Augmenting Low-latency HPC Network with Free-space Optical Links

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

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

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

***Abstract*—Various network topologies can be used for deploying High Performance Computing (HPC) clusters. The network topology, which connects switches in cabinets on a machine room floor, is typically defined once and for all at system deployment time. For a diverse application workload, there are downsides to having a single wired topology. In this work, we propose using free-space optics (FSO) in large-scale systems so that a diverse application workload can be better supported. A high-density layout of FSO terminals on top of the cabinets is determined that allows line-of-sight communication between arbitrary cabinet pairs. We first show that our proposal reduces both end-to-end network latency and total cable length when compared to a wired topology. We then demonstrate that the use of FSO links improves the embedding/partitioning capabilities of a wired topology. More specifically, we show that a recently proposed random low-latency topology can be augmented with a reasonable number of FSO links to support multiple k-ary n-cube and fat tree embedded topologies. Finally, we investigate power-aware on/off link regulation techniques and show how adding/reconfiguring FSO links leads to both performance and power efficiency improvements.**

I. Introduction

The common approach for building a large-scale High Performance Computing (HPC) cluster is to define a topology of switches, typically picking one of a few popular topologies such as low-radix k-ary n-cubes or high-radix fat trees. The selected topology is then deployed across several cabinets on a machine room floor, using inter- and intra-cabinet network cables to connect switches. One drawback of picking a particular network topology is that in a diverse application workload some applications may find this topology far from optimum. For instance, decades of research have gone into developing efficient mappings of parallel scientific applications onto k-ary n-cube topologies (e.g., numerical linear algebra kernels on 2-D or 3-D tori). For such applications, especially when large messages are exchanged, known mappings to known structured topologies lead to maximum performance. By contrast, parallel applications that have irregular and/or dynamically evolving communication patterns require low average network latencies across all switch pairs [1]. These applications can perform poorly on k-ary n-cube topologies due to long shortest path lengths between some switches, but they are well-suited to random topologies [2]. Conversely, traditional regular parallel applications would perform poorly on random topologies that do not have a k-ary n-cube structure. In the datacenter area, a conventional design is based on various tree-like topologies. However, emerging datacenter applications, such as graph analysis for social simulations, massive-scale MapReduce operations, or big-data sorts and searches, have characteristics similar to those of traditional HPC workloads [3]. And in fact, supercomputers designed for HPC workloads are ranked on Graph500 [4]. It is thus reasonable to expect that future datacenters will require different network topologies to support both legacy and emerging applications.

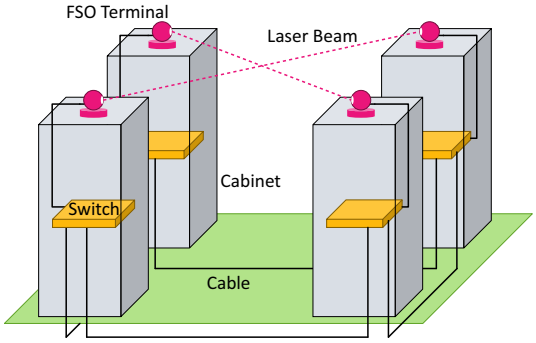

Fig. 1. Interconnection network that uses FSO links.

Given that different applications benefit from different topologies, defining the topology once and for all at system deployment time is problematic if a diverse application workload is to be supported. An approach to address this problem is to deploy multiple topologies that are each accessed via a separate network interface (e.g., IBM's BlueGene/L provides a torus topology and a tree topology separately). But the number of supported topologies remains limited and may not cater to all relevant applications. In this work, we take a radically different approach and propose to deploy network topologies in which some of the network link endpoints can be reconfigured arbitrarily. To enable reconfigurable endpoints, we use free-space optical (FSO) links. One or more FSO terminals, each consisting of a lens and a pointing control mechanism, are placed between the top of each cabinet and the ceiling of the machine room. These FSO terminals can be precisely re-

oriented so as to establish FSO links with various endpoints (see Figure 1 for a simple illustration). FSO technology is currently used in production for various applications, and we propose to use it as part of the network infrastructure in HPC clusters as proposed for datacenters in [5].

Beyond network reconfiguration to better match the communication patterns of parallel applications, FSO links can also be used for improved power management of the network infrastructure. Once the connection is established, an FSO link does not lead to an increase in power consumption when compared to a wired link [5]. For a given traffic pattern, rearranging FSO links thus makes it possible to create custom power-efficient topologies with large numbers of deactivated links. Typical power-efficient on/off network studies usually discuss the tradeoff between performance degradation and power reduction [6], [7]. This tradeoff occurs due to deactivation and activation time overheads, which can reach a few microseconds even for 10GBASE-T in IEEE 802.3az Energy Efficient Ethernet (EEE) [8], [9]. With FSO links, instead, it is possible to dynamically choose FSO endpoints and to deactivate wired links to improve both network performance and power efficiency.

Our main contributions in this work are:

- We propose several layouts of FSO terminals on top of cabinets that make it possible to establish line-of-sight FSO links between all or a large number of terminal pairs depending on the number of cabinets. (Section III)
- We quantify reductions in end-to-end network latency and total cable length when using FSO links. (Section IV)
- We show that a random network topology, which is well-suited to irregular parallel applications, can be augmented with FSO links to support multiple embedded k-ary n-cube and fat tree topologies. (Section V)
- We show how combining existing on/off link regulation with dynamically configured FSO links can improve both performance and power efficiency. (Section VI)
- We discuss how our FSO approach compares to the use of optical circuit switches (OCS) for the same purpose, how an embedded topology that uses FSO links can provide better application performance than its physical counterpart, and how our approach raises interesting questions for job scheduling in a production deployment. (Section VII)

Background information and related work are discussed in Section II. Section VIII concludes with a summary of our findings and perspectives on future work.

II. Background and Related Work

A. Network Topology

Current production HPC clusters are deployed using a handful of topologies, the most popular being low-radix tori and high-radix fat trees. For instance, in the Top500 list from November 2013 [10], six of the top ten systems use a torus and three use a fat tree. The remaining system uses the high-radix Dragonfly topology [11]. With exascale systems on the horizon, interest in such high-radix topologies has increased. For instance, today IBM proposes several high-radix topologies including high-dimensional tori (BlueGene/Q systems) and Dragonfly topologies (PERCS Power 775 systems). In this work we consider torus topologies, fat tree topologies, and high-radix random network topologies.

Communication-efficient mappings of application processes to compute nodes have been traditionally obtained by matching regular and deterministic communication patterns to a structured topology (e.g., tiling matrices onto a torus for parallel numerical linear algebra algorithms). By contrast, finding an efficient mapping for irregular parallel applications onto such topologies is challenging. Topology properties such as diameter, average shortest path length, or bisection bandwidth are crucial for these applications because any two processes may need to communicate. In this context, random network topologies are attractive because random graphs are known to achieve low hop counts [12], [13]. As a result, it has recently been proposed to use random network topologies for large-scale clusters [2], [14], [15].

In this work we use FSO for network reconfiguration. Network reconfiguration has also been proposed in [16]. Their approach relies on combining electrical and optical switches so as to support both packet-switching and circuit-switching. The circuit-switching network can be reconfigured dynamically to support bulk data transfers and network paths can be optimized to match particular traffic patterns. By contrast, FSO communication makes it possible to reconfigure connections between electrical switches directly and reconfiguration operations are free from cable geometry constraints.

B. Topology Embedding

Topology embedding is needed when running a parallel application that is implemented for a particular logical topology but is executed on a different physical topology. The general graph embedding problem consists in finding a mapping $\mathcal{F}$ from the vertices of a graph G to the vertices of a (larger) graph H along with a routing scheme, in our case shortest-path, so that certain metrics are optimized. The two main metrics considered in the literature are edge *dilation* and edge *congestion*. Given an edge in G between two vertices v and w, its dilation is the length of the shortest path between $\mathcal{F}(v)$ and $\mathcal{F}(w)$ in number of hops. Given an edge e in H, its *congestion* is the number of paths between $\mathcal{F}(v)$ and $\mathcal{F}(w)$ that go through e, over all (v, w) vertex pairs in G. The objective is to minimize maximum dilation and/or to minimize maximum congestion. The optimal value for dilation and congestion is 1. Finding an embedding with dilation 1 is NP-complete in the general case (since, e.g., the maximum clique problem is NP-complete).

