Exascale Interconnect Topology Characterization and Parameter Exploration

Shang Li
Department of Electrical and Computer Engineering
University of Maryland
College Park, MD 20740
Email: shangli@umd.edu

Po-Chun Huang
Department of Electrical and Computer Engineering
University of Maryland
College Park, MD 20740
Email: hpcalex@umd.edu

Bruce Jacob
Department of Electrical and Computer Engineering
University of Maryland
College Park, MD 20740
Email: blj@umd.edu

***Abstract*—Exascale promotes numerous new challenges in supercomputer designs; Interconnect topology is one of them. Emerging interconnect topologies have been proposed but have not been thoroughly studied at large scale. In this study, we use a cycle-accurate simulator to conduct a design space exploration for both new and traditional interconnect topologies at the scale of 50,000 and 100,000 node networks. For each topology, we evaluate the performance with various network parameters (link latency and bandwidth), routing algorithms and multiple prevalent MPI workloads. Using these, we are able to characterize these topologies with regards to workloads, network parameters, and scalability to provide insights on the network system design for future Exascale systems.**

1. Introduction & Related Work

For decades, data movement has been a challenging factor for supercomputer systems and it has bounded their computation efficiency [1], [2]. Nowadays, the top supercomputers in the Top500 List [3] would only achieve a fraction of their designed peak performance on a benchmark with significant data movement, such as HPCG [2], [4]. For example, today's No. 1 supercomputer, Sunway TaihuLight, has a designed peak floating point performance of 125.4 Pflop/s and can achieve 93 Pflop/s on Linpack, which is 74% of peak performance. However, its HPCG benchmark performance is only 0.3% of peak [5], more than two orders of magnitude lower. Moving data around in a system more efficiently is clearly a significant aspect of improving performance in supercomputer systems and it will be a foreseeable challenge for Exascale systems.

To address this problem, numerous interconnect topology models were proposed in recent years [6], [7], [8]. In addition to interconnect topology, there are many additional factors that could influence the performance of data movement for a supercomputer system, including routing and flow control, interface technology, and physical link properties (latency, bandwidth). It is also worth noting that software stack and application behaviors can significantly affect the network's performance [9]. Therefore, these parameters need to be carefully chosen to meet the design requirements for an Exascale-class system. In order to do this, a thorough comparison across all the dimensions is needed to fully comprehend the characteristics of the topologies.

In this paper, we examine several promising interconnect topologies for Exascale systems. We are first to propose more efficient routing algorithms for the Fishnet topology [7]. We extend the Structural Simulation Toolkit (SST) [10], [11] to support the concerned topologies for accurate simulation and evaluation. We conduct cycle-accurate simulations and sweep through a design space of various network configurations under different types of MPI workloads. Based on the simulation results, we present a topology characterization and network parameter exploration for Exascale-system network design.

A summary of our simulated configurations and workloads can be found in Table 1, and we will describe these parameters in more detail later in the paper.

TABLE 1: Simulated Configurations and Workloads

System Size *	50,000 and 100,000
Topology	Dragonfly, Slimfly, Fat-tree (3 to 4 levels), Fishnet, Fishnet-Lite
Routing Algorithm †	Minimal, Valiant, Adaptive(UGAL)
Link Latency	10ns, 20ns, 50ns, 100ns, 200ns
Link Bandwidth	8GB/s, 16GB/s, 32GB/s, 48GB/s, 64GB/s
MPI Workloads	AllPingPong, AllReduce, Halo, Random
Total # configurations	> 3000

*Note that the system size here is approximate since most topologies cannot be configured to be these exact numbers.
† Not all topologies supports all of these routing algorithms.
e.g. We only simulated deterministic and adaptive routing for Fat-tree.

With the help of the large volume of data obtained from simulation, we are able to gain a comprehensive view of these topologies and it is therefore helpful for assessing their usage in Exascale. To conclude, we summarize our contributions in this paper as follows:

- We perform large scale, fine grained network simulations and design space explorations. We simulate the network size with up to 100,000 nodes and collect more than 3,000 data points. To our knowledge, it is by far the largest design space exploration at such large scale.
- We propose and evaluate adaptive routing algorithms tailored for Fishnet and Fishnet-lite topologies. Our

evaluation shows properly implemented routing can reduce the execution slowdown by 20x under heavy adversarial workloads.
- We show how the different network parameters influence the performance of each topology under different workloads. We observe that most topologies benefit from an increasing link bandwidth while they are less sensitive to longer link latency.
- We evaluate the interconnect topologies for their scaling efficiency and their capability to handle increasing workloads, which provides useful insights on the interconnect system design for Exascale.

The rest of the paper describes the background and related work, routing schemes, our experimental setup, and our experimental results and conclusions.

1.1. Related Work

Studying large scale interconnection networks has always been challenging. On the one hand, simulating a system with even thousands of nodes is difficult (let alone millions), because it can require significant resources and time. Therefore, carefully engineered tools are required to perform such simulations in an efficient way. Jiang et al. developed *Booksim*, a cycle accurate interconnection network simulator [12], but its single-threaded design makes it very hard to simulate large scale network. Carothers et al. designed and implemented the Rensselaer Optimistic Simulation System (ROSS) simulator [13] based on time warp technique and is highly efficient on parallel execution. Rodrigues et al. developed Structural Simulation Toolkit (SST) [10], a parallel discrete event simulation toolkit that aims to help design and evaluate supercomputer systems. Its modular design makes it easier to extend their models.

On the other hand, there are many factors that could influence the performance and cost of large scale interconnection networks and a variety of studies exist on characterizing such interconnection networks. Mubarak et al. has studied and simulated high dimension torus networks with up to 1 million nodes and showed that large scale simulations are critical for designing Exascale systems [14]. [15], [16], [17], [18] studied different aspects of Dragonfly networks and characterized Dragonfly as a highly scalable and efficient interconnect. [19], [20] evaluated diameter-2 topologies with various routing algorithms and traffic patterns. Li et al. introduced more scalable Fishnet and Fishnet-lite diameter-4 interconnect topology and performed a preliminary performance/cost analysis [7]. [6]

2. Topologies

In this section we introduce some background knowledge of interconnect topologies and their relevance to this study.

