Technology-Driven, Highly-Scalable Dragonfly Topology*

John Kim
Northwestern University
Evanston, IL 60208
jjk12@northwestern.edu

William J. Dally
Stanford University
Stanford, CA 94305
dally@stanford.edu

Steve Scott
Cray Inc.
Chippewa Falls, WI 54729
sscott@cray.com

Dennis Abts
Google Inc.
dabts@google.com

Abstract

Evolving technology and increasing pin-bandwidth motivate the use of high-radix routers to reduce the diameter, latency, and cost of interconnection networks. High-radix networks, however, require longer cables than their low-radix counterparts. Because cables dominate network cost, the number of cables, and particularly the number of long, global cables should be minimized to realize an efficient network. In this paper, we introduce the dragonfly *topology which uses a* group *of high-radix routers as a virtual router to increase the effective radix of the network. With this organization, each minimally routed packet traverses at most one global channel. By reducing global channels, a dragonfly reduces cost by 20% compared to a flattened butterfly and by 52% compared to a folded Clos network in configurations with* $\geq 16K$ *nodes.*

We also introduce two new variants of global adaptive routing that enable load-balanced routing in the dragonfly. Each router in a dragonfly must make an adaptive routing decision based on the state of a global channel connected to a different router. Because of the indirect *nature of this routing decision, conventional adaptive routing algorithms give degraded performance. We introduce the use of selective virtual-channel discrimination and the use of credit round-trip latency to both sense and signal channel congestion. The combination of these two methods gives throughput and latency that approaches that of an* ideal *adaptive routing algorithm.*

1 Introduction

Interconnection networks are a critical component of modern computer systems. From large scale systems [1, 19, 27] to multicore architectures [17, 33], the interconnection network that connects processors and memory modules significantly impacts the overall performance and cost of the system. As processor and memory performance continues to increase, multicomputer interconnection networks are becoming even more critical as they largely determine the bandwidth and latency of remote memory access.

A good interconnection network is designed around the capabilities and constraints of available technology. Increasing router pin bandwidth, for example, has motivated the use of *high-radix* routers [15] in which the increased bandwidth is used to increase the number of ports per router, rather than maintaining a small number of ports and increasing the bandwidth per port. The Cray BlackWidow system [1], one of the first systems to employ a high-radix network, uses a variant of the folded-Clos topology and radix-64 routers [26] — a significant departure from previous low-radix 3-D torus networks [27]. Recently, the advent of economical optical signalling [12, 22] enables topologies with long channels. However, these long optical channels are significantly more expensive than short electrical channels. In this paper, we introduce the *dragonfly* [1] topology that exploits emerging optical signaling technology by grouping routers to further increase the *effective* radix of the network.

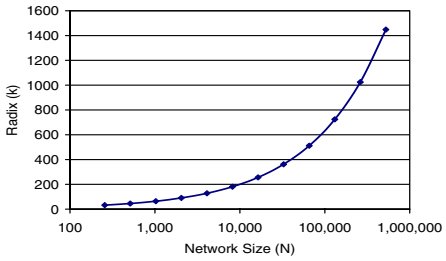


Figure 1. Radix (k) of the routers required to scale the network (N) if only one global hop is required for each packet.

The topology of an interconnection network largely determines both the performance and the cost of the network [8]. Network cost is dominated by the cost of channels, and in particular the cost of the long, *global*, inter-cabinet channels. Thus, reducing the number of global channels can significantly reduce the cost of the network. To reduce global channels without reducing performance, the number of global channels traversed by the average packet must be reduced. The dragonfly topology introduced in this paper reduces the number of global channels traversed per packet using minimal routing to one.

To achieve this unity *global diameter*, very high-radix

*This work was done while John Kim was a Ph.D. student at Stanford University and Dennis Abts was with Cray Inc.

[1]The dragonfly name is used for the topology because of the dragonfly's wide body but narrow wings – similar to the proposed topology with a large group but narrow channels connecting the groups.

1063-6897/08 $25.00 © 2008 IEEE
DOI 10.1109/ISCA.2008.19

cables	distance	data rate	power	E/bit	Optical Technology
Intel Connects Cable [12]	<100m	20Gb/s	1.2W	60pJ	VCSELs, multi-mode fiber
Luxtera Blazar [21]	<300m	42Gb/s	2.2W	55pJ	CMOS Photonics, single-mode fiber
conventional electrical cable [23]	<10m	10Gb/s	20mW	2pJ	–

Table 1. Comparison of the different cables and their characteristics. The comparison is shown for 4x cables and the delay/power consumption are for the active components of the cable.

routers, with a radix of $\sim 2\sqrt{N}$ (where N is the size of the network) are required. [2] While radix 64 routers have been introduced [26], and a radix of 128 is feasible, much higher radices are needed to build machines that scale to 8K - 1M nodes, as shown in Figure 1. To achieve the benefits of a very high radix, we propose using a *group* of routers connected into a subnetwork as one very high radix *virtual router*. This very high effective radix in turn allows us to build a network in which all minimal routes traverse at most one global channel. It also increases the physical length of the global channels, exploiting the capabilities of emerging optical signaling technology.

