Pred
11:     end while
12: MAIN-PROCESS PHASE ———————————————
13:     repeat
14:         if l_u = p − 1 then
15:             u ← u.Succ
16:         else if l_u = l then
17:             u ← u.Shortcut
18:         else
19:             u ← u.Local_Succ
20:         end if
21:         l ← ⌊log 2^p/Δ_uj⌋ + 1
22:     until s_u = s_j
23: FINISH PHASE ————————————————————
24:     repeat
25:         u ← u.Local_Succ or u ← u.Local_Pred
26:     until l_u = l_j
27:     repeat
28:         u ← u.Layer_Link
29:     until u = j
30: end procedure
```

Figure 2. Our custom routing algorithm for DSN-F. The notation of "u.*Link*" indicates the adjacent node of node u connected by *Link*.

C. Graph Properties of DSN-F

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

Theorem 1: On the properties of DSN-F:

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

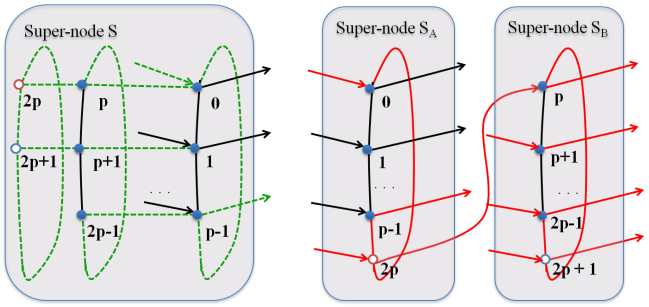
b. The maximum routing diameter is $2 \times (p + K - 1)$ and is logarithmic.

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. Fig.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.

Moreover, most of the nodes in layer-1 or higher has 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 super-node 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. Let us consider a path from the source node i to the destination node j. In the PRE-WORK phase, we first route from i to an corresponding layer-0 node using Layer_Links. It takes at most $K - 1$ hops. Then we route to a proper



(a) Before splitting (b) After splitting

Figure 3. Illustration of our transformation method. Red lines are additional links. Green lines are removed links.

node that is high enough to look down at the super-node s_j by successively taking Local_Succ or Local_Pred links. This action needs at most $\frac{p}{2}$ hops (along the local ring of layer-0). In conclusion, the PRE-WORK phase needs $\frac{p}{2} + K - 1$ hops. The MAIN-PROCESS phase also takes at most p hops because we always go down in terms of the height of the nodes (going up with the levels).

In the FINISH phase, we take Local_Succ link to reach the node $\{l_j, 0, s_j\}$ and route to the destination node j using Layer_Link. The fact is that we are at a node in the same super-node with j when the MAIN-PROCESS phase just finished. Therefore, the maximum path length for the former action in the FINISH phase is $\frac{p}{2}$. Clearly, the later action needs 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 $2 \times (p + K - 1)$ hops. 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 the other words, we can say $p = \log n + Constant$, which means that the maximum routing diameter is logarithmic. ■

From this theorem, the degree-diameter factor of DSN-F is nearly 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 routing diameter, DSN-F is significantly better. Note that the diameter of DSN is about $3 \log n + r$ with $r < \log n$, whereas that of DSN-F is $2 \times (p + K - 1)$. By the definition of p, we can proof that $(p+K-1)$ is approximately equal to $\log n$. Therefore, the routing diameter of DSN-F is about a half of the basic DSN. This value is equal to the improved diameter version mentioned in Section V-B of [3].

D. Incremental Expansion Method

Section III-C proves that our new design has low degree and logarithmic diameter properties. The following section provide 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 arbitrary size properties. Moreover, the inside structure of super-nodes 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 the 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 super-node based on the labeling scheme of construction method, the structure inside the super-node is maintained. Hence, the maximum/average degree does not change, and the diameter increase 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 add new cables for the new node.

Moreover, if the number of additional nodes grows high, we propose a method to transform the structure of super-nodes 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 super-nodes reach $2p + 2$, we firstly split each super-node into two new smaller ones with size of $p + 1$, then add shortcuts to the new super-nodes 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 incremental expandability properties 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 more 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 super-nodes, each with a full set of p shortcuts. Inside each super-node, $2p + 2$ nodes are arranged into 3 layers. Without loss of generality, we use ID number from 0 to $2p+1$ to identify those nodes, as shown in Fig.3(a). After transforming, a super-node S is spitted into two smaller super-nodes S_A and S_B. Each of them has $p + 1$ nodes, and is constructed of only one layer. The super-node 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 nodes are pushed to the super-node S_B. In the view of links, we remove/add links following the steps below:

- All the Layer_Links are removed (p links between layer-0 and layer-1, and two links between layer-1 and layer-2).
