Flattened Butterfly : A Cost-Efficient Topology for High-Radix Networks

John Kim, William J. Dally
Computer Systems Laboratory
Stanford University, Stanford, CA 94305
{jjk12, billd}@cva.stanford.edu

Dennis Abts
Cray Inc.
Chippewa Falls, WI 54729
dabts@cray.com

ABSTRACT

Increasing integrated-circuit pin bandwidth has motivated a corresponding increase in the degree or radix of interconnection networks and their routers. This paper introduces the *flattened butterfly*, a cost-efficient topology for high-radix networks. On benign (load-balanced) traffic, the flattened butterfly approaches the cost/performance of a butterfly network and has roughly half the cost of a comparable performance Clos network. The advantage over the Clos is achieved by eliminating redundant hops when they are not needed for load balance. On adversarial traffic, the flattened butterfly matches the cost/performance of a folded-Clos network and provides an order of magnitude better performance than a conventional butterfly. In this case, global adaptive routing is used to switch the flattened butterfly from minimal to non-minimal routing — using redundant hops only when they are needed. Minimal and non-minimal, oblivious and adaptive routing algorithms are evaluated on the flattened butterfly. We show that load-balancing adversarial traffic requires non-minimal globally-adaptive routing and show that *sequential* allocators are required to avoid transient load imbalance when using adaptive routing algorithms. We also compare the cost of the flattened butterfly to folded-Clos, hypercube, and butterfly networks with identical capacity and show that the flattened butterfly is more cost-efficient than folded-Clos and hypercube topologies.

Categories and Subject Descriptors

C.1.2 [**Computer Systems Organization**]: Multiprocessors—*Interconnection architectures*; B.4.3 [**Hardware**]: Interconnections—*Topology*

General Terms

Design, Performance

Keywords

interconnection networks, topology, high-radix routers, flattened butterfly, global adaptive routing, cost model

1. INTRODUCTION

Interconnection networks are widely used to connect processors and memories in multiprocessors [23, 24], as switching fabrics for high-end routers and switches [8], and for connecting I/O devices [22]. As processor and memory performance continues to increase in a multiprocessor computer system, the performance of the interconnection network plays a central role in determining the overall performance of the system. The latency and bandwidth of the network largely establish the remote memory access latency and bandwidth.

Over the past 20 years k-ary n-cubes [6] have been widely used – examples of such networks include SGI Origin 2000 [17], Cray T3E [24], and Cray XT3 [5]. However, the increased pin density and faster signaling rates of modern ASICs is enabling high-bandwidth router chips with >1Tb/s of off-chip bandwidth [23]. Low-radix networks, such as k-ary n-cubes, are unable to take full advantage of this increased router bandwidth. To take advantage of increased router bandwidth requires a high-radix router – with a large number of *narrow* links, rather than a conventional router with a small number of *wide* links. With modern technology, high-radix networks based on a folded-Clos [4] (or fat-tree [18]) topology provide lower latency and lower cost [14] than a network built from conventional low-radix routers. The next generation Cray BlackWidow vector multiprocessor [23] is one of the first systems that uses high-radix routers and implements a modified version of the folded-Clos network.

In this paper, we introduce the high-radix *flattened butterfly* topology which provides better path diversity than a conventional butterfly and has approximately half the cost of a comparable performance Clos network on balanced traffic. The flattened butterfly is similar to a generalized hypercube network [2] but by utilizing concentration, the flattened butterfly can scale more effectively and also exploit high-radix routers. Routing algorithms on the flattened butterfly are compared and we show that global-adaptive routing is needed to load-balance the topology. We show that greedy implementations of global-adaptive routing algorithms give poor instantaneous load-balance. We describe a *sequential* implementation of the UGAL [27] routing algorithm that overcomes this load-imbalance and also introduce the CLOS AD routing algorithm that removes transient load imbalance as the flattened butterfly is routed similar to a folded-Clos network. In addition to the performance analysis, we provide a detailed cost comparison of high-radix topologies and discuss the cost advantages of the flattened butterfly.

The remainder of the paper is organized as follows. We

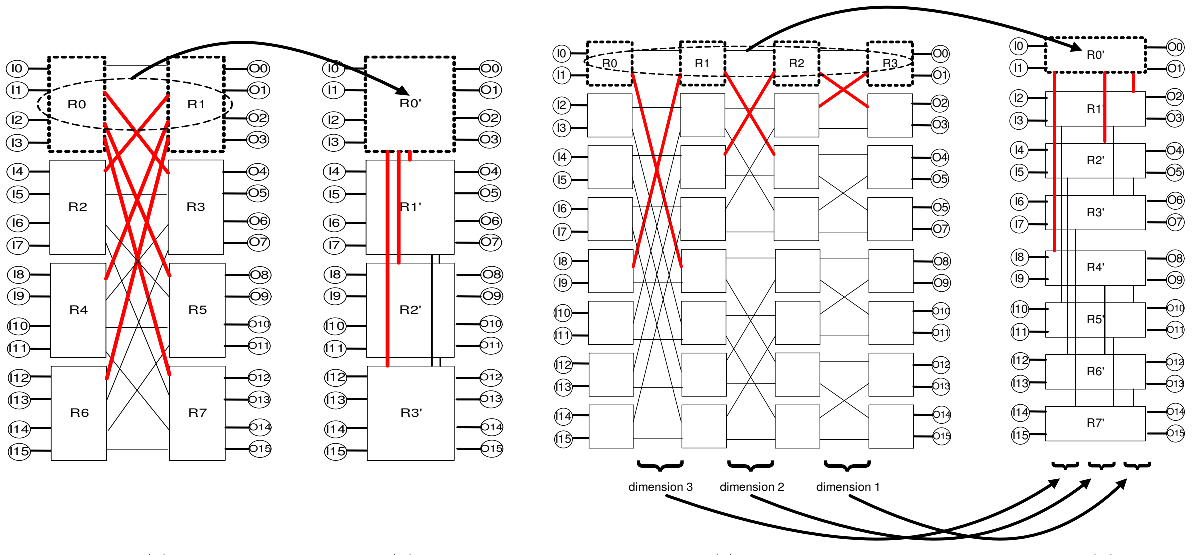


Figure 1: Block diagram of (a) 4-ary 2-fly butterfly and (b) 4-ary 2-flat – the corresponding flattened butterfly with a single dimension, (c) 2-ary 4-fly butterfly and (d) 2-ary 4-flat – the corresponding flattened butterfly with three dimensions. Lines denote unidirectional links in the butterfly and bidirectional links (i.e. two unidirectional links) in the flattened butterfly.

describe the flattened butterfly topology in detail in Section 2 and discuss routing algorithms in Section 3. Section 4 presents a cost model for interconnection networks and provides a cost comparison between the flattened butterfly and alternative topologies. Additional discussion of the flattened butterfly is provided in Section 5 including a power comparison. Section 6 discusses related work, and Section 7 presents our conclusions.

2. FLATTENED BUTTERFLY TOPOLOGY

The butterfly network (k-ary n-fly) can take advantage of high-radix routers to reduce latency and network cost [9]. However, there is no path diversity in a butterfly network which results in poor throughput for adversarial traffic patterns. Also, a butterfly network cannot exploit traffic locality. A Clos network [4] provides many paths[1] between each pair of nodes. This path diversity enables the Clos to route arbitrary traffic patterns with no loss of throughput. The input and output stages of a Clos network can be combined or *folded* on top of one other creating a folded Clos or fat-tree [18] network which can exploit traffic locality.

A Clos or folded Clos network, however, has a cost that is nearly double that of a butterfly with equal capacity and has greater latency than a butterfly. The increased cost and latency both stem from the need to route packets first to an arbitrary middle stage switch and then to their ultimate destination. This doubles the number of long cables in the network, which approximately doubles cost, and doubles the number of inter-router channels traversed, which drives up latency.

The flattened butterfly is a topology that exploits high-radix routers to realize lower cost than a Clos on load-balanced traffic, and provide better performance and path diversity than a conventional butterfly. In this section, we describe the flattened butterfly topology in detail by comparing it to a conventional butterfly and show how it is also similar to a folded-Clos.