Achieving good performance on a wide range of traffic patterns on a dragonfly topology requires a routing algorithm that can effectively balance load across the global channels. Global adaptive routing (UGAL) [29], can perform such load balancing if the load of the global channels is available at the source router, where the routing decision is made. With the dragonfly topology, however, the source router is most often not connected to the global channel in question. Hence, the adaptive routing decision must be made based on remote or *indirect* information. The indirect nature of this decision leads to degradation in both latency and throughput when conventional UGAL (which uses local queue occupancy to make routing decisions) is used. We propose two modifications to the UGAL routing algorithm that overcome this limitation with performance results approaching an *ideal* implementation using global information. Adding selective virtual-channel discrimination to UGAL (UGAL$_{VC_H}$) eliminates bandwidth degradation due to local channel sharing between minimal and non-minimal paths. Using credit-round trip latency to both sense global channel congestion and to propagate this congestion information upstream (UGAL$_{CR}$) eliminates latency degradation by providing much stiffer backpressure than is possible using only queue occupancy for congestion sensing.

The rest of the paper is organized as follows. In Section 2, we provide background into signaling technology and develop a cost model for signaling. The dragonfly topology is described in detail in Section 3 and the different routing algorithms are discussed in Section 4. Section 5 provides additional discussion on the topology and comparison to other topologies. Related work is presented in Section 6 and Section 7 presents our conclusions.

[2] A fully connected topology with a concentration of $\sqrt{N}$ is assumed.

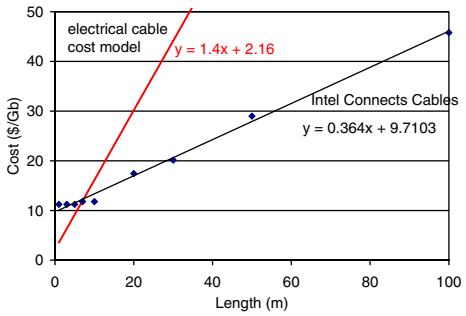


Figure 2. Cost model comparison of active optical cables [12] and electrical cables with repeaters [14].

2 Technology Model

High-radix networks reduce the diameter of the network but require longer cables compared to low-radix networks. In this section, we discuss signaling technology and how the recent development of active optical cables enables high-radix topologies with longer cables. In addition, we present a cost model for cables that will be used to compare alternative topologies in Section 5.

An interconnection network is embedded in a packaging hierarchy. At the lowest level, the routers are connected via circuit boards, which are then connected via a backplane or midplane. One or more backplanes are packaged in a cabinet, with multiple cabinets connected by electrical or optical cables to form a complete system. The global (inter-cabinet) cables and their associated transceivers often dominate the cost of a network. To minimize the network cost, the topology should be matched to the characteristics of the available interconnect technologies.

The maximum bandwidth of an electrical cable drops with increasing cable length because signal attenuation due to skin effect and dielectric absorption increases linearly with distance [6].[3] For typical high-performance signaling rates (10-20Gb/s) and technology parameters, electrical signaling paths are limited to about 1m in circuit boards and 10m in cables. At longer distances, either the signaling rate must be reduced or repeaters inserted to overcome attenuation.

Historically, the high cost of optical signaling limited its use to very long distances or applications that demanded per-

[3] Attenuation in circuit boards (including backplanes) is primarily due to dielectric absorption while in cables skin effect dominates.

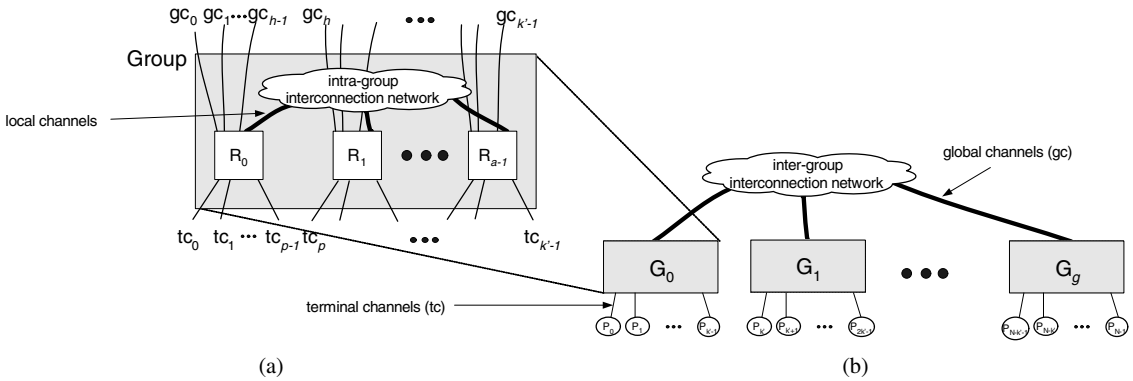


Figure 3. (a) Block diagram of a group (virtual router) and (b) high-level block diagram of a dragonfly topology composed of multiple groups. gc_i corresponds to global channels for inter-group connections and tc_i corresponds to channels connected to the terminals (or processors).

