Cable-Geometric and Moderate Error-Proof Approach for Low-Latency Interconnection Networks

TRUONG THAO NGUYEN

Doctor of Philosophy

Department of Informatics
School of Multidisciplinary Sciences
SOKENDAI (The Graduate University for Advanced Studies)

Cable-Geometric and Moderate Error-Proof Approach for Low-Latency Interconnection Networks

by

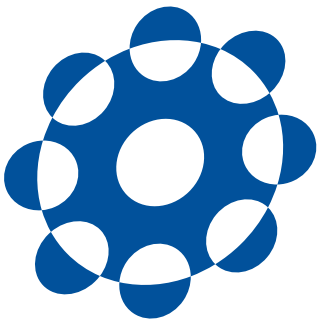
TRUONG THAO NGUYEN

Dissertation

submitted to the Department of Informatics

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy



SOKENDAI (The Graduate University for Advanced Studies)

March 2018

Committee

Advisor	Dr.,KOIBUCHI Michihiro
	Associate Professor of National Institute of Informatics/SOKENDAI
Subadvisor	Dr. JI Yusheng
	Professor of National Institute of Informatics/SOKENDAI
Subadvisor	Dr. FUKUDA Kensuke
	Associate Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. YONEDA Tomohiro
	Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. GOSHIMA Masahiro
	Professor of National Institute of Informatics/SOKENDAI
Examiner	Dr. HIROKI Matsutani
	Associate Professor of Keio University

Abstract

For decades, computational science and computer industry pursue a low latency interconnection network. In particular, as numerous endpoints of highly parallel computing systems (such as data centers or supercomputers) demand high computing and large storage, an interconnection network becomes a critical component of the highly parallel computing systems. The interconnection network is expected to provide a low latency and a high communication bandwidth. Indeed, the switch delay to forward a message becomes dozens or hundreds of nanoseconds. The receiving and sending overhead at a host could be hundreds of nanoseconds by enabling intelligent network interfaces. As device technology and its corresponding software overhead continue to improve, an expected Message Passing Interface (MPI) level communication naturally becomes latency sensitive, such as dozens of nanoseconds for custom supercomputers or hundreds for commodity clusters.

In this dissertation, we propose a new type of low latency interconnection network working towards Exascale and Big-Data computing challenges. Our approach is a combination of three following technologies; a short-diameter network topology with its low-cost physical layout, a custom routing avoiding latency overheads for accessing routing tables, and a low-latency error control mechanism for the usage of high-bandwidth optical links.

We first focus on designing short diameter network topologies and their physical layouts for achieving both low total switch latency and low total cable latency. We propose a new non-random approach named Distributed Shortcut Networks (DSNs), for designing a cost-effective and layout-conscious interconnect topology. The DSN approach is based on a common technique seen in traditional regular topology designs, e.g., the use of “virtual supernodes”, combined with a different design philosophy learned from observing small-world networks. 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. By these techniques, DSN achieves logarithmic diameters which are in proportional to the total switch latency. DSN also has advantages of structured topological properties for shorter cable lengths when it is implemented into a machine room, i.e., leading to a low total cable latency. In addition, we illustrate that DSN is fully flexible to a required network size and to an incremental expandability after its installment on a machine room. When incrementally adding nodes and cabinets to the proposed topology, its diameter and average shortest path length increase modestly. Its network cost and power consumption modestly increase when compared to the counterpart non-random topologies.

Secondly, we study the ideal performance of a proposed interconnection network when a routing algorithm is implemented. Numerous endpoints require large CAM (Content Addressable Memory)-based routing tables at each switch consuming electric power drastically. A large CAM table implemented outside the router chip becomes a latency bottleneck especially when the switch delay becomes extremely small, e.g. 40ns. In this regard, avoiding routing table is a goal for a lower switch delay. We propose an effective non-minimal custom routing algorithm on DSNs without routing tables by computing directly of synthesis hardware at each switch. These latency analyses show that our custom routing algorithm achieves a good result in the low switch-latency era.

Thirdly, we address the high switch-delay problem by reducing the latency overhead of error correction codes in high-bandwidth optical communications, e.g., 100 Gb/s or 400 Gb/s. Recent high-bandwidth optical communication has error correction codes (ECC), such as forward error correction components (FEC) at each switch for maintaining the same bit error rate (BER) as that in traditional low-bandwidth interconnection networks. However, the FEC operation latency overhead becomes higher than the sum of all the other switch operation overhead including routing computation and switch allocation. The FEC operation overhead significantly degrades the performance of parallel applications in highly parallel computing systems especially when the BER is high. In this study, we design low-latency unreliable networks using a Hamming code. Although it does not provide rigid error-free communication, some parallel applications obtain the acceptable quality of computation results with short execution time.

Finally, we illustrate our best interconnection networks integrated with DSN

topologies with its custom routing and the high-speed cables with our moderate error-proof approach. We show that our low-degree, cable-geetric moderate error-proof approach achieved a better performance compared with the same-degree counter-part network such as Torus or Random.

Acknowledgements

This dissertation becomes a reality with the kind support and help of many individuals. I would like to extend my sincere thanks to all of them.

First and foremost, I would like to express my profound gratitude to my supervisor, Professor Koibuchi Michihiro for his valuable advice, support and kind encouragement on both research and private issues. Professor Koibuchi's guidance, enthusiasm, and passion opened my eyes and turned my life into the world of academic research. During the past three years, his patience (even in the hard period) gave me a lot of freedom and energy in pursuing my research direction. Without him, this thesis would not have been possible.

In addition, I would like to deeply thank my advisors, Professor Kensuke Fukuda, Professor Yusheng Ji and the committee members, Professor Yoneda Tomohiro, Professor Goshima Masahiro and Professor Hiroki Matsutani, Keio University for their encouragement, insightful comments and hard questions. My special thanks go to Professor Nguyen Khanh Van, Hanoi University of Science and Technology and Professor Henri Casanova, the University of Hawaii for sharing with me their deep knowledge on graph theory, network simulation and writing skill. Their knowledge was very useful for me to complete my dissertation.

I also would like to acknowledge the Graduate University for Advanced Studies, National Institutes of Informatics (NII) for awarding me a scholarship, providing me with financial support to complete three years of my doctor course. I especially thank the NII international support team for the endless effort of providing a good research environment.

I would like to deeply thank members of Koibuchi lab, Dr. Ikki Fujikawa, Dr. Fabien Chaix, Dr. Shoichi Hirasawa, Dr. Yao Hu, and Ms. Mari Sato for their suggestions and comment on many works. Moreover, I would like to thank all members of the

Vietnamese group and my colleagues at NII for sharing many useful research skills as well as life in Japan. Their sharing information and daily help make me familiar with Japanese environment both in life and research.

Last but not least, I would like to thank my family member, especially my wife Do Quynh Anh who always brings happiness to my life and endures this long journey with me.

Table of Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation and Objectives

In high-end computing, i.e., supercomputer and high-performance computing (HPC), to force the system performance cross a threshold of 10^{3k} operations per second is a major milestone [2]. The first Petascale (10^{15}) was achieved in November 2008 with the Roadrunner BladeCenter system operated by the U.S. Department of Energy (DOE). In the latest rankings, the Sunway TaihuLight maintains its top position from June 2016 with a performance of 93 petaflops [3]. Subsequently, the Exascale systems are expected to break the performance barrier of the Exaflop (10^{18} floating point operations per second) in near future [4]. In addition, as the outbreak of the Internet of Things (IoT) and Big-Data applications in recent years, the demands of huge storage capacity and high-speed data access for data center (DC) systems become extremely emergent. These Exascale and Big-Data challenges thus motivate the academic research and industry technological developments of recent decades.

The interconnection network is one of the most critical concerns of the architecture design for high-performance computing and data center systems. As the number of

endpoints has significantly increased, an interconnection network is expected to provide a low latency and high communication bandwidth to meet the high computing and storage demand. A number of reports [5, 2, 6, 7] have focused on the requirements and challenges of constructing such kind of systems, revolving around many design aspects such as network topology, routing algorithm, power consumption, cost, reliability and fault tolerance, scaling ability, etc. Specifically, the key requirements for an HPC system [1] at the scale of more than 100,000 endpoints were defined in [5] as follows:

- **Low latency**: maintaining approximately 1 µs end-to-end latency across system.
- **High bandwidth**: link bandwidths are greater than 800 Gb/s.
- **High message throughput**: greater than 100 million message/s and 1000 million message/s for MPI and load/store communication models, respectively.
- **High reliability**: targeting less than 10^{-23} unrecovered bit error rates.

Motivated by these requirements, this dissertation aims to design an efficient interconnection network concentrating on the low-latency purpose. We also consider the others requirements such as high bandwidth and high reliability in the relations with the low-latency point of view. In the following, we first address the challenges of design an interconnection network and then we focus on the specific challenges related to the latency requirement.

1.2 Challenges of Interconnection Network Design

1.2.1 Overview

Interconnection network design is a well-researched area due to its important role in the HPC system design. In spite of that, the system design with 100,000 endpoints still needs considering thoroughly for a proper solution on several aspects. It is reported in [8] that these aspects include the following:

[1] Because the demands of interconnection connection design for data center and high-performance computing systems becomes converging, in this dissertation, if there is no more explanation we use the term "*HPC system*" to mention both of them, i.e., HPC system and data center system.

1 **Performance requirements**. In HPC system, the endpoints synchronize, communicate together, or perform data access operations through the interconnection network by passing messages between a particular pair of source and destination nodes. Indeed, the time taken for a message reaches its destination (message *latency*) directly affects the performance of overall system. Low maximum/average message latency is expected for a high-scale system. In addition, the number of messages delivered through the network at the same time (network *throughput*) is also an important concern in term of high-performance system design.

2 **Incremental expandability**. A large number of data center and supercomputers are gradually increased their size after deploying because the difficulty in precisely estimating future user demands and the financial/political nature. Some interconnection networks are difficult to expand because of their coarse design size with a fixed number of endpoints. In other cases, expandability is provided by using the wasting resources, e.g., reservation switch ports for future expansion usage. In this context, the interconnection networks should provide incremental expandability. That is the network could be built up with any required size and allows the addition of a small number of nodes while minimizing resource wasting.

3 **Reliability**. An interconnection network should be able to pass a message reliably. For a limited number of occurred faults on the delivered path, it is expected to send the message through some alternative paths. In addition, bit flips rate (soft error) become higher with the higher scale of networks size and speed of cables (link bandwidth). Thus, a cost-effective error detection/correction mechanism is essential for HPC systems in order to achieve the reliability requirement.

4 **Physical constraints**. An interconnection network connects endpoints such as processors, memories, etc, via a large number of components such as switches and cables. As the number of endpoints has been increased, e.g., 100,000 endpoints, the number of such components also increases that leads to take some physical constraints into account. A major challenge for a large interconnection network when it is implemented into a server room is how to arrange these components in a limited area (a *floor plan*). A floor plan is able to highly affect the network latency and establishing cost due to the long length of connecting cables or affect the power consumption of the cooling systems. Thus, physical layouts of an

interconnection network are also desirable for carefully considering.

5 **Cost constraints**. It is unreasonable to deploy an interconnection network with an infinite cost. Indeed, interconnection network design is to balance the trade-offs between cost and other design factors such as overall system performance. Therefore, increasing the cost-efficiency, i.e., maximize the performance with a particular cost, is a practical concern for designing HPC systems.

In this dissertation, we focus on minimizing the network latency for high-performance purpose while still step-by-step consider the other problems. In the next section (1.2.2), we clarify our targeted obstacles for achieving low-latency communication in an HPC system.

1.2.2 Low-Latency Challenges

Let us firstly give a brief definition of the network latency. "*The latency of a network is the time required for a packet to traverse the network, from the time the head of the packet arrives at the input port to the time the tail of the packet departs the output port*" [1]. Principally, network latency includes switch latency, time of flight (cable latency), and serialization latency. The switch latency is the time spent at switches on the traversed path of a message. Typically, the switch latency depends on the number of intermediate switches between source and destination (also known as hop count H_{min}[2]), and the delay at a switch t_s for routing computation, switch allocation, error control operation, etc. On the other hand, the time of flight is the time spent on the cables that are in proportional to the total cable length L_{min} and the inverse of propagation velocity v. The serialization latency is the time for a message of size S to cross a channel with bandwidth b [1]. In the ideal case where no contention occurs, the following expression illustrates the ideal latency of a network:

$$\text{Latency} = H_{min}\ t_s + \frac{L_{min}}{v} + \frac{S}{b} \tag{1.1}$$

Due to the wide base of installed applications using Message Passing Interface (MPI),

[2]In the ideal case, we assume that all the messages are delivered through the shortest routing paths. Thus we use H_{min} to denote the hop count of this minimal path. In the other cases, routing path depends on a particular routing algorithm and may not be the shortest path. In this case, the switch latency is proportional to the real routing path length.

and Partitioned Global Address Space (PGAS) programming models, small delivered messages with a collective communication is frequently generated in the interconnection networks. Thus, as the high-scale of endpoints, the switch latency and time of flight become bottlenecks of network latency. In this dissertation, we are interested in three design challenges of interconnection networks including network topology, routing algorithm, and high-speed cable in term of minimizing network latency.

- **Network topology and physical layout**. Previously, the delay at a switch was relatively larger than the delay on cables, i.e., hundreds of nanoseconds per switch, and 5 nanoseconds per meter, respectively. Low latency network design is targeted at reducing the maximum hop count between any pairs of source and destination which corresponds to the diameter of a network topology. As the technology has been driven, the switch delay becomes small, such as 40.1 nanoseconds on Anton-2 and about 100 nanoseconds even in a commodity InfiniBand switch which uses a forwarding table. The total cable latency (or total cable length) also significantly contributes to the communication latency. Thus considering short-diameter network topologies and their physical layout for achieving both low total switch latency and low cable latency becomes a crucial concern. Such cable-geometric design is needed, but their good design methodology is not given.

- **Routing algorithm**. A large number of endpoints drastically requires the large CAM (Content Addressable Memory)-based routing tables at each switch that consumes more power and potentially donates an extra switch latency overhead. In this context, compact routing with the usage of energy-efficiency Ternary CAMs (TCAMs) has been researched to cope with this problem. Another approach is to design a routing algorithm that does not require routing table by it's capable of computing the routing path with hardware synthesis at each switch.

- **High-speed cable usage**. Not only network latency, but also network throughput is an important factor for the performance of HPC systems. In order to achieve high network throughput, besides network topology design, increasing the speed of cables is also an essential approach. In order to achieve the high-bandwidth requirement, high-speed optical fiber such as Gigabit Ethernet has been studied and commercially designed with the use of advanced modulation technology and forward error correction (FEC) to maintain the reliability of the network

communication. However, the use of FEC contributes a considerable extra latency overhead that might not be allowed since it significantly slows down the systems. Thus, cost-effective error detection techniques and other fault-tolerant mechanisms like light-weight fast correction code would be a challenge of HPC system design.

1.3 Contributions

In the beginning work, we propose short diameter network topologies called Distributed Shortcut Networks (DSNs) by exploiting the small world network model. The DSN topology takes full advantages of the structured topological properties such as the low degree and the non-random that help to gain a short cable length when it is implemented into a server room. That is, our new DSNs have performance somewhat comparable to the state-of-the-art low-degree interconnection networks but have clear advantages in saving cables that help to reduce the deployment cost. In addition, we illustrate that our network topologies support incremental expandability for further extension demands after its deployment.

In the following work, we aim at avoiding the routing tables with the purpose of a low switch delay. We propose an effective hardware-based non-minimal custom routing algorithm on our DSN network topologies. This custom routing also could be implemented as a compact routing with TCAMs usage. Although our proposed routing does not ensure the shortest routes for all pairs of source and destination nodes, it overall latency still small due to taking the advantages of routing over the short-cable-length paths, i.e., low cable latency. Furthermore, we also show that for a given number of occurred faulty/congested links on the delivered paths, the message still reaches the destination through the alternative paths with a proper increment of routing path length.

In the final work, we consider the low-reliable network design using Hamming code (moderate error-proof high-speed cables). The Hamming code does not provide rigid error-free communication, its BER is indeed acceptable for a large number of parallel applications. Hamming code has advantages in reducing the switch delay then help to improve the system performance.

Finally, we illustrate our best interconnection networks in which DSNs topology with its custom routing and moderate error-proof high-speed cables are implemented and compare them to counterpart conventional interconnection networks. Table 1.1 highlights our contributions in this work.

Table 1.1: Problems statement and our contributions.

Network Topology (Chapter 3)	**Problem**	Scalability limit
	Objective	Short network diameter with short cable length in server room
	Proposal	Non-random small-world network
	Efficiency	Reduce both switch latency and cable latency
Routing Algorithm (Chapter 4)	**Problem**	Switch latency by implementation of routing
	Objective	Avoid routing table usage
	Proposal	Algorithmic routing (custom routing)
	Efficiency	Reduce the switch latency
Point-to-point Link (Chapter 5)	**Problem**	Latency overhead by FEC
	Objective	Avoid RS-FEC usage
	Proposal	Hamming FEC
	Efficiency	Reduce the switch latency in high-bandwidth link

1.4 Dissertation Organization

The remainder of this dissertation is organized as follows. Chapter 2 elaborates on the background and related work in the area of network topology design, routing algorithm designing, and reliability point-to-point link design, respectively. Chapter 3 presents our first approach to design a new small-diameter short-cable network topology. We also illustrate that our network topology supports incremental expandability for further extension demands after its deployment. In chapter 4, we introduce the routing algorithm on our proposal DSNs network and show its ability to deal with faults/congestion. Chapter 5 gives the details of our fast moderate error-proof mechanism, which has a low latency overhead that helps to improve the overall system performance. In chapter 6, we show the effects of our three approaches via the comparison with the counterpart conventional networks. Finally, in chapter 7, we summarize the contributions of this dissertation and discuss the future work. Figure 1.1 illustrates the outline of this thesis.

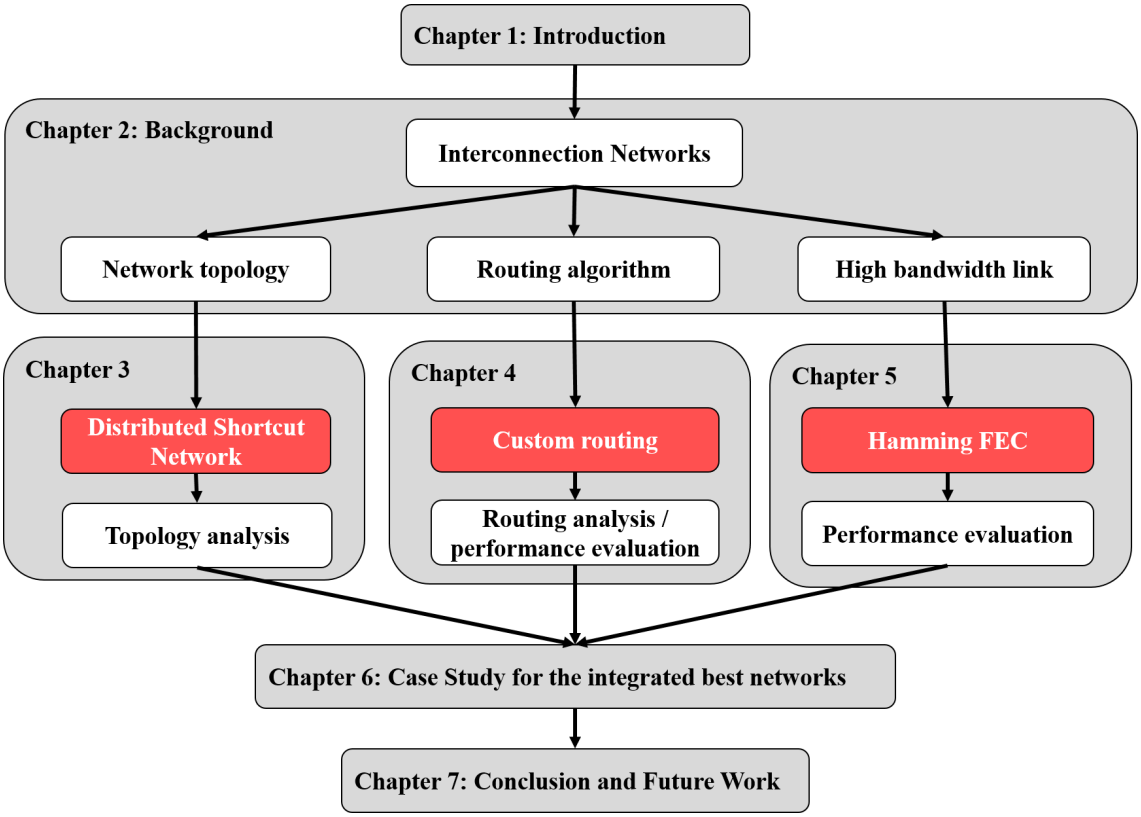


Figure 1.1: Outline of our thesis with highlighted contributions in red.

2

Background

In this chapter, we introduce basic background and notations that are used throughout the dissertation. We firstly give a definition of the network topology and summary some well-known topologies that frequently used in HPC system. We also introduce the baseline layout for deploying an interconnection network into a server room. This layout is used for out interconnection network design where physical layout and cable length is one of the important design aspects. Then, we give an introduction to the basis of routing and two approaches to express a routing algorithm. In Chapter 4, we consider these two approaches for design an effective custom routing on our proposed interconnection network. Finally, we briefly explain the Physical Layer for high-speed reliable links which may use for HPC systems in the future.

2.1 Interconnection Network Topology

2.1.1 What is Network Topology

Designing the network topology is the first step in designing an interconnection network because the physical layout, the routing scheme, and the switching method depend heavily on the topology. "*Network topology refers to the static arrangement of channels and nodes in an interconnection network*" [1]. A node may be an endpoint, e.g., a processor, a memory, and a storage device, or a switch node that forwards packets for an end-to-end communication. The network in which every node play both roles as an endpoint and a switch refers to a *direct* network. Instead, if endpoints are connected to the others through some switch nodes, the network is classified as an *indirect* network. In this section, we formally define the direct network and the related terminologies. These definitions also would be used for representing the switch-to-switch subnetwork of an indirect network.

A direct network topology and its terminologies can be borrowed from the graph theory. We first define the network topology as follows:

Definition 2.1 *A direct network topology is represented by a connected graph G whose vertices V(G) and edges E(G) represent the nodes and the communication channels/ links, respectively. Two vertices u and v are adjacent if a link forms an edge* $e(u,v)$.

Each node u has a set of links that connect it to other nodes. The number of links of a given node is called node *degree* that particularly refers to the radix of a switch, i.e., the number output/input ports. Formal related definitions of degree are given in Definition 2.2

Definition 2.2 *For a given network G:*
1. Degree of a given node u, $deg_G(u)$ *is the number of adjacent nodes of u.*
2. Radix / degree of a network G is presented by maximum value of node degrees, $\deg(G) = \max_{u \in V(G)} (deg_G(u))$.
3. A network is "regular" when all the nodes have the same degree.

An interconnection network is required to support the communication between any pair of nodes in the network. That is, every vertex pair u,v is joined by a path $p(u,v)$ as in Definition 2.3. Typically, there exists more than one path for a pair of nodes. Thus, the actual traveling path for a packet is expected to be the path with the

shortest length because the length of the routing path heavily affects the switch latency as mentioned in Chapter 1. The length of this shortest path for a pair of nodes u and v is called the *distance* between them. In network topology design, the maximum distance between every pair of nodes (known as network *diameter*) is one of the critical concerns. Designing a shorter diameter network topology helps to support the lower ideal communication latency of a network.

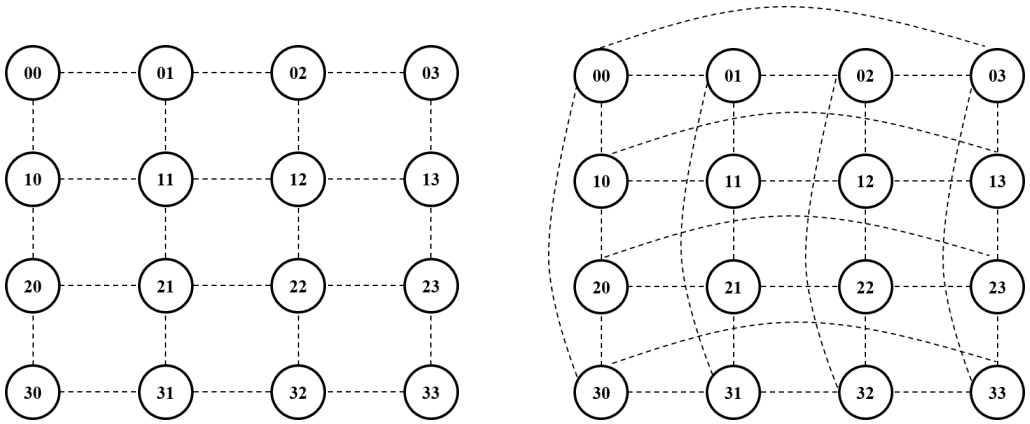
Definition 2.3 *For a given network G:*
1. A path between a given pair of nodes u and v is a sequence of adjacent edges, $p(u,v) = \{e_1(u,v_1), e_2(v_1,v_2), \ldots, e_n(v_n-1,v)\}$.
2. The length or hop count of a path $p(u,v)$ *is the number of edges,* $len(p(u,v)) = |p(u,v)|$.
3. Distance between vertices u and v, $dist_G(u,v)$ *is the length of shortest path joining u and v.*
4. Average distance (average shortest path length - ASPL) in network G is $\frac{\sum_{u,v} dist_G(u,v)}{N(N-1)}$ *where* $u \in V(G), v \in V(G), u \neq v$ *and* $N = |V(G)|$.
5. Diameter of network G is the maximum distance between any two vertices of G, $diam(G) = \max_{u,v} dist_G(u,v)$.

Beside network latency, as we mentioned in Chapter 1, network *throughput* is also an important concern in term of the high-performance system. Throughput is the number of messages delivered through the network at the same time. Throughput depends not only on network topology but also on routing algorithm and also flow control. However, it is well-known that the ideal throughput can be determined through graph theory with the assumption of perfect routing and flow control [1]. The upper bound on the ideal throughput of a network topology is given as in Equation 2.1 where N is the number of nodes, b is the link bandwidth and $bw_e(G)$ is the (edge) bisection width of the network G (Definition 2.4).

$$\Theta_{ideal} \leq \frac{2 \times b \times bw_e(G)}{N} \tag{2.1}$$

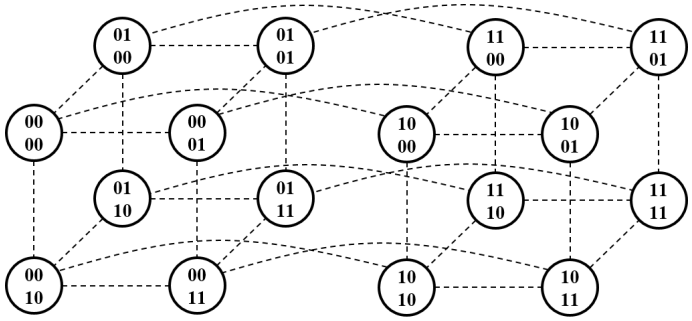
Definition 2.4 *The edge bisection width* $bw_e(G)$ *of a given network G is the smallest number of edges removal of which divides G into two parts of equal size.*

In summary, the objective of our topology design is to have a short diameter and a high bisection bandwidth. Although many network topologies have been proposed, very a few of them have ever been implemented. In the next section, we summary some popular network topologies.



(a) 4-ary-2-mesh (4 × 4 Mesh) network

(b) 4-ary-2-torus (4 × 4 Torus) network



(c) 4-cube Network (Hypercube of 16 nodes)

Figure 2.1: Mesh-based topology.

2.1.2 Popular Network Topologies

Mesh-Based Topologies

Mesh-based topologies, also called strictly orthogonal topologies, include n-dimensional mesh (k-ary-n-mesh), n-dimensional torus (k-ary-n-cube) [9] and n-dimensional binary hypercube (n-cube).

In the k-ary-n-mesh, the nodes are arranged in an $k_1 \times k_2 \times \cdots \times k_n$ grid with $k_i \leq k$ nodes in each dimension i. Formally, a mesh is defined as a graph M, where each vertex a is identified by a n-tuple $(a_1, a_2, \ldots, a_n); 0 \leq a_i \leq k_i - 1$ for all $i \in 1 \ldots n$. Two node u and v are adjacent if $u_i = v_i$ for all i except one, $u_j = v_j \pm 1$. Clearly, mesh is not a regular network because the node degrees are from n to $2n$ and the diameter of mesh is $\sum_{i=1}^{n} k_i - 1$. Figure 2.1(b) presents an example of 16-node 4×4 mesh with the diameter 6 where the corner nodes and the internal nodes have degree 2 and 4, respectively.

