A Case for Random Shortcut Topologies for HPC Interconnects

Michihiro Koibuchi
National Institute of Informatics / SOKENDAI
2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN 101-8430
koibuchi@nii.ac.jp

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

D. Frank Hsu
Fordham University
113 West 60th Street, New York, NY 10023
hsu@cis.fordham.edu

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. Fortunately, modern High Performance Computing (HPC) systems can exploit low-latency topologies of high-radix switches. In this context, we propose the use of random shortcut topologies, which are generated by augmenting classical topologies with random links. Using graph analysis we find that these topologies, when compared to non-random topologies of the same degree, lead to drastically reduced diameter and average shortest path length. The best results are obtained when adding random links to a ring topology, meaning that good random shortcut topologies can easily be generated for arbitrary numbers of switches. Using flit-level discrete event simulation we find that random shortcut topologies achieve throughput comparable to and latency lower than that of existing non-random topologies such as hypercubes and tori. Finally, we discuss and quantify practical challenges for random shortcut topologies, including routing scalability and larger physical cable lengths.**

***Index Terms*—Topology, interconnection networks, high performance computing, high-radix switches, diameter.**

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 relatively 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 that use many more links per switch than traditional high-diameter topologies, e.g., the 3-D torus used in the Blue Gene/L supercomputer [4].

Various topologies of high-radix switches have been proposed that have advantages and drawbacks in terms of diameter, average shortest path length, layout (wire length), routing algorithms, and fault tolerance. These topologies use highly regular structures that can match application communication patterns. 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].

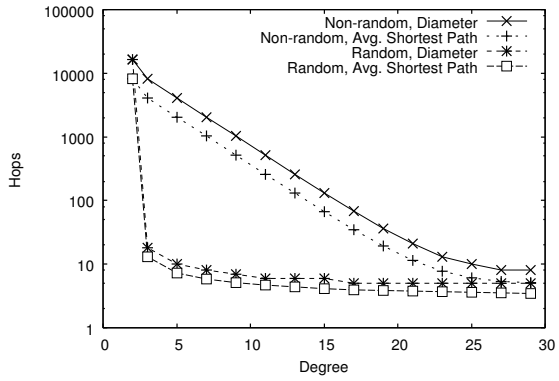

Figure 1. Diameter and average shortest path length vs. degree for a 2^{15}-vertex ring topology with non-random shortcuts and a ring topology with random shortcuts.

In this work, we investigate the use of random shortcuts, i.e., additional random edges, in network topologies for HPC systems. Bollobás et al. [6] note that, for a given degree and an upper bound on the diameter, random graphs are much larger than non-random graphs. Consequently, adding random links to a base graph can significantly reduce its diameter. Let us take the example of a ring with N vertices so that each vertex has degree two. Consider a procedure by which K non-random shortcuts are added to each vertex i so that it is connected to vertices $i + \lfloor N/2^k \rfloor \mod N$, for $k = 1, \ldots, K$. The diameter is thus reduced by approximately a factor two each time a new set of shortcuts is added until no new shortcut can be added. Consider now an alternate procedure by which these shortcuts are added randomly instead, i.e., by picking

each destination vertex uniformly among the possible $2^n - 1$ vertices, while enforcing that all the vertices have the same degree so as to allow a comparison between same-degree topologies. Figure 1 plots diameter and average shortest path length vs. degree, using a logarithmic scale for the vertical axis, for two 2^{15}-vertex topologies generated using the above two procedures. The striking observation is that using random shortcuts affords dramatically lower diameter and average shortest path length compared to using non-random shortcuts. Furthermore, a small number of random shortcuts is sufficient to obtain large improvements. For instance, adding one random shortcut per vertex yields a diameter of 18 compared to a diameter of 8,192 with one non-random shortcut. Achieving a diameter below 20 using only non-random shortcuts requires a degree of 23.

Our goal in this work is to make a case for using random shortcuts when designing network topologies for HPC systems, which we term "Random Shortcut Topologies." Our main contributions are as follows:

- We show that adding random shortcuts to a base topology drastically reduces both diameter and average shortest path length, improving vastly upon non-random popular topologies of the same degree.
- Using flit-level discrete event simulation we show that random shortcut topologies achieve throughput comparable to and latency lower than that of existing non-random topologies such as hypercubes and tori.
- We show that random shortcuts lead to robustness to random edge removals due to a small-world effect. In large HPC systems that use traditional topologies, a large number of redundant cables or resource sparing is typically needed to ensure reasonable fault-tolerance with custom routing algorithms (e.g., in the K-computer [5]). Instead, when using random shortcut topology, we demonstrate that a topology-agnostic deadlock-free routing scheme [7] can be used to degrade the throughput gracefully in the present of faulty links.
- We compare different methods for generating a random shortcut topology, namely the use of different base topologies and of different shortcut generation approaches. This comparison leads us to conclude that a good method for generating a random shortcut topology is to use the simplest base topology, a ring, and to add random shortcuts using a simple uniform distribution.
- We discuss and quantify the limitations of random shortcut topologies in terms of routing scalability and cabling cost.

The rest of this paper is organized as follows. Related work is discussed in Section II. In Section III, using graph analysis, we compare random shortcut topologies to standard non-random topologies and we investigate various methods for random shortcut topology generation. In Section IV, we use discrete-event simulation to compare random shortcut topologies to standard non-random topologies. Section V discusses various practical issues that arise with random shortcut topologies. Finally, Section VI concludes the paper with a brief summary of our findings.

II. Related Work

A. Graphs with Low Diameters

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 [8]. Several graphs with tractable and hierarchical structure and good diameter properties have been proposed for interconnection networks, including the well-known De Bruijn graphs [9], (n,k)-star graphs [10], etc. These graphs are rarely used in interconnection topologies of current HPC production systems. However, their diversity shows that the design space for interconnection topologies is large.

B. 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. 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, with a degree of $2n$, which lead to tori, meshes, and hypercubes. Each topology leads to a particular trade-off between degree and diameter. These topologies are *regular*, meaning that all switches have the same degree as each switch is connected to the same number of switches.

Indirect topologies, i.e., topologies in which some switches are connected only to the neighboring 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 such as the Omega and Butterfly networks. A popular option is (p, q, r) Fat trees, with a degree of $p+q$, where p is the number of upward connections, q is the number of downward connections, and r is the number of tree levels.

More recently, variations of these networks, such as the flattened butterfly [11], have been proposed as a way to improve cost effectiveness. In fact, for large-scale HPC systems, the network layout has a high impact on network cost, especially because longer wires must be optical if high bandwidth is required [12]. The Dragonfly network uses a group of routers as a virtual router to reduce the wire length in the context of high-radix topologies, which is down by distinguishing inter-cabinet and intra-group networks [13]. Various topologies, including our random shortcut topologies, can be used for both the intra-group and inter-cabinet topologies.

Besides popular interconnects based on electric switches, optical switching devices can provide an attractive alternative for future interconnects in HPC systems but would require a topology that has unique properties. Applying optical packet or burst switching to commercial HPC interconnects still faces steep challenges, including buffering. By contrast, optical circuit switching has been widely used for academic Internet

backbones, and it can be applied to HPC interconnects. Since it usually takes milliseconds to make or release an end-to-end circuit path, optoelectric hybrid networks have been discussed in the HPC context [14]. However, topology design has rarely been investigated. To minimize large circuit operation overhead, it is essential to reduce the number of switching elements on a circuit path. Although random shortcut topologies would be amenable to this technology, a quantitative evaluation is beyond the scope of this paper.

