HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks

Jung Ho Ahn, Nathan Binkert, Al Davis, Moray McLaren, and Robert S. Schreiber
HP Labs
{jung-ho.ahn, binkert, ald, moray.mclaren, rob.schreiber}@hp.com

ABSTRACT

In the push to achieve exascale performance, systems will grow to over 100,000 sockets, as growing cores-per-socket and improved single-core performance provide only part of the speedup needed. These systems will need affordable interconnect structures that scale to this level. To meet the need, we consider an extension of the hypercube and flattened butterfly topologies, the HyperX, and give an adaptive routing algorithm, DAL. HyperX takes advantage of high-radix switch components that integrated photonics will make available. Our main contributions include a formal descriptive framework, enabling a search method that finds optimal HyperX configurations; DAL; and a low cost packaging strategy for an exascale HyperX. Simulations show that HyperX can provide performance as good as a folded Clos, with fewer switches. We also describe a HyperX packaging scheme that reduces system cost. Our analysis of efficiency, performance, and packaging demonstrates that the HyperX is a strong competitor for exascale networks.

1. INTRODUCTION

Every 11 years since 1976, supercomputer performance has increased by a factor of 1000, reaching petaflop performance in 2008 [1]. By 2019, Moore's law is expected to provide a 160-fold increase in transistor density. While single-thread performance is rising only modestly, we expect a commensurate improvement in peak per-socket performance due to increased core count coupled with advances in memory technology to provide the necessary memory bandwidth to maintain system balance. This implies that at least a sixfold increase in the number of sockets will be needed to achieve exaflop performance. In 2008, the first two petascale systems were delivered: IBM's 18,802 socket Roadrunner [10], and Cray's Jaguar [14] comprising approximately 38,000 sockets. Using these machines as a rough guide, we can expect the exascale systems of 2019 to require 100,000–200,000 compute sockets. At this scale, the interconnection networks in these systems will have a major impact on their cost, performance, power, fault tolerance, and complexity. A fundamental cost driver of any interconnection network is the topology and its effect on packaging and cabling. Performance is heavily influenced by the topology, switches, and routing method.

We define radix to be the number of bidirectional ports that the switch supports, where a port consists of separate unidirectional input and output channels. Previously, limited chip-edge bandwidth has forced a tradeoff in switch chips between radix and port bandwidth. Given the ITRS [18] prediction that pin count and per pin bandwidth will remain relatively flat in the next decade, it is likely that this tradeoff will persist for electrical interconnects. Optical technology is promising in that: 1) bit transport energy is independent of length for link lengths of interest in an exascale system, 2) dense wave division multiplexing can be employed to improve bandwidth per port, and 3) optical cables and connectors offer high interconnect densities. These advantages have already been exploited in long haul networking and in the global interconnect for both the Roadrunner and Jaguar petascale systems. Silicon nanophotonics may permit direct optical connection to the switch chip, breaking the chip IO-bandwidth barrier and significantly changing the tradeoffs in network design [9, 20]. We therefore focus on networks based on high-radix switches.

Previous work has shown that high-radix switches can help achieve high bandwidth and low latency at reasonable power [8]. High-radix switches with high bandwidth links cut the number of routing chips while maintaining high network bandwidth, and reduce latency due to reduced hop count. Supercomputers have often used a folded Clos (also called a fat tree) network topology. Kim, Dally, and Abts [6] showed, however, that when high-radix switches are available, the flattened butterfly is more cost effective than the folded Clos.

We improve on this result in several ways. First, we introduce (Section 3) a class of n-dimensional networks that we call HyperX. In a HyperX, each switch is connected to all of its peers in each dimension and the number of switches in each dimension can be different. Second, we present a formal algebraic framework that describes the HyperX topology and use it to develop a search procedure that finds a least-cost HyperX configuration satisfying constraints on bisection bandwidth, network size, and switch radix. Third, we introduce (Section 3.3) a new adaptive routing algorithm (DAL) for HyperX networks. Simulation shows that DAL takes better advantage than existing flattened butterfly routing of the path diversity that the HyperX topology provides. We compare the performance of HyperX and folded Clos (Section 4), confirming that when high-radix switches are available, the HyperX is more cost effective. Finally we use the framework to select HyperX configuration parame-

SC09 November 14-20, 2009, Portland, Oregon, USA

ters to significantly simplify packaging (Section 5) which improves fault tolerance (Section 6) and cost when compared to similar sized folded Clos networks.

HyperX can easily be viewed as a simple extension of either the flattened butterfly or the hypercube concepts[1], and HyperX certainly has similarities to both. Interestingly, both flattened butterfly and hypercube topologies can be expressed in our formal framework and therefore could be viewed as particular points in the HyperX topology space. In Section 3.1.2 we find that the general HyperX with fewest switches (and given network size and bisection bandwidth) is from 8 to 28 percent smaller than the best flattened butterfly. More importantly, we want to emphasize that the novel contribution of this paper is not the topology but rather the formalized framework, search procedure, DAL, and packaging results.

2. FOLDED CLOS AND FLATTENED BUTTERFLY NETWORKS

This section provides an overview of folded Clos and flattened butterfly networks and their routing algorithms. Clos is the predominant high-radix topology in use in parallel computers today [2, 4, 10, 12]. Folded Clos networks can be constructed with switches of radix 4, but become more cost effective as switch radix increases due to reduced component count. Several routing strategies can exploit the path diversity of folded Clos networks. As switches have more ports, high-radix direct networks can be implemented. The flattened butterfly can be viewed as transformations of folded Clos networks. The resulting networks permit several routing strategies, including those derived from the folded Clos.

2.1 Network topologies

2.1.1 Folded Clos networks

Figure 1 illustrates a two-level folded Clos or fat tree [11] built from 8-port switches. In general, an L-level folded Clos based on a radix R switch consists of a core layer having $(R/2)^{L-1}$ downward-facing switches each having R ports. There are $2(R/2)^L$ links connecting adjacent layers. The lower layers can be seen, hierarchically, as consisting of exactly R separate subnetworks, each of which is a virtual crossbar having $(R/2)^{L-1}$ ports. These are perfect-shuffle connected to the core layer. Let N, the *network size*, denote the number of terminals in the network, and P denote the total number of switches. With an R-port switch, an L-level folded Clos has: $N = 2 \times (R/2)^L$ terminals; $(R/2)^{L-1}$ core switches; $P = (2L-1)(R/2)^{L-1}$ switches in total; and a maximum path length of $2L-1$ switches and $2L$ hops. Note that an arbitrarily large folded Clos network can be built from switches of fixed radix. The ratio of switches to terminals in a folded Clos is $P/N = (2L-1)/R$. Since the number of levels $L = O(\log N/\log R)$, increasing switch radix reduces switch count faster than linearly: $O(1/(R \log R))$.

The folded Clos is an *indirect* network: terminals are connected to leaf switches and packets can be routed through terminal-less internal switches. In a *direct* network, all switches connect to a number of terminals, and use their remaining ports to connect to other switches. The mesh and the torus are direct networks; a torus is symmetric—all switches are identical and identically located with respect to the network as a whole. The HyperX network is also a symmetric, direct network.

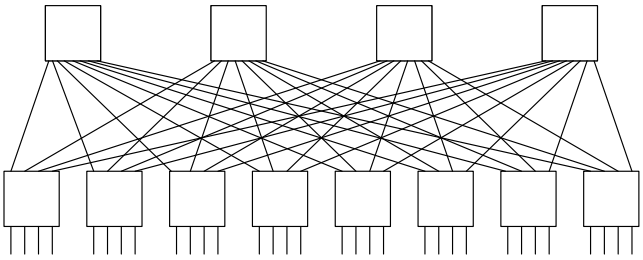


Figure 1: Folded Clos network $L = 2, R = 8$.