The embedding of grids/tori into other graphs has been studied extensively [17]–[22], with a large fraction of that literature devoted to designing optimal or guaranteed algorithms for embedding grids and tori into hypercubes. In Section V we study embeddings in a random graph. More specifically, given an embedding with low maximum dilation and/or congestion, we use FSO links to augment the physical topology so that the embedding becomes optimal (with maximum dilation and congestion both equal to 1).

C. Network Power Consumption

Reducing the power consumption of interconnection networks is crucial since the network can represent a significant fraction of the total power consumption of a system (e.g., over 40% for the first-generation Earth Simulator). An approach for reducing network power consumption is to use DVFS (Dynamic Voltage and Frequency Scaling) for switches [23]. In addition, some commercial Ethernet switches reduce their power consumption by slowing down the link speed [24]

An orthogonal approach, termed "on/off link regulation," consists in deactivating network links that are not used and reactivating them later. Since deactivating/activating a link has a time overhead, it has been shown in various case studies that the times during which links are deactivated must be large enough to reach power consumption break-even points. For instance, when there is no traffic on a cable, Energy Efficient Ethernet (EEE) puts its NIC/switch PHY in Low Power Idle (LPI) mode. 10GBASE-T EEE has link deactivation/activation overheads on the order of a few microseconds [25]. It is shown in [9] that these overheads degrade the performance of latency-sensitive HPC benchmarks. A proposed solution therein is to allow for packet arrivals during the link activation and de-activation periods. In [26], the total system power budget (or cap) is maintained by dynamically shifting some power consumption between the compute nodes and an on/off interconnection network. In addition to the above *dynamic* approach, we also study *static* on/off link regulation in which network links are deactivated before an application is executed and remain deactivated throughout its execution. Network topology and routing strongly impact the effectiveness of the static approach because they determine the amount of traffic at each link. When using the static on/off link regulation, the activated network topology is a subset of the full topology. Some of the aforementioned previous works show that there is then a tradeoff between network performance and network power consumption. However, if some external links are added to the activated network, then it is possible to achieve both higher performance and lower network power consumption. This is what we propose to achieve with the use of FSO links.

D. Radio Communication Technology

Reconfiguring link endpoints, as proposed in this work, requires wireless communication. One option is 60-GHz band radio communication, such as IEEE802.11ad [27], which has been considered in the context of datacenters. It has been shown that offloading a portion of the network traffic to the radio network can improve the overall performance of datacenter applications [27]–[29]. The bandwidth of a 60-GHz radio link is up to several Gbps. Even though MIMO (Multiple Input Multiple Output) transfers have been proposed to improve the aggregate bandwidth [30], the resulting bandwidth does not reach that of current HPC systems, which can be in the 40 Gbps range. Moreover, 60-GHz radiowaves spread and interfere with each other. This prevents radio transceivers from being densely installed in a machine room.

E. FSO Communication Technology

Free-Space Optical (FSO) technology uses wider bandwidth and higher frequency [31], [32] than radio communication, with several hundred THz band for the carrier signal. The use of FSO technology in datacenter was recently proposed by Hamedazimi et al. [5]. They prove the feasibility of 10-Gbps datacenter-scale FSO communications using standard SFP+ tranceivers, multi-mode fibers, lenses, and mirrors. The lens transparently couples the fiber to/from a collimated laser beam; no additional opto-electric conversion is needed for FSO. An acceptable tradeoff between signal power density and misalignment tolerance can be achieved. They propose two possible optomechanical designs for an FSO terminal device to be steerable: (i) an array of mirrors that can be electrically switched between mirror (reflection) mode and glass (transparent) mode; and (ii) a Galvo mirror, which can be electrically rotated up to, e.g., $\pm 20^\circ$. Possible pairs of communicating FSO terminals are constrained by (i) the number of switchable mirrors or (ii) the angle of rotation of the Galvo mirror.

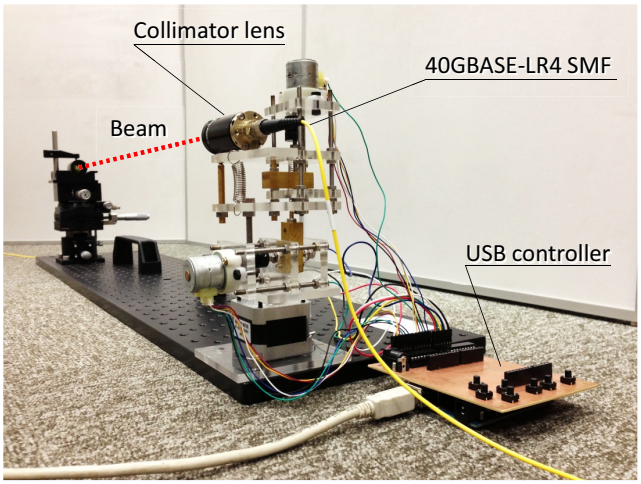

Fig. 2. Our FSO terminal prototype.

FSO communication technology is traditionally developed for long-range outdoor use, such as satellite, airplane, or inter-building communications. In this context, Arimoto et al. have developed a state-of-the-art FSO terminal device [33], [34]. One of its key features is a mutual beacon tracking system that acquires target terminal direction in 0.1° field-of-view (FOV) and maintain stable link within 1.0° FOV. This outstanding feature accomplishes full 360° steerability of the FSO terminal in combination with a commercial pan-tilt camera mechanical unit, e.g., FLIR's PTU-D46, which can steer terminal direction with a 0.013° resolution at a 300°/second steering speed. However, because it is intended for outdoor use, Arimoto's device is bulky and heavy-duty, and would not be necessarily practical for indoor use in a machine room.

To verify the potential of combining the ideas in [5] and [33], [34] we have developed a simple prototype FSO terminal with the goal of achieving 360°-steerable FSO communication in an HPC cluster. This prototype is shown in Figure 2. Built with off-the-shelf motors and gears, it is 360° steerable within 0.003° error, which translates into a 1.6-mm positioning error at a 30-m distance, which is well within the 6-mm tolerance of an FSO link [5]. Its footprint is 50 mm across, and its height can be lowered with a little design effort. This design

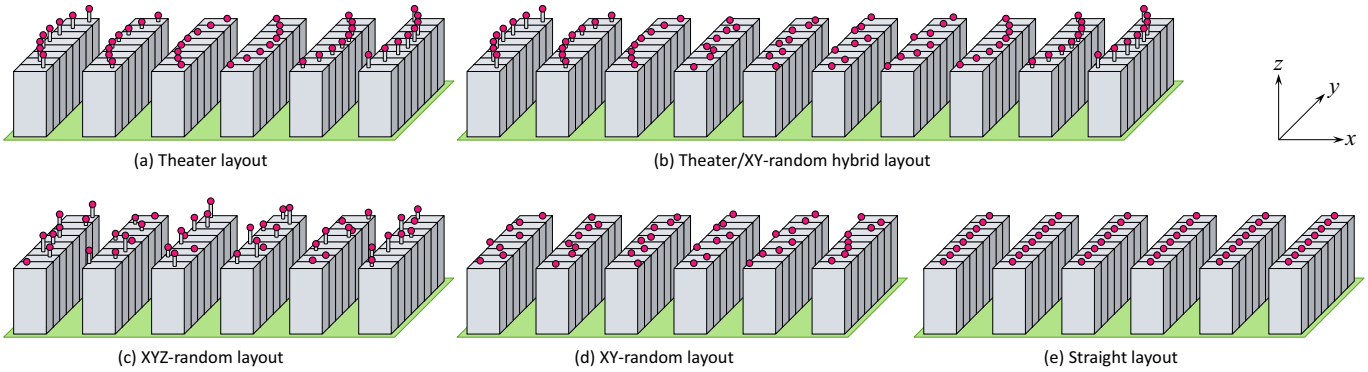

Fig. 3. Various direct layouts of FSO terminals.

thus largely alleviates the constraints in [5] regarding the number of possible pairs of communicating FSO terminals. Consequently, in the rest of this paper we assume that each FSO terminal can communicate directly with a large fraction of the deployed FSO terminals, and that the design is small enough to accommodate dense indoor layouts. Note that, in some cases, our proposed approach for topology embedding and network power management could be applied to the more constrained design in [5].

