Seminar High Performance Computing - Recent trends and developments
Winter Term 2015/2016

HPC interconnection topologies - a state-of-the-art analysis

Stefan Effenberger
LMU

2.4.2016

Abstract

HPC systems of tomorrow will have to scale to a much larger size than today in order to enable performance of one exaflop and above. To achieve this scale without sacrificing performance in terms of latency and throughput, their interconnection networks and underlying topologies will have to support a high amount of nodes while keeping the diameter low and bisection bandwidth high. Additionally, these interconnection networks need to address new challenges: energy consumption and acquisition cost. A trade-off between these metrics has to be found, that is currently not satisfied by today's HPC networks. For this reason, new topologies are being developed in order to find a topology that enables HPC systems of exascale level. This paper introduces state-of-the-art network topologies and evaluates them. The results show that these topologies provide improvements over classical topologies in terms of cost, energy consumption, latency and resiliency.

1 Introduction

Exascale computing is expected to be enabled between 2021 and 2022 [15] and networks will take a major role in that matter [3]. On the other hand, the properties of such networks are determined by their underlying topologies [11]. Since computing power per endpoint is expected to increase following Moore's Law, network size has to increase drastically to enable exascale levels until 2021. Network topologies for these HPC systems therefore need to overcome limitations imposed by classical topologies in terms of scalability, power consumption and acquisition cost. In this paper, state-of-the-art topologies developed specifically for the use in HPC data centers are examined in order to get an overview of the current state of HPC interconnection topologies. Additionally, their advantages and drawbacks in comparison to classical topologies used in HPC data centers today are investigated.

Section 2 gives an overview of the challenges that have to be faced during the development of a network topology enabling a high performance large-scale network. In Section 3, various topologies along with their structure and routing algorithm are introduced. Section 4 proceeds to evaluate these topologies and compares them against each other and two popular topologies that are used in many live HPC data centers. Finally, in Section 5, a conclusion is presented.

2 Challenges

Creating a high performance network topology is subject to several challenges, which are explained in this section.

Even if performance increases per compute node will follow Moore's Law (due to increases in the number of cores per node), their number in current-gen super-computers (for example 16,000 compute nodes in the Tianhe-2 [1]) will not be sufficient for reaching exascale levels by 2021. This means that the number of compute nodes is expected to increase, with some sources estimating more than 100,000 nodes [3]. Maintaining low latency thereby is a key issue when scaling a system to a size this large. This is due to the fact that all of these nodes cannot be easily connected together with just one hop, as such a setup is limited by router radix, with radix being defined as the number of bidirectional ports in the router [3]. Currently, routers with radix 64 are available [17] and work in [19] indicates that combined routers that consist of several smaller ones could have similar performance, making routers with even higher radix possible. Still, a router cannot handle nearly as many ports as a network of 100,000 nodes would require. Thus, with growing network size, packets have to traverse a higher amount of hops resulting in higher latency.

A major scalability concern therefore is to keep the number of hops low for a high amount of nodes so as to maintain low latency. Additionally, throughput of the network is an important metric that determines its quality [13]. Advances in throughput can be achieved by allowing the packets in the network to traverse a different path if the chosen path is congested [10], leading to a high path diversity. This path diversity is enabled by the underlying topology but enforced by a routing algorithm. Depending on the topology, standard routing algorithms might not be sufficient (see [11]), so for a highly functional topology a suitable algorithm is needed as well.

Link failure is the most common reason for network collapse in large networks [3]. A good topology accounts for resiliency by allowing random link removal [5]. High path diversity not only leads to better throughput, it also enables higher resiliency. To increase path diversity, a topology has to increase its amount of links and routers. Because of the increased size of future exascale systems in comparison to today's HPC systems, a shift to low-cost architectures is happening right now [9]. Furthermore, increasing the number of routers also leads to higher power consumption, which is seen as one of the major challenges for exascale computing [12]. Thus, a trade-off between throughput, resiliency, cost-effectiveness and energy consumption has to be found.

Cost for a network is not only driven by the amount of routers and cables needed to build it. In fact, packaging of the components (including compute nodes) and the physical cabling play an important role [3]. A topology not only needs to address theoretical properties – it has to enable a proper mapping between the topology and the actual physical layout of a possible computing center.

3 Topologies

In this section, 6 different network topologies designed for high performance computing are described.

3.1 Flattened Butterfly

A *flattened butterfly* [10] topology can be directly created from any conventional *butterfly* network. It aims to add path diversity to the *butterfly* network structure in order to gain better performance on adversarial (or worst-case) traffic patterns while keeping the low cost. To achieve this, the topology is built upon the use of high-radix routers.

