Layout-conscious Random Topologies for HPC Off-chip Interconnects

Michihiro Koibuchi, Ikki Fujiwara
National Institute of Informatics
2-1-2, Hitotsubashi, Chiyoda-ku,
Tokyo, JAPAN 101-8430
{koibuchi,ikki}@nii.ac.jp

Hiroki Matsutani
Keio University
3-14-1, Hiyoshi, Kohoku-ku,
Yokohama, JAPAN 223-8522
matutani@ny.ics.keio.ac.jp

Henri Casanova
University of Hawai'i at Manoa
1680 East-West Road,
Honolulu, HI 96822
henric@hawaii.edu

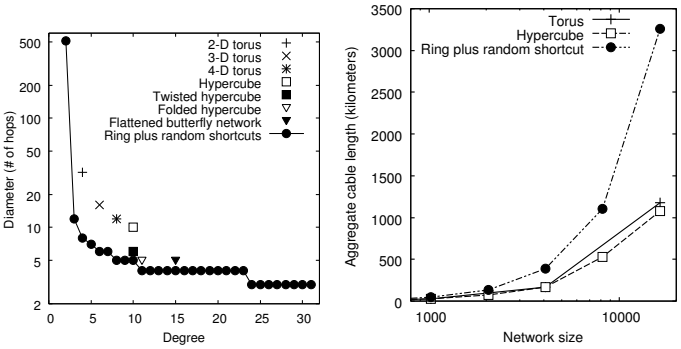
***Abstract*—As the scales of parallel applications and platforms increase the negative impact of communication latencies on performance becomes large. Random network topologies can be used to achieve low hop counts between nodes and thus low latency. However, random topologies lead to increased aggregate cable length and cable packaging complexity on a machine room floor. In this work we propose two new methods for generating random topologies and their physical layout on a floorplan: randomize links after optimizing the physical layout, or optimize the layout after randomizing links. The first method randomly swaps link endpoints for a given non-random topology for which a good physical layout is known. The resulting topology has the same cable length and cable packaging as the original topology, but achieves lower communication latency. The second method creates a random topology with random links picked so that they will not lead to a long physical cable length, and then solves a constrained optimization problem to compute a physical layout that minimizes aggregate cable length. We quantitatively compare these two methods using both graph analysis and cycle-accurate network simulation, including comparisons with previously proposed random topologies and non-random topologies.**

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

I. 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 (SerDes). 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. Fortunately, high-radix switches with dozens of ports are now available, as seen in the YARC routers for folded-Clos or Fat-tree networks [3]. These switches make it possible to design low-latency topologies with higher degree than traditional high-diameter topologies, e.g., the 3-D torus used in the BlueGene/L supercomputer [4].

Traditional topologies use regular structures that can match application communication patterns [4], [5]. One drawback of using a regular structure is that it strictly defines network size (e.g., k^n vertices in a k-ary n-cube topology) even though the scale of an HPC 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 [4], [5]. As the number of inter-cabinet cables increases, the number of backup cables proportionally increases. These backup cables must be installed at deployment time since adding cables once the platform is deployed is costly.



(a) (b)

Figure 1. (a) Diameter vs. degree for a 2^{10} topology, for non-random topologies and the random shortcut topology proposed in [6], and (b) aggregate cable length vs. N (degree is $log_2 N$).

Random shortcut topologies are generated either as fully random graphs [7] or by adding random links to classical topologies [6], [8]. These topologies achieve low diameter, low average shortest path length, and thus low end-to-end network latencies [6]. Figure 1(a), which reproduces results in [6], plots diameter vs. node degree, using a logarithmic scale on the vertical axis, for seven non-random topologies with 2^{10} vertices: $2/3/4$-D torus, hypercube, twisted hypercube [9], folded hypercube [10], and flattened butterfly network [11]; and for the "ring plus random shortcuts" topology proposed in [6]. The striking observation is that, for a given degree, random shortcut topologies have (often dramatically) lower diameter than non-random topologies. Furthermore, a small number of random shortcuts is sufficient to obtain low diameter, as seen in the curve's sharp initial drop. Similar results are obtained when considering average shortest path length. Another advantage of random topologies is that they naturally mitigate the problems of component failures and of human errors that lead to mis-

connected link ports during system deployment.

A practical concern for random shortcut topologies is long cable length for a physical deployment [7], [6]. Aggregate cable length can reach astronomical proportions in deployed systems that use non-random network topologies. For example, the first generation Earth Simulator required over two thousand kilometers of cabling [12], while the K-computer requires one thousand kilometers [5]. The use of random shortcuts further increases cable length, and thus cost. Figure 1(b) shows average cable length versus network size for a hypercube topology, a torus topology and for the random shortcut topology proposed in [6]. For a given network size all the topologies have the same degree, and they use a standard physical layout by which switches are taken in the canonical topological order and mapped to cabinets sequentially. We see that the cable length of the random topology is about three times larger than that of the hypercube and torus, meaning that its cost would be significantly higher in practice. Another concern with random shortcut topologies is cable packaging. As the number of cabinet pairs that are directly connected by at least one cable increases, the cable packaging becomes more complex, also increasing cost. Previously proposed random topologies tend to connect all cabinet pairs. We conclude that, in spite of good performance results, random shortcut may thus become impractical at large scale due to the added cost.

To address the above issue we propose and evaluate two methods for generating random topologies and their physical layouts: one method randomizes links after optimizing the physical layout while the other randomizes links before optimizing the physical layout. The first method, which we term *permutation*, uses a known good physical layout for a non-random topology as a starting point and swaps endpoints between pairs of links in a way that conserves cable length. One advantage of topology permutation is that it can be applied to HPC systems that are already deployed in a machine room using a non-random topology: it only requires swapping the endpoints of some pairs of physical links and updating routing tables. The second method, which we term *constrained shortcutting*, generates a random shortcut topology starting from a ring and adding shortcut links to it, as in [6], but these shortcuts are created to bypass a small number of nodes so as to reduce the potential for long cable lengths. Then, a graph clustering algorithm is used to group switches together in cabinets, and cabinets are mapped to a physical floor by solving a facility location problem. Our main contributions are as follows:

- We find that a permuted topology has better performance properties than its non-random counterpart (lower diameter, lower latency, identical bisection bandwidth, identical or higher throughput). The performance improvement is lower than that achieved by random shortcut topologies [6], but comes at no increase in cable length. This result shows that randomness of endpoints, rather than the shortcutting effect of bypassing switches, is sufficient to improve performance.
- We find that constrained shortcutting in which shortcuts connect only nodes that are at most $N/4$ hops away on a N-switch ring produces topologies that have essentially the same performance properties as the unconstrained random shortcut topologies in [6]. The main advantage of constrained random shortcut topologies is that they can be mapped to cabinets on a standard floorplan in a way that reduces cable length significantly when compared to their unconstrained counterparts.
- Our results provide a quantitative comparison of topology permutation and constrained shortcutting in terms of performance and aggregate cable length. One important result is that as long as the degree is relatively large, e.g., order $\log N$ for an N-switch network, then topology permutation leads to the best trade-off between performance and cabling cost. Note that such "large" degrees are feasible due to the availability of affordable high-radix switches.
- We find that the path computations necessary to fully define our random topologies can be completed in only a few minutes for networks with tens of thousand of switches, meaning that they can be deployed in real-world systems.

The rest of this paper is organized as follows. Related work is discussed in Section II. Sections III and IV detail and evaluate topology permutation and constrained shortcutting, respectively, while Section V provides qualitative and quantitative comparisons of both methods. Section VI discusses the scalability of path computation. Finally, Section VII concludes the paper with a brief summary of our findings, including recommendations regarding which random topologies should be deployed in practice.