In both the work by Hamedazimi et al. [5] and ours, an FSO terminal is a purely passive device that consumes no power (except when steering). An FSO terminal is just a lens in an optical sense; it transforms a laser light in an optical cable to a laser beam in the air, and vice versa. The same routing and flow control mechanisms as Ethernet work with the FSO links, because an FSO link is bi-directional (full duplex) thanks to an optical circulator. Possible sources of laser beam attenuation include atmospheric turbulence, dust/gas absorption, and crossed beam interference. We have conducted an experiment using Arimoto's devices (which have more complicated optical circuit than our prototype) and confirmed that the optical power loss over a 30-m indoor FSO link is well within SFP/XFP/QSFP standards. Since most optical transceivers emit a constant optical power regardless of the link loss, we conclude that an FSO link does not affect the power consumption in the physical layer.

The prior work [5] focuses on the feasibility of FSO terminal devices for indoor use, because there are no custom/commercial FSO terminal designs available for datacenters. In this study we thus assume that FSO devices will be feasible in datacenters and supercomputers. Our main interests are "how much benefit will FSO network bring to supercomputers and high-end datacenters in terms of topology optimization, system partitioning and on/off link regulation?" We feel that both our work and that in [5] are needed for obtaining a feasible and efficient network architecture using FSO links on HPC clusters.

III. Layout of FSO Terminals in a Machine Room

We propose to outfit cabinets in a machine room with FSO terminals so that lenses can be reoriented to establish FSO links between cabinet pairs. In this section, we discuss several FSO terminal layout designs. We assume a standard 2-D grid layout of N cabinets in a machine room so that our approach can be applied to existing systems. A cabinet is 0.6 m wide and 2.1 m deep including space for an aisle, following the recommendations in [35]. Headroom, i.e., the distance between the top of a cabinet and the ceiling, is 1.2 m. Each cabinet has one FSO terminal, placed on top of the cabinet or possibly over an aisle. An FSO terminal, which consists of a lens and actuators, is packaged in a sphere of diameter R. The lens can be oriented in an arbitrary direction. A laser beam is emitted/received at the center of the package. For simplicity we ignore the thickness of the laser beam.

A. Direct Layout

Ideally there should be a direct line of sight between each FSO terminal. For every pair of FSO terminals to have direct line-of-sight, they must be placed so that no laser beam between any two terminals is interrupted by other terminals. An intuitive solution to this problem is a "theater" layout, as depicted in Figure 3(a) for 6 cabinet rows and 8 cabinets per row. Let us call the x-axis, resp. y-axis, the axis across, resp. along, cabinet rows. The z-axis is the vertical axis. In this layout, each terminal is placed in the center of a cabinet along the y direction but at various points along the x direction. The layout is computed incrementally as follows. First, place four terminals at the lowest possible height on top of the four center cabinets. Two of these cabinets belong to one cabinet row and the other two to another cabinet row. For each of these two rows, we add terminals to the other cabinets in the row so as to form an arc by varying their x-axis positions. See the two center cabinet rows with FSO terminals depicted in Figure 3(a). The curvature radius of this arc should be as large as possible so as to accommodate as many cabinets/terminals as possible. However it must be small enough to allow all terminals to see each other given the cabinet and terminal sizes. The layout is then built incrementally by using a similar technique for the two adjacent cabinet rows, but raising the terminals along the z-axis so as to allow direct lines of sight with all previously placed terminals. In this way, there is a direct line of sight between any pair of terminals.

One drawback of the above layout is that the number of terminals is limited by the headroom and the cabinet depth (size along the x-axis). Table I shows the maximum number of FSO terminals that can be used in a theater layout assuming a headroom of 1.2 m and a cabinet depth of 2.1 m, for various

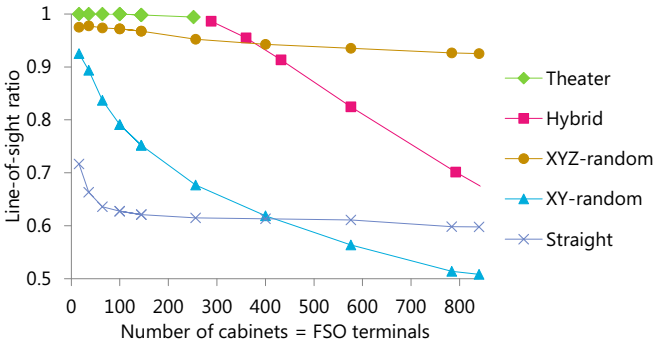

Fig. 4. Line-of-sight ratio vs. the number of cabinets. Each cabinet has 1 FSO terminal.

values of the terminal diameter R. For instance, assuming $R = 50$ mm, the theater layout supports up to $18 \times 14 = 252$ terminals on 252 cabinets. Another layout approach is needed to push system scale to larger numbers of cabinets.

The theater layout ensures that 100% of terminal pairs have direct lines of sight. We can instead come up with "incomplete" layouts that offer no such guarantee but hopefully yield many direct lines of sight in practice. The easiest approach is to use randomness and we consider three options: (i) The theater/XY-random hybrid layout in which terminals on the center cabinet rows are placed randomly directly on top of cabinets ($z = 0$) and the theater layout is used only for the maximum number of outside cabinet rows given a headroom limitation (Figure 3(b)); (ii) the XYZ-random layout in which terminals are placed randomly along all three dimensions (Figure 3(c)); and (iii) the XY-random layout in which terminals are placed directly on top of the cabinets ($z = 0$) but randomly along the x and y dimensions (Figure 3(d)). For the XYZ-random layout, we assume that terminals are placed on vertical cylindrical rods that have a diameter of 10 mm (and can obstruct the lines of sight between other terminals).

For each of these layouts we use a ray tracer [36] to determine whether a pair of terminals has a direct line of sight. We then calculate the line-of-sight ratio (LSR) as $\frac{2L}{N(N-1)}$, where L is the number of terminal pairs with a direct line of sight and N is the number of terminals. Figure 4 shows the LSR for the proposed layouts, including the theater layout and a baseline Straight layout (Figure 3(e)) for comparison. In these results, we assume $R = 50$ mm. Results show that 100% LSR is achieved at $N = 252$ by using the theater layout. The XYZ-random layout achieves 92.5% LSR at $N = 840$, which is close to the number of cabinets of the K-computer [37]. Although LSR might be improved using metaheuristics or other optimization techniques (which we leave as future work), we conclude that the direct layouts proposed in this section should suffice for many systems in practice.

TABLE I. MAXIMUM NUMBER OF TERMINALS THAT THE THEATER LAYOUT SUPPORTS, DEPENDING ON THE TERMINAL SIZE R.

R (mm)	40	50	60	70	80	100	160
#Rows	20	18	16	14	14	12	10
#Columns	16	14	12	12	10	10	8
#Terminals	320	252	192	168	140	120	80

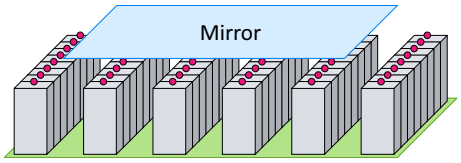

Fig. 5. Indirect layout using a mirror on a ceiling.

B. Indirect Layout

A way to sidestep the challenges involved in creating large direct layouts is to consider "indirect" layouts that rely on mirrors, as proposed in [5], [29]. We have seen in Section II-E that FSO can be used with mirrors without negative impact on performance. Figure 5 shows a straightforward indirect layout with a single planar mirror (placed on the ceiling of the machine room). In this layout there is an immediate indirect line of sight between any two terminals. While in principle there is no scale limitation, producing a very large mirror may not be practical. In fact, a promising approach would be to augment one of the incomplete layouts in the previous section with multiple smaller mirrors. There is a large design space for such indirect layouts, and it is reasonable to expect that an indirect layout is feasible with large numbers of FSO terminals and cabinets in a machine room. In the rest of this paper we assume such a layout, meaning that there is a direct/indirect line of sight between any two FSO terminals.

C. Multiple FSO Terminals per Cabinet

So far we have assumed only one FSO terminal per cabinet, but cabinets are large enough to accommodate multiple terminals. In terms of layout, a simple approach is to divide cabinet tops into same-size areas and place one terminal per area. In this case the maximum number of terminals supported by the theater layout is decreased, but the other layouts proposed in the previous sections are still applicable. In the rest of this paper we assume that up to 4 FSO terminals can be placed on top of each cabinet.