- In each super-node, 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 remove one link and add two links inside each super-node.

- In the super-node S_A:
  - Keep all the shortcuts (both incoming and outgoing) from nodes 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 super-node $(S+1)_A$.
  - Add a new Succ link from the node level $p+1$ of S_A to the node level 1 of S_B.
- In the super-node 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 super-node $(S+2^{p+1-l})_B$.
  - Add a new Succ link from the node level $p+1$ of S_B to the node level 1 of $(S+1)_A$.

Figure 3 illustrates our transformation method. Fig.3(a) shows a super-node S before splitting, and Fig.3(b) presents two smaller super-nodes after splitting. Clearly, the inside structure of the super-node S_A (or S_B) looks like DSN-F topology with the size of the shortcut set changed from p to $p+1$. On the other hand, the distance between S_A and $(S+1)_A$ is 2 because S_B is in the middle of them. Similarly, the distance between S_A and $(S+2^{p-l})_A$ is $2 \times 2^{p-l} = 2^{(p+1)-l}$. Therefore, the length of a typical shortcut from the node i with $\{l, 0, S_A\}$ to the node j with $\{l+1, 0, (S+2^{p-l})_A\}$ is as far as it is in the design of topology. The two points above prove that the topology after transforming is still an instance of DSN-F.

E. Practicality of DSN-F

Because of arbitrary size and expandability properties, the DSN-F topology can be used in supercomputer and datacenter networks, where the interconnection networks are required to gradually be expanded to meet their growing demands. The expansion method of DSN-F perfectly fits these requirements. Most of time, adding new switches requires a little effort with the simple actions as described in the previous section. We only have to connect the new switch to an existing topology without any rewiring.

The managing and expanding efforts become simpler when we carefully select the number of switches per cabinet. For example, we choose the number of switches in each layer as $p = 4$ and we use the cabinet that stores 16 switches. Naturally, the four layers are installed into one cabinet. Therefore, all the switches in the same layer are arranged in the same cabinet. All the local links in logical design now are installed inside only one cabinet. With this arrangement, we can easily manage the working with switches in terms of labeling and maintenance (i.e. replacement of old switches). Moreover, from the expansion point of view, the newly-added cabinet can be installed by connecting its cables to switches that in a single existing cabinet.

When the number of switches in the network becomes huge, we still take the above advantages by rearranging some cables to keep the network simple. From the description in the previous section, we analyze that the transformation needs $p+7$ cables rewired at each super-node (i.e. $(p+7) \times 2^p$ cables for the whole network). Each super-node has $4p+4$ links with $2p+2$ *Local_Link*, $p+2$ *Layer_Link*, and p for both *Shortcut* and *Succ*. Therefore, the number of rewired cables of our transformation method is $(p+7)/(4p+4)$. It may be high in case of small value of p. But for the future design where the number of nodes and p value are very high, the rewired cables rate can be acceptable (from 25% to 30%).

Transformation requires the change of node ID and labeling. In the general view, we renumber the ID of nodes after each splitting action. More detail, each node is presented by a tuple of three number label l, layer k and super-node s. It is easy to update the level and layer of nodes inside one super-node. The challenging action is renewing the super-node ID number that nodes belong to because we separate one super-node into two smaller ones. As a simple approach, we label super-node ID using binary numbers. For example, a supper-node S can be named by a binary number, e.g. $\overline{000}$. After transformation, super-node S is splitting into S_A and S_B which are labeled by two binary numbers $\overline{0000}$ and $\overline{0001}$.

The renumber and rewire cables action mentioned above may take some effort but the transformation does not frequently occurs. Instead, the action of adding nodes into layers of super-nodes is used usually. Therefore, we conclude our proposed expansion method is practical enough.

IV. Graph and Layout Analysis

In this section, we compare our topology generated by the incremental network expansion with typical topologies that has the same low average degree (a non-random topology Torus, a random topology DLN-4 [7], and our precedent topology DSN [3]). First, we compare them in terms of diameter and average shortest path length by means of graph analyses. Next, we compute the average cable length considering their floorplan in a machine room, using parameters of recent interconnect technology.

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

Figure 4 shows the diameter and the average shortest path length of 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 1,024 nodes are added, i.e. the network grows twice as large as the baseline, each original super-node set also grows twice as large. Due to this regularity, the diameter at $x = 1024$ is slightly lower than at $x = 768$.