formance regardless of cost. Recent advances in silicon photonics and their application to active optical cables such as Intel Connects Cables [12] and Luxtera Blazar [21, 22] have enabled economical optical interconnect. These active optical cables have electrical connections at either end and EO and OE [4] modules integrated into the cable itself. The characteristics of the active optical cables from Intel and Luxtera as well as a conventional electrical cables are compared in Table 1.

Figure 2 compares the cost of electrical and optical signaling bandwidth as a function of distance. The cost of Intel Connect Cables [12] is compared with the electrical cable cost model presented in [14]. [5] Optical cables have a higher fixed cost (y-intercept) but a lower cost per unit distance (slope) than electrical cables. Based on the data presented here, the crossover point is at 10m. For distances shorter than 10m, electrical signaling is less expensive. Beyond 10m, optical signaling is more economical. The topology proposed in this paper exploits this relationship between cost and distance. By reducing the number of global cables it minimizes the effect of the higher fixed overhead of optical signaling, and by making the global cables longer, it maximizes the advantage of the lower per-unit cost of optical fibers.

3 Dragonfly Topology

The following symbols are used in our description of the dragonfly topology in this section and the routing algorithms in Section 4.

N	Number of network terminals
p	Number of terminals connected to each router
a	Number of routers in each group
k	Radix of the routers
k'	Effective radix of the group (or the virtual router)
h	Number of channels within each router used to connect to other groups
g	Number of groups in the system
q	Queue depth of an output port
q_{vc}	Queue depth of an individual output VC
H	Hop count
Out_i	Router output port i

3.1 Topology Description

The dragonfly is a hierarchical network with three levels: router, group, and system as shown in Figure 3. At the bottom level, each router has connections to p terminals, $a-1$ local channels — to other routers in the same group — and h global channels — to routers in other groups. Hence the radix (or degree) of each router is $k = p + a + h - 1$. A group consists of a routers connected via an intra-group interconnection network formed from local channels (Figure 3(a)). Each group has ap connections to terminals and ah connections to global channels, and all of the routers in a group collectively act as a *virtual router* with radix $k' = a(p + h)$. This very high radix, $k' >> k$ enables the system level network (Figure 3(b)) to be realized with very low global diameter (the maximum number of expensive global channels on the minimum path between any two nodes). Up to $g = ah + 1$ groups ($N = ap(ah + 1)$ terminals) can be connected with a global diameter of one. In contrast, a system-level network built directly with radix k routers would require a larger global diameter.

In a maximum-size ($N = ap(ah + 1)$) dragonfly, there is exactly one connection between each pair of groups. In smaller dragonflies, there are more global connections out of each group than there are other groups. These excess global connections are distributed over the groups with each pair of groups connected by at least $\lfloor \frac{ah+1}{g} \rfloor$ channels.

The dragonfly parameters a, p, and h can have any values. However to balance channel load on load-balanced traffic, the network should have $a = 2p = 2h$. Because each packet traverses two local channels along its route (one at each end

[4] EO : Electrical to Optical, OE : Optical to Electrical

[5] The optical cost is based on prices available at http://shop.intel.com. If purchased in bulk, the prices will likely be lower. The use of single-mode fiber instead of multi-mode fiber may also result in lower cost.

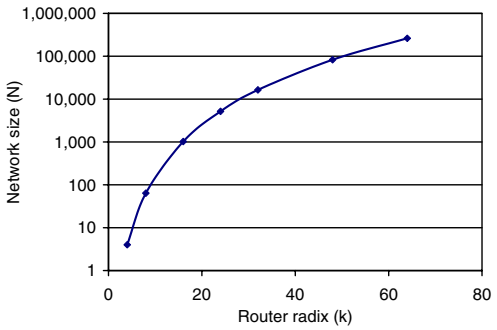


Figure 4. Scalability of the dragonfly topology as the router radix (k) is increased.

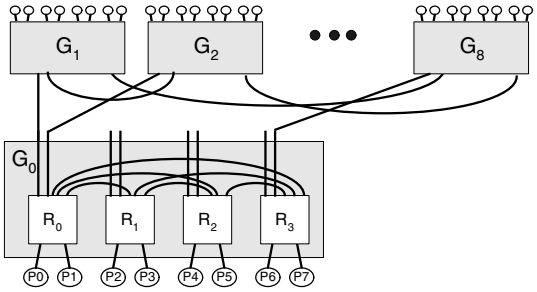


Figure 5. An example block diagram of a dragonfly topology with $N = 72$.

of the global channel) for one global channel and one terminal channel, this ratio maintains balance. Additional details of routing and load-balancing will be discussed in Section 4. Because global channels are expensive, deviations from this 2:1 ratio should be done in a manner that overprovisions local and terminal channels, so that the expensive global channels remain fully utilized. That is, the network should be balanced so that $a \geq 2h, 2p \geq 2h$.