A folded Clos provides very high throughput even on difficult traffic patterns, and any permutation can be routed without blocking. Depending on their relative location, two given terminals, s and d, will have $a(s,d) = (R/2)^m$ lowest common ancestor switches, for some $0 \le m \le L-1$. The shortest paths connecting them use the unique up-link path to any of the lowest common ancestor switches, and then route the message on the unique down-link path to the destination. To route, the common ancestor can be chosen at random (with uniform probability) to balance the traffic load. Except at the edge level, any failed switch can be avoided without losing connectivity. Once the common ancestor is chosen, however, there is no residual capacity to adapt to congestion or faults.

Although a folded Clos is nonblocking for any permutation, global knowledge is required to allocate a set of link-disjoint routes; this is impractical in a large-scale network. With practical routing methods that lack global information, packets can be delayed at internal switches by contention for switch ports and links. Folded Clos networks are deadlock free, since they contain no cycles.

The plethora of paths in a folded Clos suggests that by modifying the topology and the routing strategy to reduce the number of alternate paths, a more efficient network can be constructed. The flattened butterfly [6] network is an example. For large networks, a folded Clos that provides full bandwidth between any two levels is expensive and may not be necessary. Designers save cost by *tapering* the link count in the upper levels of the hierarchy [15].

2.1.2 Flattened butterflies

As shown in Figure 2, a flattened butterfly can be derived from a butterfly network by mapping switches from successive layers onto a single physical switch [6]. The figure shows a 4-ary, 2-fly: there are 4 inputs and outputs per switch, and 2 switch levels. An m-ary, n-fly, with m^n input terminals at the bottom and the same number of output terminals at the top, consists of n levels with butterfly interconnects: at level ℓ, where $\ell = 0, 1, \ldots, (n-1)$, nodes are connected to others at distances that are multiples of m^ℓ. To flatten a $2-fly$, each switch in the lower layer is first associated with the switch directly above it, as shown in Figure 2(b). A 2-fly collapses to an all-to-all network, with each switch connecting to all of its peers using bidirectional links. In

[1]We thank the program committee, in particular K. Scott Hemmert, for pointing out that several HyperX aspects that we previously viewed as unique have been discussed in the context of flattened butterfly networks even though this discussion has not appeared in a citeable forum.

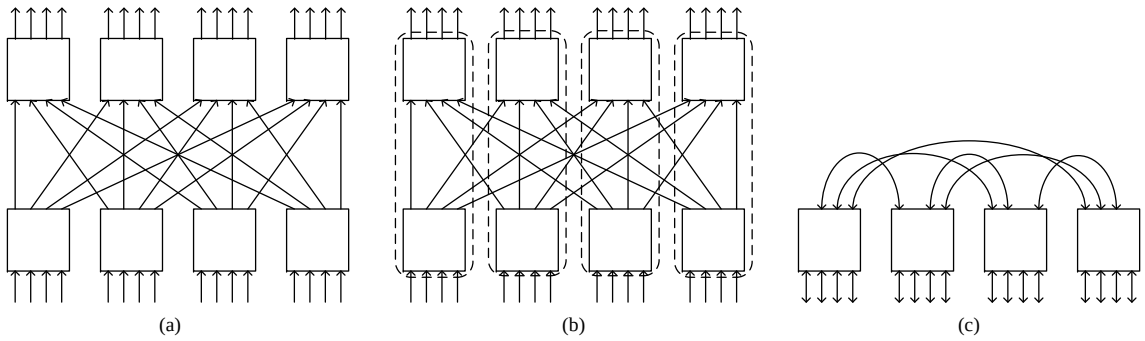


Figure 2: Flattening a butterfly network. (a) A 2-ary 4-fly, with unidirectional links. (b) Switches in the same vertical position will be coalasced. (c) The 4-ary 2-flat. Interswitch links are bidirectional.

general, when an m-ary, n-fly is flattened, the resulting network is identical to an $n-1$ dimensional array of switches, with extent m in each dimension; each switch connects to m terminals; and each switch is connected to each of the $m-1$ other switches that align with it in each of the dimensions, hence $(n-1)(m-1)$ neighboring switches. In order to maintain full bisection bandwidth and preserve the ability to route any permutation without contention, the intra-switch links require twice the bandwidth of the terminal links. The number of shortest paths in the flattened butterfly is at most $(n-1)!$. Since $n = O(\log N)$, we expect that n will be a small integer in practice. The resulting path diversity, considering shortest paths only, is far less than in the folded Clos.

2.2 Routing algorithms for the folded Clos

We use the term router to mean the logic that determines the output port taken by a packet arriving at an intermediate switch. *Deterministic* routers choose a fixed route between any source-destination pair. *Oblivious* routers choose paths dynamically but the choice is not based on network load. *Adaptive* routers choose a path based on network load in an attempt to avoid congested regions of the network. Deterministic routing leads to hot spots and poor throughput under certain traffic conditions [3]. Adaptive routing on a per packet basis can cause out of order delivery within a flow of packets between two terminals, which must be handled by the end to end packet protocols.

A folded Clos may be routed deterministically, obliviously, or adaptively. Oblivious routing sends each packet via a different randomly chosen common ancestor to distribute network load over all of the shortest paths. Adaptive routing selects paths based on buffer occupancy on the candidate paths. The advantages of adaptive folded Clos routing and its adequate implementation are discussed in [5]. We will describe routing options for the flattened butterfly in Section 3.3.

3. THE HYPERX TOPOLOGY

A HyperX is a direct network of switches in which each switch is connected to some fixed number T of terminals. A terminal can be a compute node, cluster of compute nodes, I/O node, or any other interconnected device. The switches are viewed as points in an L-dimensional integer lattice. Each switch is identified by a coordinate vector, or multi-index $I \equiv (I_1, \ldots, I_L)$ where $0 \leq I_k < S_k$ for each $k = 1..L$. In each dimension, the switches are fully connected. Thus, there are bidirectional links from each switch to exactly $\sum_{k=1}^{L}(S_k - 1)$ other switches: a switch connects to all others whose multi-index is the same in all but one coordinate. The number P of switches in the HyperX satisfies $P = \prod_{k=1}^{L} S_k$.

In a simple HyperX all links have uniform bandwidth. The topology can be generalized by allowing the link bandwidths to be multiples of some unit of bandwidth, to model the option of trunking of multiple physical layer links. This flexibility can be exploited to provide uniform bandwidth between dimensions with different values of S, and to allow different bandwidths between terminal and intra-switch links. We let $K \equiv (K_1, \ldots, K_L)$ represent the relative link bandwidths in each of the dimensions, where the unit of bandwidth (conceptually offered by one hardware link and one switch port) is the bandwidth of the terminal-to-switch connections. A *regular* HyperX is one for which $S_k = S$ and $K_k = K$ for all $k = 1..L$. Thus, a regular HyperX is determined by the parameters L, S, K, and T and we shall refer to it as a regular (L, S, K, T) HyperX.

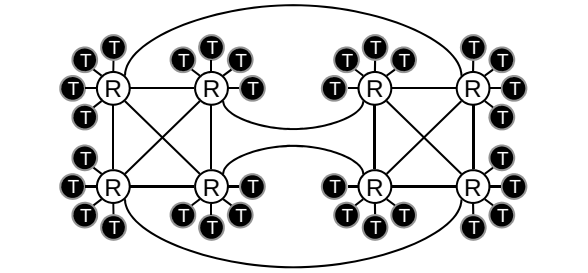
Figure 3 shows two examples of the HyperX topology. There are 32 terminals shown in Figure 3(a) where 4 terminals are attached per switch. Switches are organized in two dimensions. There are two switches in the first dimension, and four switches in the second dimension creating an irregular HyperX. The regular HyperX in Figure 3(b) also has 4 terminals per switch and two switch dimensions, but each dimension consists of three switches and supports 36 terminals.