[1]One for each middle-stage switch in the Clos.

2.1 Topology Construction: Butterfly to Flattened Butterfly

We derive the flattened butterfly by starting with a conventional butterfly (k-ary n-fly) and combining or *flattening* the routers in each *row* of the network into a single router. Examples of flattened butterfly construction are shown in Figure 1. 4-ary 2-fly and 2-ary 4-fly networks are shown in Figure 1(a,c) with the corresponding flattened butterflies shown in Figure 1(b,d). The routers R0 and R1 from the first row of Figure 1(a) are combined into a single router R0′ in the flattened butterfly of Figure 1(b). Similarly routers R0, R1, R2, and R3 of Figure 1(c) are combined into R0′ of Figure 1(d). As a row of routers is combined, channels entirely local to the row, e.g., channel (R0,R1) in Figure 1(a), are eliminated. All other channels of the original butterfly remain in the flattened butterfly. For example, channel (R0,R3) in Figure 1(a) becomes channel (R0′,R1′) in Figure 1(b). Because channels in a flattened butterfly are symmetrical, each line in Figures 1(b,d) represents a bidirectional channel (i.e. two unidirectional channels), while each line in Figures 1(a,c) represents a unidirectional channel.

A k-ary n-flat, the flattened butterfly derived from a k-ary n-fly, is composed of $\frac{N}{k}$ radix $k' = n(k-1)+1$ routers where N is the size of the network. The routers are connected by channels in $n' = n-1$ dimensions, corresponding to the $n-1$ columns of inter-rank wiring in the butterfly. In each dimension d, from 1 to n', router i is connected to each router j given by

$$j = i + \left[m - \left(\left\lfloor \frac{i}{k^{d-1}} \right\rfloor \textit{mod } k \right) \right] k^{d-1} \qquad (1)$$

for m from 0 to $k-1$, where the connection from i to itself is omitted. For example, in Figure 1(d), R4′ is connected to R5′ in dimension 1, R6′ in dimension 2, and R0′ in dimension 3.

The number of nodes (N) in a flattened butterfly is plotted as a function of number of dimensions n' and switch radix k' in Figure 2. The figure shows that this topology is

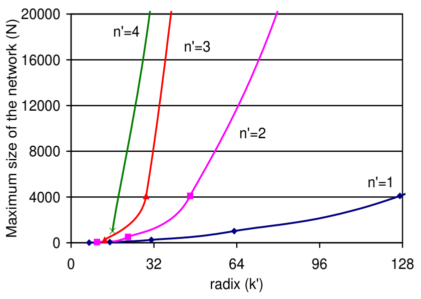


Figure 2: Network size (N) scalability as the radix (k') and dimension (n') is varied.

suited only for high-radix routers. Networks of very limited size can be built using low-radix routers ($k' < 16$) and even with $k' = 32$ many dimensions are needed to scale to large network sizes. However with $k' = 61$, a network with just three dimensions scales to 64K nodes.

2.2 Routing and Path Diversity

Routing in a flattened butterfly requires a hop from a node to its local router, zero or more inter-router hops, and a final hop from a router to the destination node. If we label each node with a n-digit radix-k node address, an inter-router hop in dimension d changes the d^{th} digit of the current node address to an arbitrary value, and the final hop sets the 0^{th} (rightmost) digit of the current node address to an arbitrary value. Thus, to route minimally from node $a = a_{n-1}, \ldots, a_0$ to node $b = b_{n-1}, \ldots, b_0$ where a and b are n-digit radix-k node addresses involves taking one inter-router hop for each digit, other than the rightmost, in which a and b differ. For example, in Figure 1(d) routing from node 0 (0000_2) to node 10 (1010_2) requires taking inter-router hops in dimensions 1 and 3. These inter-router hops can be taken in either order giving two minimal routes between these two nodes. In general, if two nodes a and b have addresses that differ in j digits (other than the rightmost digit), then there are $j!$ minimal routes between a and b. This path diversity derives from the fact that a packet routing in a flattened butterfly is able to traverse the dimensions in any order, while a packet traversing a conventional butterfly must visit the dimensions in a fixed order – leading to no path diversity.

Routing non-minimally in a flattened butterfly provides additional path diversity and can achieve load-balanced routing for arbitrary traffic patterns. Consider, for example, Figure 1(b) and suppose that all of the traffic from nodes 0-3 (attached to router R0$'$) was destined for nodes 4-7 (attached to R1$'$). With minimal routing, all of this traffic would overload channel (R0$'$,R1$'$). By misrouting a fraction of this traffic to R2$'$ and R3$'$, which then forward the traffic on to R1$'$, load is balanced. With non-minimal routing, a flattened butterfly is able to match the load-balancing (and non-blocking) properties of a Clos network – in effect acting as a *flattened Clos*. We discuss non-minimal routing strategies for flattened butterfly networks in more detail in Section 3.

2.3 Comparison to Generalized Hypercube

The flattened butterfly is similar to a generalized hypercube [2] (GHC) topology. The generalized hypercube is a k-ary n-cube network that uses a complete connection, rather

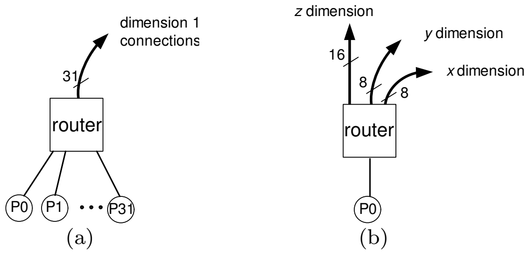


Figure 3: Block diagram of routers used in a 1K network for (a) flattened butterfly with one dimension and (b) (8,8,16) generalized hypercube.

than a ring, to connect the nodes in each dimension. In the 1980s, when this topology was proposed, limited pin bandwidth made the GHC topology prohibitively expensive at large node counts. Also, without load-balancing global adaptive routing, the GHC has poor performance on adversarial traffic patterns.

The flattened butterfly improves on the GHC in two main ways. First, the flattened butterfly connects k terminals to each router while the GHC connects only a single terminal to each router. Adding this k-way concentration makes the flattened butterfly much more economical than the GHC - reducing its cost by a factor of k, improves its scalability, and makes it more suitable for implementation using high-radix routers. For example, routers used in a 1K node network of a flattened butterfly with one dimension and a (8,8,16) GHC are shown in Figure 3. With 32-terminal nodes per router, the terminal bandwidth of the flattened butterfly is matched to the inter-router bandwidth. In the GHC, on the other hand, there is a mismatch between the single terminal channel and 32 inter-router channels. If the inter-router channels are of the same bandwidth as the terminal channel, the network will be prohibitively expensive and utilization of the inter-router channels will be low. If the inter-router channels are sized at 1/32 the bandwidth of the terminal channel, serialization latency will make the latency of the network prohibitively high and the overall bandwidth of the router will be low, making poor use of the high-pin bandwidth of modern VLSI chips.

The second improvement over the GHC is the use of non-minimal globally-adaptive routing (Section 3) to load-balance adversarial traffic patterns and the use of adaptive-Clos routing with *sequential* allocators to reduce transient load imbalance. These modern routing algorithms enable the flattened butterfly to match the performance of a Clos network on adversarial traffic patterns. In contrast, a GHC using minimal routing is unable to balance load and hence suffers the same performance bottleneck as a conventional butterfly on adversarial traffic.

3. ROUTING ALGORITHMS AND PERFORMANCE COMPARISON

In this section, we describe routing algorithms for the flattened butterfly and compare their performance on different traffic patterns. We also compare the performance of the flattened butterfly to alternative topologies.