IV. LATENCY AND CABLE LENGTH

The use of FSO links in place of wired links reduces not only the cable length (which is desirable in terms of topology deployment cost [38]) but also the end-to-end network latency (which is desirable in terms of application performance). The reason is twofold: (i) the speed of light is faster in the air (3.3 ns/m) than in an optical cable (5.0 ns/m), and (ii) the FSO laser beam travels between two terminals along the Euclidean distance (i.e., $\sqrt{x^2 + y^2}$) while an optical cable is usually installed in a machine room along the Manhattan distance (i.e., $x + y$). In this section we quantify latency and cable lengths reductions due to the use of FSO links.

We consider 512 cabinets arranged in a 16×32 grid on a machine room floor. Each cabinet has $s = 2$ switches, $t = 0, 2, 4$ FSO terminals and $s \times m$ compute nodes, where each switch is connected to m compute nodes. Each FSO terminal can be connected to an arbitrary switch in its cabinet. We then consider four topologies of those 1,024 switches: 3-D torus ($2 \times 16 \times 32$) with degree $d = 6$, 5-D torus ($4 \times 4 \times 4 \times 4 \times 4$) with degree $d = 10$, fully random with degree $d = 6$, and

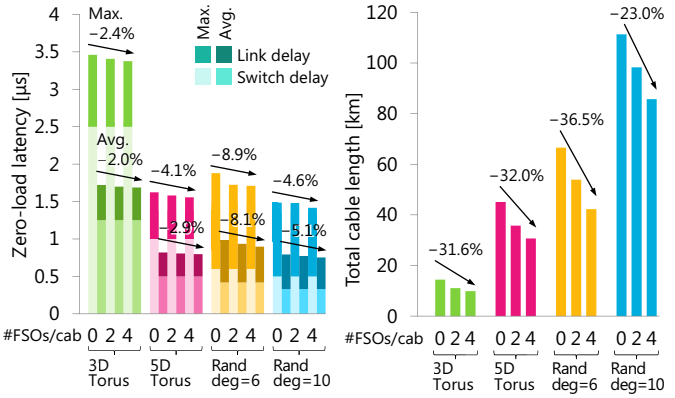

Fig. 6. Zero-load latency (left) and total cable length (right) vs. number of FSO links per cabinet.

fully random with degree $d = 10$. A fully random topology of degree d is generated by adding d random perfect matchings to a graph with no edges [2]. Other methods have been proposed to generate random topologies, such as random permutation and constrained shortcuts [39], but these methods lead to larger hop counts. Given a cabinet layout and a topology, we sort the edges in descending order of their Manhattan distance between the two cabinets, and assign each edge an FSO link (if possible) or a wired link (otherwise). The cable lengths are calculated as follows: an intra-cabinet cable is 2 m long, an inter-cabinet cable has 4 m wiring overhead (2 m in each cabinet) plus the Manhattan distance between the two cabinets, and each cabinet is 0.6 m wide and 2.1 m deep including space for the aisle [35]. The end-to-end zero-load network latency is calculated by graph analysis as follows: a switch delay is 100 ns, an FSO link delay is 3.3 ns/m, a wired link delay is 5.0 ns/m, and packets are routed along shortest-hop paths. For the sake of simplicity, we assume the Straight layout and 100% LSR as described in Section III.

Figure 6 (left) shows the zero-load latency vs. the number of FSO terminals on each cabinet. As expected, FSO links reduce end-to-end latency. For instance, the average end-to-end latency is reduced by 8.1% in a fully random topology of degree 6 when 4 FSO terminals are placed on each cabinet. In theory, the maximum possible reduction is 27.6% when an FSO link replaces a wired link (if it exists) connecting two cabinets on the opposite corner of the machine room.

Figure 6 (right) shows the total cable length vs. the number of FSO terminals on each cabinet. As expected, using FSO links reduces cable length, especially for those topologies with many long links, such as the random topologies and the high-dimensional torus topology. For instance, the total cable length is reduced by 36.5% when 4 FSO terminals are placed on each cabinet in a fully random topology of degree 6.

In this section, we have *replaced* some wired links with FSO links so as to compare the same topology built with/without FSOs. In the following three sections, we try to *augment* a network with FSOs, where wired links are untouched and FSO links are appended to an existing topology.

V. Topology Embedding with FSO Links

Parallel applications exhibit various communication patterns. A communication pattern can be defined as a graph in which the vertices are the processes and an edge links two processes if they communicate directly during the application execution. In many cases, application developers strive to map communication patterns to known logical topologies. For instance, it is typical to implement parallel numerical linear algebra algorithms assuming that processes are arranged in a logical torus topology. If the physical network topology of the parallel platform is also a torus, then the application can usually achieve high performance. If the physical topology is not a torus, then one must try to embed the logical torus into the physical topology with the goal of minimizing maximum dilation and congestion (see Section II-B). In this work, we consider topologies of switches rather than topologies of compute nodes, meaning that both the logical and physical topologies are graphs in which vertices are switches. Furthermore, we only consider injective embeddings so that different logical switches are mapped to different physical switches.

Part of the motivation for this work is the need to support both regular applications that can run efficiently on a structured topology (e.g., torus, hypercube, fat tree) and irregular applications that would benefit from running on a random topology. As in the previous section, we consider a topology of (electrical) switches placed in cabinets arranged in a 2-D grid on a machine room floor. Each cabinet contains s switches and is equipped with t FSO terminals. Each switch is connected to m compute nodes and each FSO terminal can be connected to an arbitrary switch in its cabinet. The switches are connected together via a fully random topology of degree d, as described in Section IV. This topology design thus assumes high-radix switches with $d + m + t$ ports.

The random topology has low (maximum and average) path lengths and is thus well-suited to irregular parallel applications [2]. However, for regular applications, it is necessary to embed the application's logical topology into the random physical topology. If an embedding with dilation 1 (and thus congestion 1 since the embedding is injective) is found, then the application can be executed without performance penalty when compared to an execution on the application's preferred physical topology. We consider the embedding of tori and of fat trees into the above random topology. We cannot reuse the embedding algorithms cited in Section II-B because they are designed for particular non-random physical topologies. Instead, we rely on non-guaranteed heuristics and meta-heuristics. Few embeddings with dilation 1 can be identified with these methods, but few such embeddings may exist in the first place. Note that the heuristics can be applied to any physical topologies, such as Dragonfly.

We propose to use FSO links to reconfigure/augment part of the physical topology so as to reduce an embedding's dilation, saying that an embedding is "valid" only if it has dilation 1. An embedding with higher dilation would still be useful to the application, but would degrade performance. Our objective is to determine to which extent FSO technology can enhance the embedding capabilities of a random physical topology, i.e., increase the number of valid embeddings that can be found.

A. Embedding Tori into the Random Topology

We consider a random topology with 1,024 switches, with $d = 8, 10, \ldots, 40$ and with $s = 2$ switches per cabinet. We consider that each cabinet can host $t = 0, 1, 2, 4$ FSO terminals. Experiments show that the t/s ratio determines the number of possible embeddings. Consequently, we opt to keep the number of switches constant and vary the number of FSO terminals. To embed a torus in the random topology, we first attempt to find an embedding with the lowest possible number of embedded edges with dilation strictly greater than one. We then attempt to "repair" these embedded edges using FSO links.

Finding a torus embedding with the lowest number of embedded edges with non-1 dilation is NP-complete. Since Genetic Algorithms (GAs) have been used successfully for graph embedding problems [40], this is the approach we take in this section. We have also attempted various global/local random searches and Simulated Annealing, but with significantly less success than with a GA. We encode candidate embeddings as individuals in the GA's population as follows. All individuals are same-length vectors. The i-th value in a vector is the index of the physical switch to which the i-th switch of the torus is mapped, using the canonical "by row" ordering to number the switches of the torus. The fitness of an individual is the number of embedded edges with dilation strictly greater than one, with a low fitness being preferable. No value is repeated in a vector since we consider injective embeddings. We use a crossover operator that produces two new individuals from two parents by concatenating a prefix of one parent with a suffix of the other parent, removing duplicate values if necessary, and possibly completing the vector with random values so as to generate individuals with the correct length. We define a mutation operator that replaces one value by a random value that is not already in the vector. We use a population size of 1,000, a crossover probability of 0.1, and a mutation probability of 0.25. We execute the GA for 500 generations or until no improvement in fitness is seen for 50 consecutive generations. At each generation we select individuals using tournaments of size 3. We execute the GA 10 times until an optimal embedding is computed, returning the best embedding out of these 10 trials otherwise. We then attempt to use FSO links to make the embedding valid, if necessary. After we find an embedding we (i) mark the physical switches it uses as unusable in future embeddings; (ii) remove the physical edges it uses; and (iii) remove the FSO terminals it uses. In this manner, we attempt to embed as many consecutive tori of a given size into our random physical topology until no further valid embedding can be found.