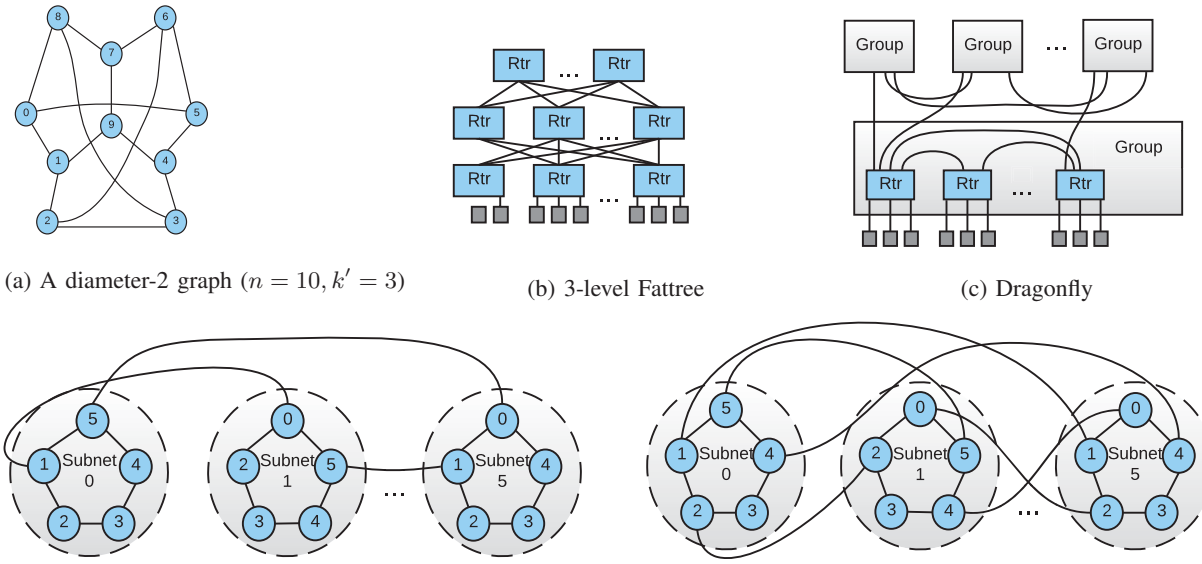
Fat-Tree, also referred as a folded Clos topology [21], [22], enables low latency and high bisection bandwidth using high-radix routers. The architecture of Fat-tree provides very high bisection bandwidth and is used in real world supercomputer systems [23]. The disadvantage of the fat-tree topology is its scalability: for example, a 2-level Fat-tree would require 100 ports per router to scale to 50,000 nodes. The port cost and its associated power cost would be prohibitively expensive for building an Exascale network within 2 levels. Increasing the number of levels of a Fat-tree will alleviate its scalability issues but would also introduce more end-to-end latencies. Given the network scale that we target (100k node), we focus on 3- and 4-level Fat-tree topologies, which could scale to 100k node within 100 ports per router.

Dragonfly [24] was proposed to address the scalability issues of previous topologies such as Fat-tree networks and flattened butterfly topologies [25]. The concept of a *virtual router* was introduced here as a means to reduce the port cost and achieve scalability. A virtual router is essentially a hierarchical design that groups routers and treats them as one, thereby reducing the number of expensive global links. There are different ways of setting up a Dragonfly topology so that one can obtain high efficiency (by maximizing system size with minimal number of ports per router) or high performance (by adding more global links per router). In Figure 2 we show how different setups of Dragonfly can result in very different scalability ("Dragonfly (max)" in Figure 2 refers to the high efficiency setup while "Dragonfly (min)" refers to high performance setup). Because our goal is to explore extremely large scale networks, we follow the efficiency setup recommended by [24], which allows us to scale up to 100k nodes with only 51 ports per router.

Slimfly: Bao et al. first proposed using Hoffman-Singleton graph (a 50-node diameter-2 graph) as an interconnection topology for high density servers [26]. Besta et al further proposed Slimfly topology to construct much larger scale networks based on Moore graphs [8]. The low network diameter provides potentially lower latency and the Moore graph architecture provides high path diversity. An example of 10 node, diameter-2 Moore graph is shown in Figure 1a. In Slimfly, assuming the number of links of a nodes that connects to other nodes in the network is k' and number of endpoints attached to a node is p, then the number of endpoints in a system will be approximately $k'^2 \times p$. To avoid oversubscribing a router, a setup of $p \leq k'/2$ is recommended by [8]. Following this setup, we are able to construct a network with $k' = 79$ and $p = 18$ to get a 101,124 node network within 100 ports per router. We do not want to oversubscribe the router for it would not be a fair comparison against other topologies.

Fishnet is a recently proposed topology [7] that builds on top of diameter-2 graphs and yields a diameter-4 topology. Here we briefly summarize how to construct a Fishnet based on a diameter-2 graph.

First, assuming we are using a diameter-2 Moore graph as a basic building unit and there are n nodes in the Moore graph with k' links from each node to other nodes. We make $n + 1$ copies of the diameter-2 graph and number them as subnet 0 to n, and within each subnet, we number their n nodes from 0 to n, skipping their subnet number. e.g. for



(a) A diameter-2 graph ($n = 10, k' = 3$)

(b) 3-level Fattree

(c) Dragonfly

(d) Fishnet-lite, 1 link between subnets ($n = 5, k' = 2$)

(e) Fishnet, k' links between subnets ($n = 5, k' = 2$)

Figure 1: Illustration of topologies studied, for simplicity not all nodes and links are presented in the graph.

subnet 1, the nodes are numbered as 0, 2, ... n. We use the notation of (i, j) to refer subnet i, node j.

Then, we make connections between the subnets with the following rule: To connect subnet i to subnet j, we find the k' adjacent nodes of (i, j), and k' adjacent nodes of (j, i), and connects these k' pair of nodes. Finally we follow this procedure to connect all the subnets, and we will have a $N = n(n + 1)$ node Fishnet, with $n + 1$ subnets, and k' links between any two subnets.

In this way, if any node in subnet i tries to talk to any node in subnet j, it takes at most one hop to get to one of the adjacent nodes of (i, j), then it takes one more hop to get to subnet j, and finally from there at most 2 hops to get to any node in subnet j.

A lite version of fishnet can also be constructed at an even lower cost. The difference from Fishnet is when making connections between subnets, we only connect (i, j) to (j, i) directly, producing a $N = n(n + 1)$ node network with only 1 global link between any two subnets. It has a diameter of 5 but can be constructed at a much lower cost.

Both Fishnet and Fishnet-lite show great scalability, the can scale up to 1 million node within 80 ports, making them promising candidates for Exascale network.

A brief graphic illustration of the aforementioned topologies can be found in Figure 1. The scalability comparison of different topologies (router radix versus system size) can be found in Figure 2.