II. Related Work

A. Topologies of HPC Systems

A few topologies are traditionally used to interconnect compute nodes in most HPC systems, and these topologies can be used to interconnect high-radix switches [13]. In *direct topologies*, each switch connects directly to a number of compute nodes as well as to other switches. Popular direct topologies include 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 link end points are "shuffled" deterministically at each stage, so that re-arrangeable or non-blocking data transfers are possible.

B. 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 [14]. Several graphs with tractable and hierarchical structure and good diameter properties have been proposed for interconnection networks, including the well-known De Bruijn graphs [15], (n,k)-star graphs [16], etc. Another approach is to augment known topologies. For instance, in the case of the hypercube, many variations have been proposed: folded hypercube [10], twisted hypercube [9], hierarchical hypercube [17], enhanced hypercube [18], Hyper De Bruijn (hypercube plus De Bruijn) [19], hierarchically constructed Hypernets [20], etc. Some of these variations have also been proposed for k-ary n-cubes, such as express cubes [21].

The low diameter properties of random graphs have been identified in the theoretical literature, e.g., for a ring with random chordal shortcuts [22]. The effectiveness of random, or seemingly random, shortcuts to reduce diameter has been exploited for real-world complex networks, e.g., social networks and Internet topology. The *small-world* property of these networks has been studied in the literature. Watts and Strogatz [23] propose a small-world network model based on a probability parameter that smoothly turns a single-dimensional lattice into a random graph, in which a small number of long edges are used to reduce the diameter drastically. Other small-world networks rely on lattice structure plus random links that are generated by accounting for the distance along the lattice structure [24].

Most of the above hierarchical or random topologies are constructed for fixed numbers of nodes and/or strive to achieve a node degree as low as possible. Furthermore, some of these topologies use non-uniform node degrees, which complexifies their use in a real deployment. By contrast, in this work we focus on topologies of high-radix switches in which maintaining as low node degree as possible is not a pressing concern. Furthermore, some of our proposed topologies do not impose any constraints on the number of nodes. Finally, all our proposed topologies have uniform node degree.

Recently, small-world graphs have been proposed for designing data center networks with increased expandability, fault tolerance, and throughput [8], [7]. In [6] such graphs have been proposed to reduce the latency in HPC interconnects, based on the observation that adding random shortcuts to a ring produce topologies with drastically lower diameter and average shortest path length than same-degree non-random topologies used traditionally in HPC systems. These 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 [25], face the challenge of long aggregate cable length.

C. Cabinet Layout of Topologies

Cabinet layout on a floorplan is a concern when designing large systems because it affects costs [11], [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 $0.57\mathrm{m}\times\ 1.44\mathrm{m}$ footprint, with 128 nodes per cabinet, and that the node/m^2 density should be 75 [11]. A common way to view this problem is to come up with specifications for 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. 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 degree (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) aggregate cable length. This is no longer the case for high-degree, and especially random, topologies. System designers are thus faced with the difficult task of mapping switches to a physical layout so as to reduce aggregate cable length. In addition, the cost of the cabling medium increases with the cable length between cabinets (e.g., for InfiniBand the typical maximum length of passive copper is 10m, while embedded optical is 100m [27]). In this work we focus on the problem of reducing aggregate cable length when mapping random topologies onto pre-determined physical layouts.

III. Topology Permutation

A. Two Permutation Methods

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 non-random traditional topology, e.g., a 3-D torus, that maps well to the cabinet layout in terms of aggregate cable length. We use the notation $x \leftrightarrow y$ to denote a link between switch x and switch y. A natural permutation method proceeds in two steps. 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 $a \leftrightarrow b$ and $c \leftrightarrow d$. If all four switches a, b, c, and d are distinct, then replace $a \leftrightarrow b$ by $a \leftrightarrow d$ and $c \leftrightarrow d$ by $c \leftrightarrow b$, otherwise do nothing. Remove both links from consideration, and repeat until all links in E_i have been considered. In the second step, consider all pairs of cabinets (i, j) with $i \neq j$. Considering $E_{i,j}$, the set of all inter-cabinet links connecting a switch in cabinet i to a switch in cabinet j, swap the endpoints of these links using the same method as described for intra-cabinet links.

Because all endpoint permutations are for links between the same pair of cabinets, bisection bandwidth and cable length are conserved. Figure 2 (a) shows an example initial topology and Figure 2 (b) shows a sample permuted topology generated using the above method.

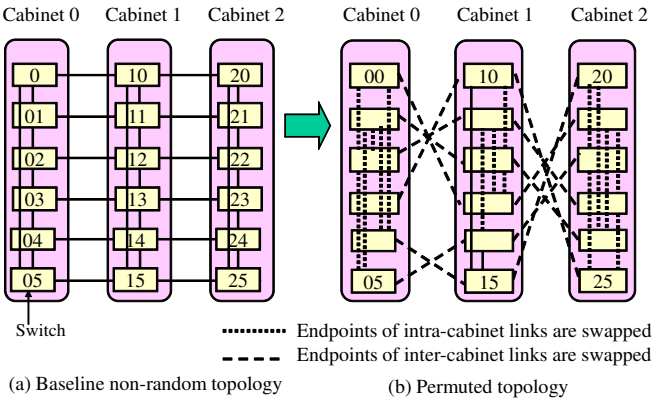


Figure 2. Example 18-switch 3-cabinet baseline non-random topology and permuted topology.

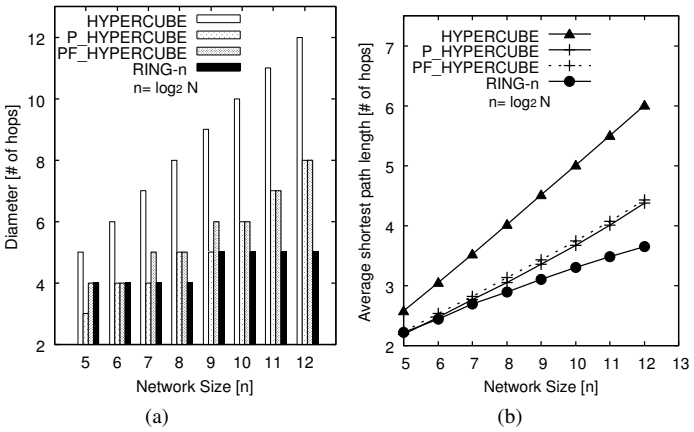


Figure 3. Diameter (a) and average shortest path length (b) vs. network size for the HYPERCUBE, P-HYPERCUBE, and PF-HYPERCUBE topologies and the RING-x topology of the same degree.

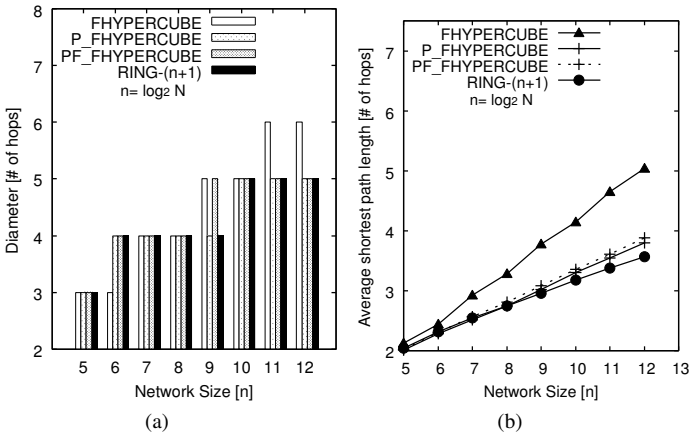


Figure 4. Diameter (a) and average shortest path length (b) vs. network size for the FHYPERCUBE, P-FHYPERCUBE, and PF-FHYPERCUBE topologies and the RING-x topology of the same degree.