The scalability of a balanced dragonfly is shown in Figure 4. By increasing the effective radix, the dragonfly topology is highly scalable – with radix-64 routers, the topology scales to over 256k nodes with a network diameter of only three hops. Arbitrary networks can be used for the intra-group and inter-group networks in Figure 3. In this paper, we use a 1-D flattened butterfly or a completely-connected topology for both networks. A simple example of the dragonfly is shown in Figure 5 with $p = h = 2, a = 4$ that scales to $N = 72$ with $k = 7$ routers. By using virtual routers, the effective radix is increased from $k = 7$ to $k' = 16$.

3.2 Topology Variations

The global radix, k', can be increased further by using a higher-dimensional topology for the intra-group network. Such a network may also exploit intra-group packaging locality. For example, a 2-D flattened butterfly is shown in Figure 6(a) which has the same k' as the group shown in Figure 5 but exploits packaging locality by providing more bandwidth

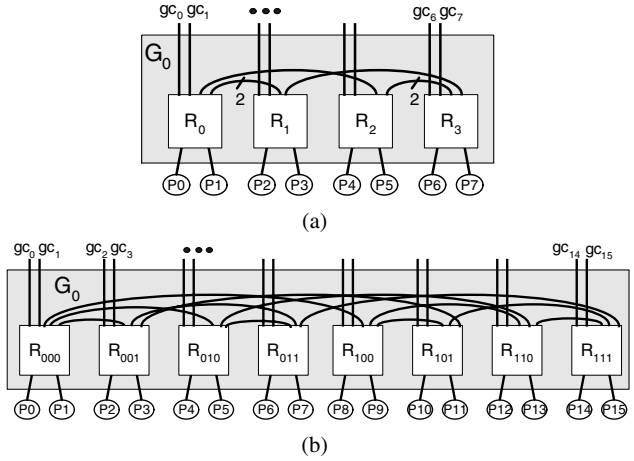


Figure 6. Alternative organization of a group in dragonfly. (a) The same group radix is maintained as in Figure 5 but packaging locality is exploited by providing more bandwidth to the neighboring routers. (b) Increasing group radix by increasing the number of dimensions within the group. The routers within the group are connected by a 3-D flattened butterfly. With $p = 2$, the resulting 3-D flattened butterfly is equivalent to a simple 3D cube.

to local routers. A 3-dimension flattened butterfly is used in Figure 6(b) to increase the effective radix from $k' = 16$ to $k' = 32$ – allowing the topology to scale up to $N = 1056$ using the same $k = 7$ router as in Figure 3.

To increase the terminal bandwidth of a high-radix network such as a dragonfly, channel slicing [8] can be employed. Rather than make the channels wider, which would decrease the router radix, multiple network can be connected in parallel to add capacity. Similarly, the dragonfly topology can also utilize parallel networks to add capacity to the network. In addition, the dragonfly network described so far assumed uniform bandwidth to all nodes in the network. However, if such uniform bandwidth is not needed, bandwidth tapering can be implemented by removing inter-group channels among some of the groups.

4 Routing

In this section, we discuss minimal and non-minimal routing algorithms for the dragonfly topology. We show how global adaptive routing using local information leads to limited throughput and very high latency at intermediate loads. To overcome these problems, we propose new mechanisms to global adaptive routing, which provide performance that approaches an *ideal* implementation of global adaptive routing.

4.1 Routing on the Dragonfly

Minimal routing in a dragonfly from source node s attached to router R_s in group G_s to destination node d attached to

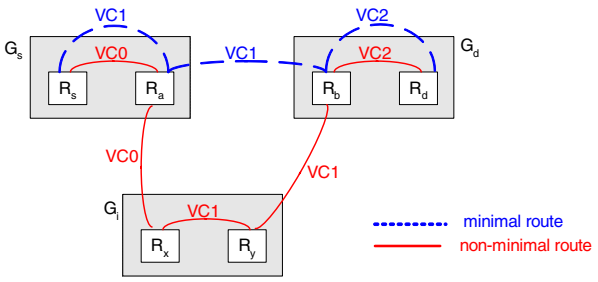


Figure 7. Virtual channel assignment to prevent routing deadlock in a dragonfly topology with both minimal and nonminimal routing.

router R_d in group G_d traverses a single global channel and is accomplished in three steps:

Step 1 : If $G_s \neq G_d$ and R_s does not have a connection to G_d, route within G_s from R_s to R_a, a router that has a global channel to G_d.

Step 2 : If $G_s \neq G_d$, traverse the global channel from R_a to reach router R_b in G_d.

