Swap-and-randomize: A Method for Building Low-latency HPC Interconnects

Ikki Fujiwara, *Member, IEEE,* Michihiro Koibuchi, *Member, IEEE,* Hiroki Matsutani, *Member, IEEE,* and Henri Casanova, *Member, IEEE*

Abstract—Random network topologies have been proposed to create low-diameter, low-latency interconnection networks in large-scale computing systems. However, these topologies are difficult to deploy in practice, especially when re-designing existing systems, because they lead to increased total cable length and cable packaging complexity. In this work we propose a new method for creating random topologies without increasing cable length: randomly swap link endpoints in a non-random topology that is already deployed across several cabinets in a machine room. We quantitatively evaluate topologies created in this manner using both graph analysis and cycle-accurate network simulation, including comparisons with non-random topologies and previously-proposed random topologies.

Index Terms—Network topologies, cabinet layout, interconnection networks, high-performance computing.

1 Introduction

Large parallel applications to be deployed on next generation High Performance Computing (HPC) systems will suffer from communication latencies that could reach hundreds of nanoseconds [1], [2]. There is thus a strong need for developing low-latency networks for these systems. Switch delays (e.g., about 100 nanoseconds in InfiniBand QDR) are large compared to the wire and flit injection delays even including serial and parallel converters. To achieve low latency, a topology of switches should thus have low diameter and low average shortest path length, both measured in numbers of switch hops.

Traditional topologies have regular structures that can match application communication patterns. For example, lattice communication, which occurs in many scientific parallel applications, fits naturally to n-dimensional torus interconnection networks [3], [4]. However, these topologies have relatively high average shortest path length and high maximum shortest path length, or diameter. As a result, latency-sensitive parallel applications that do not map well to a regular structure typically experience poor performance. These include datacenter applications with dynamic workload, and irregular parallel applications with non-deterministic and/or complex communication patterns. For these applications, it is difficult to compute an efficient mapping of the processes to the compute nodes in a structured topology. Another drawback of traditional topologies is that their regular structures strictly define network size (e.g., k^n vertices in a k-ary n-cube topology) even though the scale of a system should be determined based on electrical power budget, surface area, and cost. Furthermore, additional mechanisms must often be used as part of routing algorithms so as to maintain topological structure in the face of network component failures [3], [4]. Also, a potentially large number of redundant backup cables may need to be installed at deployment.

Random topologies have been proposed to alleviate the above drawbacks [5]–[7]. Randomness affords low diameter and average shortest path lengths, so that good performance can be achieved by arbitrarily mapping the processes of irregular applications to compute nodes. Also, random topologies can be generated for any network size and can also be built incrementally. Finally, no or fewer backup cables are needed as these topologies are inherently fault-tolerant. A practical concern is the long cable length in physical deployments [6], [7]. This is because these random topologies use "random shortcuts" to bypass switches between possibly distant cabinets in a machine room. Total cable length already reaches astronomical proportions in deployed systems that use non-random network topologies (∼2,000km for the first generation Earth Simulator [8] and ∼1,000km for the K-computer [4]). The use of random shortcuts further increases cable length, and thus cost, significantly.

In this work, we address the above cable length concern by proposing a method, which we term *permutation*, for generating random topologies and their physical layouts [9]. Using a known good physical layout for a non-random topology as a starting

- *I. Fujiwara and M. Koibuchi are with National Institute of Informatics, 2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN 101-8430. M. Koibuchi is also with the Graduate University for Advanced Studies (SOKENDAI).*
  E-mail: {koibuchi,ikki}@nii.ac.jp
- *H. Matsutani is with Keio University, 3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa, JAPAN 223-8522.*
  E-mail: matutani@arc.ics.keio.ac.jp
- *H. Casanova is with the University of Hawai'i at Manoa, 1680 East-West Road, Honolulu, HI 96822, U.S.A.*
  E-mail: henric@hawaii.edu

point, permutation swaps endpoints between pairs of links in a way that conserves cable length. An advantage of this approach is that it can be applied to systems that are already deployed in a machine room: swap the endpoints of some pairs of physical links and update the routing tables. Our main finding is that a permuted topology has identical or better performance properties than its non-random counterpart (lower diameter, lower latency, identical bisection bandwidth, identical or higher throughput). The performance improvement is admittedly lower than that achieved by previously proposed random shortcut topologies [7], but it comes at no increase in cable length whatsoever.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 details our topology permutation methods. Section 4 describes the topologies we consider, along with their layouts in physical cabinets. Sections 5, 6, and 7 evaluate our proposed topologies and compare them to competitors in terms of graph properties, latency and throughput, and cable length, respectively. Section 8 concludes with a summary of our main findings.

2 Related Work

2.1 Topologies of HPC Systems

A few topologies have been traditionally used to interconnect compute nodes in HPC systems, and these topologies are also used in large systems to interconnect high-radix switches [10]. In *direct topologies* each switch connects directly to compute nodes as well as to other switches. Popular direct topologies are k-ary n-cubes, which include tori, meshes, and hypercubes. Each topology leads to a different trade-off between degree and diameter. All these topologies are *regular*, meaning that all switches have the same degree (i.e., each switch has the same fixed number of links to other switches).

Indirect topologies, i.e., topologies in which some switches are connected only to other switches, have also been proposed. They have low diameter at the expense of larger numbers of switches when compared to direct topologies. The best known indirect topologies are Fat trees, Clos network and related multi-stage interconnection networks (MINs) such as the Omega and Butterfly networks. MINs have uniform access latency, and they use different schemes by which endpoints are "shuffled" deterministically at each stage, so that re-arrangeable or non-blocking data transfers are possible.

2.2 Graphs with Low Diameter

The problem of maximizing the number of vertices in a graph for given diameter and degree has been studied by graph theoreticians for decades, striving to approach the famous Moore bound [11]. Several low-diameter large graphs have been proposed for interconnection networks [12], [13]. Another approach is to augment known topologies with additional edges in a hierarchical manner [14]–[20]. Most of these graphs are constructed for fixed numbers of switches and/or strive to achieve a switch degree as low as possible. As a result, switch degree is often non-uniform, which would complexify network deployment.