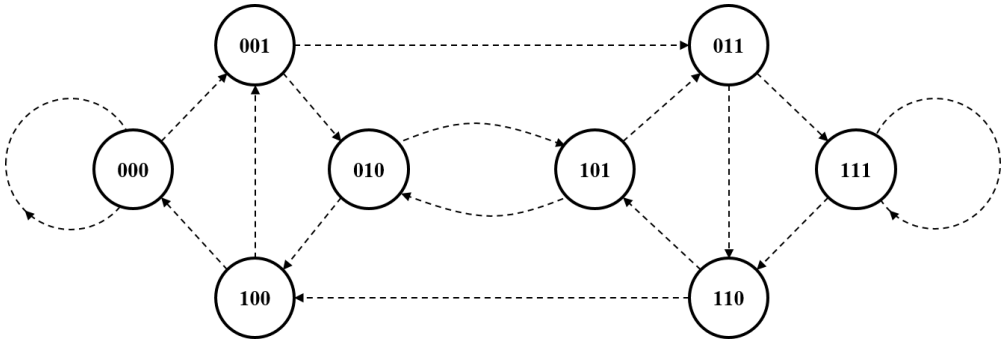


Figure 2.2: De-Bruijn(2,3) network.

To decrease the network diameter, the degree of nodes in the conner can be increased to $2n$ by adding links as shown in Figure 2.1(a). That forms the k-ary-n-cube (or torus) network with the diameter of $\sum_{i=1}^{n}(\lceil \frac{k_i}{2} \rceil)$. Formally, a torus is defined as a graph T, where each vertex a is identified by a n-tuple $(a_1, a_2, \ldots, a_n); 0 \leq a_i \leq k_i - 1$ for all $i \in 1 \ldots n$. Two node u and v are adjacent if $u_i = v_i$ for all i except one, $u_j = (v_j \pm 1) \mod k_i$. Every node has degree $2n$ if $k > 2$, thus torus is a regular network.

Another regular topology is the n-dimension hypercube which is a special case of the two network above . The hypercube H is a graph with 2^n vertices which labeled by n-bit binary strings with edges connecting two vertices if their labels differ in a single bit. Each node of hypercube has the same degree n. Figure 2.1(c) shows an example of 4-dimension hypercube. The diameter of a hypercube network is n because in the worst case, a source and a destination address of a message can differ in all n bits.

Mesh-based topologies have been widely used for the interconnection network of HPC system. It is frequently used in supercomputers such as BlueGene/L Anton-2 [10] or Cray XT5 [11]. For the latest update, the top-4 fastest supercomputer by June 2017 [3] named Titan [12], implements Cray Gemini interconnect which based on 3-D Torus network. The top-5 supercomputer named Sequoia, and top-9 Mira supercomputer use IBM BlueGene/Q architecture [13, 14] which implemented 5-D Torus network. In addition, Fujitsu introduced Tofu-interconnection network based on 6-D Torus in K-Computer [15].

Shuffle-Based Topologies

Many topologies have been proposed for the purpose of minimizing the network diameter for a given number of nodes and node degree. Typically, the shuffle-based topologies

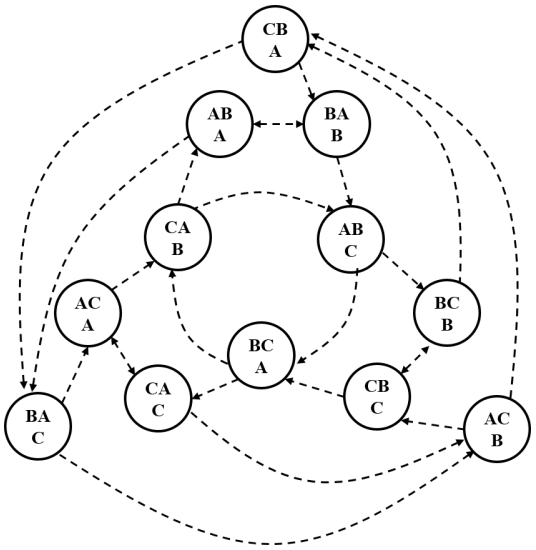


Figure 2.3: Kautz (3,3) network.

including De-Bruijn [16], (n,k)-Star [17, 18] and Kautz graph, have a logarithmic diameter and constant node degree.

De-Bruijn network of degree d and dimension n (denoted as De-Bruijn(d,n)) has d^n nodes, each node x is represented by a set of n symbols $(x_{n-1},x_{n-2},\ldots,x_1,x_0)$ where x_i in range of d different symbol values, e.g., a base-d digit; $0 \leq x_i \leq d-1$. A node u is adjacent to node v if the first $n-1$ symbols of v equal to the last $n-1$ symbols of u. Because any node can be obtained from another node by at most n shifts, hence the diameter of De-Bruijn topology is just n hops. However, it is reported that the average shortest path length of De-Bruijn is very close to the diameter. This is a disadvantage compared to mesh-based topologies. Figure 2.2 shown an example of 8-node De-Bruijn(2,3) with degree of 4 and diameter of 3.

A vertex of the n-Star network is labeled by permutations of 1 though n. A permutation is connected to every other permutation that can be obtained from it by interchanging the first symbol with any of the other symbols. A n-Star graph has $n!$ nodes with a degree of (n-1) and a diameter of $\frac{3(n-1)}{2}$.

A Kautz(n,m) graph is a labeled graph, vertices are labeled by strings of length n above an alphabet with m letters. Every two consecutive letters in a string must be different. There is a directed edge from a vertex u to another vertex v if it is possible to transform the string of u into the string of v by removing the first letter and appending a letter to it. Figure 2.3 presents Kautz(3,3) network. Each node is presented by a string of

length-3 above 3 different letters A,B,C. The node ABC has directed outgoing links to the node BCA and the node BCB. A Kautz network has a size of $(m-1)^n + (m-1)^{n-1}$ nodes with a constant degree, i.e., $2(m-1)$ for unidirectional links and $m-1$ for bidirectional links, and a short diameter, i.e., approximates $(n-1)$ hops.

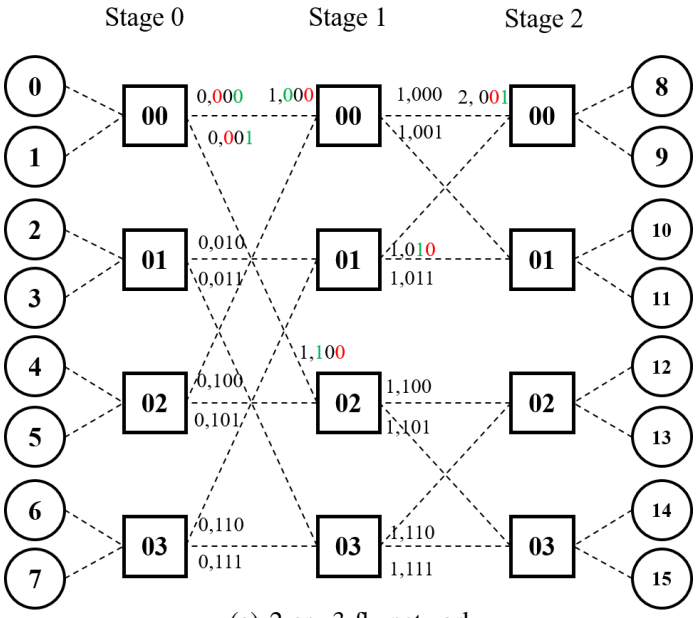
Although shuffle-based topologies has a short diameter and a constant degree, it is not well applied to HPC system due to its complicated complexity of routing algorithms.

Indirect Switch-Based Topologies

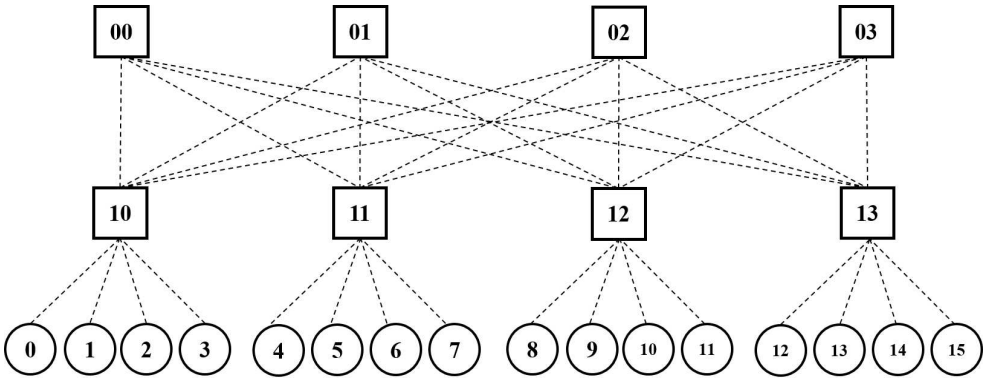
Other topologies have been proposed for the purpose of minimizing the network diameter is indirect switch-based topologies. In this section, we introduce two popular networks include Butterfly (k-ary n-fly) [19] and k-ary n-tree [20].

k-ary n-fly network (n stages of degree-k switches) is an indirect network that consists of $2 \times k^n$ endpoint, n stage of k^{n-1} cross bar switches. We consider the network of switch nodes as a direct network. Each vertex of this network, i.e., a switch, has k input links and k output links which can be identified by a stage-level i and an n-digit radix-k number, $\{d_{n-1}, d_{n-2}, \ldots, d_0\}$. An output link of stage $i-1$ is wired to an input link of stage i by permuting the d_{n-i} and d_0, i.e., wire the switch (i, $\{d_{n-1}, \ldots, d_{n-i}, \ldots, d_0\}$) and the switch ($i+1$, $\{d_{n-1}, \ldots, d_0, \ldots, d_{n-i}\}$) together. For example, Figure 2.4(a) presents the 2-ary 3-fly network topology with 3 stages, 4 switches per each stage. Links are labeled as shown in the figure with 3-digit of radix-2 addresses. An output link of stage-0 connects to an input link of stage-1 by exchanging the first and the third digits of address, e.g., (0,001) connects to (1,100). Similarly, an output link of stage-1 connects to an input link of stage-2 by exchanging the second and the third digits of address, e.g., (1,010) connects to (2,001).

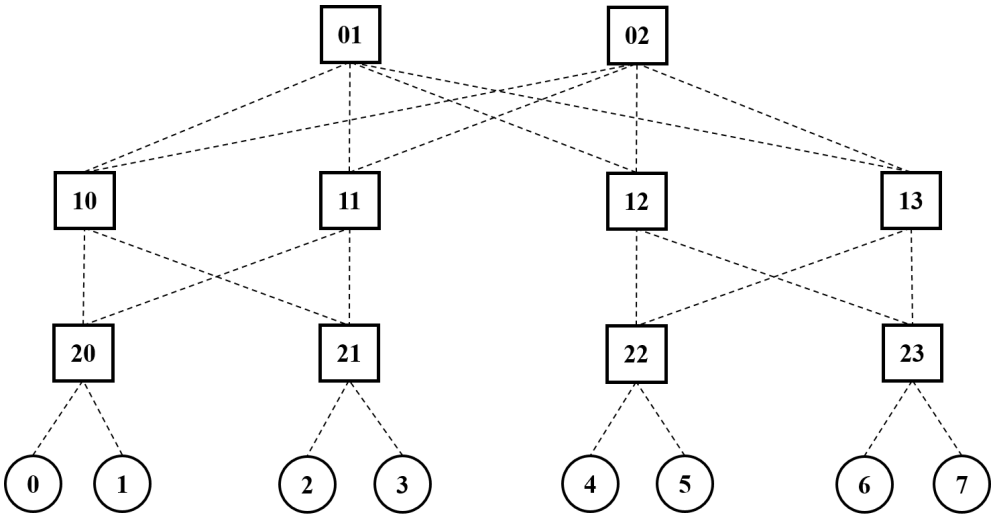
A tree in which every node but the leaves has a fixed number k of descendants is a k-ary tree. A k-ary n-tree is a k-ary tree with n levels of nodes. Root nodes are count as the first level-0 while leaf nodes are in the level-(n-1). Figure 2.4(b) and Figure 2.4(c) show examples of 4-ary 2-tree and 2-ary 3-tree, respectively. Clearly, decreasing the height of the tree, i.e., the number of levels, helps to reduce the network diameter but it requires a higher node degree. For low latency purpose, fat-trees which are high-degree low-level trees frequently use in HPC system. Indeed, the top-1 supercomputer, i.e, Sunway TaihuLight, the top-2, i.e., Tianhe-2 (MilkyWay-2), and the top-7 Oakforest-PACS [3] are built up based on fat-tree structure [21, 22].



(a) 2-ary 3-fly network



(b) 4-ary 2-tree network



(c) 2-ary 3-tree network

Figure 2.4: Indirect network topology.

Random Network and Small-World Network Model

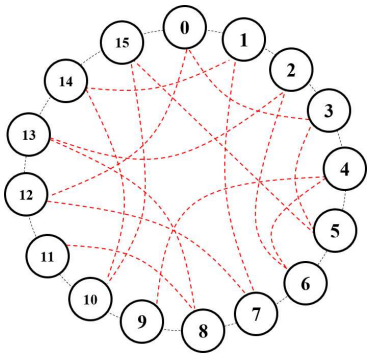
As a subject of extensive studies, small-world networks have been introduced to model network structures of popular real-world large-scale networks [23]. Typically, a small-world model is a simple regular topology such as a torus with added random links. In Watts and Strogatz's model, a ring is augmented with random links: with a given probability parameter p, each local ring link is considered to be replaced by a uniform random link [23]. This work drew a lot of attention on small-world models, where just a few added random links can drastically reduce the diameter. Below we briefly review some existing topologies, which are inspired by small-world networks.

1) Random Shortcut Networks [24]: Typically, (uniform) random shortcuts are added to a ring of nodes. RSN(n,k) is formed based on a ring of n nodes, such that every pair of nodes has the same probability to have a shortcut connecting them, and the degree of all the vertices of the resulting graph is the same k. Figure 2.5(a) shows an example of RSN(16,4).

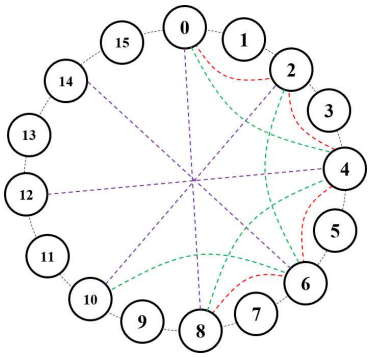
2) For DLN-2^x (Figure 2.5(b)) with a given degree x, we reuse a special Distributed Loop Network description from [25]: n vertices are arranged in a ring, and a shortcut is added between vertices i and j such that $j = i + \lfloor n/2^k \rfloor \mod n$ for $k = 1,\ldots,x-2$. For $x = \log n$, obviously, for any given destination node t, any other node u has at least one shortcut link that halves the distance to t. Thus, it is easy to see that this graph has a logarithmic diameter.

These long-range shortcuts have made possible the distance-halving technique, which is widely used in analyzing graphs as well as designing topologies with a small diameter. Generally, in these graphs, when a node u searches for a path to another node v, it will find a long link from itself or a nearby node that goes closer to v by at least half of the (u,v) distance (for a given predefined distance metric). As mentioned, DLN-2^x with $x = \log n$ has a logarithmic diameter and also a natural simple routing logic, but has degree $\log n$, which is not as low as desired.

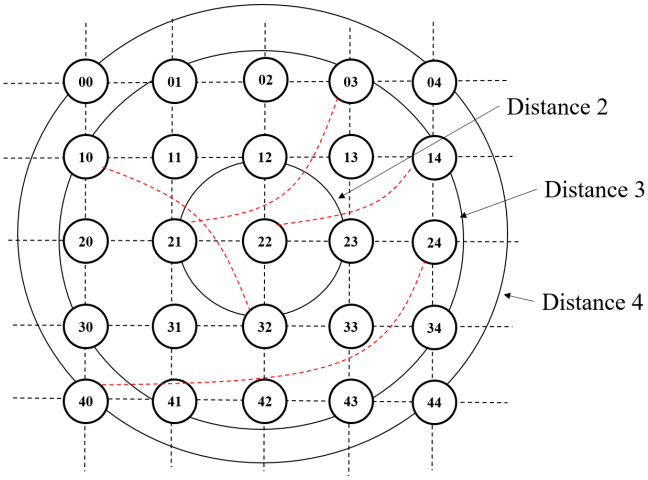
Now we look closer at Kleinberg's small-world network [26]. In a base $n \times n$ grid graph, each node has at most 4 links to the neighbor nodes (Figure 2.5(c)). A random shortcut is attached to each node, where the node u's shortcut goes to another node v with the probability inversely proportional to the square of the lattice distance between u and v. Kleinberg's small-world network has short cable length, with a constant degree and logarithmic diameter. This graph features the small-world effect in full: short routes



(a) Random shortcut network: ring network plus uniform random links (red).



(b) Distributed loop network: ring network plus various range shortcut links (color line)



(c) Kleinberg's Small-world: grid network plus random links. Probability of link between u and v, $p(u,v) \sim \text{distance}(u,v)^{-\alpha}$.

Figure 2.5: Small-world network model.

between any two nodes are abundantly available, and moreover, any node u can find a path of length $O(\log^2 n)$ to any other node v by using the local information only. This is done by greedy routing, which works naturally based on the distance-halving technique. However, greedy routing can only find paths of length $\theta(\log^2 n)$ [27], asymptotically quadratic of the minimum.

Recent research in the interconnection network for HPC systems has been attention on the random network and Small-world network model. Shin et al. propose a degree-6 random topology [28] based on Kleinberg small-world network model [29] for datacenters which have a small diameter, i.e., approximately $\log n$. In addition, Singla et al.

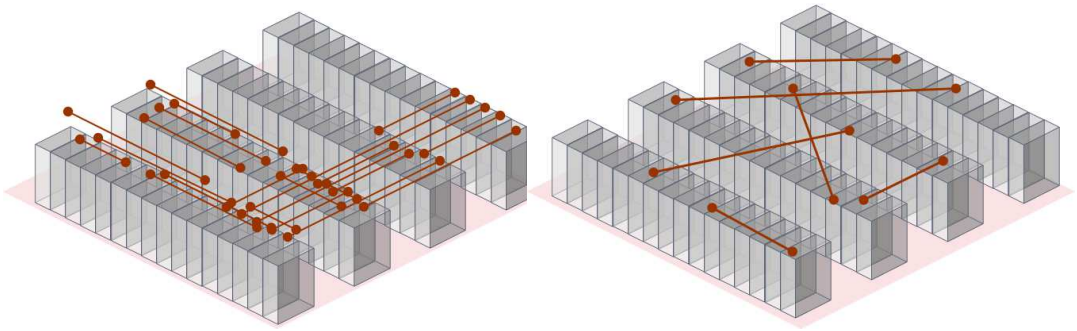
propose a random regular graph called Jellyfish for data-centers. For a given r-regular random graph, where all the nodes have the same degree r, the authors show that the path length of Jellyfish increases logarithmically (with the base r) with the number of nodes in the network. Although Jellyfish can be used in both low-degree and high-degree network designs, the authors claimed that Jellyfish accommodates a large number of servers by using a small number of links at each switch for interconnection while still achieving high capacity. Moreover, the work in [24] investigates the idea of adding random links into classical topologies for HPC systems. They found that this method can significantly reduce the diameter even in a low-degree design.

2.2 Physical Layout

2.2.1 Baseline Layout in a Server Room

A server room of a High-Performance Computing system or a high-density data center is practically deployed as rows of cabinets (or racks) as shown in Figure 2.6. Such a floor layout helps to achieve the maximum number of possible cabinet locations in a limited space and support effective power and cooling distribution systems [30, 31]. For example, cabinets are arranged front-to-front and back-to-back to form the air cooling system with hot aisles and cold aisles [31, 32]. The cold air flows up from the underfloor to the cabinets via the cold aisles, while the heated air flows out from the cabinets to the room air conditioning unit (CRAC) through the hot aisles and the ceiling.

The computing nodes, the servers, and the switches are arranged into cabinets and connected together via cables. Cables within a cabinet are called intra-cabinet cables, and the other cables are called inter-cabinet cables. Modern data centers typically are deployed using Top-of-Rack (ToR) architecture in which the switches reside at the top of the server cabinets. Thus, inter-cabinet cables are put into several cable trays (pathways) which are a few feet above the top of cabinets. As the number of switches has been increased, group cables into direct pathways become difficult because of the frequent appearance of diagonal cables and pathways. This diagonal cabling approach (illustrated in Figure 2.6(b)) requires more efforts of cable maintenance and air flow control. Instead, current designs normally arrange the pathways based on the Manhattan cabling approach as in Figure 2.6(a). That is: (1) the pathways run through the horizontal and vertical directions and (2) the crossing points of the pathways are only allowed above the cabinet



(a) Conventional cabling (Manhattan) (b) Diagonal cabling

Figure 2.6: Manhattan and diagonal cabling approaches in a server room.

rows so as not to disturb the air-flows.

2.2.2 Cable Length Optimization

Establishing a floor plan for a logical topology can be viewed as placing switches into cabinets and the cabinets into possible locations in a server room. Previously, traditional topologies used in supercomputers, e.g., k-ary n-cube topologies, have a low degree and regular structure that can be easily and cost-effectively deployed. However, recent advances of routing algorithm allow to use any topology which has shorter minimum routing path length, e.g., higher-degree topologies [15, 33, 34] or unstructured random topologies [35, 28]. Therefore, attention has been paid to the way to deploy these new kinds of topologies with a very short total cable length.

Recently, layout-conscious topologies have been proposed aiming at providing both short path length and short cable length. The main idea of this approach is to *manually* design the cabinet layout to minimize the number of inter-cabinet cables, i.e., highly connected switches are arranged into the same cabinet by taking advantage of the topological properties of the logical topologies. For example, high-degree networks such as Flattened butterfly [33], Dragonfly [34] or Slimfly [36] present the term like "*group of switches*"[1] in their topology designs and map one or several groups into a cabinet. The inter-cabinet cable now can be measured and reduced by considering the inter-group links. By contrast, in Skywalk [37] the designers consider the layout first, i.e., they start from free switches placed in a grid of cabinets, then inter-cabinet links are added

[1] it is called "dimension" in Flattened butterfly [33]

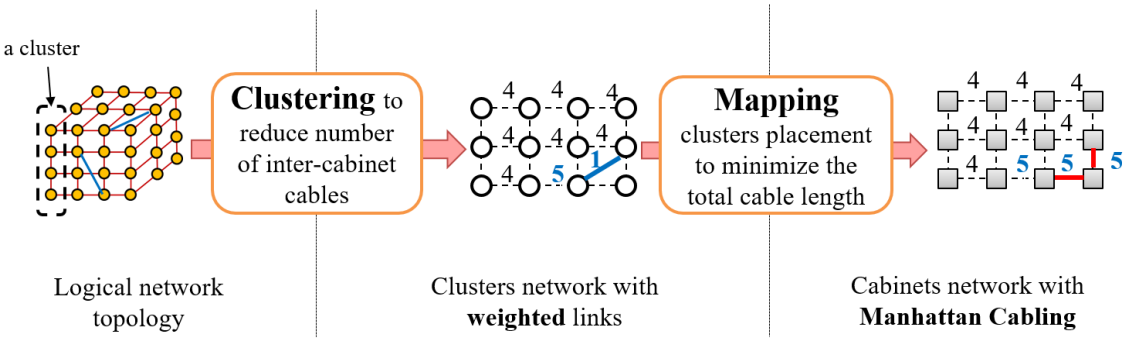


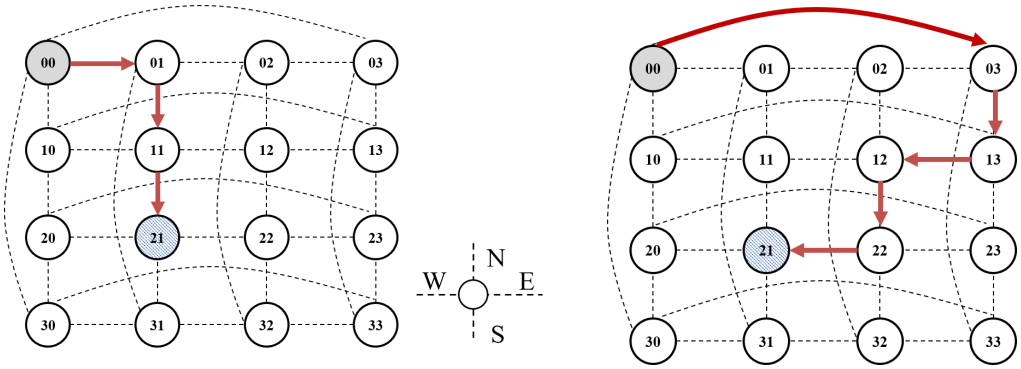
Figure 2.7: An approach for shortening the total cable length.

randomly along horizontal and vertical directions.

Another approach is to formulate the mapping problem into a well-known optimization problem and then solve it *algorithmically*. For example, Fujiwara et al. separate this problem into two optimization steps: (1) clustering a logical topology into a weighted graph of cabinets with the goal of minimizing the number of inter-cabinet cables, and (2) mapping this graph onto a physical layout on a floor plan which can be modeled as a quadratic assignment problem (QAP) [38]. In 2015, HP published their approach to data center cabling and defined the problem as a Minimum Linear Arrangement Problem (MLAP) [39]. Both QAP and MLAP are NP-hard and can be solved using a heuristic approach.

In this dissertation, we aim at achieving the short total cable length by targeting the layout-conscious topology approach. In addition, for the fair comparison with another network topologies, we assume that each of the compared networks is optimally deployed in a floor plan by using the algorithmic approach mentioned in [38]. Figure 2.7 presents the overview of this method. In this example, a grid-like topology[2] of $4 \times 3 \times 3$ nodes is deployed into a layout of 4×3 cabinets. Four nodes have been grouped into a cluster in an optimal way in which there exist only one inter-cluster links (the blue link in the network of clusters). This network of clusters then is mapped into the server room where an inter-cluster link is equivalent to an inter-cabinet cable with Diagonal Cabling or severals normalized cables if applying Manhattan Cabling. In figure 2.7, two red cables are normalized from the earlier mentioned inter-cluster link.

[2] A $4 \times 3 \times 3$ grid pluses two extra blue links in the logical topology



(a) Minimal Dimension Ordering Routing (DOR) (b) Non-minimal routing

Figure 2.8: Examples of routing algorithm on 4×4 torus.

2.3 Routing algorithm

2.3.1 What is Routing

Routing involves selecting a path from a source node to a destination node in a particular topology. A well-designed routing algorithm keeps the path lengths as short as possible while balances the load of links in the networks. Moreover, a routing algorithm also needs to provide the fault tolerance ability when one or several links fail.

Regarding the network performance, the fact is that a short routing path helps to reduce the overall latency of a message. If all the routing paths are shortest paths, the routing algorithm is "minimal routing", otherwise it refers to "non-minimal routing". Figure 2.8 shows an example of two different routing algorithm on a 4×4 Torus network. The routing path from source node 00 to destination node 21 in Figure 2.8.(a) is called minimal routing because every package is routed via the shortest path. By contrast, the route in Figure 2.8 is a non-minimal path.

2.3.2 Routing Implementation

A *routing* is a mechanism that can transfer a message (packets of information) from any source nodes to any destination nodes of the network. Generally, routing algorithms are classified into three types based on their different relations [1].

1. The $N(\text{source}) \times N(\text{destination}) \mapsto P$ routing relation (all-at-once), where N is the

Table 2.1: Dimension ordering routing table for 4×4 torus network in figure 2.8.

To	From						
	00	01	02	03	10	...	33
00	-	W	W	E	N	...	E
01	E	-	W	E | W	E	...	E | W
02	E | W	E	-	W	E | W	...	W
03	E	E | W	E	-	W	...	S
10	S	W	E | W	E	-	...	E
...	...	...	...	...	...	...	...
33	E	E	E	S	W		-

node set and P is the path set.

2. The $C \times N \mapsto C$ routing relation, which considers the input port of the packet to find the output channel C.

3. The $N \times N \mapsto C$ routing relation, which only takes into account the current and destination nodes to determine the output channel.

All of the three types can be deployed using a routing table. A routing table stores the value of the relations, e.g., the identifier of the output port on the routing path, that is indexed by the destination identifier. To forward a packet, the switch has to compare the destination identifier of the packet to all the indexes to get the routing information (the "routing lookup") and thus choose an appropriate route (the "routing decision"). With source routing, the routing lookup and decision are made only at the source node. Then all the routing path information is packed into the packet header for the later fast forwarding at an intermediate switch without doing lookup again. Typically, a routing table of source routing implements the first relation.