Figure 7 shows the *coverage* of the platform as a percentage, i.e., the fraction of the 1,024 switches that are used by embedded tori, vs. d. The curves have jagged shapes because we use a heuristic to optimize a function with many local minima. We show results for tori with 8 (4×2) switches assuming 1, 2, or 4 FSO terminals per cabinet, and 16 (4×4) switches assuming 2 or 4 FSO terminals per cabinet. Results with no FSO terminals are not shown because our heuristic never finds a valid embedding. This does not mean that valid embeddings do not exist, but simply that they are "hard" to find, at least using a genetic algorithm approach. Similarly we

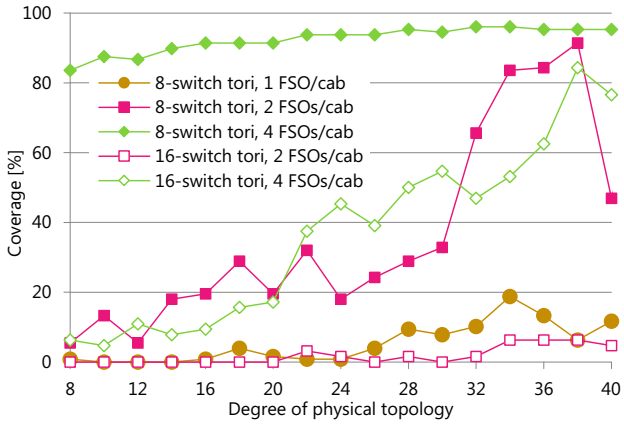

Fig. 7. Coverage of the 1,024-switch physical topology augmented with 1, 2, or 4 FSO terminals per cabinet when embedding 8- or 16-switch 2-D tori, vs. d.

do not show results when embedding 16-switch tori with 1 FSO terminal per cabinet as no valid embedding is found by our heuristic. As expected, decreasing the torus size makes the embedding problem less difficult. With still only 1 FSO terminal per cabinet, our heuristic finds valid embeddings of 8-switch tori covering more than 10% of the platform at high degree. Many more embeddings can be found if 2 or 4 FSO terminals are available at each cabinet. With 4 FSO terminals per cabinet, we find that it is always possible to cover more than 84% and up to 96% of the entire platform with 8-switch tori. With 16-switch tori, one can cover up to 84% of the platform but only at very high degree. Nevertheless, one can cover more than 50% of the platform provided $d \geq 28$. With only 2 FSO terminals, one can still achieve high coverage with 8-switch tori (up to 91% and more than 66% provided $d \geq 32$). However, the coverage becomes low with 16-switch tori, always below 7%.

At first glance, the size of our embedded tori may seem low. However, each switch may connect to tens of multi-core compute nodes. For example, assuming a switch that connects to as few as 16 6-core compute nodes, an 8-switch torus would contain 128 nodes for a total of 768 cores. It turns out that such a torus corresponds to the majority of jobs executed on production HPC systems. For instance, Xing et al. have analyzed the workload of the Kraken Cray XT5 system from January 2011 to June 2013 [41]. They show that 86.5% of the jobs use $\leq$ 512 cores, and only 0.6% use $>$ 8192 cores out of the entire 112,896 cores.

Overall, although a random topology has poor torus embedding capabilities, we conclude that one can find many useful embedded tori with a reasonable number of FSO links.

B. Embedding Fat Trees into the Random Topology

In this section, we consider the embedding of fat tree topologies into our random physical topology of degree d. It is not feasible to embed high-degree fat trees, e.g., Myrinet Clos [42], unless d is very large. Instead we consider lower-degree fat trees as described in [43]. We denote a fat tree of switches as Fattree(h, u, v). h is the number of levels in the tree, with level 1 being the leaf switches and level h being the highest

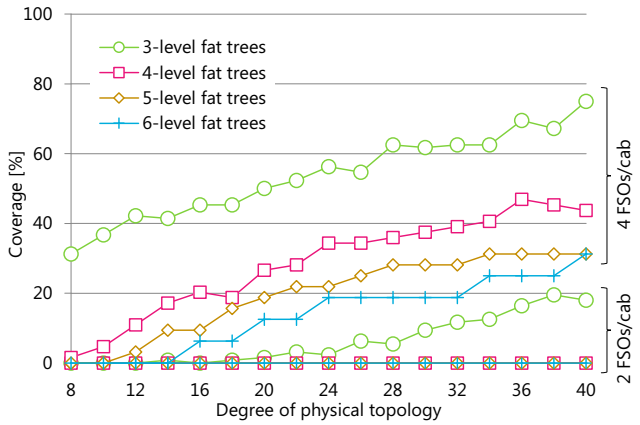

Fig. 8. Coverage of the 1,024-switch physical topology augmented with 2 or 4 FSO terminals per cabinet when embedding 3- to 6-level fat trees, vs. d.

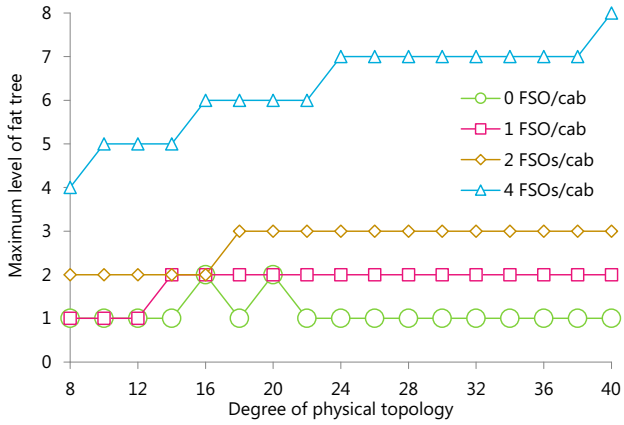

Fig. 9. Number of levels of the largest fat tree embedding in the 1,024-switch physical topology augmented with 0, 1, 2, or 4 FSO terminals per cabinet, vs. d.

level. u is the number of upward connections from each switch to different switches in the level above, and v is the number of downward connections from each switch to different switches in the level below. Fattree(h, u, v) with $h > 1$ has $2^{h-2}v$ leaf switches and $(2^{h-1} - 1)u$ non-leaf switches. Note that the highest level contains u switches. Fat trees are *indirect* topologies, meaning that the leaf switches are connected to compute nodes as well as to other switches, but the non-leaf switches are only connected to other switches.

We use a simple approach to embed a fat tree in our random physical topology. This approach is recursive since each switch at level $h+1$ connects to v switches in level h that are each the root of a h-level fat tree. We can thus embed fat trees bottom-up starting with 2-level fat trees. To find an embedding of Fattree$(2, u, v)$, we first find an embedding for a 2-level tree with 1 root and v leaves. We then find $u-1$ switches that each connects to the v leaves. Given two Fattree(h, u, v) fat tree embeddings, we then find u switches that each connects to v roots of the two fat trees. We thus need two heuristics: (H1) a heuristic to embed a 2-level tree with v leaves; and (H2) a heuristic to find a switch that connects to v given switches.

For H1, we consider all switches in the physical topology in a randomized order, and find the first switch that has v one-hop neighbors. If no such switch is found the heuristic fails. For H2, and for v given switches, we consider all other switches in a randomized order. We find the switch with the largest number of one-hop neighbors that are among the v given switches. We then use FSO links to connect the switch to all given switches if possible. If no such switch is found, then the heuristic fails. Otherwise, we pick the switch that uses the fewest number of FSO links. In case of a tie, we pick the switch that leaves the most cabinets with at least one unused FSO terminal.

After each embedding, we remove all used edges and FSO terminals used by the embedding. We also mark all leaf switches as unusable as leaves for other embeddings. Because the fat tree is an indirect topology, a switch that is used as a leaf in the embedding of one fat tree can still be used as a non-leaf switch in another embedding. Furthermore, a switch can be used as a non-leaf switch in multiple embeddings.

