COST-EFFICIENT DRAGONFLY TOPOLOGY FOR LARGE-SCALE SYSTEMS

IT IS MORE EFFICIENT TO USE INCREASING PIN BANDWIDTH BY CREATING HIGH-RADIX ROUTERS WITH A LARGE NUMBER OF NARROW PORTS INSTEAD OF LOW-RADIX ROUTERS WITH FEWER WIDE PORTS. BUILDING NETWORKS USING HIGH-RADIX ROUTERS LOWERS COST AND IMPROVES PERFORMANCE, BUT ALSO PRESENTS MANY CHALLENGES. THE DRAGONFLY TOPOLOGY MINIMIZES NETWORK COST BY REDUCING THE NUMBER OF GLOBAL CHANNELS REQUIRED.

John Kim
Northwestern University

William Dally
Stanford University

Steve Scott
Cray

Dennis Abts
Google

The interconnection network that connects processors and memory modules in scalable multiprocessors and large-scale systems significantly impacts the system's overall performance and cost. As processor and memory performance continue to increase, large-scale interconnection networks are becoming even more critical because they largely determine the bandwidth and latency of remote memory access. A good interconnection network is designed around the capabilities and constraints of available technology. Previous interconnection networks have been built with low-radix routers—that is, routers with a small number of ports. As a result, these networks used low-radix topologies such as 2D or 3D mesh or torus networks. Examples of machines employing such networks include the Cray T3D, T3E, and XT3, SGI Origin2000, and Alpha 21364. Earlier work showed that, for the packaging and technology constraints of the 1980s and 1990s—and the relatively low pin bandwidth available at that time—low-radix networks provided optimal latency for a given cost.[1,2] However, this is no longer the case.

Over the past 20 years, the pin bandwidth of router chips has increased by approximately an order of magnitude every five years—a rate similar to Moore's law (see Figure 1). This increase in bandwidth is a result of both an increase in the signaling rate and an increase in the number of signals available to a router chip.

However, the best way to exploit this increasing off-chip bandwidth isn't just to make the ports wider—that is, building low-radix routers with *fat* channels. Rather, it's more efficient to increase the number of ports—to build high-radix routers with *thin* channels.[3] The use of high radix reduces hop count and leads to lower latency and lower cost. The zero-load latency consists of *header latency*, which is proportional to the

0272-1732/09/$25.00 © 2009 IEEE Published by the IEEE Computer Society

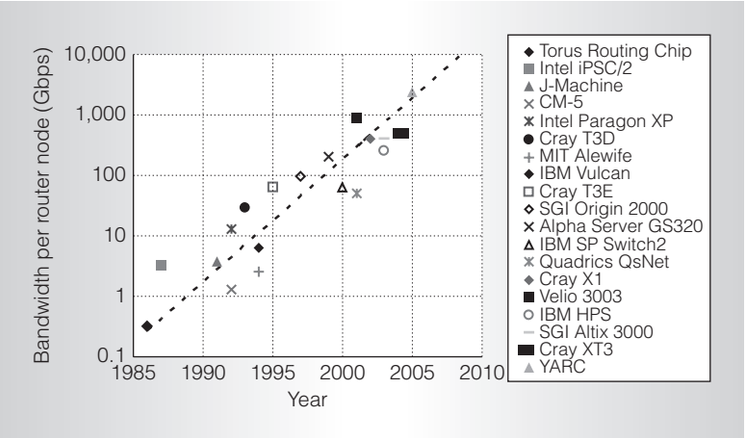

Figure 1. Router bandwidth scaling relationship.