A *butterfly* network consists of several rows and dimensions as depicted in Figure 1 c). In each dimension, different rows are connected together. To create a *flattened butterfly*, each of these rows containing multiple routers is combined into one single router. Figure 1 shows how *butterfly* networks (a, c) are converted into *flattened butterflies* (b, d):

- All routers in a row are combined into one router. For example, routers R0 and R1 in a)

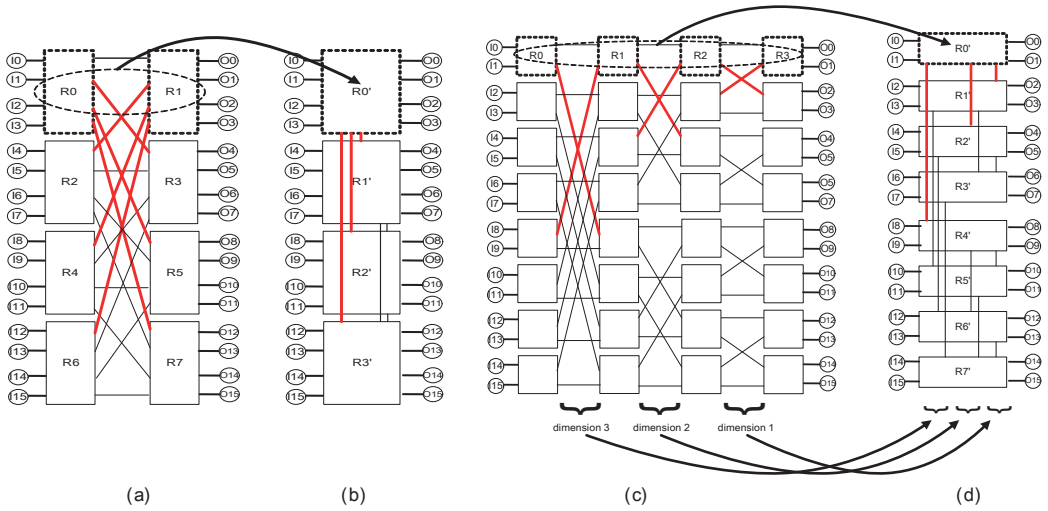

Figure 1: Conversion of Butterfly networks (a,c) to Flattened Butterfly (b,d)[10]

become R0' in b).

- Links between routers in the same row are removed.
- Unidirectional links of the *butterfly* network are combined into bidirectional links in the *flattened butterfly*. For example, Links (R0, R3) and (R2, R1) in a) become (R0', R1') in b).

This construction shows how the hop-count of a *butterfly* network can be significantly reduced while bisection bandwidth stays the same. Since multiple routers are combined into one single router, they have to handle more connections making the *flattened butterfly* only suitable for routers with many ports, i.e. high-radix routers.
The network size of a *flattened butterfly* depends on the number of dimensions (see Figure 1 c)) of the *butterfly* and the radix of the routers. Whether a *butterfly* network of specific size can actually be converted into a *flattened butterfly* is solely determined by the radix of the routers that can be used for building the physical network.
Routing in topologies like the *flattened butterfly* can be accomplished using different algorithms:

- *Minimal routing (MIN)* [11]: Packets are always sent along the path with the least hops. If multiple paths have the same hop-count, minimal routing can also be adaptive (*MIN-AD*).
- *Valiant routing (VAL)*: *Valiant's algorithm* [18] is used to route packets non-minimally. Packets are sent to a randomly selected intermediate router before they reach their destination in order to gain path diversity.
- *Universal Globally-Adaptive Load-Balanced routing (UGAL)*: Packets are sent using *MIN* by default. Queue-length and hop-count in each router is used to calculate delay. If delay on the minimal paths is too high, *UGAL* uses *VAL* to send the packets instead.

  Delay can be computed using only local information in each router (*UGAL-L*) or a global information base (*UGAL-G*) [11].

The *flattened butterfly* improves over the conventional *butterfly* network by enabling path diversity. While all packets in a *butterfly* network have to pass dimensions in a strict order, packets in a *flattened butterfly* can traverse the dimensions in any order using minimal routes (*MIN-AD*). Moreover,

packets can take non-minimal routes and thus gain additional path diversity. For example, a packet from router R0' in Figure 1 b) could reach router R1' by traversing router R2' or R3' at first.
John Kim et al. have introduced an adaptive routing algorithm for the topology called *CLOS-AD* which, like UGAL, decides for each packet whether it should be sent minimally or not at the packet's source. If a packet is chosen to be sent non-minimally however, it is routed to one of the closest ancestor routers of the source and destination nodes. These closest ancestors are computed by viewing the network as a fat-tree [14] that has been transformed into a *flattened butterfly*. According to [3], the path diversity in a *flattened butterfly* is still considerably lower than in a *fat-tree* network.