Figures 8 and 9 show results with the same 1,024-switch random topology with variable degree and $s = 2$ switches per cabinet as in the previous section. These results are all for Fattree$(h, 2, 4)$ embeddings, as computed with the above heuristics. Figure 8 shows the *coverage* of the platform as a percentage, i.e., the fraction of the 1,024 switches that are used *as leaf switches* in embedded fat trees, vs. d. Results are shown for fat trees with 3 to 6 levels (i.e., with 8 to 64 leaves), assuming either 2 or 4 FSO terminals per cabinets. Results are not shown for 0 or 1 FSO terminal per cabinets as no embeddings were found in these cases. With 2 FSO terminals per cabinet some embeddings of 3-level fat trees can be found, covering more than 11% of the platform for $d \geq 32$ up to 18% of the platform for $d = 40$. Expectedly, it is easier to cover the platform with fat trees with lower numbers of levels. With only 2 FSO terminals, embeddings of fat trees with more than 3 levels are only found occasionally by our heuristics. Results are vastly improved with 4 FSO terminals per cabinet. For 3-level fat trees, coverage is higher than 31% and up to 75%. Even for fat trees with 6 levels, i.e., with 64 leaves, more than 25% of the topology can be covered, provided $d \geq 34$.

Figure 9 shows the number of levels of the largest single fat tree that we are able to embed in our random topology. We see that only 1-level fat trees can be embedded with 0 FSO terminal per cabinet. With 1 FSO terminal per cabinet, some 2-level fat trees can be embedded when $d \geq 14$. With 2 FSO terminals per cabinet, fat trees with up to 3 levels (8 leaves) can be embedded in the topology with $d \geq 18$. With 4 FSO terminals per cabinet, fat trees with 5, 6, and 7 levels can be embedded for $d \geq 10$, $d \geq 16$, and $d \geq 24$, respectively. For high degree $d = 40$, 8-level fat trees (256 leaves) can be embedded, thus covering 1/4 of the physical topology.

We conclude that, although a random topology has poor fat tree embedding capabilities, the use of FSO links makes it possible to embed multiple and/or large fat trees. As expected, results for embedding fat trees with larger numbers of upward/downward links at each switch lead to lower coverage and lower maximum fat tree sizes. For instance, for a random topology with 1,024 switches, degree 30, 2 switches per cabinet, and 4 FSO terminals per cabinet, the largest embedded

fat tree with 4 upward and 4 downward link per switch identified by our heuristics has 6 levels, as opposed to 7 levels if only 2 upward links are used.

C. Combining Direct and Indirect Topology Embeddings

So far, we have embedded a collection of tori or a collection of fat trees into the physical topology. It is of course possible to embed both kinds of topologies simultaneously, thereby mixing direct and indirect topology embeddings. A single switch can thus serve as a leaf node of a fat tree or a node of a torus, and at the same time as a non-leaf node of one or more fat trees. After attempting to embed the maximum number of tori into the physical topology there remain sufficient unused FSO terminals so that fat tree embeddings can still be found. For instance, with the parameters in the previous two sections and assuming 4 FSO terminals per cabinet and a physical topology with degree 20, it is possible to embed 11 16-switch tori. We find that 7 additional fat trees can be embedded as well, each with 16 leaf switches. We conclude that a single physical platform with a random topology augmented with FSO links, as proposed in this work, can support a reasonably diverse workload that includes applications that map well to tori, applications that map well to fat trees, and applications that map to a random topology.

VI. Power Management with FSO Links

Existing on/off link regulation approaches and standard technologies such as Energy Efficient Ethernet (EEE) can be applied to regulate link activations in a way that matches an application's traffic pattern. Previous works have shown that on/off link regulation makes it possible to achieve a trade-off between the power consumption of the network and its performance. In this section, we study how existing dynamic and static on/off link regulation techniques can be enhanced with the use of FSO links to reconfigure link endpoints, in order to reduce both power consumption and hop counts.

A. Dynamic On/Off Link Regulation

Dynamic on/off link regulation, which is now supported commercially by EEE, consists in active monitoring of the traffic on each link and in automatic link deactivation if the link is unused for a given time window. In this context, we evaluate a simple FSO-based approach for minimizing the average number of packet hops for a given traffic pattern.

We enhanced a cycle-accurate network simulator called booksim written in C++ [44] with support for irregular custom topologies, topology-agnostic routing algorithms and simulation of dynamic on/off link regulation. Each simulated switch is configured to use virtual cut-through switching. A header flit transfer requires over 100 ns including 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 ns. We set the deactivation and activation overheads of each link to 2 μs since this delay is a few microseconds in EEE 10GBASE-T and would have smaller overhead for over-10Gbps links.

In all the experiments in this section the baseline topology is a 64-switch 3-D torus with dimension-order routing. We build custom topologies from this baseline by replacing p percent of the wired links with FSO links. These custom topologies could be partitioned subtopologies created for running a single job in a larger platform, e.g., embedded in a physical high-radix network topology as discussed in Section V. Custom topologies are generated using a simple greedy algorithm that replaces wired links by FSO links so as to minimize average hop counts over all routing paths for a given application traffic pattern. A topology with $p = 0$ denotes a 3-D torus, while a topology with $p > 0$ denotes a custom topology obtained by configuring FSO links optimized to the application's traffic pattern. The power consumption due to the links (including wired and FSO links) is constant regardless of p. This is because the lenses used in FSO terminals are passive components that do not consume power. In these topologies we use a topology-agnostic deadlock-free routing scheme and each simulated input-queuing router has four virtual channels. The input buffer size for each virtual channel is set to 128 flits. Network event traces obtained from executions of MPI (Message Passing Interface) implementations of the BT, CG, IS, LU, and SP NAS Parallel Benchmarks [45] in a cluster are provided as input to the network simulator.

Figure 10 shows simulation results in terms of (i) the relative latency, i.e., the elapsed time between the generation of a packet at a source and its delivery at a destination relative to that in the baseline 3-D torus (lower is better); and (ii) the fraction of the time when links are idle (higher is better, with 100% correspond to the case in which all links are turned off). We need to set the duration of the time window after which an unused link becomes deactivated. For each experimental scenario we tune this value empirically, through calibration experiments, so as to achieve the best latency and power trade-off. We can thus perform fair comparisons of experimental results across the experimental scenarios, assuming that a good time window is used in all cases. The question of how to determine the best time window for a given traffic pattern analytically, while interesting, is outside the scope of this work.

Results are consistent across benchmarks and show that using FSO links makes it possible to achieve lower latency and equivalent or longer link idle periods when compared to the baseline. For instance, when $p = 40$ percent of links are FSO links, for the IS benchmark the network latency is reduced by more than 20% and the link idle period is longer than in the baseline by a few percent. FSO links can effectively "shortcut" paths on the torus topology to match a communication pattern, thus leading to both lower latency and longer link idle time when compared to the baseline. We conclude that FSO links can be used in dynamic on/off link regulation to improve both performance and power efficiency.

B. Static On/Off Link Regulation

The overhead of on/off link regulation can degrade the performance of latency-sensitive HPC applications even with the low deactivation/activation overheads of EEE [9]. Besides, off-the-shelf network and InfiniBand products currently lead to large deactivation/activation overheads that can reach up to a few seconds [7]. As a result, the dynamic on/off link regulation approach in the previous section can be impractical. In this case, a simple workaround is to statically select the links that

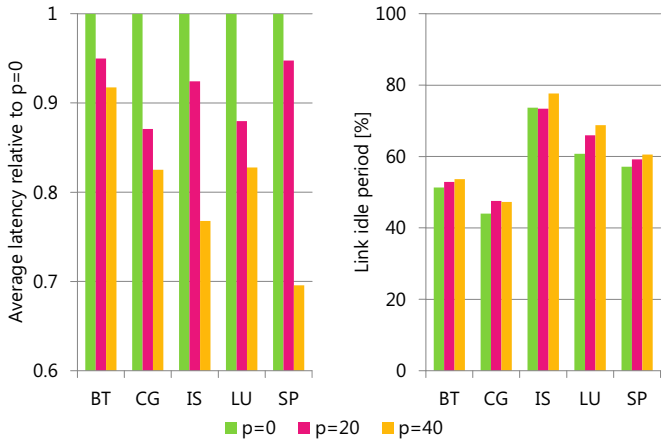

Fig. 10. Average latency relative to that of $p = 0$ (left) and total deactivated link period (right) for dynamic on/off networks for the 5 benchmarks.