Step 3 : If $R_b \neq R_d$, route within G_d from R_b to R_d.

This minimal routing works well for load-balanced traffic, but results in very poor performance on adversarial traffic patterns.

To load-balance adversarial traffic patterns, Valiant's algorithm [32] can be applied at the system level — routing each packet first to a randomly-selected intermediate group G_i and then to its final destination d. Applying Valiant's algorithm to groups suffices to balance load on both the global and local channels. This randomized non-minimal routing traverses at most two global channels and requires five steps:

Step 1 : If $G_s \neq G_i$ and R_s does not have a connection to G_i, route within G_s from R_s to R_a, a router that has a global channel to G_i.

Step 2 : If $G_s \neq G_i$ traverse the global channel from R_a to reach router R_x in G_i.

Step 3 : If $G_i \neq G_d$ and R_x does not have a connection to G_d, route within G_i from R_x to R_y, a router that has a global channel to G_d.

Step 4 : If $G_i \neq G_d$, traverse the global channel from R_y to router R_b in G_d.

Step 5 : If $R_b \neq R_d$, route within G_d from R_b to R_d.

Figure 7 shows how virtual channels (VCs) [5] are used to avoid routing deadlock. To prevent routing deadlock [7], two VCs are needed for minimal routing and three VCs are required for non-minimal routing. This assignment eliminates all channel dependencies due to routing. For some applications, additional virtual channels may be required to avoid protocol deadlock — e.g., for shared memory systems, separate sets of virtual channels are required for request and reply messages.

4.2 Evaluation

We evaluate the following routing algorithms for the dragonfly topology.

Minimal (MIN) : The minimal path is taken as described in Section 4.1.

Valiant (VAL) [32] : Randomized non-minimal routing as described in Section 4.1.

Universal Globally-Adaptive Load-balanced [29] (UGAL-G, UGAL-L) UGAL chooses between MIN and VAL on a packet-by-packet basis to load-balance the network. The choice is made by using queue length and hop count to estimate network delay and choosing the path with minimum delay. We implement two versions of UGAL.

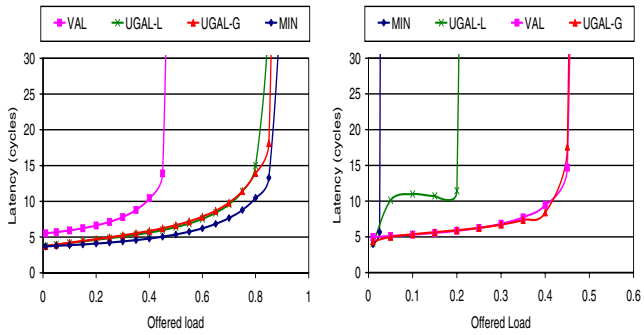
UGAL-L – uses *local* queue information at the current router node.

UGAL-G – uses queue information for all the global channels in G_s — assuming knowledge of queue lengths on other routers. While difficult to implement, this represents an *ideal* implementation of UGAL since the load-balancing is required of the global channels, not the local channels.

Cycle accurate simulations are used to evaluate the performance of the different routing algorithms. We simulate a single-cycle, input-queued router switch but provide sufficient speedup in order to generalize the results and ensure that routers do not become the bottleneck of the network. 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. Unless otherwise noted, the simulation results are shown for dragonfly of size 1K node using $p = h = 4$ and $a = 8$ parameters. Simulations of other size networks follow the same trend and are not presented due to space constraints. Single flit (flow control unit) packets are used to separate the routing algorithm from flow control issues such as the use of wormhole or virtual cut-through flow control. [6] The input buffers are initially assumed to be 16 flits deep. The impact of different buffer sizes is also evaluated.

The different routing algorithms are evaluated using both benign and adversarial synthetic traffic patterns. The use of synthetic traffic pattern allows us to stress the topology and routing algorithm to fully evaluate the network. For benign traffic such as uniform random (UR), MIN is sufficient to provide low latency and high throughput (Figure 8(a)). VAL achieves approximately half of the network capacity because its load-balancing doubles the load on the global channels. Both UGAL-G and UGAL-L approach the throughput of MIN, but with slightly higher latency near saturation. The higher latency is caused by the use of parallel or *greedy* allocation where the routing decision at each port is made in parallel.

[6] Simulations show that larger packets with sufficient buffering to provide virtual cut-through do not change the result trends presented in the paper.



(a) (b)

Figure 8. Routing algorithm comparison on the dragonfly for (a) uniform random traffic and (b) adversarial traffic pattern.

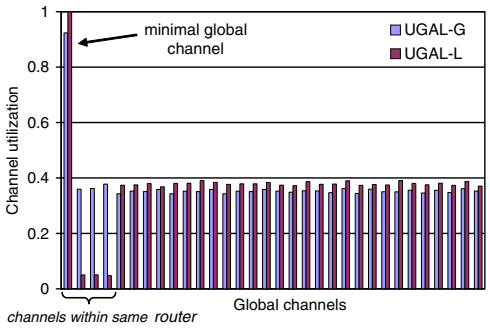


Figure 9. Global channel utilization for the dragonfly topology with UGAL-L and UGAL-G routing using the adversarial traffic pattern. The data is collected from an offered load of 0.2, just prior to the saturation of UGAL-L.

