Skywalk: a Topology for HPC Networks with Low-delay Switches

Ikki Fujiwara*, Michihiro Koibuchi*†
*National Institute of Informatics / JST
†The Graduate University for Advanced Studies (SOKENDAI)
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo, JAPAN 101-8430
Email: {ikki, koibuchi}@nii.ac.jp

Hiroki Matsutani‡
‡Keio University
3-14-1 Hiyoshi, Kohoku-ku, Yokohama, Kanagawa, JAPAN 223-8522
Email: matutani@arc.ics.keio.ac.jp

Henri Casanova§
§University of Hawaiʻi at Manoa
1680 East-West Road, Honolulu, HI, U.S.A. 96822
Email: henric@hawaii.edu

***Abstract*—With low-delay switches on the horizon, end-to-end latency in large-scale High Performance Computing (HPC) interconnects will be dominated by cable delays. In this context we define a new network topology, Skywalk, for deploying low-latency interconnects in upcoming HPC systems. Skywalk uses randomness to achieve low latency, but does so in a way that accounts for the physical layout of the topology so as to lead to further cable length and thus latency reductions. Via graph analysis and discrete-event simulation we show that Skywalk compares favorably (in terms of latency, cable length, and throughput) to traditional low-degree torus and moderate-degree hypercube topologies, to high-degree fully-connected Dragonfly topologies, to the HyperX topology, and to recently proposed fully random topologies.**

***Keywords*—Interconnection network, network topology, cabinet layout, high performance computing**

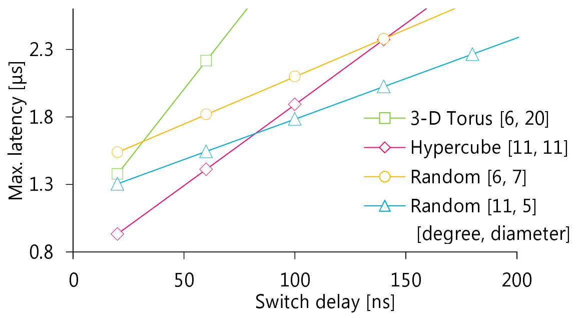


Fig. 1. Maximum latency of shortest-hop paths vs. switch delay in 256-cabinet, 2,048-switch networks. Links between switches and compute nodes are not considered. For each topology, the legend indicates degree and diameter.

I. Introduction

An acknowledged objective of next generation High Performance Computing (HPC) systems is to achieve ultra-low end-to-end latencies, e.g., under 1 μs across an exascale system [1]. This objective is being partially achieved thanks to decreasing switch delays. At the time of this writing Cisco's SFS7000D InfiniBand DDR switch achieves less than 200 ns, QLogic's 12300 InfiniBand QDR switch achieves 140 ns, and some Mellanox switches achieve less than 100 ns delays. Furthermore, the clock rate of commercial ASIC switching fabric (e.g., 200MHz–1GHz under 65nm or 90nm) is behind state-of-the-art processor technology. One can thus expect the latency (which correlates with data rate and clock rate) to decrease in upcoming technology, even if new functionalities are added. Finally, the pipeline structure of switches may be improved. For instance, the RHiNET-2/SW switch has a 160 ns pipeline delay (125 MHz, 20 cycles), and the RHiNET-3/SW switch has a 240 ns pipeline delay (100MHz, 24 cycles) [2]. These pipelines include various stages (e.g., routing computation, switch allocation, output allocation, switch transfer, ECC decoder and encoding). Aggressive speculation may allow to execute some stages in parallel so as to decrease delay [3].

The main motivation for this work is that once switch delays become very low (e.g., 60 ns), cable delays (e.g., 5 ns/m) will dominate the latency in large-scale topology deployments. Consequently, it will be physical cable lengths rather than hop counts that drive network latencies in future large-scale HPC systems.

Figure 1 shows the maximum end-to-end latency vs. the switch delay for a 2,048-switch network laid out in 256 cabinets. Data points are shown for the traditional hypercube and torus topologies, and for two fully random topologies with different degrees. For each topology the legend in the figure indicates degree and diameter (i.e., the hop count of the longest shortest path). A conventional shortest-hop routing method is assumed. See Section III for all details on the topologies, their physical layout in cabinets, and the models used for computing latency and cable length. Random topologies have been proposed recently as a practical way to achieve low diameter with only moderate degree, thus leading to low latencies [4]–[6]. In spite of their low diameters, the random topologies lead to higher latency than a hypercube when switch delay is low, say below 60 ns. For hypothetical 20 ns switch delays, the random topologies would even have higher latencies than tori! The overall lesson is that, when switch delays are comparable to or lower than link delays, one should not design topology solely (or at all) to reduce diameter.

In this work we propose a new variable-degree topology, Skywalk, combined with the use of a routing scheme, to achieve low end-to-end network latencies in the upcoming low-delay switch era. Skywalk uses randomness to reduce latency, but it does so in a way that accounts for the physical layout

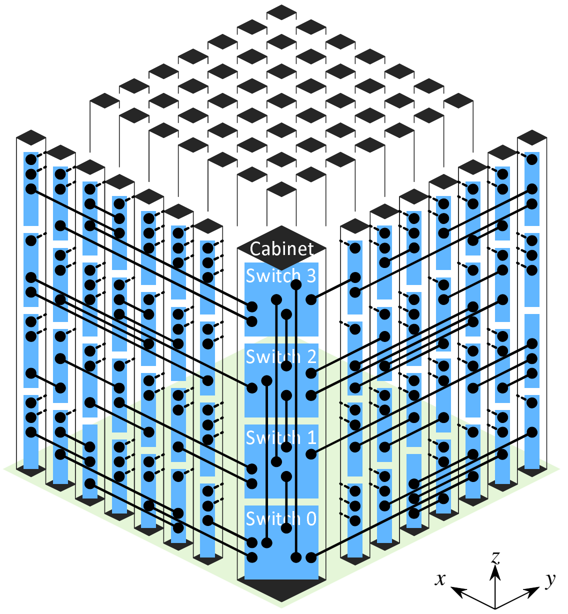


Fig. 2. An instance of Skywalk with 256 switches ($z = 4$ switches in each of $c = 64$ cabinets), with at each switch $d_\mathrm{i} = 3$ intra-cabinet links, $d_\mathrm{s} = 3$ inter-cabinet straight links, and $d_\mathrm{d} = 0$ inter-cabinet diagonal links.

of the topology. Our main findings are that:

- When compared to traditional low- to moderate-degree tori and hypercubes, a same-degree Skywalk leads to lower latency and lower cable length;
- When compared to the high-degree, low-latency fully-connected Dragonfly topology [7], a smaller-degree Skywalk leads to only marginally higher latency while reducing cable length by several factors;
- A similar observation can be made when comparing Skywalk to the high-degree HyperX topology [8], albeit with a more modest cable length reduction;
- When compared to recently proposed low-latency fully random topologies, a same-degree Skywalk leads to improvements in both latency and cable length;
- Even though Skywalk is primarily designed to reduce latency, it achieves relatively high throughput without requiring high degree.

This paper is organized as follows. Section II introduces Skywalk. Sections III and IV evaluate Skywalk and compare it to previously proposed topologies using graph analysis and discrete-event simulation, respectively. Section V discusses related work. Finally, Section VI summarizes our contributions and gives guidelines for deploying Skywalk in practice.