Another table-based approach is distributed routing that refers the second and third relations. By considering the information of the current nodes, e.g., an input port or its identifier, rather than the source node, it reduces significantly the size of the routing table at each switch. Following the above taxonomy, the number of entries in each switch is at most $N \times N \times C$ in the first type, $N \times C$ in the second type, and N in the third type, respectively. The first routing relation can express an arbitrary routing algorithm but its

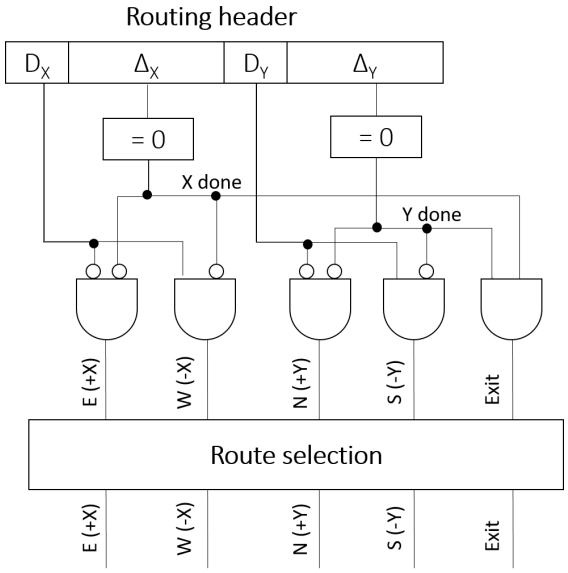


Figure 2.9: An example of algorithmic routing logic for Torus (figure copyright by [1]).

routing table becomes large. By contrast, the third routing relation requires the smallest number of routing table entries. This is commonly used in commercial interconnection networks, such as InfiniBand and Ethernet.

Table 2.1 expresses the routing table of dimension ordering routing for 4×4 torus network in Figure 2.8. For the source routing, based on this table, a routing path is computed at the source node and attached to the message header. For the distributed routing, e.g., with the $N \times N \mapsto C$ routing relation, this table is distributed and stored separately at each node. That is, each node i stores only the column i for performing table loop up and routing decision function.

Another way to deploy a routing algorithm is to instantly compute the route by using a combinational logic circuit at each switch instead of looking up from a routing table. This approach is known as “algorithmic routing", custom routing, or finite-state machine routing. Algorithmic routing is often more efficient in terms of both area and speed than table-based routing [1] [8]. However, it requires the routing algorithm logic simple and nature to be easily effectively implemented as a logic circuit. Continuing with the example of routing in Torus network, Figure 2.9 illustrates a simple logic circuit for Dimension Ordering Routing (the figure is redrawn from the one in [1]). The algorithm requires the relative address stored in routing header, i.e., the distance, routing direction in X and Y dimension δ_x, D_x, δ_y and D_y, respectively. This information then is used for

determining the routing decision by using a set of AND gate components. Through these logic modules, the signals of routing to the East (E), West (W), North (N), South (S) or Exit (receipt the message) are generated.

2.4 Point-to-point reliable links

2.4.1 Overview

Recently, demands for increasing the speed of the data access in HPC systems drive the development of high-speed optical fiber communication to become one of the critical issues. The Institute of Electrical and Electronics Engineers (IEEE) standard 802.3ba specified 100 Gb/s Ethernet standard in 2010 [40]. Subsequently, the 400 Gb/s Ethernet has been studied by IEEE 802.3bs Task Force from 2014 and planned to release officially in 2017 with a major objective of design the 500m single mode fiber (SMF) to support the intra-datacenter connection requirement [41]. In this study, we assume to use multi-level modulation formats such as Pulse Amplitude Modulation (PAM) and Quadrature Amplitude Modulation (QAM) [42]. The data bits are mapped into a series of signal symbols in which each symbol carries two or more bits. For example, when using 16-QAM, 4-bit per symbol can be transferred instead of 1-bit per symbol as in the conventional optical communication that implements on/off keying (OOK) or binary phase-shift keying (BPSK). Therefore, four times of bandwidth can be achieved with the same rate of the signal. However, multi-level modulations are more sensitive to noise than on/off keying. Thus, in order to design an error-free communication system, Error Control Codings (ECC), e.g., Forward Error Correction (FEC), are proposed beside the CRC error detection code at the Media Access Control (MAC) layer.

Figure 2.10 shows the overview of network physical layer specified by IEEE Standard for 100 Gigabit Ethernet [40]. The Physical Medium Dependent (PMD) sublayer translates the encoded bits of data to and from signals suitable for the specified medium, i.e., the modulations. The Physical Medium Attachment (PMA) sublayer aims at adapting the data formatted in Physical Coding Sublayer (PCS) to an appropriate number of physical lanes of PMD sublayer. The PCS, also known as line coding, formats the data with control characters that help to determine when a packet has been established. PCS is also used for compensating any rate difference between the upper level, e.g, MAC, and PMA through the insertion or deletion of idle control characters. Depend on

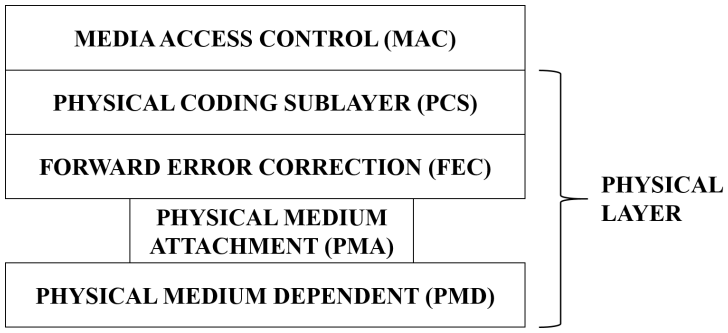


Figure 2.10: Overview of physical layer in 100 Gigabit Ethernet.

the physical type, the FEC sublayer which implements an error correction code for the purpose of error-free communication is optional or omitted.

2.4.2 Forward Error Correction (FEC)

The main idea behind FEC is to add some redundant bits to the original message that is used by receivers to check the correctness of the transmitted message and recover it to the correct data. A challenge of FEC design is to implement an error correction code with low redundancy and high capacity of errors correction. For the low-redundancy target, the amount of the redundant bits is expressed by the Code Rate R and Overhead ρ. Typically, $R = \frac{k}{n}$ and $\rho = \frac{1}{R} - 1$ for every k bits of data and $(n-k)$ bits of checksum over n transferred bits (n also known as the length of FEC's codewords). In addition, the error correction capacity is characterized through the post-FEC Bit Error Rate (post-FEC BER), i.e., the average probability that an error bit is received after passing through FEC at a given level of Signal-To-Noise-Ratio (SNR) E_b/N_0[3]. Specifically, an FEC-implemented system achieves a smaller BER for a given SNR, and conversely, a smaller SNR is required for a given BER, than those in an uncoded system using the same modulation. This reduction of SNR is referred as the Coding Gain. In other words, designing a correction code for FEC is to consider both low coding overhead and high coding gain.

According to the survey in [43, 44], three generations of FEC have been proposed with various different targeted coding overhead and coding gain. For example, the first generation, i.e., hard decision block codes such as Hamming, BCH, and Reed Solomon

[3]SNR is the ratio of signal energy per bit to noise power density per Hertz. Typically, the unit of the SNR is expressed by using the logarithmic scale (decibel or dB)

Table 2.2: Ethernet coding

Ethernet Standard	Line/PCS Code	FEC	Remarks
100BASE-X/T4	4B/5B, 8B/6T	NO	
1000BASE-X	8B/10B	NO	
10GBASE-LX4/CX4			
10GBASE-SR/LR/ER	64B/66B	RS(255,239)	Clause 85 (802.3ba)
40GBASE-CR4			
100GBASE-CR10			
100GBASE-CR4	64B/66B with 256B/257B Transcode	RS(528,514)	Clause 92 (802.3bj)
100GBASE-SR4			Clause 95 (802.3bm)
>400G	256B/257B Directcode	RS(528,514)	Expectedly in 802.3bs

(RS), achieves at most 8-dB of coding gain with a coding overhead no greater than 7 percents at a 10^{-15} output BER. Without changing the targeted coding overhead, the second generation achieves better coding gain, i.e., 8-9 dB, with the use of hard decision concatenated codes alongside interleaving and convolutional-decoding techniques. By contrast, the third generation such as Turbo or Low-Density Parity-Check (LDPC) achieves much higher coding gain, i.e., 10-11 dB, by accepting a higher overhead, i.e., 20 percents that leads to lower throughput.

Although the second and the third FEC generations have better correcting performance, some of the first generation codes such as Reed-Solomon are still suitable for short-haul optical high-speed networks due to their relatively simple implementation, and relative short block length [44]. For instance, RS(255,239) is standardized for FEC in 40GBASE-CR4 Ethernet or 100GBASE-CR10 Ethernet [40] and RS(528,514) is implemented in 100GBASE-CR4 [45], or 100GBASE-SR4 [40]. Recently, RS(544,514) and RS(528,514) are also suggested to be implemented by 400 Gb/s Ethernet Task Force Group [46] [41]. Table 2.2 gives a brief overview of Error Control Coding that used in Ethernet Standard. Further introductions to high-speed optical cable design and surveys on FECs are reviewed in [42, 47, 44, 43].

3

Network Topology

This chapter introduces our cable-geometric approach in design network topology for high-scale HPC system in which network diameter is not the only major requirement. The challenges or design goals are to reduce the network latency by achieving good trade-offs between switch latency and cable latency (time of flight). Our approach is based on the idea of random small-world network models for a small-diameter but exploits its effect in a deterministic manner. That is a non-random approach with well-designed topological properties that helps to easily map into a server room with a short total cable length.

In this chapter, we propose short-diameter short-cable length network topology called Distributed Shortcut Networks (DSNs). We then show the analysis of our proposed topology and discuss its extension to support incremental expandability requirement.

3.1 Introduction

3.1.1 Problem Statement

Designing low-latency interconnection networks is a main concern in highly parallel computers as they become larger, such as 3M cores for Tianhe-2 [3]. 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 [10] and about 100 ns even in a commodity InfiniBand QDR switch which uses a forwarding table. 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/data centers in 2018 [48, 10].

Low-latency network topology 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 [10] and K-Computer [15]. 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 [49], 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 [36] 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 [50, 37], 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 topology that has low-cable delays when mapped onto a floor plan, and (2) a custom routing algorithm that does not require routing-table implementation at switches.

In this chapter, we only focus on the first challenge, i.e., low-degree non-random network topology design. We discuss the custom routing algorithm on our proposal network topology in Chapter 4.

3.1.2 Related Work

As mentioned, 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 [49]. 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 has a diameter no less than $\theta(\log n)$ asymptotically [28, 35, 24]. 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) [24] 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 [50]. This fact keeps the cable delay relatively large compared to the switch delay.

Figure 3.1 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 [35], Skywalks which consider the cable delay [37], and Slimfly as a reference of high-degree topologies. The legend in the figure

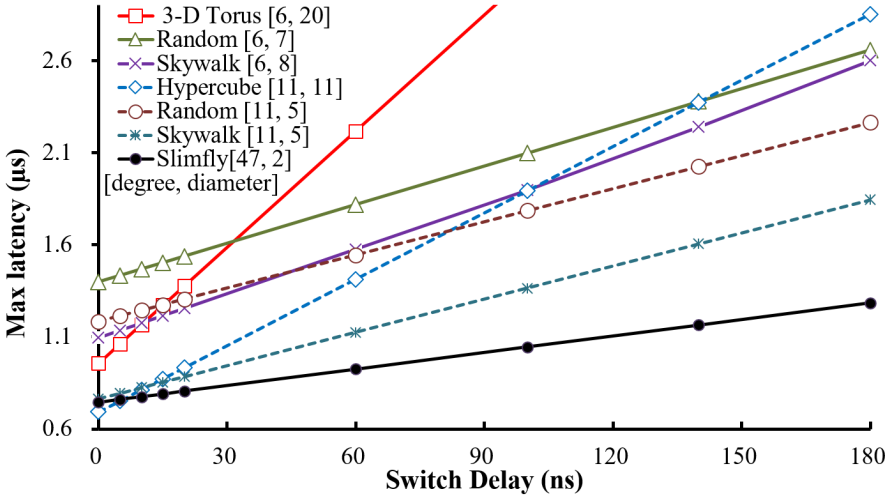


Figure 3.1: Maximum latency along the shortest-hop paths vs. switch delay in 2,048-switch networks (except for Slimfly, which has 1,922 switches). Links between switches and compute nodes are not considered. The legend indicates the degree and the diameter.

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. 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 in the 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 the low 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 Figure 3.1). 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 has clear advantages in saving cables and in having a custom routing.

3.2 Distributed Shortcut Networks and Properties

3.2.1 Our Basic Approach

Our approach is learned from observing small-world network model, i.e., the Kleinberg's model [26]. The small-world model has been introduced to explain the structures of popular real-world large-scale networks [23]. 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) [24], (uniform) random shortcuts are added to a ring of nodes. DLN-2^x [24] and Chord-like topologies [51] have logarithmic-diameter that can be formed by setting up each node with a collection of various-range shortcut links, typically links jump 2^x nodes when the nodes are placed on a ring [1]. 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 [28][27]. 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, besides 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

[1] The DLN-x, Distributed Loop Network of degree x [24], consists of n vertices arranged in a ring and additional shortcuts between vertices i and j such that $j = i + \lfloor n/2^k \rfloor \mod n$ for $k = 1, \ldots, x-2$.

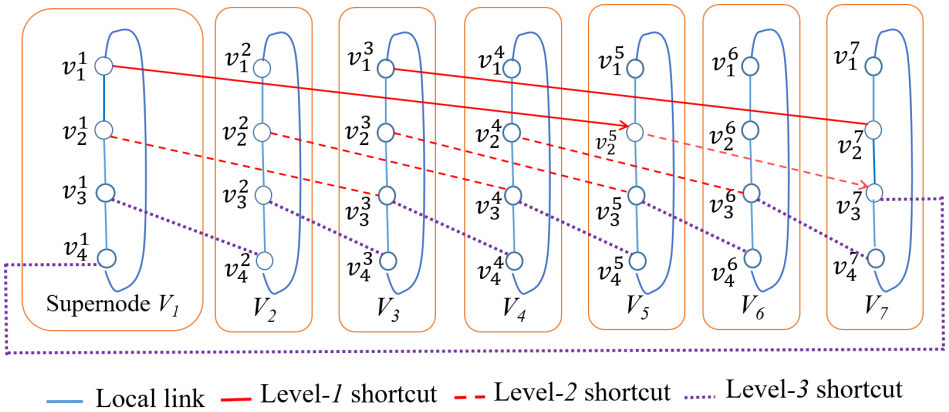


Figure 3.2: 1-D DSN with 28 nodes arranged into a ring of 7 supernodes. Some links are omitted.

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 the following we describe in detail two specific design proposals: one is for a ring and one is for a 2-D grid as the base graph. The reason we include the ring is that 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).

3.2.2 Ring-based DSN Topology (1-D DSN)

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 or ring-based DSN. The basic idea of our ring-based DSN topology is that virtually see it as a smaller ring of n supernodes, and then add $z \le p = \lceil \log n \rceil$ shortcuts for each supernode by using the distance-halving principle. We assume a ring of regular nodes is grouped into separate supernodes V_i, $i = 1 \ldots n$, which have roughly the same size $m \ge p$. For each supernode V_i, we can distribute z shortcuts for regular nodes by picking up a sequence of z nodes such that a node v_j^i, $j = 1 \ldots z$, is connected to the node v_{j+1}^i (called local link) and to the node v_{j+1}^k (called level-j shortcut), where $k = i + 2^{p-j}$

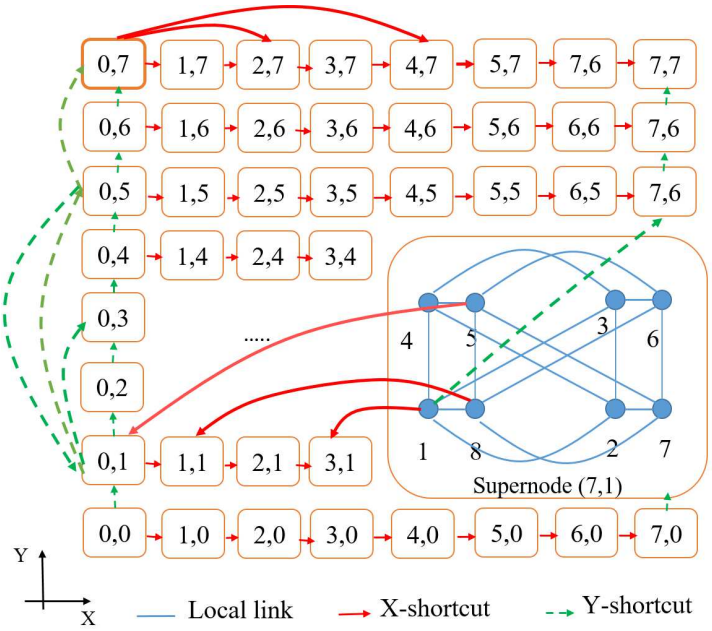


Figure 3.3: 2-D DSN with 512 nodes arranged into grid of 8x8 supernodes. Each supernode includes 8 regular nodes. Some links are omitted.

mod 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 a supernode V_i.

Figure 3.2 illustrates an example of our ring-based topology for the case of 28 nodes grouped into rings of $n = 7$ supernodes $V_1, V_2, \ldots V_7$, which have the same size, $m = 4$. Each supernode has $z = p = 3$ types of shortcuts with the lengths of 1, 2, and 4 supernodes on the ring.

3.2.3 Grid-based DSN Topology (2-D DSN)

Below we describe the construction of grid-based DSN which is more practical for deployment in a machine room where supernodes can be naturally deployed as cabinets.

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

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 Ring-based DSN per each grid row or each column.

Figure 3.3 presents an example of our grid-based topology for the case of 512 nodes arranged into a grid of 8×8 supernodes (x, y), where $0 \le x \le 7$ and $0 \le y \le 7$. Each supernode includes $m = 8$ regular nodes with an internal structure as shown in the supernode (7,1). This structure has been designed in order to achieve a diameter $\Delta = 2$ with the network degree no greater than $\delta = 4$. We then construct the shortcuts between supernodes where each row (X dimension) and each column (Y dimension) form a ring-based DSN. We consider the X-shortcut / Y-shortcut (the solid red lines / the dashed green lines in 3.3) for connecting supernodes that is in the same row/column, respectively. We configure each supernode with $z = \log X = \log Y = \log 8 = 3$ different types of shortcuts, i.e., lengths of 1, 2, and 4. For instance, the supernode (7,1) has three outgoing X-shorcuts (and also three incoming X-shortcuts) targeting to supernode (3,1), (1,1), and (0,1), respectively. These shortcuts are subsequently distributed for the regular nodes inside each supernode that helps to maintain a low degree of each node, i.e., each node has at most one outgoing and one incoming shortcut for each dimension.

Let us now investigate the degrees and diameters. Given a DSN network G, 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 supernodes. The degrees in G_L are upper bounded by δ. Thus, the facts below are straightforward.

Fact 1 *For 1-D DSN defined above* $p = \lceil \log n \rceil$ *(i.e., n supernodes with* $n\lceil \log n \rceil$ *regular nodes), the graph diameter is at most* $\log n + 2\Delta$ *if each subgraph of G induced by a supernode is connected and has a diameter* Δ.

Fact 2 *For the 2-D DSN defined above with* $z = \lfloor \log X \rfloor = \lfloor \log Y \rfloor$, *each node has at most one outgoing and one incoming shortcut for each dimension. The average degree of the graph is at most* $\delta + 4$. *For a supernode of size m, the average degree of the graph is* $\le \delta + 2\frac{z}{m}$. *The graph diameter is at most* $\log n + 3\Delta + 2$.

Fact 1 and Fact 2 show that the internal structure of supernodes effects the degree and the diameter of the whole network. 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 [2], and it needs to be flexible enough for easily being expanded. We recommend and discuss a few choices in the Section 3.2.4

3.2.4 Our Concrete Topologies

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: (i) a grid for setting the supernode positions, (ii) locality for setting the supernode internal structure, (iii) 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.

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 Figure 3.3, we achieve diameter 2.

However, fact 2 shows that an increment by 1 in Δ causes an increment by 3 in the diameter of the whole 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:

[2] as discussed in Chapter 4

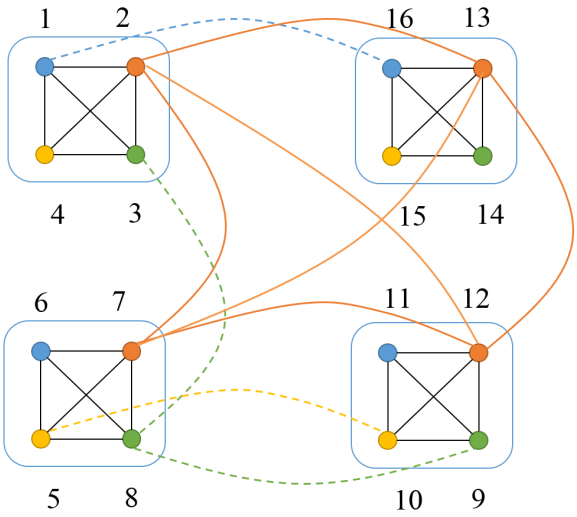


Figure 3.4: The internal structure of supernodes with 16 nodes arranged in CoC-16: a *complete* graph of 4 *complete* subgraphs of size 4. Since dashed inter-subgraph links are in the same form as solid links, some links are omitted.

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 CoC (*Complete of Complete*), has degree $(m_s - 1) + (m_n - 1)$. Figure 3.4 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.

3.3 Physical Layout of DSN

In this section, we consider the physical layout of our 2-D DSNs topologies with the baseline layout mentioned in Section 2.2.1. The main target of this work is to (1) minimize the number of inter-cabinet cables by operating the clustering function and (2) to minimize the total cable length by solving the mapping function optimally.

We learn from the approach of manually clustering groups of nodes into cabinets as in Flattened butterfly [33], Dragonfly [34] or Slimfly [36]. For our DSN topology, a supernode is considered to be implemented as a cabinet in order to keep a small number of inter-cabinet cables, i.e., the number of shortcuts links. In addition, following our prior survey in [52], we found that the existing of diagonal cables in a system that uses

Manhattan Cabling leads to a considerable increase of total cable length, e.g., up to 10% when the network size is set to 512 switches [52]. This fact implies that reducing or avoiding the diagonal cables is also an important concern for physical layout design. Interestedly, our 2-D DSN topologies can support this required avoidance because no diagonal shortcut appears in our design, i.e., we arrange X-shortcuts with horizontal pathways and Y-shortcuts with vertical pathways.

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 << 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 $\leq X$ for a grid-based DSN, i.e., $z = \log X$ outgoing shortcuts with the length of $2^i, 0 \leq i \leq z-1$; then the total length is $\sum_{i=0}^{\log X - 1} 2^i \leq X$. Similarly, per each grid column, the length is $\leq Y$ for a given grid-based DSN. Thus, we have the 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 3 *Let n is the number of supernodes. For a $X \times Y$ grid-based DSN, the total shortcut length is at most $n(X+Y)$ and the average shortcut length is at most $\frac{X+Y}{2z}$, respectively.*

In a grid-based random model for creating shortcuts (such as the one in[37]), 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 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 4 *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, 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 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.*

Proof.
In the random network based on a $N \times N$ grid, let $p_{A;B}$ is the probability of the event that having an inter-cabinet cable between a cabinet A and B, $p_{A;B} = p = \frac{1}{N^2}$. Let X is the random variable present the length of a cable between any two cabinets. Clearly,
Probability of the event that a cable has length 0 is $\Pr(X = 0) = n \times p$.
Probability of the event that a cable has length 1 is $\Pr(X = 1) = 2 \times (N-1) \times p$.
Probability of the event that a cable has length k is $\Pr(X = k) = 2 \times (N-k) \times p$.
We then consider the average length of cables as the expectation of X:

$$
\begin{aligned}
&E(X) = \sum_{k=0}^{N-1} k \times \Pr(X = k) \\
&\Rightarrow E(X) = \sum_{k=0}^{N-1} (k \times 2 \times (N-k) \times p) \\
&\Rightarrow E(X) = \frac{2}{N^2} \sum_{k=0}^{N-1} (k \times (N-k)) \\
&\Rightarrow E(X) = \frac{2}{N^2} \sum_{k=0}^{N-1} (Nk - k^2) \\
&\Rightarrow E(X) = \frac{2}{N^2} \times (N \times \frac{(N-1) \times N}{2} - \frac{(N-1) \times N \times (2N-1)}{6}) \\
&\Rightarrow E(X) = \frac{(N-1)(N+1)}{3N} \leq \frac{N}{3}
\end{aligned}
\tag{3.1}
$$

For our grid-based DSN, the average inter-cabinet cable length is at most $\frac{N+N}{2z} = \frac{N}{z}$ as mentioned in Fact 3. Thus the average inter-cabinet length of DSN is at most about $\frac{N}{z} \times \frac{3}{N} = \frac{3}{z}$ of the random counterpart. □

3.4 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 [24]

and Skywalk [37] 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 latency by considering their floor plans in a machine room to evaluate them from a practical perspective.

Fact 2 showed that the average degree of our proposed topologies is around 6 when the network has 128 to 16384 nodes. Thus, we compare them with the same-degree, same-sized counterparts. We choose 3-D Torus, RR-6 [24][3], and Skywalk-4-2 [37][4] to be the representative of torus, RSN, and Skywalk, respectively. Also, Dragonfly, Slimfly is considered as a reference to state of the art high-degree topologies in this comparison. For Dragonfly topologies, we follow the description in [34] with the fixing of group's size, i.e., 8 nodes per group for all network sizes with a high degree from 9 to 263 [5]. We use the Slimfly topologies provided in [36] with different sizes (from 98 to 5618 nodes) and degree (from 11 to 79).

3.4.1 Graph Analysis

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. Figures 3.5(a) and 3.5(b) show the diameter and the average shortest path length (ASPL) of each topology. Smaller values are considered to be better.

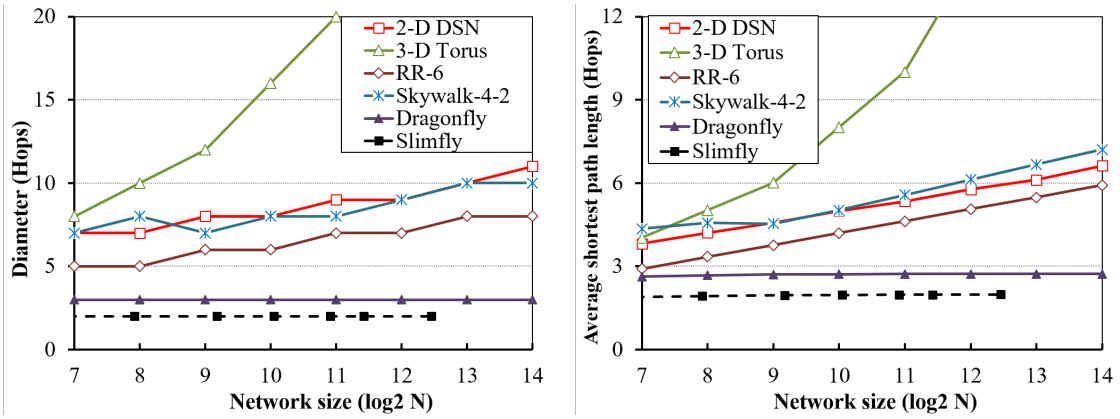
In most network sizes, RR-6 achieves the smallest diameter and ASPL whereas 3-D Torus leads to the largest. 2-D DSN has the same result as Skywalk-4-2, which is much smaller than that of 3-D Torus and is considered as a good achievement. Specifically, when compared to 3-D Torus, the diameter and the ASPL of 2-D DSN decreases up to 78.8% and 74.5%, respectively. These values are not far from the smallest value, e.g., 10.5% 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

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

[4] 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 [37]. Each node has d_i intra-cabinet links and d_o inter-cabinet links.

[5] Typically, a n nodes Dragonfly with c nodes per a group, i.e., $N = \frac{n}{c}$ groups, is constructed by a fully connection between nodes in the same group. In addition, each node has $\frac{N}{c}$ inter-group links for connection between groups. As instance, for network of 256 nodes, 32 groups of size 8 is generated. Each node has degree 11 with 7 intra-group links and 4 inter-group links.



(a) Network Diameter

(b) Average shortest path length

Figure 3.5: Diameter, average shortest path length vs. network size. Dragonfly and Slimfly are considered as references to high-degree topologies.

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.

Network Bisection Bandwidth

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 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 [53].