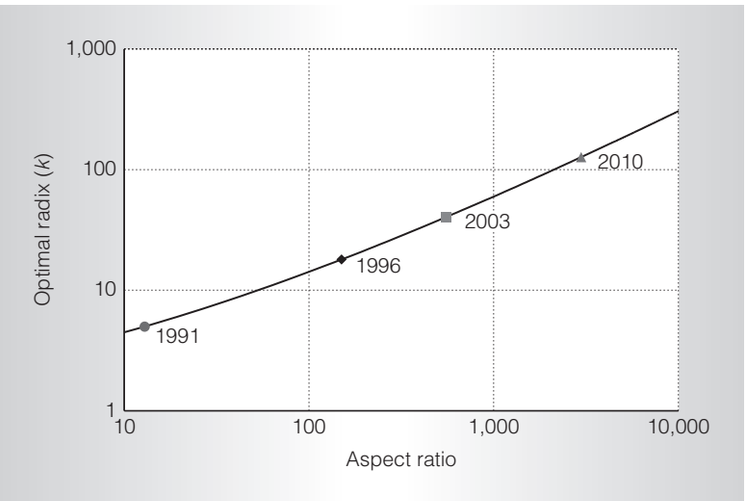

Figure 2. Relationship between optimal radix as a function of the aspect ratio. The labeled points show the approximate aspect ratio for a given year's technology and an estimate technology for 2010.

hop count of the network, plus *serialization latency,* which is inversely proportional to the bandwidth of each channel. As a router's radix or degree increases, hop count and hence header latency decrease, because more nodes can be reached in a single hop. At the same time, serialization latency increases because the bandwidth per channel decreases. The optimum latency, which occurs when we balance these two components, is proportional to the aspect ratio (A) quantity, which can be defined as $(Bt_r \log N)/L$, where B is the total router pin bandwidth, t_r is the per-hop router latency, N is the network size, and L is the packet size. As pin bandwidth and hence aspect ratio increase, the optimal radix also increases, as Figure 2 shows. By 2010, the optimal radix will be approximately 128.

With the reduced hop count, high-radix routers also reduce the network's cost, which is largely determined by its total channel bandwidth. As hop count decreases, each packet consumes less channel bandwidth; hence, the same network performance can be achieved with lower total channel bandwidth. As a result, network cost decreases monotonically as radix increases, and in a similar manner, the network's total power also decreases with the use of high-radix routers in the network.

Migration to high-radix networks presents many benefits. (See the sidebar, "Cray BlackWidow System," for an example of a recent system using high-radix routers.) However, efficiently exploiting the benefits of high-radix routers requires rethinking conventional network topologies. We propose the dragonfly topology, which uses a group of routers as a virtual router to increase the effective radix of the network, and hence reduce network diameter, cost, and latency. Because it reduces the number of global cables in a network, while at the same time increasing their length, the dragonfly topology is particularly well suited for implementations using emerging active optical cables—which have a high fixed cost but a low cost per unit length compared to electrical cables.

Dragonfly topology

Topology is a critical aspect of any interconnection network because it sets performance bounds for the network by establishing the network diameter and bisection bandwidth. The topology also largely determines the system's cost, in terms of both capital and energy consumption—thus, using a cost-efficient network topology is critical. Existing topologies, such as folded-Clos and fat-tree, pay too high a penalty on load-balanced traffic (for example, uniform random) to provide good performance in an adversarial traffic pattern. In essence, they consume costly bandwidth to load-balance traffic that is already balanced.

A conventional butterfly network, on the other hand, incurs significantly lower

Cray BlackWidow System

The Cray BlackWidow system,[1] introduced as part of the Cray $XT5_h$ system, is one of the first systems to exploit high-radix routers. The network in the BlackWidow system uses radix-64 routers and a variant of the high-radix folded-Clos topology to scale up to 32 K processing nodes with a maximum diameter of only seven hops.

The network is a significant departure from previous Cray machines, which relied on low-radix networks such as 2D or 3D mesh or torus networks. The Cray XT MPP series, introduced in 2004, used the 7-ported SeaStar router with a total bandwidth of more than 460 Gbps.[2] The more recent BlackWidow system is built from the Cray YARC router, which provides a total bandwidth of 2.4 Tbps, and is divided into 64 ports.