II. The Skywalk Topology

A. Rationale

Reducing hop count, i.e., the number of links along paths between each source and destination compute node, has been a main focus of network topologies proposed for HPC interconnects. Recently, randomly generated topologies have received a fair amount of attention because they have low diameter at low degree and afford other attractive features (fault-tolerance, arbitrary numbers of vertices, expandability) [4]–[6]. However, in a future in which switch delays are below 100 ns, shortening physical cable lengths along each path can become more crucial than reducing hop counts. For instance, a switch delay of 60 ns is equivalent to a link delay on a 12 m cable (optical cable delays are 5 ns/m). Inter-cabinet optical links in physical deployments can reach tens of meters and would then be the main contributors to end-to-end delay. Note that large cable lengths also contribute to large topology deployment costs.

Based on these observations, we propose a topology that is randomized, but that also distinguishes between intra-cabinet and inter-cabinet sub-topologies. The objective is to minimize out-of-cabinet cable lengths while requiring a reasonably low number of cables and switch ports.

TABLE I. Parameters of Skywalk.

c	Number of cabinets
z	Number of switches in a cabinet
d_i	Number of intra-cabinet links at a switch
d_s	Number of inter-cabinet straight links at a switch
d_d	Number of inter-cabinet diagonal links at a switch

B. Topology Specification

We assume a grid-like alignment of cabinets on a machine room floor, as depicted in Figure 2. We name our topology Skywalk by analogy to skywalks between skyscrapers in an urban center. Skywalk, like any cabinet-conscious topology that distinguishes the intra-cabinet layer and the inter-cabinet layer, can be seen as an instance of the Dragonfly meta-topology in [7].[1]

Table I lists the parameters that define our topology. Given c cabinets, each cabinet hosts z switches for a total of $N = zc$ switches. These cabinets are arranged in a 2-D grid on a machine room floor. Skywalk consists of three separate sub-topologies for (i) intra-cabinet links, (ii) inter-cabinet straight links (along a cabinet row or column), and (iii) inter-cabinet diagonal links (spanning multiple cabinet rows and columns). d_i, d_s, and d_d denote the number of intra-cabinet, inter-cabinet straight, and inter-cabinet diagonal links at each switch, respectively. We define d_o, the number of inter-cabinet links at a switch as $d_\mathrm{o} = d_\mathrm{s} + d_\mathrm{d}$. The degree of the topology, i.e., the maximum number of links at a switch excluding links between the switch and the compute nodes, is $d = d_\mathrm{i} + d_\mathrm{o}$. The upper bound on d_i is $d_\mathrm{imax} = z - 1$. Given an $x \times y$ grid of cabinets, the upper bound on d_s is $d_\mathrm{smax} = \lceil (x + y - 2)/z \rceil$ and the upper bound on d_d is $d_\mathrm{dmax} = \lceil (x-1)(y-1)/z \rceil$. If $d_\mathrm{i} = d_\mathrm{imax}$ and $d_\mathrm{o} = d_\mathrm{smax} + d_\mathrm{dmax}$, then Skywalk is identical to the fully-connected Dragonfly topology in [7].

In this work we use the term "switch" to refer to a small cluster of m compute nodes connected to a single switch. Therefore, if a p-port switch is used, then we must have $d + m \leq p$, since d does not include the number of links between a switch and its compute nodes. The total number of compute nodes is mN.

[1]The original Dragonfly is described as "three-layered," including compute nodes, but in this work we ignore compute nodes and consider only intra- and inter-cabinet links.

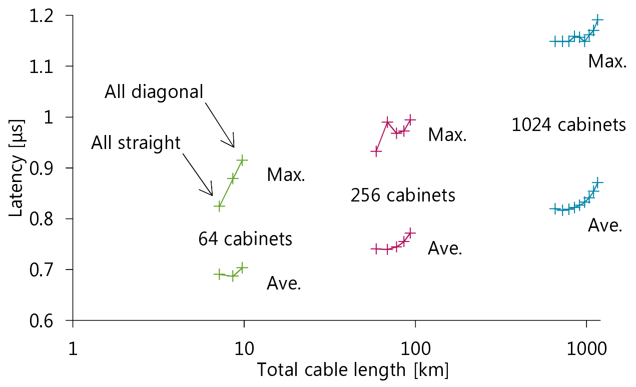


Fig. 3. Maximum and average latency vs. total cable length for Skywalk with 8 switches per cabinet and 64, 256, and 1024 cabinets, for different mixes of straight and diagonal links.

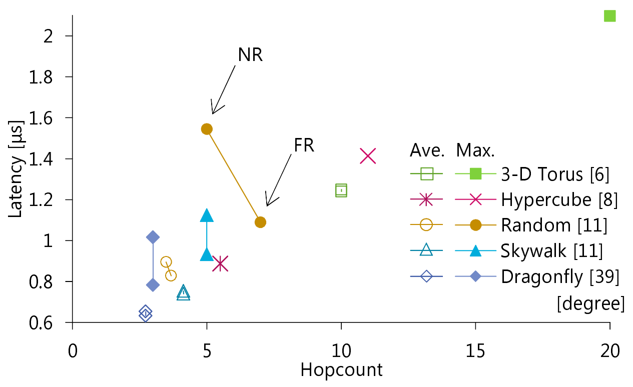


Fig. 4. Maximum and average latency vs. hop count, depending on routing scheme, in 256-cabinet, 2,048-switch networks of 60-ns switches.

C. Topology Construction

Each sub-topology of Skywalk employs a d_k-degree Uniform Random topology ($d_k \in \{d_\mathrm{i}, d_\mathrm{s}, d_\mathrm{d}\}$). A Uniform Random topology is generated by the following algorithm, where V denotes the set of all relevant vertices (switches or cabinets, depending on which sub-topology is being constructed):

1) Make a copy of V, say W;
2) Randomly pick a vertex v_1 in W;
3) Randomly pick another vertex v_2 in W so that v_1 and v_2 are not already connected and satisfy some constraints (described below);
4) If such a v_2 is found connect v_1 and v_2 with a link, otherwise do nothing;
5) Remove v_1 and v_2 (if it was found) from W;
6) If W is not empty go to step 2;
7) Repeat d_k times.

Links between vertices v_1 and v_2 above are constrained for each sub-topology so that the links are either (i) intra-cabinet, (ii) inter-cabinet straight, or (iii) inter-cabinet diagonal. When the vertices are the cabinets, for cases (ii) and (iii), a new link is added between two cabinets by picking its endpoint switches in a cyclic manner. In other words, for each cabinet we keep track of the index of the switch that should be the endpoint of the next inter-cabinet link involving this cabinet, and increment this index once that link is added.

Due to randomness in the algorithm, the constructed instances have different number of links, and they may even be disconnected if the degree is very low. We construct 10 topology instances, each obtained with a different random seed of the random number generator, and pick among these 10 instances the one that is connected and has the largest number of generated links. Although results are not presented here due to lack of space, we have confirmed that using 10 random samples is sufficient to obtain stable results (i.e., all samples are connected, most of them have almost identical numbers of links while only a few have notably fewer links).