An alternate method is to randomly swap endpoints without using two separate steps for intra- and inter-cabinet links. For each pair of cabinets i and j, $i \neq j$, let $\widehat{E}_{i,j} = E_{i,j} \cup E_i \cup E_j$, i.e., the set of all intra-cabinet links and all inter-cabinet links with both endpoints in cabinets i and/or j. Randomly pick two links in $\widehat{E}_{i,j}$, say $a \leftrightarrow b$ and $c \leftrightarrow d$. If all switches a, b, c, and d are distinct and if the two links are not both intra-link cabinets in different cabinets, then replace $a \leftrightarrow b$ by $a \leftrightarrow d$ and $c \leftrightarrow d$ by $c \leftrightarrow b$, otherwise do nothing. Remove both links from consideration and repeat until all links in $\widehat{E}_{i,j}$ have been considered. The number of intra-cabinet links and the number of inter-cabinet links between any two cabinets are identical to those for the baseline topology. Here again, both the aggregate cable length and the bisection bandwidth are conserved. But the topology can be considered as "more random" than when using the two-step method. We term this method *fully random permutation* and the two-step method *partially random permutation*.

Note that this topology generation procedure may produce a partitioned network. However, this happens with very low probability, making it possible to simply regenerate the topology until a non-partitioned network is obtained. Note also that the generation procedure may lead to multiple edges between the same pair of switches, but only the first such edge is actually added and all other redundant edges are ignored if the link duplication is prohibited.

B. Graph Analysis Evaluation

In this section we use graph analysis to evaluate the merits of topology permutation when compared to non-random standard topologies and to fully random shortcut topologies. More specifically, we consider the following random topology:

- RING-n: A ring of degree two with $n-2$ additional random shortcut links at each vertex [6];

and the following three non-random topologies:

- TORUS-n: A $n/2$-dimensional torus of degree n;
- HYPERCUBE: A hypercube of degree n for $N = 2^n$ vertices; and
- FHYPERCUBE: A folded hypercube of degree $n+1$ for $N = 2^n$ vertices, in which an edge is added between each vertex and its most distant multi-hop neighbor [10].

Let 2^p be the number of vertices that fit in a cabinet. The last three topologies above admit a natural mapping of the vertices into cabinets that is known to have low aggregate inter-cabinet cable length: simply assign vertices taken in the canonical topological order to cabinets sequentially. In the case of the TORUS-n topology, however, such straightforward mapping is only valid when the total number of vertices is $N = 2^{p \cdot n/2}$. These three topologies are thus good candidates for evaluating our topology permutation approach. Considering that each cabinet can hold $2^4 = 16$ switches we denote by P-*topo*, resp. PF-*topo*, the partially, resp. fully, random permuted version of base topology *topo*, where *topo* is one of TORUS-n, HYPERCUBE, or FHYPERCUBE.

Figure 3 shows exact values of the diameter and the average shortest path length for HYPERCUBE, its two permuted versions P-HYPERCUBE and PF-HYPERCUBE, and RING-n as the number of vertices, $N = 2^n$, increases. For a given n value all four topologies have the same degree, thus allowing for a fair comparison. We see that all random topologies improve both metrics over the non-random HYPERCUBE. In

Table 1. Diameter and average shortest path length (ASPL) for the TORUS-4 and TORUS-6 topologies, and for the RING-n topology of the same degree.

$N = 2^n$	Topology	diameter	ASPL
256	TORUS-4	16	8.00
	P-TORUS-4	10	5.59
	PF-TORUS-4	11	5.96
	RING-4	7	4.38
4,096	TORUS-6	24	12.00
	P-TORUS-6	16	8.41
	PF-TORUS-6	17	8.70
	RING-6	7	5.06

all cases, P-HYPERCUBE leads to equivalent or better results than PF-HYPERCUBE, showing that randomly swapping links in two separate steps for intra- and inter-cabinet links is more effective than using a single step. In terms of diameter, P-HYPERCUBE is outperformed by RING-n but improves significantly over HYPERCUBE (e.g., for $n = 12$ its diameter is larger than that of RING-n by 3 hops but lower than that of HYPERCUBE by 4 hops). Similarly, P-HYPERCUBE leads to larger average shortest path lengths than RING-n but still improves significantly over HYPERCUBE (e.g., for $n = 12$ its average shortest path length is 0.72 hops larger than that of RING-n but 1.63 hops lower than that of HYPERCUBE). The gaps between P-HYPERCUBE and RING-n increase as n increases. However, $N = 2^{12}$ already represents a very large-scale platform. Assuming switches with 36 ports, each switch would support 24 compute nodes for a total of 98k compute nodes, or almost the number in the largest platform in the Top500 list at the time this article is being written.

Figure 4 shows results for FHYPERCUBE and its permuted versions. Note that the random shortcut topology used is RING-$(n + 1)$ so as to allow same-degree topology comparisons. As expected, diameters and average shortest path lengths are lower than with HYPERCUBE. All random topologies improve both metrics over the non-random HYPERCUBE, with the exception of the diameter when $n = 6$. Again, in all cases, P-FHYPERCUBE leads to equivalent or better results than PF-FHYPERCUBE. In terms of diameter, P-FHYPERCUBE leads to the same diameter as RING-$(n+1)$, improving over FHYPERCUBE by up to two hops. In terms of average shortest path length, P-FHYPERCUBE leads to a large improvement over FHYPERCUBE and is close to RING-$(n + 1)$ (e.g., for $n = 12$ its average shortest path length is 0.23 hops larger than that of RING-n but 1.24 hops lower than that of HYPERCUBE).

Table 1 shows results for the TORUS topologies (with a number of vertices constrained to be an integral power of $p = 16$ so that a straightforward layout of the vertices into cabinets is possible). The main observations are similar to those for HYPERCUBE and FHYPERCUBE. The random topologies outperform the non-random base topology. P-TORUS-n is more effective than PF-TORUS. While P-TORUS-n is not as impressive as RING-n it improves significantly over the base non-random topology. The gap between P-TORUS-n and RING-n is larger than observed for the higher-degree HYPERCUBE and FHYPERCUBE topology. This is because the RING-n topology achieves low diameter and average shortest path length for low n values, as seen in Figure 1(a).

We conclude that topology permutation leads to largely improved diameter and average shortest path length when compared to original non-random topologies. Furthermore, for high-degree topologies such as hypercubes, it leads to result that can be close to that of the random shortcut topology proposed in [6]. In the next section we evaluate topology permutation in terms of actual network latency and throughput measured in simulation.