Figure 3.6 presents the min-cuts of the compared topologies with different network size. 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

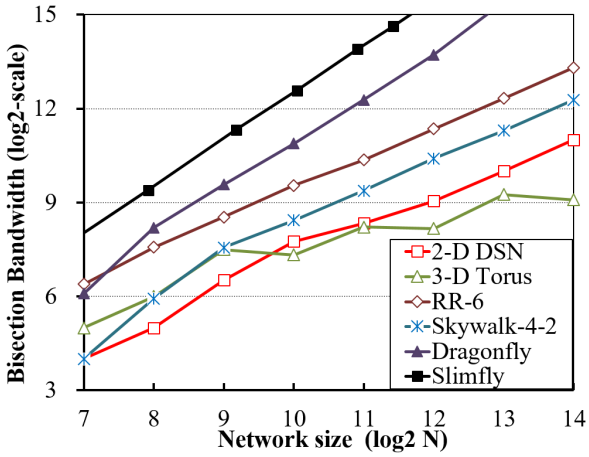


Figure 3.6: Network bisection bandwidth vs. network size. Dragonfly and Slimfly are considered as references to high-degree topologies.

good performance in terms of throughput.

3.4.2 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 [32]. 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 the physical location (x,y) includes switches with logical index $(x,y,*)$. Since a supernode of our proposed topologies includes 8 nodes, we assume that each cabinet has 8 switches.

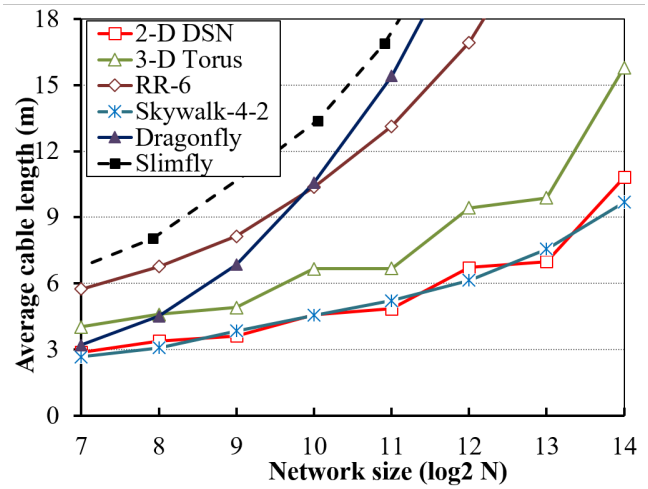


Figure 3.7: Average cable length vs. network size in 8-switches/cabinet networks of 40-ns switches and 40-Gbps 5-ns/m switch-to-switch cables. Dragonfly and Slimfly are considered as references to high-degree topologies.

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 [33]. 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 Figure 3.7. 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., 4 intra-cabinet and 2 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% when compared to 3-D Torus and RR-6, respectively. Therefore, DSNs are non-random topology optimized for short cable length, which leads to low cost and low end-to-end latency.

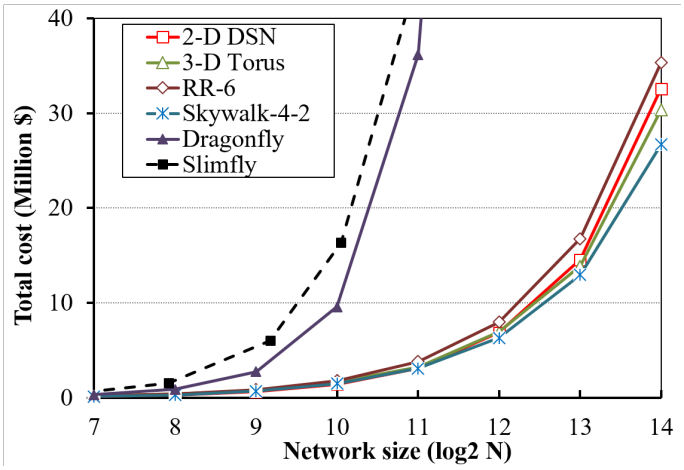
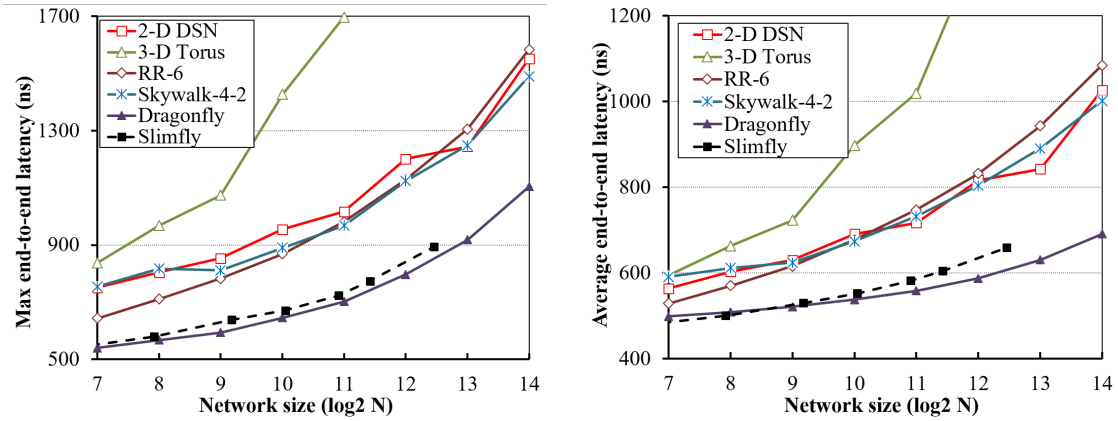


Figure 3.8: Total cost vs. network size in 8-switches/cabinet networks of 40-ns switches and 40-Gbps 5-ns/m switch-to-switch cables. Dragonfly and Slimfly are considered as references to high-degree topologies.

Network cost

We now provide a further cost comparison of the above topologies. We use the cost model mentioned in [36], 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 [36]. In this study, we set the cable bandwidth to 40 Gbps.

Figure 3.8 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 4096. 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 1024, 2048, and 4096 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. 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 achieves low cost in the low-degree topology design.



(a) Maximum latency vs. network size

(b) Average latency vs. network size

Figure 3.9: Maximum and average latency of lowest-latency paths vs. network size in 8-switches/cabinet networks of 40-ns switches and 40-Gbps 5-ns/m switch-to-switch cables. Dragonfly and Slimfly are considered as references to high-degree topologies.

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 Figure 3.9. 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% lower than those of the 3-D Torus in the networks of 2048 and 16384 nodes, respectively. When compared to RR-6, we see that our 2-D DSN almost always lead to lower average latencies (up to 10%) 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 [37, 24]. Therefore, we say that our 2-D DSN are low-latency low-degree non-random topologies.

This is expected that the communication latency of all the low-degree network topologies, i.e., 3-D Torus, RR-6, Skywalk and 2-D DSN with degree-6, are higher than

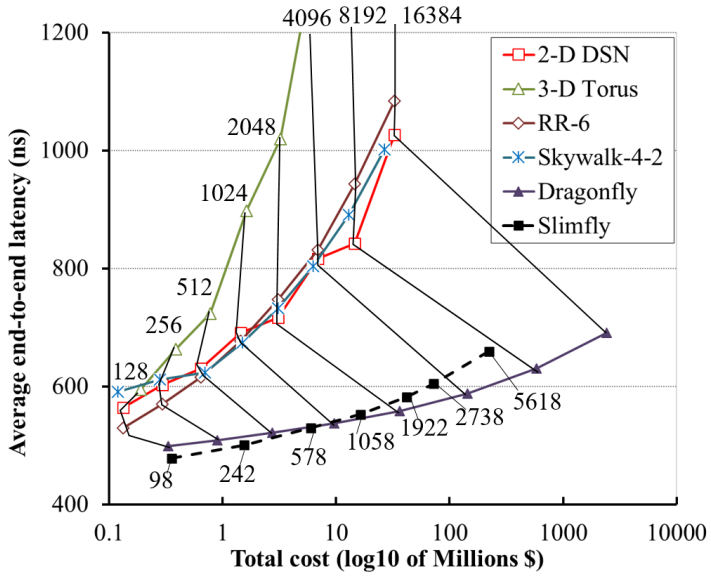


Figure 3.10: Average latency of lowest-latency paths vs. total cost. The numbers indicates the network sizes. All the network topologies have 2^k size except for Slimfly.

the high-degree network, e.g., Dragonfly with the degrees from 9 to 263 and Slimfly with the degrees from 11 to 79. However, it is unreasonable if we do not take the network cost into account. In this study, we also consider the trade-off between network latency and cost. The average network latency versus total cost of various interconnection networks is shown in Figure 3.10. The results show three groups of network topology design including: (1) low-degree non-random network, e.g., 3-D Torus, with low-cost but high-latency (2) high-degree non-random network such as Dragonfly, Slimfly with very low-latency but these topologies require a huge cost, and (3) low-degree random network such as RR-6 and Skywalk that are more balanced in cost and latency. Our proposed topology lays into the third group. Although our 2-D DSN does not reach the best network latency,i.e., higher than the second group, we achieve a much lower cost. Thus, we state that our 2-D DSN provides a good trade-off between network performance and cost.

There is an interesting point is that although our DSN belongs to the third group but, is not based on the random links. This helps to not only accomplish a lower cost as mentioned but also more easily in expanding the network size and implementing the routing. In the next section, we discuss the incremental expandability properties of our network and. We also present the relationship between the structured non-random design and the routing (Chapter 4).

3.5 DSN-F with Incremental Expandability

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 their number of cores by 100%, and 87.76%, respectively [3]. In addition, commercial data centers have also been expanded. Facebook's data center server population had been doubled from November 2009 by June 2010 [35].

In this context, we target network-topology expandable property with two main characteristics:

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

In this section, we discuss how to refine our DSN topologies for achieving the incremental expandability requirement while still maintain low degree and logarithmic diameter properties. We named this refinement as DSN-F.

3.5.1 Background

The survey in [35] indicated that the conventional tree-based topologies such as Fat-tree could not meet the expansion requirement because of their coarse design size. For example, a system with Fat-tree only has size 3456, 8192, 27648 if 24, 32, and 48 ports switches are used, respectively [35]. 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.

Low-degree 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 a short cable length and a constant switch degree. Koibuchi et al introduced random shortcut network [24] aimed at these properties but did not discuss the expandability. The JellyFish work [35] 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.

To our best knowledge, in this context, we do not have an efficient method to incrementally add nodes to a small-world topology. This is our main challenge in this study for extending our DSN topology.

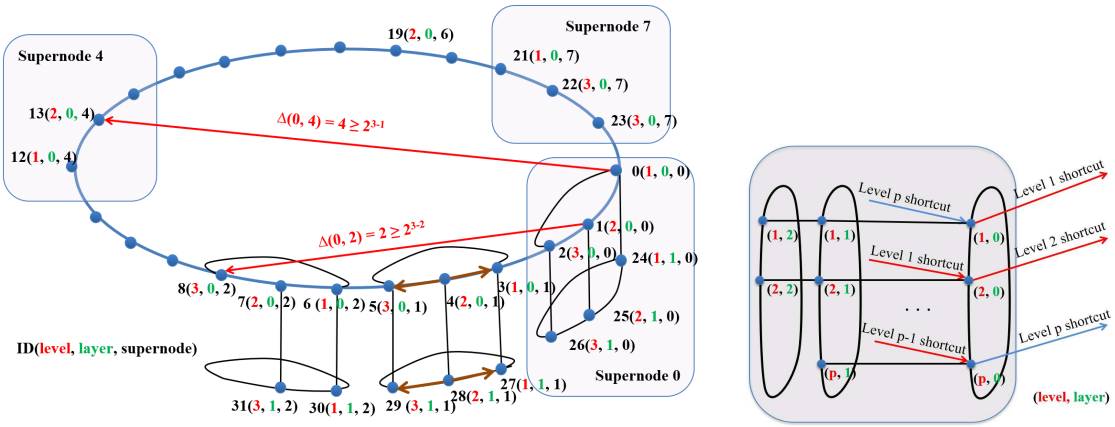
3.5.2 Topology Description

Let us firstly remind the notations used to describe our topologies. Consider a ring-based DSN topology of n supernodes, each of which is a group of m nodes. Each supernode has $z \leq p = \lceil \log n \rceil$ different range shortcuts, e.g., the length of the level-i shortcut is $l_i = 2^{p-i}$ for $i = 1 \ldots z$. Per each supernode, the local links are arranged to form a subgraph of diameter $\leq$ a given Δ and degrees $\leq$ a given δ. For expandability purpose, we assume that the number of supernodes is fixed to be 2^p. Now, the topology is a ring of 2^p supernodes that include at least p nodes and p outgoing shortcuts.

Inside a supernode, we arrange the 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 supernode, we can easily add nodes to an existing topology without rewiring the shortcuts (long and complicated cables) by pushing nodes into the layer of supernodes. Furthermore, if the number of additional nodes grows high, we propose a method to transform the DSN topology from the view of p-node-layer to $(p+1)$-node-layer, 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.

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 $p \times 2^p \leq n < (p+1) \times 2^{(p+1)}$



(a) Topology DSN-F of 32 nodes (b) Structure inside a supernode

Figure 3.11: 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.

denotes the maximum level of all the nodes. The level of the node i is $l = i \bmod p + 1$.

- **Node layer** k with $0 \le k \le \lceil n/N \rceil$, 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 $\lfloor i/N \rfloor = k$.

- **Supernode ID** s with $0 \le s \le 2^p - 1$. Nodes are grouped into 2^p supernodes identified by the supernode ID s. A supernode s is a group of nodes i with $i/p \bmod 2^p = s$. Since $0 \le i \le n-1$ and $n \ge p \times 2^p$, each supernode has at least p adjacent nodes.

Hereafter we refer to a node as $\{l,k,s\}$. 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 $\{(l-1 \bmod p),k,s\}$ and $\{(l+1 \bmod p),k,s\}$, respectively.

- Each node $\{l,k,s\}$ with $k \ge 1$ 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.

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 3.11 illustrates our topology construction in detail. Figure 3.11(a) presents a full network for the case of $n = 32$ and $p = 3$. In this case, the topology is a ring of $2^3 = 8$ 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 $2^{p-l} = 2^{3-1}$.

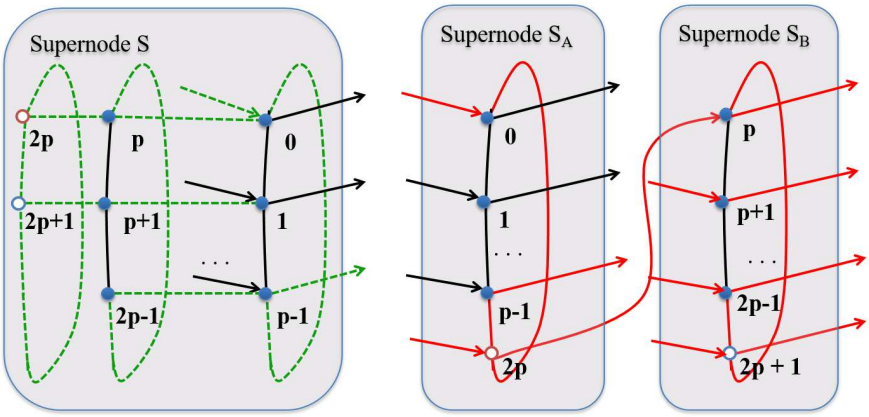
The structure inside a full supernode with $2p+2$ nodes arranged into 3 layers is presented in Figure 3.11(b). The node 0 with $\{1,0,0\}$ has two links, *Local_Pred* and *Local_Succ*, which are connected to nodes 2 with $\{3,0,0\}$ and node 1 with $\{2,0,0\}$. It also linked to the node 24 with $\{1,1,0\}$ by a *Layer_Link*.

3.5.3 Incremental Expansion Method

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

The number of shortcuts 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 of 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 $r = n \bmod N$ nodes in it ($r < p$). If the layer-$(K-1)$ has full p nodes, we will first 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



(a) Before splitting (b) After splitting

Figure 3.12: Illustration of our transformation method. Red lines are additional links. Green lines are removed links.

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.

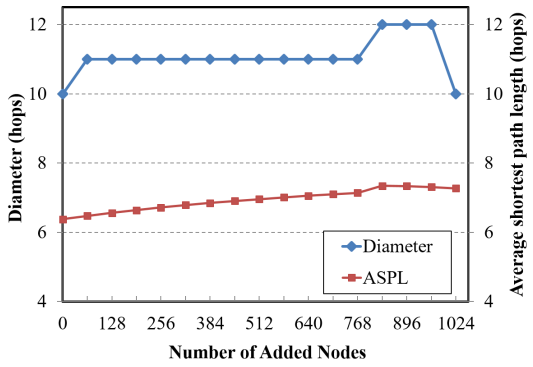
Additionally, in case of the number of additional nodes grows high, we propose a method to transform the structure of supernodes from p-node-layer to $(p+1)$-node-layer, with an acceptable number of cables rewired. The main idea is (i) we add nodes to the topology using the above-mentioned method; and (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 $2p+1$ to identify those nodes, as shown in Fig.3.12(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

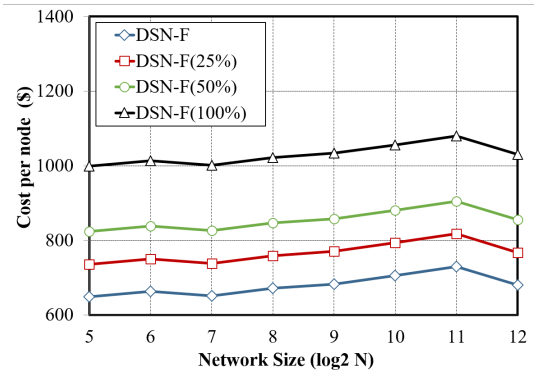
layer-3, level-1. Clearly, now this node can be considered as a level-$(p+1)$ node of S_A. Similarly,
remove/add links following the steps below:

- All the Layer_Links (p links between layer-0 and layer-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 remove one link and add two links inside each supernode.
- In the supernode 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 supernode $(S+1)_A$.
  - Add a new Succ link from the node level $p+1$ of S_A to the node level 1 of S_B.
- In the 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 $(S+2^{p+1-l})_B$.
  - Add a new Succ link from the node level $p+1$ of S_B to the node level 1 of $(S+1)_A$.

Figure 3.12 illustrates our transformation method. Fig.3.12(a) shows a supernode S before splitting, and Fig.3.12(b) presents two smaller supernodes after splitting. Clearly, the inside structure of the supernode 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.



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



(b) Cost per node vs. network size

Figure 3.13: Degradation of the network diameter and cost for DSN-F.

3.5.4 Topology Analysis

In this section, we analyze the graph properties of the ring-based DSN-F topologies generated by the incremental network expansion. Our target is to learn how much the degradation of the network efficiency, i.e., diameter, and how wasting of the resource, e.g., establishing cost.

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

Figure 3.13(a) 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 of $x = 768$. 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 $x = 768$.

Network Cost

We also estimate the cost per node of DSN-F network as the number of nodes increases. We use the cost modal mentioned in [36], which considers 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 3.13(b) plots the cost per node vs. the number of nodes N of networks with DSN-F. Spare ports are generally needed for network expansion purposes. We thus evaluate the cost per node of the DSN-F network for the cases that support 25, 50, and 100% of expandability 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.

3.6 Summary

In this chapter, we introduce our cable-geometric approach in design network topology for high-scale HPC system with the consideration the trade-offs between switch latency and cable latency. We propose the Distributed Shortcut Network (DSN) and analyze its both topological properties such as degree, diameter and physical layout such as cable length. We show that by exploiting the small world network model in a deterministic manner, our network topology has a low degree, a logarithmic diameter and significant shorter cable length in the comparison with the same degree same size counter path topology. Such kind of network properties makes DSN has a good communication latency. We also illustrate the capability of customized our network topology for incremental expansion demand.

4

Custom Routing

"*Whereas a topology determines the ideal performance of a network, routing is one of the two key factors that determine how much of this potential is realized*" [1][1]. As discussed in Chapter 3, the increase of network size brings new challenges in low-latency network design to develop (1) a low-degree non-random network topology that has a low-cable delay when mapped onto a floor plan, and (2) a custom routing algorithm that does not require routing-table implementation at switches. We have just introduced our network topology with a logarithmic diameter and a short cable length in Chapter 3. In this chapter, we present how to route a message without the use of routing table at each switch on this topology (custom routing). Besides achieving the low switch delay by the avoidance of large routing table, our target is to keep routing path lengths as short as possible.

[1] Flow control is the other factor. In this dissertation, because the network latency is our main target, we skip the consideration of flow control (switching technique).

4.1 Introduction

4.1.1 Problem Statement

Network topologies and their routing algorithms in interconnection networks have been widely studied for decades. A well-known combination is k-ary n-cube with dimension-order routing or Duato's protocol [1]. It is frequently used in supercomputers such as BlueGene/L Anton-2 [10] or Cray XT5 [11]. It requires $\frac{k \times n}{4}$ hops on average for the inter-node communication, and the diameter is as large as $\frac{k \times n}{2}$ when k is an even number. Some unique network structures, such as De-bruijn, Star, and Kautz, provide better average shortest path length (ASPL) when compared to the k-ary n-cubes.

In terms of ASPL and diameter, the random network topology is well-known to be better than the above network topologies, and its family appears in the graph catalog in the GraphGolf competition [54]. It assumes a topology-agnostic routing algorithm, e.g., up*/down* routing [55], and it requires routing tables that have $N \times C$ entries at each switch where N and C are the network size and the number of input ports of a switch, respectively.

As the number of nodes becomes large, e.g., 100,000, the required number of routing table entries will be larger than that implemented in recent switches, e.g., 48,000 in InfiniBand switch products [56] or 32,000 in Ethernet switch products. In this context, compact routing schemes aimed at finding the best trade-off between the number of routing table entries and the *stretch factor* of the routing path has been studied. The stretch factor is the worst-case ratio between the routing path length and the topological minimal distance for a source-and-destination pair. Two routing design directions/fashions have been proposed: (1) Hierarchical Routing and Addressing [57, 56] for regular topologies, and (2) the *universal compact routing* that can apply for arbitrary topologies [58, 59, 60] especially for Internet-like networks [61, 62].

Both of these two approaches require the implementation of routing table by off-chip Content-Addressable Memory (CAM/TCAM) [63, 64]. However, it is reported that the power consumption grows linearly with the size of CAM/TCAM, i.e., the number of entries, and becomes one of the most hungry modules, e.g., several tens of Watts [56, 65]. In addition, as the CAM/TCAM size has increased, the delay for its table lookup operation at each switch also become considerable in the low-switch-latency era. For example, extra latency overhead is about over 5 nanoseconds per switch with the use of

4 Mbits TCAM [65]. This extra latency may significantly affect the performance of HPC systems. Thus, design a routing algorithm that does not require a large routing table (with CAM/TCAM) is critical for low-latency interconnection network design.

In this desertion, we aim at avoiding the routing table usage by computing the routing paths at each switch using a combinational logic circuit. The main challenge is how to design the routing algorithm as simple as possible while keeping the routing path length as short as possible. In the following, we first overview the table-based routing approach with CAM/TCAM as the related work. We then describe in detail our custom routing for our proposed DSN topologies in Section 4.2.2 and illustrate how the routing path can be computed with the hardware synthesis in Section 4.2.3. We also present the possibility of extending our custom routing for dealing with fault/congestion or implementing using routing tables for other network design area.

4.1.2 Compact Routing Algorithms with Routing Table

Compact routing refers to the design of routing scheme that uses a small number of routing table entries (*RT*) at each node so that a small ratio between the path length of the route and the shortest path, i.e., the *stretch* of routes, is provided. For a given family of network topologies, finding the best trade-off between RT and stretch is the focus of this research area. The routing scheme that can apply to all the networks is called *universal compact routing*.

Focusing on small routing table size, Kleinrock and Kamoun proposed Hierarchical Routing and Addressing technique, which is the basis of today's Internet routing approach [57]. The main idea is that the nodes in a network are grouped into clusters, then the clusters may be grouped into clusters of clusters, and so on. Each node keeps complete information of the nodes that are close to it, e.g., in the same cluster, while stores less information of the further away nodes, e.g., stores the nodes in the other clusters by only one entry. The summary of related works in this area [61] shows that the efficiency of the hierarchical approach in terms of RT-stretch trade-off depends on (1) the abundance of remote nodes or (2) the regularity of the network topologies. Typically, the torus topology family has the former property, e.g., 3-D Torus with Dimension-Order Routing (DOR). Dragonfly [34], which is one of the current state-of-the-art high-radix topologies, and the *b*-ary tree structure are considered as the latter type of topologies. For example, Mariano et al. proposed to use the Hierarchical Routing and Addressing

technique for Exascale HPC Systems that employ Flattened Butterfly, Folded-Clos, and Dragonfly topologies in [56].

By contrast, the hierarchical approach is not suitable for random network topologies because these network topologies have small average path length (sparse remote nodes) by exploiting randomness (irregularity). In this context, researchers paid attention to the universal compact routing that has a good RT-stretch trade-off by finding the lower bound of the routing table size versus the stretch.

In a general view, the shortest path routing (stretch-1) requires $O(n)$ entries or $O(n\log(n))$ bits at each node, where n is the number of nodes. Below, we summarize the well-known theoretical bound of routing table size at each node with the “bits" metrics to keep the consistency with the previous work. The analysis in [58] showed that a compact routing scheme that uses only $O(n)$ bits leads to route a message with a stretch of a factor at least 3. Thorup and Zwick proved that any compact routing scheme with stretch less than 5 can not archive the RT smaller than $\Omega(n^{1/2})$ [66].

Looking insight the concrete routing schemes, Cowen proposed the landmark-based algorithms [60] with stretch-3 and $\tilde{O}(n^{2/3})$ of RT[2]. Improving this landmark-based approach, Thorup and Zwick designed a new routing scheme that archives stretch-3 with only $\tilde{O}(n^{1/2})$ [59]. The main idea of this approach lays on finding a representative set of nodes as landmarks and finding the neighborhood for each remaining node (the cluster of a node). A non-landmark node keeps complete information of the nodes in its own cluster and all the landmarks. A non-landmark node also uses a landmark node to be its representative, i.e., the closest landmark in terms of graph distance. A landmark node does not have a cluster but it stores the information of all the nodes that it represents. Thus, the routing is simple: if the destination node is in the same cluster of the source node, it uses shortest path routing; otherwise, the message is routed to the closest landmark of the destination node and then forwarded to the destination node.

4.2 Custom Routing Algorithm

4.2.1 Our basic approach

In this section, we provide the overview of the custom routing for our DSN topology. We remind the notations used to describe our topologies and routing algorithms as listed

[2] $\tilde{O}(f(n)) = f(n) \times \text{polylog}(n)$

Table 4.1: 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, $z \leq p = \lceil \log n \rceil$
l_i	Length of level-i shortcuts, $l_i = 2^{p-i}$
d_{st}	Distance between V_s and V_t, $d_{st} = (t-s) \mod n$

in Table 4.1.

We firstly consider the custom routing in Ring-based DSN topology between supernode V_s to V_t by using proper shortcuts and a few local links. Let $d_{st} = (t-s) \mod n$ 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 $k = \lfloor \log \frac{2^p}{d_{st}} \rfloor + 1$. 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 Figure 4.1. 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.

In Grid-based DSN topology, 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

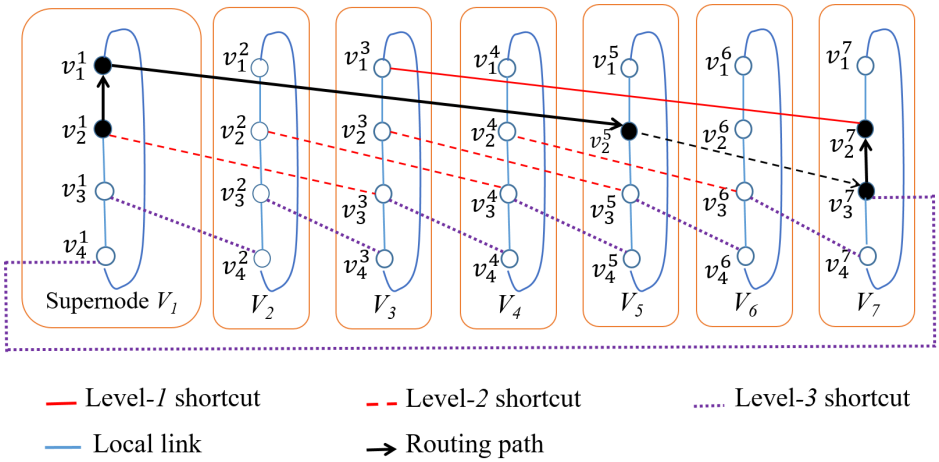


Figure 4.1: An example of custom routing in 1-D DSN.

in the same column as V. Thus, the route can be formed as follows three phases:

1. Find a route from $u \in U$ to node $w \in W$.
2. Find a route from $w \in W$ to node $v' \in V$.
3. 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 3 stages of L-routing and 2 of S-routing. The path length is thus upper bounded by $3\Delta + \log X + 1 + \log Y + 1$.