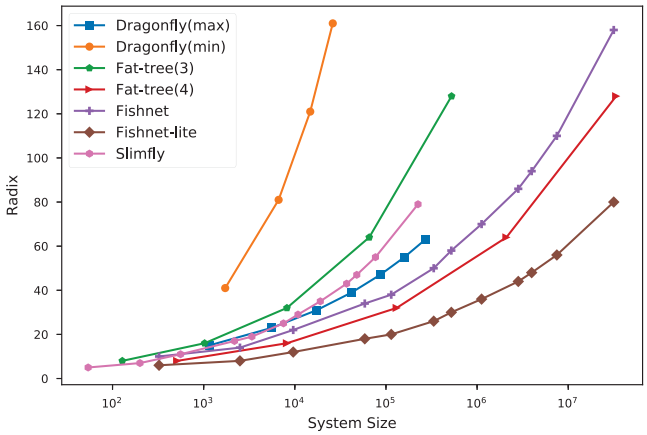


Figure 2: Scalability of different topologies studied in this work

3. Routing Algorithms

Routing algorithms play an important role in fully exploring the potentials of an interconnect topology. Previous studies have shown that applying proper routing algorithms could result in significant latency and throughput improvements [15], [19], [27] on various topologies. In this section, we will explore routing algorithms for recently proposed topologies as well as review options for traditional topologies.

3.1. Routing for Fishnet Family

Fishnet and Fishnet-lite interconnects were proposed in [7] but only minimal routing was discussed in their study. It is necessary to further study more efficient routing schemes for such topologies to fully explore their potentials. Especially for Fishnet-lite, where only one global link is used to connect between subnets, using minimal routing could

congest the global link easily and thus leads to performance degradation.

To address this problem, we propose Valiant random routing and adaptive routing algorithms tailored for the architecture of Fishnet and Fishnet-Lite.

3.1.1. Valiant Random Routing Algorithm (VAL). The Valiant Random Routing algorithm [28] is used in multiple interconnect topologies to alleviate adversarial traffics [8], [24]. The idea of Valiant routing is to randomly select a intermediate router (other than the source and destination router) and route the packet through 2 shortest paths between the source to intermediate and between the intermediate to destination, respectively. By doing so, additional end-to-end distance is added into the path, but it may also avoid a congested link and balance the load on more links, and lower the overall latency.

Applying Valiant routing to Fishnet family will be similar to Dragonfly topology, where global links between groups/subnets are more likely to be congested when the traffic pattern requires more communication between groups/subnets. In Dragonfly, a random intermediate group is used to reroute the packet to the target group.

Similarly, for Fishnet-lite, we randomly select a intermediate subnet and route the packet to the intermediate subnet and then to its destination subnet. This could increase the worst case hop count from 5 to 8, but would also increase the path diversity, with $k' - 1$ more paths, and reduce the minimal route link load to $1/k'$ of its previous value.

For Fishnet, we apply a similar technique, which will result in a hop count from 4 to 6 in worst case, but expand the path diversity from k' to k'^2.

3.1.2. Adaptive Routing. The idea of adaptive routing is to make routing decisions based on route informations. One of the widely used adaptive routing schemes, Universal Globally-Adaptive Load-balanced Algorithm (UGAL), [29] takes VAL generated routes and compares them with the minimal route, selecting the one with less congestion. The key here is to decide which route has less congestion. Ideally, if we have global information of all routes and all routers, it would be easy to make such decisions. However, in real systems, it is impractical to have such information across the system. Therefore, a more reasonable approach is to only use local information, (UGAL-Local, or UGAL-L) such as examining the depth/usage of local output buffers.

UGAL-L works well on topologies such as Dragonfly and Slimfly. However, its effectiveness will be limited in Fishnet since the local information obtained from output buffer cannot reflect route congestion accurately when the next link is congested and the information is not propagated back. This also happens in Dragonfly networks, as described in [30].

An example of how traditional UGAL-L might not work well for Fishnet-lite is shown in Figure 3. Imagine the worst case scenario where all k' nodes in the source subnet want to send packets to the destination subnet. Because in Fishnet-lite there is only one global link connecting between the source and destination subnet, all the minimal routes will pass through that router (router 0 in fig. 3). If all the output buffers towards that router have very low usage, traditional adaptive routing will prefer the minimal path over Valiant path. This would keep happening until the intermediate buffers are almost full, and by then there will be a lot of packets in the buffers waiting for the global link to be available, jamming routers on both side of the global link.

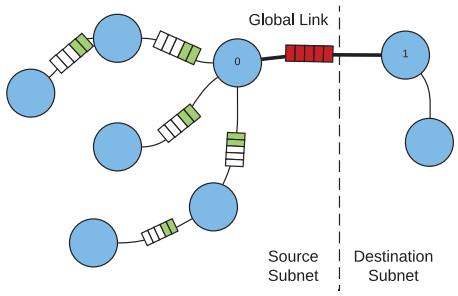


Figure 3: Example of how inappropriate adaptive routing in Fishnet-lite will cause congestion. Green tiles means low buffer usage while red tiles means high buffer usage.

To avoid this situation, we have tailored adaptive routing for the Fishnet family in the following way: When the router connects to the destination subnet is not greater than 1 hop away, we adapt the minimal path, otherwise use a VAL path. By doing this, we effectively enforce the path diversity between subnets from 1 to k and reduce the number of packets to be routed minimally to the congested link from k'^2 to k' in the worst case traffic pattern. Moreover, because all other $k^2 - k$ packets are routed randomly to other $k - 1$ intermediate subnets through $k-1$ global links, those global links will route k packets per link as well. Therefore, all the global links will have equal workloads in worst case traffic pattern.

For Fishnet, there is another place where adaptive routing decision can be made. Since there are k' links between any two subnets, minimal routing would arbitrarily route to one of them. For adaptive routing, we can examine the output buffer usage to those k' routers that offer global links to the destination subnet and choose the one with the lowest buffer usage. Because there are at most 2 hops in this process, the back propagation problem discussed earlier will be less severe here.

We refer these routing algorithms as "adaptive routing" for the rest of the paper and we will evaluate the effectiveness of these routing algorithms along with other comprehensive evaluations in the later parts of this paper.

3.2. Routing for Other Topologies

There are already a variety of established routing algorithms for other topologies involved in this study. For example, [27] proposed and compared deterministic routing and adaptive routing for Fat-tree. Their evaluation show both deterministic and adaptive routing are effective in reducing

packet latencies. Minimal, VAL, and UGAL are successfully used in Dragonfly and diameter-2 topologies [8], [19], [24]. We will implement these routing algorithms and evaluate them by simulation to get a comprehensive understanding of the effectiveness of routing algorithms.