The use of sequential allocation [13] will reduce the latency at the expense of a more complex allocator.

To test the load-balancing ability of a routing algorithm, we use a worst-case (WC) traffic pattern where each node in group G_i sends traffic to a randomly selected node in group G_{i+1}. With minimal routing, this pattern will cause all nodes in each group G_i to send all of their traffic across the single global channel to group G_{i+1}. Non-minimal routing is required to load balance this traffic pattern by spreading the bulk of the traffic across the other global channels.

The evaluation for this WC traffic is shown in Figure 8(b). Because MIN forwards all of the traffic from each group across a single channel, its throughput is limited to $\frac{1}{ah}$. VAL achieves slightly under 50% throughput which is the maximum possible throughput with this traffic. [7] UGAL-G achieves similar throughput as VAL but UGAL-L leads to both limited throughput as well as high average packet latency at intermediate load. In the following section, we show how the *indirect* nature of adaptive routing on the dragonfly leads to performance degradation. We identify the issues with UGAL-L and present mechanisms that can overcome these problems.

4.3 Indirect Adaptive Routing

Adaptive routing on the dragonfly is challenging because it is the global channels, the group outputs, that need to be balanced, not the router outputs. This leads to an indirect routing problem. Each router must pick a global channel to use using only local information that depends only *indirectly* on the state of the global channels. Previous global adaptive routing methods [3, 29, 30] used local queue information, source queues and output queues, to generate accurate estimates of network congestion. In these cases, the local queues were an accurate proxy of global congestion, because they *directly* indicated congestion on the routes they initiated. With the dragonfly topology, however, local queues only sense congestion on a global channel via backpressure over the local channels. If the local channels are overprovisioned, significant numbers of packets must be enqueued on the overloaded minimal route before the source router will sense the congestion. This results in a degradation in throughput and latency as shown earlier in Figure 8(b).

4.3.1 Problem I: Limited throughput

The throughput issue with UGAL-L is due to a single local channel handling both minimal and non-minimal traffic. For example, in Figure 13, a packet in R1 has a minimal path which uses gc_7 and a nonminimal path which uses gc_6. Both paths share the same local channel from R1 to R2. Because both paths share the same local queue (and hence have the same queue occupancy) and the minimal path is shorter (one global hop vs two), the minimal channel will always be selected, even when it is saturated. This leads to the minimal global channel being overloaded and the non-minimal global channels that share the same router as the minimal channel being under utilized. This effect is shown in Figure 9. The first global channel is the minimal global channel, the next three global channels are non-minimal channels that share the same router with the minimal channels ($h = 4$), and the remaining channels are non-minimal channels that share the same group. With UGAL-G, the minimal channel is preferred and the load is uniformly balanced across all other global channels. With UGAL-L, on the other hand, the non-minimal channels on the router that contains the minimal global channel are under utilized – resulting in a degradation of network throughput.

To overcome this limitation, we modify the UGAL algorithm [8] to separate the queue occupancy into minimal and non-minimal components by using individual VCs (UGAL-L$_{VC}$).

```
if   (q_m_vc H_m ≤ q_nm_vc H_nm )
     route minimally;
else
     route nonminimally;
```

[7] If additional buffering is provided, the theoretically expect throughput of 50% is achieved.

[8] The original UGAL routing algorithm can be described as – if ($q_m H_m \leq q_{nm} H_{nm}$) route minimally; else route nonminimally; [29].

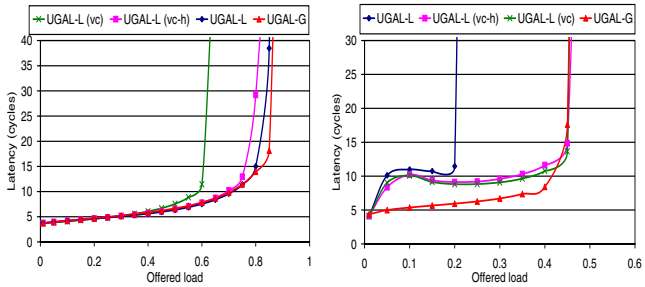


Figure 10. Evaluation of alternative UGAL-L implementation for (a) uniform random traffic and (b) worst-case traffic.