D. Straight vs. Diagonal Links

Given a specified number of inter-cabinet cables, $d_\mathrm{o} = d_\mathrm{s} + d_\mathrm{d}$, one must pick appropriate d_s and d_d values. In other words, which fraction of the inter-cabinet links at a switch should be straight or diagonal? We assume rectilinear ("Manhattan") cabling between cabinets, as done in practice for physical deployments of production topologies. With this cabling scheme diagonal links lead to longer cable lengths, thus suggesting building Skywalk by using straight links whenever possible rather than diagonal links.

To verify this intuition Figure 3 shows maximum and average latency vs. total cable length for three instances of Skywalk, each with $z = 8$ switches per cabinet and c cabinets, with $c \in \{64, 256, 1024\}$. The models used to compute latency and cable length are described in Section III, and the routing scheme is described in Section II-E. For each topology instance several data points are shown. Each data point corresponds to a different $(d_\mathrm{s}, d_\mathrm{d})$ couple, ranging from $(d_\mathrm{o}, 0)$ (only straight cables) to $(0, d_\mathrm{o})$ (only diagonal cables), and going from left to right along the horizontal axis since the total cable length increases as d_d increases. For simplicity, in this experiment we assume that $d_\mathrm{o} = \min(d_\mathrm{smax}, d_\mathrm{dmax})$.

We see that for the three Skywalk instances the maximum latency is smallest when all links are straight links. Interestingly, the maximum latency does not increase monotonically with the number of diagonal links. In the case of the average latency, we find that using only straight links does not necessarily lead to the lowest value, but to a value close to the lowest value. For instance, for $c = 64$ cabinets, $(d_\mathrm{o}, 0)$ leads to average latency 0.50% higher than $(d_\mathrm{o} - 1, 1)$. As the scale of the topology increases this difference becomes smaller. For instance, it becomes only 0.38% for $c = 1024$ cabinets.

We conclude that prioritizing straight links over diagonal links leads to the lowest total cable length, the lowest maximum latency, and close to the lowest average latency. Consequently, unless specified otherwise, in the rest of this paper we use the following *straight-first* linking scheme. Given a specified number of inter-cabinet links at a switch d_o, we first create the maximum number of possible straight links, up to $d_\mathrm{s} = \min(d_\mathrm{o}, d_\mathrm{smax})$ per switch. We then create diagonal links to use the possibly remaining switch ports, up to $d_\mathrm{d} = \min(d_\mathrm{o} - d_\mathrm{s}, d_\mathrm{dmax})$ per switch.

E. Routing

Most existing routing algorithms rely on a shortest-hop path search technique [9], which we call Nearest Routing (NR) in this work. As switch delay decreases and no longer dominates end-to-end network latencies, this routing approach is no longer sensible. Instead, routing should be done along lowest-latency paths, which we call Fastest Routing (FR). Such paths can be computed using Dijkstra's well-known polynomial-time algorithm by defining distances in a way that accounts for cable delays and switch delays. Using a priority-queue implementation of the algorithm [10], we find for instance that all paths can be computed for a 16k-switch Skywalk in under 140 seconds on a 3.47 GHz Intel Xeon X5690 processor. Therefore, path computation to be used with topology-agnostic deadlock-free routing can be done at large scale in practice. To implement a deterministic topology-agnostic routing, we can use either table-based distributed routing or source routing.

We carry out a simple experiment to quantify the advantage of FR vs. NR when switch delay is 60 ns. Figure 4 shows the maximum and the average latency vs. the corresponding hop count when using NR and FR for various topologies, assuming 2,048 switches in 256 cabinets. We refer the reader to Section III for all details on the topologies, their physical layout in cabinets, and the models used for computing network latencies. These results show that FR achieves either equal or smaller latency when compared to NR, while possibly increasing the hop count. In the rest of this paper we only present results using FR.

F. *Expandability and Maintainability*

An advantage of random topologies is that they can be easily expanded to increase the scale of the network [4]–[6]. They can also be easily maintained because, provided they have a sufficient number of cables, faulty cables do not have a high impact on the overall performance of the network. Similarly, Skywalk can be expanded and maintained easily after an initial deployment, though such incremental expansion is of course possible only if existing switches have available ports. If the deployed switches are at full port capacity, then a time-consuming and expensive re-cabling of the topology must be done to include new switches in possibly new cabinets. Note that, in this case, the expanded Skywalk will not achieve as high a performance because its degree is unchanged while its scale is increased.

III. Graph Analysis

A. *Physical Layouts and Network Topologies*

We consider topologies deployed as sets of compute nodes, switches, and cables for the following physical layout. The compute nodes and the switches are enclosed in cabinets arranged on a machine room floor in a 2-D grid pattern. The cabinets are 0.6 m wide and 2.1 m deep including space for the aisle, following the recommendations in [11]. Given c cabinets, we assume an $x \times y$ grid, with $x = \lceil \sqrt{c}\ \rceil$ (number of cabinet rows) and $y = \lceil c/x \rceil$ (number of cabinet columns). We compute cable lengths as follows. The intra-cabinet cables are 2 m long. The inter-cabinet cables have 4 m overhead (2 m at each end) in addition to the Manhattan distance between source and destination cabinets. This cable length computation is similar to but more conservative than that in [12].

Given the above physical layout and the notations in Table I, we consider five topologies:

- Skywalk$(z, c, d_{\mathrm{i}}, d_{\mathrm{o}})$: A Skywalk topology of degree $d_{\mathrm{i}} + d_{\mathrm{o}}$. If $d_{\mathrm{o}} \leq d_{\mathrm{smax}}$ then $d_{\mathrm{s}} = d_{\mathrm{o}}$ and $d_{\mathrm{d}} = 0$; otherwise $d_{\mathrm{s}} = d_{\mathrm{smax}}$ and $d_{\mathrm{d}} = d_{\mathrm{o}} - d_{\mathrm{smax}}$ (see Section II-D).
- Dragonfly(z, c, d): A *fully-connected* Dragonfly topology of degree $d = z-1+\lceil \frac{c-1}{z} \rceil$, as described in [7]. A cabinet corresponds to a "group of routers" or a "virtual router" in [7]. All the switches (routers) in a cabinet (group) are fully connected, and all the cabinets (groups) are also fully connected.
- HyperX(z, c, d): A HyperX topology of degree $d = x + y + z - 3$ tailored to map onto the physical layout of cabinets as described in [8]. Note that the popular Flattened Butterfly topology [12] is subsumed by the HyperX topology.
- Hypercube(N, d): A d-dimensional hypercube of degree $d = \lceil \log_2 N \rceil$.
- Torus(z, c, d): An n-dimensional Torus topology of degree $d = 2n$. We set the size of first dimension to z and the remaining $n-1$ dimensions to $c^{1/(n-1)}$. For instance, a 3-D Torus with 2,048 switches, 256 cabinets, with 8 switches per cabinet, is generated as a $8 \times 16 \times 16$ torus.
- Random(N, d): An unstructured Uniform Random topology [4]–[6] of degree d generated using the method in Section II-C, but considering all N individual switches as vertices, without imposing any constraints on the links.