4.2.2 Concrete Routing Algorithms

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 $i = 1 \ldots z$. 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 we have just mentioned. 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. Figure 4.2(a) shows our ring-based routing algorithms in detail with three main features:

The ring-based DSN custom routing algorithm

```
1: procedure DSN-ROUTING(v_i^s, v_j^t)
2:     if d_st ≤ 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 ← 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 ← v_i^s                                  ▷ Start at source node
16:    repeat
17:        k ← ⌊log 2^p/d_xt⌋ + 1
18:        if level of u is not k then
19:            u ← v_k^s                          ▷ i.e., L-Routing from u to v_k^s
20:        else
21:            u ← u.Shortcut                     ▷ i.e., S-Routing
22:        end if
23:    until u in V_t                             ▷ i.e., reach supernode V_t
24:    u ← v_j^t                                  ▷ i.e., L-Routing
25: end procedure
```

(a) Routing for ring-based DSN

The grid-based DSN custom routing algorithm

```
1: procedure 2-D-DSN-ROUTING(i, j)
2:     DSN-Routing in X dimension
3:     DSN-Routing in Y dimension
4: end procedure
```

(b) Routing for grid-based DSN

Figure 4.2: Concrete custom routing for Distributed Shortcut Networks

Table 4.2: An example routing table of local routing between nodes inside a supernode.

To	From							
	1	2	3	4	5	6	7	8
1	-	2	3	4	4,5	3,6	2,7	8
2	1	-	3	4	4,5	3,6	7	1,8
3	1	2	-	4	4,5	6	2,7	1,8
4	1	2	3	-	5	3,6	2,7	1,8
5	8,1	7,2	6,3	4	-	6	7	8
6	8,1	7,2	3	5,4	5	-	7	8
7	8,1	2	6,3	5,4	5	6	-	8
8	1	7,2	6,3	5,4	5	6	7	-

- 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 $\frac{d_{st}}{2} \leq l_k \leq d_{st}$ (based on the distance-halving technique).
- Finally, the L-Routing path between any pairs of nodes inside a supernode are chosen as one of the shortest paths between them. For the ring-based DSN topology, it is the shortest path along the ring. For the grid-based DSN the routing paths are listed as in the Table 4.2. Each cell of this table shows the routing path. Nodes are indicated by its level.

In the following, we look into the details of the concrete custom routing algorithm for the grid-based DSN topology, 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). Figure 4.3 is an example of 512 nodes in the 2-D DSN topology where the supernodes of 8 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 $x_k = (7+4) \mod 8 = 3$ and y_k is the same as the source (also called the level-1 X-shortcut). Clearly, the X-shortcuts build up

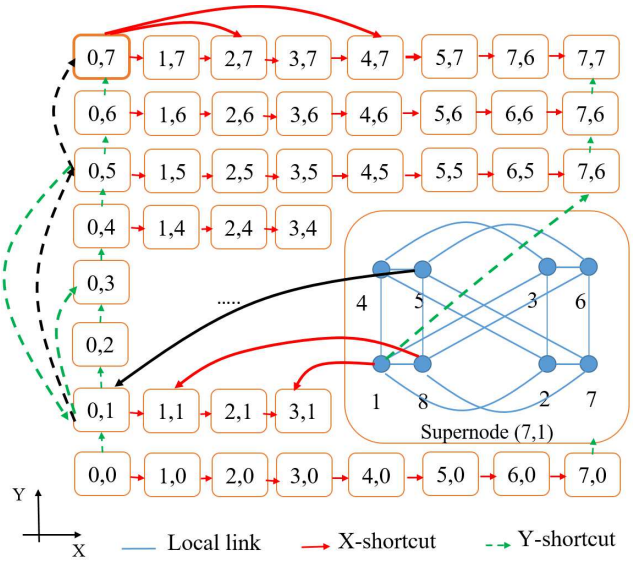


Figure 4.3: An example of custom routing in 2-D DSN.

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. In Figure 4.3, the black lines present the routing path from a node in supernode (7,1) to the supernode (0,7). This path is the concatenation of S-Routing in X dimension from supernode (7,1) to supernode (0,1) and the S-Routing in Y dimension from (0,1) to (0,7), respectively. 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.

4.2.3 Compute Routing Path with Hardware Synthesis Approach

Instead of storing the routing path in a table, we present how to compute our custom routing based on a combinational logic circuit (algorithmic routing). Figure 4.4 shows an

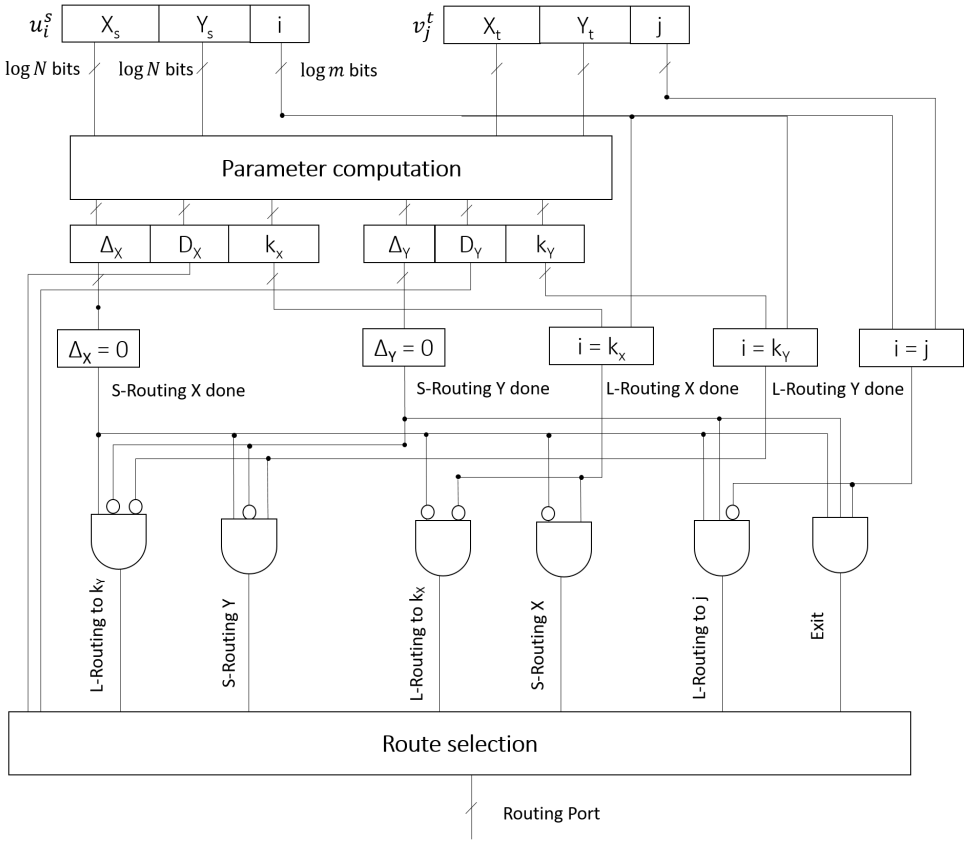


Figure 4.4: Algorithmic routing logic for DSN.

example of a logic circuit for our mentioned algorithmic routing for DSN topologies. The circuit at a node u_i^s (switch) accepts its information and the information of destination v_j^t in routing header as the input, i.e., the X, Y address of the supernode and node's ID. Based these input, the Parameter computation module determines (i) the distances between these two supernodes for each dimension Δ_x and Δ_y, (ii) the sign bit to present routing directions such as FORWARD_ROUTING or BACKWARD_ROUTING, i.e., D_x and D_y, and (iii) ID of nodes that hold the shortcuts based on halving-distance technique, i.e., k_x and k_y where $k_x = \lfloor \log \frac{2^p}{\Delta_x} \rfloor + 1$, $k_y = \lfloor \log \frac{2^p}{\Delta_y} \rfloor + 1$ (Let us call these as jumping nodes). The distances then are puts into the zero checkers modules to determines whether the packet has finished its movement in X(or Y) dimension or not. Typically a zero checker generates one signal bit such as "S-Routing X done" or "S-Routing Y done" for this purpose. Similarly, "L-Routing X done" and "L-Routing Y done" are generated with the comparison of current node ID i and jumping nodes ID k_x, k_y, respectively. These signals are then inputted to several AND gates that help to indicate which routing

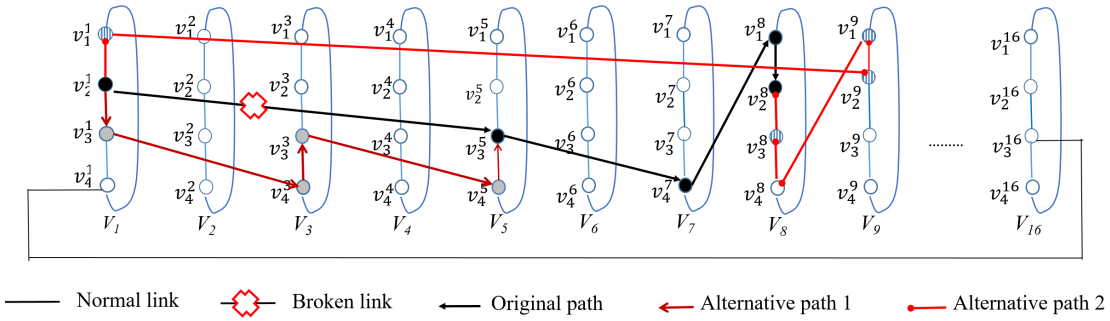


Figure 4.5: Alternative paths for failed links in 1-D DSN network of 64 nodes.

actions should be done. For example, the fourth gate from the lefts determines that if S-Routing is not done and the current node is the jumping node in X dimension, "S-Routing X" signals will be set. The Routing Selection module using the set of such signals to forward the packet to the proper ports.

4.2.4 Alternative Routing Path

This section illustrates that our routing algorithm can be extended to deal with fault and congestion by enabling alternative paths to the destination. We assume that a node u has information about the neighbor failures/congestion, i.e., a 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.

Alternative Routing in Ring-based DSN

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. Figure 4.5 illustrates an example of routing path from node v_2^1 in supernode $V_1$to node v_2^8 in 64-nodes (16 supernodes) 1-D DSN topologies by black nodes and arrow line in this figure. Clearly, the message has to travel a distance $d_{st} = (8-1) \mod 16 = 7$. Based on the routing mechanism discussed in section 4 of our paper, this sequence

Table 4.3: Alternative local routing in grid-based DSN.

To	From							
	1	2	3	4	5	6	7	8
1	-	2	3	4	8,5	8,6	8,7	8
2	1	-	3	4	7,5	7,6	7	7,8
3	1	2	-	4	6,5	6	6,7	6,8
4	1	2	3	-	5	5,6	5,7	5,8
5	8,1	7,2	6,3	4	-	6	7	8
6	8,1	7,2	3	5,4	5	-	7	8
7	8,1	2	6,3	5,4	5	6	-	8
8	1	7,2	6,3	5,4	5	6	7	-

presents the routing path $d_{st} = \sum_{i=2}^{p} a_i * 2^{p-i} = 0 * 2^3 + 1 * 2^2 + 1 * 2^1 + 1 * 2^0$. That is, we use the 2^2-distance shortcut links (level-2 shortcut) to move first, and then take the level-3, level-4 shortcut to reach the destination. Another way to present the routing path is that $v_2^1 \underset{S}{\rightarrow} v_3^5 \underset{S}{\rightarrow} v_4^7 \underset{S}{\rightarrow} v_1^8 \underset{L}{\rightarrow} v_2^8$ in which the symbol "$\underset{L}{\rightarrow}$" means L-routing, and "$\underset{S}{\rightarrow}$" stands for S-routing.

Without loss of generality, we assume that a 2^k-distance shortcut fails, e.g. $k = 2$ in example of figure 4.5. 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 2 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.

Alternative Routing in Grid-based DSN

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. For example, if an X-shortcut fails, we find the alternative S-routing path to move in X-dimension, then use Y-dimension routing. The alternative paths for L-Routing in grid-based DSN are shown in Table 4.3 where the alternative paths are still minimal routing.

In addition, we can select the routing order, i.e., X-dimension after Y-dimension and vice versa, for taking shorter path hops avoiding faulty links. In Section 4.3, we will evaluate two mechanisms to avoid faults, i.e., XY-routing, and XY-YX routing (choose the shorter path between XY- and YX- routing). Our result shows that the XY-YX routing has lower average shortest path length and increasing-rate on average path length compare to XY routing.

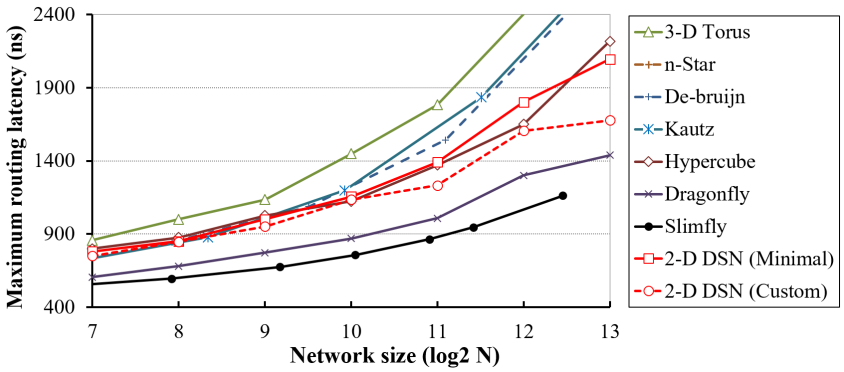
4.3 Custom Routing Analysis

We now measure the end-to-end latency of our custom routing algorithm described in Section 4.2.2 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 the degree-6. We also choose Hypercube, Dragonfly, and Slimfly as the 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 [38].

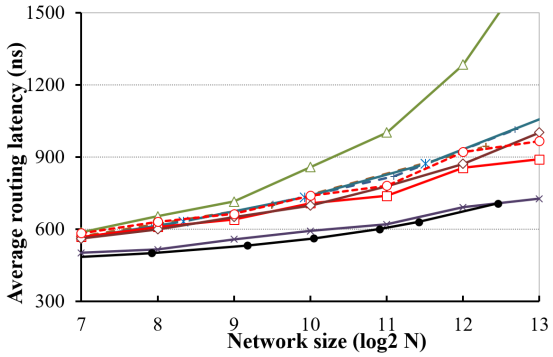
4.3.1 Maximum and Average Routing Latency

Figure 4.6 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 1024 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



(a) Maximum routing latency



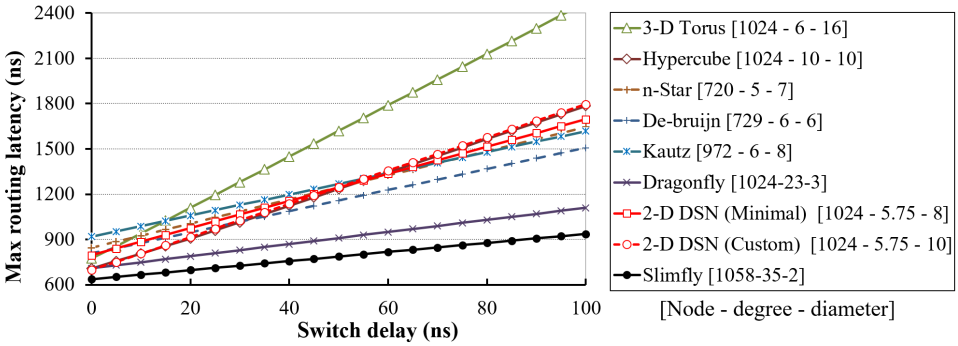
(b) Average routing latency

Figure 4.6: Maximum and average routing latency vs. network size. All the topologies have the degree of 6 except for Hypercube (from 7 to 13), Dragonfly (from 9 to 135), and Slimfly (from 11 to 79).

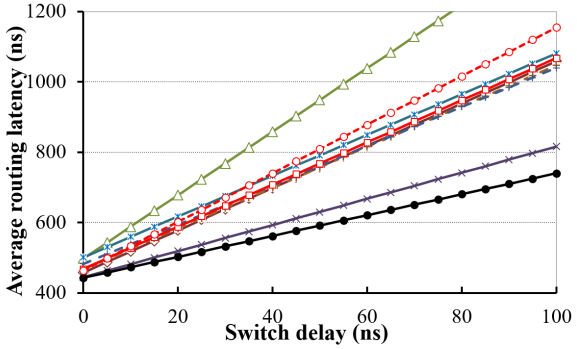
achieves lower latency than the other low-degree topologies with the minimal routing. Its performance is comparable to that of Hypercube with the minimal routing, since both topologies achieve logarithmic routing path hops. In addition, for our 2-D DSN topology, the custom routing gains a better performance 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.

4.3.2 Routing Latency vs. Switch Delay

To measure the range of switch delay in which our custom routing could achieve good performance, we estimate the routing latency of 1024-switch networks while changing the switch delay. Figure 4.7 shows the maximum and the average routing latency vs. the switch delay. For each topology, the legend in the figure indicates the network size, the



(a) Maximum routing latency



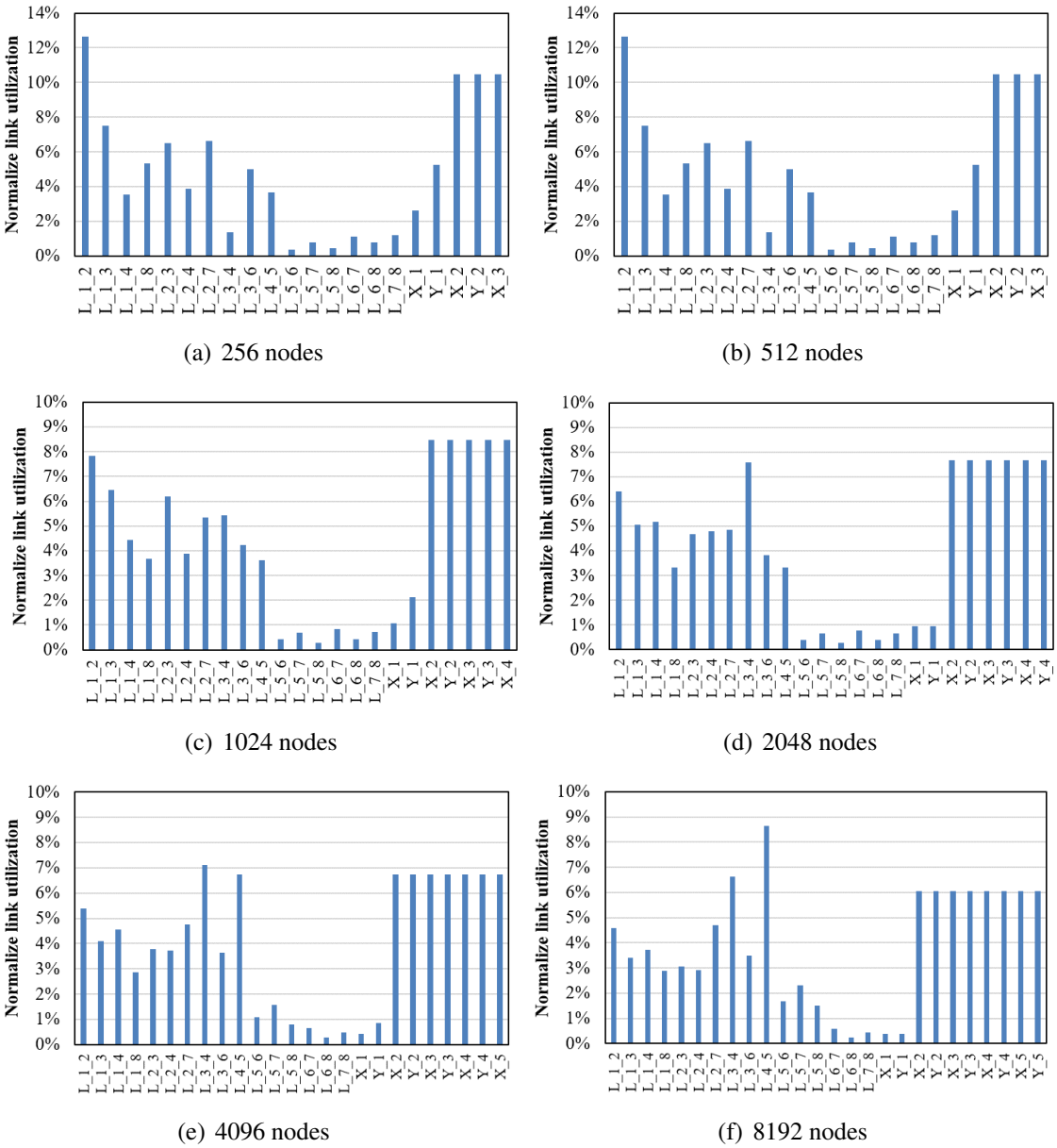
(b) Average routing latency

Figure 4.7: Maximum and average routing latency vs. switch delay in network of 128 cabinets, 1024 switches.

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% 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.

4.3.3 Alternative Routing

In this subsection, we first analyze the link utilization that helps to estimate the possibility of congestion occurs at each link. We count the communication source-and-destination pairs on each link for the case of the uniform random traffic pattern. Because DSNs are



(a) 256 nodes

(b) 512 nodes

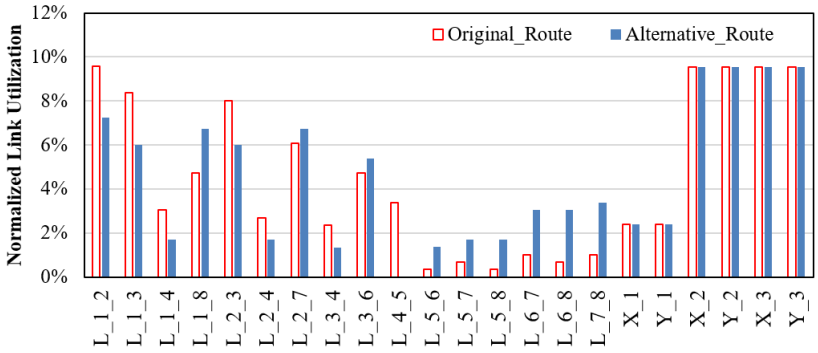
(c) 1024 nodes

(d) 2048 nodes

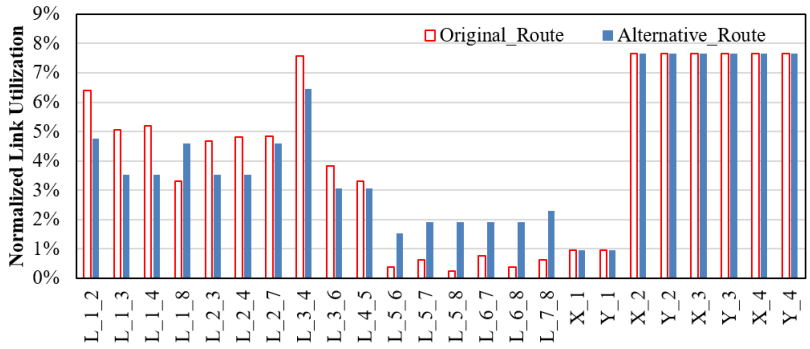
(e) 4096 nodes

(f) 8192 nodes

Figure 4.8: Link utilization of custom routing in various network sizes.



(a) 512 nodes



(b) 2048 nodes

Figure 4.9: Link utilization of custom routing with alternative paths.

symmetric topologies in terms of supernode point of view, without loss of generality, we measure the number of communication pairs of each link inside a supernode. Figure 4.8 presents the link utilization of our custom routing in various network sizes, e.g., from 256 to 8912 nodes. For each network topology, we use“L_*i*_*j*” to indicate the local link that connects node level i and level j while “X_k" and “Y_k" refer the level-i X-Shortcut and level-i Y-Shortcut, respectively. The result implies that the link utilization is uneven if an all-to-all communication is applied.

Specifically, all the shortcut links have the same link utilization excepts level-1 shortcuts. This is expected result because the level-1 shortcuts are used for route two supernodes with distance $\frac{n}{2}$, where n is the number of supernodes in a dimension (X or Y). That is they are used in the paths presented by $d_{st} = 2^{\log n-1}$. Level-k shortcuts, $k > 1$, are used in the path presented by $d_{st} = \sum_{i=2}^{\log n} a_i * 2^{p-i}$ where $a_k = 1$ and $a_i \in (0,1)$. Thus, level-k shortcuts are used in $2^{\log n-1}$ different set of routing paths. The utilization of local links are also uneven. Some local links are not frequently used, e.g., from L_5_6 to

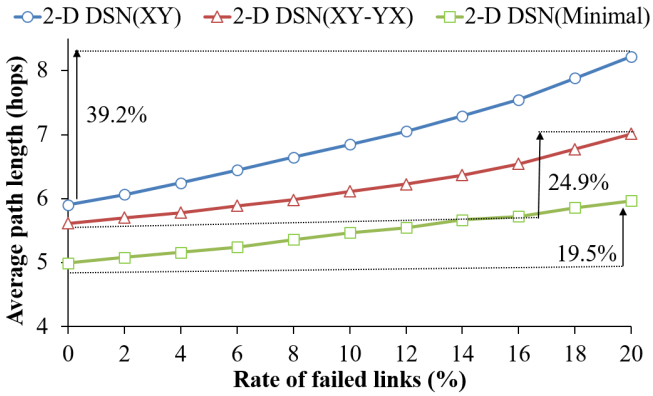


Figure 4.10: Average routing path length vs. the rate of faulty link in network of 1,024 switches.

L_7_8 in the network of 512 nodes. It is reasonable because node 5, 6, 7, and 8 do not have out-going shortcuts, then the related local links are rarely used. By contrast, the links that connect level-k nodes, $k \leq \log n$ are often used, e.g., L_1_2.

The uneven link utilization may lead to the communication congestion, especially in the case of a heavy load. Thus it may cause the network communication jitter, i.e., the increase of latency due to blocking at an intermediate switch. We thus provide the alternative routing path to (1) improve the load balancing and (2) automatic forward the packet if a link is blocked. Figure 4.9 illustrates the link utilization of custom routing with the local alternative paths mentioned in Section 4.2.4. Note that such alternative paths do not increase the total routing path length between nodes. Although the link utilization in Figure 4.9 is still uneven, it is possible to improve the load balancing of local links, i.e., intra-supernode link, by optimizing the local routing, especially when sacrificing a small latency. In addition, because most of the shortcuts have the same load except the level-1 shortcuts, a simple method is to increase the link bandwidth of a heavy link, e.g., by using link aggregation.

We then estimate the increasing of routing path length if the alternative paths are used if the network congestion and link faults appear. We randomly remove the links from the network of 1024 switches and evaluate the path lengths. Besides our two proposed mechanisms, namely XY-routing and XY-YX-routing (which chooses the shorter path between XY-routing and YX-routing), we consider the minimal routing as the baseline.

Figure 4.10 represents the average routing path length vs. the ratio of faulty/block

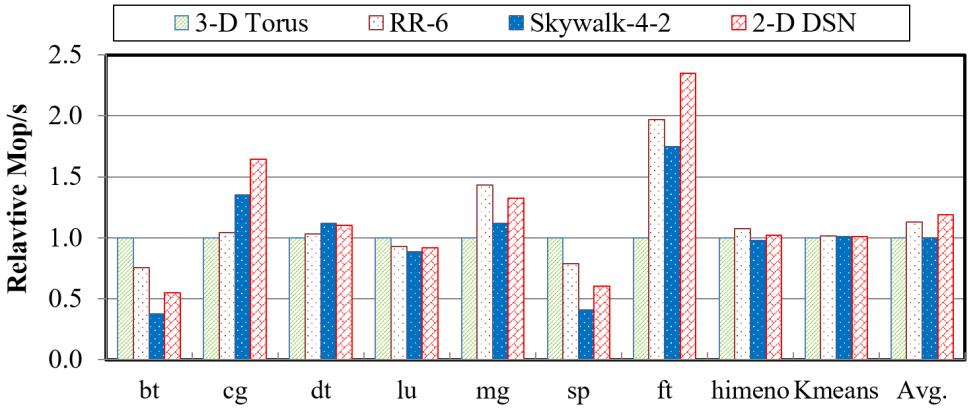


Figure 4.11: Application performance evaluated using NAS Parallel, Himeno, and BigDataBench benchmarks. "Avg" means the average over all the benchmarks. Values are normalized to those of 3-D Torus.

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 a 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% for XY-YX while these are 9.45% and 19.49% for minimal routing at the faulty link rate of 10% and 20%, respectively.

4.4 Performance Simulation

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

4.4.1 Parameters

We use the SimGrid simulation framework (v3.12) [67]. SimGrid implements validated simulation models, is scalable and simulates the execution of unmodified parallel applications that use the Message Passing Interface (MPI). We use NAS Parallel Benchmarks (version 3.3.1, MPI versions) [68, 69, 70] (Class B for BT, CG, DT, LU, MG and SP, and Class A for FT benchmarks), the Himeno Benchmark [71], and the

BigDataBench (version 3.1) [72] (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[73]. We simulate various topologies of 256 nodes with 40-ns switch delay and 40-Gbps link bandwidth. Each host has 100 GFlops in computation speed. We configure SimGrid to utilize its built-in version of MVAPICH2 for MPI collective communications [74]. 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.