Another type of known low-diameter graphs are random graphs [21], recently popularized by the study of real-world networks with *small-world* properties [22], [23]. Random networks have been proposed for designing data centers with increased expandability, fault tolerance, and throughput capabilities [5], [6]. In [7] random networks have been proposed for low-latency HPC interconnects because the use of random shortcuts drastically decreases graph diameter and average shortest path length when compared to same-degree non-random topologies that are traditionally used in HPC systems. A drawback of random networks is that topology-agnostic routing must be used, requiring routing-table or source-routing implementations because the network does not have a simple structure. We note that 83.8% of the systems on the November 2013 Top500 supercomputer list are based on Ethernet or InfiniBand, both of which use routing tables in their switches. However, the remaining 16.2% usually use custom routing on non-random topologies. In this context, random networks, including the topology proposed in this work, cannot be used directly for these systems (i.e., additional mechanisms would be needed to support routing). Another documented drawback of random networks, whether for a data center that uses a top-of-rack switch for inter-cabinet connection or an HPC system in which a large number of switches are connected by inter-cabinet links [24], is that the total cable length and the complexity of cable packaging are increased when compared to non-random networks with similar numbers of network links [6], [7].

2.3 Cabinet Layout of Topologies

The layout of cabinets on a floorplan and the assignment of switches to these cabinets are important concerns when designing large systems because they affects costs [24], [25]. The salient features of a layout include cabinet footprint, number of compute nodes and switches per cabinet, and cabinet spacing. For instance, in the case of the Cray BlackWidow system, it is estimated that each cabinet has a $0.57\text{m}\times1.44\text{m}$ footprint, with 128 nodes per cabinet, and that the nodes/m^2 density should be 75 [25]. A common approach is to specify the widths of the aisles between rows of cabinets. The ANSI/TIA/EIA-942 standard recommends site layouts with alternating cold and hot aisles with width at least 4ft and 2ft, respectively.

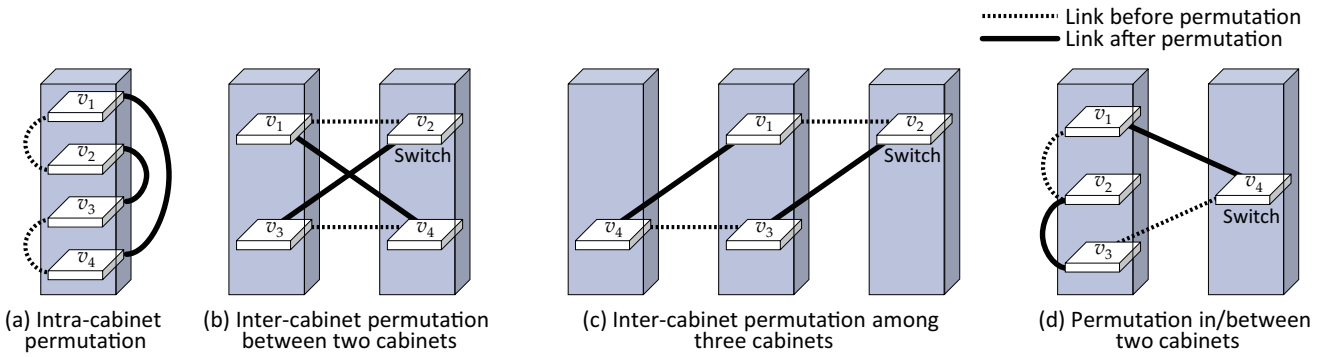

Fig. 1. Topology permutation methods. *Two-cab permutation* invokes (a) and then (b). *Three-cab permutation* invokes (a) and then (b) ∪ (c). *One-step permutation* invokes (a) ∪ (b) ∪ (d).

A similar specification is found in [26]. In this work we assume that some 2-D physical layout of cabinets has been determined to comply with the power/heat constraints of the system to be deployed.

The topologies used traditionally in HPC systems exhibit both highly regular structures and low degrees (e.g., the 3-D torus in BlueGene/L). As a result, they map naturally to a simple 2-D grid-like cabinet layout with low (or even optimally low) total cable length. This is no longer the case for high-degree topologies, and especially topologies that use random shortcuts. System designers are thus faced with the difficult task of mapping switches to a physical layout so as to reduce total cable length. In [27] a metaheuristics-based scheme is proposed to compute such a mapping, but the resulting total cable length is still much larger than that of non-random topologies. In addition, many cables reach a large length, which is problematic because the cost of the cabling medium increases with the length (e.g., for InfiniBand the typical maximum length of passive copper is 10m, while embedded optical is 100m [28]). Two options were explored in our previous work [9] to reduce cable lengths. The first option consists in generating random shortcut links that are constrained to connect switches that are not too distant. The second option is the topology permutation approach studied in this work. The results therein show topology permutation to be superior to the first option. In this work we build on these results, proposing an alternate link endpoint swapping method, applying topology permutation to indirect topologies, and comparing permuted topologies to non-random high-degree topologies.

3 Topology Permutation

3.1 Direct Networks

Consider an arbitrary physical layout of cabinets on a floorplan, so that each cabinet contains the same number of switches (and possibly compute nodes connected to these switches). The switches are interconnected in some traditional non-random topology that maps well to the cabinet layout in terms of total cable length. We use the notation $x \leftrightarrow y$ to denote a link between switch x and switch y. We define two permutation methods, namely *two-cab permutation* and *three-cab permutation*, as described below.

The *two-cab permutation* method proceeds in two steps: first swap all intra-cabinet links, and then swap inter-cabinet links that connect the same pair of cabinets. More formally, in the first step, for each cabinet i, determine E_i, the set of all intra-cabinet links that connect two switches in cabinet i. Consider two links picked randomly in E_i, say $v_1 \leftrightarrow v_2$ and $v_3 \leftrightarrow v_4$. If all four switches v_1, v_2, v_3 and v_4 are distinct, and if none of the links $v_1 \leftrightarrow v_4$ and $v_3 \leftrightarrow v_2$ already exists, then replace $v_1 \leftrightarrow v_2$ by $v_1 \leftrightarrow v_4$ and $v_3 \leftrightarrow v_4$ by $v_3 \leftrightarrow v_2$, otherwise do nothing. Remove both links from consideration, and repeat until all links in E_i have been considered. Figure 1 (a) shows an example permutation of intra-cabinet links. In the second step, consider all pairs of cabinets (i, j) with $i \neq j$ and let $E_{i,j}$ be the set of all inter-cabinet links connecting a switch in cabinet i to a switch in cabinet j. Swap the endpoints of the links in $E_{i,j}$ using the same method as in the first step for the links in E_i. Figure 1 (b) shows an example permutation of inter-cabinet links between two cabinets. Because all permutations are for links between the same pair of cabinets, cable length is conserved.