C. Distributed Loop Networks (DLN)

A Distributed Loop Network (DLN) is constructed from a simple ring topology to which are added chordal edges, or shortcuts. The objective is for these shortcuts to decrease the diameter while not leading to a large degree increase. An option is to add sets of "evenly spaced" shortcuts, so that the diameter decreases exponentially as the degree increases. It turns out that it is more efficient to add shortcuts in a less regular manner. For instance, [15] shows an example in which adding only five shortcuts for a 36-vertex ring can reduce the diameter from 18 to 9. Determining the minimum-diameter DLN given a bound on the degree is an open problem. In this work we make no theoretical claims regarding diameter properties of DLNs but achieve good practical results based on the lessons learned in [6].

D. Complex Networks

The usefulness of random, or seemingly random, shortcuts has been noted for complex networks, e.g., social networks, and Internet topology. The *small-world* property of these networks has received a fair amount of attention in the literature. Watts and Strogatz [16] proposed a small-world network model based on a probability parameter that smoothly turns a single-dimensional lattice into a random graph, and in which a small number of long edges are used to reduce the diameter drastically. In addition, scale-free and clustering properties in complex small-world networks lead to low diameter and average shortest path length, and also to robustness to random edge removal.

Researchers have designed approaches that exploit the small-world property in the areas of computer data networks [17], peer-to-peer overlay networks [18] [19], or wireless sensor networks [20]. HPC interconnects have a different structure and thus different salient properties such as uniform degree, high radix, and low latency. These properties impact network design when attempting to exploit the small-world property. For instance, a method that generates random shortcuts while accounting for topological distance improves the average shortest path length in large low-radix networks [19]. However, as demonstrated in Section III-F, in the case of (high-radix) HPC interconnects this method achieves only marginal improvement over a method that generates random shortcuts based on a uniform distribution.

Although the small-world property is attractive in interconnects of HPC systems, to the best of our knowledge no works have been published in this context. In this work, by going beyond traditional topologies in current HPC systems, we propose an approach that adds random shortcuts to a base topologies, thereby achieving a small-world effect.

III. Graph Analysis

In this section we use graph analysis to compare random shortcut topologies to non-random topologies in terms of diameter and average shortest path length, scalability, and fault tolerance. We then discuss our shortcut random topology generation method and compare it to alternate methods in terms of random topology sampling, random shortcut generation, and choice of base topology.

A. Methodology

We define a network topology as a graph G with N vertices, where the vertices correspond to identical network switches with the same radix (i.e., number of ports). On each switch some of these ports are used to connect to other switches via network links that correspond to the edges of G. Unless specified otherwise, we also consider *regular* topologies, meaning that all vertices have the same degree, d, which we call the degree of the topology. Assuming a radix of 100, the number of compute nodes supported by the topology is at most $N \times (100 - d)$.

We use three metrics for evaluating and comparing topologies (unless specified otherwise, we only compare same-degree topologies). Considering the length, in number of hops, of the shortest path between every vertex pair, the *diameter* is defined as the maximum such length while the *average shortest path length* is defined as the average. Having a small diameter and a small average shortest path length is desirable as these metrics are strongly correlated with communication latencies experienced between compute nodes, especially when the network is not saturated. We also define a topology's *fault tolerance* as the average percentage of edges that must be randomly removed, in 1% increments, from the topology so that the diameter is increased by at least 2, which we arbitrarily take as our threshold for unacceptable increase in communication latency, or so that the graph is no longer connected. This average is computed using a number of samples sufficient to obtain a 95% confidence interval of length 2.

We consider the following "base" topologies:

- DLN-x: A Distributed Loop Network of degree x. Vertices are arranged in a ring, and a shortcut is added between vertices i and j such that
$$j = i + \lfloor N/2^k \rfloor \ mod\ N \ for\ k = 1,\ \ldots,\ x - 2\ .$$
- MESH-x: An mesh of maximum degree x in which vertices are arranged in an $x/2$-dimensional space so that each vertex is connected to all its one-hop neighbors.
- TORUS-x: An mesh with additional wrap-around edges so that each vertex is connected to exactly x one-hop neighbors.
- HYPERCUBE: A hypercube of degree x for 2^x vertices.
- F-HYPERCUBE: A folded hypercube of degree $x+1$ for 2^x vertices, as defined in [21].

- T-HYPERCUBE: A multiply-twisted hypercube of degree x for 2^x vertices, as defined in [22].
- FLATBUTTERFLY: A k-ary n-fly flattened butterfly topology of degree $(k-1) \times (n-1)$ as defined in [11]. Given a number of vertices, several (k, n) couples are possible that achieve different trade-offs between degree and diameter. Picking the largest possible n produces a hypercube. Picking a lower n and a higher k can lead to a smaller diameter at the expense of larger degree. For large values of k, however, one may obtain a topology of impractical degree. Given 2^x vertices, we pick (k, n) so that the degree is strictly larger than x (the degree of an hypercube), is as low as possible, and is no larger than $2x$. If no such (k, n) couple can be found, we simply do not report any result for this topology.

In this section we do not present results for mesh topologies as they are strictly inferior to tori for the metrics we consider. We use a TOPO-y naming scheme where TOPO is one of the above base topologies and y is the additional number of random shortcuts at each vertex. For instance, TORUS-6-3 corresponds to a 3-D torus topology in which each vertex has degree 9 (6 edges to the 6 one-hop neighbor vertices, and 3 edges to 3 randomly selected vertices). To add random shortcuts to a N-vertex graph with degree x so as to obtain a topology with degree $x + y$, we simply add y perfect matchings to the graph. This is done by scanning a list of vertices sequentially. For a vertex with degree d, we add $\max(0, d-x-y)$ *new edges* between this vertex and $d-x-y$ randomly selected vertices that have degrees strictly lower than $x + y$. If there is already an edge between the two vertices, another random vertex is attempted. We attempt to find each such random vertex at most 10,000 times. If no feasible vertex is found after 10,000 trials, we scan the list of vertices sequentially starting from the last attempted random vertex. It is possible for this procedure to fail (e.g., if the last two vertices in the list have degree $x + y - 2$), in which case it is attempted again. For a given base topology and a given value for y, we generate 100 sample random topologies using this method and pick the one with the smallest diameter. Among those that achieve the same diameter, the one with the smallest average shortest path length is selected.

We implemented our graph generation analysis program using the C++ Snap graph library [23], which provides all necessary abstractions and algorithms. We use this library to compute exact diameters and average shortest path lengths of each generated graph. This computation is costly, thus limiting the size of our graphs to 2^{13} vertices. Nevertheless, assuming 100-radix switches and a (logarithmic) degree of 13, the largest topologies considered in this work can support over 700,000 compute nodes.

B. Number of Shortcut Links

In this section we generate random shortcut topologies using the base topologies with the lowest degree, DLN-2. Figure 2 plots diameter vs. degree for a topology with

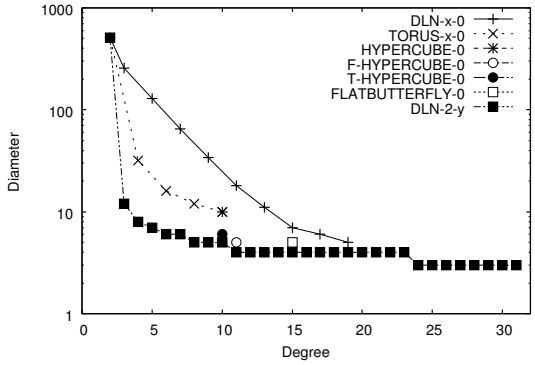

Figure 2. Diameter vs. degree for non-random topologies and for random shortcut DLN topologies ($N = 2^{10}$ vertices).