When deploying a topology of switches over a cabinet layout an important question is that of assigning each switch to a particular cabinet. Skywalk, Dragonfly, and HyperX are constructed given the physical layout of the switches in the cabinets (rather than mapping the switches to cabinets once a topology graph has been constructed). For Torus and Hypercube, we assign switches to cabinets sequentially considering switches in the topological order (i.e., by increasing order of the canonical numbering of the vertices), as done in production physical deployments. For Random the switches are assigned arbitrarily to the cabinets.

Given the above and a switch delay value, for each topology and its physical deployment into cabinets we can compute the end-to-end latencies of all network paths. For this computation we assume that cable delay is 5 ns/m and that the packet injection/reception delay is 300 ns (150 ns at each end). We present results for the maximum and the average latency. Another important metric to evaluate the performance of a topology is the bisection bandwidth. We refer the reader to Section IV in which we present simulation results for various traffic patterns that quantify the achieved throughput, which is correlated with the bisection bandwidth.

B. *Switch Delay*

Before presenting more detailed analysis results, we assess whether Skywalk achieves its main objective: to provide low end-to-end latencies in a future in which low-delay switches are available. Figure 5 shows latency vs. switch delay for a physical deployment of 2,048 switches over 256 cabinets using each of the topologies in the previous section: Skywalk$(8, 256, 7, 4)$, Dragonfly$(8, 256, 39)$, HyperX$(8, 256, 37)$, Hypercube$(2048, 11)$, Torus$(8, 256, 6)$, and Random$(2048, 11)$. These results, unlike those in Figure 1, are obtained using Fastest Routing instead of Nearest Routing.

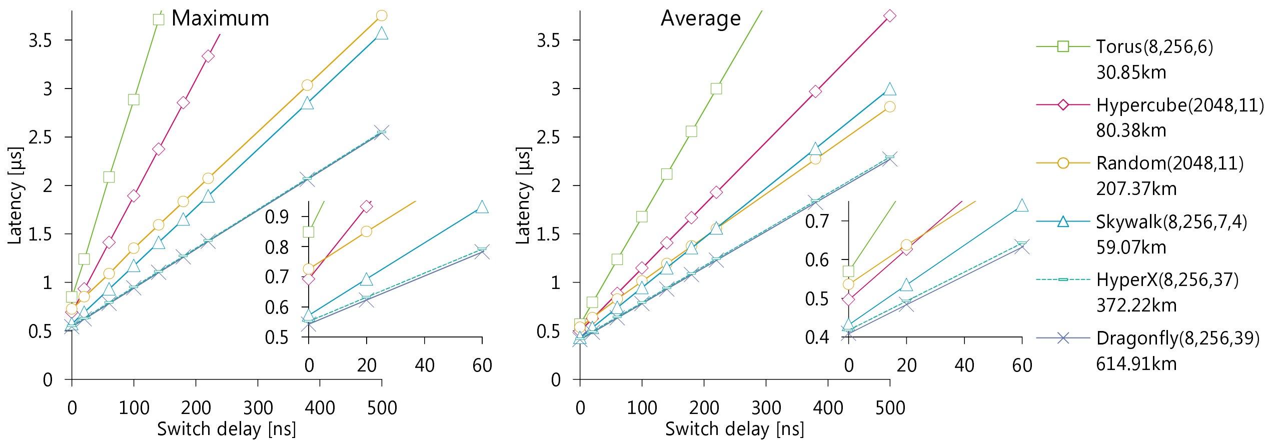


Fig. 5. Maximum and average latency vs. switch delay in 256-cabinet, 2,048-switch networks.

For each topology, the degree and the total cable length are indicated. At one extreme, Torus has degree 6 and a 31km total cable length and is thus expected to lead to low cost but to high latency. At the other extreme, Dragonfly has degree 39 and a 615km total cable length, and HyperX has degree 37 and a 372km total cable length. Both these topologies are thus expected to lead to low latency but at high cost. Hypercube, Random, and Skywalk all have degree 11, with total cable lengths of 80km, 207km, and 59km, respectively.

Results show that the maximum latency of Skywalk is always lower than that of Torus, Hypercube, and Random (as long as switch delays are lower than 500 ns). It is particularly notable that Skywalk achieves lower latency than Random with the same degree and a total cable length more than 3 times shorter. Dragonfly, resp. HyperX, leads to maximum latency lower than that of Skywalk but cable length more than 10 times, resp. 6 times, larger. At 60 ns switch delay, Skywalk leads to a maximum latency only 19% larger than that of Dragonfly/HyperX. In terms of average latency, Skywalk leads to better results than Torus and Hypercube across the board. It leads to lower average latency than Random for switch delays under 200 ns. As mentioned in Section I, 100-ns switches are already commercially available. Consequently, Skywalk is already an attractive topology with current switching technology when compared to a fully random topology. Like for the maximum latency results, the only topologies that outperform Skywalk in these results are Dragonfly and HyperX. But Skywalk leads to an average latency only 17% larger assuming a 60 ns switch delay, for more than 10-fold, resp. 6-fold, savings in total cable length when compared to Dragonfly, resp. HyperX.

We conclude that Skywalk is promising not only for future deployments with low-delay switches, but already for today's fastest switches. Nevertheless, to focus on switches in the near future, in all that follows we set the switch delay to 60 ns.

C. Degree

In this section we evaluate how the latencies of the topologies vary with the degree. The degree of a topology of switches is an important metric. Given a switch radix, the higher the degree the lower the number of ports that remain available to connect to compute hosts, thus limiting the scale of the system. Torus, Hypercube, Dragonfly, and HyperX all have fixed degree for a given number of switches, but for Skywalk and Random the degree can be chosen arbitrarily. We pick $d_\text{i} \in \{1, 4, 7\}$ and $d_\text{o} \in \{1, 2, 4, 6, 8, 12, 16, 20, 24, 28\}$ for Skywalk and $d \in \{4, 5, 6, 7, 9, 11, 15, 19, 23, 27, 31, 35, 39\}$ for Random. Figure 6 shows latency vs. degree, showing three curves for Skywalk depending on the d_i value, for a 2048-switch topology deployed in 256 cabinets.

These results show that Skywalk achieves lower latency than Torus and Hypercube of the same degree. For example, the maximum latency of Skywalk$(8, 256, 7, 4)$ is 34% lower than that of Hypercube$(2048, 11)$. When comparing Skywalk to Random, we see that Skywalk almost always leads to lower latency with a few exceptions. The exceptions correspond to cases in which d_o is low. It is a remarkable result that Skywalk outperforms Random, since previous work has shown that fully random topologies lead to drastic reductions in latency [5]. Skywalk uses less randomness, but by being cabinet-conscious it lowers inter-cabinet cable lengths and thus cable delays. Note that the three Skywalk curves have crossover points, which means that the latency decreases *non*-monotonically as the degree increases. We see that Dragonfly, not surprisingly, leads to the lowest maximum latency because it has a high degree of 39. Interestingly, due to the fact that Dragonfly$(8, 256, 39)$ is cabinet-conscious, it outperforms Random$(2048, 39)$. HyperX leads to results similar to that of Dragonfly, with a slightly lower degree (37 vs. 39) and a marginally higher maximum latency. The average latency results are similar and more to the advantage of Skywalk. In particular, Skywalk$(8, 256, 7, *)$ achieves latency similar to that of Dragonfly$(8, 256, 39)$ or HyperX$(8, 256, 37)$, but at much lower degrees. For instance, Skywalk$(8, 256, 7, 12)$ leads to average latency only 5.98% higher than that of Dragonfly$(8, 256, 39)$, for about half the degree (19 compared to 39). Similarly, Skywalk$(8, 256, 7, 12)$ leads to average latency only 4.27% higher than that of HyperX$(8, 256, 37)$ for a degree of 19 compared to 37.