C. Simulation Evaluation

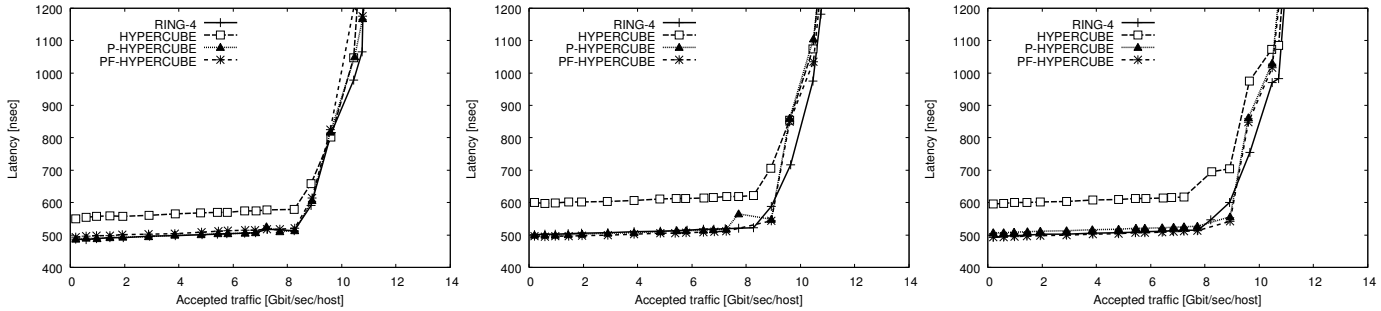
1) Methodology: We use a cycle-accurate network simulator written in C++ [6]. Every simulated switch is configured to use virtual cut-through switching. A header flit transfer requires over 100ns that include the routing, virtual-channel allocation, switch allocation, and flit transfer from an input channel to an output channel through a crossbar. The flit injection delay and link delay together are set to 20ns. Each cabinet stores 16 switches (except in the case of 64-switch networks, in which case there are eight switches per cabinet). Routing in hypercubes and tori is done with the protocol proposed by Duato [28], and we use dimension-order routing for the escape paths. For random topologies we use the topology-agnostic adaptive routing scheme described in [29], with up*/down* routing for the escape paths. In our simulation, four virtual channels are used in all topologies. 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 [31]. The hosts inject packets into the network independently of each other. In each synthetic traffic the packet size is set to 33 flits (one of which is for the header). Each flit is set to 256 bits, and effective link bandwidth is set at 96 Gbps. We pick relatively small packet sizes since we wish to study the performance of latency-sensitive traffic that consists of small messages [1].

Our results quantify two metrics: *latency* and *throughput*. The latency is the elapsed time (in nsec) between the generation of a packet at a source host and its delivery at a destination host. The throughput is the largest amount of traffic (in Gbit/sec) accepted by the network before network is not saturated.

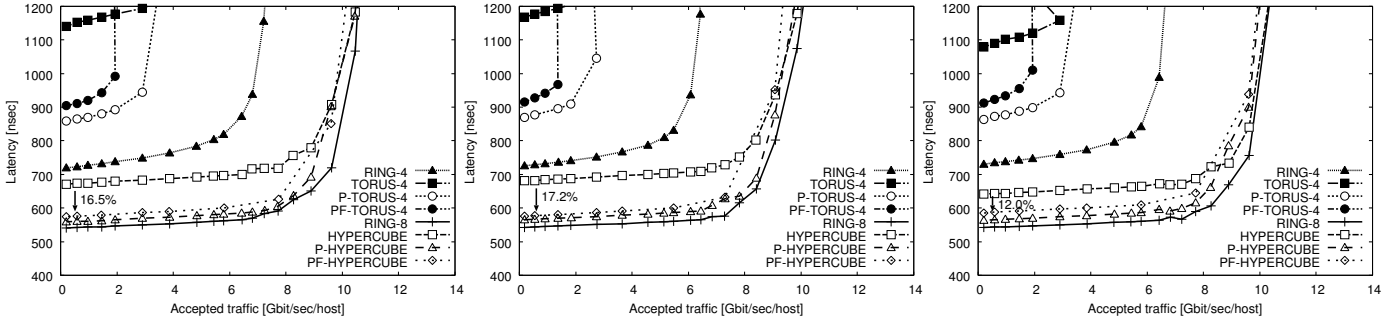
Because discrete event simulation is compute intensive, we simulate networks with at most 512 switches. However, our simulation results are consistent with the graph analysis results presented in the previous section. Those results are for networks with up to 4,096 switches and show stable trends as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

2) Simulation Results: Figures 5, 6, and 7 plot communication latency vs. accepted traffic for 64-, 256-, and 512-switch direct topologies, respectively. Each figure shows results for our three synthetic patterns. All figures show results for



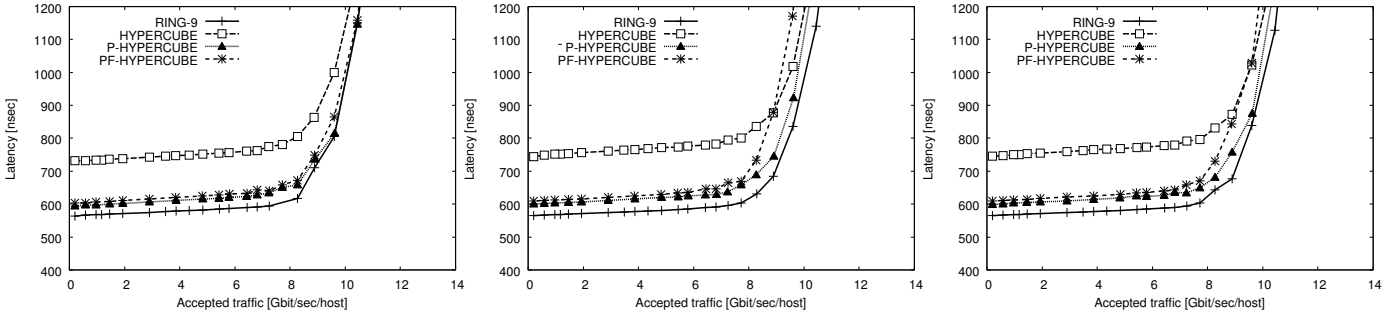
(a) uniform traffic (b) matrix transpose traffic (c) bit reversal traffic

Figure 5. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (64 switches, 256 hosts).



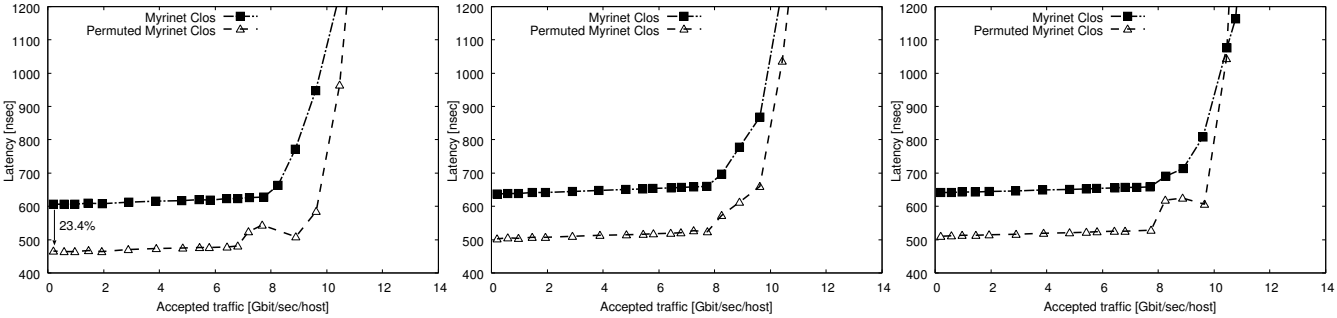
(a) uniform traffic (b) matrix transpose traffic (c) bit reversal traffic

Figure 6. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (256 switches, 2,048 hosts).