The YARC router uses an 8×8 array of tiles (see Figure A).[3] Each YARC tile consists of an input and an output port, an 8×8 crossbar subswitch providing connectivity between the eight tiles in the row and the eight tiles in the column, several sets of buffers, and associated routing logic. The YARC router is implemented in a 90-nm CMOS standard-cell ASIC technology, with 192 6.25-Gbps serializer/deserializers (SerDes) around the periphery of the 17×17-mm silicon. The 192 SerDes are divided among the 64 ports to provide each channel in the network with a bandwidth of 18.75 Gbps.

The Cray BlackWidow employs a variant of the folded-Clos topology. Instead of using only uplinks and downlinks to connect the nodes, BlackWidow employs *sidelinks* in the topology to connect the subtrees of the networks. The use of sidelinks reduces the network's cost and latency by reducing intermediate routers. The recently proposed flattened butterfly topology improves on this topology by creating a topology in which all links are essentially sidelinks. The dragonfly topology described in this article improves on the flattened butterfly by reducing the number of global channels required in the topology.

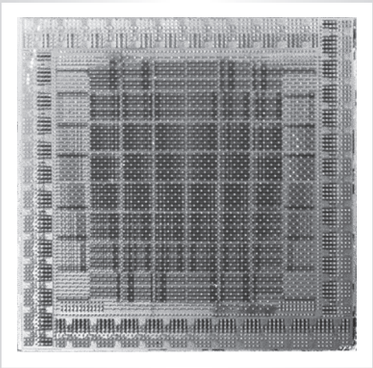

Figure A. Die photo of radix-64 YARC router used in the Cray BlackWidow system.

References

1. D. Abts et al., ''The Cray BlackWidow: A Highly Scalable Vector Multiprocessor,'' *Proc. ACM/IEEE Conf. Supercomputing* (SC 07), ACM Press, 2007; http://doi.acm.org/10.1145/1362622.1362646.
2. R. Brightwell et al., ''SeaStar Interconnect: Balanced Bandwidth for Scalable Performance,'' *IEEE Micro,* vol. 26, no. 3, May/June 2006, pp. 41-57.
3. S. Scott et al., ''The Black Widow High-Radix Clos Network,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 06), IEEE CS Press, 2006, pp. 16-28.

cost on balanced traffic—approximately half that of a folded-Clos network. However, because a conventional butterfly has no path diversity, its performance is severely limited on adversarial traffic patterns. The recently proposed *flattened butterfly* approaches the cost of a conventional butterfly network on balanced traffic while matching the cost/performance of a folded-Clos topology on adversarial traffic.[4] We derive the flattened butterfly topology from a conventional butterfly network by combining or *flattening* each row of routers and maintaining the same inter-router connection. However, the flattened butterfly's scalability is limited by the radix of a single router. In addition, the cost of a network is dominated by channels—especially the long global channels. The flattened butterfly requires each packet to traverse multiple global channels, which increases the network cost.

The proposed *dragonfly* topology, on the other hand, effectively increases the radix of a network by combining a number of high-radix routers into a *group* which acts as a very-high-radix virtual router. The dragonfly also reduces the *global diameter* (the maximum number of expensive global channels on the minimum path between any two nodes) to one.[5]

Achieving this unity global diameter requires very high-radix routers, with a radix of $\sim 2\sqrt{N}$ (where N is the size of the network), assuming a fully connected topology with a concentration of $\sqrt{N}$. Although radix-64 routers have been introduced,[6] building machines that scale to 8 K to 1 M