The *three-cab permutation* method also proceeds in two steps. Its first step is identical to the first step of the two-cab method. In the second step, consider all triads of cabinets (i, j, k) with $i \neq j \neq k$. Let $E_{i,j,k} = E_{i,j} \cup E_{j,k} \cup E_{k,i}$, i.e., the set of all inter-cabinet links among the three cabinets i, j and k. Randomly pick two links in $E_{i,j,k}$, say $v_1 \leftrightarrow v_2$ and $v_3 \leftrightarrow v_4$ such that v_1 and v_3 are in the same cabinet. If all switches v_1, v_2, v_3 and v_4 are distinct, and if none of the links $v_1 \leftrightarrow v_4$ and $v_3 \leftrightarrow v_2$ already exists, then replace $v_1 \leftrightarrow v_2$ by $v_1 \leftrightarrow v_4$ and $v_3 \leftrightarrow v_4$ by $v_3 \leftrightarrow v_2$, otherwise do nothing. Remove both links from consideration and repeat until all links in $E_{i,j,k}$ have been considered. Figure 1 (c) shows an example permutation of inter-cabinet links among three cabinets. Note that the three-cab permutation is a superset of the two-cab

permutation, i.e., the four switches v_1, v_2, v_3 and v_4 are distributed over two or three cabinets. In this sense the three-cab permutation method leads to "more random" topologies than the two-cab method, which should be to its advantage since it has been shown in the literature that more randomness in a topology leads to lower path lengths [5]–[7]. The number of intra-cabinet links and the number of inter-cabinet links between any two cabinets are identical to those in the base topology, meaning that total cable length is conserved.

3.2 Indirect Networks

We also consider applying topology permutation to indirect topologies, e.g., fat-tree, Clos and Myrinet-Clos. These topologies have hierarchical structures and their deployment of switches to cabinets may also be hierarchical, i.e., some cabinets contain only upper-tier switches without any compute nodes while others contain leaf switches connected to compute nodes. A two-step permutation of inter-cabinet and intra-cabinet links does not make sense in such a hierarchical structure because there are no direct links between leaf switches. For these hierarchically deployed topologies we define another permutation method, *one-step permutation*.

The *one-step permutation* method swaps two links without using two separate steps for intra- and inter-cabinet links. For each pair of cabinets i and j with $i \neq j$, let $\widehat{E}_{i,j} = E_{i,j} \cup E_i \cup E_j$, i.e., the set of all intra-cabinet and inter-cabinet links with both endpoints in cabinets i and/or j. Randomly pick two links in $\widehat{E}_{i,j}$, say $v_1 \leftrightarrow v_2$ and $v_3 \leftrightarrow v_4$. If all switches v_1, v_2, v_3 and v_4 are distinct, and if none of the links $v_1 \leftrightarrow v_4$ and $v_3 \leftrightarrow v_2$ already exists, and if the two links are not both intra-cabinet links in different cabinets, then replace $v_1 \leftrightarrow v_2$ by $v_1 \leftrightarrow v_4$ and $v_3 \leftrightarrow v_4$ by $v_3 \leftrightarrow v_2$, otherwise do nothing. Remove both links from consideration and repeat until all links in $\widehat{E}_{i,j}$ have been considered. Figure 1 (d) shows an example one-step permutation of links in and between two cabinets. The total numbers of intra- and inter-cabinet cables are unchanged after each permutation operation. The total cable length is thus conserved.

3.3 Topology Generation

For both direct and indirect topologies the topology generation procedures we have described can in principle produce a topology that has a lower bisection bandwidth than the original topology (or even a topology that is partitioned). In our experiments we have never seen a decrease in bisection bandwidth due to topology permutation, which likely indicates that such reduction happens with low probability. As a result, it is possible to keep invoking the generation procedure until a topology is generated that has the desirable bisection bandwidth.

In all the results presented in the rest of the paper we generate 20 random trials for each permuted topology. Among these 20 trials we pick the one with the lowest diameter, breaking ties using the average shortest path length. In all our experiments we observe at most a 1% coefficient of variation in average shortest path length across the 20 trials.

4 Topologies and Physical Layouts

4.1 Direct Topologies

We consider the following direct topologies as starting points for topology permutation:

- 2DTORUS: A 2-dimensional torus of degree 4;
- 5DTORUS: A 5-dimensional torus of degree 10;
- HYPERCUBE: A hypercube of degree n for $N = 2^n$ switches;
- FHYPERCUBE: A folded hypercube of degree $n+1$ for $N = 2^n$ switches, in which a link is added between each switch and its most distant multi-hop neighbor [15].

The two-cab and three-cab permuted versions of topology *topo* are denoted by P2-*topo* and P3-*topo*, respectively. For the 2DTORUS and 5DTORUS topologies we always opt for a shape that has the smallest maximum dimension, so as to make the topology as close as possible to a cube. For instance, we generate a 2^9-switches 5DTORUS with dimensions $4\times4\times4\times4\times2$ (instead of, e.g., $2 \times 4 \times 8 \times 4 \times 2$).

We compare the above permuted topologies to the following competitors:

- RING-n: A DLN [7] of degree n, which consists of a ring plus $n-2$ *random* shortcut links at each switch;
- DRAGONFLY-a-h: A Dragonfly [24] of degree $a-1+h$, which has a fully-connected switches in each group and h inter-group links at each switch;
- HYPERX-s_1-s_2-s_3: A 3-dimensional HyperX [29] of degree $s_1+s_2+s_3-3$, which has s_1 switches in each cabinet, s_2 switches along cabinets row, and s_3 switches along cabinets column.

For all these topologies, we assume that each cabinet stores 16 switches. We assign switches taken in the canonical topological order sequentially to cabinets, i.e., the first 16 switches go into the first cabinet, the next 16 switches go into the second cabinet, and so on. Topology permutation does not affect the assignment of switches to cabinets.

4.2 Indirect Topologies

We consider the following indirect topology as starting points for topology permutation:

- MYRICLOS: The Myrinet Clos [30] built from 16-port switches for N compute nodes.

TABLE 1
Structural parameters of MYRICLOS.