3.2 Dragonfly

Developed by the authors of the *flattened butterfly* topology, the *dragonfly* topology [11] primarily aims to improve over the *flattened butterfly* by reducing its cost and keeping latency low and throughput high. It consists of three levels:

1. A router with a fixed amount of compute nodes connected to it.
2. A group of routers inside a cabinet connected to each other.
3. The system consisting of all groups connected to each other.

The routers inside of a group effectively act as one *virtual router*, allowing the topology to emulate routers of radix higher than what is possible with single routers. Thus, the topology can maintain a low global diameter, reducing latency.
In the *dragonfly* topology, a packet traverses one global (inter-cabinet) channel at maximum. This setup allows the topology to maintain a low number of global channels. These global channels consist of optical cables enabling a network of large areal size. The purchase of optical cables is subject to higher fixed costs than electrical cables but cost per length-unit is lower [11]. The topology thus minimizes cost

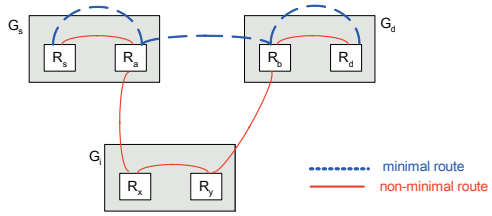

Figure 2: Routing paths in a dragonfly topology [11]

by reducing the number of global optical cables and using longer cables.
A *dragonfly* with a certain diameter is bound to a maximum network size at which only one global channel between a pair of groups is possible. *Dragonflies* with a lower amount of nodes can have additional global cables that are evenly distributed among the pairs of groups.
Since each packet can traverse up to two local inter-group channels to evenly distribute load on load-balanced traffic, John Kim et al. propose that the number of routers be exactly two times the number of inter-group connections per router. Additionally, the number of compute nodes connected to a router shall be equal to this number of inter-group connections per router.
Such a load-balanced *dragonfly* can scale to 256K nodes with a diameter of 3 and readily available radix-64 routers.
The scalability of the topology is also subject to the topologies used for inter- and intra-cabinet connections – since there are no restrictions, arbitrary configurations can be used for various use cases and budgets.
Minimal routing (*MIN*) is straightforward: A packet sent inside of a group is directly sent to the router attached to it. If a packet is sent to another group and the router of the source node does not have a connection to that group, the packet is first sent internally to a router that has such a connection. It is then sent over the global channel to the destination group. If the destination node is not attached to the router the packet arrived at, it has

the be sent again to the destination router.
In non-minimal routing (using *VAL*), one additional group is used as an intermediate group as can be seen in Figure 2.
John Kim et al. have created an adapted version of the *UGAL* routing algorithm to achieve desirable performance on the *dragonfly*. With global information about all global channels (since these are the channels that actually have to be balanced) *UGAL-G* would be ideal. Unfortunately, this is difficult to implement, so routing decisions have to be made with a version of the algorithm using local information (*UGAL-L)*. Since latency and throughput are hurt by using *UGAL-L*, the algorithm has been slightly modified to approach the optimal performance of *UGAL-G* on the network. For deadlock avoidance, virtual channels are used.

3.3 HyperX

HyperX [3] describes a topology framework. It is based upon the *flattened butterfly* (see Section 3.1) and the *hypercube* [6]. Both topologies can be completely described by the framework. It additionally comes with an algorithm that calculates the best possible topology inside the HyperX specification for various intended constraints like bisection bandwidth. For networks with full bisection bandwidth, a best-possible HyperX aims to approach performance of a *fat-tree* network, with similar or lower cost than a *flattened butterfly* network.
The *HyperX* consists of a fixed number of dimensions. For each dimension, there is a variable number of routers meaning that each dimension can consist of a different amount of routers. In each dimension the switches are fully connected.
Figure 3 shows an example of a HyperX with two dimensions (L) and 4 compute nodes (T) per router. The first dimension consists of two routers (S_1). The second dimension consists of four routers respectively (S_2), resulting in a total of 8 routers. Each router is fully connected to all other routers in the same dimension.
A *HyperX* can also describe the bandwidth (K) of the links. This bandwidth can vary for each dimension and is expressed as a multiple of the bandwidth