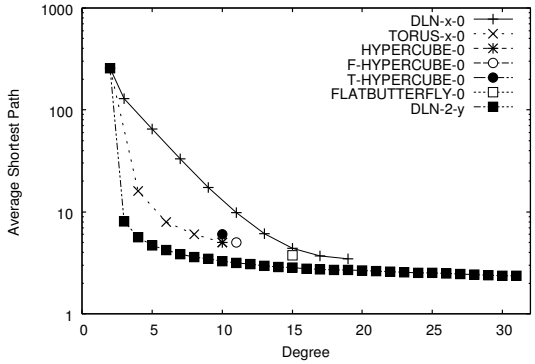

Figure 3. Average shortest path length vs. degree for non-random topologies and for random shortcut DLN topologies ($N = 2^{10}$ vertices).

$N = 2^{10}$ switches for our six classes of non-random topologies: DLN-x-0 ($x = 2, 3, \ldots, 31$), TORUS-x-0 ($x = 4, 6, 8$), HYPERCUBE-0, F-HYPERCUBE-0, T-HYPERCUBE-0, and FLATBUTTERFLY-0; and for the DLN-2-y random shortcut topology ($y = 1, 3, \ldots, 29$). The curve for DLN-x-0 stops at $x = 19$ as there is no further improvement for larger degrees. The vertical axis uses a logarithmic scale. The main observation is that the random shortcut DLN widely outperforms non-random topologies. It achieves equivalent or better diameter than its competitors at lower degrees. For instance, it achieves a diameter lower than 10 at degree 4, while no other topology reaches this diameter until the degree is at least 10. Expectedly, FLATBUTTERFLY-0 leads to lower diameter than TORUS-x-0 and HYPERCUBE-0 at the expense of significantly larger degree, e.g., a diameter of 5 at degree 15. By comparison, the same diameter is achieved by DLN-2-6 at degree only 8. F-HYPERCUBE-0 and T-HYPERCUBE-0, which are variations on a standard hypercube, lead to the best results among the non-random topologies, both achieving a diameter only one hop larger than that achieved by DLN-2-y with same degree. Figure 3 is similar to Figure 2 but plots the average shortest path length (hence the smoother curves). DLN-2-y outperforms its competitors for this metric as well, and in fact wins by a larger margin than in Figure 2. Interestingly, F-HYPERCUBE-0 and T-HYPERCUBE-0 do not fare better than the standard hypercube in terms of average shortest path

length.

C. Scalability

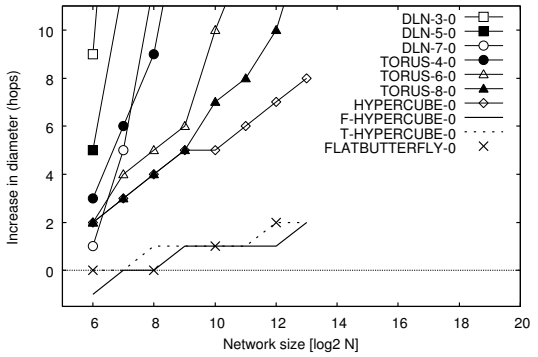

Figure 4. Diameter increase over DLN-2-y when using non-random topologies, where y is chosen so that comparisons are between topologies of the same degree, vs. N.

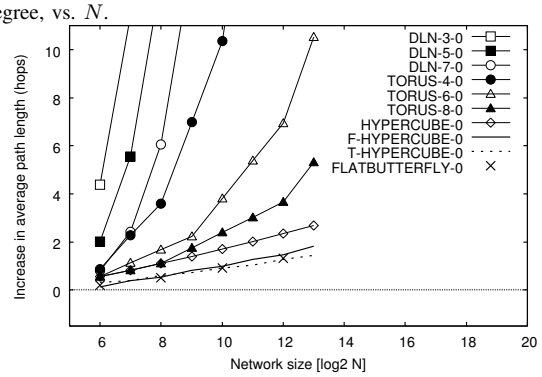

Figure 5. Average shortest path length increase over DLN-2-y when using non-random topologies, where y is chosen so that comparisons are between topologies of the same degree, vs. N.

The overriding concern for a topology is its scalability, i.e., how the diameter and average shortest path length grow as the topology becomes larger. To see how the advantage of using random shortcuts evolves as N increases, we compare the diameter of the DLN-2-y to that of non-random topologies. To allow for fair comparisons, we only compare DLN-2-y, which has degree $y + 2$, to topologies that have also degree $y+2$. Figure 4 plots the difference in diameter when compared to DLN-2-y as N goes from 2^6 to 2^{13}. A positive value means that DLN-2-y has the advantage. For all non-random topologies these differences increase with N, meaning that as topologies become larger it is increasingly beneficial to use random shortcuts. The increase is steep for DLN-x-0, TORUS-x-0, and HYPERCUBE-0, and moderate for F-HYPERCUBE-0, T-HYPERCUBE-0, and FLATBUTTERFLY-0. In fact, for a small topology with $N = 2^6$ F-HYPERCUBE-0 has a diameter of 3 (for a degree of 7) while DLN-2-5 has a diameter of 4. Once $N \geq 2^9$, none of these three topologies achieve a diameter as low as DLN-2-y. At $N = 2^{13}$ they have diameters larger than that of DLN-2-y by 2 hops. Figure 5 is similar but shows relative difference in average shortest path length. We observe that all curves are increasing and that no value is negative, meaning that DLN-2-y always has the advantage and that this advantage increases as topologies become larger. Note that in some cases the ranking between the non-random topologies is different than that seen in Figure 4. For instance, F-HYPERCUBE-0 leads to a lower diameter increase over DLN-2-y than T-HYPERCUBE-0 but to a higher average path length increase.

Lowering the diameter of a topology is a worthwhile objective because the diameter is strongly correlated to the worst-case source-destination latency. However, the diameter has another important implication: a lower diameter leads to a lower end-to-end path length diversity since the number of possible path lengths becomes more constrained. As a result, latencies between all possible source-destination pairs are more uniform. The implication is that topology-aware allocation of tasks to compute nodes, a challenging undertaking when developing a parallel application, becomes less critical as the diameter decreases. Put another way, a topology with more uniform latencies across all possible node pairs makes it more difficult to produce poor task allocations. While much parallel application development and tuning effort is spent for determining good task allocations on standard non-random topologies, random shortcut topologies provide us with an attractive alternative. Although they make it likely impossible to determine an optimal task allocation for a given application due to their randomness, they should lead to reasonable application performance regardless of task allocations.

D. Fault Tolerance

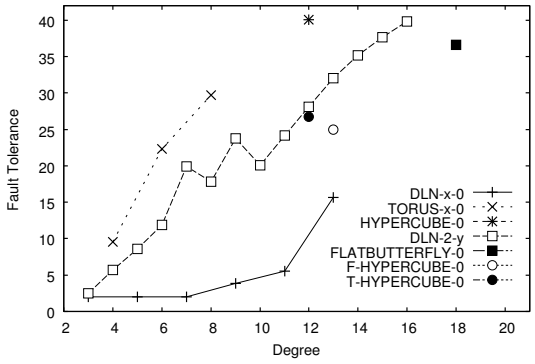

Figure 6. Fault tolerance vs. degree ($N = 2^{12}$ vertices).

An important property of a topology is its resilience to link failures. In Section III-A we have defined a fault-tolerance metric, and in this section we evaluate this metric for several topologies while varying the degree.