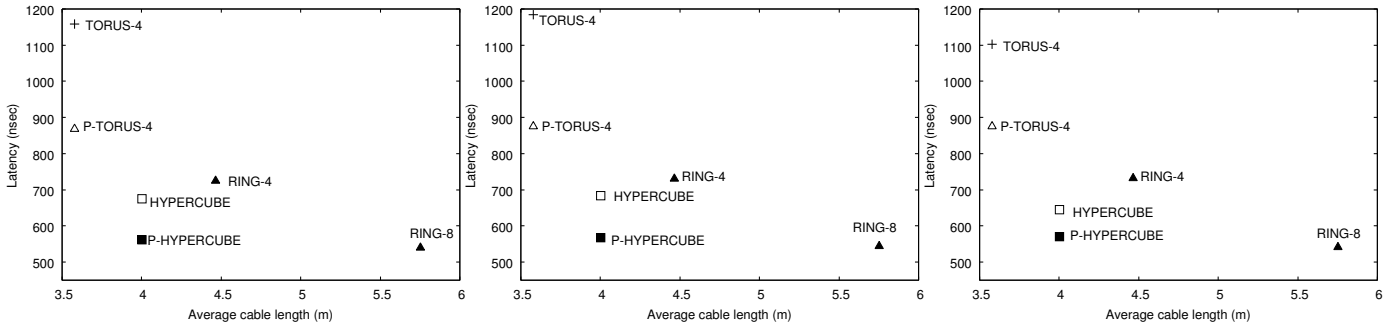
(a) uniform traffic (b) matrix transpose traffic (c) bit reversal traffic

Figure 7. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (512 switches, 4,096 hosts).



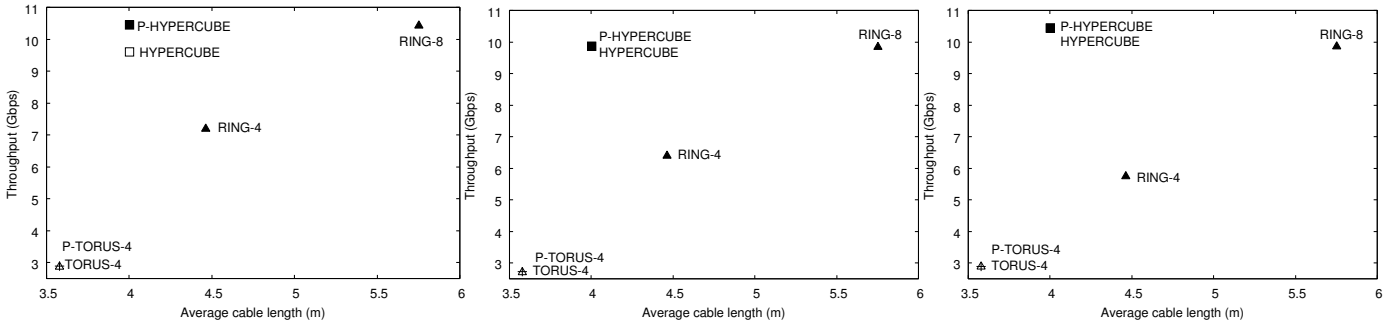
(a) uniform traffic (80 switches, 256 hosts) (b) uniform traffic (160 switches, 512 hosts) (c) matrix transpose traffic, (160 switches, 512 hosts)

Figure 8. Latency vs. accepted traffic for non-random topologies and random shortcut topologies.



(a) uniform traffic (b) matrix transpose traffic (c) bit reversal traffic

Figure 9. Latency vs. average cable length (256 switches, 2,048 hosts).



(a) uniform traffic (b) matrix transpose traffic (c) bit reversal traffic

Figure 10. Throughput vs. average cable length (256 switches, 2,048 hosts).

HYPERCUBE, P-HYPERCUBE, and PF-HYPERCUBE, as well as for the same degree RING-n topology, where 2^n is the number of switches. Results for 256-switch topologies also include TORUS-4, P-TORUS-4, and PF-TORUS-4, as well as the same degree RING-4 topology. We do not include FHYPERCUBE here because results lead to the same conclusion as those for HYPERCUBE (which is expected given the graph analysis results). The network latency of a given topology is the value of the corresponding curve on the left of the horizontal axis. The achieved throughput is quantified by the points at which the latency curve "shoots up."

For 2^n-switch topologies, RING-n leads to the better results in terms of both latency and throughput than all HYPERCUBE versions. For instance, for a 256-switch topology, it achieves latency lower than that of HYPERCUBE by 19.4%, 20.1%, and 15.6% for the uniform, matrix-transpose, and bit-reversal traffic, respectively. By comparison, the latency of P-HYPERCUBE improves over that of HYPERCUBE by 16.5%, 17.2%, and 12.0%. PF-HYPERCUBE leads to performance inferior to that of P-HYPERCUBE. The advantage of random topologies over the baseline HYPERCUBE increases as network size increases. It is worth noting that all same-degree hypercube and RING-n topologies achieve similar throughput. In the case of 256 switches, results show that the non-random TORUS-4 is widely outperformed by same-degree random topologies. P-TORUS-4 is significantly better than PF-TORUS-4 (with a latency 5% lower) but not as good as RING-4 (which has a latency 21% lower than PF-TORUS-4). Furthermore, for these low-degree topologies, we do observe differences in throughput, with network saturation being reach first by TORUS-4, then PF-TORUS-4, then P-TORUS-4, and finally by RING-4. Overall, our simulation results corroborate the graph analysis results in the previous section. This is expected because network latency is correlated with diameter and average shortest path length.

We also present results for the indirect Myrinet-Clos topology in Figure 8, which plots communication latency vs. accepted traffic for 80- and 160-switch topologies with 256- and 512-hosts, respectively, for our synthetic traffic patterns. We assume that leaf switches are stored in cabinets that store their local compute nodes (up to 128), while the other switches are stored in switch-only cabinets. The permuted Clos topology is computed using our fully randomness method. We observe that the original and permuted topologies achieve sensibly the same throughput, but the permuted topology improves communication latency for all traffic patterns, by up to 23.4%. Results for the bit reversal traffic pattern are similar but are omitted due to lack of space. We conclude that topology permutation is effective in reducing latency not only for direct but also for indirect topologies such as Myrinet-Clos. Since this topology belongs to the Fat-tree family, we expect our approach to apply to Fat trees in general, and thus to data center networks. Random topologies have in fact received recent attention for such networks [8], [7].

D. Cable Length Evaluation

In this section we estimate the cable length required for deploying the previous 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

m cabinets, the number of cabinet rows is $q = \lceil \sqrt{m} \rceil$ and the number of cabinets per row is $p = \lceil m/q \rceil$. We assume that each cabinet is 0.6m wide and 2.1m deep including space for the aisle, following the recommendations in [26]. The distance between the cabinets is computed using the Manhattan distance. We estimate average cable length based on [11]: 2m intra-cabinet cables, and a 2m wiring overhead added to the length of inter-cabinet cables at each cabinet. We ignore cables between compute nodes and switches, since their lengths are constant regardless of the layout.

Figures 9 and 10 plot latency and throughput, respectively, vs. average inter-cabinet cable length for 256-switch topologies and all three traffic patterns. The layout of all the topologies, but for RING-n, is based on sequentially mapping switches to cabinets according to the canonical topological order. For RING-n the mapping to the cabinets is computed using a method described in an upcoming section (Section IV-D). 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 before network saturation. Throughput values are computed as the largest accepted traffic at which network delay is less than 1.2μs. To avoid clutter, these results exclude PF-$topo$ topologies since they are always inferior to their P-$topo$ counterparts.

In all results, and as expected, a topology $topo$ is either equivalent to or outperformed by the P-$topo$ topology since both topologies have the same cable length. Let us first consider the latency results in Figure 9. Among the topologies with degree 4, RING-4 leads to latency between 19% and 20% smaller than P-TORUS-4 but at the expense of between 24% and 25% longer average cable length. Considering higher degree topologies, then we find that RING-8 leads to latencies only between 3% and 5% smaller than P-HYPERCUBE, but incurs an increase in cable length between 30% and 31%. Throughput results in Figure 10 paint a similar picture. RING-4 improves on P-TORUS-4 by between 100% and 150%, but RING-8 improves on P-HYPERCUBE by less than 0.04% in the case of the uniform and matrix transpose traffics and even leads to lower throughput than P-HYPERCUBE for the bit reversal traffic. 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 (see Figure 1(a)). However, as the degree increases, a permuted topology leads to similar performance as RING-n at a much lower cabling expense.