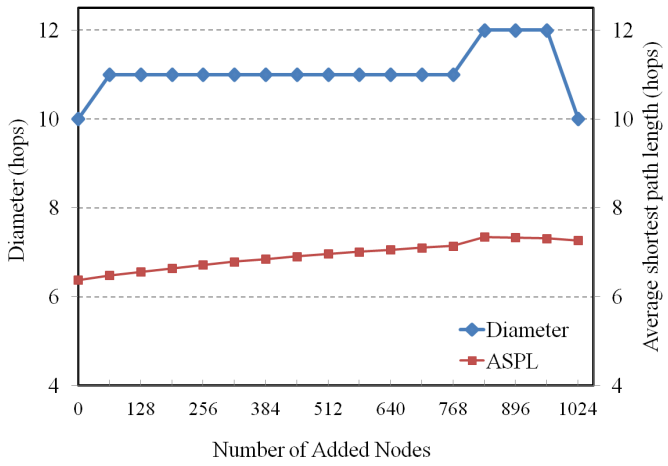


Figure 4. Diameter and average shortest path length (ASPL) vs. the number of added nodes over 1,024-node DSN-F topology.

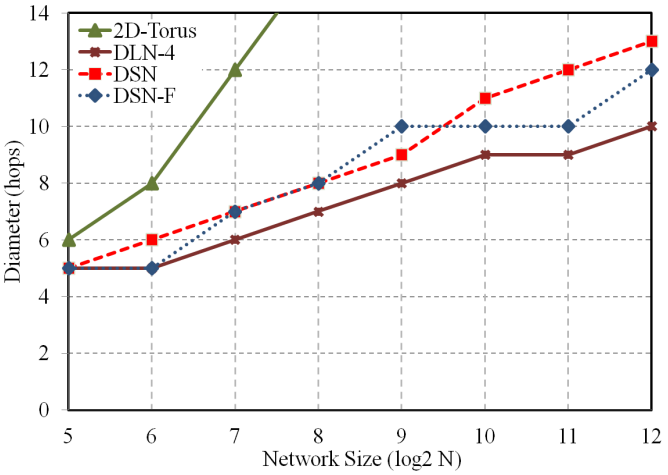


Figure 5. Diameter vs. network size.

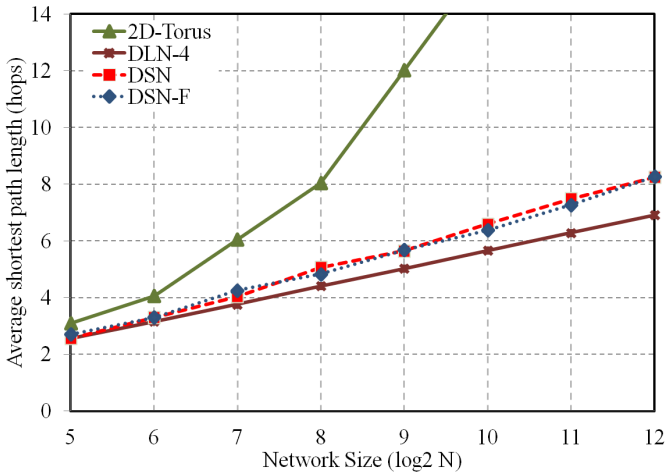


Figure 6. Average shortest path length vs. network size.

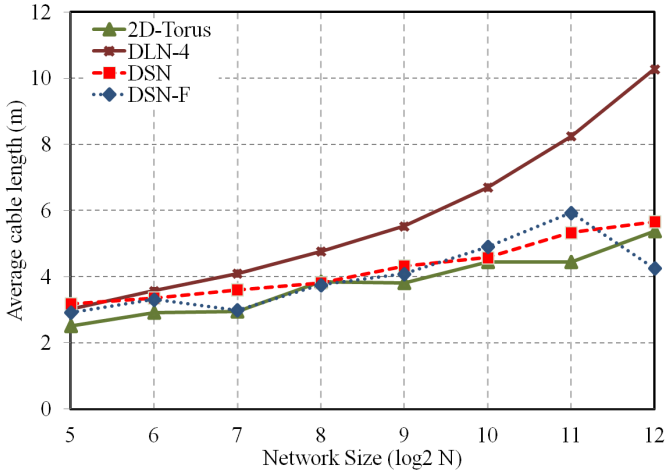


Figure 7. Average cable length vs. network size.