Note that a hypercube is a regular HyperX with $(S = 2, K = 1, T = 1)$. A fully connected graph is a HyperX with $L = 1$. We can also describe the flattened butterfly topology as a regular HyperX with $T = S$ and either $K = 1$ or for full bisection bandwidth, $K = 2$. The topology of the YARC high-radix switch [17] is $(L = 2, S = 8, K = 1, T = 1)$.

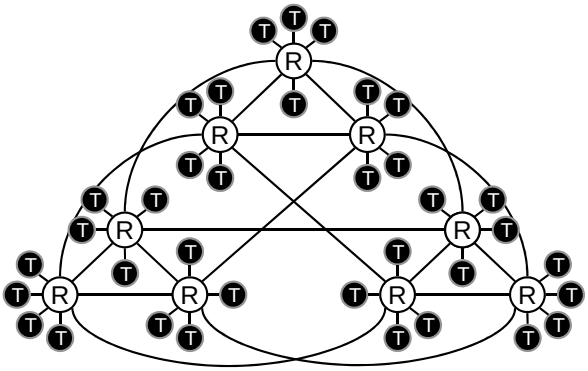
The bisection bandwidth of a HyperX is realized by cutting one of its dimensions in half. The channel bisection of such a cut is

$$C_m \equiv (1/4)\ K_m S_m \prod_{k=1}^{L} S_k = (P/4)\ K_m S_m \qquad (1)$$

if dimension m is bisected. If $K_m S_m$ is smallest among all the dimensions of the HyperX, so that $K_m S_m \leq K_k S_k$ for all $k = 1, \ldots, L$, then (1) determines the bisection bandwidth.



(a) $L = 2, S_1 = 2, S_2 = 4, K = 1, T = 4$



(b) $L = 2, S_1 = 3, S_2 = 3, K = 1, T = 4$

Figure 3: Two examples of the HyperX topology. Dark circles (marked T) are terminals, light circles (marked R) are switches.

A HyperX network needs at least $TP/2$ bidirectional links crossing any bisection in order to be nonblocking. Thus, the ratio $\beta = K_m S_m / 2T$ measures the relative bisection bandwidth of the architecture. As discussed above, for a regular $(L, S, 2, S)$ HyperX, *i.e.* a flattened butterfly with double-wide switch-switch links, we have $\beta = 1$.

3.1 Achievable HyperX networks

We consider the problem of finding a best possible, by some criterion, HyperX satisfying switch radix, network size, and bisection bandwidth requirements. HyperX is a direct network; for switches with a given radix R we must therefore conform to the bound:

$$T + \sum_{k=1}^{L} K_k (S_k - 1) \leq R \,. \tag{2}$$

To create a system consisting of N terminal nodes, we need at least this many terminal links. With a total of $P = \prod_{k=1}^{L} S_k$ switches, each having T terminal links, this constraint becomes

$$TP = T \left(\prod_{k=1}^{L} S_k \right) \geq N \,. \tag{3}$$

With both R and N viewed as given, fixed constants, these equations provide both an upper bound (2) and a lower bound (3) on T for each possible network shape S. Werequire that there be enough bisection bandwidth. For some specified minimum relative bandwidth B we require that

$$\beta \equiv \frac{\min(K_k S_k)}{2T} \geq B \,. \tag{4}$$

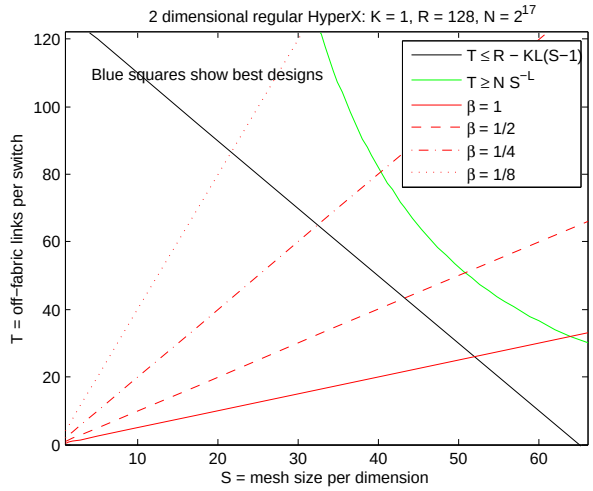


Figure 4: HyperX Design Space ($L = 2$, $K = 1$, $R = 128$, $N = 2^{17}$). No solutions: the lower bound exceeds the upper bound on T for all S.

Each dimension must consist of at least 2 switches:

$$S_k \geq 2 \,, \;\; k = 1, 2, \ldots, L \,. \tag{5}$$

Our objective is to find HyperX networks that satisfy the constraints above. Among the feasible networks, we prefer those with least cost, which to a first approximation is proportional to the number of switches, $\prod_{k=1}^{L} S_k$. A low-dimensional solution (with small L) is also good, since it decreases the average hop count and yields more easily realized hardware implementations.

In contrast to the folded Clos, with switches of a given radix one cannot build an arbitrarily large HyperX. In fact, the largest HyperX network buildable with radix R switches is an $(R-1)$-cube with one terminal per switch, so clearly we must have $N \leq 2^{R-1}$. As N approaches this limit, the smallest feasible L grows, making these HyperX topologies unattractive; the study in Section 4.2 bears this out. For high-radix switches, fortunately, the HyperX designs achievable in three or at most four dimensions turn out to be large enough for the exascale systems we have in mind.

3.1.1 *The space of regular HyperX networks*

Since it is determined by fewer parameters, we can visualize the regular HyperX design space more easily than the general. In a regular HyperX, the network shape S and the trunking factor K are scalars.

Figures 4 and 5 show the design space of regular HyperX for a baseline example with $R = 128$ (typical of the high-radix switches we expect in the near future) and $N =$ 128K (a network size consistent with exascale performance). There are four free design space parameters, namely L, K, S, and T. We choose to fix the trunking factor K and dimension L and think about the resulting (S, T) space. Hence, we are seeking integer (S, T) pairs in the region bounded on the left by the vertical line $S = 2$; below by the nonlinear network size bound (3), and above by the linear bandwidth (4) and switch port (2) bounds that have opposite slopes and form a roof over the allowable space. For some (R, N, K, L) combinations, the lower bound lies completely above these two upper bounds, and there are no feasible HyperX de-

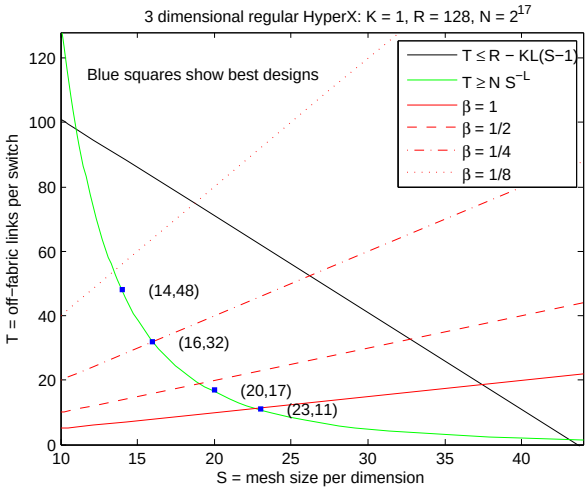


Figure 5: HyperX Design Space ($L = 3$, $K = 1$, $R = 128$, $N = 2^{17}$).

signs (Figure 4). For others, the region of allowed designs is bounded by these three constraints, having a leftmost corner, a topmost corner, and a rightmost corner (Figure 5). The leftmost is the most interesting, as it is the point of least switch count. These points (for several values of β) are shown by blue squares and labeled with their (T, S) values.