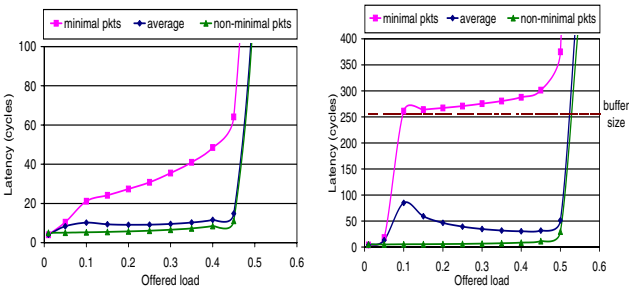


Figure 11. Latency vs. offered load for the dragonfly topology with UGAL-L routing and adversarial traffic pattern with the input buffers of depth (a) 16 and (b) 256.

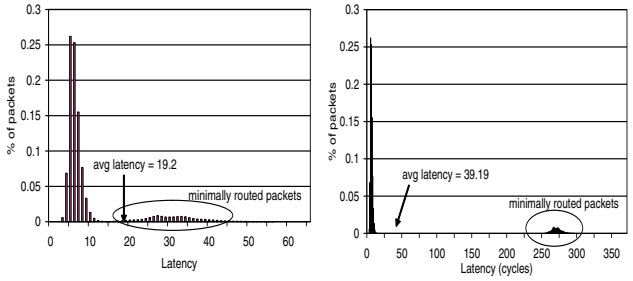


Figure 12. Histogram distribution of average packet latency at an offered load of 0.25 in the dragonfly topology with UGAL-L routing adversarial traffic pattern and the input buffers of depth (a) 16 and (b) 256.

where the subscript m and nm denote the minimal and non-minimal paths. If the VC assignment of Figure 7 is used, $q_{m\_vc} = q(VC1)$ and $q_{nm\_vc} = q(VC0)$.

The modified routing algorithm (UGAL-L$_{VC}$) is compared for both WC and UR traffic in Figure 10. UGAL-L$_{VC}$ matches the throughput of UGAL-G on WC traffic pattern but for UR traffic, the throughput is limited, with approximately 30% reduction in throughput (Figure 10(a)). For the WC traffic where most of the traffic needs to be sent non-minimally, UGAL-L$_{VC}$ performs well since the minimal queue is heavily loaded. However, for load-balanced traffic when most traffic should be sent minimally, individual VCs do not provide an accurate representation of the channel congestion – resulting in throughput degradation.

To overcome this limitation, we further modify the UGAL algorithm to separate the queue occupancy into minimal and non-minimal components *only* when the minimal and non-minimal paths start with the same output port. Our hybrid modified UGAL routing algorithm (UGAL-L$_{VC_H}$) is

```
if (q_m H_m ≤ q_nm H_nm && Out_m ≠ Out_nm ) ||
   (q_m_vc H_m ≤ q_nm_vc H_nm && Out_m = Out_nm )
    route minimally;
else
    route nonminimally;
```

Compared to UGAL-L$_{VC}$, UGAL-L$_{VC_H}$ provides the same throughput on WC traffic pattern but matches the throughput of UGAL-G on UR traffic but resulting in nearly 2× higher latency at an offered load of 0.8, near saturation. For WC traffic, UGAL-L$_{VC_H}$ also results in higher intermediate latency compared to UGAL-G (Figure 10(b)). In the next section, we discuss the issue of higher intermediate latency and a mechanism to provide stiffer backpressure to reduce intermediate latency.

4.3.2 Problem II: Higher intermediate latency

The high intermediate latency of UGAL-L is due to minimally-routed packets having to fill the channel buffers between the source and the point of congestion before congestion is sensed. In Figure 11, we plot the latency of minimally-routed and non-minimally-routed packets as well as the overall average latency.[9] The figure shows that non-minimally routed packets have a latency curve comparable to UGAL-G while minimally-routed packets see significantly higher latency. Figure 11(b) shows that as input buffers are increased, the latency of minimally-routed packets increases and is proportional to the depth of the buffers. A histogram of latency distribution (Figure 12) shows two clear distributions – one large distribution with low latency for the non-minimal packets and another distribution with a limited number of packets but with much higher latency for the minimal packets.

To understand this problem with UGAL-L, in the example dragonfly group shown in Figure 13, assume a packet in R1 is making its global adaptive routing decision of routing either minimally through gc_0 or non-minimally through gc_7. The routing decision needs to load balance global channel utilization and ideally, the channel utilization can be obtained from the queues associated with the global channels, q_0 and q_3. However, q_0 and q_3 queue informations are only available at R0 and R2 and not readily available at R1 – thus, the

[9]The average latency is the *weighted* average between minimal and non-minimal packets.

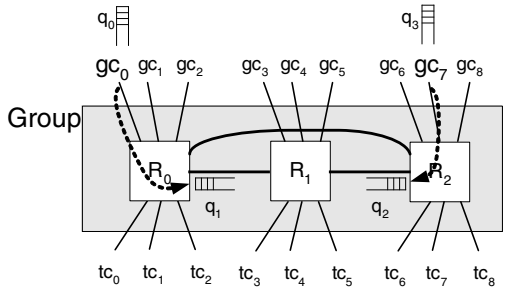


Figure 13. A block diagram of a dragonfly topology to illustrate indirect adaptive routing.