3.3. Deadlock Avoidance

In this study, we will adapt and implement the virtual channel method proposed in [31] for each topology. Since previous studies have illustrated how to implement such methods, we will not repeat the details here.

4. Simulation Setup

In this section we describe how our simulation is set up. We introduce the simulator used in this study, SST, and the network parameters and workloads chosen for this study.

4.1. Simulation Environments

We use SST as our simulator for this study. SST is a discrete event simulator developed by Sandia National Lab, designed for modeling and simulating DOE supercomputer systems [10], [11]. To support massively parallel simulation, SST is built on top of MPI and is able to partition simulated objects across multiple MPI ranks; this can significantly speed up simulations. SST has a modular design that separate router models and end-point models. SST's *Merlin* high-radix router model has built-in support for torus, Fat-tree, and Dragonfly topologies, and it also provides a set of MPI workloads (by its *Ember* endpoint model). Additionally, SST also has built in configurable NIC model and middleware model such as *firefly* and *hermes* which provide the ability to simulate low-level protocols and message passing interfaces.

We extended SST's *Merlin* router model to support Slimfly, Fishnet and Fishnet-lite topologies along with their routing algorithms, and open sourced the code (link is not provided here for reviewing purposes). An overview of our simulated system can be found in Figure 4.

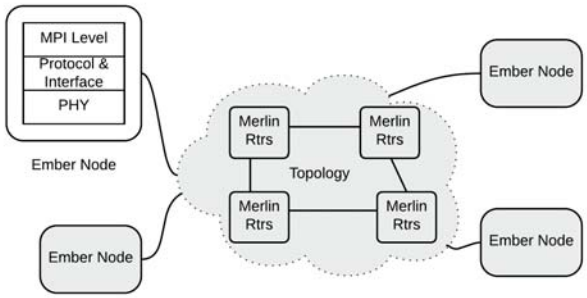


Figure 4: Overview of Simulation Setup

Each simulation is configured to run 10 iterations, and simulated execution time data is collected as the metric for performance.

4.2. Network Parameters and Workloads

In this subsection, we justify the considerations in selecting the network parameters and workloads that are used in the simulations.

4.2.1. Network parameters. In this study we primarily focus on link bandwidth and link latency as network parameter variables. Not only do these parameters have a significant influence on system performance, they also represent one of the more significant physical costs of the system.

To focus the range of the simulation parameters appropriately, we referenced some real world systems to get a perspective. Sunway TaihuLight reports the communication between nodes via MPI has a bandwidth of 12GB/s and a latency of $1\mu s$ [5]. The Tianhe-2 (MilkyWay-2) supercomputer has a MPI broadcast bandwidth of 6.36GB/s and latency of $9\mu s$ [23]. The Titan supercomputer is built on Cray's Gemini interconnect, which can achieve a peak bandwidth of 6.9GB/s and has a latency of $1\mu s$ [32]. The Sequoia supercomputer has a 5D torus network, and each node has ten 2GB/s links [33].

For Exascale, we are envisioning better physical interconnection technologies than today, with 400Gbps fabric and even 1Tb Ethernet is on the way [34], [35], [36]. Therefore, we will simulate physical link bandwidth from 8GB/s to 64GB/s and latency from $10ns$ to $200ns$, which are likely to be achieved in the foreseeable future. Note that these are just the the properties of physical links; there are also other network parameters that are tunable in SST. However, simulating all of them is impractical and out of the scope of this study, therefore we use typical or default values for those parameters unless otherwise specified. For example, the flit size is 64 Bytes and each MPI message size (payload) is 4KB; the input latency (queuing/buffering) is $30ns$ and output latency (switching and routing decision) is $30ns$. There are also delays introduced on the host side, e.g. router to host NIC latency is $4ns$, etc.

4.2.2. Workloads. As for workloads, many previous studies of such large-scale networks have used synthetic traffic patterns [7], [8], [12], [14], [20], [24], [37], e.g. uniform random traffic, nearest neighbor traffic, etc. While it is a simple way to characterize networks, it also hides communication overheads, which can be significant [38]. Therefore, we chose to use more fine-grained simulated MPI workloads offered by the SST simulator's Ember endpoint model [10]. SST not only generates traffic to the network, but also simulates the real-world behaviors during the entire lifecycle of an MPI program, as well as low level protocol and interfaces. Here are brief descriptions of the workloads.

Halo-2D: *Halo exchange* pattern is a commonly used communication pattern for domain decomposition problems [39]. Data is partitioned into grids which are mapped to MPI ranks, and at each time step, adjacent ranks exchange their boundary data.

AllPingPong: *AllPingPong* is a communication pattern that tests the network's bisection bandwidth performance:

half of the ranks in the network send/receive packages to/from the other half of the network.

AllReduce: *AllReduce* tests the network's capability of data aggregation. The communication pattern resembles traffic from a tree's leaf nodes to its root. It is the reverse process of "mapping".

Random: *Random* pattern does as the name suggests: each node sending packets to uniformly random target nodes within the network. So unlike previous workloads which all have some locality or certain traffic patterns, Random does not has locality and could thus test the network's ability to handle global traffics.

5. Evaluation & Characterization

As mentioned earlier, our experiments cover the effective cross-product of the parameter ranges given in Table 1. We present slices through the dataset, from different angles, to provide the full scope of our results.

In each of the following subsections, we will discuss one aspect from our dataset.

Also, to increase the readability of data visualization, we applied the following general rules to process the graphs plotted from the dataset:

- We only present at most 2 routing algorithms for each topology in the graph to reduce the number of datapoints in each graph. For example, the difference between deterministic and adaptive routing for Fat-tree is relatively small (comparing to Dragonfly and Fishnet-Lite) in most cases and therefore we only show the results of its adaptive routing. For those topologies with minimal, Valiant, and adaptive routings, we only present the results of adaptive and minimal routings in the graphs as they usually deliver best/worst results while VAL is often in between the two.
- In cases where the data points are too close to each other to tell apart, we provide a table below the graph. (for example, Figure 5)
- We use the following abbreviations in graphs and tables for simplicity: FT3=3-level Fat-Tree, FT4=4-level Fat-Tree, DF=Dragonfly, SF=Slimfly, FN=Fishnet, FL=Fishnet-Lite, min=minimal routing, ada=adaptive routing. For example, DF-ada will refer to Dragonfly with adaptive routing.