3.1.2 *Selecting an optimum HyperX: An algorithm*

We have implemented an algorithm to find a general HyperX meeting the requirements of Section 3.1 and having fewest switches. Given the parameters N, R, and β of the design problem, our algorithm chooses a design specified by the dimension, L; the number of terminals per switch, T; the shape of the network, S (an L-vector); and the trunking factors, K (another L-vector); so that the network size (2), switch port (3), and bisection bandwidth (4) inequalities are satisfied and the switch count is smallest possible. We do this by an exhaustive search of the design space, but prune away designs early that are known to be either infeasible or suboptimal. The runtime of the algorithm for practical, large cases is a few seconds.

To restrict the design space to be searched, we exploit certain symmetries. First, we lose nothing by restricting the search to designs for which

$$S_1 \le S_2 \le \cdots \le S_L. \tag{6}$$

Since the bisection bound depends on the smallest of the $S_k K_k$, we also can assume that

$$K_1 \ge K_2 \ge \cdots \ge K_L\ . \tag{7}$$

Once a feasible design is found, its switch count is an upper bound on the best achievable design. Any other design that cannot be better will be rejected. Let O represent the smallest switch count for all feasible designs so far discovered. We add the inequality

$$\prod_{k=1}^{L} S_k \le O \tag{8}$$

to further restrict the set of designs to be searched. In particular, the network size constraint tells us that if we are seeking to find configurations that reduce O, we can restrict our attention to designs for which T is large enough to achieve the desired network size given that the number of switches in an improved design is going to be less than O; we therefore restrict the search values of T to those that satisfy

$$N \le T(\prod_{k=1}^{L} S_k) \le TO \quad \Rightarrow \quad T \ge \lceil N/O \rceil \equiv T_{\min}\,. \tag{9}$$

As for the range of dimensions to search, we have $1 \le L \le R-1$, the upper bound being achieved for a hypercube.

3.2 Comparing general and regular HyperX

We can ask whether or not an irregular HyperX can be significantly more attractive than a best possible regular HyperX or flattened butterfly. To that end, we conducted an experiment. For typical values of the design problem parameters ($N = 2^{17}$, $R = 2^7$, and $B \in \{0.125, 0.25, 0.5, 1\}$) we found best (having fewest switches) regular and general HyperX designs. Table 1 shows the results.

The irregular HyperX has fewer switches in all cases, the reduction in number ranging from 8 to 28 percent. Interestingly, the best results were always three dimensional in the general case, whereas the best regular designs were four dimensional in two cases. Flattened butterflies are regular HyperX networks for which $K = 2$. If we add this restriction then for the full bisection-bandwidth case ($\beta = 1$) we get an inferior design ($L = 4$, $S = 11$, $K = 2$, $T = 9$) having 14,641 switches. If we add the additional restriction imposed by the flattened butterfly, namely $T = S$, then we have a network with $S = 11$ and size $N = 11^5 = 161,051$ with 29,979 more terminal ports than needed.

3.3 Routing algorithms for the HyperX

The set of shortest paths between two switches in a HyperX is determined by their coordinate vectors. Consider paths from switch zero (all its coordinates are 0) to switch $I = (I_1, \ldots, I_L)$. The length of each shortest path is equal to the number of nonzero elements of I. Let Δ be the set dimensions k for which $I_k \ne 0$. We call these the *offset* dimensions; the others are *aligned* dimensions. Every shortest path goes directly (by one step) from the source 0 to the destination in each of the offset dimensions in some sequence. If the number of offset dimensions (i.e. the cardinality of Δ) is D, then there are $D!$ shortest paths.

A minimal deterministic routing method is obvious: route to the destination by the shortest path determined by moving in the dimensions of Δ in a fixed order; without loss of generality, in order of increasing dimension number. This is known as dimension-order routing. Minimal adaptive routing is possible. When one dimension is blocked due to buffer congestion at the downstream switch, we choose another offset dimension. We call this Min-AD. Min-AD adaptively explores the space of $D!$ shortest paths. An oblivious routing algorithm, inspired by Valiant [19], routes all packets through an intermediate node chosen at random with uniform probability.

As shown by Kim, Dally, and Abts [6], adaptive non-minimal routing is in general better than either deterministic or oblivious routing for a flattened butterfly, and this is true for the HyperX. They proposed the adaptive Clos-AD algorithm, which chooses, at a packet's source node, between dimension ordered minimal routing and non-minimal rout-

B	Best Regular	Switch Count	Best General	Switch Count
0.125	$L=4,\ S=7,\ K=2,\ T=55$	2401	$S=(5,19,19),\ K=(4,1,1),\ T=76$	1805
0.25	$L=3,\ S=14,\ K=2,\ T=48$	2744	$S=(3,27,30),\ K=(9,1,1),\ T=54$	2430
0.5	$L=3,\ S=16,\ K=2,\ T=32$	4096	$S=(3,35,36),\ K=(12,1,1),\ T=35$	3780
1.0	$L=4,\ S=10,\ K=3,\ T=14$	10000	$S=(5,38,38),\ K=(8,1,1),\ T=19$	7220

Table 1: Best regular and general HyperX networks for $N=2^{17}$ and $R=128$.

ing, based on estimates of queuing delay. If it chooses to route non-minimally, Clos-AD routes to a randomly chosen lowest common ancestor switch, considering the network as a transformed folded Clos network. The packet therefore takes dimension ordered minimal paths from source to intermediate and from intermediate to destination nodes. The decision is made at the source node, and no packet can be derouted (take a non-minimal route) twice.

We argue that Clos-AD can be improved in two respects. First, by deciding to route minimally or not only at the source, it loses significant ability to adapt to congestion encountered *en route*. Second, by making a choice of intermediate node that is informed by a mapping of the folded Clos into the HyperX, it introduces dimensional asymmetry which has no apparent advantage due to the inherent symmetry in HyperX dimensions. To address the issues raised above, we propose the DAL (Dimensionally-Adaptive, Load-balanced) routing algorithm:

1. Mark all offset dimensions as deroutable (unmarked) on creation of the packet at its source. On arrival at a switch:

2. Find an offset dimension with an unblocked path to an aligned switch in that dimension. If none exist, then:

3. Find an unmarked offset dimension and unblocked switch that is offset from the destination and if one exists then route and mark the dimension as no-longer-deroutable; else

4. Push the packet into a minimal, dimension-order, deterministic routed virtual channel.

Dimension-order routing, and therefore DAL, is deadlock free; the virtual channel used as a last resort is used to prevent deadlock.

Unlike Clos-AD, DAL deroutes in one of the HyperX dimensions at a time, treating each HyperX dimension independently and symmetrically. This tends to cause it to take shorter adaptive paths. While it may deroute just once per dimension, DAL can deroute as many times as there are offset dimensions. Unlike Clos-AD, DAL routes a packet only through the subnetwork of those nodes that are aligned with both the source and the destination. Once a packet comes into alignment with the destination in a certain dimension it remains there. Unlike Clos-AD, DAL may deroute a packet at each hop on its route, thereby adapting to congestion not visible at the source node.

4. EVALUATING HYPERX

We evaluate DAL routing for the HyperX topology and the DAL/HyperX combination in comparison to the folded Clos. We begin by showing simulation results that explore the performance of various routing algorithms on HyperX. They show that DAL is more effective than adaptive minimal, Valiant, and Clos-AD routing. Further simulations show that HyperX with DAL is generally better than an adaptively routed folded Clos with similar system configurations. We then analytically compare the number of switches in optimum HyperX and folded Clos topologies across a range of system parameters. For large networks built with radix 32 switches, folded Clos may be best, but HyperX is more cost effective when higher radix switches are used, even for million-node networks.