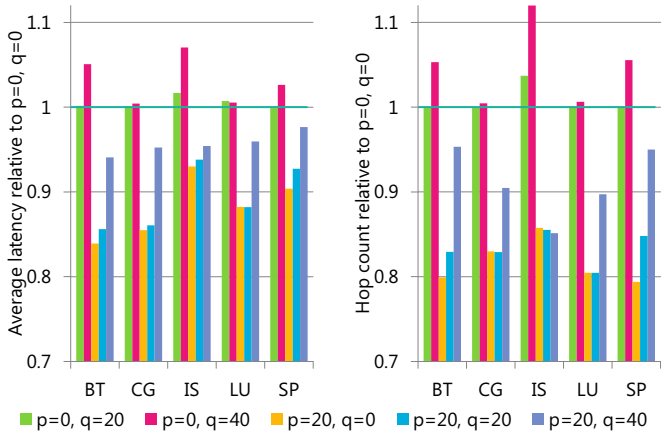

Fig. 11. Average latency (left) and average packet hop count (right) relative to those of $p = 0, q = 0$ for static on/off networks for the 5 benchmarks.

are to be deactivated just before an application runs [6], [7]. Assuming such a static on/off link regulation scheme, we study how FSO links can compensate the increase in hop counts due to deactivated links.

We use the same simulator, topology, and simulation parameters as in the previous section. The link power consumption does not depend on p, provided the number of deactivated links is the same. We use q to denote the percentage of links that are deactivated. Figure 11 shows the simulation results, in terms of latency (left) and hop counts (right), relative to the baseline with $p = 0, q = 0$. Without FSO link reconfiguration, the latency increases by up to 7.1% as the number of deactivated links becomes high for the BT, IS and SP benchmarks. By contrast, FSO link reconfiguration enables to improve both the number of deactivated links and network latency in all the traces. For example, even though 40% of links are deactivated, network latency is improved by 5.9% with FSO link reconfiguration. The latency trends are similar to that of the average hop count. Unlike dynamic on/off link regulation, paths for $q > 0$ can be different from that in the baseline 3-D torus. When some links are deactivated with $p = 0$, the hop counts increase. By contrast, if $p > 0$, the hop counts decrease. This is again because the reconfigured FSO links work effectively as shortcuts. Reconfiguring FSO links is thus also beneficial for static power-aware on/off link regulation.

VII. Discussion

A. Optical Circuit vs. FSO

Instead of using FSO links for topology reconfiguration one can use the same number of wired links connected to optical circuit switches (OCS). If OCSes are attached to electrical packet switches, then various optical circuit paths can be configured and established before running an application (these paths do not change during the application's execution). Neither switching delay nor packet contention occurs at the intermediate OCS on such a path. Conceptually, each optical circuit path works just as a link that connects two electrical switches.

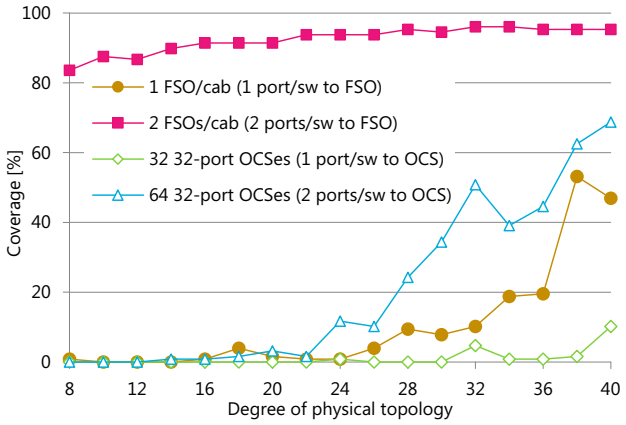

Fig. 12. Coverage of the 1,024-switch physical topology augmented with FSO or OCS when embedding 8-switch 2-D tori, vs. d.

In this section we compare the topology embedding capabilities of FSO and OCS. In both cases, we compute embeddings using the GA approach described in Section V-A. We assume a fully random topology of degree d of 1,024 electrical switches, as described in Section IV, but placed in 1,024 cabinets ($s = 1$ switch per cabinet). When using FSO, each cabinet has 1 or 2 FSO terminals connected to the switch inside that cabinet. When using OCS, each switch is connected to an OCS with 1 or 2 links. For example, when using 64 32-port OCSes, switches 0 through 15 are connected to OCS 1, switches 16 through 31 are connected to OCS 2, etc., each with 2 links.

Figure 12 shows the coverage of the platform when embedding 8-switch tori using FSO or OCS, vs. d. Comparing the case with 1 FSO terminal per cabinet and the case with 32 32-port OCSes, we see that using FSO leads to higher coverage than using OCS. This difference is even clearer for case with 2 FSO terminals per cabinet and the case with 64 32-port OCSes. While using the same number of extra ports, FSO links leads to coverage between 84% at $d = 8$ and 95% at $d = 40$, whereas OCSes leads to coverage 0% for $d < 24$, gradually increasing up to 69% at $d = 40$. The reason for these results is that

the FSO links have higher freedom of reconfiguration than the OCS links. In other words, an FSO link can be established between arbitrary pairs of terminals, while the endpoints of an OCS link are constrained by the number of OCS ports.

B. Exploiting Unused Links for Improved Embedded Topologies

Typically, the degree of an embedded topology is lower than that of the physical topology in which it is embedded. For instance, traditional parallel algorithms (e.g., numerical linear kernels) lead to low-radix traffic patterns (e.g., that map directly to 2-D or 3-D tori). This is why in Section V we have studied the embedding of k-ary n-cubes or fat trees. However, we note that an application running on an embedded (logical) topology can achieve a higher performance than if it were running on an equivalent (physical) topology. This is because unused physical links, i.e., links that do not participate in any embedding, can be used as extra shortcut links for the embedded topology, thereby reducing communication latency.

Consider the embedding of 4×4 tori into a 1,024-switch 36-degree random topology with 4 FSOs per cabinet, as done in Section V-A, which produces 44 embedded tori. For 41 of the tori there is at least 1 unused physical link that can be used as a shortcut. For more than half of the tori there are 3 or more such links, with 2 tori with 6 such links. Using these links as shortcuts leads to occasional improvements in diameter (from 4 hops to 3 hops for 4 of the 44 tori) and significant improvements in average shortest path length (by more than 5.0% for 34 of the 44 tori, and by up to 13.6%). We conclude that the abundant links in a high-radix physical topology can be utilized to improve low-radix embedded topologies.

C. Practical Scheduling Considerations

We have shown the potential of FSO technology for enhancing both the topology embedding and the network power management capabilities of wired topologies. Interesting questions remain regarding the production use of FSO technology in practice for HPC deployments. In production use, the topology is partitioned into subsets, each subset running a parallel application, or job, and jobs arrive to and depart from the system dynamically (e.g., managed by a batch scheduler). Since our topology embedding and power optimization techniques are job/traffic dependent, they must be applied to each subset of the topology dedicated to a job. Given currently running jobs and current network configurations, a scheduler component must perform network configuration for a newly arrived job, for topology embedding and/or power management. The best configuration, however, is likely not always achievable. For instance, the new job may be allocated to compute nodes that are located in cabinets whose FSO terminals are all currently in use, which may preclude perfect topology embedding (i.e., with dilation 1) or the most effective power management scheme. In terms of embedding, the maximum number of embeddable non-random topologies (e.g., tori or fat trees) may already have been reached, in which case the job may be running on a random sub-topology with degraded performance due to non-optimal dilations.

VIII. Conclusions

In this work, we have proposed augmenting topologies of switches, as used in large-scale HPC clusters, with FSO terminals so that a fraction of the topology's link endpoints can be reconfigured. FSO devices can achieve a wire-rate data transfer over tens of meters [5], and we have proposed effective layouts of FSO terminals in a machine room. An FSO link has the key advantage over a wired link that it reduces both the end-to-end network latency and the total cable length of the deployed topology. FSO links also enable link endpoint reconfiguration, which can be used for many purposes. In particular, we have shown how the use of FSO links can increase the topology embedding capabilities of a random physical topology. We have also shown how existing on/off link regulation techniques for network power management can benefit from FSO links so that both power efficiency and network performance are improved.

The selection of the best network topology for a parallel application, in terms of network performance and power consumption, is an open problem. We contend that the use of FSO links, as proposed in this work, makes a fundamental step toward answering this open problem because FSO link reconfiguration affords ad-hoc network topology tuning capabilities. A future direction for this work is to design a scheduler component that makes judicious network reconfiguration decisions for a newly arrived job based on the current workload on the platform (including advance reservations) and current FSO terminals availability, so as to achieve desirable performance and power consumption trade-offs.