3.1 Routing Algorithms on Flattened Butterfly

We evaluate the flattened butterfly on five routing algorithms: minimal adaptive (MIN AD), Valiant's non-minimal oblivious algorithm (VAL), the UGAL non-minimal adaptive algorithm (UGAL), a variant of UGAL using sequential allocation (UGAL-S), and non-minimal adaptive routing in a flattened Clos (CLOS AD). We describe each briefly here. We consider routing in a k-ary n-flat where the source node s, destination node d, and current node c are represented as n-digit radix-k numbers, e.g., $s_{n-1}, \ldots, s_0$. At a given step of the route, a channel is productive if it is part of a minimal route; that is, a channel in dimension j is productive if $c_j \neq d_j$ before traversing the channel, and $c_j = d_j$ after traversing the channel.

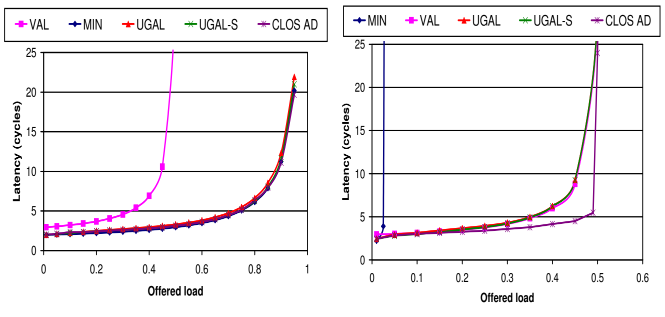
We assume an input-queued virtual channel router [9]. Adaptive algorithms estimate output channel queue length using the credit count for output virtual channels which reflects the occupancy of the input queue on the far end of the channel. For MIN AD and UGAL, the router uses a *greedy* allocator [13]: during a given routing cycle, all inputs make their routing decisions in parallel and then, the queuing state is updated en mass. For UGAL-S and CLOS AD, the router uses a *sequential* allocator where each input makes its routing decision in sequence and updates the queuing state before the next input makes its decision.

Minimal Adaptive (MIN AD): The minimal adaptive algorithm operates by choosing for the next hop the productive channel with the shortest queue. To prevent deadlock, n' virtual channels (VCs) [7] are used with the VC channel selected based on the number of hops remaining to the destination.

Valiant (VAL) [30]: Valiant's algorithm load balances traffic by converting any traffic pattern into two phases of random traffic. It operates by picking a random intermediate node b, routing minimally from s to b, and then routing minimally from b to d. Routing through b perfectly balances load (on average) but at the cost of doubling the worst-case hop count, from n' to $2n'$. While any minimal algorithm can be used for each phase, our evaluation uses dimension order routing. Two VCs, one for each phase, are needed to avoid deadlock with this algorithm.

Universal Globally-Adaptive Load-balanced (UGAL [27], UGAL-S) : UGAL chooses between MIN AD and VAL on a packet-by-packet basis to minimize the estimated delay for each packet. The product of queue length and hop count is used as an estimate of delay. With UGAL, traffic is routed minimally on benign traffic patterns and at low loads, matching the performance of MIN AD, and non-minimally on adversarial patterns at high loads, matching the performance of VAL. UGAL-S is identical to UGAL but with a sequential allocator.

Adaptive Clos (CLOS AD): Like UGAL, the router chooses between minimal and non-minimal on a packet-by-packet basis using queue lengths to estimate delays. If the router chooses to route a packet non-minimally, however, the packet is routed as if it were adaptively routing to the middle stage of a Clos network. A non-minimal packet arrives at the intermediate node b by traversing each dimension using the channel with the shortest queue for that dimension (including a "dummy queue" for staying at the current coordinate in that dimension). Like UGAL-S, CLOS AD uses a sequential allocator. The routing is identical to adaptive routing in a folded-Clos [13] where the folded-Clos is flattened into the routers of the flattened butterfly. Thus, the intermediate node is chosen from the closest common ancestors and not among all nodes. As a result, even though CLOS AD is non-minimal routing, the hop count is always equal or less than that of a corresponding folded-Clos network.



(a) (b)

Figure 4: Routing algorithm comparisons on the flattened butterfly with (a) uniform random traffic and (b) worst case traffic pattern.

3.2 Evaluation & Analysis

We use cycle accurate simulations to evaluate the performance of the different routing algorithms in the flattened butterfly. We simulate a single-cycle input-queued router switch. Packets are injected using a Bernoulli process. The simulator is warmed up under load without taking measurements until steady-state is reached. Then a sample of injected packets is labeled during a measurement interval. The simulation is run until all labeled packets exit the system.

We simulate a 32-ary 2-flat flattened butterfly topology ($k' = 63$, $n' = 1$, $N = 1024$). Simulations of other size networks and higher dimension flattened butterfly follow the same trend and are not presented due to space constraints. We simulate single-flit packets[2] and assume the total buffering per port is 32 flit entries. Although input-queued routers have been shown to be problematic in high-radix routers [14], we use input-queued routers but provide sufficient switch speedup in order to generalize the results and ensure that routers do not become the bottleneck of the network.

We evaluate the different routing algorithms on the flattened butterfly using both benign and adversarial traffic patterns. Uniform random (UR) traffic is a benign traffic pattern that balances load across the network links. Thus, minimal routing is sufficient for such traffic patterns and all of the routing algorithms except VAL achieve 100% throughput (Figure 4(a)). VAL achieves only half of network capacity regardless of the traffic pattern[3] [28]. In addition, VAL adds additional zero-load latency with the extra hop count associated with the intermediate node.

We also simulate an adversarial traffic pattern where each node associated with router R_i sends traffic to a randomly selected node associated with router R_{i+1}. With this traffic pattern, all of the nodes connected to a router will attempt

[2]Different packet sizes do not impact the comparison results in this section.

[3]The capacity of the network (or the ideal throughput for UR traffic) is given by $2B/N$ where B is the total bisection bandwidth and N is the total number of nodes [9]. For the flattened butterfly, similar to the butterfly network, $B = N/2$, thus the capacity of the flattened butterfly is 1.

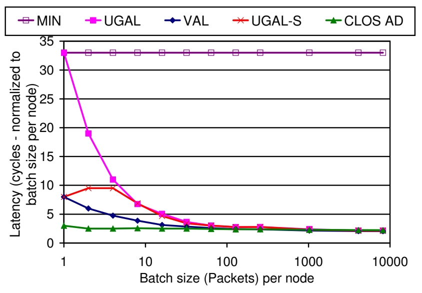


Figure 5: Dynamic response comparison of the routing algorithms.

to use the same inter-router channel. Non-minimal routing is required to load balance this traffic pattern by spreading the bulk of the traffic across the other inter-router channels. Figure 4(b) compares the performance of the routing algorithms on this pattern. MIN is limited to 1/32 or approximately 3% throughput while all of the non-minimal algorithms achieve 50% throughput (the maximum possible on this traffic).

With the adversarial traffic pattern, CLOS AD provides much lower latency near saturation — nearly half the latency of UGAL-S at an offered load of 0.45. This reduction in latency is analogous to the reduction in latency achieved by adaptive routing compared to oblivious routing in a Clos network [13]. Because CLOS AD adaptively picks the intermediate node, it is able to dynamically load balance traffic across the intermediate nodes and links. VAL, UGAL, and UGAL-S obliviously pick the intermediate node which balances average traffic across nodes and links but results in transient load imbalance that increases latency.

To highlight the effects of transient load imbalance, Figure 5 shows the time required by each algorithm to deliver a batch of traffic normalized to batch size. We use the adversarial traffic pattern described earlier. As the batch size increases, the normalized latency approaches the inverse of the routing algorithm throughput. For small batch sizes, however, batch latency is affected by transient load imbalance. On these small batches, UGAL performs very poorly because of the greedy nature of its allocator. When the minimal queue is short, all inputs pick the minimal queue (overloading this output) before the queuing state is updated. With UGAL-S, the transient load imbalance due to greedy allocation is eliminated, but transient load imbalance across intermediate nodes remains. VAL also shares this transient load imbalance. CLOS AD eliminates both sources of transient load imbalance.