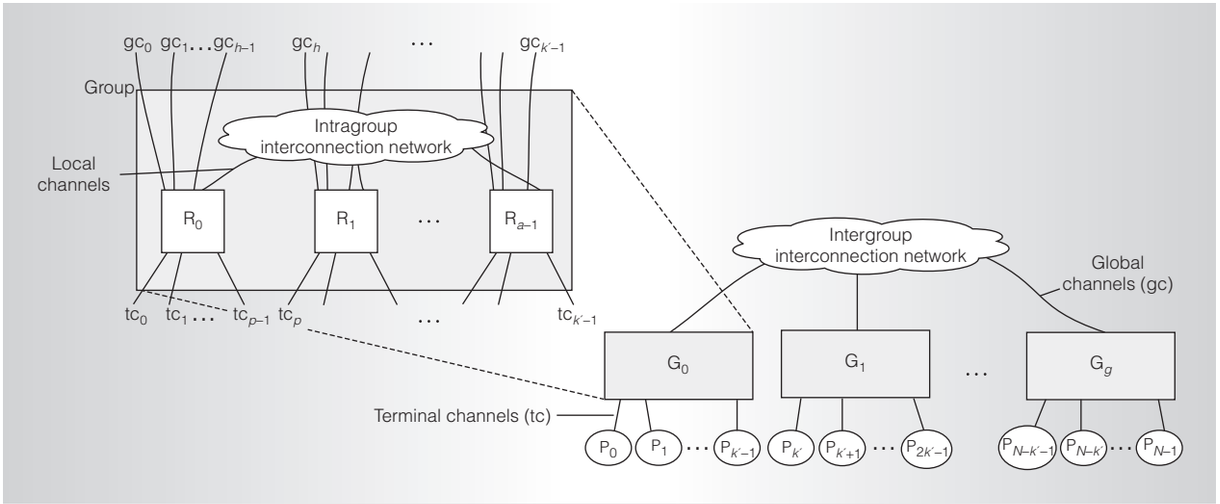

Figure 3. A high-level block diagram of a dragonfly topology and a diagram of a group or a virtual router.

nodes with unity global diameter requires higher radices. To achieve the benefits of a very high radix, the dragonfly topology uses a group of routers connected into a subnetwork to create one very high-radix virtual router, as shown in Figure 3. This very high effective radix in turn lets us build a network in which all minimal routes traverse at most one global channel. The high effective radix also means the dragonfly topology can provide high scalability—with radix-64 routers, the topology can scale to over 256 K nodes with a network diameter of only three hops.

As Figure 3 shows, the dragonfly is a hierarchical network[7] with three levels: router, group, and system. At the bottom level, each router has connections to p terminals, $a-1$ local channels (to other routers in the same group), and h global channels (to routers in other groups). Hence the radix (or degree) of each router is $k = p + a + h - 1$. A group consists of a routers connected via an intragroup interconnection network formed from local channels. Each group has ap connections to terminals and ah connections to global channels, and all of the routers in a group collectively act as a *virtual router* with radix $k' = a(p + h)$. This very high radix, $k' >> k$, lets us realize the system-level network with very low global diameter. Up to $g = ah + 1$ groups—$N = ap(ah + 1)$ terminals—can be connected with a global diameter of one. In contrast, a system-level network built directly with radix-k routers would require a larger global diameter. In a maximum-size dragonfly—$N = ap(ah+1)$—there is exactly one connection between each pair of groups. In smaller dragonflies, there are more global connections out of each group than there are other groups. These excess global connections are distributed over the groups, with each pair of groups connected by at least $\lfloor ah + 1/g \rfloor$ channels.

The dragonfly parameters a, p, and h can have any values. However, to balance channel load on load-balanced traffic, the network should have $a = 2p = 2h$. Because each packet traverses two local channels along its route (one at each end terminal channel), this ratio maintains balance. Because global channels are expensive, deviations from this 2:1 ratio should be done in a manner that overprovisions local and terminal channels, so that the expensive global channels remain fully utilized. That is, the network should be balanced so that $a = 2h$, and $2p = 2h$. Arbitrary networks can be used for the intragroup and intergroup networks in Figure 3.