IV. Constrained Shortcutting

A. Overview

Constrained shortcutting, which is inspired by the random topology generation approach in [6] but aims at lower cable length, proceeds in three steps. First, starting with a simple ring, random shortcuts are generated that only bypass a small number of switches. Second, switches are aggregated into groups of 2^p switches where 2^p is the number of switches that can fit in a single cabinet. This aggregation is done using a graph clustering algorithm so as to reduce the number of edges between groups. Third, the groups are mapped onto a physical floorplan by solving a facility location problem. Figure 11 shows an example for a topologies with $N = 64$ vertices. The first step creates the topology on the left-hand side of the figure, i.e., a ring with random shortcuts that, in this example are constrained to not bypass more than $0.35 \times N$ vertices (note that there are no cross-cutting shortcuts). The result of the graph clustering algorithm used in the second step is shown in the middle part of the figure, in which six clusters are formed. Each cluster is denoted by a symbol, and each node in that cluster is depicted with the same symbol in the topology shown on the left-hand side. The thickness of an edge between two clusters corresponds to the number of edges between the vertices in those two clusters. The third step maps each cluster to a cabinet on a floorplan, as shown on the right-hand side of the figure.

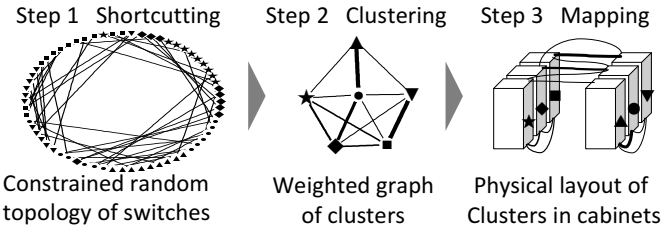


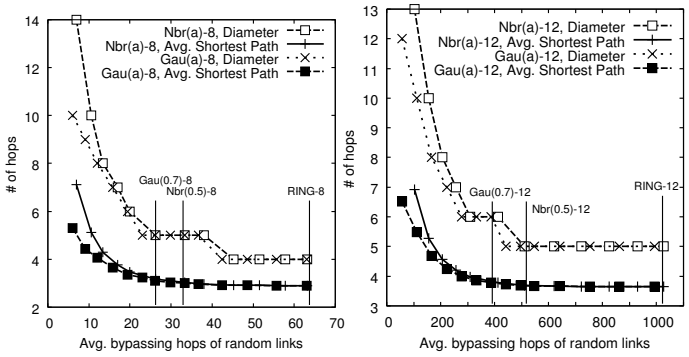
Figure 11. Constrained shortcutting on an example.

B. Shortcut Generation

1) Methods: In the RING-n topology, the two endpoints of each shortcut are randomly selected regardless of their distance (i.e., hop counts), which can lead to a large number of longer cables in a physical layout [6]. Instead, we propose to add a shortcut only between vertices with bounded hop counts so that some of these long cables can be avoided. Results in [6] show that only marginal benefit can be achieved by generating long random shortcuts, i.e., shortcuts between vertices that have high hop counts. Consequently, it is fair to expect that generating shortcuts between vertices with bounded hop counts, or not-as-long random shortcuts, may only degrade latency slightly when compared to RING-n. This reasoning provides the intuition and motivation for the constrained shortcutting approach.

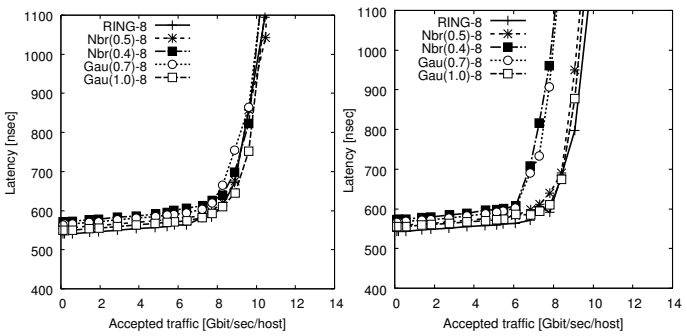
We consider two methods for generating constrained shortcuts on an N-vertex ring, leading to two families of topologies:

- Nbr(p)-n: Consider V, the set of all vertices on the ring. Given a randomly selected shortcut endpoint u in V, the other endpoint v is randomly selected among the vertices in $V_u^p \cap \bar{V}_u$, where V_u^p is the set of vertices that are less than $N \cdot p/2$ hops away from u along the ring, and $\bar{V}_u$ is the set of vertices that are not already connected with u. If such a v is found, then connect u and v and remove them from V; otherwise simply remove u from V. Repeat this



(a) 256 switches (b) 4,096 switches

Figure 12. Diameter and average shortest path length vs. degree for random constrained shortcut topologies.



(a) uniform traffic (b) matrix transpose

Figure 13. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (256 switches, 2,048 hosts).

process until V is empty. Reset V and repeat this process $n-2$ times to obtain a topology of degree n. Nbr(1.0)-n corresponds to RING-n.

- Gau(α)-n: Same as above but v is selected among the vertices in $\bar{V}_u$ so that the distance from u is picked randomly by sampling a Gaussian distribution with mean 0 and standard deviation $N \cdot \alpha/2$, truncated so that it takes values between $-N/2$ and $+N/2$. A positive, resp. negative value, means that the path between u and v is going clockwise, resp. counter-clockwise, along the ring. Lower values of α make average shortcut lengths lower. Unlike with Nbr(p)-n, the probability of having long shortcuts is not zero. As α increases, Gau(α)-n generates topologies closer to RING-n.

2) Graph Analysis Results: Figure 12 shows diameter and average shortest path length for random constrained shortcut topologies with 256 and 4,096 switches versus the average number of hops along the ring between two switches connected by a shortcut. Results are shown for Nbr(p)-n and Gau(α)-n. The last data points on the right of the curves correspond to RING-n. We see that Nbr(≥ 0.5)-n and Gau(≥ 0.7)-n have diameters at most one hop larger than RING-n, and comparable average shortest path length, at least for the 256- and 4,096-switch cases. We conclude that constrained shortcutting can produce high-quality topologies. There is a small advantage to the Gau(α)-n method as it achieves lower or identical diameter and average shortest path length values at lower average shortcut hop counts.

3) Simulation Results: Using the methodology described in Section III-C1 we conduct network simulation experiments to evaluate network latency and throughput of topologies generated using constrained shortcutting. Figure 13 shows results with 256-switch topologies for the uniform and the matrix transpose traffic patterns, for RING-8, Nbr(0.4)-8, Nbr(0.5)-8, Gau(0.7)-8, and Gau(1.0)-8. Results for the bit reversal traffic pattern are omitted because virtually identical to results obtained with the uniform traffic pattern. The results for the uniform traffic pattern show that all topologies lead to sensibly the same results. By contrast, for the matrix transpose traffic, we observe that the throughput decreases significantly for Nbr(0.4)-8 and Gau(0.7)-8 compared to the other topologies. Based on these empirical results, we conclude that Nbr(≥ 0.5) or Gau(≥ 1.0) should be used for constrained shortcutting. In all that follows, we only use Nbr. The advantage of Gau over Nbr seen in the previous section is almost insignificant. One advantage of Nbr over Gau, which is seen in the experiments presented in upcoming sections, is that the number of pairs of cabinets that are directly connected is much larger with Gau than with Nbr. As a result, Nbr leads to less complex cable packaging at only an insignificant performance penalty.