Figure 6 shows fault-tolerance results for topologies with $N = 2^{12}$ vertices (higher values mean better fault-tolerance). Unsurprisingly, the non-random DLN topology achieves the worst fault-tolerance results. The torus topology, culminating in the hypercube, leads to the best results. For instance, for the hypercube with a degree of 12 fault-tolerance is just above 40%, meaning that 40% of the edges can be randomly removed before the diameter reaches a value of 14. The random DLN topology falls in between with, for instance, a

fault-tolerance rating around 29% at degree 12. To achieve the fault-tolerance value of hypercube, the random DLN requires degree 16. Note that removed edges may be either edges in the original DLN-2 ring or shortcuts. F-HYPERCUBE-0 and T-HYPERCUBE-0 lead to fault tolerance lower than HYPERCUBE-0, comparable to that of DLN-2-y. Recall that our fault tolerance metric is about diameter degradation due to edge removals. F-HYPERCUBE-0 and T-HYPERCUBE-0 have such lower diameter than HYPERCUBE-0 that their diameter is more sensitive to edge removal. Also, it may seem surprising that FLATBUTTERFLY-0, which has a higher degree and thus more edges, leads to a fault-tolerance value below that of HYPERCUBE-0. But recall that our metric is computed based on the number of edge removals relative to the total number of edges, rather than on an absolute number of edge removals. We observed similar results for all values of N between 2^6 and 2^{12}. While not as impressive as the torus and hypercube topology, the random DLN topologies achieve good fault-tolerance results. Experiencing a diameter increase of only two when around 30% of the edges are randomly removed is likely more than sufficient in practice. Furthermore, although random edge removal is interesting from a graph analysis point of view, in practice fault-tolerance depends on the routing scheme. In the case of standard non-random topologies, custom deadlock-free routing algorithms are used. These algorithms are typically not robust in the face of failed links. Instead, robustness must rely on topology-agnostic routing schemes that can route "around" failed links.

E. Random Topology Sampling

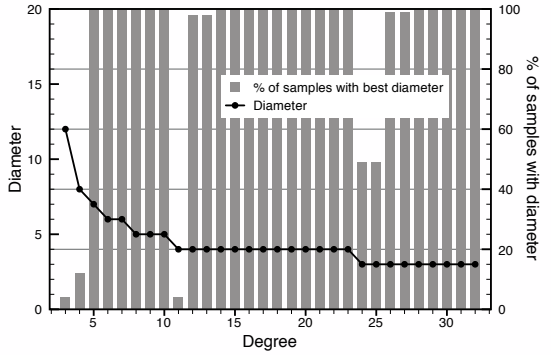

Figure 7. Diameter and percentage of samples that achieve the best diameter across the 100 samples vs. degree for DLN-2-y with $N = 2^{10}$ vertices.

Our method for generating random shortcut topologies involves selecting one topology among 100 random samples (the one with the lowest diameter). In this section we first investigate the variability among these samples and whether 100 samples are sufficient. While answering such questions theoretically is outside the scope of this work, we can provide empirical answers. In all sets of 100 samples, we find that the diameters of the samples differ at most by 1. Figure 7 shows results for DLN-2-y with $y = 1, \ldots, 30$ in the case $N = 2^{10}$. The curve plots the best diameter achieved over the 100 samples vs. the degree, while the bar graph plots the percentage of samples that achieve this best diameter value. We see that in many cases more than 95% of the samples have the same (best) diameter. Drops in this percentage occur when there is a diameter decrease as the degree increases. For instance, while 100% of the samples achieve a diameter of 5 at degree 10, only 4% of them achieve a diameter of 4 at degree 11. Once the degree reaches 12, then most of the samples have a diameter of 5. One may thus wonder whether using more samples for degree 10 would yield a "lucky" topology with diameter 4. We ran simulation experiments using 10,000 samples instead of 100, for $N = 2^6, \ldots, 2^{13}$, and did not observe a single improvement in diameter. In other words, in all our experiments, a "good" topology is always found in the first 100 samples. This conclusion holds as well when considering the variability of the average shortest path length. In all our experiments the coefficient of variation of this metric computed over the samples decreases as N increases and as the degree increases, starting with a value at about 2.3% for $N = 2^5$ and degree 3 (i.e., DLN-2-1). For instance, over the 100 samples for DLN-2-9 for $N = 2^{10}$, the coefficient of variation of the average shortest path length is below 0.25%. The largest value of average shortest path length across these samples is less than 1% larger than the smallest value. Our overall conclusion is that variability across samples is low, and that a moderate number of samples is sufficient. Furthermore, if the number of samples is too small, then the diameter is typically only increased by one and the average shortest path is impacted only marginally.

F. Random Shortcut Generation Method

Our approach for building random shortcut topologies does not take into account the quality/usefulness of the random shortcuts. All shortcuts may not be equal in their abilities to reduce diameter and average shortest path length. In this section we experiment with a more sophisticated, but also more costly, shortcut generation method. When adding y shortcuts to a vertex, $k \times y$ random feasible destination vertices are found with $k \geq 1$. The length (in number of hops) of the shortest path to each of these vertices is computed. The y vertices with the longest shortest paths are selected as the destinations of shortcuts. The goal is to pick shortcuts judiciously, i.e., to avoid adding shortcuts between vertices that already have a short shortest path between them. We repeated all our experiments with the DLN-2-y topology for $N = 2^6, \ldots, 2^{11}$, and $y = 1, \ldots, 28$, for a total of $25 \times 6 = 150$ experiments. We used $k = 1$ (i.e., our original approach), $k = 2$ and $k = 5$. Over all 150 experiments, using $k = 2$ improves the diameter in 11 of the 150 cases when compared to using $k = 1$. Using $k = 5$ leads to an improved diameter over $k = 2$ in 6 cases. In all these cases, the diameter is decreased by 1 and increasing y by 1 leads to identical diameters regardless of the value of k. In terms

of average shortest path, relative percentage improvements are at most 0.02%. We conclude that selecting shortcuts based on shortest path lengths yields only minor improvements. This improvement comes at the expense of longer computation time since the number of shortest path computations is proportional to $k-1$. This additional cost is tolerable if the topology needs to be computed once and for all. In this work, because we generate many topologies for the purpose of analysis, we use $k=1$ even though marginally better results could be achieved with $k>1$. We experimented with other, less costly, methods for picking "good" shortcut destination vertices (e.g., ensuring that they are evenly distributed over the ring, ensure that half of them are more than $N/4$ ring-hops away from the source) but did not observe any improvement over our base approach.

We have only considered random shortcut topologies that are regular, meaning that all vertices have the same degree. We now investigate the impact of this constraint by generating y shortcuts from a source to a random destination regardless of the destination's degree. Furthermore, if an edge between a source and a destination already exists, then we simply ignore it. The resulting topology has vertices of varying degrees, and is thus *irregular*. Figures 8 and 9 show results for diameter and average shortest path length, respectively, where topologies generated using this new method are labeled "irr". We see that in our results irregular topologies never lead to improvements over their regular counterparts. Minimum and maximum vertex degrees for these irregular topologies are shown for selected network sizes in Table 1 when 2 or 6 random shortcuts are added to each vertex. Results in this table show the large degree diversity in the topology, which turns out to be detrimental to diameter and average shortest path length. We conclude that our approach, which consists in iteratively adding random matchings to a base topology, is preferable.

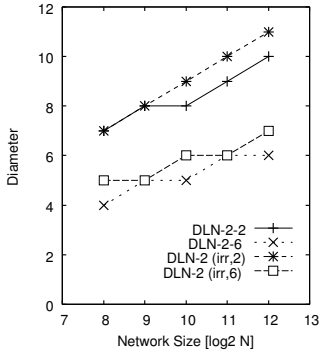

Figure 8. Diameter vs. N for regular and irregular random shortcut topologies.