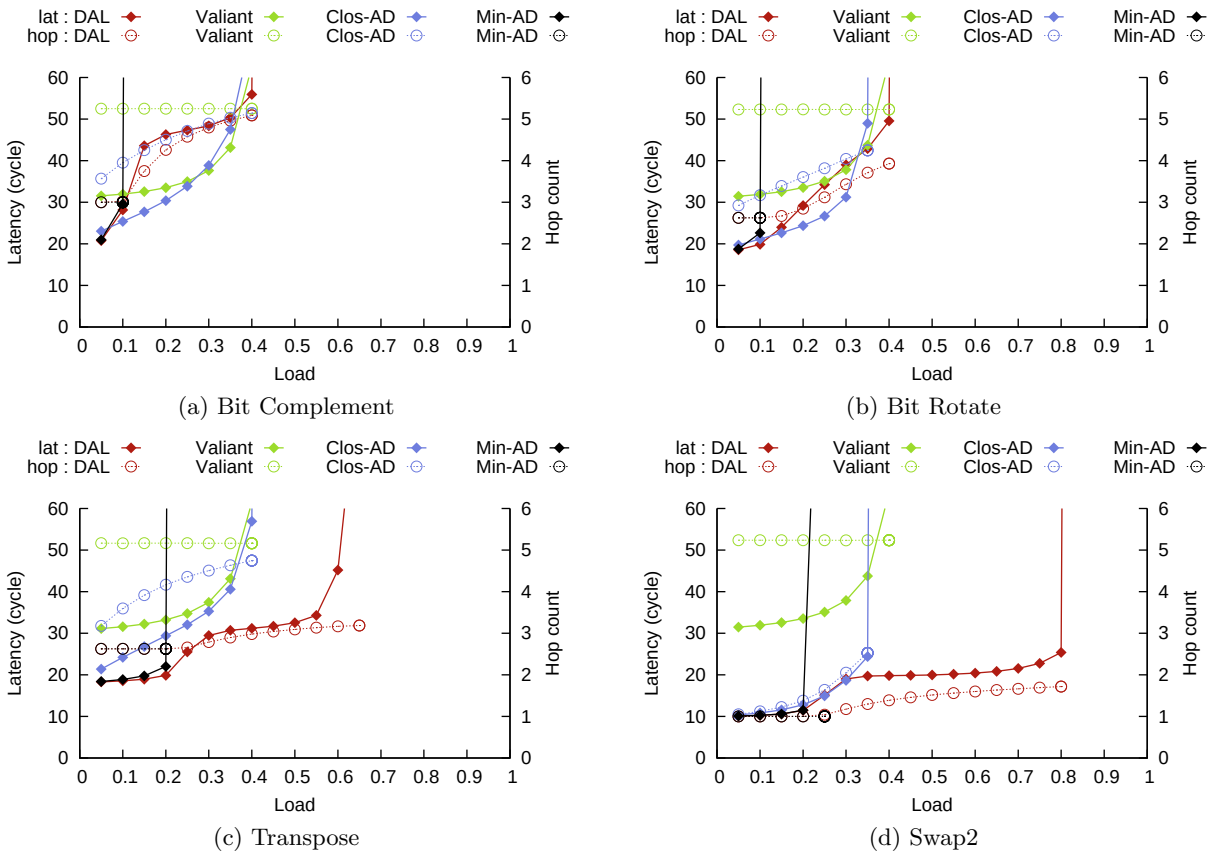
4.1 Performance results

We use a cycle-accurate simulator on a variety of synthetic traffic patterns. In all cases, switches have a four cycle delay and channels have a one cycle channel transmission delay, where we define a cycle as the time for a switch to send or receive a flit via one port. We define latency as the difference between the time a packet is received at a destination terminal and the time this packet is generated at the source terminal. We assume single-flit packets. If a packet makes four hops through the network from source to destination, it will transit 4 channels and 3 switches in 16 cycles. For simulation, sequential allocation is used for adaptive route computation [5]. For HyperX, we assume 6 virtual channels (VCs) per port: each port has 3 sets of 2 VCs. Cut-through flow control is used. One VC in each set is reserved for deadlock avoidance in the adaptive minimal and DAL routing algorithms. Each VC can hold up to 32 flits. Switches are input-queued; the iSLIP algorithm [13] is used for switch allocation. A crossbar has input speedup of 2.

Figure 6 presents a comparison of the routing algorithms (described in Section 3.3) on a typical HyperX (regular, with $T=8$, $S=8$, and $K=1$) with 32-port switches, 4096 terminals, 512 switches, and where β is 0.5. We present three standard traffic patterns: Bit Complement; Bit Rotate; and Transpose [3], and one additional pattern, Swap2. Swap2 is a permutation designed to highlight the difference between Clos-AD and DAL. In this pattern, the even-numbered nodes exchange messages with a peer across the bisection in one dimension and the odd-numbered nodes exchange messages with a peer across the bisection in another dimension. These two offset dimensions correspond to the top two levels of the folded Clos that underlies Clos-AD.

Valiant routing [19] reliably achieves throughput of β: about half of all packets traverse any given bisection, twice, when routed via a random intermediate node (independent of traffic pattern). This throughput certainty comes at the price of higher latency for light traffic loads due to the increased hop count. Min-AD achieves throughput of 2β on traffic patterns that do not cause congestion, such as uniform random, but throughput can degrade to $1/T$ on patterns that cause congestion at the source.

The adaptive algorithms, Clos-AD and DAL, are better choices. For low loads, both algorithms choose a minimal



(a) Bit Complement

(b) Bit Rotate

(c) Transpose

(d) Swap2

Figure 6: Load-latency graphs on the (a) Bit Complement, (b) Bit Rotate, (c) Transpose, and (d) Swap2 traffic patterns [3] of a regular HyperX network with $N = 4096$, $R = 32$, $\beta = 0.5$, $T = 8$, $S = 8$, $K = 1$.

route and achieve low latency, whereas at high load, they adapt, choosing nonminimal routes to avoid congestion. The primary difference between the two algorithms is when they decide to adapt. Clos-AD estimates, at the source, the delay of a minimal route and a non-minimal route and makes its adaptation decision eagerly. DAL makes decisions lazily as the packet traverses the network and decides on the route independently per dimension. Eager adaptation means that Clos-AD has less information which could be out of date, while lazy adaptation allows DAL to be far more nimble.

Clos-AD is usually better than Min-AD and Valiant, but for transpose some limitations become apparent. Clos-AD makes its adaptation decision at the source. With the limited information available, its delay estimate is inherently imperfect. The delay estimate will be biased towards minimal paths for some patterns and towards non-minimal paths for others. Our delay estimate tends to underestimate the cost of the non-minimal path. This bias is the reason for the higher average hop counts of Clos-AD at all load levels when compared to DAL. On the Bit Complement pattern, the terminals connected to a given switch communicate with terminals connected to a single other switch, a pattern which congests the links on the shortest paths between them. Moreover, these shortest paths span all the dimensions, so DAL will deroute most packets in each dimension. For these patterns, CLOS-AD has the advantage of making that deroute decision once and for all, early, and this improves latency for low to medium load. On the Bit Rotation pattern, this congestion occurs but on few links. So the average latency of DAL approaches that of CLOS-AD for medium load.

DAL has higher saturation throughput than the alternatives on all patterns that we have tested. This is because it will take the minimal path whenever possible, but route around congestion when necessary. Furthermore DAL assesses congestion at each hop and independently for each dimension. Our Swap2 pattern is a good example of why DAL excels. The theoretical throughput of this pattern should be 2β, which DAL is able to achieve. For this pattern, Clos-AD overlays a folded Clos network on top of the HyperX, and can therefore pick a route that goes through offset intermediate nodes in already aligned dimensions.