N = #nodes	32	64	128	256	512	1024	2048	4096
#switches	6	12	24	80	160	320	896	1792
4th tier	—	—	—	—	—	—	128	256
3rd tier	—	—	—	16	32	64	256	512
2nd tier	2	4	8	32	64	128	256	512
1st tier	4	8	16	32	64	128	256	512
#enclosures	1	1	1	6	12	24	48	96
#cabinets	2	2	2	4	7	14	28	56
switch-only	1	1	1	2	3	6	12	24
node-only	1	1	1	2	4	8	16	32

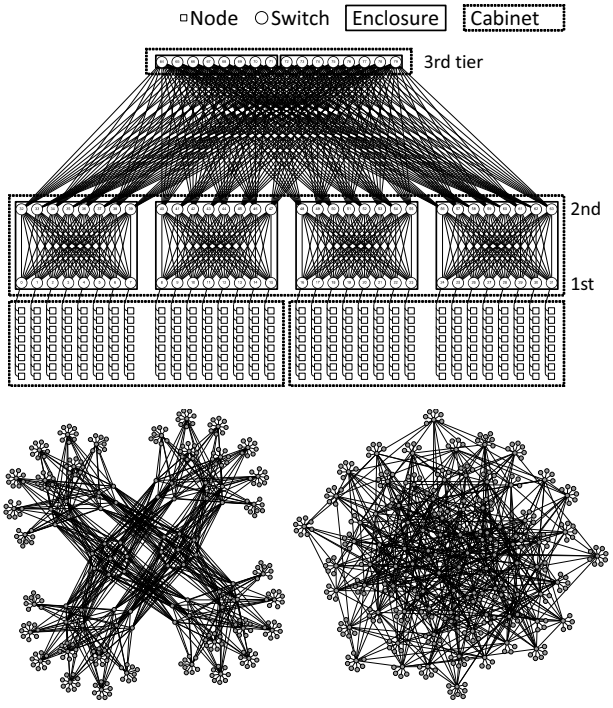

Fig. 2. Example deployment of 80-switch 256-node MYRICLOS (top), its topological view (bottom left) and its permuted version P1-MYRICLOS (bottom right).

The one-step permuted version of MYRICLOS is denoted by P1-MYRICLOS. For the MYRICLOS topology, we follow the deployment guidelines provided in [30], which are summarized in Table 1. We assume that 128 compute nodes or four M3-E128 switch enclosures (i.e., 512 switch ports) can fit in a cabinet. An M3-E128 switch enclosure has 128 switch ports and works either as 16 8-port switches or as 8 16-port switches. We assign all compute nodes to node-only cabinets in the canonical order, and then assign all switch enclosures to switch-only cabinets from lower tier to upper tier. See Figure 2 for an example with 256 compute nodes, i.e., corresponding to the 4th column in Table 1. The figure also shows an instance of the one-step permutation of this topology.

The Myrinet Clos topology may also be built using recent high-radix switches. For a given number of nodes, as the number of switch ports increases, the diameter and the average shortest path length become lower in both the permuted and the original topologies. The advantage of topology permutation is thus expected to be maintained, even when using high-radix switches.

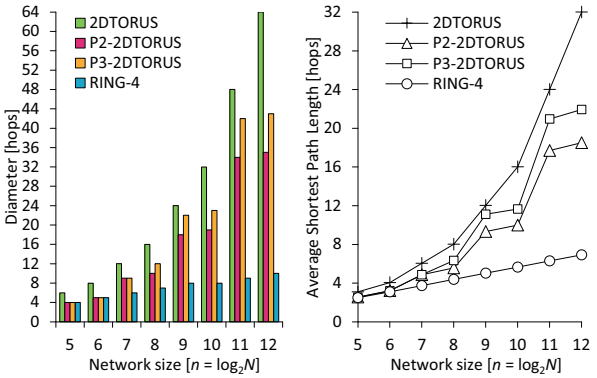

Fig. 3. Diameter (left), average shortest path length (right) vs. network size for 2DTORUS and the same-degree RING-n.

5 Graph Analysis

5.1 Methodology

In this section we use graph analysis to evaluate the merits of topology permutation when compared to non-random topologies and to previously proposed random shortcut topologies. Unless otherwise specified, all comparisons are between topologies with the same degree so as to be fair. All analyses are performed using R with the igraph library to compute topology diameters and average shortest path lengths.

5.2 Main Results

Figure 3 shows the diameter and the average shortest path length for 2DTORUS, its two permuted versions P2-2DTORUS and P3-2DTORUS, and RING-4 vs. the number of switches, $N = 2^n$, up to $N = 2^{12}$. Note that $N = 2^{12}$ already represents a large-scale system. Assuming switches with 36 ports, each switch would support 32 compute nodes for more than 131k compute nodes in total. We see that all random topologies improve both metrics over the non-random 2DTORUS. P2-2DTORUS leads to equivalent or better results than P3-2DTORUS, showing that swapping links between two cabinets is more effective than swapping among three cabinets. This same result is observed for all the topologies discussed hereafter. In terms of diameter, P2-2DTORUS is outperformed by RING-4 but improves significantly over 2DTORUS (e.g., for $n = 12$ its diameter is larger than that of RING-4 by 25 hops but lower than that of 2DTORUS by 29 hops). Similarly, P2-2DTORUS leads to larger average shortest path lengths than RING-4 but still improves over 2DTORUS (e.g., for $n = 12$ its average shortest path length is 11.59 hops larger than that of RING-n but 13.50 hops lower than that of 2DTORUS).

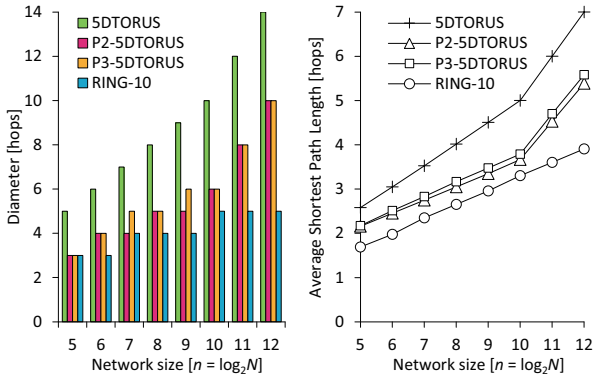

Fig. 4. Diameter (left), average shortest path length (right) vs. network size for 5DTORUS and the same-degree RING-n.

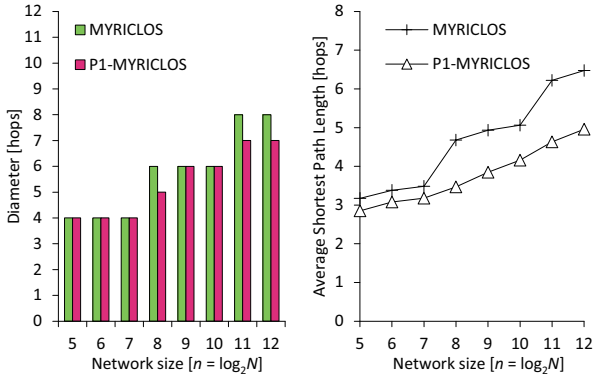

Fig. 5. Diameter (left), average shortest path length (right) vs. network size for MYRICLOS.