B. Diameter and Average Shortest Path Length vs. Network Size

The average degree of DSN-F is 4. We thus compare it with the same-degree topologies, namely 2-D torus, DLN, and DSN, with the same number of nodes. Figures 5 and 6 show the diameter and the average shortest path length of each topology. Lower values are considered better.

In all the network sizes, DLN 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.

C. 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 [5]. 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 [8]. The distance between the cabinets is computed using the Manhattan distance. We estimate average cable length in a more conservative way than in [9]: 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[2] is computed and shown in Fig. 7. Lower values are considered better. Our DSN-F topology features an average cable length that is similar to

[2]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 wraparound links are long. Thus we fairly compare 2-D torus, DSN and DSN-F in terms of cable length.

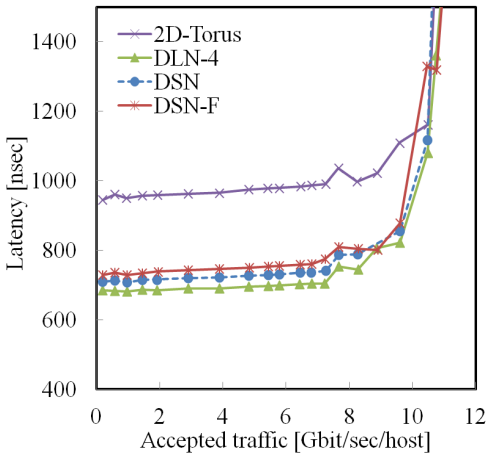


Figure 8. Throughput and latency for 128-switch networks in uniform traffic.

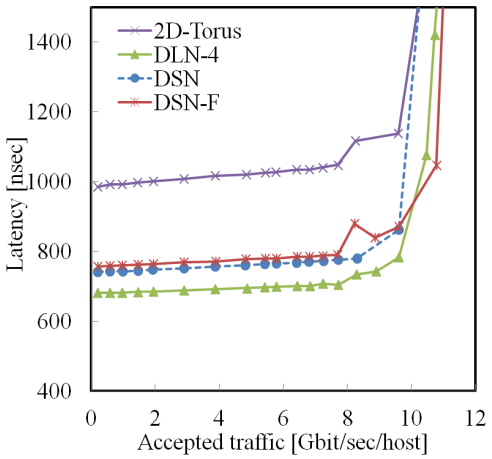


Figure 9. Throughput and latency for 128-switch networks in matrix transpose traffic.

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 super-node is mapped exactly to one cabinet.

V. Network Latency

A. Flit-level Simulation

To evaluate the latency and the throughput of our topologies, we use a cycle-accurate network simulator written in C++ [7]. Every simulated switch is configured to use virtual cut-through switching. A header flit transfer requires over 100ns including the routing, virtual-channel allocation, switch allocation, and flit transfer from an input channel to an output channel through a crossbar. We use the topology-agnostic adaptive routing scheme with the escape paths described in [10].

We simulate two synthetic traffic patterns that determine each source-destination pair: *random uniform*, and *matrix transpose*. In the random uniform traffic a random source-destination pair is sampled from a uniform distribution for each message. In the matrix transpose traffic each endpoint is also assigned a binary address and communicates with the endpoint with the binary address that is shifted by $n/2$ bits. The hosts inject packets into the network independently of each other. In each synthetic traffic the packet size is set to 33 flits (one of which is for the header). Each flit is set to 256 bits. Effective link bandwidth is set to 96 Gbps.

B. Simulation Results

Figures 8 and 9 show the simulation results for 128-switch networks for uniform traffic and matrix transpose traffic, respectively. Our main finding is that all the latency results match the observations in the graph analysis on diameter and average shortest path length. This is expected since the network latency is correlated to the hop count. By contrast, all the topologies have similar throughput, i.e. the maximum amount of accepted traffic, since they have almost the same number of links. Since this simulation assumes short packets, i.e. less than 3 KB, we conclude that a latency-sensitive short-packet traffic fit DSN-F, DSN, and DLN topologies.

VI. Application Performance

A. Event-driven Simulation

We use SimGrid as a parallel-computer simulator [11]. The network size is set to 128 switches that is the same as the flit-level simulation in the previous section. The network link bandwidth is set to 40 Gbps, and switch delay is set to 200 nsec. Whereas the processor power is set to 1 GFLOPS. We measured the performance of NAS Parallel Benchmarks (Class A, MPI version) and Himeno Benchmark for each topology generated by our expansion method. We take a minimal routing.