C. Clustering

1) Methods: A topology is an unweighted, undirected simple graph in which vertices represent switches. Grouping switches together in a cabinet is equivalent to contracting the vertices — in other words, converting the graph into a weighted undirected simple graph in which vertices represent cabinets — where loop edges are removed and multiedges are converted to weighted simple edges. One can use clustering methods to group densely-connected vertices together in the same cabinet so that the number of inter-cabinet cables is minimized.

In [32] we have evaluated the use of several clustering methods applied to the RING-n topology. The Walktrap method [33] produces the best results in our experiments. In this work we thus attempt to use this method for clustering constrained random shortcut topologies. For completeness, we present details of the method here. It starts from a state in which each vertex is contained in a one-vertex cluster and recursively merges the two clusters that minimize the increase in the variance of the distance between each vertex and the cluster center. In this work, we define the distance between vertices i and j as $d_{ij} = \sqrt{\sum_{k=1}^{n}(P_{ik}^t - P_{jk}^t)^2/\deg(k)}$, where P_{ik}^t denotes the probability of arriving at vertex k by doing a t-step random walk from i, and $\deg(k)$ denotes the degree of k. We modify the Walktrap method to force it to generate clusters whose size does not exceed a specified cabinet size. We also define a sequential method as a baseline, which groups every 2^p or $2^p - 1$ vertices (so that the grouping is as even as possible) in order of generation, where 2^p denotes the cabinet size. If the random shortcut topology has absolutely

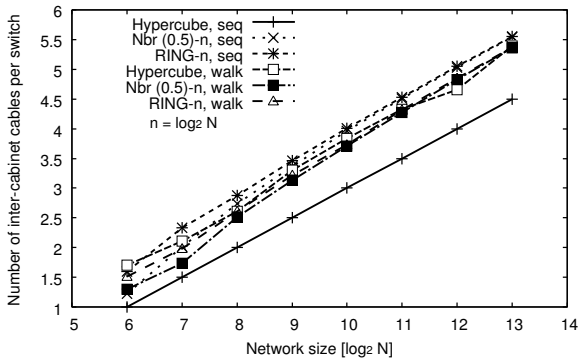


Figure 14. Number of inter-cabinet links vs. network size (degree is $log_2 N$).

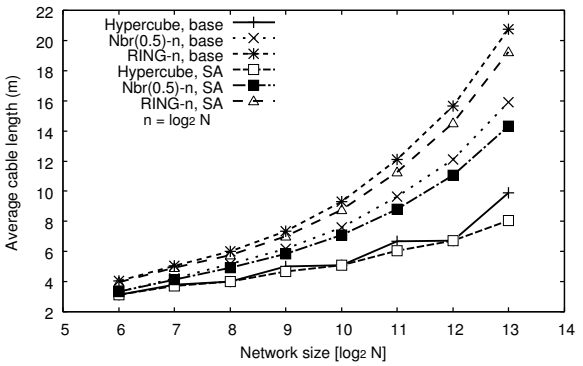


Figure 15. Average cable length vs. network size (degree is $log_2 N$).

no locality, then the sequential method leads to good (random) clustering.

2) Graph Analysis Results: Figure 14 shows the number of inter-cabinet cables produced by the sequential (seq) and Walktrap (walk) clustering methods for several topologies versus the number of switches (for a given number of switches, all topologies have the same degree). We omit the results for Gau(1.0)-n because it leads to results similar to those obtained with Nbr(0.5)-n. The sequential method produces better results than the Walktrap method for the HYPERCUBE topology since for this regular topology assigning nodes to cabinet in the canonical topological order minimizes the number of inter-cabinet links. The Walktrap method outperforms the sequential method for Nbr(0.5)-n and RING-n. The number of inter-cabinet links for RING-n is 59% larger than that of HYPERCUBE, but Nbr(0.5)-n reduces the number of inter-cabinet links by up to 24% when compared to RING-n.

D. Mapping

1) Methods: The input to the mapping method is a floorplan that indicates the possible locations for the cabinets. The method is applicable to an arbitrary floorplan since it only uses the distances between each pair of cabinet locations, for some arbitrary distance definition. Our method then assigns each cabinet to a location so that the inter-cabinet total cable length is minimized. This process can be framed as a facility location problem and formulated as a quadratic assignment problem (QAP). We employ Simulated Annealing (SA) [34], a well-known metaheuristic that has been successfully applied for solving QAPs. In [32], in the context of the RING-n topology, we have experimented with several other heuristics, but they lead to almost the same results as SA. We run SA for 100 million iterations and pick the best solution out of five trials. We also define a baseline method that assigns the locations from left to right in the first row, from right to left in the second row, etc. If the distribution of inter-cabinet cables has no locality, then the baseline method produces a high-quality mapping.

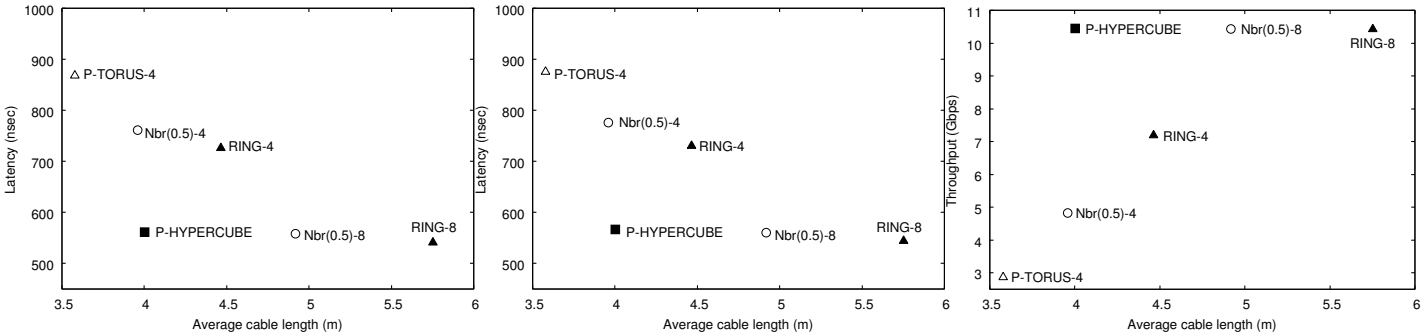
2) Results: Figure 15 shows the average length of inter-switch cables achieved by Simulated Annealing (SA) compared to those by the baseline (base) method for RING-n, Nbr(0.5)-n, and HYPERCUBE, vs. the number of switches (for a given number of switches, all topologies have the same degree). Cable lengths are computed as described in Section III-D.

Results show that Nbr(0.5)-n improves the average cable length by 26% compared to RING-n, but HYPERCUBE leads to markedly lower average cable length. For all topologies, the SA mapping method improves upon the baseline mapping method. The improvement is around 10% for Nbr(0.5)-n and RING-n. When using SA, Nbr(0.5)-n provides a good compromise between RING-n and HYPERCUBE in terms of cable length. For instance, consider 8,192 switches, each with 10 attached compute nodes. This would imply a total of 81,920 compute nodes, which is around the number of compute nodes in the K-computer [5]. At this scale, using the SA mapping method, the average cable length for Nbr(0.5)-n is around 11m, while it is around 7m for HYPERCUBE but more than 14m for RING-n.