We follow our comparison of routing algorithms on HyperX by comparing HyperX networks using DAL to a folded Clos network using adaptive routing. Figure 7 presents the results of this comparison across the same set of traffic patterns. $N = 4096$ and $R = 32$ in all three networks and we use the same configuration as in Figure 6 for the HyperX with $\beta = 0.5$. The HyperX with $\beta = 0.25$ has $T = 14$, $S = 7$, $K = 1$ so that $P = 343$. For a tapered folded Clos, we define β as

$$\beta \equiv \frac{\text{number of links from core switches}}{\text{network size}}.$$

The folded Clos (with $\beta = 0.5$) needs three levels. There are 205 level 1 switches with 10 uplinks and 20 downlinks; 130

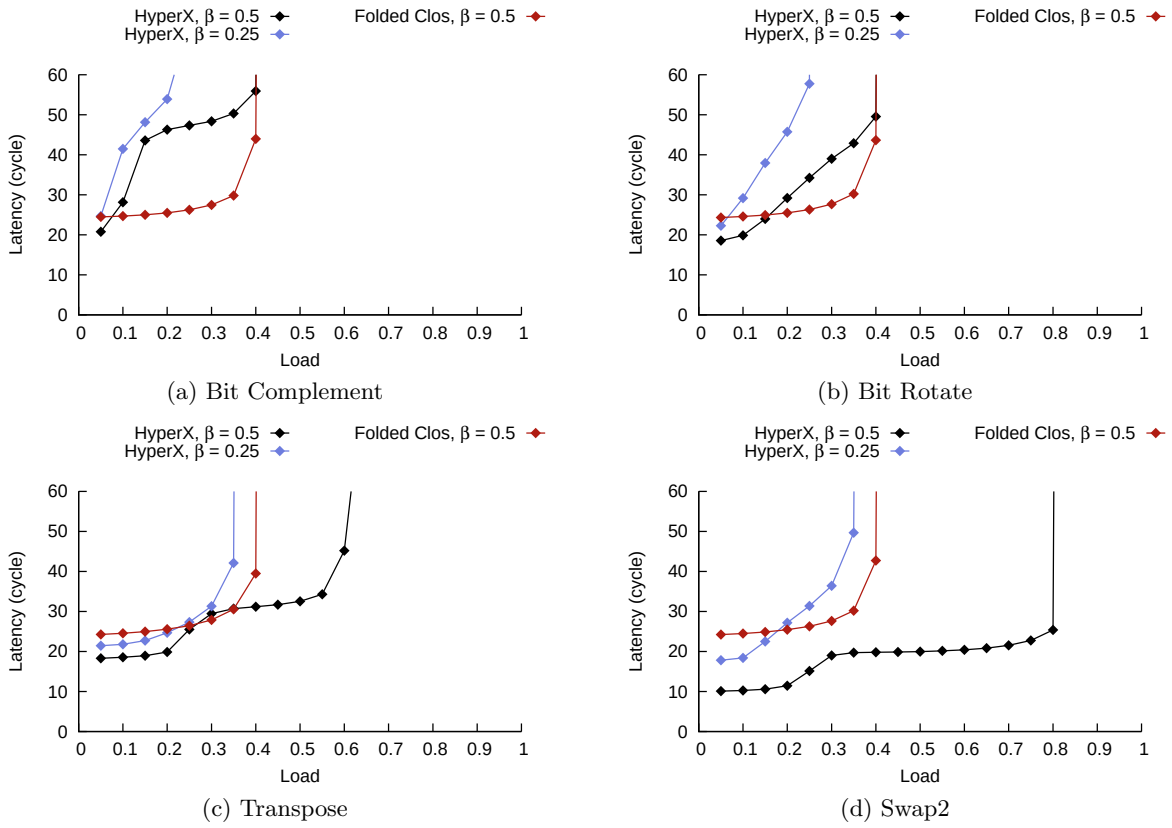


Figure 7: Load-latency graphs of HyperX networks using DAL routing compared with folded Clos networks using adaptive routing on the (a) Bit Complement, (b) Bit Rotate, (c) Transpose, and (d) Swap2 traffic patterns when $N = 4096$, $R = 32$, and β is varied.

level 2 switches that have 16 uplinks and 16 downlinks; and 80 level 3 switches with 13 downlinks, 2 of them packaged per physical switch so that $P = 415$. In all cases, the folded Clos achieves a saturation throughput of β because the topology effectively forces all patterns to behave as if they are routed by Valiant. For the same reason, the hop count on folded Clos is generally higher than HyperX with DAL as seen in the Valiant routing case of Figure 6. The benefit of HyperX with DAL routing is that it is able to achieve a load of 2β on patterns that don't exhibit congestion while maintaining a load of β on patterns that do.

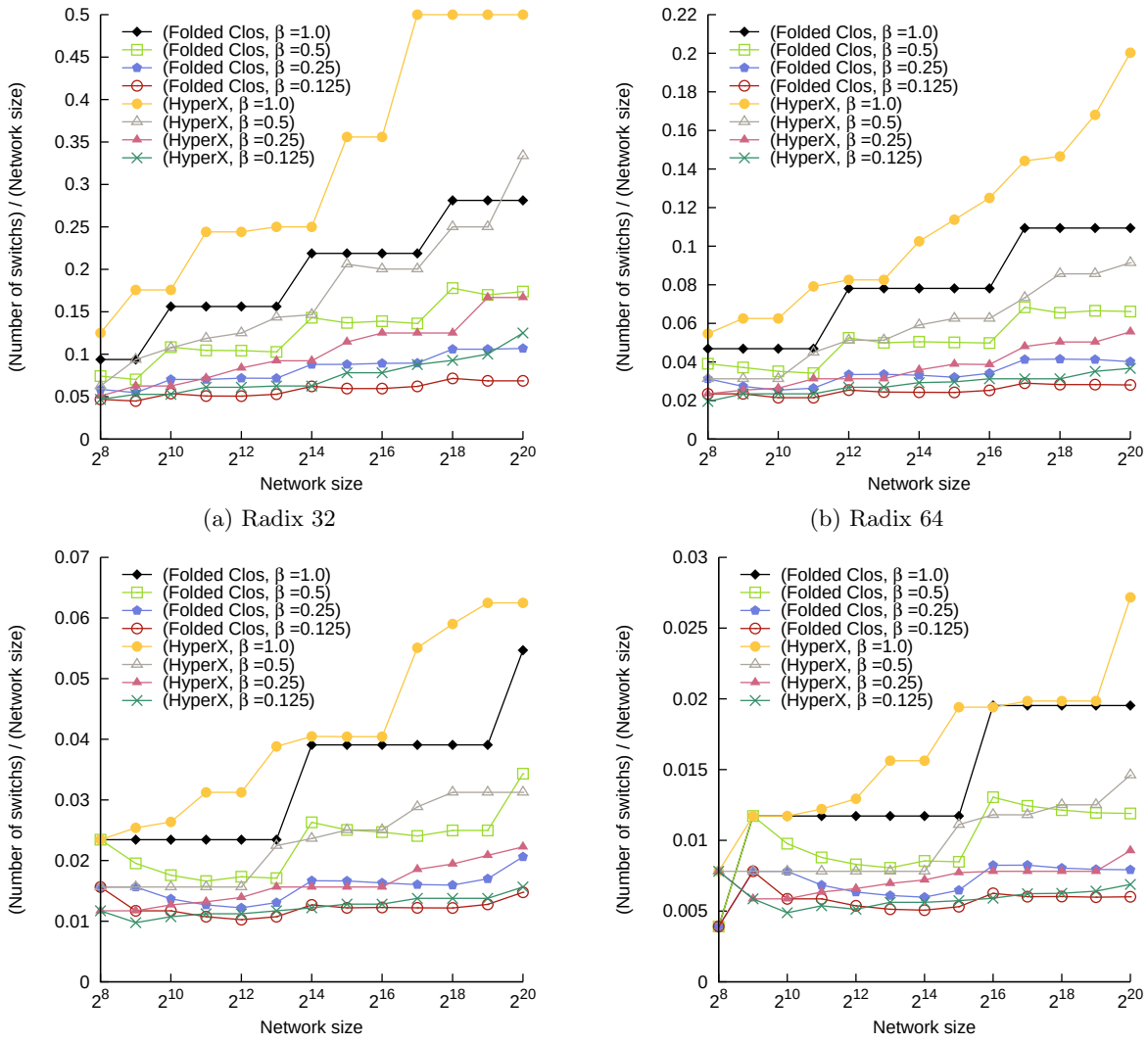
4.2 Resource comparison

The previous section shows that DAL routing on HyperX networks provides very low network latencies for a range of traffic patterns. The focus of this section is a cost comparison of the folded Clos topology and the HyperX topology for a wide range of topological parameters.