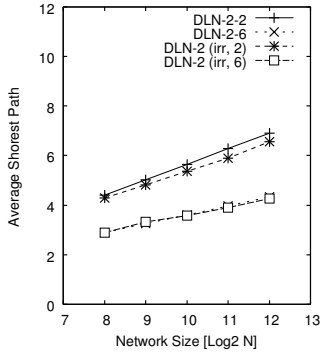

Figure 9. Average shortest path length vs. N for regular and irregular random shortcut topologies.

Table 1. Minimum and maximum vertex degrees in irregular topologies for selected N values.

Network size	$N=2^8$		$N=2^{10}$		$N=2^{12}$	
Avg. degree	min	max	min	max	min	max
4	2	8	2	9	2	11
8	2	12	3	16	2	21

G. Choice of the Base Topology

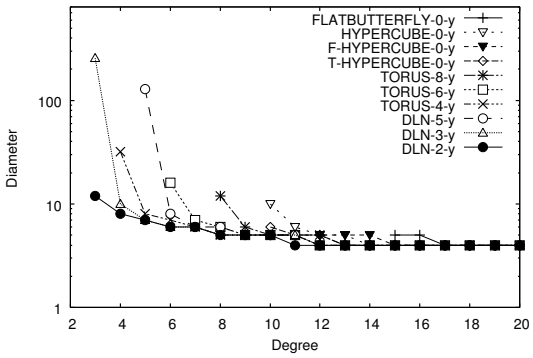

Figure 10. Diameter vs. degree for random shortcut topologies generated using different base topologies ($N=2^{10}$ vertices).

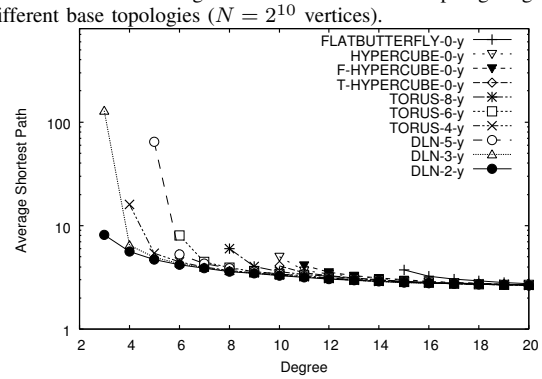

Figure 11. Average shortest path length vs. degree for random shortcut topologies generated using different base topologies ($N=2^{10}$ vertices).

In all previous results we have constructed our random shortcut topologies using DLN-2 as a base topology, i.e., the connected topology with the lowest possible degree. In this section we investigate whether improvements can be achieved by using other base topologies that have higher degrees but lower diameters and average shortest path lengths. We conduct experiments for $N=2^{10}$ vertices, generating random shortcut topologies using the method in the previous section but employing the following base topologies: DLN-2, DLN-3, DLN-5, TORUS-4, TORUS-6, TORUS-8, HYPERCUBE, F-HYPERCUBE-0, T-HYPERCUBE-0, and FLATBUTTERFLY-0. For each obtained topology we compute its diameter, average shortest path length, and fault-tolerance. As before, we only compare topologies with identical degrees (e.g., DLN-2-5 and TORUS-4-3). Results are shown in Figure 10 for the diameter and Figure 11 for the average shortest path length. As the degree increases, all generated topologies lead to increasingly lower and similar values. In other words, once many random shortcuts are added the choice of the base topology has no impact. The main observation, however, is that across the board random shortcut topologies generated from the DLN-2-y are never outperformed. In other words, among same-degree random shortcut topologies, the best topology is the one with the smallest number of non-random edges!

Comparing these same topologies in terms of fault-tolerance, no strong trends emerge and no base topology is always the best. Out of 117 one-to-one topology comparisons

between DLN-2-y and another random topology of the same degree but generated using a different base topology, DLN-2-y leads to a lower fault-tolerance in 56 of the cases. However, in only 15 of these cases is the difference larger than 5 (recall that fault-tolerance values are percentage values between 0 and 100), and the largest difference is below 14. These larger differences disappear once the degree is increased by one. We conclude that the fault-tolerance of the random shortcut topology is not sensitive to the choice of the base topology.

H. Guidelines for Generating Random Shortcut Topologies

Our graph analysis results provide several empirical guidelines for generating random shortcut topologies:

- The ring (DLN-2) topology is the base topology of choice when generating random shortcut topologies.
- Few random samples are needed to find a good DLN-2-y topology.
- Striving to pick high-quality random shortcuts (e.g., ones between vertices with long shortest paths) yields only marginal improvements.
- However, preserving base topology regularity by generating shortcuts based on random matchings leads to better results than adding shortcuts without considering vertex degree.

IV. Simulation Evaluation

In this section, we use discrete event network simulation to evaluate the performance of random shortcut topologies (DLN-x-y) and compare them to meshes (MESH-x-0), tori (TORUS-x-0), and hypercube family (HYPERCUBE-0 and F-HYPERCUBE-0). We do not evaluate T-HYPERCUBE-0 in this section. However, given the results in Section III, it is unlikely that this topology would lead to significant advantage over F-HYPERCUBE-0. Because discrete event simulation is compute intensive, we simulate networks with at most 512 switches. Note, however, that the analysis results in Section III show that the advantage of random shortcut topologies over non-random topologies increases as topologies grow in size.

A. Simulation Environment

We use a cycle-accurate network simulator written in C++ [24]. Every simulated switch is configured to use virtual cut-through switching. As in the previous section, all switches have the same number of ports. We only consider regular topologies, meaning that all switches connect directly to the same number of hosts (compute nodes) and to the same number of other switches. A model consisting of channel buffers, a crossbar, a link controller and the control circuits is used to simulate the switching fabric. A header flit transfer requires over 100 nanoseconds 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 20 nanoseconds. Routing in meshes, hypercubes, and folded hypercubes is done with the protocol proposed by Duato [25], and we use dimension-order routing for the escape path. For random shortcut DLNs we use a topology-agnostic adaptive routing scheme as described in [26], with up*/down* routing as the escape path. Once packets are sent along an escape path they cannot use adaptive paths, but most packets can take minimal paths. In our simulation, two virtual channels are used in all topologies. Tori use dimension-order routing, because the protocol in [25] requires three virtual channels.

We simulate 3 synthetic traffic patterns that determine each source-and-destination pair: *random uniform*, *bit-reversal*, *matrix-transpose*. These traffic patterns are commonly used for measuring the performance of large-scale interconnection networks, as described for instance in [27]. The hosts inject packets into the network independently of each other. In our 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 these relatively small packet sizes since we wish to study the performance of latency-sensitive traffic, which is expected to consist of small messages [1].

Our results show two important metrics: *latency* and *throughput*. The latency is the elapsed time between the generation of a packet at a source host and its delivery at a destination host. We measure latency in simulation cycles, and then convert to nanoseconds using 2.5ns per cycle. The throughput, measured in Gbps, is defined as the maximum accepted traffic, where the accepted traffic is the flit delivery rate.

B. Number of Shortcut Links

Figures 12 to 20 plot network latency vs. accepted traffic for 64-, 256-, and 512-switch topologies, in each case for uniform synthetic traffic (Figures 12, 15, and 18), for bit-reversal synthetic traffic (Figures 13, 16, and 19), and for matrix-transpose synthetic traffic (Figures 14, 17, and 20).