Both CLOS AD and UGAL-S require sequential allocations which can increase the router clock cycle or the pipeline depth if they are implemented sequentially. However, these algorithms can be implemented using parallel prefix schemes [16] to speed up the computation. Although CLOS AD provides performance benefit over UGAL-S, it leads to a higher complexity implementation since it requires comparing the depth of all of the non-minimal queues. Techniques to reduce the complexity of adaptive routing in high-radix Clos networks have been discussed [13] and can be implemented on the flattened butterfly as well.

3.3 Comparison to Other Topologies

We compare the performance of the flattened butterfly to three other topologies : the conventional butterfly, the folded Clos, and the hypercube. We use a network of 1024 nodes – thus, the conventional butterfly is built using 2 stages of radix-32 routers, the folded-Clos network is a 3-stage network using radix-64 routers, and the hypercube is a 10 dimensional hypercube. The bisection bandwidth is held constant across the four topologies. The routing algorithms used for each of the topologies are described in Table 1 and the performance comparison is shown in Figure 6. For the different topologies, the total buffer storage is held constant i.e. the product of the VCs and the depth of each buffer is kept constant.

Topology	**Routing**	num of VCs
Flattened Butterfly	CLOS AD	2
Conventional Butterfly	destination-based	1
Folded Clos	adaptive *sequential* [13]	1
Hypercube	e-cube	1

Table 1: Topology and Routing used in performance comparison. The VCs are used to break routing deadlocks.

On UR traffic (Figure 6(a)), all of the topologies providing 100% throughput except for the folded-Clos. By holding bisection bandwidth constant across the topologies, the folded Clos uses 1/2 of the bandwidth for load-balancing to the middle stages – thus, only achieves 50% throughput. In addition, the folded Clos has slightly higher latency because of the extra middle stage and the hypercube also has much higher latency because of its higher diameter. On WC traffic (Figure 6(b)), the conventional butterfly throughput is severely limited and performs identically to a flattened butterfly with minimal routing – leading to an order of magnitude difference in throughput. However, the other topologies perform similarly as they result in a throughput of 50%. Thus, the flattened butterfly provides 2x increase in performance over the folded-Clos on benign traffic while providing the same performance on the worst-case traffic pattern when the cost (i.e. bisection bandwidth) is held constant. In the next section, we provide a detailed cost model that includes the cost of the channels in different packaging hierarchy as well as the router cost and show how cost-efficient flattened butterfly is compared to the other topologies.

4. TOPOLOGY COST COMPARISON

In this section, we provide a cost comparison of four different topologies evaluated in Section 3.3. We present a cost model of an interconnection network based on the cost of routers and links of different types and lengths. We then describe the packaging of the topologies and estimate the length of the links. Using our cost model and the estimated link lengths we compute the cost of each topology.

4.1 Cost Model

The interconnection network can account for as much as 1/3 of the overall system cost [21]. In this work, we compare the cost of networks with a given level of performance (bandwidth).

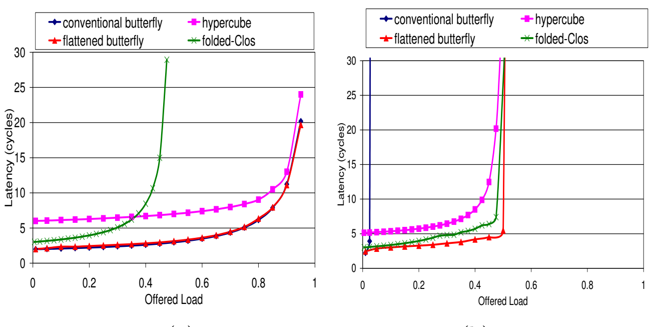


Figure 6: Topology comparisons of the flattened butterfly and the folded Clos with (a) uniform random traffic and (b) worst case traffic pattern.

The system components, processing nodes and routers, are packaged within a *packaging hierarchy*. At the lowest level of the hierarchy are the compute modules (containing the processing nodes) and routing modules (containing the routers). At the next level of the hierarchy, the modules are connected via a backplane or midplane printed circuit board. The modules and backplane are contained within a *cabinet*. A *system* consists of one or more cabinets with the necessary *cables* connecting the router ports according to the network topology. The network cables may aggregate multiple network links into a single cable to reduce cost and cable bulk.

Component	Cost
router	\$390
router chip	\$90
development	\$300
links (cost per signal 6Gb/s)	
backplane	\$1.95
electrical	\$3.72 + \$0.81 ℓ
optical	\$220.00

Table 2: Cost breakdown of an interconnection network.

Network cost is determined by the cost of the routers and the backplane and cable links. The cost of a link depends on its length and location within the packaging hierarchy (Table 2). A link within the backplane[4] is about \$1.95 per differential signal[5], whereas a cable connecting nearby routers (routers within 2m) is about \$5.34 per signal, and an optical cable is about \$220 per signal[6] to connect routers in a distant cabinet. Inexpensive backplanes are used to connect modules in the same cabinet over distances that are typically less than 1m. Moderate cost electrical cables connect modules in different cabinets up to lengths of 5-10m.[7] Transmitting signals over longer distance require either an expensive optical cable or a series of electrical cables connected by *repeaters* that retime and amplify the signal. Because optical technology still remains relatively expensive compared to electrical signaling over copper, our cost analysis uses electrical signaling with repeaters as necessary for driving signals over long distances.

Router cost is divided into development (non-recurring) cost and silicon (recurring) cost. The development cost is amortized over the number of router chips built. We assume a non-recurring development cost of ≈\$6M for the router which is amortized over 20k parts, about \$300 per router. The recurring cost is the cost for each silicon part which we assume to be ≈\$90 per router using the MPR cost model [19] for a TSMC 0.13μm 17mm×17mm chip which includes the cost of packaging and test.

The cost of electrical cables can be divided into the cost of the wires (copper) and the overhead cost which includes the connectors, shielding, and cable assembly. Figure 7(a) plots the cost per differential signal for Infiniband 4x and 12x cables as a function of distance. The cost of the cables was obtained from Gore [11]. The slope of the line represents the cable cost per unit length (\$/meter) and the y-intercept is the overhead cost associated with shielding, assembly and test. With the Infiniband 12x cables, the overhead cost per signal is reduced by 36% because of cable aggregation – Infiniband 12x contains 24 wire pairs while Infiniband 4x contains only 8 pairs, thus 12x cable is able to amortize the shielding and assembly cost. [8]

When the cable length exceeds the critical length of 6m, repeaters need to be inserted and the cable cost with repeaters is shown in Figure 7(b), based on the Infiniband 12x cable cost. When length is 6m, there is a step in the

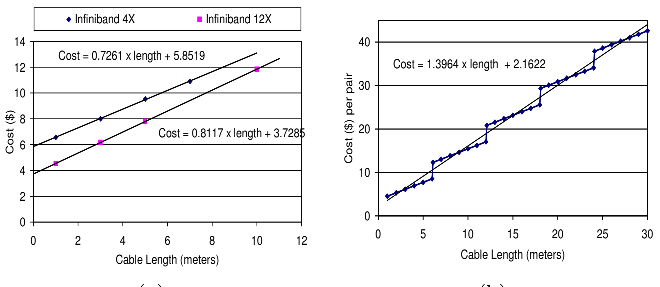


Figure 7: Cable Cost Data. (a) Cost of Infiniband 4x and 12x cables as a function of cable length and (b) cable cost model with the use of repeaters for cable ¿6m. The model is based on the Infiniband 12x cost model and the data point is fitted with a line to calculate the average cable cost.

[4]We assume \$3000 backplane cost for 1536 signals, or \$1.95 per backplane signal which includes the GbX connector cost at \$0.12 per mated signal to support signal rates up to 10 Gb/s [1].

[5]To provide signal integrity at high speed, signals are almost always transmitted differentially - using a pair of wires per signal.

[6]Pricing from www.boxfire.com – \$480 for 20 meter 24 strand terminated fiber optic cable results in \$20 per signal for the cost of the cable. Although optical transceiver modules have a price goal of around \$200, that has not yet been achieved with current modules costing \$1200 and up.