The high-radix topology, especially the dragonfly topology, increases the global channels' physical length; however, exploiting emerging optical signaling technology can reduce the impact of long global channel lengths. Historically, researchers have proposed many networks using optical signaling,

but because of this technology's high cost, it hasn't been used in large-scale systems. However, the recent advent of economical optical signaling[8] enables topologies with long channels, but they're still more expensive than electrical channels. The proposed dragonfly results in a hierarchical topology that exploits economical optical signaling for the global channels but uses the cheap electrical channels for short local communication.[7] Previously proposed hierarchical topologies have often been built as tree structures that introduce a bandwidth bottleneck and increase hop count as the packets travel up the hierarchy.

Indirect adaptive routing

Minimal routing in a dragonfly, from source node s attached to router R_s in group G_s to destination node d attached to router R_d in group G_d, traverses a single global channel and is accomplished in three steps:

1. If $G_s \neq G_d$ and R_s does not have a connection to G_d, route within G_s from R_s to R_a, a router that has a global channel to G_d.
2. If $G_s \neq G_d$, traverse the global channel from R_a to reach router R_b in G_d.
3. If $R_b \neq R_d$, route within G_d from R_b to R_d.

This minimal routing works well for load-balanced traffic, but results in poor performance on adversarial traffic patterns. To load-balance adversarial traffic patterns, we can apply Valiant's algorithm[9] at the system level—routing each packet first to a randomly selected intermediate group G_i and then to its final destination d. Applying Valiant's algorithm to groups suffices to balance load on both the global and local channels. This randomized nonminimal routing traverses at most two global channels and requires five steps:

1. If $G_s \neq G_i$ and R_s doesn't have a connection to G_i, route within G_s from R_s to R_a, a router that has a global channel to G_i.
2. If $G_s \neq G_i$, traverse the global channel from R_a to reach router R_x in G_i.
3. If $G_i \neq G_d$ and R_x doesn't have a connection to G_d, route within G_i from R_x to R_y, a router that has a global channel to G_d.
4. If $G_i \neq G_d$, traverse the global channel from R_y to router R_b in G_d.
5. If $R_b \neq R_d$, route within G_d from R_b to R_d.

The dragonfly topology's benefits can't be fully exploited without adaptive routing—that is, adapting between minimal and nonminimal routing on the basis of the network's state. Although the topology provides high path diversity, it needs nonminimal global adaptive routing to properly exploit the diverse paths. Achieving good performance on a wide range of traffic patterns on a dragonfly topology requires a routing algorithm that can effectively balance load across the global channels. Global adaptive routing (UGAL)[10] can perform such load balancing if the load of the global channels is available at the source router, where the routing decision is made. With the dragonfly topology, however, the source router is most often not connected to the global channel in question. Hence, the adaptive routing decision must be made on the basis of remote or indirect information—relying on backpressure through the queue to sense downstream congestion. With conventional UGAL, the indirectness of this decision (using local queue occupancy to make routing decisions) degrades both latency and throughput. Thus, the dragonfly topology requires *indirect* adaptive routing to load-balance the global channels.

We use UGAL-L (UGAL local) as the baseline routing algorithm where the routing decision is based on local queue information at the current router node. With two modifications, the UGAL-L routing algorithm can overcome its limitation with regard to the dragonfly topology, and indeed yield performance results approaching an ideal implementation using global information. Adding selective virtual-channel discrimination to UGAL (UGAL-$L_{VC\text{-}H}$) eliminates bandwidth degradation due to local-channel sharing between minimal and nonminimal paths. Using credit round-trip latency both to sense global-channel congestion and to propagate this congestion information upstream (UGAL-L_{CR}) eliminates latency degradation by providing much stiffer