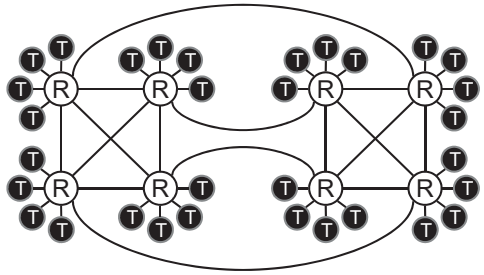

Figure 3: Example of an irregular HyperX [3]

between routers and compute nodes.
If the bandwidth K and the amount of routers S is uniform among all dimensions, a *HyperX* can be seen as regular. Such a regular HyperX is described by the tuple (L, S, K, T). A *flattened butterfly* can be expressed as a regular *HyperX* with T = S and K = 1.
Jung Ho Ahn et al. have shown that an irregular (or general) *HyperX* will always result in the usage of less routers compared to a regular one.
Since the topology space of *HyperX* is so wide, only few topologies inside of this space are desirable. For this reason, an algorithm has been proposed which can find a *HyperX* topology meeting certain requirements and minimizing the amount of routers. The requirements consist of the desired network size, radix of the routers and bisection bandwidth.
Minimal routing (*MIN-AD*) in *HyperX* is done by adaptive dimension order routing. Essentially, the packets traverse the topology one dimension after another, not traversing any dimension that already has been traversed. The more dimensions a packet has to traverse, the higher the path diversity gets as the packet can traverse dimensions in any order, adaptively choosing the path with the least buffer congestion.
HyperX introduces a new adaptive routing algorithm called *Dimensionally-Adaptive, Loadbalanced (DAL)* that is based upon the CLOS-AD algorithm described in Section 3.1. Packets are routed minimally using dimension order routing. Any router on the path of a packet can adaptively choose to der-

oute the packet on a non-minimal route based on path congestion. A maximum of one deroute per dimension is allowed. Packets are sent through dimension order virtual channels, making the routing algorithm deadlock free.

3.4 Slim Fly

Slim fly [5] is a topology based on graphs approaching solutions to the degree-diameter problem [16]. Its primary goal is thus reducing the network diameter to a minimum.

The idea behind the topology is to maximize the number of compute nodes in a network with given router radix and network diameter. The optimal graph for this problem can only be approached with the maximum number of achievable nodes in a graph with given diameter and radix being defined by the *Moore Bound* [16]. To achieve full global bandwidth, routers in the topology connect 33% of their ports to compute nodes. All other ports are connected to other routers. Using radix-128 routers and a network diameter of 2, the Moore Bound with this setup is set to approximately 300K nodes. For radix-64 routers, it is only set to around 40K nodes. This means, that the topology relies on high-radix routers.

Slim fly is based on diameter-2 graphs introduced in [16]. Figure 4 shows an exemplary graph. It is partitioned into two subgraphs each consisting of an equal amount of routers. In each graph, routers are furthermore grouped into subgroups with each subgroup not being connected to the other subgroups. Routers inside the subgroups are connected to each other. For each subgraph, the routers are connected in a different way. Additionally, each router in a subgraph is connected to some other routers in the opposing subgraph.

Each router can now reach all other routers either directly or with maximum one intermediate hop.

Minimal routing (MIN) in a *slim fly* topology is straightforward: If the router of the source node is directly connected to the router of the destination node, the packet can be sent directly. Otherwise, the packet is sent through an intermediate router.

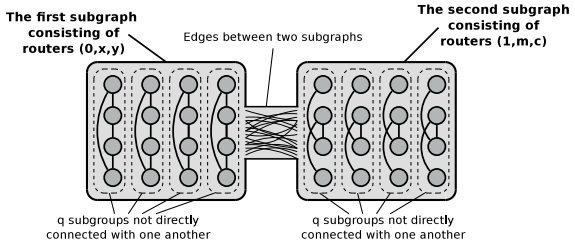

Figure 4: Slim Fly network graph with diameter 2 [5]

Maciej Besta et al. propose to use the topology with the *UGAL-L* routing algorithm.

3.5 Random topologies

Network size in conventional topologies is usually determined by their structure [13], for example because nodes and links have to be evenly distributed. In [13], an assumption is made that network size should rather be determined by surface area and cost for energy and hardware using non-structured topologies that are not subject to such limitations. Following this principle, *random topologies* have been introduced which gain latency improvements by adding random shortcut links to classical topologies like *hypercubes*. These random links help reduce the diameter of such topologies by supplying shortcuts for paths that would normally require many hops.