The gap between P2-2DTORUS and RING-4 is large because 2DTORUS has very low degree (4) and RING-n is known to achieve low path lengths even at very low degrees [7].

Figure 4 shows results for the 5DTORUS topologies. The trends are similar to those shown for 2DTORUS, but the gap between the topologies are reduced because of the higher degree (10). For $n = 12$, P2-5DTORUS leads to diameter, resp. average shortest path lengths, larger than that of RING-10 by 5 hops, resp. 1.48 hops, but lower than that of 5DTORUS by 4 hops, resp. 1.61 hops. We note an inflection point at $n = 10$. This is because the dimensions of the 2^{10}-switch 5DTORUS are $4 \times 4 \times 4 \times 4 \times 4$. With more switches one of these dimensions becomes 8, which leads to longer path lengths. This is also the reason for the inflection points seen in Figure 3.

Results for HYPERCUBE and FHYPERCUBE are provided and discussed in Section 1 of the supplementary material. They show similar trends and lead to the same overall observations.

Figure 5 shows results for the MYRICLOS and its one-step permuted version P1-MYRICLOS. We see that topology permutation leads to some improvements, at most one hop in diameter and up to 1.51 hops in average shortest path length (for $n = 12$).

We conclude that topology permutation leads to

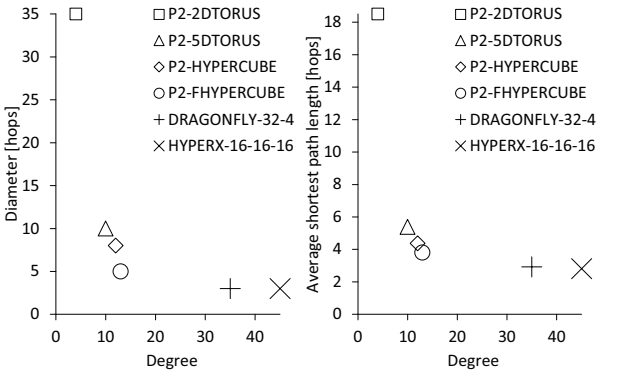

Fig. 6. Diameter (left) and average shortest path length (right) vs. degree for 4,096-switch direct networks including DRAGONFLY and HYPERX.

improved diameter and average shortest path length when compared to original non-random topologies, both for direct and indirect topologies. Surprisingly, the three-cab permutation method, although it leads to "more random" topologies, is slightly outperformed by the two-cab method. The notion that more randomness is better in terms of path lengths [5]–[7] does not necessarily hold true in the context of topology permutation.

5.3 Comparison with High-degree Topologies

The analysis in the previous section is confined to direct topologies with relatively low degree (at most $1 + \log n$), and only presents comparisons between same-degree topologies. Due to the availability of high-radix switches, several high-degree topologies have been proposed. Two well-known such topologies are Dragonfly [24] and HyperX [29] (which subsumes the Flattened Butterfly topology [25]). In this section we consider DRAGONFLY-32-4 (degree 35)[1] and HYPERX-16-16-16 (degree 45). These configuration parameters are chosen to match well with the physical deployment of $N = 2^{12}$ switches stored in 2^8 cabinets laid out in a 16×16 grid on a floor.

Figure 6 shows diameter and average shortest path length vs. degree for the four permuted direct topologies evaluated in the previous section in comparison with Dragonfly and HyperX. The results show that Dragonfly achieves sensibly the same path lengths as HyperX, with a lower degree. As expected, P2-2DTORUS leads to the poorest results. The three other permuted topologies are clustered in the same neighborhood, each corresponding to a different and possibly desirable trade-off between degree and path length. P2-FHYPERCUBE (degree 13) achieves lower path lengths than P2-5DTORUS (degree 10) and P2-HYPERCUBE (degree 12). Dragonfly outperforms P2-FHYPERCUBE by 2 hops for the diameter, and 0.87

1. This setting of Dragonfly is not balanced when assuming 8 compute nodes per switch, but has the same path length as the balanced counterpart DRAGONFLY-32-16 (degree 47).

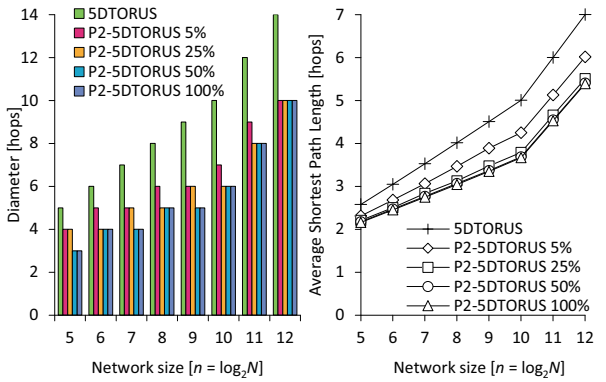

Fig. 7. Diameter (left), average shortest path length (right) vs. network size for 5DTORUS and P2-5DTORUS with 100%, 50%, 25%, and 5% swappable links.

hops for the average shortest path length. These improvements, however, come at the cost of a much larger degree (35 vs. 13). High degrees are feasible given the availability of high-radix switches, but using more ports per switch reduces the number of compute nodes that a system can support. We conclude that our proposed permuted topologies can provide a viable alternative to non-random high-degree topologies.

5.4 Fraction of Swappable Links

For topology swapping to be feasible links must be swappable. In practice groups of links can be implemented as printed circuit cards rather than with actual cables. As an example, the midplane of the Blue Gene/Q supercomputer "hard wires" a 5-D Torus over 512 compute nodes. Link swapping would thus require that multiple such cards be produced, which may incur high cost. Nevertheless, a fraction of the links are still implemented as physical cables, and can thus be swapped. Furthermore, it is expected that more links will become optical and thus be implemented as optical cables. The results in Section 5.2 assume that 100% of the links are swappable. In this section we attempt to answer the question: does topology permutation still work if only a fraction of the links are swappable?

We modify our topology permutation procedure so that a swapping operation is allowed with a probability p. While not necessarily representative of actual physical deployment scenarios, which may impose some structure on the set of links that are swappable, this method allows us to gauge the impact of unswappable links on topology permutation in a broad sense. Figure 7 shows path length results for 5D-TORUS and P2-5DTORUS with $p = 100\%$, $p = 50\%$, $p = 25\%$, and $p = 5\%$. We see that results for $p = 50\%$ and $p = 25\%$ are close to those for $p = 100\%$. Even with p as low as 5% topology permutation achieves significant improvement over the base topology (e.g., several hops of diameter). We conclude that topology permutation can remain effective even in scenarios in which many links are not swappable.