In Figure 8, we vary network size (N), switch radix (R), and tapering ratio (β). The vertical axis of the graphs depicts the number of switches divided by the number of terminals of the network. This measure can be thought of as an estimation of networking cost per terminal node. Steps in the graphs indicate that the dimension of the network increased. This is particularly important because an increase in dimension will result in an increase in cost for both HyperX and folded Clos. In HyperX networks, an increase in dimension effectively allows fewer terminal nodes to be connected to each switch resulting in an increase of the total number of switches to achieve a given network size. Folded Clos networks add another complete layer of switches to the network with each increase in dimension. In HyperX, because fewer terminal nodes can be connected to each switch as dimensions are added, the cost of each additional dimension is higher than a folded Clos where additional dimensions result in a constant cost increase. This addition is marginal at very high dimensions thus making folded Clos a better choice in this regime. Our data does indeed show that at low radices (such as radix 32) and high node counts, the cost of a folded Clos with $\beta = x$ is comparable with that of a HyperX with $\beta = 0.5x$. The advent of high-radix switches changes this story because very large networks can be built with few dimensions. In the radix 128 and 256 cases, the cost of a folded Clos with $\beta = x$ is comparable with that of HyperX with $\beta = x$ on many configurations. We have shown that a HyperX can, on many traffic patterns, handle a higher load (as much as double) than a folded Clos of equal bisection bandwidth; this makes a compelling case for HyperX.

5. PACKAGING HYPERX

System packaging is a central practical concern in network design, especially for systems that occupy warehouse sized machine rooms. A good mapping between the network topology and the packaging hierarchy greatly simplifies the



(a) Radix 32

(b) Radix 64

(c) Radix 128

(d) Radix 256

Figure 8: The number of switches while the network size is varied using switches with radix (a) 32, (b) 64, (c) 128, and (d) 256.

system wiring. For example an entire level of interconnect may be contained within a subsystem, minimizing the number of individual cables between subsystems. Depending on the topology, the number of cable assemblies can be further reduced by using cables with multiple parallel connections. This is particularly beneficial in optically connected systems where the number of parallel connections is not constrained by considerations of cable bulk or connector density. Parallel ribbon cables with up to 100 fibers have been demonstrated using MT connectors [16].

Minimizing the total cable length and using a regular structure that is easy to deploy are also key considerations. The Dragonfly [7] is an example of a system in which network and system packaging are co-designed to provide the most cost effective solution. The key difference between HyperX and Dragonfly is that HyperX assumes that all cables will likely be optical in the future, whereas Dragonfly assumes that intra-rack cables will be electrical (and cheaper) while inter-rack cables will be optical. As data rates increase and as optical cables become more commonplace, power and cost will likely favor optics even for intra-rack applications.

To minimize cabling in folded Clos networks, two packaging strategies are common. The first strategy embeds the switch hierarchy within the package hierarchy to exploit local interconnect within the enclosures. The second strategy combines multiple levels of switch components into a larger switch chassis. A 128K port folded Clos network using radix 128 switches requires three levels of switches. The first level can be integrated with groups of 64 processing elements. The top two levels can be combined to make 8192-port switch units.

In this arrangement only one stage of cabling is required, between the 2048, 64-node processing enclosures, and the 16, 8192-port switches. Each processor enclosure has 4 links to each of the switches. This requires a total of 32768 cables each with 8 fibers. There is a wide range of connection lengths, and the distribution of cable lengths is only known after detailed floor planning of the entire system.

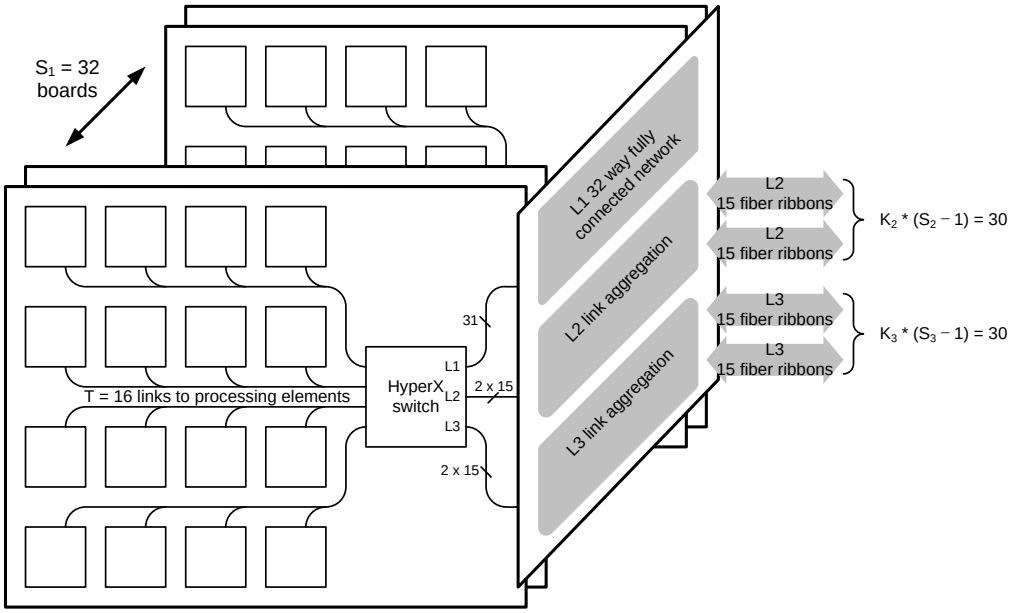


Figure 9: HyperX chassis showing wiring.

The flexibility of the HyperX topology allows a configuration to be selected such that the different dimensions of HyperX connectivity map directly onto the physical dimensions of the system. In the case of a 128K port system we can choose, for example, a three-dimensional HyperX with $T = 16$, $S_1 = 32$, $S_2 = S_3 = 16$, $K_1 = 1$, and $K_2 = K_3 = 2$. The first dimension is routed within the enclosure, and the two further dimensions correspond to the placement of the enclosure within a two-dimensional array in the machine room.

Figure 9 shows the connectivity of switches with the enclosure. As HyperX is a direct topology the switch can be located on the same card as the processing elements. Each board thus comprises 16 terminal nodes and one switch. The 32 boards in an enclosure communicate via an optical backplane that links all switches in the rack in a fully connected network, forming the first HyperX dimension. This connects $\binom{S_1}{2} = 496$ pairs of nodes using 992 waveguides or fibers. The optical backplane further aggregates the second and third dimension links to enable the use of parallel fiber ribbons for interconnect between enclosures. The aggregation networks require a further 1920 waveguides per level.