V. Comparison of Random Topologies

In this section we compare the best topologies obtained using the permutation method (P-TORUS-x and P-HYPERCUBE), using constrained shortcutting (Nbr(0.5)-n with Simulated Annealing for cabinet mapping), and the random topology proposed in [6] (RING-n). Figure 16 shows latency and throughput vs. cable length for 256-switch topologies for our three synthetic traffic patterns. Results for the bit reversal traffic are similar, and are omitted due to lack of space. For all traffic patterns considering latency or throughput, the results show that the best topology is permuted topology P-HYPERCUBE at degree 8. The other topologies lead to similar (or even worse) performance at higher cabling costs. At degree 4, the choice of the topology would depend on budget constraints for cabling cost and on the target performance: the permuted topology P-TORUS is the most economical, RING-n is the most high-performance, and Nbr(0.5)-n strikes a compromise between the two.

An important concern is the cabling and installation costs for a topology and its physical layout in a machine room. Costs can be estimated using the method and parameters available in recent studies [35], [36]. The cost of 10Gbps cables varies according to the technology (copper or optical, connector types), and is simply assumed to be in the \$50–\$200 range. Installation and re-wiring costs are in the \$10–\$50 range. Using these parameters, expected costs can be computed for



(a) uniform traffic, latency (b) matrix transpose traffic, latency (c) uniform traffic, throughput

Figure 16. Latency and throughput vs. average cable length for 256-switch random topologies.

our topologies. For instance, P-HYPERCUBE decreases cost by up to 27% when compared to RING-12 in a network with 4,096 switches.

Another important concern is the reliability of a topology, i.e., its robustness to link failures, of a topology. The two methods proposed in this work, permutation and constrained shortcutting, and that in [6], all produce topologies of similar reliability. This is because they are based on graphs with the same level of edge redundancy and can all use the same deadlock-free topology-agnostic routing.

VI. Routing Scalability

The scale of a network topology can be limited by routing table size at a switch. We note that 83% of the supercomputers posted on the June 2012 Top500 list [37] are based on Ethernet or InfiniBand. For all these systems, the routing table size limits scalability regardless of the topology, though various types of topologies and deadlock-free routing can be implemented [38].

Another potential scalability issue is path calculation cost for topology-agnostic deadlock-free routing, which is more complex than when routing on structured topologies (see the survey in [39]). The computation cost of path search on most deadlock-free topology-agnostic routings, such as up*/down* routing, or Silla's routing with virtual channels used in the simulation [29], is almost the same as the problem of finding shortest paths in a graph. Several algorithms can be used to compute these paths (e.g. Dijkstra, Bellman-Ford, Floyd-Warshall). In this section we use a priority-queue implementation of Dijkstra's algorithm, with computational complexity $O((N + E)logN)$, where N is the number of vertices and E is the number of edges [40]. Figure 17 shows the path calculation time vs. network size (N) for a RING-log_2N topology when executed as a single-threaded program on a 3.47 GHz Intel Xeon X5690 with 144GB of RAM. Similar results are obtained for HYPERCUBE and Nbr(0.5)-log_2N since these topologies all have the same number of vertices and edges. The results show that it is feasible to compute paths for 16k-switch random topologies to be used with topology-agnostic deadlock-free routings since the computation requires under 82 seconds. Furthermore, path computation can be done in parallel for each destination and a faster but complicated algorithm can be used, such as a Fibonacci-heap implementation of Dijkstra's algorithm.

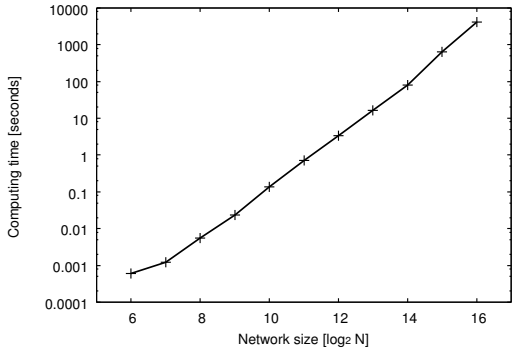


Figure 17. Routing computation time vs. N

VII. Conclusions

In this work we have proposed and evaluated two methods for generating random topologies that lead to lower cable lengths (and less complex cable packaging) than the random shortcut topology proposed in [6] once deployed in a physical cabinet layout. The first method consists in randomly swapping link endpoints in a non-random topology. One advantage of this method is that the generated random topology has the same cable length and packaging 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 an already deployed topology. In our results permuting a topology can improve latency (by up to $\sim$15% for direct topologies and $\sim$25% for indirect topologies) while leading to the same or even slightly higher throughput. While the performance is lower than that of the topology proposed in [6], the cabling cost is much lower. The second method consists in adding shortcuts to a ring topology, but ensuring that these shortcuts do not bypass too many vertices so as to limit cable length. The vertices in the obtained topology are then logically clustered in as many clusters as cabinets in the physical layout so as to minimize the number of inter-cabinet links. Finally, these clusters are physically mapped to cabinets so as to minimize aggregate cable length. One advantage of this approach is that it is applicable to any type of floorplan for any definition of the distance between two

cabinets. In our results we found that topologies generated using this method can achieve high performance but lead to a significant cable length increase compared to traditional non-random topologies. They also lead to more complex cable packaging because any two cabinet have inter-cabinet links between them. However, they achieve essentially the same performance as the topology in [6] at reduced cable length.

Our results show that topology permutation is the best approach for "high" (i.e., logarithmic) degree. It leads to performance at least equivalent to the other methods with cabling costs and cable packaging complexity identical to that of non-random topologies. When the degree is low (e.g., as in a 2-D or 3-D torus), then all three methods can be viable options, with topology permutation being the most economical, the method in [6] the most high-performance, and constrained shortcutting a middle ground. Given that high-radix switches are increasingly available, high-degree topologies are no longer merely attractive due to their good performance, but also feasible in practice. In this context, our results indicate that topology permutation is the method of choice for generating high-performance random topologies.

ACKNOWLEDGMENTS

This work was partially supported by JST CREST and by 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] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The BlackWidow High-Radix Clos Network," in *Proc. of the International Symposium on Computer Architecture (ISCA)*, 2006, pp. 16–28.

[4] 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.

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

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

[7] 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.

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

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

[10] 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.

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

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

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

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

[15] 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.

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

[17] 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.

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

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

[22] 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.

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

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

[25] 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)*, 2008, pp. 77–88.

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

[28] 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.

[29] 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.

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

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

[32] 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*, Dec 2012, pp. 227–232.

[33] P. Pons and M. Latapy, "Computing communities in large networks using random walks," in *Computer and Information Sciences ISCIS*, 2005, pp. 284–293.

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

[35] T. Carpenter, M. Elsheikh, A. Lopez-Ortiz, and S. Keshav, "REWIRE: An optimization-based framework for unstructured data center network design," in *Proc. of the International Conference on Computer Communications (INFOCOM)*, Mar. 2012.

[36] J. Mudigonda, P. Yalagandula, and J. C. Mogul, "Taming the flying cable monster: a topology design and optimization fr amework for data-center networks," in *Proc. of the USENIX conference on USENIX annual technical conference*. USENIX Association, 2011, pp. 8–8.

[37] Top 500 Supercomputer Sites, http://www.top500.org/.

[38] M. Koibuchi, T. Otsuka, T. Kudoh, and H. Amano, "A Switch-Tagged Routing Methodology for PC Clusters with VLAN Ethernet," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 22, no. 2, pp. 217–230, Feb. 2011.

[39] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J. C. Sancho, "A Survey and Evaluation of Topology Agnostic Deterministic Routing Algorithms," *IEEE Trans. on Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 405–425, 2012.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (3rd ed.)*. The MIT Press, Jul. 2009.