Overall, HYPERCUBE-0 or F-HYPERCUBE-0 leads to the best results among the non-random topologies in terms of latency and throughput. Still using same-degree comparisons, we see that DLN-2-y achieves latency up to 18% lower than HYPERCUBE-0 and up to 12% lower than F-HYPERCUBE-0 before network saturation in the case of uniform traffic, and up to 16% and 6% lower in the case of bit reversal traffic, respectively. This corroborates our results in Section III in the sense that latency is expected to be correlated with diameter and average shortest path length. Importantly, DLN-2-y and HYPERCUBE-0 achieve similar throughput, and in the case of 512-Switch topologies DLN-2-8 achieves 36% throughput improvement over F-HYPERCUBE-0. Turning to lower-degree topologies, we see that the advantage of DLN-2-y is more pronounced. For instance, DLN-2-4 reduces latency by up to 27% compared to that of TORUS-6-0, and improves throughput by up to 43%. Throughput is affected both by traffic balance and average shortest path lengths, and DLN-2-y achieves a good trade-off between the two, in part because it can achieve low average shortest path length at low degree. Our results show that as N increases, the TORUS/MESH-4-0 and TORUS/MESH-6-0 topologies do not scale well.

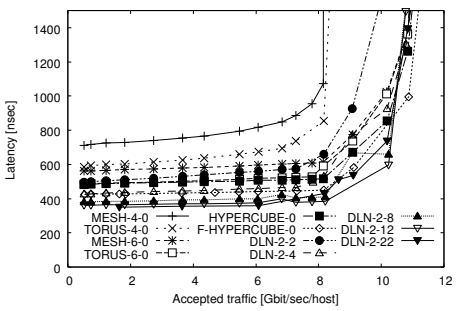

Figure 12. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (64 switches, 256 hosts, uniform).

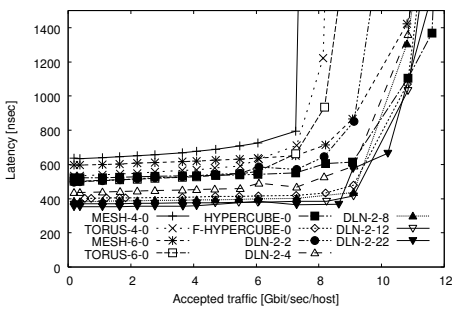

Figure 13. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (64 switches, 256 hosts, bit reversal).

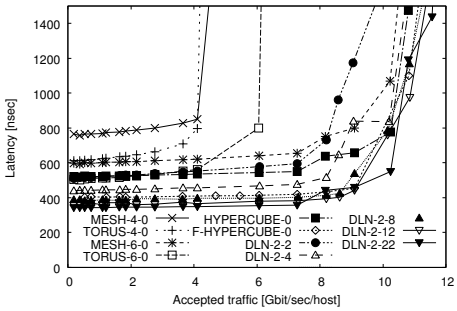

Figure 14. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (64 switches, 256 hosts, matrix transpose).

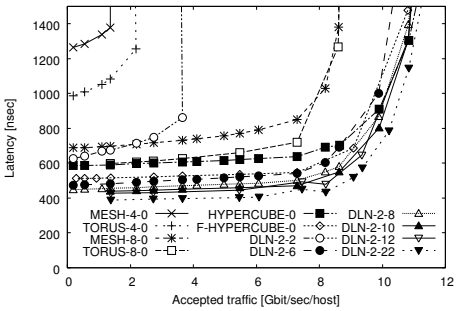

Figure 15. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (256 switches, 2,048 hosts, uniform).

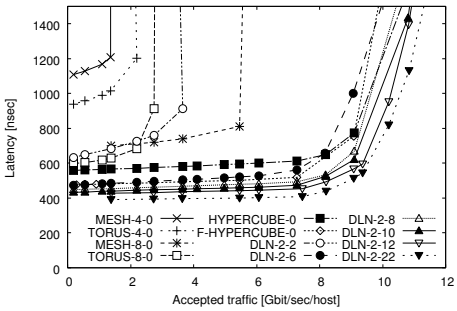

Figure 16. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (256 switches, 2,048 hosts, bit reversal).

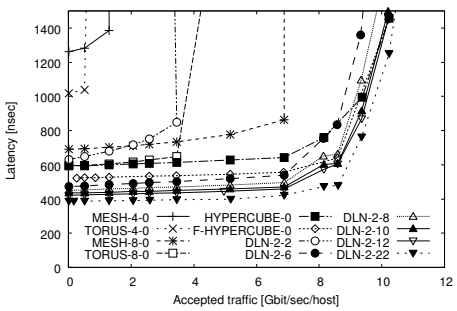

Figure 17. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (256 switches, 2,048 hosts, matrix transpose).

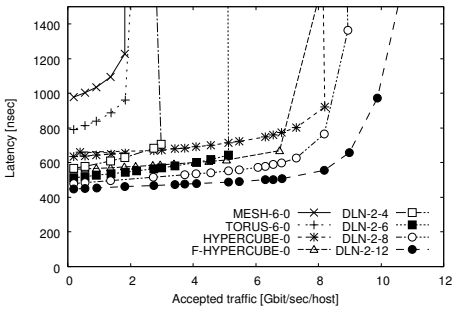

Figure 18. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (512 switches, 8,192 hosts, uniform).

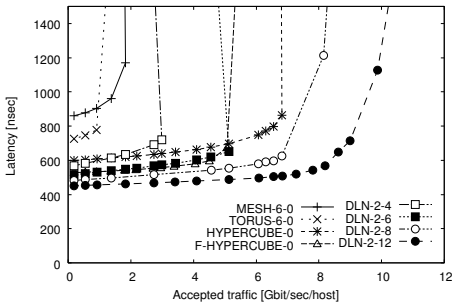

Figure 19. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (512 switches, 8,192 hosts, bit reversal).

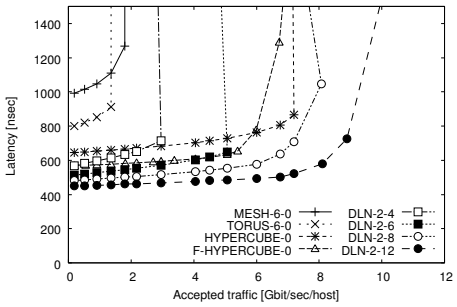

Figure 20. Latency vs. accepted traffic for non-random topologies and random shortcut topologies (512 switches, 8,192 hosts, matrix transpose).

By contrast, the latency and throughput of HYPERCUBE-0, F-HYPERCUBE-0 and DLN-2-y degrade gracefully as N increases. Finally, we observe that DLN-2-y leads to good results regardless of the traffic pattern.

A valid concern with the DLN random topology is that of performance variability between different random instances of the DLN-2-y configuration. In Section III-E, we have seen that diameter and average shortest path length variability across different DLN-2-y samples is low. It turns out that variability is also low both for throughput and latency. We conducted experiments for 20 different sample 256-switch DLN-2-6 topologies, i.e., each sample contains different shortcuts. Figures 21 and 22 show the lowest and highest achieved latency over these 20 samples vs. accepted traffic for both the matrix-transpose and bit-reversal synthetic traffic patterns (results are identical for the uniform traffic pattern). The main result is that both curves are close to each other, with a relative difference at most 0.2%.

C. Fault Tolerance

Distributed dynamic or static network reconfiguration techniques to update routing information upon port/link failures are well developed on irregular topologies [28]. Such techniques can be applied to random shortcut topologies as well. In this section, we evaluate the impact of link failures on throughput and latency once the network has been reconfigured to route "around" the failed links.