Shortcut links are added by sequentially traversing the list of routers and randomly selecting the desired amount of other routers to connect them to.

This algorithm is run 100 times. The 100 resulting topologies are then compared for diameter and average shortest path and the topology with the lowest values is used.

It has been shown empirically that generating only 100 different topologies is sufficient for finding the best possible solution. Furthermore, is has been found that using a generation method for shortcuts which takes the usefulness of the links into account only yields an overall 0.02 percent increase for average shortest path length. This method simply com-

putes the average shortest path length for a fixed number of randomly selected routers and picks the one with the lowest value.

Adding random shortcut links to various base topologies resulted in an interesting outcome: The more shortcut links are added, the more these topologies approach the same values in diameter and average shortest path length. Adding the shortcut links to a simple ring topology leads to the fastest decrease for these values, meaning that the ring topology is indeed the best option for adding random shortcut links. Also, not all topologies benefit from random links. A *fat-tree* for example still gains improvements in latency but it comes at the cost of massively reduced throughput. At this point, determining whether random links are beneficial or not for a specific topology can only be figured out by simulation.

Routing for random topologies can not exploit topological structures and thus is bound to topology-agnostic deadlock-free algorithms [7]. These algorithms have a higher complexity in terms of path computation and rely on routing table size which might limit scalability. These issues are still to be researched.

3.6 Skywalk

Skywalk [8] is a topology built on the premise that routing technology will reach a point where latency is dominated by cable- rather than router-delay. Ikki Fujiwara et al. define this turning point at around 60ns for router delay. Currently, the fastest routers achieve a delay of roughly 100ns [8].

Skywalk is a random topology (as described in Section 3.5). Since it has been shown that with decreasing router delay random topologies have higher latencies in comparison to topologies like the hypercube, a new method for random shortcut generation is proposed in this work. This generation method explicitly takes the physical structure of the topology into account so that cable lengths are minimized.

The topology consists of cabinets containing multiple switches each. The cabinets are aligned on a two-dimensional grid. Three different sub-topologies are specified:

- intra-cabinet
- inter-cabinet with straight links (along a row or column of the alignment grid)
- inter-cabinet with diagonal links

For each of these sub-topologies, links between the nodes (either routers or cabinets) are randomly generated. For cabinets, each newly generated link is connected to the respective next router in the cabinet so that the links are equally distributed among the routers. As Ikki Fujiwara et al. have shown, straight links between cabinets lead to lower cable length and thus latency. For this reason, the maximum number of straight links is created before diagonal links are additionally set up. This method is repeated 10 times so that 10 different topologies are generated. The topology with the largest amount of links is picked (the random shortcut generation method skips a link if the randomly chosen router already is connected to the current router).

The main differences between *Skywalk* and random topologies lie in the fact that random links are always constrained by their sub-topology and that no base topology is used.

When cable delay dominates network delay, routing on the basis of hop-count is not sufficient. Therefore, a routing algorithm should be used that routes packets based upon a combination of cable- and router-delays. A topology-agnostic deadlock-free routing algorithm (as described in Section 3.5) is proposed, for which the routing tables are computed using Dijkstra's algorithm with the distance between the nodes being defined as a combination of cable- and router-delay. Ikki Fujiwara et al. have however not defined how to compute this combination.

4 Evaluation

The topologies described in Section 3 are now evaluated and compared against each other and the

classical topologies *fat-tree* and *hypercube* regarding various metrics. For each topology, the number of compute nodes per router is set to a value that enables full global bandwidth [5]. *Fat-tree* and *flattened butterfly* are analyzed in a three-level and three-dimensional configuration, respectively.

4.1 Graph analysis

In this section, basic metrics like the nodal degree, diameter and bisection bandwidth are analyzed and compared to each other.
Table 1 shows these values for the various topologies discussed in this paper. D refers to the number of dimensions and L to the level of a topology.
The degree (amount of connections between the routers) of the topologies can vary. For example, a *random topology* can have a very low degree or a high degree depending on how many random shortcut links are generated and which base topology is used. The rating depicted here shows if low degree is possible at all.
In [5] random topologies are assumed to have a diameter between 3 and 10. For the Skywalk topology, since it is set up similarly to a *dragonfly*, the diameter is 3 at minimum and the maximum depends on the number of random links added. *Slim fly* networks have the lowest diameter. Other topologies (like *fat-tree*) can have a diameter of two as well but with significantly lower scalability [5].
The bisection bandwidth is given in relation to the network size. Bisection bandwidth in *HyperX* networks is one of the input values of the generation algorithm, making it possible to create networks with different values. The upper bound depends on the other input values but can exceed full bisection bandwidth of N/2 [3].
Figure 5 shows the average shortest path of all compared topologies except *skywalk* and *HyperX*. *Slim fly* has a value below two, since its diameter is two. *Random topologies* also have a very low value despite the fact that their diameter can be much higher. The *dragonfly* topology, with its diameter of 3, averages at around 2.5 hops per path. The other topologies average at a much higher hopcount, ultimately resulting in higher latency.