D. Network Size

In this section we evaluate topology scalability, i.e., how latency increases with network size. Figure 7 shows latency vs. N, the total number of switches, assuming that there are $z =$

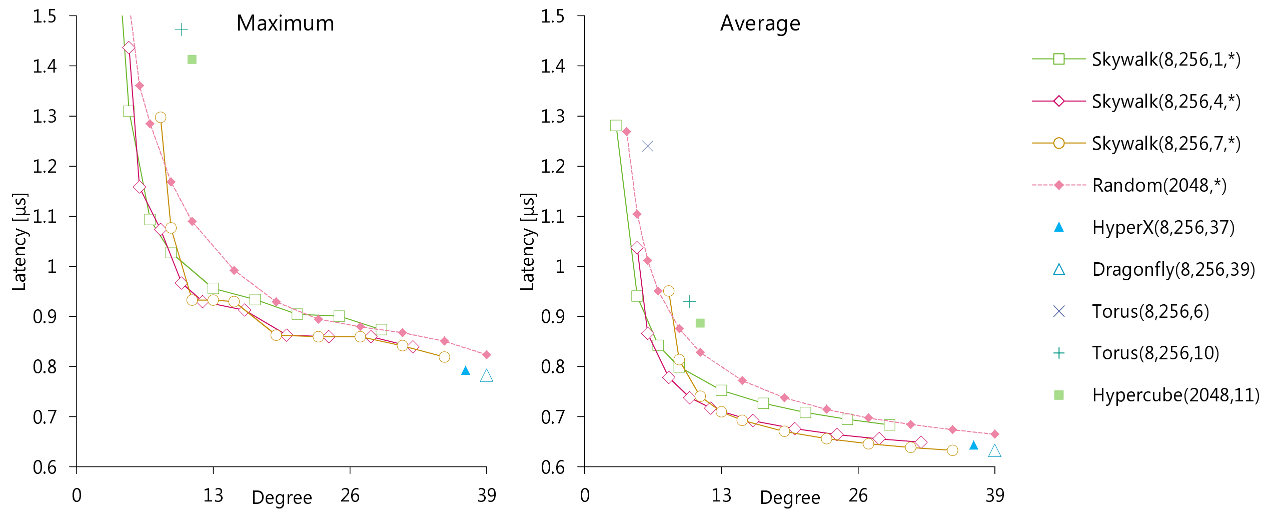


Fig. 6. Maximum and average latency vs. degree of topology in 8-switches/cabinet networks of 60-ns switches.

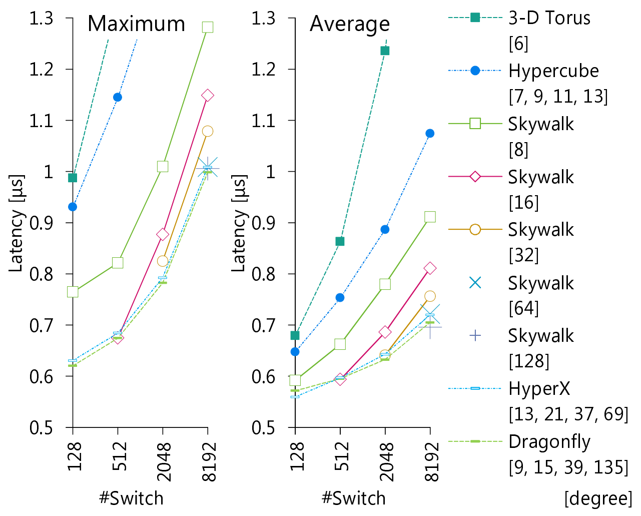


Fig. 7. Maximum and average latency vs. number of switches in 8-switches/cabinet networks of 60-ns switches.

8 switches per cabinet. Results are shown for Torus (degree 6), Hypercube (degree $\log_2 N$ between 7 and 13), Dragonfly (degree between 9 and 135), HyperX (degree between 13 and 69), and Skywalk (degrees 8, 16, 32, 64, and 128) instances.

For all topologies, latency increases as the network size increases. As expected, Hypercube and Torus are the least scalable topologies among the topologies we consider. Two observations can be made regarding Skywalk's scalability. The first observation is that the latency of Skywalk increases slowly. At degree 8 it has behavior comparable to that of Hypercube, even though Hypercube has a higher degree at large scale. At degree 16, when the network size is increased 4-fold from $N = 2048$ to $N = 8192$, the maximum, resp. average, latency increases a mere 31%, resp. 18%. For such a large network (assuming 16 compute nodes per switch for 32-port switches the network can comprise $8192 \times 16 = 131,072$ compute nodes), Skywalk with degree 16 achieves maximum latency of only 1,149 ns and average latency of only 811 ns. The second observation is that the payoff of increasing the degree to large values is small. While some improvements in latency can be achieved, results show that this improvement is likely not worth the increase in degree (and thus in switch radix, total cable length, and cost). We conclude that Skywalk can scale without requiring high degree.

The figure also includes results for Dragonfly and HyperX topologies, showing that these topologies are also scalable with moderate growth in latency as the network size increases. While the degree of Skywalk can be fixed regardless of the network size, the degrees of Dragonfly and HyperX increase with network size. Nevertheless, we can compare Skywalk to these topologies across network sizes by choosing the Skywalk instance with the largest degree that is lower than the degree of Dragonfly or HyperX. For instance, for a network with 2,048 switches Dragonfly has degree 39 and HyperX has degree 37. We find that Skywalk with degree 32 leads to maximum latency, resp. average latency, at most 5.4% larger, resp. 1.3% larger, than either topology. For a network with 8,192 switches Dragonfly has degree 135 and HyperX has degree 69. We find that Skywalk with degree 64 leads to maximum latency, resp. average latency, at most 1.0% larger, resp. 2.3% larger, than either topology. We conclude that Skywalk leads to results comparable to that of Dragonfly and HyperX but with lower degree. In the next section we show that Skywalk leads to significantly shorter cable lengths.

E. Total Cable Length

Cable length is one of the drivers of topology deployment cost. Large-scale low-delay networks can be "cable monsters" [13], and randomness can lead to even larger cable lengths [14]. In this section we assess the total cable length of Skywalk and compare it to that of other topologies.