We perform simulation experiments for a DLN-2-6 topolo-

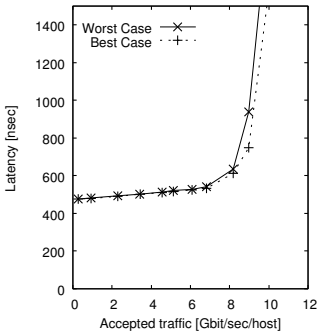

Figure 21. Latency vs. accepted traffic for 20 random shortcut patterns of DLN-2-6 (256 switches, bit reversal).

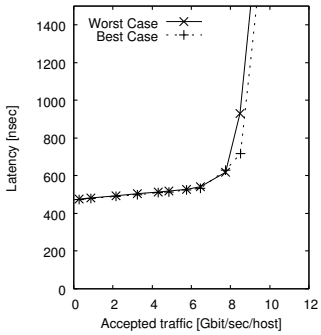

Figure 22. Latency vs. accepted traffic for 20 random shortcut patterns of DLN-2-6 (256 switches, matrix transpose).

gies with 256 switches. We randomly remove 0%, 5%, 10%, and 20% of the network links, and simulate the uniform, bit-reversal, and matrix-transpose synthetic traffics used in the previous section. Recall that even links that are part of the original ring may be removed. For the random shortcut DLN we employ topology-agnostic routing even when there are no faults, as opposed to standard non-random topologies that employ custom routing to exploit the regularity of their structures in fault-free conditions. In the results presented hereafter the topology remains connected after link removal.

Figures 23 to 25 show latency and accepted traffic results for uniform and bit reversal traffic patterns, respectively. The key observation here is that latency and throughput decrease gracefully regardless of the traffic pattern, which indicates that random shortcut DLN topologies are robust to link failures.

V. Discussion of Limitations

A. Case Study: Random Shortcut Links on Myrinet-Clos

A popular topology in HPC systems is the indirect Myrinet-Clos [29], and in this section we investigate the impact of adding random shortcuts to it. We consider a 256-host three-layer Myrinet-Clos with 80 16-port switches. Hosts are split in four 64-host (non-blocking) Clos groups each with 16 switches (including leaf and midspine).

We consider two ways in which this topology can be randomized. First, we consider the *swapping* of end-points between randomly selected leaf-midspine links. We experiment with swapping 20%, 50%, or 80% of the links. We also consider *adding* random links to the topology, thus increasing the degree. We experiment with adding 1 or 2 random links per switch. We also add 8 or 16 random links per 64x64 Clos group. This corresponds to adding 1 or 2 links per switch so that each group is randomly connected to a midspine block with 8 or 16 links. Latency and accepted traffic results for all these variants, and for the original topology, are shown in Figure 26.

Some of the results in this figure reveal that using random shortcuts decreases throughput significantly, although random shortcuts are always beneficial for latency, as is expected due to the reduced average shortest path length. Myrinet-Clos provides well distributed paths, with a relatively low diameter. Replacing original links by random links, or adding random links between midspine and leaves, introduces imbalance that reduces the achievable throughput. The only cases in which the throughput is not impacted, and in fact slightly improved, is when random links are added to 64x64 blocks. This is because these additions do not create imbalance since each block is connected to each midspine blocks.

These results show that randomness is no always beneficial in terms of throughput, in this case due to path imbalance. Given a base topology, identifying topological features that determine whether the addition or random shortcuts would be beneficial or detrimental is an interesting open problem.

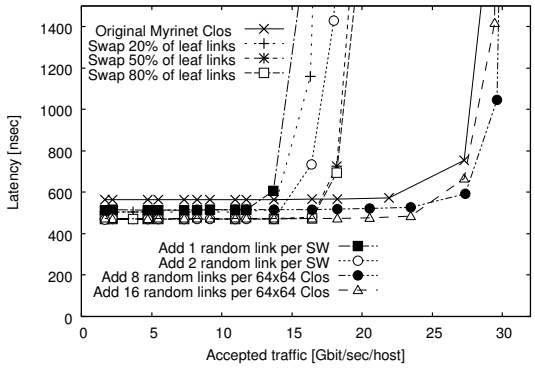

Figure 26. Latency vs. accepted traffic for random shortcut patterns on Myrinet Clos (80 switches, 256 hosts).

B. Routing Scalability

Custom routing implementations in large-scale HPC systems that use non-random topologies can exploit topological regularity, e.g., dimension-order routing, to make routing logic simple and small. By contrast, random shortcut topologies cannot rely on such schemes because the topology does not have a simple structure. Source routing or distributed routing using routing tables is thus needed. Consequently, the scale of random shortcut topologies can be limited by routing table size at each switch. However, we note that 87% of the supercomputers posted on the November 2011 Top500 list [30] are based on Ethernet or InfiniBand. For all these systems, the routing table size is thus also a scalability limitation regardless of the topology used. Techniques have been developed to compact routing table entries, such as region-based routing in network-on-chip settings[31], and can be used to mitigate this scalability concern.

Another potential scalability issue is the computational cost of path computation for topology-agnostic deadlock-free routing, which is more complex than when routing on structured topologies (see the survey in [7]). Essentially, the complexity of the path computation is $O(N^2)$ or higher, where N is the number of switches. However, this computation needs to be performed only when initially deploying the system and after a change in system configuration (e.g., due to link/switch failure). Consequently, concerns about path computation should

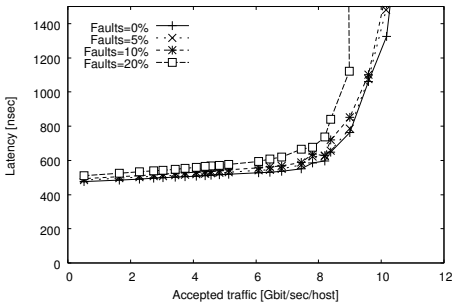

Figure 23. Latency vs. accepted traffic for faulty DLN-2-6 (256 switches, 2,048 hosts, uniform).

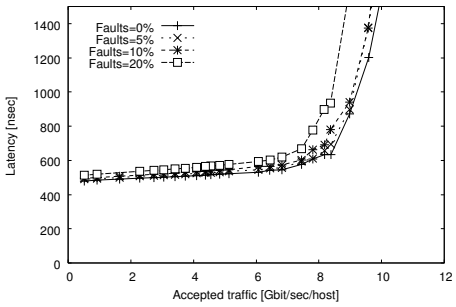

Figure 24. Latency vs. accepted traffic for faulty DLN-2-6 (256 switches, 2,048 hosts, bit reversal).

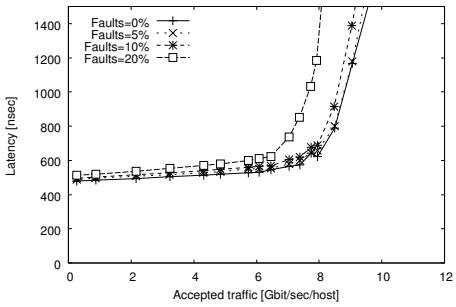

Figure 25. Latency vs. accepted traffic for faulty DLN-2-6 (256 switches, 2,048 hosts, matrix transpose).

arise only at extremely large scale.

C. Physical Cable Length and Maintainability

Beyond the metrics that we have already investigated in previous sections, another practical concern when deploying an HPC interconnect is physical cable length, which impacts cost [13]. If the cable length for inter-cabinet connection is one or more orders of magnitude longer than the intra-cabinet connection, inter-cabinet cables would need to be optical. In the case of InfiniBand, typical maximum length of passive copper is 10m, that of active copper is 40m, connectorized copper is 30m, and embedded optical is 100m [12]. Aggregate cable length can reach astronomical proportions, e.g., about two thousands kilometers in the first Earth Simulator platform [32]. The addition of random shortcuts can further increase cable length, and thus cost. In this section we approximately quantify the impact of random shortcuts on average cable length, comparing DLN-2-y to tori and hypercubes.