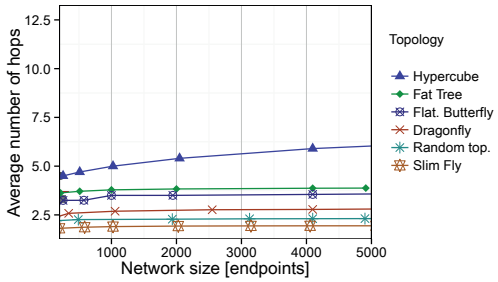

Figure 5: Average shortest path of various topologies [5]

For Skywalk, diameter and average shortest path length are not sufficient for performance analysis [8]. Therefore, no data is available.

4.2 Latency and Throughput

Latency and throughput are the two most important metrics concerning network performance. In this section, the topologies are compared to each other regarding different messaging patterns and their impact on these metrics.
Two different scenarios are used to simulate network traffic [5]: Firstly, nodes in the network send traffic to other, randomly chosen nodes. This creates uniform traffic which is expected to evenly distribute load across the network. Secondly, a worst case scenario is simulated by choosing the respective worst case traffic pattern for each topology. An explanation of these patterns can be obtained from [5]. Worst-case scenarios are only available for *fat-tree*, *dragonfly* and *slim fly*.
The results of the traffic stress tests are shown in Figure 6. All graphs show the relation between latency and the network load. This also gives insight on the throughput: The network is saturated at the point where latency increases drastically. a) and b) show random traffic and worst-case scenarios respectively for *fat-tree*, *dragonfly* and *slim fly* topologies. c) shows a (different) random traffic scenario for the rest of the topologies in this paper. It is based on routers with 60ns delay so that a comparison with *skywalk* can be made on a fair

Table 1: Graph analysis for various topologies [5]

	Hyper-cube	Fat-tree	Flat. But.	Dragon-fly	HyperX	Slim Fly	Random	Skywalk
Degree	low [6]	high [14]	high [10]	high [11]	high [3]	high [5]	low [13]	low [8]
Diameter	D [6]	$(L-1)^2$ [14]	D [10]	3	D [3]	2	3-10	3-? [8]
Bisection BW	$N/2$	$N/2$	$N/4$	$N/4$	*	$\approx N/3$	$\approx N/2.3$	-

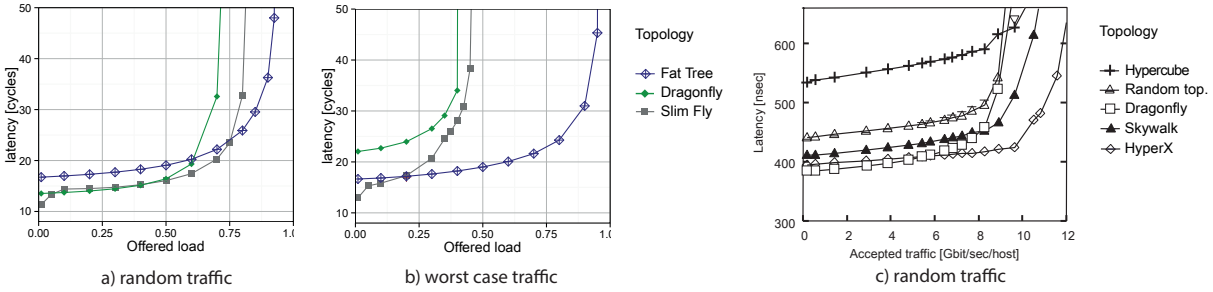

Figure 6: Traffic scenarios (a,b) [5] and random traffic scenario with 60ns router delay (c) [8]

basis. Additionally, the topology setup of *dragonfly* is different to the setup used in the other two graphs. Therefore, the graph from c) cannot be directly compared with the graphs from a) and b).

a) shows that latency in *dragonfly* and *slim fly* networks is lower than in a classical 3-layer *fat-tree* network. It can support higher network load though, so throughput is slightly higher for *fat-tree* and *slim fly* can handle more load than *dragonfly*.

b) shows that the *fat-tree* network has a clear advantage over the other two topologies when worst case traffic occurs. This comes at the expense of a higher cost as can be seen in Section 4.3.

c) shows that the *hypercube* topology has a high latency overall despite the fact that latency should not be dominated by its high average shortest path length in this 60ns router delay environment. *HyperX* can handle the highest load, while *dragonfly* is showing the best latency for lower load. Interestingly, *skywalk* is still outperformed by *HyperX* and *dragonfly* in this environment.