[7]In our analysis, we will assume that a 6m cable is the longest distance that can be driven at full signalling rate of 6.25 Gb/s, based on SerDes technology similar to that used in the Cray BlackWidow [23].

[8]It is interesting to note that the 12x has a slightly higher wire cost per unit length. The Infiniband 4x is a *commodity* cable whereas the 12x cables needed to be custom ordered – thus, resulting in the higher price.

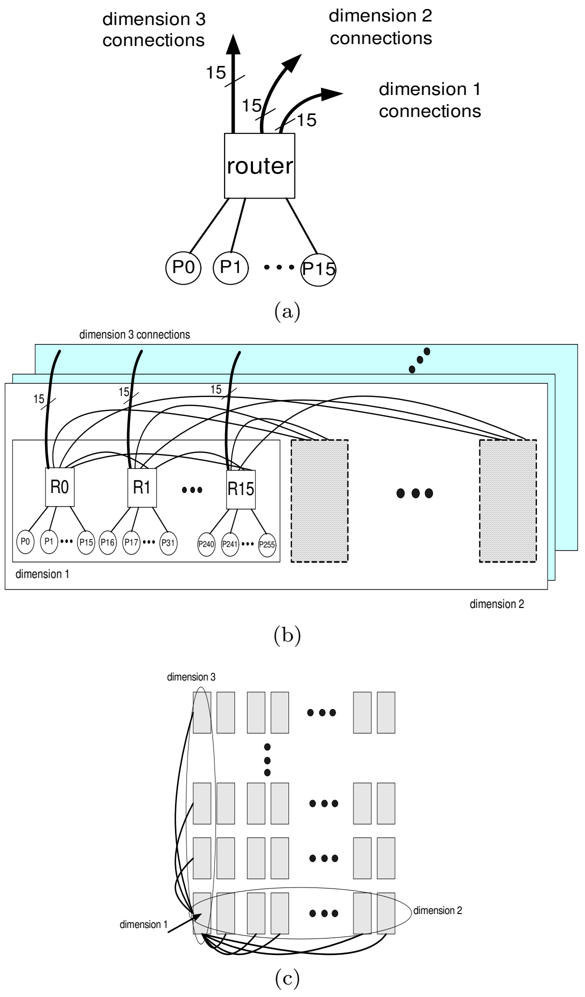


Figure 8: Block diagrams of a 16-ary 4-flat flattened butterfly. (a) Block diagram of each router where 16 ports are used for the terminal nodes and 15 ports are used for connections in each dimension. (b) Block diagram of the topology which can scale up to more than 64K nodes. (c) A packaging block diagram of the topology, with the connection for only the lower left cabinet shown.

cost which reflects the repeater cost. Since the cost of the repeaters is dominated by the extra connector cost, the increase in the cost function is approximately the additional connector cost. The linear cost model shown in Figure 7(b) is used for all cables.

4.2 Packaging and Cable length

Figure 8 shows a possible packaging of a 16-ary 4-flat ($k' = 61$, $n' = 3$) flattened butterfly network. Each router has channels to 16 terminal nodes and to 45 other routers, 15 in each of three dimensions (Figure 8(a)). Figure 8(b) shows how the routers connected in dimension 1 are packaged in a subsystem containing 256 nodes.[9] The like elements from 16 of these dimension 1 subsystems are connected in dimensions 2 forming a subsystem with 4,096 nodes. Up to 16 of these subsystems can be combined in dimension 3 leading to a network with up to 65,536 nodes. A possible packaging of this configuration is shown in Figure 8(c) where each box represents a pair of cabinets that contains a dimension 1 subsystem. Dimension 2 is mapped across the columns, and dimension 3 is mapped across the rows. The connections are only shown for the lower left cabinet.

[9]We assume packaging 128 nodes per cabinet as in Cray BlackWidow. Hence a dimension 1 subsystem requires two cabinets connected by short cables.

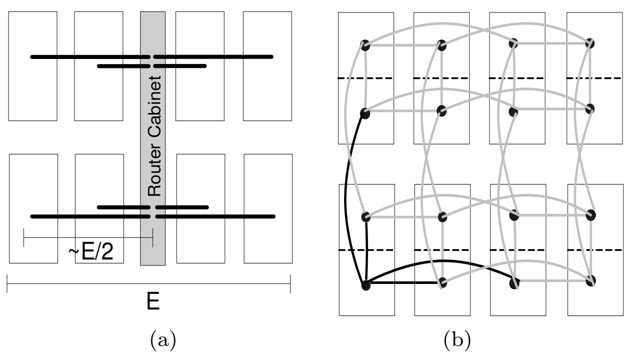


Figure 9: Sample cabinet packaging layout of 1024 nodes in (a) folded-Clos and (b) hypercube topologies. Each box represents a cabinet of 128 nodes and the hypercube is partitioned into two chassis. In the folded-Clos, the router cabinet is assumed to be placed in the middle and for illustration purpose only, the router cabinet is drawn taller.

From Figure 8(c), the maximum cable length (L_{max}) of the flattened butterfly topology will be approximately equal to the length of one edge (E) of the 2-D cabinet layout. The average *global* cable length (L_{avg}) for connections in dimensions 2 and 3 is approximately $L_{max}/3 = E/3$. Dimension 1 connections between nodes are made over backplanes or very short (1-2m) cables between nodes in different cabinets. It can be seen that the L_{max} and the L_{avg} of the butterfly will be the same as that of the flattened butterfly since the flattened butterfly's channels were derived from the channels in the conventional butterfly.

Packaging of the folded-Clos and the hypercube are shown in Figure 9. For the folded-Clos, L_{max} is approximately $E/2$ since the cables only need to be routed to a central routing cabinet (Figure 9(a)). L_{avg} for the Clos is approximately $L_{max}/2 = E/4$. The hypercube has similar L_{max} cable lengths as the folded-Clos (Figure 9(b)). The diagram illustrates only the *global* cables – inter-cabinet connections for the higher dimensions in the hypercube. Each node of the hypercube connects to a single node in each dimension of the network, thus the longest cable will be $E/2$, the next longest cable will be $E/4$ and so forth. The cable lengths are a geometric distribution, which can be summed to arrive at the average cable length, L_{avg} of approximately $(E-1)/log_2(E)$. Because of the logarithmic term, as the network size increases, the average cable length is shorter than the other topologies.

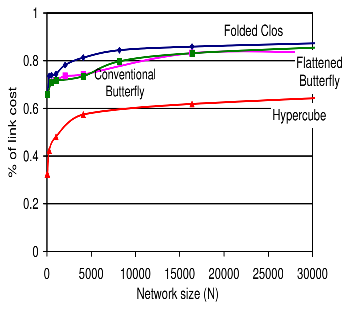
The length of an edge (E) in a cabinet packaging layout can be estimated as

$$E = \sqrt{\frac{N}{D}}.$$

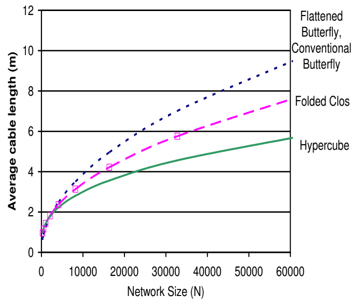
where N is the number of nodes and D is the density of nodes (nodes/m^2).

parameter	value
radix	64
# of pairs per port	3
nodes per cabinet	128
cabinet footprint	0.57m x 1.44m
D (density)	75 nodes/m^2
cable overhead	2m

Table 3: Technology and packaging assumptions used in the topology comparison. The values are representative of those used in the Cray BlackWidow parallel computer.



(a)



(b)

Figure 10: (a) The ratio of the link cost and the (b) average cable length of the different topologies as the network size is increased. The cable overhead is not included in this plot.

4.3 Cost Comparison

The assumptions and the parameters used in the topology cost comparison are shown in Table 3. In determining the actual cable length for both L_{max} and L_{avg}, we add 2m of cable overhead – 1m of vertical cable run at each end of the cable. We multiply a factor of 2 to the depth of the cabinet footprint to allow for spacing between rows of cabinets. Based on these parameters and the cost model described earlier, we compare the cost of conventional butterfly, folded-Clos, hypercube, and flattened butterfly topologies while holding the random bisection bandwidth constant.