4.4.2 Evaluation

Figure 4.11 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, normalized to that of 3-D Torus. Higher values are considered better. Overall, our proposed topology outperforms 3-D Torus by 19% 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.

4.5 Discussion

4.5.1 Achieving Deadlock-Free Routing

In this subsection 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.

4.5.2 Expression of Routing with Table-Based Approach

In this section, we discuss how to implement our custom routing with CAM/TCAM (compact routing). We learn the approach in Dimension Ordering Routing of the conventional torus network that uses a Hierarchical Routing and Addressing. We consider the grid-based DSN network of $N \times N$ supernodes, each includes m regular nodes. In our routing logic, the network is equally partitioned into several same

v_j^t	X_t (log N bits)	Y_t (log N bits)	Level j (log m bits)

(a) Hierarchical Addressing that requires $\log(N^2 \times m)$ bits.

Entry Number	Destination Address			Remarks
1	0b**000**	0b**000**	0b**000**	This switch
2 to *m*	0b**000**	0b**000**	0b---	Same-supernode switches. L-Routing
m+1	0b**000**	0b**001**	0b***	L-Routing to the node that holds the length-1 Y-shortcut
m+2	0b**000**	0b**01***	0b***	L-Routing to the node that holds the length-2 Y-shortcut
m+3	0b**000**	0b**1****	0b***	S-Routing since this node (node 0) the hold length-4 Y-shortcut
m+4	0b**001**	0b***	0b***	L-Routing to the node that holds the length-1 X-shortcut
m+5	0b**01***	0b***	0b***	L-Routing to the node that holds the length-2 X-shortcut
m+6	0b**1****	0b***	0b***	S-Routing since this node (node 0) the hold length-4 X-shortcut

(b) The routing table of switch v_0 of supernode (0,0) in DSN network of $8 \times 8 \times 8$ switches (Figure 4.3).

Figure 4.12: Hierarchical Addressing and Routing Table at a switch in a $N \times N$ grid-based DSN network. Each supernode includes m regular nodes.

size clusters, i.e., a supernode becomes one cluster. Our Hierarchical Addressing is designed to support the longest-prefix-matching lookup technique implemented in current TCAM [63]. Each switch keeps the complete information of all the same-cluster switches. Besides, all the switches that can be moved to by using the same outgoing shortcuts are grouped and stored by only one entry in the routing table. Thus, the address of a switch becomes a combination of the switch ID, and its supernode ID as shown in the Figure 4.12(a). This address mechanism and routing requires the small size of memory as the conventional addressing of torus network. That is, $2\log N + \log m$ bits are required for the destination address field and $m + 2\log N$ entries are needed at each switch. Figure 4.12(b) illustrates the routing table of switch v_0 of supernode (0,0) in DSN network of $8 \times 8 \times 8$ switches (Figure 4.3). By using TCAM, the wildcards can be stored with the data to indicate some bits of the address should not be used for determining the look-up function.

4.6 Summary

In this chapter, we present how to avoid the use of CAM/TCAM in routing implementation at each switch. We discuss support a custom routing mechanism for both the ring-based DSN and 2-D grid-based DSN topologies. 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 simple and small routing logic circuit. 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. 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.

5 Moderate Error-Proof Approach

We have already presented our cable-geometric approach including our network topology and its routing algorithm based on the current common link bandwidth configuration, i.e., 40 Gbps. We now present our approach in designing an interconnection network that uses ≥ 100Gbps link for HPC system. In this chapter, we firstly present the background of designing reliable high-speed optical cables with a requirement of a low-latency high-correction capacity FEC. We then briefly describe our light-weight low-latency FEC approach, e.g., Hamming FEC for the purpose of high-speed cable usage.

5.1 Introduction

5.1.1 Problem Statement

Recent high-bandwidth optical communication has used multi-level modulation, such as Pulse Amplitude Modulation (PAM), for increasing link bandwidth [40]. Since the use of multi-level modulation is more sensitive to noise than on/off keying, 40GBASE-R, 100GBASE-R and 100GBASE-P standards state an error correction code (ECC) such as

forward error correction component (FEC) on a hop in order to maintain the same Bit Error Rate (BER) as that in traditional interconnection networks, such as 10GBASE-KX Ethernet or 40Gbps-InfiniBand. Three generations of FEC has been proposed with the significant improvement of correction abilities including hard decision (HD) block codes, hard decision concatenated codes and soft decision iterative coding [43].

However, for the purpose of HPC system design, FECs become a performance bottleneck due to their considerable extra latency overhead for the data buffering and decoding computation, e.g., 84 nanosecond overhead per switch for storing 2112-bit in the conventional switch-to-switch Reed Solomon FEC [75] or potentially extremely high latency for storing more than 10,000 bits in the state of the art end-to-end FECs[76]. Figure 5.1 illustrates the relationship between switch delay and the relative execution time of NAS Parallel Benchmarks on a 256-switch system. We measured their performance by SimGrid discrete-event simulation. Our main finding is that each NPB application is latency sensitive (these results are consistent with the report in [77]). When the switch delay is 160ns, this is a case for FEC, the average performance decreases up to 2 times when compared to the case of 60ns switch delay without FEC

In this context, we aim at exploiting an approach that avoids such heavy FEC operations in HPC systems. This avoidance approach introduces difficulties in reaching the low BER, such as 10^{-15}, that can be marked as "error-free". In this study, our network design aggressively investigates the relationship between the HPC applications execution time and BER. Our network design thus provides just the BER required by HPC applications.

5.1.2 Related Work

For the purpose of high-speed HPC system design, in spite of correction capacity, the requirement of low latency becomes critical in FEC design. It is reported that the utilization of recent FEC leads to a high extra latency overhead due to its time-consuming for the data buffering in the decoding computation. For example, 25 Gbps high-reliability links need at least 84 ns overhead at each switch for storing the 2112-bit encoding blocks in the conventional RS(255,239) FEC [75]. Similarly, RS(528,514) requires approximately 100 ns overhead due to its 5280-bit buffer [46]. In addition, the second and the third generation FEC such as BCH and LDPC code, respectively also have long lengths of encoding blocks [76] that potentially contribute an extremely high latency.

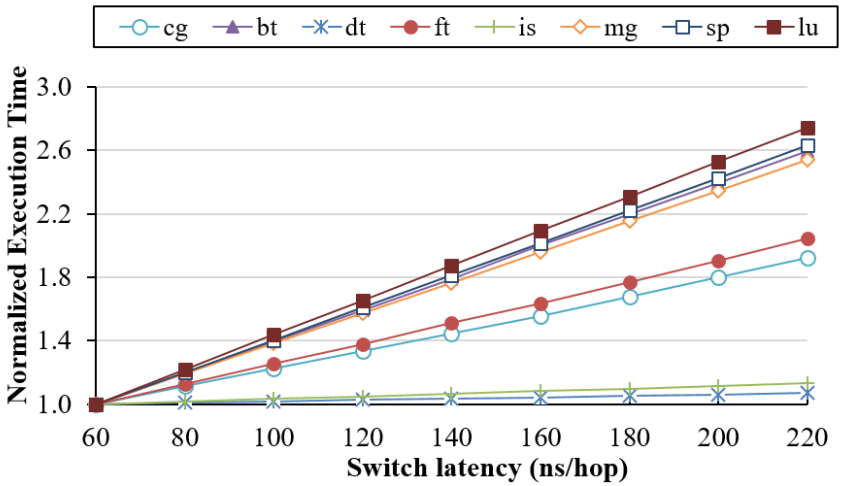


Figure 5.1: Application execution time versus switch latency in 256-switch system (except IS, which has 64 switches). Values are normalized to those of 60ns.

M.N.Sakib et. al. summarized the characteristics of various FEC schemes have already been proposed for PAM signals in data center applications [78]. According to this survey, the decoding latency overhead of 100 Gbps system with PAM signals are in a range from 66 to 140 ns. Such kind of latency drastically reduces the HPC system's performance as shown in Figure 5.1.

In this context, an alternative approach of FEC-avoided is considered in HPC system where the bit errors occur randomly instead of under burst conditions. As an instance, Daichi et. al. proposed to use the Quadrature Amplitude Modulation (M-QAM) for high-bandwidth and a symbol mapping encoding/decoding mechanism (gray code) to partially correct the errors occurred in some parts of the communication data in order to reduce the numerical error rate [77], i.e., reliable, slow transmission for significant bits while the less important bits are transferred with a high-speed unreliability mode. This approach is reported to significantly outperforms the conventional 100 Gbps interconnect network but only apply for the approximate and algorithm-based fault tolerance (ABFT) applications [77].

A related approach is proposed by IEEE 802.3bs 400-Gigabit Ethernet Task Force [41], i.e., improving the error correction/detection capabilities of line code/-transcode at the Physical Coding Sublayer (PCS) combined with the MAC-CRC. The main idea behind this approach is to increase the Hamming distance of PCS code by using the saved checksum overhead when FEC is disabled, i.e., 8 bits per 256-bit data block. Thus, the same PMA interface and coding rate are untouched. For example,

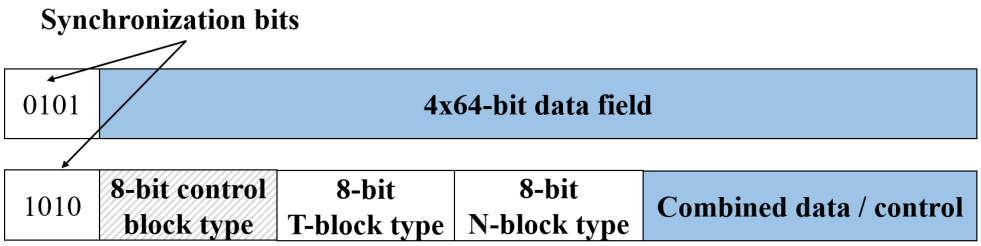


Figure 5.2: 256b/260b PCS code format

Figure 5.2 illustrates the 256b/260b PCS code which is the alternative for RS(528,514) FEC with 256b/257b PCS code [79]. The only difference between 256b/260b and 256b/257b is the synchronization bits, i.e., 4-bit instead of 1-bit respectively. This improvement helps to significantly reduce the undetected error rate (improve the Mean Time To False Packet Acceptance - MTTFPA). However, as BER rates gradually up-scaled, this approach becomes ineffective because of the MTTFPA of less than one year and it also is incapable of correct the errors.

In this study, we propose to use a low-latency light-weight FEC for such HPC application. That is we aim at designing an FEC code that has a short length of code word with a low but acceptable capability of correcting bit errors. We investigate the feasibility of such FEC code in the next section.

5.2 Low-Latency Light-Weight FEC in HPC Systems

5.2.1 Requirements for Light-Weight FEC

In this section, we analyze FEC requirements in HPC system with various aspects including correction abilities, coding gain, and coding overhead.

Error Correction Capacity

We firstly consider the required error correction capacity to assure the accuracy and integrity of communicated data that is paramount importance for most of HPC applications. A key parameter of any error control scheme is the rate at which damaged packets are accepted as valid, also known as Undetected Error Rate or Mean Time To False Packet Acceptance (MTTFPA). In general, such a failure is capable of hard crashing the HPC system.

For our target error correction scheme, we assume that the MAC-CRC checksum and the PCS Line Code are implemented beside FEC in order to keep the compatibility with the current network standard. We thus analyze the relationship of MTTFPA and the post-FEC bit error rate with various typical PCS codes, e.g., 66b/66b that described in IEEE-802.3ba standard [40] and 256b/257b or 256b/260b that suggested by IEEE 802.3bs Task Force [79].

We estimated the MTTFPA of these three systems by using the same method mentioned in [80], which considers the possibility that a false MAC frame generation occurs due to either (i) the errors hit some control bits of the PCS code such as Sync bits (p_{sync}) or Block Type Field bits (p_{btf}), (ii) or the errors that MAC-CRC cannot detect, i.e., four or more bits occur in MAC frame (p_{crc}). We also consider the practical overestimation of 2^{-32} that the CRC32 result of an error MAC frame coincident to the frame check sequence (FCS). The MTTFPA is described in the Equation 5.1. The Equation 5.2 presents the p_{crc} and the Equation 5.3 shows the possibility of i-bit errors occur in the packet of length L. For each PCS code, p_{sync} and p_{btf} are calculated based on its format. For example, in 256b/260b code, 4-bit synchronization header with the hamming distance of 4 and three different of block type fields with 8-bit, distance-4 code for each field are used. That is any every error patterns with more than 2-bit errors in these fields will cause the false MAC frame generation. Thus $p_{sync,256/260} = p(3,4,p_e) + p(4,4,p_e)$ and $p_{btf,256/260} = 3\sum_{i=3}^{8} p(i,8,p_e)$.

$$\text{MTTFPA} = \frac{\text{time of a bit}}{p_{sync} + p_{btf} + p_{crc}} \times \frac{1}{2^{-32}} \tag{5.1}$$

$$p_{crc} = \frac{\sum_{i=4}^{L} p(i,L,p_e)}{L} \tag{5.2}$$

$$\text{where} \begin{cases} p(i,L,p_e) = C_L^i (1-p_e)^{L-i} (p_e)^i \\ p_e = \text{post-FEC Bit Error Rate} \\ C_L^i \text{ denotes the combination} \begin{pmatrix} L \\ i \end{pmatrix} \end{cases} \tag{5.3}$$

Figure 5.3 shows the relationship of MTTFPA and the post-FEC bit error rate with a 1518-byte-length MAC frame and a bit time of 10^{-11} second, i.e., 100 Gbps of bit rate. Expectedly, the result shows that the powerful FEC that can produce output BER at

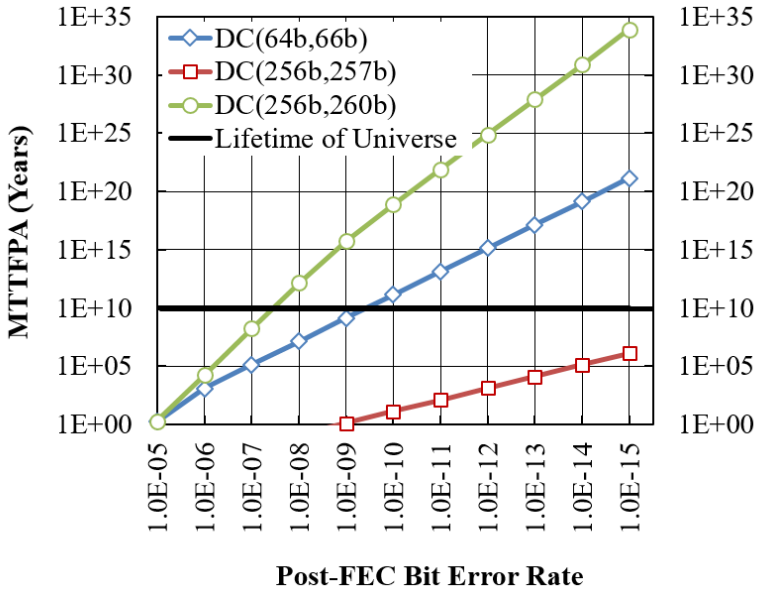


Figure 5.3: False Packet Acceptance Rate versus the post-FEC Bit Error Rate. 100Gbps-systems of various PCS codes integrated with 1518-bytes MAC-CRC are considered.

10^{-15} will lead to a long time of MTTFPA, e.g., longer than the lifetime of the universe, and conversely, a very short time of MTTFPA if the post-FEC BER becomes higher, e.g., a few hours for 256b/257b PCS code at the 10^{-8} of BER. Specifically, we found that for 64b/66b and 256b/260b PCS code, the requirement of output BER for FEC is just around 10^{-9} to maintain the acceptable MTTFPA, as long as the lifetime of the universe. In addition, the MTTFPA of 256b/260b at a 10^{-9} BER is approximately equal to the 64b/66b performance at 10^{-12} BER that specified in 100Gbps Ethernet Standard [40]. It implies that a low-latency light-weight FEC with a PCS that has a high error correction capacity such as 256b/260b can be an alternative option for the conventional Reed Solomon FEC with 64b/66b or 256b/257b PCS code.

Another critical concern is to take into account the bit error rate of data in the application layer. Because the PCS code and the MAC-CRC check bits are only able to detect instead correct error bits, the propagation of such errors to the upper layer is capable of corrupting the final output of the application. We below investigate the successful rate of NAS Parallel Benchmarks when BER is changed on a 256-switch system that arranged in a 4-D torus network. We measured the accuracy of applications by Simgrid discrete-event simulation [81]. The detail of the simulation environment is described in Section 5.3. We simulated each configuration 100 times and counted the

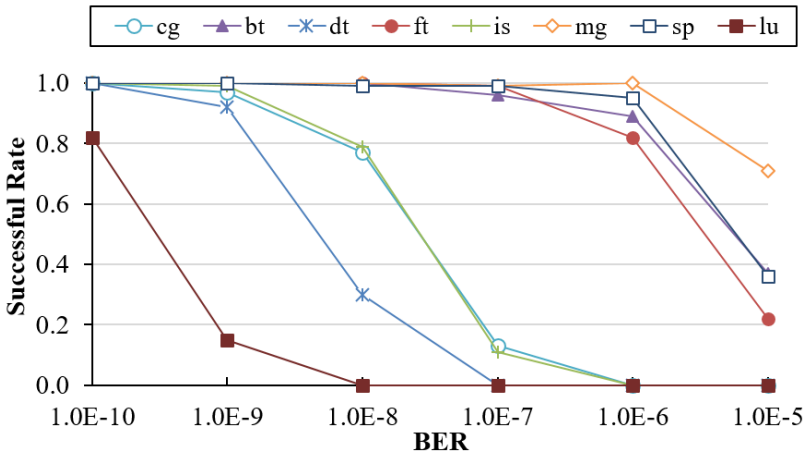


Figure 5.4: Successful rate of NPB applications versus Bit Error Rate of Data. Each successful execution rate is counted over 100 simulations.

successful rate of the application. The execution is successful if the obtained result is similar to that in an error-free environment.

The results in Figure 5.4 show that most of these applications except LU obtain the correct results under BER of 10^{-10}. Therefore, a low-latency light-weight FEC should support the output BER less than 10^{-9} in order to maintain the correctness of the parallel applications.

Post-FEC BER and SNR

In the previous subsection, we found that the parallel applications do not require an error-free communication, i.e., post-FEC BER at 10^{-15}, but rather only around 10^{-9}. Below we look into detail the association between post-FEC BER and the Signal To Noise Ratio (SNR) of several different FEC codes including the conventional RS(255,239) of 100Gbps Ethernet, the RS(528,514) for Terabit Ethernet, and some potential approaches for low-latency, light-weight FEC. In this study, we estimated the output BER of these FEC codes in the system that implemented the typical multiple-level modulation such as PAM-4 and PAM-8. The theoretical BER of an uncoded system for PAM-M, is given by:

$$\mathrm{BER}_{\text{PAM-M}} = b_i = \frac{M-1}{M\log_2 M}\mathrm{erfc}\left(\sqrt{\frac{3log_2 M}{(M^2-1)}\frac{E_b}{N_0}}\right) \tag{5.4}$$

This BER then becomes the input BER for FEC, denoted as b_{i}. According to the approximation method mentioned in [82], we then calculated the post-FEC BER of Reed-Solomon(n,k) based on symbols from Galois Field GF(2^m) where each symbol consists of m bits as in Equation 5.5. b_o, s_i, s_o are the post-FEC BER, input and output symbol error rate, respectively, and $t = \frac{n-k}{2}$ is the number of symbol errors that Reed-Solomon FEC can correct.

$$\begin{cases} b_{o,\,\mathrm{RS}} = \frac{b_i \times s_o}{s_i} \\ s_i = 1 - (1 - b_i)^m \\ s_o = \frac{1}{n} \sum_{j=t+1}^{n} j C_n^j (s_i)^j (1 - s_i)^{n-j} \end{cases} \tag{5.5}$$

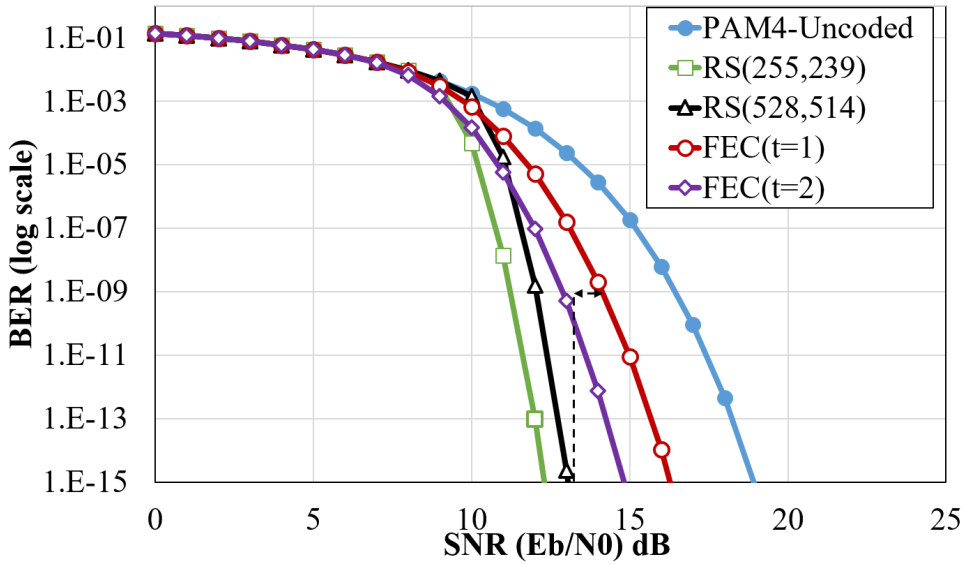
We also measure the post-FEC BER of a light-weight FEC that is capable of correcting up to t error-bits with codeword's length of N as:

$$b_{o,\,\text{FEC-}t} = \frac{1}{N} \sum_{j=t+1}^{N} j C_N^j (b_i)^j (1 - b_i)^{N-j} \tag{5.6}$$

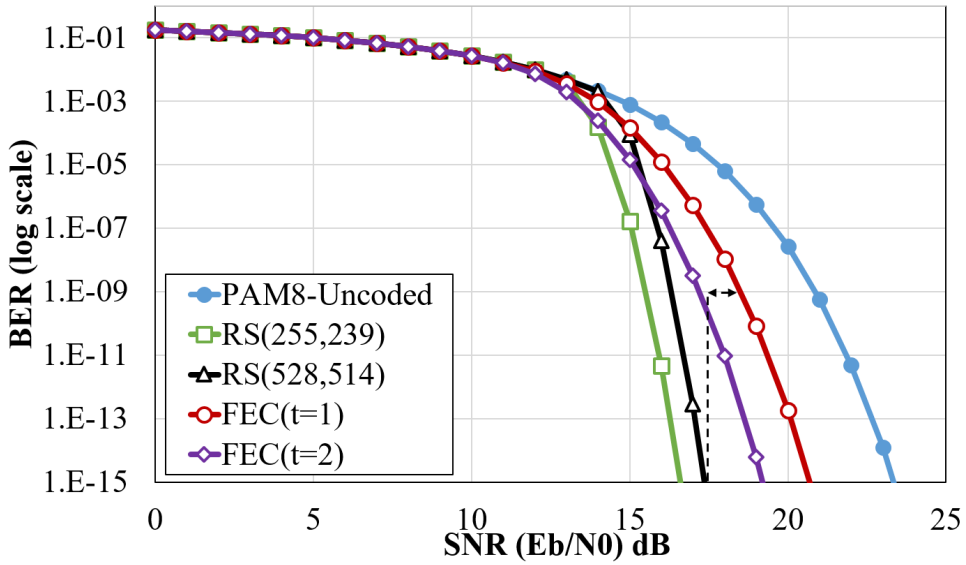
Figure 5.5 illustrates the numerical result of BER versus Signal to Noise Ratio (SNR) of systems using the same modulation, e.g., PAM-4 and PAM-8 with different FECs. FEC(t=x) refers to a system that uses a light-weight FEC that can correct up to x error bits. RS(255,239) and RS(528,514) is chosen as the representative of Reed-Solomon Code because it has been adopting for 100 Gbps optical and the copper backplane, respectively [40, 45]. Expectedly, the result shows that the higher power of FEC leads the lower energy per bit for a given targeted BER, i.e., the better Coding Gain. Interestingly, the SNR of light-weight FECs at BER of around 10^{-9} inconsiderably exceeds that of Reed-Solomon code at BER of 10^{-15}, e.g., approximately 1dB higher in the comparison between FEC(t=1) and RS(528,514). Thus, we state that a low-latency light-weight FEC is practical in terms of low energy consumption which is also one of the critical aspects of HPC system design.

Coding overhead

It is well-known in the literature that in an Additive White Gaussian Noise (AWGN) channel, for the same code rate, the longer the codeword, the better the error correction capability [82]. However, for the low-latency perspective in HPC system, the codeword



(a) PAM-4



(b) PAM-8

Figure 5.5: Numerical result of BER versus Signal to Noise Ratio (SNR) of different system using the same modulation. FEC(t=x) refers to a system that uses a light-weight FEC that can correct up to x uncorrected bits.

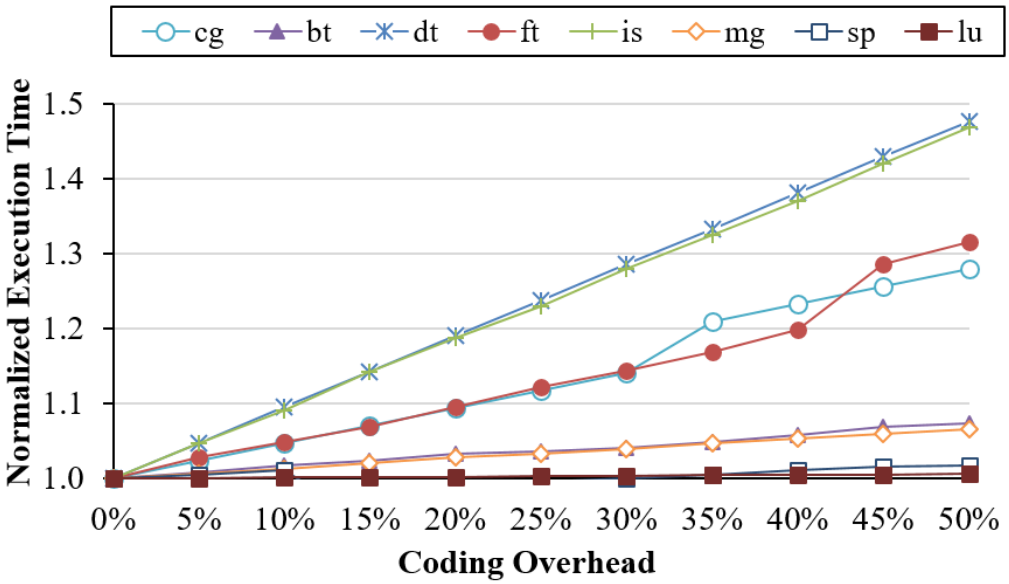


Figure 5.6: Application execution time versus Coding Overhead in 256-switch system (except IS, which has 64 switches). Values are normalized to those of 0%.

is restricted its length as short as possible. Thus, fixing a short length of code-word and increasing the number of redundant bits, i.e., coding overhead, becomes reasonable. This fact leads to a trade-off between the coding overhead and the error correction capability of FEC design that need to take into account, i.e., a trade-off between switch latency and network bandwidth in terms of overall system performance. Figure 5.1 has expressed the relationship between the switch delay and the parallel application's execution time. In this subsection, we measure their performance when the coding overhead is changed by using the Simgrid simulation on a 256-switch system that arranged in a 4-D torus network. The switch delay is fixed at 100 ns per hop.

The results in Figure 5.1 and Figure 5.6 present their classifications of parallel applications including: (i) latency-sensitive such as BT, MG, SP, and LU, (ii) throughput-sensitive such as IS, DT, (iii) and both latency-throughput-sensitive like CG and FT. For such throughput-sensitive applications, coding overhead becomes a critical aspect of the application execution time. For example, 20% coding overhead of FEC leads to 9.54% and 18,81% execution time longer of FT and IS application, respectively. In summary, for the purpose of designing a high-speed HPC system, not only FEC's codeword size which corresponds to the switch latency but also the coding overhead which is related to network throughput need to be considered. In our opinion, to maintain the same coding overhead like it is in the first generation of FEC, i.e., up to 7%, is reasonable.

Hardware Specification

In this subsection, we explore the feasibility of a lightweight FEC in terms of hardware implementation. As shown in Figure 2.10, FEC is arranged in the middle of PCS and PMA sublayers. Thus, PCS puts some constraints on FEC design and vise versa. This aspect should be considered together when FEC is physically implemented along with the complexity of FEC itself.

In the Section 5.2.1, we illustrated that PCS and FEC codesign is one of the critical concerns for achieving high correction capacity. When a powerful FEC such as RS-FEC is applied, the hamming distance of PCS code does not a matter for long MTTFPA anymore [79]. By contrast, with a low-latency lightweight FEC, PCS code is also an important factor because when the post-FEC BER becomes high, e.g., 10^{-9} instead of 10^{-12}, the MTTFPA becomes worse even MAC-CRC is used.