5.1. Link Bandwidth

In this subsection we study the effects of link bandwidth while keeping the network scale and link latency constant, (100k-node and 100ns, respectively). We then break down our data sets by 4 workloads and plot each subset. In each plot, we use the execution time of DF-ada at 8GB/s as a performance baseline and the execution time ratio of other configurations to represent their relative performance. Each bar cluster in every graph is ordered in SF-ada, SF-min, DF-ada, DF-min, FT3-ada, FT4-ada, FL-ada, FL-min, FN-ada, FN-min from left to right

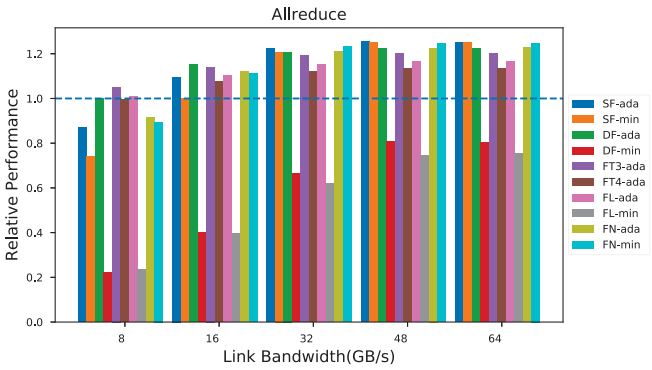


Figure 5: AllReduce workload comparison for all topology-routing combinations

AllReduce Figure 5 shows the performance of different topology-routing configurations under AllReduce workloads at different bandwidths limits. AllReduce aggregates traffic from all leaf nodes to one root node recursively. This traffic pattern is beneficial for topologies like Fat-tree, whose architecture resembles the software behavior, and is adversarial for irregularly constructed topologies. From Figure 5, we can see Dragonfly and Fishnet-lite suffers greatly when using minimal routing. Adaptive routing helps improve the performance by a factor of 5 in such cases.

We can also see that the advantage brought by topology is significant when lower bandwidths are available. e.g. both 3 level and 4 level Fat-trees have relatively better performances at 8GB/s bandwidth limits. As bandwidth limits increases, the difference in performance between topologies decreases, and ones with lower diameter and higher bisection bandwidth start to outperform others (although with thin margins).

AllPingPong The performance of AllPingPong workloads can represent the bisection bandwidth capability of a network. Not surprisingly, Fat-tree topologies again performs very well when the bandwidth limit is lower due to its high bisection bandwidth design as shown in Figure 6. But as the bandwidth limit increases, other topologies are no longer bounded by bandwidth and start to outperform Fat-trees.

Halo: Halo represents a nearest neighbor communication pattern, therefore topologies with better locality generally perform better. Consequently, without sufficient bandwidth, Fat-trees performs better than other topologies. Also note that DF-ada performs almost as good as Fat-trees at 8GB/s bandwidth, while DF-min is the slowest among all setups. Part of the reason, as previously mentioned, is the global link between groups gets congested. The other part of the reason is that Dragonfly has better "locality" because all the routers within a group are fully connected and can guarantee 1 router hop for the hosts within a group. While Slimfly connects even more hosts within 1 router hop, it's not as good as Dragonfly under 8GB/s and 16GB/s bandwidth limit. The reason behind this is consecutive MPI

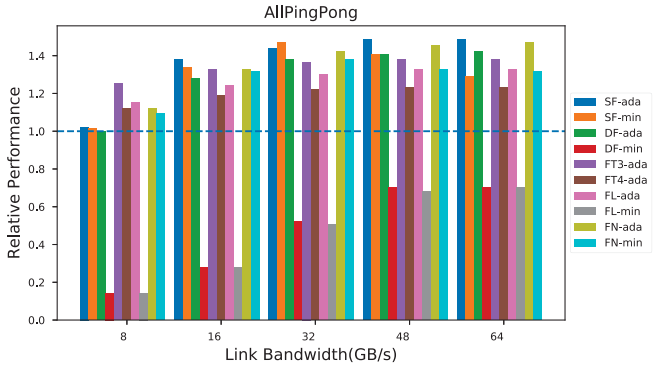


Figure 6: AllPingpong workload comparison for all topology-routing combinations

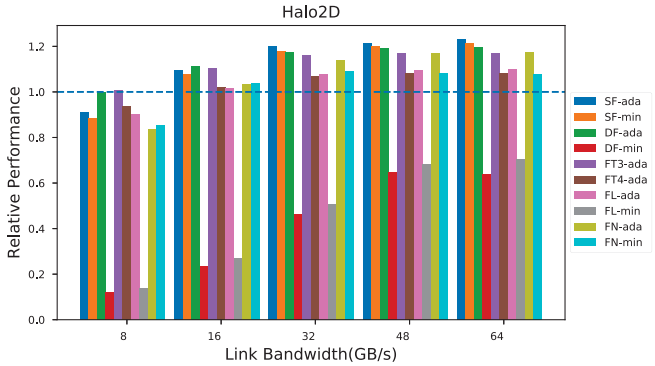


Figure 7: Halo workload comparison for all topology-routing combinations

Ranks (logical ranks) will be mapped to the hosts within a group for Dragonfly, but they are not guaranteed to be mapped to the adjacent routers in Slimfly. Consequently, some of the consecutive ranks in Slimfly will sometimes have 2-hop latency instead of 1-hop as in Dragonfly.

Random The results of random are largely different from all other workloads as shown in Figure 8. This is because random workloads generates uniform traffic pattern across all nodes, which would make use of almost no locality and the load on all the links are inherently balanced.

As a result, low diameter topologies with more path diversities, such as Fishnet, outperforms other topologies at lower bandwidth limits. The differences between topologies at lower bandwidth limits also significantly drops to a factor of less than 2 (comparing to a factor of 5 to 7 for other workloads). At higher bandwidth limit (64GB/s), the performance differences are still very significant where diameter-2 graphs beats 4-level Fat-tree by 20.5%.

Discussion We now compare across the 4 workloads and see how bandwidth affects performance for each topology. Dragonfly and Fishnet-lite with minimal routing benefit most from the growths of global link bandwidth. Increasing the bandwidth from 8GB/s to 64GB/s decreases the execution time by up to 6 to 7 times. Other topology/routing

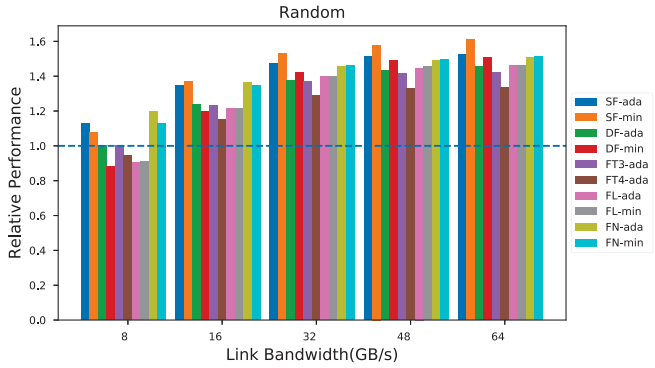


Figure 8: Random workload comparison for all topology-routing combinations

combinations tend not to gain as much performance from the bandwidth increase, but there is still an average 20% to 50% performance gain from 8GB/s to 16GB/s. To be more specific, FN-ada has a gain of 17%, SF-ada 26%, FT3-ada 36%, FL-ada 43% and DF-ada 56%.