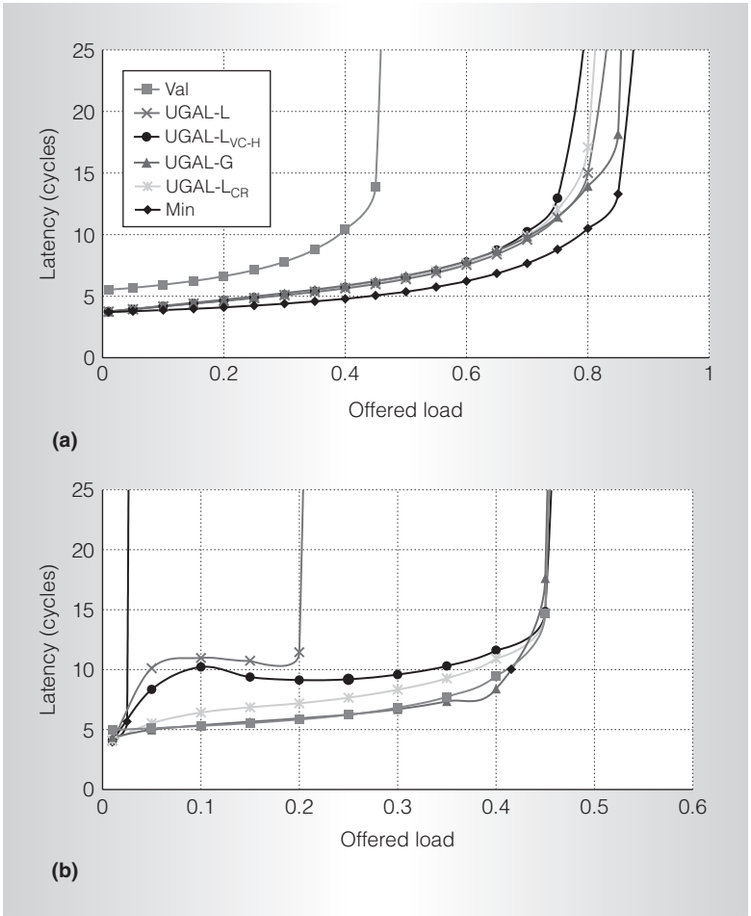

Figure 4. Routing algorithm performance comparison for uniform random traffic (a) and worst-case traffic pattern (b).

backpressure than when the algorithm uses only queue occupancy for congestion sensing.

We compared these two routing algorithms to two other UGAL implementations: the baseline UGAL-L and *UGAL-G,* which uses queue information for all the global channels within the source group. Although UGAL-G is difficult to implement, it represents an *ideal* implementation of UGAL because it requires load balancing of the global channels, not the local channels. We also compared these routing algorithms to minimal routing (Min) as well as Valiant's routing (Val), using synthetic traffic patterns that included uniform random traffic and the worst-case traffic pattern, where all nodes in group G_i send their traffic to G_{i+1}. Load-balancing the worst-case traffic pattern requires nonminimal routing, which spreads the bulk of the traffic across the other global channels.

To evaluate the performance of the different routing algorithms, we used cycle-accurate simulations. We simulated a single-cycle, input-queued router switch but provided sufficient speedup to generalize the results and ensure that routers don't become the network's bottleneck. We injected packets using a Bernoulli process. We warmed up the simulator under load without taking measurements until it reached steady state. Then, we labeled a sample of injected packets during a measurement interval and ran the simulation until all labeled packets exited the system. Figure 4 shows simulation results for a dragonfly of size 1 K nodes, using the parameters $p = h = 4$ and $a = 8$. Simulations of other network sizes follow the same trend. We use single-flit (flow control unit) packets to separate the routing algorithm from flow-control issues such as the use of wormhole or virtual cut-through flow control. The input buffers are assumed to be 16 flits deep. A 1D flattened butterfly topology is assumed for both intragroup and intergroup topology.