In Figure 10, the ratio of the link cost to the total interconnection network cost is shown. The total cost of the interconnection network is dominated by the cost of the links. Because of the number of routers in the hypercube, the routers dominate the cost for small configurations.[10] However, for larger hypercube networks ($N > 4K$), link cost accounts for approximately 60% of network cost. For the other three topologies, link cost accounts for approximately 80% of network cost. Thus, it is critical to reduce the number of links to reduce the cost of the interconnection network.

The average cable length (L_{avg}) of the different topologies is plotted in Figure 10(b) as the network size is varied, based on the cable length model presented in Section 4.2.

[10]The router cost for the hypercube is appropriately adjusted, based on the number of pins required. Using concentration in the hypercube could reduce the cost of the network but will significantly degrade performance on adversarial traffic pattern and thus, is not considered.

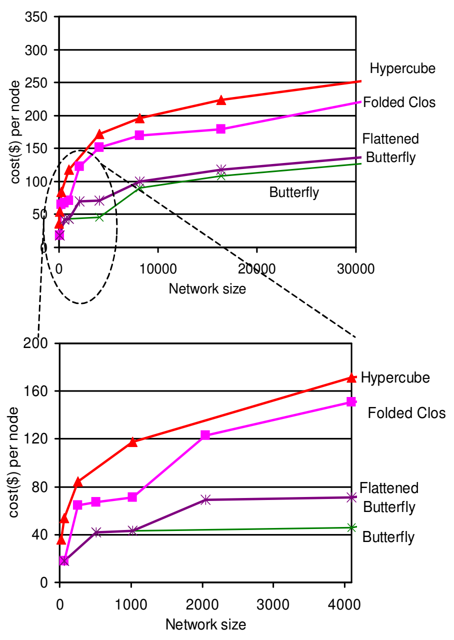


Figure 11: Cost comparison of the different topologies. The bottom plot is the same plot as the plot on the top with the x-axis zoomed in to display the smaller networks more clearly.

For small network size (<4K), there is very little difference in L_{avg} among the different topologies. For larger network size, L_{avg} for the flattened butterfly is 22% longer than that of a folded-Clos and 54% longer than that of a hypercube. However, with the reduction in the number of global cables, the flattened butterfly is still able to achieve a cost reduction compared to the other topologies.

We plot the cost per node of the four topologies in Figure 11 as the network size is increased. In general, the butterfly network provides the lowest cost while the hypercube and the folded-Clos have the highest cost. The flattened butterfly, by reducing the number of links, gives a 35-53% reduction in cost compared to the folded-Clos.

A step increase occurs in the cost of the folded-Clos when the number of stages increases. For example, increasing the network size from 1K to 2K nodes with radix-64 routers requires adding an additional stage to the network to go from a 2-stage folded-Clos to a 3-stage folded-Clos. The flattened butterfly has a similar *step* in the cost function when the number of dimensions increases. For example, a dimension must be added to the network to scale from 1K to 2K nodes. However, the cost increase is much smaller (approximately \$40 per node increase compared to \$60 per node increase for the folded-Clos) since only a single link is added by increasing the number of dimension – whereas in the folded-Clos, two links are added.

For small networks (<1K), the cost benefit of flattened butterfly compared to the folded-Clos is approximately 35% to 38%. Although the flattened butterfly halves the number of *global* cables, it does not reduce the number of local links from the processors to the routers. For small networks, these links account for approximately 40% of the total cost per node – thus, total reduction in cost is less than the expected 50%.

k	n	k'	n'
64	2	127	1
16	3	46	2
8	4	29	3
4	6	19	5
2	12	12	11

Table 4: Different k and n parameters for a $N = 4$K network and the corresponding k' and n' flattened butterfly parameter.

For larger networks (>1K), the cost benefit is greater than 40% and at N = 4K, the cost is reduced by about 53%. The cost reduction of more than 50% is the result of the *packaging locality* that can be exploited with the flattened butterfly. With the flattened butterfly, dimension 1 connections are contained within a pair of adjacent cabinets and made with short links. In the folded Clos, however, the corresponding links are routed to a central cabinet requiring *global* links. In addition, the number of links are actually reduced by more than 1/2 in the flattened butterfly – for example, with N = 1K network, the folded Clos requires 2048 links while the flattened butterfly requires 31 x 32 = 992 links, not 1024 links. For even larger network sizes (16K-32K), the cost benefit reduces to 40 - 45% since the flattened butterfly requires higher average cable length as shown in Figure 10(b).

Although the flattened butterfly was constructed from the conventional butterfly, the conventional butterfly is a lower cost network for 1K < N < 4K. With radix-64 routers, the conventional butterfly can scale to 4K nodes with only 2 stages (e.g. only one inter-router link per node). Because the flattened butterfly shares the radix of its router across stages (dimensions), it has a smaller effective radix (e.g., k = 16 for k' = 61) resulting in more stages for the same number of nodes. However, when N > 4K, the butterfly requires 3 stages with all of the inter-router links being *global* – thus, the cost of the flattened butterfly becomes very comparable to the conventional butterfly.

5. DISCUSSION

This section discusses trade-offs in the design parameters of the flattened butterfly, the impact of wire delay, and provides a power comparison of the different topologies compared in Section 4.

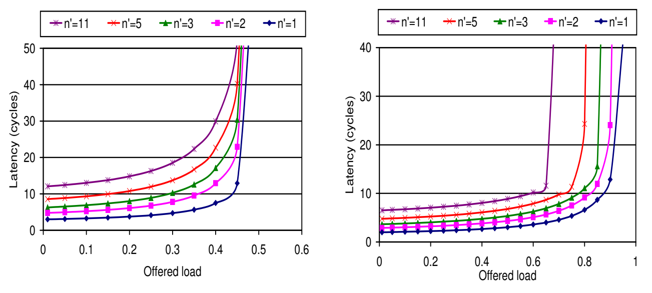
5.1 Design Considerations

5.1.1 Fixed Network Size (N)

For a network with N = 4K nodes, Table 4 shows several values of k' and n' along with the corresponding values of k and n. The performance of the different configurations on UR traffic using the VAL routing algorithm is shown in Figure 12(a). As k' decreases, the diameter of the network and hence latency increases. Because the bisection bandwidth is constant across configurations, throughput remains constant at 50% of capacity.

An additional affect of increased dimensionality is the increase in the number of virtual channels needed to avoid deadlock. With VAL, all of the networks require only 2 VCs to avoid deadlock. However, with adaptive routing, the number of VCs needed is proportional to n'. Figure 12(b) compares the performance of the different configurations using MIN AD routing with the total storage per physical channel (PC) held constant at 64 flit buffers. As with VAL, the networks with higher n' have higher latency. However, since the total storage per PC is divided among n' VCs, increasing n' decreases the storage per VC and hence decreases throughput. The figure shows that throughput degrades about 20% as n' is increased from 1 to 5.

Using the same cost model described in Section 4, the cost per node is compared for the different configurations in Figure 13. As n' increases, the average cable length and hence average cost per cable, decreases as shown in the line plot of Figure 13. However, this decrease in cost per cable is more than offset by the increase in the number of links and routers as n' increases. The cost per node increase by 45% from n' = 1 to n' = 2 and increases by 300% from n' = 1 to n' = 5. Thus, for a given N, the highest radix (and lowest dimensionality) that can be realized results in the highest performance and lowest cost.



(a) (b)

Figure 12: Performance comparison for different N = 4K Flattened Butterflies using (a) VAL and (b) MIN AD routing algorithms.

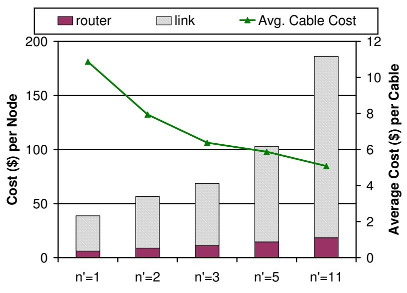


Figure 13: Cost Comparison of N = 4K Flattened Butterflies as n' is increased.