6 Network Performance Simulation

6.1 Methodology

In this section we use a cycle-accurate network simulator written in C++ [7] to evaluate the network latency and throughput of permuted topologies. Every simulated switch is configured to use virtual cut-through switching. A header flit transfer requires over 100ns that include routing, virtual-channel allocation, switch allocation, and flit transfer from an input channel to an output channel through a crossbar. The flit injection delay and link delay together are set to 20ns.

Although both adaptive and deterministic routing schemes can be supported in all the topologies used in our experiments, we use adaptive routing so as to achieve better performance. Routing in hypercubes and tori is done with the protocol proposed by Duato [31], and we use dimension-order routing for the escape paths. For random topologies we use the topology-agnostic adaptive routing scheme with escape paths described in [32]. The path calculation cost has previously been shown to be acceptable [9]. In our simulation, four virtual channels (VCs) are used in all topologies. Simulation results are essentially unchanged when using more VCs, but are markedly poorer when only two VCs are used. We also present results for the Myrinet-Clos topology [30], for which we use up*/down* routing.

We simulate three synthetic traffic patterns that determine each source-and-destination pair: *random uniform*, *bit-reversal*, and *matrix-transpose*. These traffic patterns are commonly used for measuring the performance of large-scale interconnection networks [10]. The compute nodes inject packets into the network independently of each other. In each synthetic traffic the packet size is set to 33 flits (one of which is for the header). Each flit is set to 256 bits, and effective link bandwidth is set to 96 Gbit/sec. We pick relatively small packet sizes since we wish to study the performance of latency-sensitive traffic that consists of small messages [1].

Because discrete event simulation is compute intensive, we simulate networks with at most 512 switches. However, our simulation results are consistent with the graph analysis results presented in the previous section, which are for networks with up to 4,096 switches and show stable trends as the number of switches increases.

6.2 Results

We have conducted simulation experiments for 64-, 256-, and 512-switch direct topologies (2DTORUS, 5DTORUS, HYPERCUBE) with 256, 2,048, and 4,096 compute nodes, respectively, for the three traffic patterns. Each cabinet stores 8 switches for 64-switch

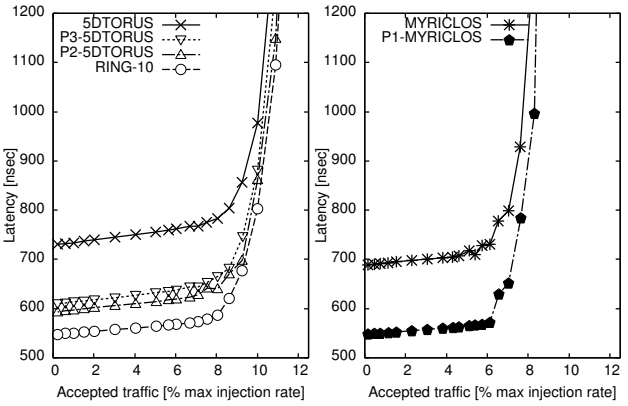

Fig. 8. Latency vs. accepted traffic for the uniform traffic pattern for 512-switch 4,096-node 5DTORUS topologies (left) and 320-switch 1,024-node MYRICLOS topologies (right).

networks or 16 switches for 256- and 512-switch networks. Our results quantify two metrics: *latency* and *throughput*. The latency is the elapsed time (in nanoseconds) between the generation of a packet at a source node and its delivery at a destination node. The throughput is the largest amount of accepted traffic relative to the maximum injection rate (96 Gbit/sec).

Figure 8 shows the results for the 512-switch 4,096-node 5DTORUS topology, for its P2- and P3- permuted versions, and for the same-degree RING-10 topology, for the uniform traffic pattern. Results for all topologies and traffic patterns are presented in a similar manner (see Figures 3–17 in Section 2 of the supplementary material). All results show similar trends. As observed in Section 5, the P3- versions of the topologies are outperformed by their P2- counterparts. Unsurprisingly, RING-n leads to the best or close to the best results for both latency and throughput. In line with the results in Section 5, the permuted topologies are in between the base topology and RING-n. For instance, results for topologies of degree 10 show that RING-10 leads to latency lower than that of 5DTORUS by 25.0%, 25.4%, and 26.1%, for the uniform, matrix-transpose, and bit-reversal traffic, respectively. By comparison, the latency of P2-5DTORUS improves over that of 5DTORUS by 18.7%, 19.2%, and 19.3%. The differences in throughput among these topologies with degree 10 are small. Significant differences in throughput can be seen for the 2DTORUS (degree 4) topologies. For these topologies, network saturation is reached first by the P3-2DTORUS, then P2-2DTORUS and the base topology, and then RING-4, with throughputs ranging roughly between 1 and 6 % maximum injection rate (see Figures 4 and 5 in Section 2 of the supplementary material). A final observation is that the advantage of permuted topologies over the baseline topology increases as network size increases. For instance, considering 5DTORUS and the uniform traffic pattern, the relative latency improvement of P2-5DTORUS over 5DTORUS is 12.4%, 16.7%, and 18.7% for topologies with 64, 256, and 512 switches, respectively. Similar trends are observed for the other two traffic patterns (see Tables 1 in Section 2 of the supplementary material). Overall, our simulation results for direct topologies corroborate the graph analysis results in the previous section (because network latency is correlated with path length).

We have conducted network simulation experiments for the high-degree Dragonfly and HyperX topologies. More specifically, we present results for a 256-switch, 2,048-node configuration (each switch has 8 nodes), for which we pick DRAGONFLY-8-4 (degree 11, imbalanced), DRAGONFLY-16-8 (degree 23, balanced), and HYPERX-16-4-4 (degree 21). Latency and throughput values are shown in Figure 9 for all our direct topologies, including Dragonfly and HyperX. These results are for the uniform traffic pattern (results are consistent across all traffic patterns and are provided in Section 2 of the supplementary material). The horizontal axis in the figure is explained in Section 7. We see that the maximum improvements in latency and throughput of these two topologies are not large. For instance, across all three traffic patterns, when compared to P2-5DTORUS, which has degree 10, Dragonfly improves latency by at most 20.2% and leads to similar throughput (relative differences between -1.6% and 0.4%), while HyperX improves latency by up to 15.2% and also leads to similar throughput (relative differences between -1.5% and 1.3%). As in Section 5.3, we conclude that our proposed permuted topology can provide an attractive alternative to high-degree non-random topologies.