Figure 8 shows latency vs. total cable length for all our topologies. As seen before, Torus and Hypercube lead to high latencies, which can be achieved by Skywalk and Random at lower cable lengths. The degrees of Skywalk and Random are the same as in Section III-C, each leading to a different cable length. These results show that, for a given latency target, the total cable length of Skywalk is drastically smaller than that

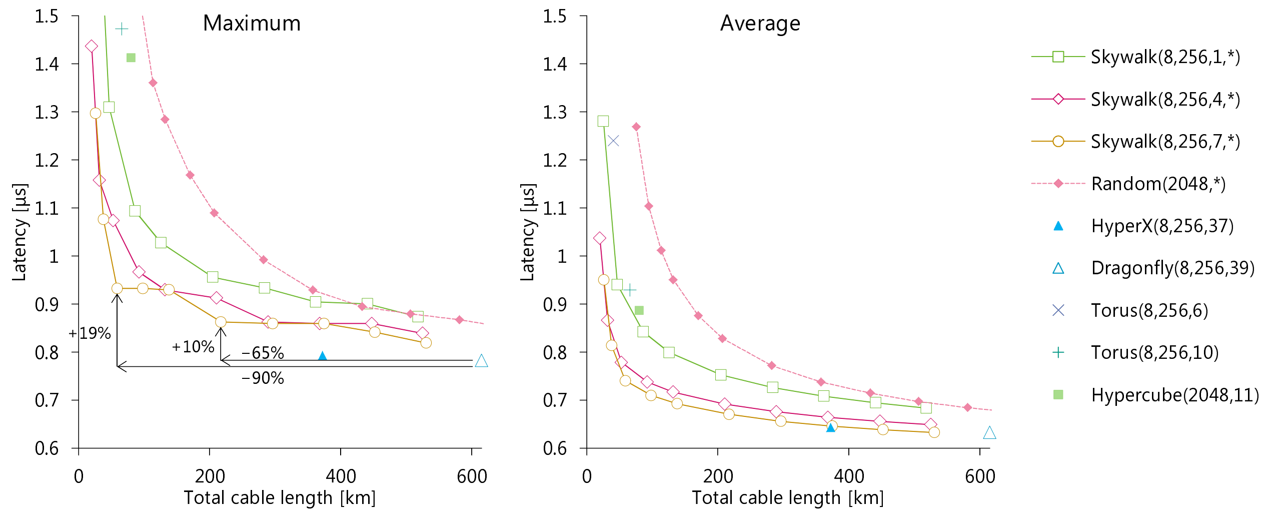


Fig. 8. Maximum and average latency vs. total cable length in 256-cabinet, 2,048-switch networks of 60-ns switches.

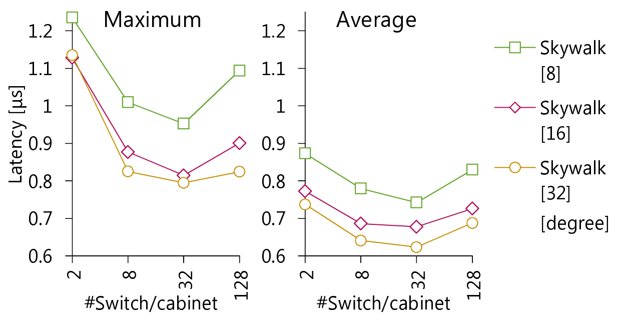


Fig. 9. Maximum and average latency vs. number of switches per cabinet in 2,048-switch networks of 60-ns switches.

of Random. Results also show that Skywalk compares well to Dragonfly and HyperX. For example, Skywalk$(8, 256, 7, 4)$ and Skywalk$(8, 256, 7, 12)$ require 90% and 65% lower cable length than Dragonfly$(8, 256, 39)$ while achieving maximum latency only 19% and 10% higher, respectively, as indicated in the left-hand side of the figure. The cable length savings of Skywalk when compared to HyperX are still large but not as large as when compared to Dragonfly. For instance, Skywalk$(8, 256, 7, 4)$ and Skywalk$(8, 256, 7, 12)$ require 84% and 42% lower cable length than HyperX$(8, 256, 37)$ while achieving maximum latency only 17.7% and 8.8% higher, respectively. These latency vs. cable length trade-offs are thus in favor of Skywalk given the cost associated with cable length.

An interesting observation is that the latency decreases monotonically as the cable length increases, without crossover points between the Skywalk curves. Given the non-monotonic observation made in Section III-C we find that with low switch delay it is the cable length, rather than the degree, that correlates directly with the latency. Results for the average latency on the right-hand side of the figure show that Skywalk can lead to average latency similar to that of Dragonfly at much lower cable lengths.

F. Cabinet Size

Since Skywalk accounts for the distribution of switches across cabinets, z, the number of switches per cabinet, impacts latency. Figure 9 shows the latency vs. z for networks with $N = 2048$ switches for Skywalk topologies of degrees 8, 16, and 32. In all cases, and for both the maximum and the average latency, the curves are concave with a minimum at $z = 32$, which corresponds to 64 cabinets. This is because for very low or very high z values Skywalk becomes similar to Random, which is outperformed by Skywalk as seen in previous sections. The results in Figure 9 show that there is an optimal cabinet size in terms of latency, which achieves the best balance between intra- and inter-cabinet connections. In practice, one should thus generate a Skywalk topology for each feasible cabinet size in order to pick the cabinet size that minimizes latency. Alternatively, one may wish to pick a cabinet size that leads to latency close to the minimum latency but that has other desirable features, such as low cabinet cost.

G. Limiting Inter-Cabinet Cabling

One approach to reduce cable length when creating a random topology is to bound the cable lengths of random links. This approach has been shown to lead to topologies that maintain the low latency of fully random topologies but with substantial savings in cable length [14]. In this section we evaluate this approach in the context of Skywalk. When picking the v_1 and v_2 vertices in the algorithm in Section II-C, we enforce that the distance between the cabinet hosting v_1 and the cabinet hosting v_2, measured using the Manhattan distance, is bounded above by $r \times l_{\max}$. Parameter r can be chosen arbitrarily between 0 and 1. $l_{\max}$ is the maximum possible cable length, i.e., the diagonal distance of the machine room floor measured using the Manhattan distance. A value of $r = 1$ corresponds to the original Skywalk, while a smaller value leads to reduced cable length.

Figure 10 shows latency vs. total cable length for Skywalk$(8, 256, 7, d_o)$, with $d_o \in \{4, 6, 8, 12, 16, 20, 24, 28\}$. For each of these topologies a curve with five data points is shown. Each of these data points corresponds to a value

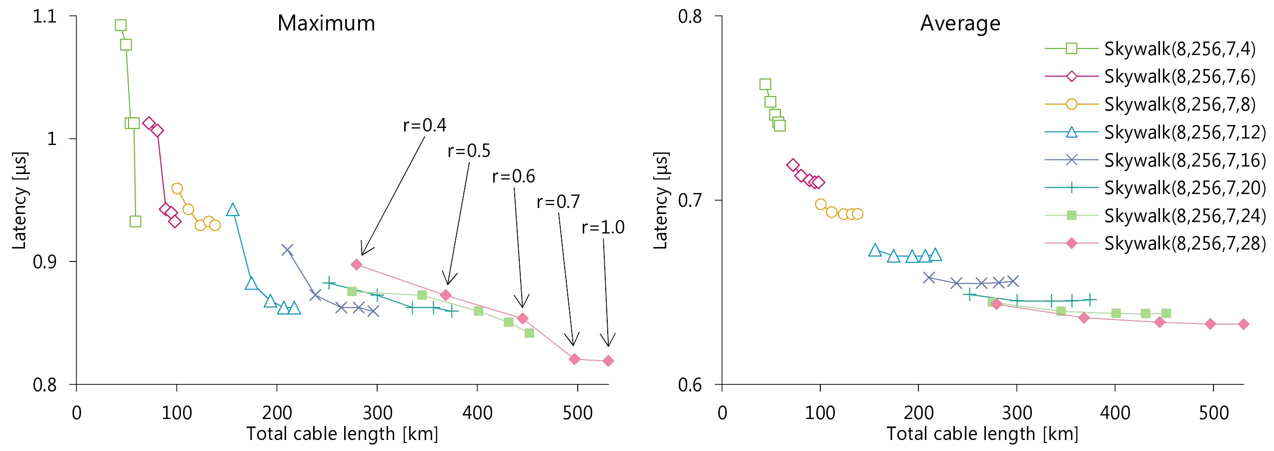


Fig. 10. Maximum and average latency vs. total cable length, depending on limitation on individual cable length, in 256-cabinet, 2,048-switch networks of 60-ns switches. $r = 1.0$ corresponds to no limitation on the length of individual cables.