Figure 10 shows the placement of enclosures in a two-dimensional array. Each enclosure is connected to each of its peers in both dimensions. Each pair of enclosures is connected by two separate fiber ribbons for resiliency. The wiring therefore consists of many replicated instances of 16-way all-to-all wiring patterns. Along the mid-line, at every row or column of racks, there are 64 optical fiber ribbons connecting all 8 enclosures on one side to the 8 on the other.

Each set of all-to-all wiring between rows or columns of 16 enclosures requires 120 fiber ribbon cables, in 15 distinct lengths. This pattern is repeated 64 times, 32 times in the X dimension and 32 times in the Y dimension, giving a total of 7680, 64-way fiber ribbon cables. Assuming a pitch between enclosures of 1.2m in each dimension with an additional meter of overhead, cable lengths range from 2.2m to

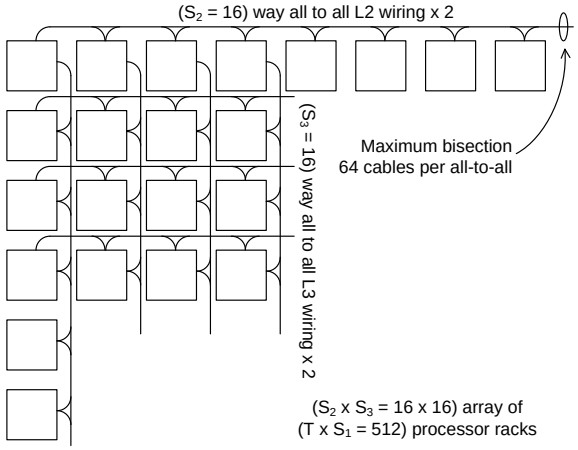


Figure 10: HyperX system floor plan.

19m. The total length of 64 way optical fiber ribbon is 60km. The direct correspondence between the HyperX dimensions and the physical dimensions of the enclosure arrays greatly simplifies the task of connecting the system. It also avoids wiring congestion points typical of folded Clos wiring at the core level.

6. FAULT TOLERANCE

Cables and connectors cause most of the failures in large scale networks. The move to optics will change the fault characteristics in some respects, but the dominant failure mechanism will remain individual links rather than switches. Timeout and retry mechanisms can be used to handle transient failures, however a network with half a million connections is likely to suffer hard link failures.

The massive path diversity in folded Clos networks creates the potential for a high degree of fault tolerance. Although

there are many alternate uplinks to common ancestors, the choice of downlink is entirely deterministic and will block at any failed link; with oblivious routing a packet can therefore get stuck at a failed link. In this case the packet must be retried from the source. In order to minimize performance degradation in the presence of failed links it is highly desirable to prohibit failed routes from being selected as part of the routing process.

Adaptive routing in folded Clos networks has the advantage that it can route around failed uplinks. Links with hard errors are simply eliminated from the possible routing alternates. In order to extend this to route around failed downlinks a mechanism is required which checks whether a given choice of uplink would cause a failed downlink to be selected in the down path. If this is the case an alternate uplink is selected. The mechanism for detecting potential downlink errors uses the packet's destination to look up a table of disallowed uplinks for this destination. The table size can be compressed by recognizing that due to the hierarchical topology contiguous ranges of destinations can be handled by a single entry.

HyperX networks offer more possibilities for routing around failed links, depending upon the routing algorithm being used. In the case of DAL routing, a packet can only be blocked by a single failed link when there is only one remaining offset dimension, and the deroute has already been taken in that dimension. A simple solution is to permit a further deroute. Provided there is only a single failed link within a fully connected subnetwork this will not livelock. In order to avoid selecting a nonminimal route that would lead to a failed link, the same technique of filters on random routes proposed for folded Clos networks can be applied. Under DAL routing, the failure information required to be held on each switch is more localized, since each switch need only be concerned with failed links attached to its peers in each dimension.

7. CONCLUSION

To design a network for an exascale system requires tradeoffs in performance, power, wiring complexity, and fault tolerance. In this paper, we investigated the impact on those tradeoffs of high-radix switches and the topologies that they enable.

We show that the HyperX topology is an attractive candidate for exascale networks because of its ability to achieve a favorable balance among these tradeoffs. We compare the HyperX to the folded Clos and show several design points where the HyperX uses significantly fewer routers to achieve similar bandwidth on benign patterns. For applications that demand full bisection bandwidth, the folded Clos is generally more efficient in terms of router components. We further investigate the routing algorithms possible on HyperX and compare our new DAL algorithm to other algorithms for the folded Clos and flattened butterfly and show it achieves superior performance because of its ability to adapt to congestion on a per-hop basis. We compare the physical packaging of HyperX and the folded Clos, and show how parallel optical fibers can be used to minimize the number of discrete cable connections in each case such that large 100,000 networks can be constructed with fewer than 10,000 cables. When these high density connections are available, HyperX further allows switches to be colocated with the processing elements in these large computer systems, leading to a simpler structure with a smaller set of component parts compared to folded Clos networks.

8. REFERENCES

[1] "Top500 List," http://www.top500.org/list.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," in *SIGCOMM*, Jun 2008.

[3] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., 2003.

[4] P. Geoffray and T. Hoefler, "Adaptive Routing Strategies for Modern High Performance Networks," in *High Performance Interconnects*, 2008, pp. 165–172.

[5] J. Kim, W. J. Dally, and D. Abts, "Adaptive Routing in High-Radix Clos Network," in *SC'06*, Nov 2006.

[6] ——, "Flattened Butterfly: a Cost-efficient Topology for High-Radix Networks," in *ISCA*, Jun 2007.

[7] J. Kim, W. J. Dally, S. Scott, and D. Abts, "Technology-Driven, Highly-Scalable Dragonfly Topology," in *ISCA*, Jun 2008.

[8] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta, "Microarchitecture of a High-Radix Router," in *ISCA*, Jun 2005.

[9] N. Kirman, *et al.*, "Leveraging Optical Technology in Future Bus-based Chip Multiprocessors," in *MICRO*, Nov 2006.

[10] A. Komornicki, G. Mullen-Schulz, and D. Landon, "Roadrunner: Hardware and Software Overview," *IBM Redpaper*, 2009.

[11] C. Leiserson, "Fat-trees: Universal Networks for Hardware-Efficient Supercomputing," *IEEE Transactions on Computers*, vol. 34, pp. 892–901, 1985.

[12] X.-Y. Lin, Y.-C. Chung, and T.-Y. Huang, "A Multiple LID Routing Scheme for Fat-tree Based InfiniBand Networks," in *IPDPS*, Apr 2004.

[13] N. McKeown, A. Mekkittikul, V. Anantharam, and J. Walrand, "Achieving 100 throughput in an input-queued switch," *Communications, IEEE Transactions on*, vol. 47, no. 8, Aug 1999.

[14] National Center for Computational Sciences, "Jaguar," http://www.nccs.gov/computing-resources/jaguar.

[15] S. R. Ohring, M. Ibel, S. K. Das, and M. J. Kumar, "Generalized Fat Trees," in *IEEE IPPS*, 1995, pp. 37–44.

[16] T. Sabano, *et al.*, "Development of Reference MT Ferrule using Insert-molded Metal Plate," *OFC/NFOEC*, Feb 2008.

[17] S. Scott, D. Abts, J. Kim, and W. J. Dally, "The BlackWidow High-Radix Clos Network," in *ISCA*, Jun 2006.

[18] Semiconductor Industries Association, "International Technology Roadmap for Semiconductors," http://www.itrs.net, 2007 Edition.

[19] L. G. Valiant, "A Scheme for Fast Parallel Communication," *SIAM Journal on Computing*, vol. 11, no. 2, 1982.

[20] D. Vantrease, *et al.*, "Corona: System Implications of Emerging Nanophotonic Technology," in *ISCA*, Jun 2008.