Finally, we show results for the indirect MYRICLOS topology and its permuted version P1-MYRICLOS with 320 switches and 1,024 compute nodes in Figure 8. Results for 80-switch 256-node, and 160-switch 512-node configurations show similar trends (see Section 2 of the supplementary material). The original and permuted topologies achieve almost the same throughput, but the permuted topology improves latency by up to 21.4% for the 320-switch configuration. Since MYRICLOS belongs to the Fat-tree family, we expect similar conclusions to hold for Fat-tree topologies as used in most datacenter networks.

Overall, we conclude that topology permutation is effective in reducing latency and achieving comparable throughput both for direct and indirect topologies.

7 Cable Length

7.1 Methodology

In this section we estimate the cable length required for deploying the topologies onto a physical layout of cabinets. We assume a physical floorplan that is sufficiently large to align all cabinets on a 2-D grid. Formally, assuming c cabinets, the number of cabinet rows is $q = \lceil \sqrt{c} \rceil$ and the number of cabinets per row

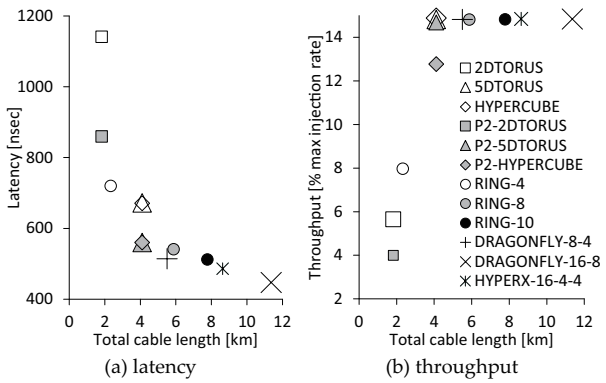

Fig. 9. Latency (a) and throughput (b) vs. total cable length in 256-switch, 2,048-node, direct networks with the uniform traffic pattern.

is $r = \lceil c/q \rceil$. For each topology, and given the mapping of switches to the cabinets, we attempt to place cabinets onto the floorplan in a way that minimizes total cable length. Instead of using regular, but possibly vastly sub-optimal placement, we compute the placement using Simulated Annealing (SA) [33], using the total cable length as the objective function. We use Taillard's implementation of Simulated Annealing [34], which we run for 100 million iterations, picking the best solution out of five independent trials. The distance between any two cabinets is computed using the Manhattan distance. Following the recommendations in [26] we assume that each cabinet is 0.6m wide and 2.1m deep including space for the aisle. We estimate total cable length using the approach in [25]. However, our estimation is more conservative since we consider 2m intra-cabinet cables and a 2m wiring overhead added to the length of inter-cabinet cables at each cabinet so that a switch can be placed in any slot within a cabinet. We ignore cables between compute nodes and switches since their lengths are constant regardless of the layout.

7.2 Results

Figure 9 plots latency and throughput vs. total cable length for 256-switch direct topologies for the uniform traffic pattern (results for all traffic patterns are consistent and provided in Section 3 of the supplementary material). Latency and throughput values are computed from simulation experiments similar to (and including) those presented in the previous section. As explained earlier, latency values are the network delays measured in low load conditions. Throughput values are the largest accepted traffic we observed, regardless of the latency. To avoid clutter, these results exclude P3- topologies since they are almost always inferior to their P2- counterparts. Each figure shows one data point for each topology. In Figure 9(a), better points are located toward the bottom-left corner of the figure (low length, low latency), while in Figure 9(b) better points are located toward the top-left corner of the figure (low length, high throughput).

In all results, as expected, a topology *topo* is either equivalent to or outperformed by the P2-*topo* topology since both topologies have the same cable length. Let us first discuss the latency results, considering all traffic patterns. For the topologies with degree 4, RING-4 leads to latency about between 15.4% and 16.6% lower than P2-2DTORUS but at the expense of up to 27.6% longer total cable length. For topologies with degree 8, we find that RING-8 leads to latency only between 3.4% and 4.6% lower than P2-HYPERCUBE, but incurs an increase in cable length of 43.1%. Finally, for topologies with degree 10, RING-10 leads to latency only about between 8.5% and 9.2% lower than P2-5DTORUS, while incurring an increase in cable length of 89.4%.

Throughput results paint a similar picture. RING-4 improves on P2-2DTORUS by between 85.1% and 135.2%. RING-8 improves on P2-HYPERCUBE by between 0.8% and 16.0%. Similar results are seen when comparing RING-10 to P2-5DTORUS.

When considering the high-degree Dragonfly and HyperX topologies, we find that HyperX leads to very large cable length, which likely does not justify its moderate gains in terms of latency (at most 15.2% improvement over P2-5DTORUS across the three traffic patterns) and throughput (at most 1.3% improvement over P2-5DTORUS across the three traffic patterns). Dragonfly has cable length higher than but comparable to that of our permuted topologies. It leads to improvements in latency (at most 20.2% improvement over P2-5DTORUS), while achieving similar throughput (between -0.4% and 1.6% higher). Given these results, once again we conclude that our proposed permitted topologies can provide an attractive alternative to high-degree non-random topologies.

The overall conclusion is that topology permutation makes it possible to combine low cable length with good performance. RING-n may be preferred in low degree situation because it can lead to good performance even with only a few shortcut links. However, as the degree increases, a permuted topology leads to performance similar to that of RING-n at a much lower cabling expense. Finally, permuted topologies provide a viable alternative to non-random high-degree topologies such as Dragonfly or HyperX.

8 Conclusion

In this work we have proposed and evaluated a method for generating random topologies. Our method consists in randomly swapping link endpoints in a non-random topology. Our results show that, when compared to the base topology, the permuted topology improves path lengths, improves latency, conserves bisection bandwidth, and leads to comparable or even slightly higher throughput. These results are obtained for both direct and indirect non-random base topologies.

One advantage of our method is that a permuted topology has the same cable length as the original non-random topology. Since traditional non-random topologies can often be deployed with low cable length onto a standard cabinet layout, then the cable length of the permuted topology is also low. Another advantage of topology permutation is that it can be applied to a topology that is already deployed. These two advantages are in sharp contrast with previously proposed fully random topologies. A drawback of random topologies in general is the increase in network packaging complexity. With a random topology it may not be possible to use regular building blocks for deploying physical network connections, leading to a significant additional labor and cost overhead at deployment time. Determining whether this overhead is worth the benefit depends on the intended workload and the network technology in use. However, we note that this overhead would be lower for permuted random topologies than for previously proposed fully random topologies.