5.1.2 Fixed Radix (k)

To build a flattened butterfly topology with radix-k routers, the smallest dimension (n') of the flattened butterfly should be selected that meets the scaling requirement of the network – e.g.

$$\left\lfloor \frac{k}{n'+1} \right\rfloor^{(n'+1)} \geq N.$$

Based on the value of n' selected, the resulting effective radix

(k') of the topology is

$$k' = \left(\left\lfloor \frac{k}{n'+1} \right\rfloor - 1 \right) (n'+1) + 1.$$

However, depending on value of n' selected, k' may be smaller than k – thus providing extra ports in the router. For example, with radix-64 routers, a flattened butterfly with $n' = 1$ only requires $k' = 63$ to scale to 1K nodes and with $n' = 3$ only requires $k' = 61$ to scale to 64K nodes. The extra ports can be used to increase the size of the network or can be used as redundant ports. An example of expanding a 4-ary 2-flat using radix-8 routers are shown in Figure 14. Since 4-ary 2-flat requires radix-7 routers, an extra port can be used to increase the scalability from $N = 16$ to $N = 20$ (Figure 14(b)). The extra port can also be used to double the bandwidth between local router nodes (Figure 14(a)) which can increase the bandwidth to neighboring router nodes. However, taking advantage of the extra ports does not fundamentally change any characteristics of the topology. The use of redundant links will add some additional cost to the network but the cost advantage of the flattened butterfly and the need for global adaptive routing on the topology are still applicable.

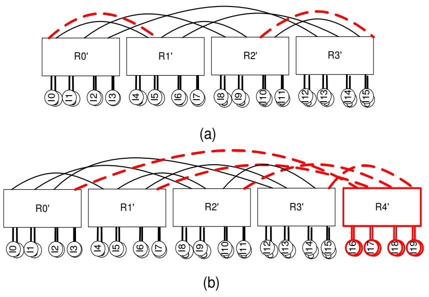


Figure 14: Examples of alternative organization of a 4-ary 2-flat flattened butterfly by (a) using redundant channels and (b) increasing the scalability. The dotted lines represent the additional links added by taking advantage of the extra ports.

5.2 Wire Delay

As shown in Figure 10(b), the flattened butterfly increases the average length of the global cables and can increase the wire delay. However, longer wire delay does not necessarily lead to longer overall latency since the wire delay (time of flight) is based on the physical distance between a source and its destination. As long as the packaging of the topology have *minimal* physical distance, the time of flight is approximately the same regardless of the hop count. Direct networks such as torus or hypercube have minimal physical distance between two nodes but indirect networks such as folded Clos and the conventional butterfly have non-minimal physical distance that can add additional wire latency. However, since there are no intermediate stages in the flattened butterfly, the packaging is similar to a direct network with minimal[11] distance physical distance and does not increase the overall wire delay. For this reason, the wire delay of a flattened butterfly may be smaller than that of an equivalent folded-Clos. For local traffic (such as the WC traffic pattern described in Section 3.2), the folded-Clos needs to route through middle stages, incurring 2x global wire delay where as the flattened butterfly can take advantage of the packaging locality and the minimum physical distance to provide lower delay.

[11]Minimal distance here means minimal *Manhattan* distance and not the distance of a straight line between the two endpoints.

5.3 Power comparison

The power consumption of an interconnect network is an increasing concern. The total power consumption of an interconnection network is the sum of two components : P_{switch} and P_{link}. P_{switch} is the power consumed internal to the router, including the switch, the arbitration logic, and the routing logic, and is proportional to the total *bandwidth* of the router. The complexity of arbitration and routing is increased with the larger number of ports in a high-radix router but Wang et al. has shown that these components have negligible impact on the overall power [31]. P_{link} is the power required to drive the links between the routers and is often the dominating component of the total power consumed by an interconnection network. In the Avici TSR Router [8], 60% of the power budget in the line card is dedicated to the link circuitry and approximately 45% of the power in YARC router [23] is consumed by the serializer/deserializer(SerDes). Depending on the medium (e.g. cables, backplane, etc), P_{link} can vary significantly. Using the same SerDes, the power consumed to drive a local link (P_{link_gl}) is 20% less than the power consumed to drive a global cable (P_{link_gg}) with the reduction in power coming from the equalizer and the transmitter/receiver [3]. For a router in an indirect topology, depending on where in the topology the router is used, a SerDes might need to drive a local link or a global link.

However, for direct topologies and for the flattened butterfly, a dedicated SerDes can be used to exploit the packaging locality – e.g. have separate SerDes on the router chip where some of the SerDes are used to drive local links while others are used to drive global links and reduce the power consumption. For example, a SerDes that can drive <1m of backplane only consumes approximately 40mW [32] (P_{link_ll}), resulting in over 5x power reduction. The different power assumptions are summarized in Table 5 and the power comparison for the different topologies are shown in Figure 15.

The comparison trend is very similar to the cost comparison that was shown earlier in Figure 11. The hypercube gives the highest power consumption while the conventional butterfly and the flattened butterfly give the lowest power consumption. For 1K node network, the flattened butterfly provides lower power consumption than the conventional butterfly since it takes advantage of the dedicated SerDes to drive local links. For networks between 4K and 8K nodes, the flattened butterfly provides approximately 48% power reduction, compared to the folded Clos because a flattened butterfly of this size requires only 2 dimensions while the folded-Clos requires 3 stages. However, for $N >$ 8K, the flattened butterfly requires 3 dimensions and thus, the power reduction drops to approximately 20%.

6. RELATED WORK

The comparison of the flattened butterfly to the generalized hypercube was discussed earlier in Section 2.3. Non-

Component	Power
P_{switch}	40 W
P_{link_gg}	200 mW
P_{link_gl}	160 mW
P_{link_ll}	40 mW

Table 5: Power consumption of different components in a router.

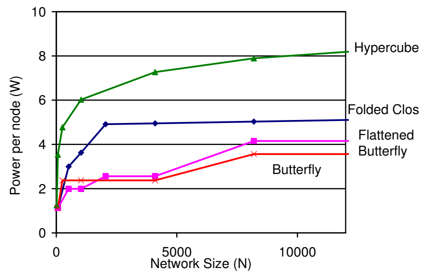


Figure 15: Power comparison of alternative topologies. The power consumption for the interconnection network normalized to N is plotted as the N is increased.

minimal routing on the generalized hypercube has been proposed in circuit switched networks [33]. The adaptive routing proposed uses the non-minimal paths to increase path diversity but does not describe how load-balancing can be achieved with the non-minimal routes.

The Cray BlackWidow network [23] implements a high-radix modified folded-Clos topology using radix-64 routers. The network introduces *sidelinks* that connect neighboring subtrees together directly instead of using another stage of routers. A Rank1.5 BlackWidow network is similar to a one-dimensional flattened butterfly. However, the size of the routing table and packaging constraints limit the scalability of the Rank1.5 BlackWidow network to 288 nodes. Also, the BlackWidow router uses minimal routing and does not load balance across the sidelinks like the flattened butterfly with non-minimal routing. As the BlackWidow network scales to larger numbers of nodes, Rank2.5 and Rank3.5 networks are hybrids that resemble a flattened butterfly at the top level but with folded-Clos subnetworks at the lower levels. The flattened butterfly topology introduced in this paper extends the concept of sidelinks to create a topology where every link is a sidelink and there are no uplinks or downlinks to middle stage routers. Our work also improves upon the BlackWidow network by using global adaptive non-minimal routing to load-balance sidelinks.

The Flat Neighborhood Networks (FNN) [10] is an interconnection network proposed for clusters that is *flat* – i.e. there is only a single router between every pair of nodes. FNN partitions the terminal node bandwidth among the multiple routers and thus, provides a single intermediate hop between any two nodes. However, the network leads to an asymmetric topology and is very limited in its scalability.