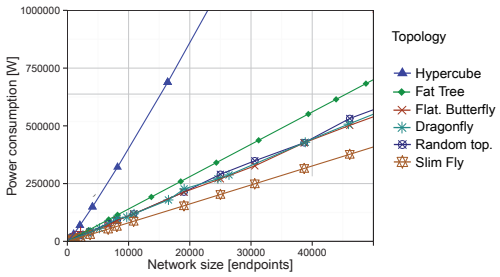

Figure 7: Energy efficiency of various topologies [5]

4.3 Energy efficiency and cost

According to [2], up to 50% of a data center's energy consumption can be caused by the underlying network. Furthermore, energy consumption is seen as a major challenge for exascale computing [12]. It is therefore important for HPC networks to reduce energy consumption by a whole magnitude to enable exascale computing. Figure 7 shows a

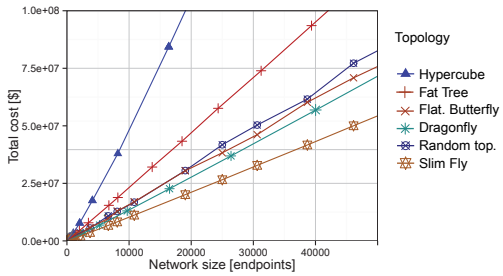

Figure 8: Cost of various topologies [5]

comparison between various network topologies in terms of energy consumption. The exact testing setup and cost functions can be obtained in [5]. The graph shows that *slim fly* is the clear winner in terms of energy consumption with approximately 26 percent less power consumption than a *dragonfly* network of similar size. This is mainly achieved by the topology using less routers than its competitors.

Routers and cables account for almost all network cost [11]. Therefore, a reduction in both leads to a better cost-effectiveness of the whole network. Figure 8 depicts the cost of a network in relation to its size for various topologies. Similar to the results for energy consumption, *slim fly* is the topology with the best cost-effectiveness of the tested topologies. *HyperX* networks also account for low cost by preferring the configuration with the least routers. *Skywalk* reduces network cost mainly by reducing cable length. However, as no direct metrics in terms of cost-effectiveness are available for both topologies, they cannot be included in the comparison.

4.4 Resiliency

In [5], various topologies have been evaluated against sudden link failure:
The *hypercube* is more resistant against link failures than the *fat-tree* regarding low amount of nodes (around 512-1024). This behavior reverses with a higher amount of nodes, because more links between routers are established in a fat-tree network with growing size.
The *dragonfly* topology is generally more resilient than *fat-tree* and *hypercube*. Fat-tree networks tend to failure when enough global links between the core routers fail. This is true for *dragonfly* as well but it is still more resilient.
The *flattened butterfly* is more resilient than the *dragonfly* because of its high path diversity and the fact that link failure of global links in a *dragonfly* topology can lead to network partitioning easily.
Random topologies and *slim fly* are more resilient than all already mentioned topologies and in this context approximately equal to each other, thanks to high path diversity.
For *skywalk* and *HyperX*, there are no metrics available in this regard.
It has to be noted that resiliency not only depends on the network structure but also on the routing algorithm [3]. This has not been taken into account in the presented metrics.

5 Conclusion

To enable exascale computing, HPC network topologies have to overcome limitations due to energy consumption, cost and performance. In this paper, several topologies have been evaluated regarding these properties and compared against each other and popular topologies already in use in today's HPC data centers.
The evaluation shows that some of the examined topologies are strong candidates for next-generation high performance computing (with dragonfly currently being the only topology in use [4]). With many of the new topologies focusing on cost-effectiveness, a high cost and energy consumption reduction in comparison to classical *fat-tree* networks that are in use in many HPC networks today [1] can be achieved with even lower latency and higher resiliency. Topologies like *dragonfly* and *slim fly* can surpass classical topologies in many regards, although there are drawbacks as well: For adversarial traffic patterns, highly redundant networks like the *fat-tree* accept significantly more traffic before they are saturated.
Random topologies have shown that latency can

be reduced by adding random links to an existing topology. This might enable already existing data centers to improve performance with little effort.