Overall, while the performance of our permuted topology is not as high as that of the fully random shortcut topology proposed in [7], its cabling length, its cable packaging complexity, and thus its overall network costs, are significantly lower. Our broad finding is that randomizing a topology via link endpoint swapping is an attractive approach for generating low-latency random network topologies.

ACKNOWLEDGMENTS

This work was partially supported by JSPS KAKENHI Grant Numbers 25280018 and 25730068, JST CREST, and NSF Award CNS-0855245.

REFERENCES

[1] 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.

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

[3] P. Coteus and et. al., "Packaging the Blue Gene/L supercomputer," *IBM Journal of Research and Development*, vol. 49, no. 2/3, pp. 213–248, Mar/May 2005.

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

[5] J. Y. Shin, B. Wong, and E. G. Sirer, "Small-World Data Centers," in *Proc. of the Symp. on Cloud Computing*, Oct. 2011.

[6] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: Networking Data Centers Randomly," in *Proc. of USENIX Symposium on Network Design and Implementation (NSDI)*, 2012.

[7] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, and H. Casanova, "A Case for Random Shortcut Topologies for HPC Interconnects," in *Proc. of the Intl. Symp. on Computer Architecture (ISCA)*, 2012, pp. 177–188.

[8] "Earth simulator project," http://www.jamstec.go.jp/es/en/index.html.

[9] M. Koibuchi, I. Fujiwara, H. Matsutani, and H. Casanova, "Layout-conscious random topologies for hpc off-chip interconnects," in *Proceedings of the 19th Intl. Symp. on High-Performance Computer Architecture (HPCA)*, 2013, pp. 484–495.

[10] W. D. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2003.

[11] M. Miller and J. Siran, "Moore graphs and beyond: A survey of the degree/diameter problem," *Electronic Journal of Combinatorics*, vol. DS14, 2005.

[12] M. R. Samatham and D. K. Pradhan, "The De Bruijn Multiprocessor Network: A Versatile Parallel Processing and Sorting Network for VLSI," *IEEE Trans. on Computers*, vol. 38, no. 4, pp. 567–581, 1989.

[13] S. B. Akers, B. Krishnamurthy, and D. Harel, "The Star Graph: An Attractive Alternative to the n-Cube," in *Proc. of the Intl. Conf. on Parallel Processing (ICPP)*, 1987, pp. 393–400.

[14] K. Hwang and J. Ghosh, "Hypernet: A communication-efficient architecture for constructing massively parallel computers," *IEEE Trans. on Computers*, vol. 36, no. 12, pp. 1450–1466, 1987.

[15] A. El-Amawy and S. Latifi, "Properties and Performance of Folded Hypercubes," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 2, no. 1, pp. 31–42, 1991.

[16] K. Efe, "A Variation on the Hypercube with Lower Diameter," *IEEE Trans. on Computers*, vol. 40, no. 11, pp. 1312–1316, 1991.

[17] N.-F. Tzeng and S. Wei, "Enhanced Hypercubes," *IEEE Trans. on Computers*, vol. 40, no. 3, pp. 284–294, 1991.

[18] W. Dally, "Express Cubes: Improving the Performance of k-ary n-cube Interconnection Networks," *IEEE Trans. on Computers*, vol. 40, pp. 1016–1023, 1991.

[19] E. Ganesan and D. K. Pradhan, "The Hyper-deBruijn Networks: Scalable Versatile Architecture," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 4, no. 9, pp. 962–978, 1993.

[20] Q. M. Malluhi and M. A. Bayoumi, "The Hierarchical Hypercube: A New Interconnection Topology for Massively Parallel Systems," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 5, no. 1, pp. 17–30, 1994.

[21] 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.

[22] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[23] J. Kleinberg, "The small-world phenomenon and decentralized search," *SIAM News*, vol. 37, no. 3, pp. 1–2, 2004.

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

[25] J. Kim, W. J. Dally, and D. Abts, "Flattened butterfly: a cost-efficient topology for high-radix networks," in *Proc, of the Intl. Symp. on Computer Architecture (ISCA)*, 2007, pp. 126–137.

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

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

[28] InfiniBand Trade Association, Pluggable Interfaces Passive Copper, Active Copper and Optical Devices (White Paper), http://www.infinibandta.org/, 2007.

[29] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, "HyperX: topology, routing, and packaging of efficient large-scale networks," in *Proc. of the Conf. on High Performance Computing Networking, Storage and Analysis (SC)*, 2009, pp. 1–11.

[30] Myricom, http://www.myricom.com/scs/myrinet/m3switch/guide/myrinet-2000_switch_guide.pdf.

[31] J. Duato, "A Necessary And Sufficient Condition For Deadlock-Free Adaptive Routing In Wormhole Networks," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 6, no. 10, pp. 1055–1067, 1995.

[32] F. Silla and J. Duato, "High-Performance Routing in Networks of Workstations with Irregular Topology," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 11, no. 7, pp. 699–719, 2000.

[33] D. T. Connolly, "An improved annealing scheme for the QAP," *European Journal of Operational Research*, vol. 46, no. 1, pp. 93–100, May 1990.

[34] [Online]. Available: http://mistic.heig-vd.ch/taillard/

Ikki Fujiwara received the BE and ME degrees from Tokyo Institute of Technology, Tokyo, Japan, in 2002 and 2004, respectively, and received the PhD degree from the Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan, in 2012. He is currently a Project Assistant Professor in the Information Systems Architecture Research Division, National Institute of Informatics, Tokyo, Japan. His research interests include the areas of high-performance computing and optimization. He is a member of the IEEE.

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

Hiroki Matsutani received the BA, ME, and PhD degrees from Keio University, Yokohama, Kanagawa, Japan, in 2004, 2006, and 2008, respectively. He is currently an Assistant Professor in the Department of Information and Computer Science, Keio University. From 2009 to 2011, he was a research fellow in the Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan, and awarded a Research Fellowship of the Japan Society for the Promotion of Science (JSPS) for Young Scientists (SPD). His research interests include the areas of computer architecture and interconnection networks. He is a member of the IEEE.

Henri Casanova received the BS degree from the École Nationale Supérieure d'Électronique, d'Électrotechnique, d'Informatique et d'Hydraulique de Toulouse, France, in 1993, the MS degree from the Université Paul Sabatier, Toulouse, France, in 1994, and the PhD degree from the University of Tennessee Knoxville, U.S.A., in 1998. He is currently an Associate Professor in the Information and Computer Science Dept. at the University of Hawai'i at Manoa. His research interests are in the areas of parallel and distributed computing. He is a member of the IEEE.