In addition, there exists a constraint between the size of codewords of PCS code and FEC. The codeword of a PCS M-bit/N-bit code should align with the FEC codeword. For example, a RS-FEC(n, k) over GF(2^m), which includes m-bit k information symbols and $n-k$ checksum symbols and guarantees to correct up to $\frac{n-k}{2}$ errors, must satisfy $0 < k < n < 2^m$. Typically the size $M = 64i$, $N = M + j$ should satisfy $a \times N = m \times k$ where $i = 1, 2, 4, 8 \ldots$, j and a are integer numbers. As instance, 20 blocks of PCS code 256b/257b (Transcode) are integrated with RS(528,514) over GF(2^{10}) in the 100GBASE-SR4 Standard (802.3bm) [40].

For the low-latency purpose, the length of codewords of lightweight FECs should be as short as possible. Considering a given lightweight FEC with a PCS code M-bit/N-bit. The length of FEC's codeword l should satisfy $l = a \times N$ where a is a small integer.

5.2.2 Our Approach with Hamming FEC

The survey in the previous section shows that it is possible to use a lightweight FEC for HPC systems for the low-latency purpose. The requirements for lightweight FECs are just capable of correcting just one or two bits errors in order to maintain the output BER of about 10^{-9}. In addition, the lightweight FEC coding should be integrated with a high error correction capacity PCS code for maintaining a long MTTFPA. It also reasonable for design an FEC with small codeword but 7% Coding overhead.

In this study, we present a simple case-study for further analyzing the potential performance benefit of lightweight FECs by reducing the latency overhead. Our main

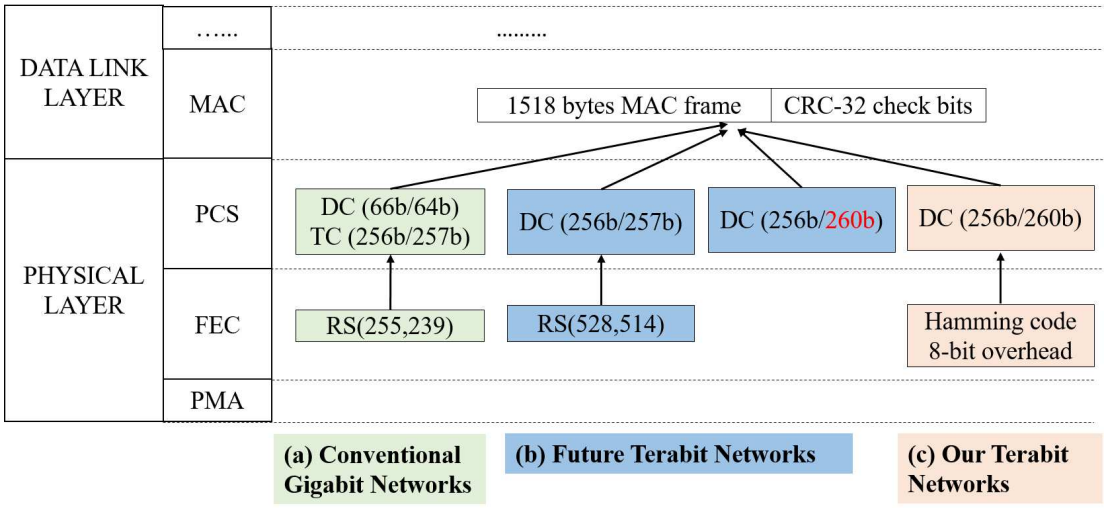


Figure 5.7: Our moderate error-proof low-latency error control mechanism.

idea is to apply a low-latency lightweight FEC with a PCS that has a high error correction capacity such as 256b/260b instead of using the Reed Solomon FEC with conventional PCS code. Aiming at providing a BER of less than 10^{-9} in order to maintain the correctness of the parallel applications and 7% of coding overhead, we straightforwardly target the Hamming code (hereafter called Hamming-FEC). The Hamming code can correct 1 bit error and detect up to 2 bits. For 260b PCS code, 10 extra checksum bits that leads to only $\frac{270}{260} - 1 = 3.84\%$ overhead. Figure 5.7 presents our high-speed approach in the comparison with the conventional gigabit and future terabit networks. In this study, we initially focus on a 1-bit error correction FEC such as Hamming Code that has a small coding overhead, e.g. 8-bit overhead over 260-bit of data, and do not require a large size of codeword for the low-latency purpose.

The Hamming code is also simple for hardware implementation. Since it is consistent with existing frame format based on standard Reed-Solomon FEC with DC(64b/66b) direct linecode and TC(256b/257b) transcode, respectively, the influences upon the other network layer design are limited. Thus, it is feasible to implement a mechanism to switch the FEC based on the error-free requirement. For example, to use an high-accuracy FEC for applications strictly require the error-free communication and switch to our light-weight FEC for applications that can tolerate bit errors. The detail of this mechanism is mentioned by IEEE Ethernet 400G Task Forces by considering the Auto-Negotiation sublayer (AN) under Physical Medium Attachment sublayer (PMD) [83].

5.3 Performance Evaluation

5.3.1 Methodology

In this section, we use discrete-event simulation, i.e., SIMGRID simulation framework version 3.14 [81], to evaluate the performance of parallel benchmarks which based on Message Passing Interface (MPI) when executed on the conventional and our proposed network. We configure Simgrid to utilize its built-in version of MVAPICH2 for MPI collective communications [74]. We implemented the random bit-error model within the Simgrid simulation models to evaluate the application accuracy. For each communication data of MPI_send/MPI_receive pairs, bit flips are generated with uniform random distribution with a possibility of bit error is set based on the network configuration.

We simulated the execution of the NAS Parallel Benchmarks version MPI 3.3.1, MPI versions [68] except for LU benchmark[1]. The detail description and characteristics of the benchmarks and algorithms have been presented in [70, 77]. In this study, we aim at analyzing the affect of different error correction mechanisms to the overall network performance with the same configuration of network topology, physical layout, and routing algorithm For all of the simulations, we use various network topologies of 256 switches such as the 3D torus, Dragonfly, and Random network. These networks are placed in a physical layout with a strong assumption that all the cable lengths are fixed to 5 meters. For each network topology, minimal routing is implemented for the simulation. We also set one host per switch with the total computation speed of 100 GFlops.

We then compare a conventional 100 Gbps interconnection network (CONV), a high-bandwidth interconnection network with Reed Solomon RS(528,514) FEC (RS-FEC), and our proposal high-bandwidth interconnection network approach (Hamming). We measured the RS-FEC and our Hamming approach with 200 Gbps, 400 Gbps, and 1000 Gbps link bandwidth. Table 5.1 shows the details configurations of these networks including the link bandwidth, latency at each switch, bit error rate and the error control mechanism.

Specifically, we choose a small value for the case of transmission without FEC as the base delay. The switch latency is the calculated by adding the latency overhead of FEC to the base delay. In this study, we estimate the base delay to around 40-60 ns as

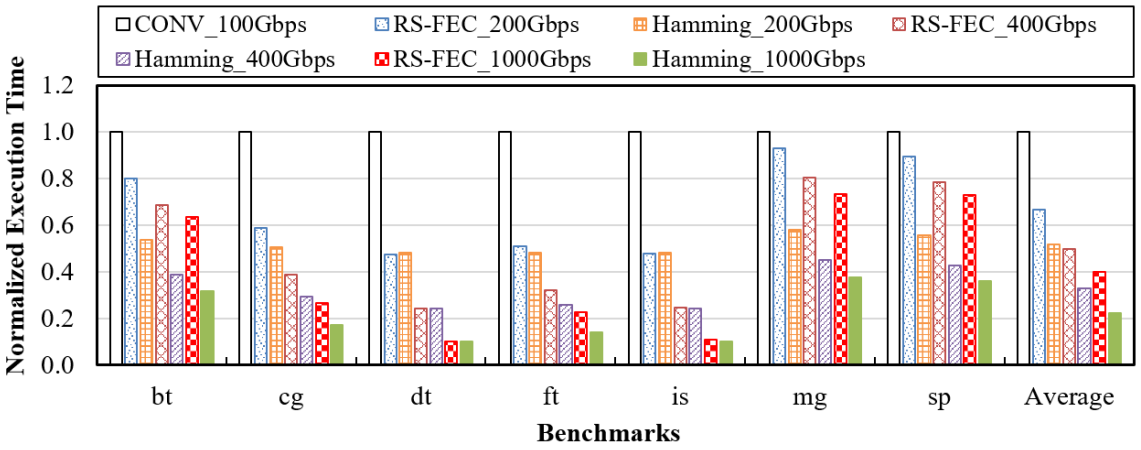
[1] As shown in our survey, our approach is not guaranty the 100-percent perfect execution of LU benchmark if the BER is set to 10^{-9}. We remove LU benchmark from our evaluation and discuss it in other work

Table 5.1: Network parameters for Simgrid simulation

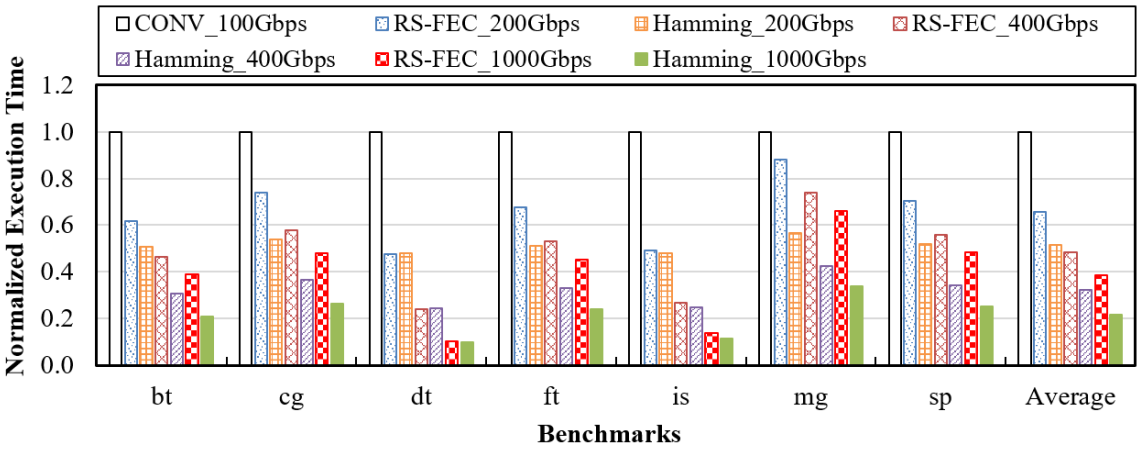
	Link BW	**Error Control Code**	**Coding rate**	**Switch latency**	**post-FEC BER**
CONV	100Gbps	TC(66,64) + RS(255,239) with 2112 bits of buffer	1.100	100ns	10^{-15}
RS-FEC	200Gbps, 400Gbps, 1Tb/s	TC(257,256) + RS(528,514) with 5280 bits of buffer	1.031	160ns	10^{-15}
Hamming	200Gbps, 400Gbps, 1Tb/s	TC(260,256) + Hamming Code with 270 bits of buffer	1.054	60ns	10^{-9}

mentioned in Chapter 3, i.e., 45.3 ns on BlueGene/Q or 40.1 ns on Anton-2, and reported in [77]. Considering the latency overhead of HAM-FEC, we assume the bit rate of 50 Gbps per lane, i.e., approximately 0.02 ns overhead for storing 1-bit of the buffer. Due to bypassing heavy FEC, the latency overhead of our proposed network is small as 5.4 ns per switch (270 bits), but it leads to the bit error rate become high, e.g., 10^{-9}. By contrast, RS-FEC approach achieves a free-error, i.e., 10^{-15} of BER but high switch latency.

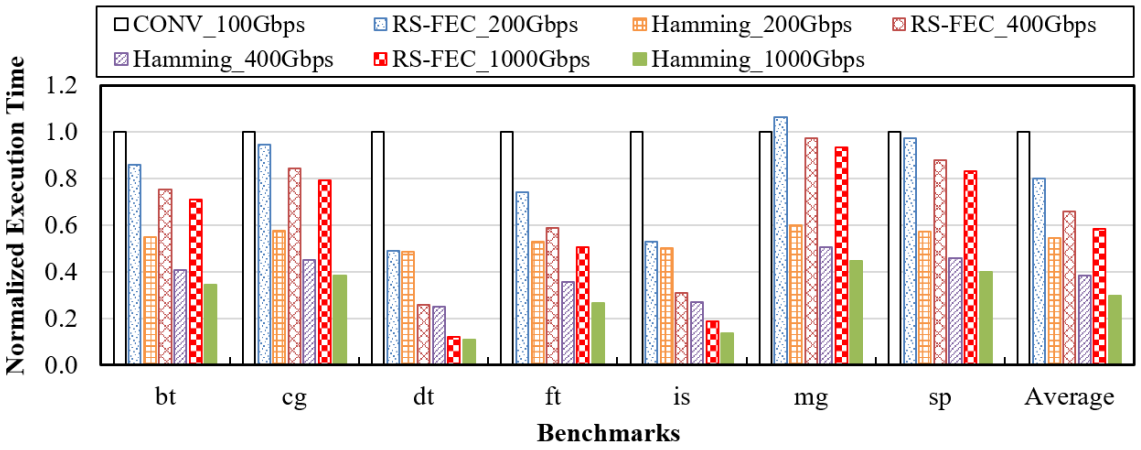
We consider the FEC configuration with a RS-FECs over b physical lanes, i.e., 100G RS(255,239) $\times 1$ and 100G RS(528,514) $\times k$ architecture for CONV, and RS-FEC approaches, respectively. Although (kx100G) RS-FEC $\times 1$ architecture provide a low latency, it is impractical because of the high signal rate at FEC component, e.g., 400 Giga-baud for 400Gps RS-FEC, and the limitation of re-use the 100G Standard such as IEEE 802.3bj. Instead, we use the 100G RS-FEC$\times k$ architecture over sixteen or eight physical lanes with 25-Gbps and 50-Gbps per lane, respectively as suggested by the IEEE 802.bs Task Force [84, 41]. The 100G RS-FEC helps to reduce the latency overhead per 1-bit of the buffer, i.e., 0.01 ns per bit, but it is reported that the latency of decoding of a RS-FEC are on the order of $2n$ where n is the size of a codeword [85, 86]. We thus approximate the latency of RS(255,239) and RS(528,514) as $\approx \frac{1}{100G} \times 2 \times 2112 \approx 42.24$ ns and $\approx \frac{1}{100G} \times 2 \times 5280 \approx 105.6$, respectively. Overall, we consider the switch latency of CONV, RS-FEC and Hamming approach as 100, 160, and 60 ns, respectively.



(a) Torus, 256 switches



(b) Dragonfly, 256 switches



(c) Random, 256 switches

Figure 5.8: Application performance evaluated using NAS Parallel. Values are normalized to those of CONV approach.

5.3.2 Application Execution Time

Figure 5.8 shows simulation results for CONV, RS-FEC, and Hamming approaches. The y-axis indicates the relative execution time of each approach normalized to that of CONV. Lower values are considered better.

The results show that RS-FEC and Hamming approaches reduce the execution time significantly in a comparison with the conventional approach for all the network topology. This is an expected result because RS-FEC and Hamming use higher link bandwidth, i.e., from 200 Gbps to 1000 Gbps compare to 100 Gbps as in CONV. Higher link bandwidth is used, more reduction time are gained. Interestingly, there exists one exception with mg benchmark that runs on the random network, 200 Gbps link bandwidth. It result refers that the performance improvement of using high bandwidth link is not enough to compare to the performance sacrifice by switch latency overhead of RS-FEC. This bad trade-off happens with the only mg because mg is a latency-sensitive approach but not throughput-sensitive application as mentioned in the survey of the previous section. For the same set of latency-sensitive application such as SP, although the overall performance still better than CONV but the reduction of execution time is smaller than the other applications, e.g., approximate 7% compare with 33% in case of mg and ft, respectively.

Regarding the comparison of our approach to the RS-FEC approach, our proposal has a better performance in all the configuration due to having smaller switch latency while maintaining the small coding overhead. Expectedly, for non-latency sensitive application such as IS and DT, our Hamming approaches trivially outperform the RS-FEC. For example, the Hamming FEC reduces 0.6%, 7.77%, and 13.01% execution time in case of IS benchmark running on the system of 400Gbps and 3-D Torus, Dragonfly, and Random topology, respectively. By contrast, the effect of low-latency Hamming FEC is much more significant in some latency-sensitive applications such as BT, MG, SP, e.g., outperforms RS-FEC approximately 38%, 44%, and 50% in case of 200Gbps, 400Gbps and 1000Gbps links are used. Our approach reduces the average execution time by around 22-44% when compared to those in RS-FEC approach. We thus conclude that the use of lightweight low-latency FEC significantly improves the HPC system performance.

5.4 Summary

In this chapter, we present an approach to designing light-weight low-latency moderate error-proof FEC for HPC system. we optimized a high-bandwidth communication network to provide low-latency message transfer by using a low-latency error correction for HPC systems. Instead of conventional RS-FEC, we proposed to apply a lightweight low-latency low-reliable error-correction to HPC interconnection networks. The main advantage of the error-correction does not require a large latency overhead. We exploited a case study using the Hamming FEC with 256b/260b PCS code for HPC interconnection networks.

The discrete-event simulation results show that our approach reduces 70-80% and 22-44% of the average execution time of NAS Parallel Benchmarks when compared to conventional 100 Gbps network and high-bandwidth network with the conventional RS-FEC, respectively. By contrast, our approach does not ensure that all the application are perfectly executed and finish successfully, e.g., only 80% success as in LU benchmark with 10^{-9} of post-FEC BER. We thus conclude that our approach achieves good trade-offs between performance and quality-of-results in practice.

Although we evaluated our proposed network when executing NAS Parallel Benchmarks in this study, rich variety of parallel applications have intrinsic to tolerate bit flips, especially emerging AI/machine-learning and algorithm-based fault tolerance (ABFT) applications. Our future work is to extend our study for further performance tuning between these applications and our proposed network.

6 Integrated Interconnection Networks

6.1 Overview

In this chapter, we illustrate our best interconnection networks in which Distributed Shortcut Network topology, its custom routing, and moderate error-proof high-speed cables are integrated. As mentioned in Chapter 3, our DSN network topology design achieves a logarithmic diameter and have a short cable length that helps to reduce both switch latency and cable latency. We also apply the custom routing based on our interconnection network. The main purpose of using custom routing is to overcome the limitation of routing table size in the high-scale system. Although our approach gains a little switch-latency by avoiding the use of CAM/TCAM, e.g., 5ns, our custom routing does not ensure all the routing paths are minimal routing. Thus, the effect of our proposed routing to the network latency may not significant. We call these two proposals as ***cable-geometric*** approach.

Moreover, in order to improve the network throughput, high-speed optical cables are considered to integrate. However, this new kind of cables require a huge latency overhead for error correction for maintaining an error-free communication. In this context, we

proposed to use the moderate error-proof high-speed cables (***moderate error-proof*** approach). With this approach, nodes are connected via more than 100 Gbps link instead of 40 Gbps as in Chapter 3 and Chapter 4. By applying the proposed Hamming FEC, we maintain a small switch latency while enlarging the network bandwidth. This is an important advantage of our interconnection network design.

Aiming at improving the network performance, we integrate the two approaches with a clear purpose is that (1) cable-geometric approach for a low-latency and a low-cost and (2) moderate error-proof approach for a low-latency and a high-bandwidth. In the next section, we discuss the synergistic effect of two approaches in terms of network performance. We also compare our integrated network to the same-size counterpart interconnection networks.

6.2 Performance Evaluation

In this section, we use discrete-event simulation, i.e., SIMGRID simulation framework [81], to evaluate the performance of parallel benchmarks which based on Message Passing Interface (MPI) when executed on several of interconnection networks. We use the same method that mentioned in Chapter 4 and Chapter 5), i.e., the same physical layout and routing algorithm as mentioned in Chapter 4 and the same configuration of error correction mechanism as mentioned in Chapter 5. For all of the simulations, we use various network topologies of 256 switches with different switch delays and link bandwidths. We also set one host per switch with the total computation speed of 100 GFlops.

6.2.1 The Synergy of Our Approaches

In this study, we discuss the effect of our two approaches and the integration of them. The configurations of these approaches are presented in Table 6.1. Our discrete-event simulation results of FT application [1] using SimGrid is shown in Figure 6.1. The primary vertical axis (the columns) indicates the relative execution time of each approach normalized to that of the Baseline. Lower values are considered better. The secondary

[1] As mentioned in Chapter 4, our cable-geometric approach provides a good performance for the applications that have the collective communication. Thus, we choose FT as a preventative of such application.

Table 6.1: Configurations of different approaches.

	Name	Topology	Routing	Error Correction	Switch latency	Link bandwidth
Cable--geometric approach	Baseline	3-D Torus	Minimal routing	-	40 ns	40 Gbps
	Topology	2-D DSN	Minimal routing	-	45 ns	40 Gbps
	Routing	2-D DSN	Custom routing	-	40 ns	40 Gbps
Moderate error-proof approach	Future Baseline	3-D Torus	Minimal routing	RS-FEC	160 ns	400 Gbps
	Moderate Error	3-D Torus	Minimal routing	Hamming--FEC	60 ns	400 Gbps
Integrated approach	RS-FEC	2-D DSN	Minimal routing	RS-FEC	160 ns	400 Gbps
	Hamming -FEC	2-D DSN	Minimal routing	Hamming--FEC	65 ns	400 Gbps
	Proposal	2-D DSN	Custom routing	Hamming--FEC	60 ns	400 Gbps

vertical-axis (line serial) shows the relative operation per second. Higher values are considered better. Typically, the values of the line serial are the reversed number of the values of the corresponded columns.

We first recall the effect of our cable-geometric approach. We compare the performance of 3-D Torus with our network topology 2-D DSN where shortest routing paths are used (Baseline and Topology, respectively). We also consider the case of custom routing is implemented for 2-D DSN (Routing). Expectedly, the Topology outperforms the Baseline by taking the advantage of small network diameter, i.e., 2-D DSN has shorter diameter than 3-D Torus. Also, the Routing improves the performance 19% in the comparison with the Baseline. This result illustrates the small effect of the custom routing to the overall performance. Although our custom routing does not ensure all the routing paths are minimal, it gains a little switch-latency by avoiding the use of CAM/TCAM, e.g., 5ns. That helps to improve the performance, especially when the switch delay is small, i.e., less than 55ns as has already shown in Figure 4.7.

Secondly, we illustrate the effect of our moderate error-proof approach. In this

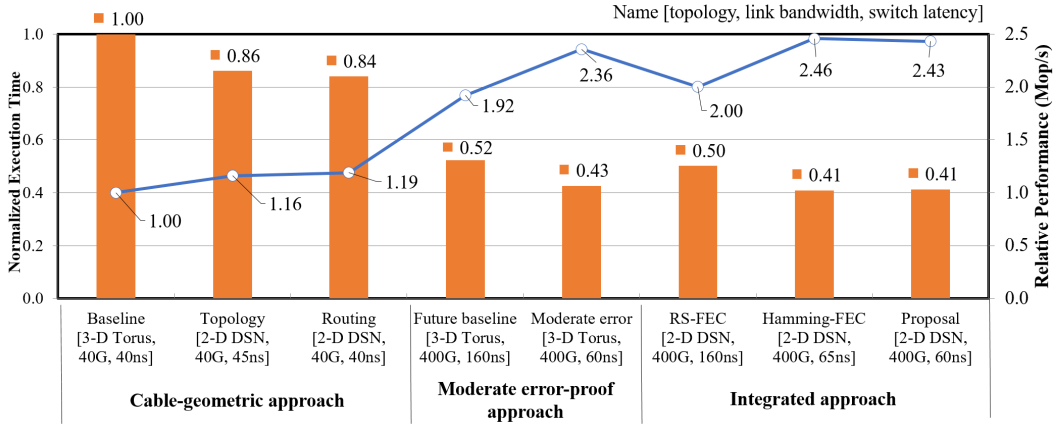


Figure 6.1: Performance of NAS Parallel - FT application with various approaches. Values are normalized to those of the baseline interconnection networks. Columns show the normalized execution time. The line serial shows the relative operation per second (secondary vertical-axis).

evaluation, besides the Baseline, we measure the performances of the cases that high-speed optical cables are integrated with RS-FEC and our approach (Future Baseline and Moderate Error, respectively). We set the bandwidths of all the links to 400Gbps. The Future Baseline maintain an error-free communication that requires a high switch latency (160ns per hop) while our approach (Moderate Error) only leads to a switch latency of 60ns per hop. The result in Figure 6.1 shows that increasing the link bandwidth, i.e., from 40 Gbps to 400 Gbps, significantly improve the network performance. For example, the Future Baseline and the Moderate Error approach reach 1.92 and 2.36 times faster than the Baseline, respectively. This result implies that the impact of our moderate error-proof is much larger than that of cable-geometric approach.

Finally, we evaluate the performance of the interconnection network that integrated our two approaches. We show the result of three combinations when high-bandwidth cables are used: (1) only network topology (RS-FEC), (2) network topology and our moderate error-approach (Hamming-FEC), and (3) network topology, custom routing, and our moderate error-approach (Proposal). The RS-FEC, Hamming-FEC, and Proposal outperform the Baseline 2.00, 2.46, and 2.43 respectively. Interestingly, the Proposal, i.e., the integration of three technologies, is worse than Hamming-FEC. It recalls the disadvantage of our custom routing, i.e., non-minimal routing. When we scarify the

Table 6.2: Interconnection network configuration

	Network topology	**Routing**	**Error Correction**	**Switch latency / Link bandwidth**
Baseline	3-D torus, degree-6	Minimal routing	CONV	100 ns 100 Gbps
3-D Torus	3-D Torus, degree-6	Minimal routing	RS-FEC	160 ns ≥ 200Gbps
RR-6	Random Ring, degree-6	Minimal routing	RS-FEC	160 ns ≥ 200Gbps
Dragonfly	Dragonfly, degree-11	Minimal routing	RS-FEC	160 ns ≥ 200Gbps
2-D DSN	2-D DSN, degree-6	Custom routing	Hamming-FEC	60 ns ≥ 200Gbps

switch latency in order to support the high-bandwidth links, it crosses the threshold of 55ns. In that case, the benefit that we gain from removing CAM/TCAM (custom routing instead of minimal routing) cannot tolerate the damage from the lengthened routing paths. However, custom routing is needed for bypassing the limitation of routing table size in the high-scale system.

Through these result, we state that our approaches are not synergy but have good effects independently. In the next section, we compare our integrated network to the same-size counterpart interconnection networks for analyzing the best interconnection network.

6.2.2 What is The Best Interconnection Network

We now measure the performance of the parallel applications executed on our proposed topology, where the custom routing is implemented for the communication between switches and the Hamming-FEC is used. The counterpart interconnection networks namely 3-D Torus, RR-6, and Dragonfly that implements the minimal routing taking the shortest-hop path and RS-FEC approach. We also compare with a Baseline interconnection network which comes up with only 100 Gbps link. Table 6.2 presents the configuration of our targeted interconnection network in this comparison. We evaluate the application performance with different link bandwidth, e.g. ≥ 200 Gbps for all the configuration excepts the Baseline. Our discrete-event simulation results using SimGrid

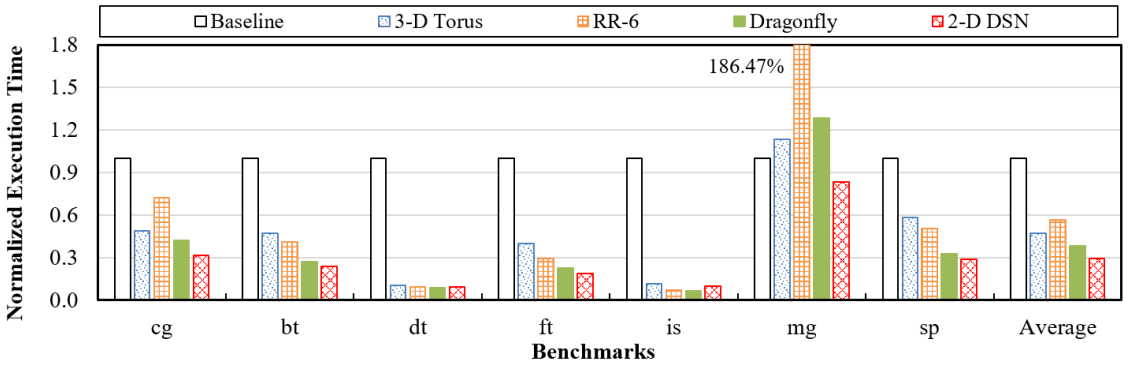
is shown in Figure 6.2. The y-axis indicates the relative execution time of each approach normalized to that of Baseline. Lower values are considered better.