Acknowledgments

We give a special thanks to Shinichi Ishida, Cyber-Koubou LLC, Japan, for making the FSO terminal prototype; and to Yoshinori Arimoto, National Institute of Information and Communications Technology, Japan, for experiments with his FSO terminal. This work was partially supported by JST CREST and JSPS KAKENHI Grant Numbers 25280018 and 25730068.

References

[1] M. Kulkarni, M. Burtscher, C. Cascaval, and K. Pingali, "Lonestar: A suite of parallel irregular programs," in *Proc. 2009 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 65–76.

[2] M. Koibuchi, H. Matsutani, H. Amano, D. F. Hsu, and H. Casanova, "A case for random shortcut topologies for HPC interconnects," in *Proc. 39th International Symposium on Computer Architecture (ISCA)*, pp. 177–188.

[3] "The workshop on Big Data and Extreme-scale Computing (BDEC)." [Online]. Available: http://www.exascale.org/bdec/

[4] "The Graph 500 List." [Online]. Available: http://www.graph500.org/

[5] N. Hamedazimi, Z. Qazi, H. Gupta, V. Sekar, S. R. Das, J. P. Longtin, H. Shah, and A. Tanwer, "FireFly: a reconfigurable wireless data center fabric using free-space optics," in *Proc. 2014 ACM conference on SIGCOMM*, pp. 319–330.

[6] V. Soteriou and L.-S. Peh, "Exploring the Design Space of Self-Regulating Power-Aware On/Off Interconnection Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 3, pp. 393–408, Mar. 2007.

[7] M. Koibuchi, T. Otsuka, and H. Amano, "An on/off link activation method for low-power ethernet in PC clusters," in *Proc. 2009 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*, pp. 1–11.

[8] "Media Access Control Parameters, Physical Layers, and Management Parameters for Energy-Efficient Ethernet," in *IEEE Standard for Information technology–Local and metropolitan area networks–Specific requirements–Part 3: CSMA/CD Access Method and Physical Layer Specifications*. IEEE Standard 802.3az, 2010.

[9] K. P. Saravanan, P. M. Carpenter, and A. Ramirez, "Power/performance evaluation of energy efficient Ethernet (EEE) for High Performance Computing," in *Proc. 2013 International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pp. 205–214.

[10] "TOP500 Supercomputer Sites." [Online]. Available: http://www.top500.org/

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

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

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

[14] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, "Jellyfish: networking data centers randomly," in *Proc. 9th USENIX conference on Networked Systems Design and Implementation (NSDI)*, p. 17.

[15] J.-Y. Shin, B. Wong, and E. G. Sirer, "Small-world datacenters," in *Proc. 2nd ACM Symposium on Cloud Computing*, pp. 1–13.

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

[17] Y. Saad and M. Schultz, "Topological properties of hypercubes," *IEEE Transactions on Computers*, vol. 37, no. 7, pp. 867–872, Jul. 1988.

[18] M. Y. Chan, "Embedding of Grids into Optimal Hypercubes," *SIAM Journal on Computing*, vol. 20, no. 5, pp. 834–864, Oct. 1991.

[19] M. Chan, F. Chin, C. Chu, and W. Mak, "Dilation-5 embedding of 3-dimensional grids into hypercubes," in *Proc. 5th IEEE Symposium on Parallel and Distributed Processing*, pp. 285–288.

[20] M. Röttger, U.-P. Schroeder, and W. Unger, "Embedding 3-dimensional grids into optimal hypercubes," in *Proc. 1st Canada-France conference on Parallel and distributed computing*, pp. 81–94.

[21] M. Röttger and U.-P. Schroeder, "Embedding 2-Dimensional Grids Into Optimal Hypercubes with Edge-Congestion 1 or 2," *Parallel Processing Letters*, vol. 08, no. 02, pp. 231–242, Jun. 1998.

[22] J. Ellis, S. Chow, and D. Manke, "Many to One Embeddings from Grids into Cylinders, Tori, and Hypercubes," *SIAM Journal on Computing*, vol. 32, no. 2, pp. 386–407, Jan. 2003.

[23] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proc. 9th International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 91–102.

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

[25] P. Reviriego, K. Christensen, J. Rabanillo, and J. A. Maestro, "An Initial Evaluation of Energy Efficient Ethernet," *IEEE Communications Letters*, vol. 15, no. 5, pp. 578–580, May 2011.

[26] J. Li, W. Huang, C. Lefurgy, L. Zhang, W. E. Denzel, R. R. Treumann, and K. Wang, "Power shifting in Thrifty Interconnection Network," in *Proc. 17th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 156–167.

[27] "Enhancements for Very High Throughput in the 60 GHz Band," in *IEEE Standard for Information technology–Telecommunications and information exchange between systems–Local and metropolitan area networks–Specific requirements-Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE Standard 802.11ad, 2012.

[28] D. Halperin, S. Kandula, J. Padhye, P. Bahl, and D. Wetherall, "Augmenting data center networks with multi-gigabit wireless links," in *Proc. 2011 ACM conference on SIGCOMM*, vol. 41, no. 4, p. 38.

[29] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, B. Y. Zhao, and H. Zheng, "Mirror mirror on the ceiling: flexible wireless links for data centers," in *Proc. 2012 ACM conference on SIGCOMM*, p. 443.

[30] Y. Katayama, T. Yamane, Y. Kohda, K. Takano, D. Nakano, and N. Ohba, "MIMO link design strategy for wireless data center applications," in *Proc. 2012 IEEE Wireless Communications and Networking Conference (WCNC)*, pp. 3302–3306.

[31] H. Henniger and O. Wilfert, "An Introduction to Free-space Optical Communications," *Radioengineering*, vol. 19, no. 2, pp. 203–212, 2010.

[32] Z. Ghassemlooy, H. Le Minh, S. Rajbhandari, J. Perez, and M. Ijaz, "Performance Analysis of Ethernet/Fast-Ethernet Free Space Optical Communications in a Controlled Weak Turbulence Condition," *Journal of Lightwave Technology*, vol. 30, no. 13, pp. 2188–2194, Jul. 2012.

[33] Y. Arimoto, "Near field laser transmission with bidirectional beacon tracking for Tbps class wireless communications," in *Proc. SPIE 7587, Free-Space Laser Communication Technologies XXII*, p. 758708.

[34] Y. Arimoto, H. Yoshida, and K. Kisara, "Wide field-of-view single-mode-fiber coupled laser communication terminal," in *Proc. SPIE 8610, Free-Space Laser Communication and Atmospheric Propagation XXV*, p. 861008.

[35] Hewlett-Packard, "Optimizing facility operation in high density data center environments," 2007. [Online]. Available: http://h10032.www1.hp.com/ctg/Manual/c00064724.pdf

[36] M. Pharr and G. Humphreys, *Physically Based Rendering, Second Edition: From Theory To Implementation*. Morgan Kaufmann, Jul. 2010.

[37] Y. Ajima, S. Sumimoto, and T. Shimizu, "Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers," *Computer*, vol. 42, no. 11, pp. 36–40, Nov. 2009.

[38] I. Fujiwara, M. Koibuchi, and H. Casanova, "Cabinet Layout Optimization of Supercomputer Topologies for Shorter Cable Length," in *Proc. 13th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT)*, pp. 227–232.

[39] M. Koibuchi, I. Fujiwara, H. Matsutani, and H. Casanova, "Layout-conscious random topologies for HPC off-chip interconnects," in *Proc. 19th International Symposium on High Performance Computer Architecture (HPCA)*, pp. 484–495.

[40] R. Chandrasekharam, V. Vinod, and S. Subramanian, "Genetic algorithm for embedding a complete graph in a hypercube with a VLSI application," *Microprocessing and Microprogramming*, vol. 40, no. 8, pp. 537–552, Oct. 1994.

[41] F. Xing and H. You, "Workload Aware Utilization Optimization for a Petaflop Supercomputer," in *Proc. 2014 Annual Conference on Extreme Science and Engineering Discovery Environment (XSEDE)*, pp. 1–8.

[42] Myricom, "Guide to Myrinet-2000 Switches and Switch Networks Myrinet M3-E32," 2001. [Online]. Available: http://www.myricom.com/scs/myrinet/m3switch/guide/myrinet-2000_switch_guide.pdf

[43] C. E. Leiserson, "Fat-trees: Universal networks for hardware-efficient supercomputing," *IEEE Transactions on Computers*, vol. C-34, no. 10, pp. 892–901, Oct. 1985.

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

[45] "NAS Parallel Benchmarks." [Online]. Available: http://www.nas.nasa.gov/publications/npb.html