With Min routing, the network achieves optimal performance (low latency and high throughput) for benign traffic such as uniform random traffic. However, because Min doesn't exploit the topology's path diversity, the throughput on worst-case traffic is severely limited, at $1/ah$. For uniform random traffic, Val achieves approximately half the network capacity, because its load-balancing doubles the load on the global channels; it also achieves similar throughput on adversarial traffic patterns. Thus, global adaptive routing aims to achieve the performance of Min on uniform random traffic while matching the performance of Val on adversarial or worst-case traffic, as illustrated with UGAL-G. By simply relying on local information, UGAL-L matches Min on throughput for uniform random traffic but leads to both limited throughput and high average packet latency at intermediate load. UGAL-L$_{VC-H}$ leads to higher throughput in worst-case traffic pattern by differentiating between minimal and nonminimal traffic to be routed

through the same local output port. Although UGAL-L_{VC-H} achieves throughput comparable to UGAL-G, it still leads to high intermediate latency because it needs to route many packets minimally before it can sense the congestion downstream and, in response, route packets nonminimally for load-balancing. In other words, soft backpressure between the congestion and the source router creates high intermediate latency. UGAL-L_{CR}, which uses credit round-trip latency, overcomes this high intermediate latency by providing the appearance of a shallow buffer to stiffen backpressure and propagate the global congestion information.

Cost comparison

Figure 5 compares costs of the dragonfly topology to alternative topologies using a detailed cost model.[4] By reducing global channels, a dragonfly reduces cost by 20 percent compared to a flattened butterfly, and by 52 percent compared to a folded-Clos network in configurations with more than 16 K nodes. Compared to a 3D torus topology, which requires relatively short electrical cables, the dragonfly still provides a cost savings of up to 60 percent, because it significantly reduces the number of cables (or channels) required. The reduction of network cost in the dragonfly also translates to a reduction of power, as prior work has shown.[4] (To provide accurate cost comparisons, we implicitly normalize the throughput—or the performance—of the alternative topologies.)

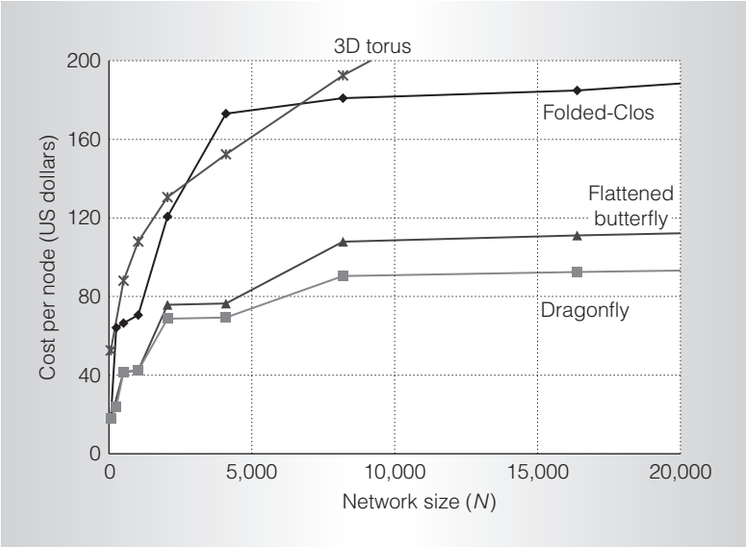

Figure 5. Cost comparison of alternative topologies.

Over time, interconnection networks will become more critical to system performance, and their size will continue to increase. Thus, high-radix routers and networks will become even more significant. Our work on the dragonfly topology is relevant to the networks used in all types of large-scale systems—server clusters, Internet routers, storage area networks, and supercomputers.