The result shows that all the high-bandwidth interconnection network, i.e., 3-D Torus, RR-6, Dragonfly, and DSN, achieve a significant performance improvement compare to the Baseline. For example, our proposal 2-D DSN requires a shorter execution time compared to Baseline, only 63.87%, 41.26% and 29.20% on average for 200, 400, and 1000 Gbps configuration, respectively. The only exception for this improvement is the mg benchmark, the latency-sensitive application. This interesting result again implies the trade-off between latency and throughput as mentioned in Chapter 5. The performance improvement by using high bandwidth link is not enough to tolerate the performance sacrifice by high switch latency overhead.

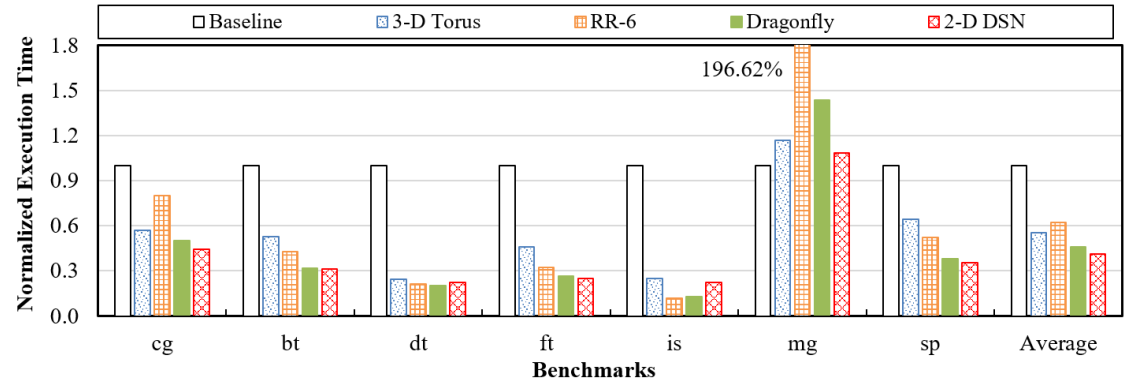
In the comparison with the same-size, same-degree 3-D Torus interconnection network, our 2-D DSN always have better performance due to its clear advantages of much lower communication latency with a smaller-diameter topology (Chapter 3) and a lower switch latency by avoiding FEC (discussed in Chapter 5). Similarly, our proposal interconnection network also outperforms random ring interconnection network RR-6. The graph analysis in Chapter 3 showed that our network topology has a similar diameter and average shortest path length (the hop count) to RR-6 but we reach the better performance by reducing the delay at each hop, i.e., switch latency by usage custom routing and FEC-avoidance.

Comparing with Dragonfly, the state-of-the-art interconnection network that is frequently used in the recent supercomputers, our 2-D DSN has lower average execution time for 400 and 1000 Gbps link, e.g., approximately 5% and 10% lower than that of Dragonfly respectively. Although Dragonfly could provide lower network diameter, our approach gains much performance from the moderate error-proof approach, i.e., providing a lower switch latency. In detail, 2-D DSN is better for applications that have collective communications across the system (e.g., CG, MG, FT) or for benchmarks that generate a large number of neighboring communications (e.g., SP and BT). However, for throughput-sensitive benchmarks such as IS and DT, or in case of a low link bandwidth, e.g., 200 Gbps, the Dragonfly outperforms our proposal.

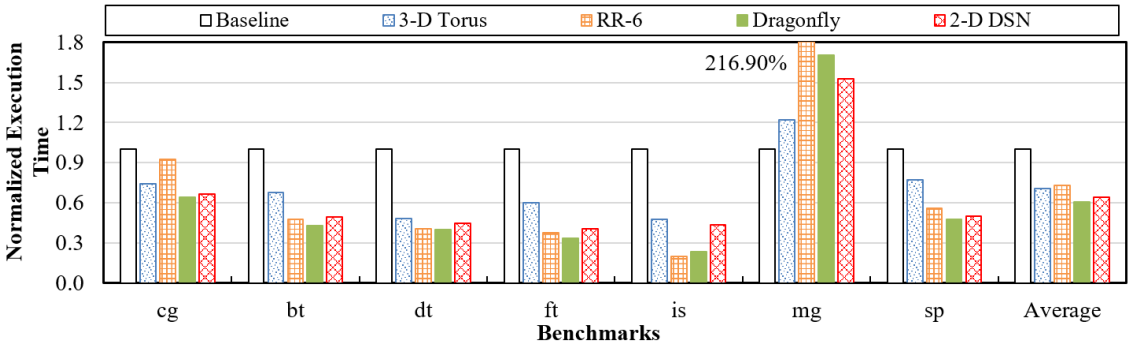
In summary, our proposed interconnection network (2-D DSN) achieves a better performance than the same-degree Torus by having a smaller diameter due to cable-geometric approach. Our proposal also outperforms the same-degree random network (RR-6), and the high-degree network (Dragonfly) because our approach provides a



(a) 1000 Gbps



(b) 400 Gbps



(c) 200 Gbps

Figure 6.2: Application performance evaluated using NAS Parallel. Values are normalized to those of the baseline interconnection networks.

lower switch latency with the use of Hamming-FEC instead of RS-FEC (moderate error-proof approach). Interestingly, in theory, the moderate error-proof approach can be applied to the high-degree Dragonfly network (or even random network). In that case, the performance of Dragonfly will be better than our proposal because it reaches a much smaller diameter. However, let us remind that for a high-scale high-degree network, the cable length becomes extremely long as discussed in Section 3.4 that leads to an unacceptable cost (especially when the cost of high-speed cables are still high). In the last word, our proposed interconnection networks have strengths comparable to high-degree network, e.g., Dragonfly, but have clear advantages in saving cables.

7 Conclusion

7.1 Summary of the Dissertation

Interconnection network design is one of the most critical concerns for highly parallel systems, e.g., supercomputer and data center. In this dissertation, we have proposed a low latency interconnection network towards Exascale and Big-Data challenges which has been motivating academic research and industry technology developments for decades. Our approach is a combination of a short-diameter network topology with its low-cost physical layout, a custom routing on this topology that avoids latency overhead for routing table usage, and a low-latency error control mechanism for the usage of high-speed optical links.

In the first part of this dissertation, 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 supernodes", combined with a different design philosophy learned from observing small-world networks. By taking the full advantages of the logarithmic

diameter and the structured topological properties, e.g., low-degree, non-random, our DSNs achieve both low switch latency and low cable latency. We also discussed the extension of DSNs for supporting incremental expandability requirement.

In the second part, we proposed a custom routing algorithm on DSN topology with the avoidance of routing table for low latency overhead. As with a typical non-random topology, we showed that it is possible to exploit the structure of our DSN topologies to create a routing algorithm with a natural routing logic at each switch which 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.

In the next part, we have focused on reducing a latency overhead of error correction component, i.e., FEC, which is the latency bottle-neck when the future high-speed link is used, i.e., 800Gbps link. Our analysis found that parallel applications such as NAS-NPB do not require the bit error rate as low as 10^{-15} (error-free communication). Thus, instead of RS-FEC, we proposed to use a light-weight low-latency FEC such as Hamming Code for error correction/detection code. Our simulation result showed that the network performance is significantly improved while the application accuracies are maintained.

In the remaining, we combine the above approaches for our concrete interconnection network. Our approaches are not synergy but have good effects independently. We show that our low-degree, cable-geometric moderate error-proof approach achieved a better performance compared with the same-degree counter-part network such as Torus or Random. Our proposed interconnection networks have strengths somewhat comparable to high-degree network, e.g., Dragonfly, but have clear advantages in saving cables.

7.2 Future Direction

In recent decades, several technologies have been discussed for increasing the performance of an HPC system including [87]:

1 Multi-core approach: designing and replicating a small number of complex cores.

2 Many-core approach: connecting a large number of simple, low power cores.

3 GPU/ accelerator: using highly specialized processors from gaming/graphics market space such as NVidia, Fermi, Cell, Intel Phi, etc.

Table 7.1: Leading technologies for HPC system design.

Rank	System	Multicore	Manycore/ Embedded	GPU/ Accelerator
1	Sunway TaihuLight			✓
2	Tianhe-2 (MilkyWay-2)			✓
3	Piz Daint			✓
4	Titan			✓
5	Sequoia		✓	
6	Cori		✓	
7	Oakforest-PACS		✓	
8	K computer	✓		
9	Mira		✓	
10	Trinity		✓	

In this dissertation, we target on the many-core system. Our proposed approaches achieve good impacts in not only low-latency for the high-performance purpose but also other requirements of interconnection network design such as high-bandwidth, cost constraints, incremental expandability, and physical constraints. We now discuss the advantages as well as disadvantages of our proposed approaches for the future HPC systems design.

The cable-geometric approach (DSNs topology in Chapter 3 and its custom routing algorithm in Chapter 4) focus on low-latency, physical constraints, and incremental expandability requirements. Although our proposal achieves the lowest network latency (highest performance) in the comparison with the same low-degree network topology, e.g., Torus, Random network, it still does not outperform the high-degree network such as Dragonfly or Slimfly network. This fact implies that our network design does not play the leading role in the high-speed HPC system design, i.e., exascale challenges. However, when the higher demands of system performance in future require a larger number of network size in the many-core approach, cost-efficient performance, e.g., operation per second per $ or data movement (bit/s) per $ becomes the critical concerns. Thus, our networks are designed with a clear purpose in finding a good trade-off between the performance and the network cost (or balancing the network diameter and the cable length).

Our cable-geometric approach is suitable for the homogeneous system with a

collective communication or uniform random traffic patterns. However, GPU and accelerator approach becomes a hot trend in recent years that leading the top-speed supercomputers as shown in Table 7.1. Interconnection network design also has to consider the heterogeneous computing network which includes both CPUs and GPUs. The performance of our approaches in such kind of area is still a reasonable question for a future work.

Not only network latency, but also network throughput is an important factor for the performance of HPC systems. In order to achieve high network throughput, besides network topology design, increasing the bandwidth of cables is also an essential approach. The existing top-speed supercomputer is designed with approximately 100G links, such as 40 and 10 Gigabit Ethernet in Cray XC50 [88] that used in the top-3 Piz Daint system [3]. In the top-2 Tianhe system [21], eight high-speed serial lanes, running up to 14 Gbps per lane are integrated into one network port while it is 16Gbps in the top-1 Sunway system [89]. For the future interconnection network design, research on how to apply the Gigabit Ethernet or Terabit Ethernet in HPC system are also needed. Our moderate error-proof approach in Chapter 5 targets this future requirement with a strong point of maintaining the low switch latency. Unfortunately, this approach does not guarantee an error-free communication that cannot apply for some parallel applications.

7.3 Future Work

In Chapter 4, we concluded that the performance of our network topology depends on the parallel application, specifically on communication patterns. Recently, new applications such as AI/machine-learning applications have been strongly outbreaking leads to the new kind of traffic patterns. Thus, how to optimize our DSN for specific applications to further improve the performance would be an interesting issue.

In addition, our moderate error-proof approach is based on a strong assumption that our targeted applications may not require the error-free communication, e.g., the parallel NPB application with bit error rate of 10^{-9}, thus we can loosening the error correction capacity of FEC to gain the latency overhead. However, how this approach capable of the new generation of high-performance applications, i.e., AI application, is a reasonable concern. On one hand, regarding to light-weight low-latency approach, a critical question is to find a better error-correction capacity code but still the small-latency overhead. On the other hand, for the FEC-avoid approach, investigate the mechanism of uneven

protection for different bits in a word also is one of our concerns. This approach is based on the observation that has been already used in several hardware components consumption such as RAM [90] and wireless communication system, i.e., the importance of bits in application's data are not equivalent. For example, IEEE 754 standard for precision floating-point numbers consists of the sign, exponent and fraction bits. An error (hereafter by a bit flip) in the sign and the exponent has likely a high impact on the numerical value of the floating point number. By contrast, a bit flip in the fraction bits has a smaller impact on the numerical value. Thus, a co-design between application and network is needed for such kind of protection approach.

Bibliography

[1] W. J. Dally and B. P. Towles, *Principles and practices of interconnection networks*. Elsevier, 2004.

[2] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller, *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep*, vol. 15, 2008.

[3]

[4] J. Escudero-Sahuquillo and P. J. Garcia, "High-performance interconnection networks in the exascale and big-data era," *The Journal of Supercomputing*, vol. 72, pp. 4415–4417, Dec 2016.

[5] K. Scott Hemmert et al, "Report on Institute for Advanced Architectures and Algorithms, Interconnection Networks Workshop 2008." http://ft.ornl.gov/pubs-archive/iaa-ic-2008-workshop-report-final.pdf.

[6] A. Subcommittee, "Top ten exascale research challenges," *US Department Of Energy Report*, 2014.

[7] B.Dally, "Issues in The Design of Exascale Networks." http://hipineb.i3a.info/hipineb2017/wp-content/uploads/sites/6/2017/05/slides_dally.pdf, February 2017.

[8] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection networks: an engineering approach*. Morgan Kaufmann, 2003.

[9] W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 775–785, 1990.

[10] 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. of the ACM/IEEE International Symposium on Computer Architecture, (ISCA)*, pp. 1–12, June. 2014.

[11] CrayXT5 Supercomputer. http://www.cray.com/.

[12] DOE/SC/Oak Ridge National Laboratory, "Titan Cray XK7." https://www.olcf.ornl.gov/computing-resources/titan-cray-xk7/.

[13] D. Chen, N. A. Eisley, P. Heidelberger, R. M. Senger, Y. Sugawara, S. Kumar, V. Salapura, D. L. Satterfield, B. Steinmacher-Burow, and J. J. Parker, "The ibm blue gene/q interconnection network and message unit," in *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for*, pp. 1–10, IEEE, 2011.

[14] P. Boyle, "The bluegene/q supercomputer," *Proceedings of Science (PoS), The 30 International Symposium on Lattice Field Theory - Lattice 2012*, p. 020, 2012.

[15] Y. Ajima, S. Sumimoto, and T. Shimizu, "Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers," *IEEE Computer*, vol. 42, pp. 36–40, 2009.

[16] M. R. Samatham and D. K. Pradhan, "The de bruijn multiprocessor network: a versatile parallel processing and sorting network for vlsi," *IEEE Transactions on Computers*, vol. 38, no. 4, pp. 567–581, 1989.

[17] S. B. Akers and B. Krishnamurthy, "A group-theoretic model for symmetric interconnection networks," *IEEE transactions on Computers*, vol. 38, no. 4, pp. 555–566, 1989.

[18] C. Wei-Kuo and C. Rong-Jaye, "The (n, k)-star graph: A generalized star graph," *Information Processing Letters*, vol. 56, no. 5, pp. 259–264, 1995.

[19] W. Dally, "Network and processor architecture for message-driven computers," in *VLSI and Parallel Computation*, pp. 140–222, Morgan Kaufmann Publishers Inc., 1990.

[20] F. Petrini and M. Vanneschi, "k-ary n-trees: High performance networks for massively parallel architectures," in *Parallel Processing Symposium, 1997. Proceedings., 11th International*, pp. 87–93, IEEE, 1997.

[21] X.-K. Liao, Z.-B. Pang, K.-F. Wang, Y.-T. Lu, M. Xie, J. Xia, D.-Z. Dong, and G. Suo, "High performance interconnect network for tianhe system," *Journal of Computer Science and Technology*, vol. 30, no. 2, p. 259, 2015.

[22] T. Boku, Center for Computational Sciences, University of Tsukuba. https://www.ccs.tsukuba.ac.jp/wp-content/uploads/sites/14/2016/05/boku.pdf, May 2016.

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

[24] 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)*, pp. 177–188, 2012.

[25] B. Bollobás and F. R. K. Chung, "The Diameter of a Cycle Plus a Random Matching," *SIAM J. Discrete Math.*, vol. 1, no. 3, pp. 328–333, 1988.

[26] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *Proc. of the 32nd annual ACM symposium on Theory of computing (STOC)*, pp. 163–170, 2000.

[27] C. Martel and V. Nguyen, "Analyzing Kleinberg's (and other) smallworld models," in *Proceedings of the 23rd annual ACM symposium on Principles of distributed computing (PODC)*, pp. 179–188, July. 2004.

[28] J.-Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *Proceedings of the 2nd ACM Symposium on Cloud Computing (SOCC11)*, (New York, NY, USA), pp. 2:1–2:13, ACM, 2011.

[29] J. Kleinberg, "The small-world phenomenon and decentralized search," *SIAM News*, vol. 37, April a short essay as part of Math Awareness Month 2004.

[30] W. T. Neil Rasmussen, "Data Center Projects: Establishing a Floor Plan." http://www.apcmedia.com/salestools/VAVR-6KYMZ7/VAVR-6KYMZ7_R2_EN.pdf?sdirect=true, 2007.

[31] P. Hannaford, "Ten Cooling Solutions to Support High-Density Server Deployment." http://www.apcmedia.com/salestools/TDOY-5U362W/TDOY-5U362W_R5_EN.pdf, 2007.

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

[33] 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)*, pp. 126–137, 2007.

[34] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," in *Proc. of the International Symposium on Computer Architecture (ISCA)*, pp. 77–88, 2008.

[35] 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)*, pp. 225–238, 2012.

[36] M. Besta and T. Hoefler, "Slim Fly: A Cost Effective Low-Diameter Network Topology," in *Proc. of the IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis (SC14)*, pp. 348–359, Nov. 2014.

[37] I. Fujiwara, M. Koibuchi, H. Matsutani, and H. Casanova, "Skywalk: A topology for hpc networks with low-delay switches," in *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pp. 263–272, IEEE, 2014.

[38] I. Fujiwara, M. Koibuchi, and H. Casanova, "Cabinet Layout Optimization of Supercomputer Topologies for Shorter Cable Length," in *Proc. of International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pp. 227–232, Dec. 2012.

[39] HP, "An Algorithmic Approach to Datacenter Cabling." http://www.labs.hpe.com/techreports/2015/HPL-2015-8.pdf, 2015.

[40] IEEE 802.3 Ethernet working group. http://www.ieee802.org/3/.

[41] IEEE P802.3bs 200 Gb/s and 400 Gb/s Ethernet Task Force. http://www.ieee802.org/3/bs/.

[42] J. Wei, Q. Cheng, R. V. Penty, I. H. White, and D. G. Cunningham, "400 gigabit ethernet using advanced modulation formats: Performance, complexity, and power dissipation," *IEEE Communications Magazine*, vol. 53, pp. 182–189, Feb 2015.

[43] G. Tzimpragos, C. Kachris, I. B. Djordjevic, M. Cvijetic, D. Soudris, and I. Tomkos, "A survey on fec codes for 100 g and beyond optical networks," *IEEE Communications Surveys Tutorials*, vol. 18, pp. 209–221, Firstquarter 2016.

[44] L. Schmalen, A. J. de Lind van Wijngaarden, and S. T. Brink, "Forward error correction in optical core and optical access networks," *Bell Labs Technical Journal*, vol. 18, pp. 39–66, Dec 2013.

[45] IEEE Standard 802.3bj, "Transcoding/fec options and trade-offs for 100 gb/s backplane and copper cable." http://www.ieee802.org/3/bj/public/nov11/cideciyan_01a_1111.pdf, Nov. 2011.

[46] IEEE P802.3bj 100 Gb/s Backplane and Copper Cable Task Force, "Fec options." http://www.ieee802.org/3/bj/public/jan12/gustlin_01_0112.pdf.

[47] B. P. Smith and F. R. Kschischang, "Future prospects for fec in fiber-optic communications," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 16, pp. 1245–1257, Sept 2010.

[48] J. Tomkins, "Interconnects: A Buyers Point of View." ACS Workshop, 2007.

[49] Combinatorics Wiki, "The degree diameter problem for general graphs." http://combinatoricswiki.org/wiki/The_Degree_Diameter_Problem_for_General_Graphs.

[50] M. Koibuchi, I. Fujiwara, H. Matsutani, and H. Casanova, "Layout-conscious Random Topologies for HPC Off-chip Interconnects," in *Proc. of IEEE International Symposium on High Performance Computer Architecture (HPCA2013)*, pp. 484–495, Feb. 2013.

[51] 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. of the ACM 2001 SIGCOMM*, pp. 149–160, 2001.

[52] T. T. Nguyen, I. Fujiwara, and M. Koibuchi, "A diagonal cabling approach to data center and hpc systems," in *Proceedings of the Seventh Symposium on Information and Communication Technology*, pp. 265–271, ACM, 2016.

[53] "METIS - Serial Graph Partitioning and Fill-reducing Matrix Ordering." http://glaros.dtc.umn.edu/gkhome/metis/metis/overview.

[54] GraphGolf: The Order/degree Problem Competition. http://research.nii.ac.jp/graphgolf/.

[55] 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 Distributed Systems.*, vol. 23, no. 3, pp. 405–425, 2012.

[56] M. Benito, E. Vallejo, and R. Beivide, "On the use of commodity ethernet technology in exascale hpc systems," *2015 IEEE 22nd International Conference on High Performance Computing (HiPC)*, pp. 254–263, 2015.

[57] L. Kleinrock and F. Kamoun, "Hierarchical routing for large networks performance evaluation and optimization," *Computer Networks (1976)*, vol. 1, no. 3, pp. 155–174, 1977.

[58] C. Gavoille and M. Gengler, "Space-efficiency for routing schemes of stretch factor three," *Journal of Parallel and Distributed Computing*, vol. 61, no. 5, pp. 679–687, 2001.

[59] M. Thorup and U. Zwick, "Compact routing schemes," in *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pp. 1–10, ACM, 2001.

[60] L. J. Cowen, "Compact routing with minimum stretch," *Journal of Algorithms*, vol. 38, no. 1, pp. 170–183, 2001.

[61] D. Krioukov, k. c. claffy, K. Fall, and A. Brady, "On compact routing for the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 41–52, July 2007.

[62] M. Enachescu, M. Wang, and A. Goel, "Reducing Maximum Stretch in Compact Routing," in *INFOCOM 2008. 27th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies*, pp. 336–340, 2008.

[63] H. Liu, "Routing table compaction in ternary cam," *IEEE Micro*, vol. 22, no. 1, pp. 58–64, 2002.

[64] H. Liu, "Reducing routing table size using ternary-cam," in *Hot Interconnects 9, 2001.*, pp. 69–73, IEEE, 2001.

[65] B. Agrawal and T. Sherwood, "Ternary cam power and delay model: Extensions and uses," *IEEE transactions on very large scale integration (VLSI) systems*, vol. 16, no. 5, pp. 554–564, 2008.

[66] M. Thorup and U. Zwick, "Approximate Distance Oracles," in *Proceedings of the Thirty-third Annual ACM Symposium on Theory of Computing*, STOC '01, pp. 183–192, ACM, 2001.

[67] H. Casanova, A. Giersch, A. Legrand, M. Quinson, and F. Suter, "Versatile, Scalable, and Accurate Simulation of Distributed Appl ications and Platforms," *Journal of Parallel and Distributed Computing*, vol. 74, no. 10, pp. 2899–2917, 2014.

[68] NASA Advanced Supercomputing Division, "The NAS Parallel Benchmarks." https://www.nas.nasa.gov/publications/npb.html.

[69] D. H. Bailey, "Title: The nas parallel benchmarks."

[70] D. H. Bailey, E. Barszcz, J. T. Barton, D. S. Browning, R. L. Carter, L. Dagum, R. A. Fatoohi, P. O. Frederickson, T. A. Lasinski, R. S. Schreiber, *et al.*, "The nas parallel benchmarks," *The International Journal of Supercomputing Applications*, vol. 5, no. 3, pp. 63–73, 1991.

[71] T. H. B. Himeno, R. http://accc.riken.jp/2444.htm, 2014.

[72] "BigDataBench." http://prof.ict.ac.cn/BigDataBench, 2016.

[73] F. Chaix, I. Fujiwara, and M. Koibuchi, "Suitability of the Random Topology for HPC Applications," in *2016 24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 301–304, Feb 2016.

[74] "MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE." http://mvapich.cse.ohio-state.edu/.

[75] M. Andrewartha, B. Booth, and C. Roth, "Feasibility and Rationale for 3m no-FEC server and switch DAC." http://www.ieee802.org/3/by/public/Sept15/andrewartha_3by_01a_0915.pdf, Sept. 2015.

[76] IEEE P802.3ca 100G-EPON Task Force, "Fec code for 25/50/100g epon." http://www.ieee802.org/3/ca/email/pdfUGjlG0uq7U.pdf, March 2017.

[77] D. Fujiki, K. Ishii, I. Fujiwara, H. Matsutani, H. Amano, H. Casanova, and M. Koibuchi, "High-bandwidth low-latency approximate interconnection networks," *Proc. of the 23rd IEEE International Symposium on High-Performance Computer Architecture (HPCA'17)*, Feb 2017.

[78] M. N. Sakib and O. Liboiron-Ladouceur, "A study of error correction codes for pam signals in data center applications," *IEEE Photonics Technology Letters*, vol. 25, pp. 2274–2277, Dec 2013.

[79] 400 Gb/s Ethernet Study Group, "Reconsider pcs coding for 400gbe." http://www.ieee802.org/3/400GSG/public/13_09/song_400_01_0913.pdf.

[80] M. Teshima, S. Kobayashi, T. Yamamoto, and O. Ishida, "Bit-error-tolerant (512*n)b/(513*n+1)b code for 40gb/s and 100gb/s ethernet transport," in *IEEE INFOCOM Workshops 2008*, pp. 1–6, April 2008.

[81] SimGrid: Versatile Simulation of Distributed Systems. http://simgrid.gforge.inria.fr/.

[82] Q. Hongyi, "On the Performance of Concatenated Reed-Solomon, Convolutional and Parity Check Codes for BWA Applications." http://www.ieee802.org/16/tg1/phy/contrib/802161pc-00_37.pdf, June 2000.

[83] I. P. G. E. S. Group, "802.3bj fec overview and status." http://www.ieee802.org/3/bm/public/sep12/gustlin_01_0912_optx.pdf, 2012.

[84] I. P. G. E. S. Group, "400gbe pcs architectural options." http://www.ieee802.org/3/400GSG/public/13_07/gustlin_400_02_0713.pdf, 2013.

[85] K. Azadet and M. Yu, "Forward error correction (fec) techniques for optical communications," in *IEEE 802. 3 High-Speed Study Group, Plenary Meeting*, 1999.

[86] B. Chen, X. Zhang, and Z. Wang, "Error correction for multi-level nand flash memory using reed-solomon codes," in *Signal Processing Systems, 2008. SiPS 2008. IEEE Workshop on*, pp. 94–99, IEEE, 2008.

[87] S. Horst, "Why we need exascale and why we wont get there by 2020," in *Optical Interconnects Conference, Santa Fe, New Mexico*, 2013.

[88] Cray, "Cray XC50 Supercomputer." https://www.cray.com/sites/default/files/Cray-XC50-NVIDIA-Tesla-P100-GPU-Accelerator-Blade.pdf.

[89] H. Fu, J. Liao, J. Yang, L. Wang, Z. Song, X. Huang, C. Yang, W. Xue, F. Liu, F. Qiao, *et al.*, "The sunway taihulight supercomputer: system and applications," *Science China Information Sciences*, vol. 59, no. 7, p. 072001, 2016.

[90] D. Li, Z. Chen, P. Wu, and J. S. Vetter, "Rethinking algorithm-based fault tolerance with a cooperative software-hardware approach," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, p. 44, ACM, 2013.

List of Publications

Journal Papers

1 N. T. Truong, I. Fujiwara, M. Koibuchi, K.-V. Nguyen,"Distributed Shortcut Networks: Low-latency Low-degree Non-random Topologies Targeting the Diameter & Cable Length Trade-of", IEEE Transactions on Parallel and Distributed Systems. 2017 Apr 1;28(4):989-1001.

2 N. T. Truong, V. K. Nguyen, I. Fujiwara, and M. Koibuchi, "Layout-conscious Expandable Topology for Low-degree Interconnection Networks", IEICE Transactions on Information and Systems, Vol. E99-D, no 5, May 2016, pp1275-1284.

Referred Proceedings

3 N. T. Truong, M. Koibuchi, "Cable-geometric error-prone approach for low-latency interconnection networks", in Proceedings of the 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2017), 699-702, May 2017, Marid, Spain.

4 N. T. Truong, I. Fujiwara and M. Koibuchi, "A Diagonal Cabling Approach to Datacenter and HPC Systems" in the Proceedings of 7th ACM international symposium on information and communication technology (ACM SoICT 2016), Ho Chi Minh, Vietnam, 2016, pp. 265-271.

5 T. C. Kieu, K. V. Nguyen, N. T. Truong, I. Fujiwara and M. Koibuchi,"An Interconnection Network Exploiting Trade-Off between Routing Table Size and Path

Length," 2016 Fourth International Symposium on Computing and Networking (CANDAR), Hiroshima, 2016, pp. 666-670.