of r, with $r \in \{0.4, 0.5, 0.6, 0.7, 1.0\}$. Too low a value of r may make constructing Skywalk unfeasible, but in these experiments $r \geq 0.4$ was sufficient for always generating the topology successfully. Given that the total cable length increases monotonically with r, the data points are for increasing r values from left to right on the horizontal axis.

Results show that the maximum latency decreases almost monotonically as r increases. Consequently, picking a particular r value achieves a particular trade-off between maximum latency and cable length. For low-degree Skywalk topologies, the savings in cable length achieved by picking $r < 1$ are marginal while the increases in maximum latency are substantial (i.e., the curves are close to being vertical), and a value $r = 1$ should be used. At higher degree the curves are closer to the anti-diagonal, meaning that different values of r achieve different possibly desirable trade-offs. The results for the average latency are similar but the curves are more flat, meaning that picking $r < 1$ can reduce cable length without large increases in average latency. In a few cases the average latency even increases slightly as r increases. Picking a moderate value of r, say, around 0.5, empirically leads to good trade-offs between average latency and total cable length.

We conclude that, depending on the degree of the topology, different values of r can lead to desirable trade-offs between cable length and latency. In all that follows we use $r = 1$, aiming to achieve the lowest latency with increased total cable length.

IV. Discrete-Event Simulation

A. Method

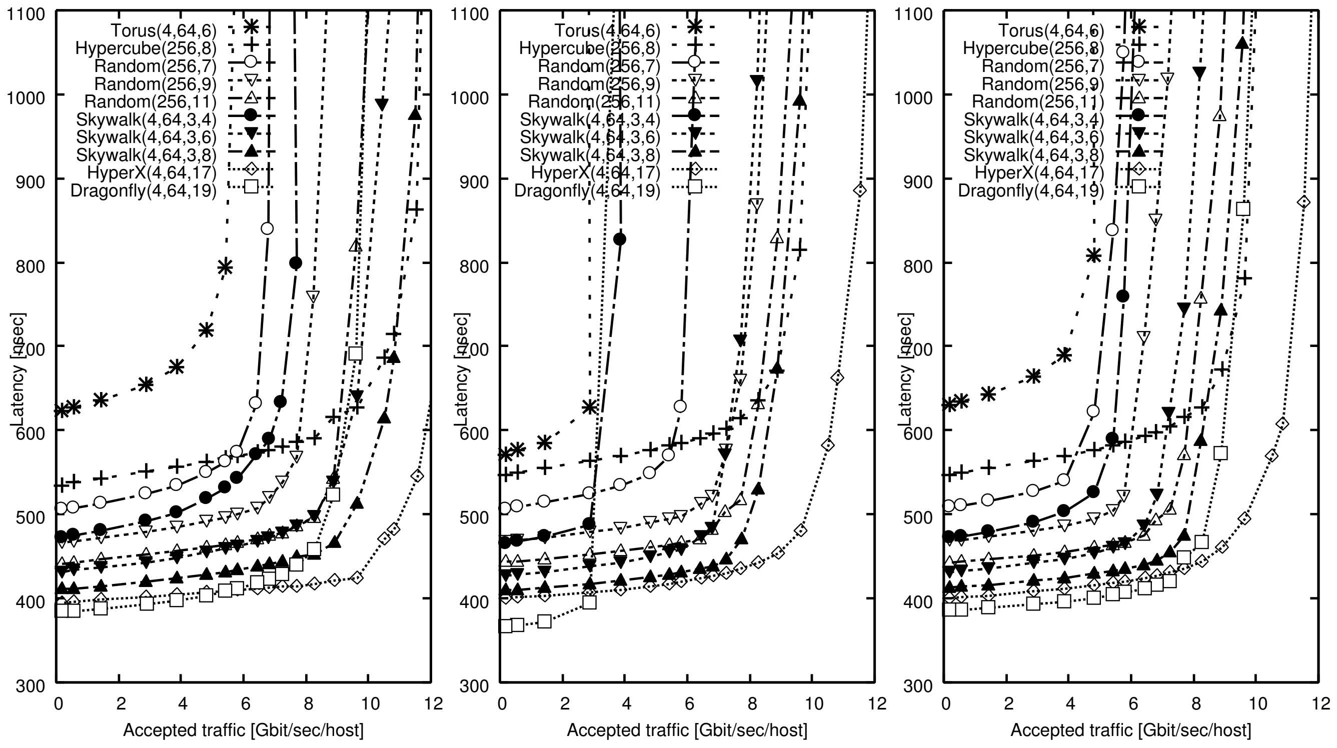
We use a cycle-accurate network simulator written in C++ [5]. Every simulated switch is configured to use virtual cut-through switching. A header flit transfer requires 60 ns, including routing, virtual-channel allocation, switch allocation, and flit transfer from an input channel to an output channel through a crossbar. Link delay is assumed to be 5 ns/m on an optical cable and the length of each cable is computed based on the cabinet layout. We use the FR scheme described in Section II-E. In this work we use the adaptive deadlock-free routing scheme described in [15] with four virtual channels, which is known to lead to well-distributed paths. To guarantee deadlock freedom, the routing restrictions stated in [15] are imposed when applying Dijkstra's path search. Interestingly, it has been reported that a random topology with the above routing performs better than a non-random topology with custom routing [5], and furthermore reducing end-to-end latency can be crucial in improving throughput. Simulation results show that our FR routing scheme sometimes improves not only latency but also throughput when compared to traditional min-hop custom routing.

We simulate three synthetic traffic patterns that determine source-and-destination pairs, namely *random uniform*, *bit-reversal* and *matrix-transpose*. These traffic patterns are commonly used for measuring the performance of large-scale interconnects [16]. 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) and each flit is set to 256 bits. We pick relatively small packet sizes since we wish to study the performance of latency-sensitive traffic that consists of small messages. Effective link bandwidth is set to 96 Gbps. In all the experiments in this section there are 8 hosts per switch, so that the maximum accepted traffic per host can be 12 Gbps.

Our results quantify two metrics: latency and throughput. The latency is the elapsed time (in ns) 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 Gbps) accepted by the network before saturation.

Because discrete event simulation is compute intensive, we simulate networks with "only" 256 switches. But the results obtained in simulation are consistent with the results in Section III, which are obtained with up to 8,129 switches and show stable trends as the number of switches increases.