Our work will also be particularly relevant for data centers. Most computer architecture research, in both academia and industry, has focused on processor architecture and, more recently, multicore architectures. However, with the increasing importance of large-scale Internet services and the large-scale systems required for their support, computer architects must also focus on ''warehouse sized computing systems, made up of thousands of computing nodes, [and] their associated storage hierarchy and interconnection infrastructure.''[11,12] Studies show that a data center's capital cost is matched by the energy (cooling) cost within the first three years of purchase. Thus, having a cost- and energy-efficient topology and interconnection network such as the dragonfly will be critical in future data centers. In addition, data center networks will require a highly scalable topology to accommodate their increasing number of terminals (or nodes). Through its use of virtual routers, the dragonfly topology provides this scalability. Ultimately, we expect to see the dragonfly topology and variations of it employed widely in future large-scale systems. MICRO

References

1. W.J. Dally, ''Performance Analysis of k-ary *n*cube Interconnection Networks,'' *IEEE Trans. Computers, vol. 39, no. 6, June 1990, pp. 775-785.*
2. A. Agarwal, ''Limits on Interconnection Network Performance,'' *IEEE Trans. Parallel*

Distributed Systems, vol. 2, no. 4, Oct. 1991, pp. 398-412.
3. J. Kim et al., ''Microarchitecture of a High-Radix Router,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 05), IEEE CS Press, 2005, pp. 420-431.
4. J. Kim, W.J. Dally, and D. Abts, ''Flattened Butterfly: A Cost-Efficient Topology for High-Radix Networks,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 07), ACM Press, 2007, pp. 126-137.
5. J. Kim et al., ''Technology-Driven, Highly-Scalable Dragonfly Topology,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 08), IEEE CS Press, 2008, pp. 77-88.
6. S. Scott et al., ''The Black Widow High-Radix Clos Network,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 06), IEEE CS Press, 2006, pp. 16-28.
7. A.K. Gupta et al., ''Scalable Opto-Electronic Network (SOENET),'' *Proc. 10th Symp. High-Performance Interconnects* (HOTI 02), IEEE CS Press, 2002, pp. 71-76.
8. Luxtera, ''Fiber Will Displace Copper Sooner Than You Think,'' white paper, 2005; http://www.luxtera.com/white-papers.html.
9. L.G. Valiant, ''A Scheme for Fast Parallel Communication,'' *SIAM J. Computing,* vol. 11, no. 2, 1982, pp. 350-361.
10. A. Singh, ''Load-Balanced Routing in Interconnection Networks,'' PhD thesis, Dept. of Electrical Engineering, Stanford Univ., 2005; http://cva.stanford.edu/publications/2005/thesis_arjuns.pdf.
11. L.A. Barroso, J. Dean, and U. Hölzle, ''Web Search for a Planet: The Google Cluster Architecture,'' *IEEE Micro,* vol. 23, no. 2, Mar./Apr. 2003, pp. 22-28.
12. X. Fan, W.-D. Weber, and L.A. Barroso, ''Power Provisioning for a Warehouse-Sized Computer,'' *Proc. Int'l Symp. Computer Architecture* (ISCA 07), ACM Press, 2007, pp. 13-23.

John Kim is a research assistant professor at Northwestern University. His research interests include computer architecture and interconnection networks. Kim has a PhD in electrical engineering from Stanford University. He's a member of the IEEE and the ACM.

William Dally is the Willard R. and Inez Kerr Bell Professor of Engineering and the chair of the Department of Computer Science at Stanford University. He's also cofounder, chair, and chief scientist of Stream Processors. His research interests include computer architecture, network architecture, and programming systems. Dally has a PhD in computer science from the California Institute of Technology. He's a Fellow of the IEEE, the ACM, and the American Academy of Arts and Sciences.

Steve Scott is the chief technology officer at Cray. His research interests include processor and system architecture and interconnection networks. Scott has a PhD in computer architecture from the University of Wisconsin. He's a member of the IEEE and ACM.

Dennis Abts is a member of the technical staff at Google, where he's a technical lead for a next-generation large-scale network. His research interests include parallel computer architecture, interconnection networks, memory system design, robust system design, and fault tolerance. Abts has a PhD in computer science from the University of Minnesota. He's a member of the IEEE, ACM, and IEEE Computer Society.

Direct questions and comments about this article to John Kim at Northwestern Univ., 2145 Sheridan Rd., Evanston, IL 60208; jjk12@northwestern.edu.