References

[1] The top 500. `http://www.top500.org`. Accessed: 2015-12-06.

[2] Dennis Abts, Michael R. Marty, Philip M. Wells, Peter Klausler, and Hong Liu. Energy proportional datacenter networks. *SIGARCH Comput. Archit. News*, 38(3):338–347, June 2010.

[3] Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S. Schreiber. Hyperx: Topology, routing, and packaging of efficient large-scale networks. In *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis*, SC '09, pages 41:1–41:11, New York, NY, USA, 2009. ACM.

[4] B. Arimilli, R. Arimilli, V. Chung, S. Clark, W. Denzel, B. Drerup, T. Hoefler, J. Joyner, J. Lewis, Jian Li, Nan Ni, and R. Rajamony. The percs high-performance interconnect. In *High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on*, pages 75–82, Aug 2010.

[5] M. Besta and T. Hoefler. Slim fly: A cost effective low-diameter network topology. In *High Performance Computing, Networking, Storage and Analysis, SC14: International Conference for*, pages 348–359, Nov 2014.

[6] L.N. Bhuyan and D.P. Agrawal. Generalized hypercube and hyperbus structures for a computer network. *Computers, IEEE Transactions on*, C-33(4):323–333, April 1984.

[7] J. Flich, T. Skeie, A. Mejia, O. Lysne, P. Lopez, A. Robles, J. Duato, M. Koibuchi, T. Rokicki, and J.C. Sancho. A survey and evaluation of topology-agnostic deterministic routing algorithms. *Parallel and Distributed Systems, IEEE Transactions on*, 23(3):405–425, March 2012.

[8] I. Fujiwara, M. Koibuchi, H. Matsutani, and H. Casanova. Skywalk: A topology for hpc networks with low-delay switches. In *Parallel and Distributed Processing Symposium, 2014 IEEE 28th International*, pages 263–272, May 2014.

[9] Georgios Kathareios, Cyriel Minkenberg, Bogdan Prisacari, German Rodriguez, and Torsten Hoefler. Cost-effective diameter-two topologies: Analysis and evaluation. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '15, pages 36:1–36:11, New York, NY, USA, 2015. ACM.

[10] John Kim, William J. Dally, and Dennis Abts. Flattened Butterfly : A Cost-Efficient Topology for High-Radix Networks. In *Proceedings of the 34th annual international symposium on Computer architecture*, pages 126–137, 2007.

[11] John Kim, William J. Dally, Steve Scott, and Dennis Abts. Technology-Driven, Highly-Scalable Dragonfly Topology. In *Proceedings of the 35th Annual International Symposium on Computer Architecture*, pages 77–88, 2008.

[12] P.M. Kogge, P. La Fratta, and M. Vance. [2010] facing the exascale energy wall. In *Innovative Architecture for Future Generation High Performance (IWIA), 2010 International Workshop on*, pages 51–58, Jan 2010.

[13] M. Koibuchi, H. Matsutani, H. Amano, D.F. Hsu, and H. Casanova. A case for random shortcut topologies for hpc interconnects. In *Computer Architecture (ISCA), 2012 39th Annual International Symposium on*, pages 177–188, June 2012.

[14] C.E. Leiserson. Fat-trees: Universal networks for hardware-efficient supercomputing. *Computers, IEEE Transactions on*, C-34(10):892–901, Oct 1985.

[15] Dong-Joon Lim, Timothy R. Anderson, and Tom Shott. Technological forecasting of supercomputer development: The march to exascale computing. *Omega*, 51:128 – 135, 2015.

[16] Mirka Miller and Jozef Siran. Moore graphs and beyond: A survey of the degree/diameter problem. *The electronic journal of combinatorics*, 2005.

[17] Steve Scott, Dennis Abts, John Kim, and William J. Dally. The blackwidow high-radix clos network. In *Proceedings of the 33rd Annual International Symposium on Computer Architecture*, ISCA '06, pages 16–28, Washington, DC, USA, 2006. IEEE Computer Society.

[18] L. G. Valiant. A scheme for fast parallel communication. *SIAM Journal on Computing*, 11(2):350–361, 1982.

[19] Juan A. Villar, Francisco J. AndúJar, José L. SáNchez, Francisco J. Alfaro, José A. GáMez, and José Duato. Obtaining the optimal configuration of high-radix combined switches. *J. Parallel Distrib. Comput.*, 73(9):1239–1250, September 2013.