B. Simulation Results

Figure 10 plots the results of DSN-F, DSN, 2-D torus, and DLN topologies. The y axis indicates the Mop/s relative to 2-D torus. Higher values are considered better. Network bandwidth rather than network latency affects the performance under these parallel applications on SimGrid environment. For example, there are a few inter-process communications, i.e. AllReduce operations, in EP application. However, the AllReduce operations take a major proportion of the execution time in our simulation environment. Multiple topologies thus have

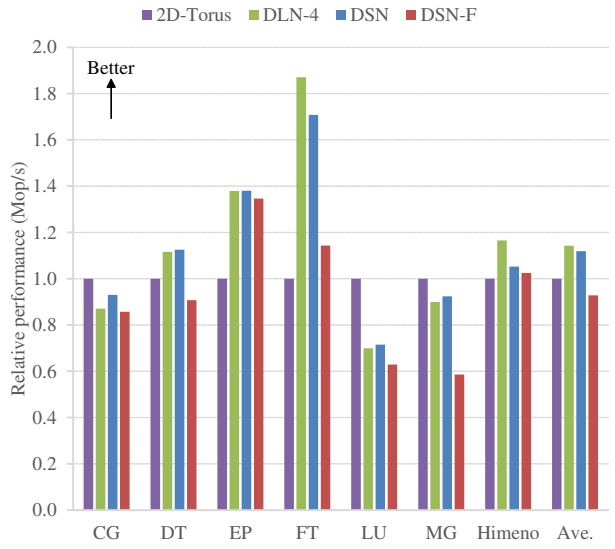


Figure 10. Performance evaluation results using NAS Parallel Benchmarks and Himeno Benchmark. "Ave" means the average over all the benchmarks. Values are normalized to those of 2D-Torus.

different Mop/s for EP application in our evaluation results. Overall, our main finding is that there is no almighty topology for a variety of bandwidth-sensitive parallel applications. All the evaluated topologies have the similar average performance over the benchmarks, because they have a similar number of network links that leads to a similar network throughput in average, as shown in the previous section.

VII. Conclusions

In this study we proposed an expandable low-degree topology, named DSN-F, for supercomputers and datacenter networks in which the number of nodes gradually increases year by year. Since their interconnection networks recently become latency-sensitive to support various massively-parallel applications [12], the network topology that exploits small-world effect, e.g. DSN [3], 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 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, DLN, and DSN topologies in terms of diameter, average shortest path length, cable length, throughput/latency, and parallel application performance. Our evaluation results demonstrated that DSN-F has similar diameter, average shortest path length, and average cable length to that of DSN and DLN. 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 DLN. As expected from the graph analysis results, the flit-level simulation results showed that DSN-F has a better latency than that of 2-D torus. The event-driven simulation showed that there is no almighty 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 and #25730068.

References

[1] Top 500 Supercomputer Sites, http://www.top500.org/.
[2] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho, "A Survey and Evaluation of Topology Agnostic Deterministic Routing Algorithms," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 405–425, 2012.
[3] V. K. Nguyen, N. T. X. Le, I. Fujiwara, and M. Koibuchi, "Distributed shortcut networks: Layout-aware low-degree topologies exploiting small-world effect," in *ICPP*, 2013, pp. 572–581.
[4] I. Fujiwara, M. Koibuchi, and H. Casanova, "Cabinet Layout Optimization of Supercomputer Topologies for Shorter Cable Length," in *Proc. of the 13th International Conference on Parallel and Distributed Computing, Applications and Technologies*, Dec. 2012.
[5] M. Koibuchi, I. Fujiwara, H. Matsutani, and H. Casanova, "Layout-conscious random topologies for hpc off-chip interconnects," in *19th International Conference on High-Performance Computer Architecture (HPCA)*, Feb. 2013, pp. 484–495.
[6] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking Data Centers Randomly," in *Proc. of USENIX Symposium on Network Design and Implementation (NSDI)*, 2012.
[7] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, and H. Casanova, "A Case for Random Shortcut Topologies for HPC Interconnects," in *Proc. of the International Symposium on Computer Architecture (ISCA)*, 2012, pp. 177–188.
[8] 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
[9] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *Proc, of the International Symposium on Computer Architecture (ISCA)*, 2007, pp. 126–137.
[10] F. Silla and J. Duato, "High-Performance Routing in Networks of Workstations with Irregular Topology," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 699–719, 2000.
[11] SimGrid, Versatile Simulation of Distributed Systems, http://simgrid.gforge.inria.fr/.
[12] J. Tomkins, "Interconnects: A Buyers Point of View," ACS Workshop, 2007.