To increase the path diversity of the butterfly network, alternative butterfly networks have been proposed. Additional stages can be inserted to the butterfly network [26] but adding additional stages to the butterfly ultimately leads to a Clos network. Dilated butterflies [15] can be created where the bandwidth of the channels in the butterflies are increased. However, as shown in Section 4.3, the network cost is dominated by the links and these methods significantly increase the cost of the network with additional links as well as routers.

Our non-minimal routing algorithms for the flattened butterfly are motivated by recent work on non-minimal and globally adaptive routing. Non-minimal routing was first applied to torus networks and have been shown to be critical to properly load-balancing tori [27, 28]. We show that these principles and algorithms can be applied to the flattened butterfly as well. We also extend this previous work on load-balanced routing by identifying the problem of transient load imbalance in high-radix adaptive routers and show how this problem can be solved by using a sequential allocator in place of a greedy allocator and the CLOS AD routing algorithm.

Patel et al. studied the impact of power in interconnection networks [20]. There have been many research efforts to reduce the power consumption through different techniques [12, 25, 29]. These power saving techniques can also be applied to the flattened butterfly but the flattened butterfly provides additional power savings over other high-radix topologies.

7. CONCLUSION

This paper introduces the *flattened butterfly* topology that exploits recent developments in high-radix routers and global adaptive routing to give a cost-effective network. The flattened butterfly gives lower hop count than a folded Clos and better path diversity than a conventional butterfly. As a result, the flattened butterfly is approximately half the cost of a folded Clos network of identical capacity on load-balanced traffic. On adversarial traffic, the flattened butterfly exploits global adaptive routing to match the cost/performance of the folded Clos.

Using a detailed cost model that accounts for packaging and the effect of distance on channel cost, we have compared the cost of the flattened butterfly to the cost of a folded Clos, a conventional butterfly, and a hypercube network — all with identical capacity. Our analysis shows that the flattened butterfly provides 35-53% cost reduction compared to the folded Clos. The exact reduction depends on network size. Our cost model also shows that the total interconnection network cost is dominated by channel cost — and in particular the cost of long, *global* channels. Our analysis also demonstrates that the flattened butterfly is able to exploit *packaging locality*, unlike the folded-Clos or conventional butterfly.

We evaluate five routing algorithms for the flattened butterfly including both minimal and non-minimal and both adaptive and oblivious. We also compare routing algorithms using both greedy and sequential allocators. We show that non-minimal globally-adaptive routing is necessary to load-balance the topology on adversarial traffic and that global adaptive routing is needed to provide good performance on both benign and adversarial traffic. Our routing studies demonstrate that transient load imbalance occurs in high-radix routers with greedy allocators but that CLOS AD routing algorithm with a sequential allocator overcomes this problem.

Acknowledgments

The authors would like to thank the anonymous reviewers and CVA members for their insightful comments. This work has been supported in part by New Technology Endeavors, Inc. through DARPA subcontract CR03-C-0002 under U.S. government Prime Contract Number NBCH3039003.

8. REFERENCES

[1] Amphenol. http://www.amphenol.com/.

[2] L. N. Bhuyan and D. P. Agrawal. Generalized hypercube and hyperbus structures for a computer network. *IEEE Trans. Computers*, 33(4):323–333, 1984.

[3] K.-Y. K. Chang, J. Wei, C. Huang, S. Li, K. Donnelly, M. Horowitz, Y. Li, and S. Sidiropoulos. A 0.4–4-Gb/s CMOS Quad Transceiver Cell Using On-Chip Regulated Dual-Loop PLLs. *IEEE Journal of Solid-State Circuits*, 38(5):747–754, 2003.

[4] C. Clos. A Study of Non-Blocking Switching Networks. *The Bell System technical Journal*, 32(2):406–424, March 1953.

[5] Cray XT3. http://www.cray.com/products/systems/xt3/.

[6] W. J. Dally. Performance Analysis of k-ary n-cube Interconnection Networks. *IEEE Transactions on Computers*, 39(6):775–785, 1990.

[7] W. J. Dally. Virtual-channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, 1992.

[8] W. J. Dally, P. P. Carvey, and L. R. Dennison. The Avici Terabit Switch/Router. In *Proc. of Hot Interconnects*, pages 41–50, August 1998.

[9] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, San Francisco, CA, 2004.

[10] H. G. Dietz and T.I.Mattox. Compiler techniques for flat neighborhood networks. In *13th International Workshop on Languages and Compilers for Parallel Computing*, pages 420–431, Yorktown Heights, New York, 2000.

[11] Gore. http://www.gore.com/electronics.

[12] E. J. Kim, K. H. Yum, G. M. Link, N. Vijaykrishnan, M. Kandemir, M. J. Irwin, M. Yousif, and C. R. Das. Energy optimization techniques in cluster interconnects. In *Proc. of the 2003 international symposium on Low power electronics and design*, pages 459–464, 2003.

[13] J. Kim, W. J. Dally, and D. Abts. Adaptive Routing in High-radix Clos Network. In *International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'06)*, Tampa, FL, 2006.

[14] J. Kim, W. J. Dally, B. Towles, and A. K. Gupta. Microarchitecture of a high-radix router. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 420–431, Madison, WI, 2005.

[15] C. P. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Trans. Computers*, 32(12):1091–1098, 1983.

[16] R. E. Ladner and M. J. Fischer. Parallel prefix computation. *J. ACM*, 27(4):831–838, 1980.

[17] J. Laudon and D. Lenoski. The SGI Origin: A ccNUMA Highly Scalable Server. In *Proc. of the 24th Annual Int'l Symp. on Computer Architecture*, pages 241–251, 1997.

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

[19] Microprocessor Report. http://www.mdronline.com/.

[20] C. S. Patel, S. M. Chai, S. Yalamanchili, and D. E. Schimmel. Power constrained design of multiprocessor interconnection networks. In *International Conference on Computer Design (ICCD)*, pages 408–416, 1997.

[21] G. Pautsch. Thermal Challenges in the Next Generation of Supercomputers. *CoolCon*, 2005.

[22] G. Pfister. *An Introduction to the InfiniBand Arechitecture (http://www.infinibandta.org)*. IEEE Press, 2001.

[23] S. Scott, D. Abts, J. Kim, and W. J. Dally. The BlackWidow High-radix Clos Network. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, Boston, MA, June 2006.

[24] S. Scott and G. Thorson. The Cray T3E Network: Adaptive Routing in a High Performance 3D Torus. In *Hot Chips 4*, Stanford, CA, Aug. 1996.

[25] L. Shang, L.-S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *International Symposium on High-Performance Computer Architecture (HPCA)*, page 91, 2003.

[26] H. J. Siegel. A model of simd machines and a comparison of various interconnection networks. *IEEE Trans. Computers*, 28(12):907–917, 1979.

[27] A. Singh. *Load-Balanced Routing in Interconnection Networks*. PhD thesis, Stanford University, 2005.

[28] A. Singh, W. J. Dally, A. K. Gupta, and B. Towles. GOAL: A load-balanced adaptive routing algorithm for torus networks. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 194–205, San Diego, CA, June 2003.

[29] V. Soteriou and L.-S. Peh. Design-space exploration of power-aware on/off interconnection networks. In *International Conference on Computer Design (ICCD)*, pages 510–517, 2004.

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

[31] H. Wang, L. S. Peh, and S. Malik. Power-driven Design of Router Microarchitectures in On-chip Networks. In *Proc. of the 36th Annual IEEE/ACM Int'l Symposium on Microarchitecture*, pages 105–116, 2003.

[32] K.-L. J. Wong, H. Hatamkhani, M. Mansuri, and C.-K. K. Yang. A 27-mW 3.6-Gb/s I/O Transceiver. *IEEE Journal of Solid-State Circuits*, 39(4):602–612, 2004.

[33] S. Young and S. Yalamanchili. Adaptive routing in generalized hypercube architectures. In *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 564–571, Dallas, TX, Dec. 1991.