We estimate average cable length based on the parameters and the layout presented in [11]: 128 nodes per cabinet, 0.57m $\times$ 1.44m cabinet footprint, 2m cable overhead when wiring inter cabinet cables, and 75 nodes/m^2 density. The inter cabinet cables are set to be 2m, and eight hosts are connected to a switch. We assume that each cabinet is located in a 2-D dimensional square. To make the wiring layout as regular as possible, each cable is not drawn as a straight line between two components, but instead drawn as a sequence of orthogonal horizontal and vertical segments inside the 2-D square. In other words, cable lengths are computed using the Manhattan distance. We consider two layout patterns. In the *distributed* layout, each switch is located near its local hosts in the same cabinet while in the *centralized* layout all switches are in central cabinets that only store switches. Although traditional direct topologies typically fit with the distributed layout, for high-radix switches with larger number of links to switches than to local hosts the centralized layout may be preferable.

Figures 27 and 28 plot average cable length, for both inter- and intra-cabinet links between switches, for both layouts. All topologies have the same number of same-length links between hosts and switches, and these lengths are not included in the computation of the average cable length.

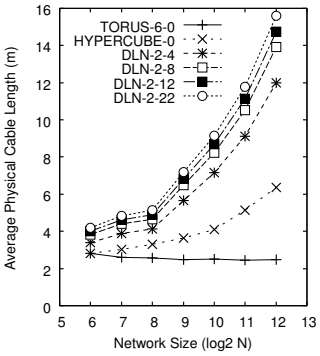

Figure 27. Average cable length vs. N for non-random topologies and random shortcut DLN with distributed switch layout.

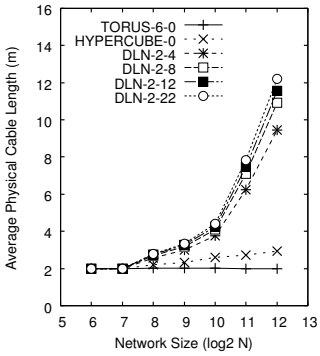

Figure 28. Average cable length vs. N for non-random topologies and random shortcut DLN with centralized switch layout.

Not surprisingly, the centralized switch layout reduces the number of inter-cabinet cables between switches compared to the distributed switch layout. The average cable length of TORUS-6-0 is low across the board. This is because all wires for two-dimensional surfaces are stored within a single cabinet, due to a 3-D natural mapping to the layout, whether centralized or distributed. In the case of the centralized switch layout, the hypercube also fits nicely within the central cabinets. At any rate, these results show that the random shortcut DLN increases the average cable length by about 2m in the centralized switch layout when $N = 2^{10}$, or about 10m when $N = 2^{12}$. While this is a significant increase, the important point is that it is not so high as to mandate the replacement of electrical cables by optical cables. This increase may thus be a small price to pay given the large benefits in terms of diameter and average shortest path length, at least for up to $N = 2^{12}$ switches.

Another cable-related concern is maintainability. Once an HPC system is built on a floor, wiring a new cable below or above this floor is costly. Over many years a fraction of the cables incur soft errors that mandates cable replacement. Consequently, some recent HPC systems with non-random topologies initially use backup cables, which is costly. By contrast, random shortcut topologies with topology-agnostic routing can work well on any set of switches and links and

do not require backup cables.

VI. Conclusions

We have proposed the use of random shortcuts in network topologies primarily as a way to improve communication latency. We have shown that, when comparing to classical topologies of identical degree, random shortcuts improve the diameter by up to one order of magnitude. For instance, a ring with two random shortcuts per switch for a total degree of 4 has a smaller diameter than a hypercube with degree 12 in the case of $N = 4,096$ switches. Similar improvements are achieved for average path length. Random shortcuts also bring benefits in terms of fault tolerance, meaning that a large fraction of network links can fail without the diameter increasing by more than two hops. Results from flit-level discrete event simulation have shown that random shortcuts can reduce communication latency by up to 18% when compared to a hypercube topology and by up to 12% to a folded hypercube topology, while leading to similar or higher throughput. More generally, as the number of shortcuts increases, both the latency and throughput improve.

We have determined that a simple method to pick shortcuts applied to a ring base topology leads to a good random shortcut topology, which we called DLN-2-y. To enable comparison to other topologies, we have only generated random shortcut topologies with numbers of switches that are powers of two. However, an advantage of DLN-2-y, unlike other non-random topologies, is that it does not impose any constraints on the total number of switches in the network. As a result the scale of the network can be determined solely based on space, power, cooling, and budget constraints. Finally, we have investigated limitations of our approach, most importantly in terms of routing scalability and cable length. We have found that although challenges arise at very large scale, high-radix DLN-2-y can be used to support systems with hundreds of thousands of compute nodes.

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] 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.

[7] 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 Transactions on Parallel and Distributed Systems*, vol. 23, no. 3, pp. 405–425, 2012.

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

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

[10] 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.

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

[13] 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.

[14] K. J. Barker, A. F. Benner, R. R. Hoare, A. Hoisie, A. K. Jones, D. J. Kerbyson, D. Li, R. G. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. B. Stunkel, and P. Walker, "On the Feasibility of Optical Circuit Switching for High Performance Computing Systems," in *Proc. of SC*, 2005, p. 16.

[15] J.-C. Bermond, F. Comellas, and D. F. Hsu, "Distributed Loop Computer Networks: A Survey," *J. Parallel Distrib. Comput.*, pp. 2–10, 1995.

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

[17] H. Fuks and A. T. Lawniczak, "Performance of data networks with random links," *CoRR*, vol. adap-org/9909006, 1999.

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

[19] F. Bonnet, A.-M. Kermarrec, and M. Raynal, "Small-world networks: From theoretical bounds to practical systems," in *OPODIS*, 2007, pp. 372–385.

[20] V. Nguyen and C. Martel, " Designing Low Cost Networks with Short Routes and Low Congestion," in *Proc. of the International Conference on Computer Communications (INFOCOM)*, 2006, pp. 1–12.

[21] A. El-Amawy and S. Latifi, "Properties and Performance of Folded Hypercubes," *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 1, pp. 31–42, 1991.

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

[23] "Snap network analysis library," http://snap.stanford.edu/.

[24] A. Jouraku, M. Koibuchi, and H. Amano, "An Effective Design of Deadlock-Free Routing Algorithms Based on 2-D Turn Model for Irregular Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, pp. 320–333, Mar. 2007.

[25] J. Duato, "A Necessary And Sufficient Condition For Deadlock-Free Adaptive Routing In Wormhole Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 6, no. 10, pp. 1055–1067, 1995.

[26] F. Silla and J. Duato, "High-Performance Routing in Networks of Workstations with Irregular Topology," *IEEE Transactions on parallel and distributed systems*, vol. 11, no. 7, pp. 699–719, 2000.

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

[28] T. M. Pinkston, R. Pang, and J. Duato, "Deadlock-Free Dynamic Reconfiguration Schemes for Increased Network Dependability," *IEEE Transactions on Parallel and Distributed Systems*, vol. 14, no. 8, pp. 780–794, 2003.

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

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

[31] A. Mejia, M. Palesi, J. Flich, S. Kumar, P. López, R. Holsmark, and J. Duato, "Region-based routing: A mechanism to support efficient routing algorithms in nocs," *IEEE Transactions on VLSI Systems*, vol. 17, no. 3, pp. 356–369, 2009.

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