B. Results



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

Fig. 11. Average latency vs. throughput. The same symbol is used for Random and Skywalk of same degree (one white, one black).

Figure 11 shows latency (averaged over all messages) vs. accepted traffic (the load that is injected into the network in Gbps/host) for all three traffic patterns, for networks with 256 switches located in 64 cabinets. The 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 point at which the latency curve "shoots up." Results are shown for Torus with degree 6, Hypercube with degree 8, Random with degree 7, 9, and 11, Skywalk with degree 7, 9, and 11, Dragonfly with degree 19, and HyperX with degree 17. Note that at degree 7, given the number of switches and of cabinets, due to our straight-first linking method Skywalk comprises only straight inter-cabinet links.

Latency results are consistent across the three traffic patterns and in line with the results in Section III. Torus and Hypercube lead to the highest latency. Skywalk achieves latency lower than that of Random for the same degree. Only Dragonfly, resp. HyperX, has latency lower than Skywalk but with a much higher degrees of 19, resp. 17, when compared to Skywalk's degree of at most 11 in these results.

In terms of throughput, as expected Torus leads to the worst results. Skywalk leads to similar or better results than Random for the same degree. The only exception is for the bit-reversal traffic. For this traffic, at low degree Skywalk leads to a throughput roughly halved when compared to that of Random. Skywalk with degree 11 approaches the throughput of Hypercube for all three traffic patterns. Dragonfly does not lead to high throughput in spite of its high degree, with particularly low throughput for the bit reversal traffic pattern. Only HyperX leads to better throughput than Skywalk for all three traffic patterns.

An important result is that Skywalk outperforms the fully random topology. In spite of its high degree, Dragonfly can lead to poor throughput while HyperX's high degree allows it to achieve the best throughput overall. We conclude that Skywalk makes it possible to achieve low latency and reasonably high throughput provided its degree is high enough, while not requiring a degree as high as that of Dragonfly or HyperX.

V. Related Work

In this work we design a topology of switches with physical layout considerations in mind. Several other layout-conscious topologies have been proposed. Among these, the Dragonfly topology [7] has received a fair amount of attention. In fact, Dragonfly is not a topology per se but a meta-topology. It distinguishes between groups of switches, but does not specify actual topologies for intra- and inter-group connections. In [7] the authors use a fully connected scheme that makes it possible to incur only one inter-group hop between any pair of switches. However, with low-delay switches, maintaining a single inter-group hop is no longer crucial since the cable length drives end-to-end delay rather than the number of hops. We have compared Skywalk to the fully-connected Dragonfly and the HyperX topologies. Both these topologies are designed explicitly to reduce cable length by accounting for the physical layout of cabinets. Other topologies have been proposed that distinguish between inter- and intra-cabinet communications. The Cube Connected Cycle (CCC) topology connects cabinets using a hypercube topology while using a ring topology for intra-cabinet connections [17], maintaining degree 3 even at large scale. Hypernet is a hierarchical topology that consists of subnets, which can be within cabinets, interconnected together by a fully connected topology [18]. Complete Connection of Torus Hypercube (CCT-Cube) is similar to hypernet, but has better diameter and degree properties [19]. We do not compare Skywalk to these topologies because they can support only a fixed number of nodes or because they strive to maintain very low degree, which is not necessary when using modern high-radix switches.

Our work relates to those works that have studied randomness in network topologies. Random shortcuts have been discussed in the graph theory literature. In particular, the low diameter properties of a ring with random chordal shortcuts are well known [20]. The effectiveness of random shortcuts to reduce diameter has been observed for real-world 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, such as the WS model by Watts and Strogatz [21]. Other small-world networks rely on a lattice structure plus random links that are generated by accounting for the distance along the lattice structure [22]. Using random topologies has been proposed in the context of datacenter interconnects, not only motivated by the low-diameter properties of such topologies, but also by fault-tolerance and expandability properties [4], [6]. In [5], [14] random topologies have been proposed as a way to reduce latency in HPC interconnects. The low hop count advantage of these random topologies does not necessarily translate to low latency for networks of low-delay switches. While Skywalk employs some randomness, it does so in a way that is cabinet-conscious, accounting for the fact that low cable length will be as important as or more important than switch delay in future networks. We have quantified the advantages of Skywalk over purely random topologies.

VI. Conclusion

In this work we have proposed Skywalk, a novel network topology for HPC interconnects. With current and upcoming low-delay switches, the traditional focus on reducing the number of switch hops along network paths in large-scale topologies of switches is no longer justified. The reason is that cable delays can exceed switch delays. Cable length is thus no longer only a driver of cost but also a driver of latency. In this context, the design goals of Skywalk are to achieve both low latency and low cable length. This is accomplished by using randomness and by distinguishing between intra-cabinet and inter-cabinet topologies. Routing is done by transferring packets along lowest-latency paths rather than along shortest-hop paths. Using both graph analysis and discrete-event simulation we have evaluated Skywalk and compared it to Torus, Hypercube, Random, fully-connected Dragonfly, and HyperX topologies. These topologies achieve various trade-offs between latency and cable length, and our results show that Skywalk leads to more desirable such trade-offs across a broad topology design space. Although Skywalk is designed for low latency, we find that it achieves relatively high throughput at moderate degree.

Our key finding is that Skywalk is promising because scalable, low-latency, moderate-degree, and with low cable length. Our results also suggest several guidelines for deploying Skywalk in practice, assuming a cable length budget. There should be balanced numbers of intra- and inter-cabinet links at each switch. The intra-switch topology should be dense, and possibly fully connected. The same recommendation holds for the inter-cabinet topologies along cabinet rows and cabinet columns on the machine room floor. If the cable length budget allows it, then sparse diagonal inter-cabinet links should be added to reduce end-to-end latency further.

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] S. Nishimura et al, "RHiNET-3/SW: an 80-Gbit/s high-speed network switch for distributed parallel computing," in *Hot Interconnects 9*, 2001, pp. 119–123.

[3] L. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. of the International Symposium on High Performance Computer Architecture (HPCA)*, 2001, pp. 255–266.

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

[5] 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.

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

[7] 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.

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

[9] 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.

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

[11] 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

[12] 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.

[13] J. Mudigonda, P. Yalagandula, and J. Mogul, "Taming the flying cable monster: A topology design and optimization framework for data-center networks," *USENIX ATC*, pp. 1–14, 2011. [Online]. Available: http://static.usenix.org/events/atc11/tech/final_files/Mudigonda.pdf

[14] M. Koibuchi, I. Fujiwara, H. Matsutani, and H. Casanova, "Layout-conscious Random Topologies for HPC Off-chip Interconnects," in *Proc. of the International Symposium on High Performance Computer Architecture(HPCA)*, 2013.

[15] 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.

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

[17] F. P. Preparata and J. Vuillemin, "The cube-connected cycles: a versatile network for parallel computation," *Communications of the ACM*, vol. 24, no. 5, pp. 300–309, 1981.

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

[19] T. Ishikawa, "CCT-Cube: A Highly Parallel Network Featuring Short Diameter and Few Links," *IEICE Transactions*, vol. J73-D-I, no. 6, pp. 559–602, 1990.

[20] 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.

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

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