Under our setup, bandwidth will no longer be a major bottleneck from 32GB/s and beyond as evident from Figures 5 to 8. Moving forward, this is not saying that bandwidth is unimportant once it's greater than 32GB/s; the demand for bandwidth can always be elevated by factors such as application behavior or node level architecture, e.g. if an endpoint utilizes GPUs or other accelerators that generate significantly more throughput, its demand for bandwidth can be very high. Therefore it might be more reasonable to assume that bandwidth demands will not be easily satisfied, and that the data points transition from 8GB/s to 16GB/s will be more likely to represent the real-world situations of how bandwidth increases can benefit performance.

5.2. Link Latency

In this section we will discuss how global link latency can affect network performance. We configured the physical link latency from 10ns to 200ns, and within this range, most network topologies only suffers a less than 20% slowdown moving from 10ns links to 200ns links. This indicates that most of these configurations are not latency sensitive in this range.

The only two exceptions here are Dragonfly and Fishnet-lite with minimal routing, both of which witness a slowdown of a factor of 2 moving from 10ns to 200ns latency. The global links between router groups here once again becomes the bottleneck, and it can be alleviated by using adaptive routing algorithms.

These results imply that within 200ns, link latency does not significantly sway the overall performance. Therefore, system architects may be able to exchange an increase in link latency, for greater benefits elsewhere in the system. For example, allowing more latency will extend the maximum allowable physical space to build the system, enabling more

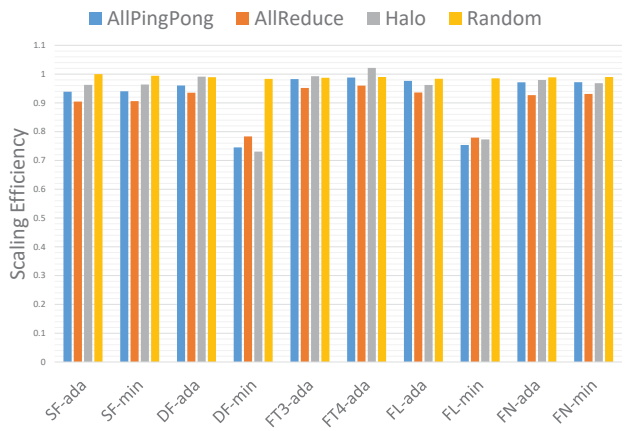


Figure 9: Averaged scaling efficiency from 50k-node to 100k-node

flexibility in physical cabinets placement, cable management, and thermal dissipation, etc.

5.3. Performance Scaling Efficiency

All the topologies that we choose to study in this paper have constant network diameters with regards to the scale of the network. So, as the network size scales up, the average distance between 2 nodes will remain the same. This does not mean there will be no performance degradations, as we will explain later with examples from our simulation data.

We simulated both 50k-node and 100k-node scale networks, each with more than 1,000 data points, on different topologies, workloads, and network parameters. By looking at this broad range of configurations, we are able to get a comprehensive view of how each topology scales.

To measure the scaling efficiency, we first find a pair of simulation data points that have the exact same configuration except for the number of nodes. Then we take the ratio of execution time of the one with 50k-node to the one with 100k-node. If there is performance degradation, meaning the same amount of workload takes more time to finish on 100k-node than 50k-node network, then this ratio will be less than 1. So the closer this ratio is to one, the better scaling efficiency the topology has.

By doing so, we obtain more than 1,000 scaling efficiency ratios for different workload, topology, and network parameter combinations. Due to the large volume of the data, we turn to a statistical approach. We observed that the scaling efficiency is relatively consistent for each workload-topology-routing combination, therefore we took the average of all the data points with the same workload-topology-routing configuration, and further reduced the number of data points to 40, as shown in Figure 9. We calculated the standard deviation for the averaged data points, and most standard deviations are below 0.01 (about 1% of the basis), indicating these averaged numbers are representative for their samples.

One would immediately notice in Figure 9 that unlike all other setups, Dragonfly and Fishnet-lite both have poor scaling efficiency when using minimal routing. The reason being that, even though the network diameter does not change, the number of nodes within a group/subnet increases. Dragonfly and Fishnet-lite both only have one global link per router group, and they will be more likely to be congested under non-uniform pattern workloads. For *Random* workload, the increased traffic generated by the host in the group/subnet are evenly distributed to more global links instead of a specific global link, thus it has good scaling efficiency. (In fact, for the same reason, *Random* has the best scaling efficiency over almost all setups)

Also note that the scaling efficiency for 4-level Fat-tree with Halo workload exceeds 1. This is because when we scale from 50,000 to 100,000 nodes, the number of nodes per router at bottom level of the Fat-tree increases, therefore more "neighbor" nodes are available within a shorter distance, which benefits nearest neighbor traffic such as Halo.

To conclude, all the topologies studied in this paper have decent scaling efficiency (greater than 0.9) with appropriate routing algorithms, which is a desired feature when moving to even larger system.

5.4. Stress Test

In this subsection, we stress test topologies with increasing workloads. We will keep the network parameters constant and increase the workload on each topology. Then we evaluate the topology's ability to handle increasing workload by observing the increase in execution time.

In this series of tests, we limit the physical link bandwidth to $8GB/s$ to make sure that light workloads are also able to cause congestions in the network, so that the efforts of increasing workloads will not be offset by high performance network parameters.

As for workloads, previous results have shown AllReduce generates adversarial traffics for most topologies while Random is benign to most topologies. Therefore we choose these two workloads for this test. To increase the workload, we double the MPI message size each time, from 512 Bytes to 64KB, which results in: 1) more packets to be sent for a message and thus more congestion in a network; 2) the input/output buffers will be filled more quickly, and NICs will have to stall to wait until the buffer is available.

Figure 10 shows the execution slowdown of different topologies under increasing AllReduce and Random workloads, respectively. The execution time of 512B message size for each configuration is chosen as the baseline (1).

Looking at the upper row of Figure 10, we can tell that Fishnet and Fishnet-lite has the modest slowdown of less than 20x in AllReduce workload when using adaptive routing, while all other configurations have more than 20x slowdown. This indicates the high bisection bandwidth and high path diversity designs of Fishnet/Fishnet-lite contributes to their performance in handling adversarial workloads.

The lower row of Figure 10 shows the slowdown of Random workload. Due to the benign nature of Random

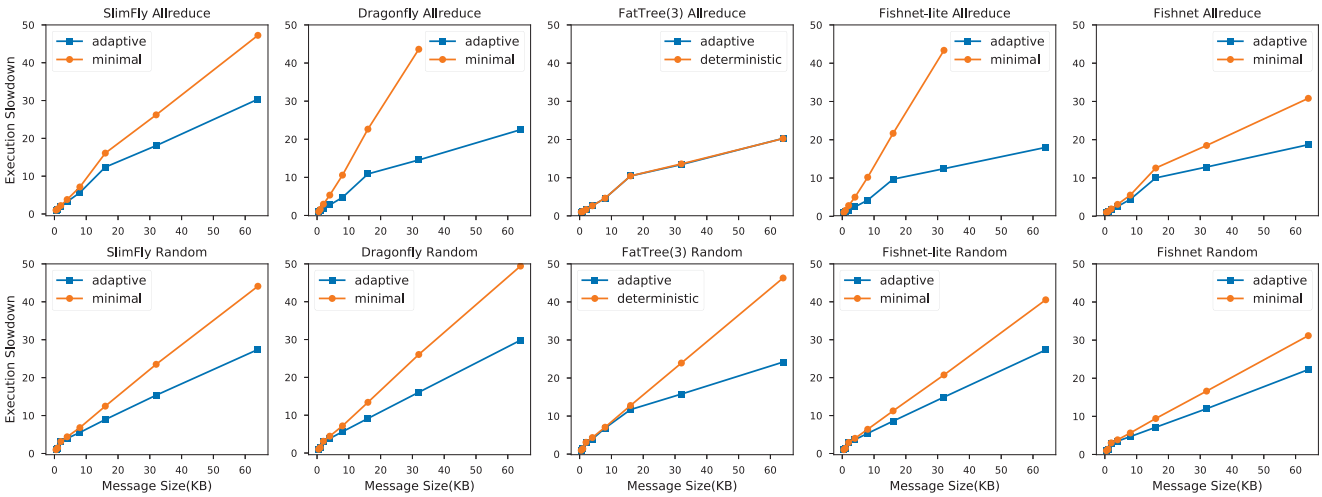


Figure 10: Execution slowdown of different topologies under increasing workload

workload, the difference in performance is not as huge as it is for AllReduce workloads when the workload increases, but it can still be seen that high bisection bandwidth architectures such as Fishnet, and Fat-tree outperform others under increasing workloads.

The effectiveness of routing algorithms against adversarial traffics could also be reflected here. By applying proper routing algorithms, the topology's ability to handle heavy workloads can be strengthened. For example, Fishnet-lite reduced the slowdown from 40x to 20x in AllReduce workload when moving from minimal routing to adaptive routing. This further proves the effectiveness of our proposed routing algorithms for Fishnet topologies.

The performance difference from routing algorithm for Fat-tree is almost negligible for AllReduce workload. This is because AllReduce is considered to be a benign traffic pattern for Fat-tree, and increasing workload does not affect the routing decision heavily. As a contrast, routing algorithm under Random workload, which causes packets to traverse more distances than AllReduce, makes more of a difference for Fat-tree, as shown in Figure 10.

6. Conclusion

In this paper, we study a wide range of network topologies that are promising candidates for Exascale high performance computing systems. We extend SST to perform large scale, fine-grained simulations for each concerned topology with different routing algorithms, various workloads and network parameters at different scales.

From a network parameter perspective, our study shows all topologies can gain a decent amount of performance from the increase of physical link bandwidth. However, the amount of performance gain from the growth of bandwidth differs greatly from topology to topology (ranging from 17% to 56%), as shown in Section 5.1. As for physical link latency, topologies with higher network diameters are naturally more sensitive to link latency, but in general, the latency range studied in this paper (10ns to 200ns) makes less contributions to the overall system performance. If allowing more latency will be beneficial for the overall system design, it might be a worthy trade-off.

The results of performance scaling efficiency and the stress test show that the studied topologies all have good performance scaling efficiency if properly set up, but their ability to handle increased workloads differs. This provides useful insights on the scenarios that we are yet unable to simulate in this study. e.g. larger scale network with even heavier workloads.

Furthermore, we identified various cases during our study where software behavior can result in significant differences in system performance. Although it is well known, we are the first to provide examples based on simulation data for a lot of the recently proposed topologies combined with network parameters, and these examples will be helpful for software optimization.

7. Acknowledgements

This research is supported in part by Northrop Grumman Corporation project "Memory-System Design for Cold Logic Program" and National Science Foundation project "3DSIM: A Unified Framework for 3D CPU Co-Simulation". We also thank Sandia National Laboratories for their support in the development of the tools used in this work.

References

[1] J. Shalf, S. Dosanjh, and J. Morrison, "Exascale computing technology challenges," in *International Conference on High Performance Computing for Computational Science*. Springer, 2010, pp. 1–25.

[2] J. Dongarra, P. Luszczek, and M. Heroux, "Hpcg: one year later," *ISC14 Top500 BoF*, 2014.

[3] "Top500 list," https://www.top500.org/, 2017.

[4] J. Dongarra and M. A. Heroux, "Toward a new metric for ranking high performance computing systems," *Sandia Report, SAND2013-4744*, vol. 312, 2013.

[5] J. Dongarra, "Report on the sunway taihulight system," *PDF). www. netlib. org. Retrieved June*, vol. 20, 2016.

[6] R. Peñaranda, C. Gómez, M. E. Gómez, P. López, and J. Duato, "The k-ary n-direct s-indirect family of topologies for large-scale interconnection networks," *The Journal of Supercomputing*, vol. 72, no. 3, pp. 1035–1062, 2016.

[7] S. Li, P.-C. Huang, D. Banks, M. DePalma, A. Elshaarany, S. Hemmert, A. Rodrigues, E. Ruppel, Y. Wang, J. Ang *et al.*, "Low latency, high bisection-bandwidth networks for exascale memory systems," in *Proceedings of the Second International Symposium on Memory Systems*. ACM, 2016, pp. 62–73.

[8] M. Besta and T. Hoefler, "Slim fly: a cost effective low-diameter network topology," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE Press, 2014.

[9] S. Amarasinghe, D. Campbell, W. Carlson, A. Chien, W. Dally, E. Elnohazy, M. Hall, R. Harrison, W. Harrod, K. Hill *et al.*, "Exascale software study: Software challenges in extreme scale systems," *DARPA IPTO, Air Force Research Labs, Tech. Rep*, 2009.

[10] A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. CooperBalls *et al.*, "The structural simulation toolkit," *ACM SIGMETRICS Performance Evaluation Review*, 2011.

[11] "Sst," http://sst-simulator.org, 2017.

[12] N. Jiang, G. Michelogiannakis, D. Becker, B. Towles, and W. Dally, "Booksim interconnection network simulator," *Online, https://nocs. stanford. edu/cgibin/trac. cgi/wiki/Resources/BookSim*.

[13] C. D. Carothers, D. Bauer, and S. Pearce, "Ross: A high-performance, low-memory, modular time warp system," *Journal of Parallel and Distributed Computing*, vol. 62, no. 11, pp. 1648–1669, 2002.

[14] M. Mubarak, C. D. Carothers, R. B. Ross, and P. Carns, "A case study in using massively parallel simulation for extreme-scale torus network codesign," in *Proceedings of the 2nd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 2014, pp. 27–38.

[15] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Cost-efficient dragonfly topology for large-scale systems," in *Optical Fiber Communication Conference and National Fiber Optic Engineers Conference*. Optical Society of America, 2009, p. OTuI2. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=OFC-2009-OTuI2

[16] C. Camarero, E. Vallejo, and R. Beivide, "Topological characterization of hamming and dragonfly networks and its implications on routing," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 11, no. 4, p. 39, 2015.

[17] M. Mubarak, C. D. Carothers, R. Ross, and P. Carns, "Modeling a million-node dragonfly network using massively parallel discrete-event simulation," in *2012 SC Companion: High Performance Computing, Networking Storage and Analysis*, Nov 2012, pp. 366–376.

[18] G. Faanes, A. Bataineh, D. Roweth, T. Court, E. Froese, B. Alverson, T. Johnson, J. Kopnick, M. Higgins, and J. Reinhard, "Cray cascade: A scalable hpc system based on a dragonfly network," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, ser. SC '12. IEEE Computer Society Press, 2012.

[19] G. Kathareios, C. Minkenberg, B. Prisacari, G. Rodriguez, and T. Hoefler, "Cost-effective diameter-two topologies: analysis and evaluation," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2015.

[20] N. Wolfe, C. D. Carothers, M. Mubarak, R. Ross, and P. Carns, "Modeling a million-node slim fly network using parallel discrete-event simulation," in *Proceedings of the 2016 annual ACM Conference on SIGSIM Principles of Advanced Discrete Simulation*. ACM, 2016.

[21] C. E. Leiserson, "Fat-trees: universal networks for hardware-efficient supercomputing," *IEEE transactions on Computers*, vol. 100, no. 10, pp. 892–901, 1985.

[22] C. Clos, "A study of non-blocking switching networks," *Bell System Technical Journal*, 1953.

[23] J. Dongarra, "Visit to the national university for defense technology changsha, china," *Oak Ridge National Laboratory, Tech. Rep., June*, 2013.

[24] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-driven, highly-scalable dragonfly topology," in *ACM SIGARCH Computer Architecture News*. IEEE Computer Society, 2008.

[25] J. Kim, J. Balfour, and W. Dally, "Flattened butterfly topology for on-chip networks," in *Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2007, pp. 172–182.

[26] W.-T. Bao, B.-Z. Fu, M.-Y. Chen, and L.-X. Zhang, "A high-performance and cost-efficient interconnection network for high-density servers," *Journal of computer science and Technology*, vol. 29, no. 2, pp. 281–292, 2014.

[27] C. Gomez, F. Gilabert, M. E. Gomez, P. López, and J. Duato, "Deterministic versus adaptive routing in fat-trees," in *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*. IEEE, 2007, pp. 1–8.

[28] L. G. Valiant, "A scheme for fast parallel communication," *SIAM journal on computing*, vol. 11, no. 2, pp. 350–361, 1982.

[29] A. Singh, "Load-balanced routing in interconnection networks," Ph.D. dissertation, Stanford University, 2005.

[30] N. Jiang, J. Kim, and W. J. Dally, "Indirect adaptive routing on large scale interconnection networks," in *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3. ACM, 2009, pp. 220–231.

[31] W. J. Dally and C. L. Seitz, "Interconnection networks," *IEEE Transactions on computers*, vol. 36, no. 5, 1987.

[32] A. Vishnu, M. ten Bruggencate, and R. Olson, "Evaluating the potential of cray gemini interconnect for pgas communication runtime systems," in *2011 IEEE 19th Annual Symposium on High Performance Interconnects*. IEEE, 2011, pp. 70–77.

[33] S. Rumley, D. Nikolova, R. Hendry, Q. Li, D. Calhoun, and K. Bergman, "Silicon photonics for exascale systems," *Journal of Lightwave Technology*, vol. 33, no. 3, pp. 547–562, 2015.

[34] B. Metcalfe, "Toward terabit ethernet," in *Conference on Optical Fiber Communication (OFC)*. Optical Society of America, 2008.

[35] K. Bergman, S. Borkar, D. Campbell, W. Carlson, W. Dally, M. Denneau, P. Franzon, W. Harrod, K. Hill, J. Hiller *et al.*, "Exascale computing study: Technology challenges in achieving exascale systems," *Defense Advanced Research Projects Agency Information Processing Techniques Office (DARPA IPTO), Tech. Rep*, vol. 15, 2008.

[36] X. Xu, E. Zhou, G. N. Liu, T. Zuo, Q. Zhong, L. Zhang, Y. Bao, X. Zhang, J. Li, and Z. Li, "Advanced modulation formats for 400-gbps short-reach optical inter-connection," *Optics express*, vol. 23, no. 1, pp. 492–500, 2015.

[37] N. Liu, A. Haider, X.-H. Sun, and D. Jin, "Fattreesim: Modeling large-scale fat-tree networks for hpc systems and data centers using parallel and discrete event simulation," in *Proceedings of the 3rd ACM SIGSIM Conference on Principles of Advanced Discrete Simulation*. ACM, 2015, pp. 199–210.

[38] D. Doerfler and R. Brightwell, "Measuring mpi send and receive overhead and application availability in high performance network interfaces," in *European Parallel Virtual Machine/Message Passing Interface Users Group Meeting*. Springer, 2006, pp. 331–338.

[39] C. Laoide-Kemp, "Investigating mpi streams as an alternative to halo exchange," 2015.