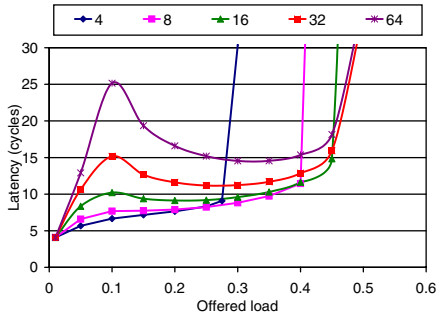


Figure 14. Latency vs. offered load for the dragonfly topology as the amount of input buffers are varied.

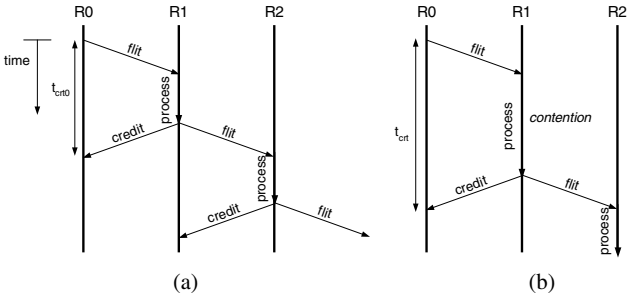


Figure 15. Credit round-trip latency timeline (a) when there is no congestion in the network (t_{crt0}) and (b) when channel between R1 and R2 is congested (t_{crt}).

routing decision can only be made indirectly through the local queue information available at R1. In this example, q_1 reflects the state of q_0 and q_2 reflects the state of q_3. When either q_0 or q_3 is full, the flow control provides backpressure to q_1 and q_2 as shown with the arrows in Figure 13. As a result, in steady-state measurement, these local queue information can be used to accurately measure the throughput. Since the throughput is defined as the offered load when the latency goes to infinity (or the queue occupancy goes to infinity) [8], this local queue information is sufficient. However, q_0 needs to be completely full in order for q_1 to reflect the congestion of gc_0 and allow R1 to route packets non-minimally. Thus, using local information requires sacrificing some packets to properly determine the congestion – resulting in packets being sent minimally having much higher latency. As the load increases, although minimally routed packets continue to increase in latency, more packets are sent non-minimally and results in a decrease in average latency until saturation.

In order for local queues to provide a good estimate of global congestion, the global queues need to be completely full and provide a stiff backpressure towards the local queues. The *stiffness* of the backpressure is inversely proportional to the depth of the buffer – with deeper buffers, it takes longer for the backpressure to propagate while with shallower buffers, a much stiffer backpressure is provided. Simulation results as the buffer size is varied are shown in Figure 14. As the buffer size decreases, the latency at intermediate load is decreased because of the stiffer backpressure. However, using smaller buffers comes at the cost of reduced network throughput. [10]

To overcome the high intermediate latency, we propose using *credit round-trip latency* to sense congestion faster and reduce latency. In credit-based flow control (Figure 17(a)), credit counts are maintained for buffers downstream. As packets are sent downstream, the appropriate credit count is decremented and once the packet leaves downstream router, credits are sent back upstream and the credit count is incremented. The latency for the credits to return is referred to as credit round-trip latency (t_{crt}) and a timeline of zero-load credit round-trip latency (t_{crt0}) is shown in Figure 15(a). If there is congestion downstream, the packet cannot be immediately processed and results in an increase in t_{crt} as shown in Figure 15(b).

The value of t_{crt} can be used to estimate the congestion of global channels. By using this information to *delay* upstream credits, we stiffen the backpressure and more rapidly propagate congestion information up stream. For each output O, $t_{crt}(O)$ is measured and the quantity $t_d(O) = t_{crt}(O) - t_{crt0}$ is stored in a register. Then, when a flit is sent to output O, instead of immediately sending a credit back upstream, the credit is delayed by $t_d(O) - \min[t_d(o)]$. The credits sent across the global channels are not delayed. This ensures that there is no cyclic loop in this mechanism and allows the global channels to be fully utilized.

The delay of returning credits provides the appearance of shallower buffers to create a stiff backpressure. However, to ensure that the entire buffer gets utilized and there is no reduced throughput at high load, the credits needs to delayed by the *variance* of t_d across all outputs. We estimate the variance by finding $\min[t_d(o)]$ value and using the difference. By delaying credits, the upstream routers observes congestion at a faster rate (compared to waiting for the queues to fill up) and leads to better global adaptive routing decisions.

The UGAL-L routing algorithm evaluation using credit latency (UGAL-L$_{CR}$) [11] is shown in Figure 16 for both WC and

[10] The input buffers are usually large to support virtual cut-through flow control for the maximum size packet. For example, YARC router input buffer has 256 flit entries [26].

[11] The UGAL-L$_{CR}$ is implemented on top of UGAL-L$_{VC_H}$.

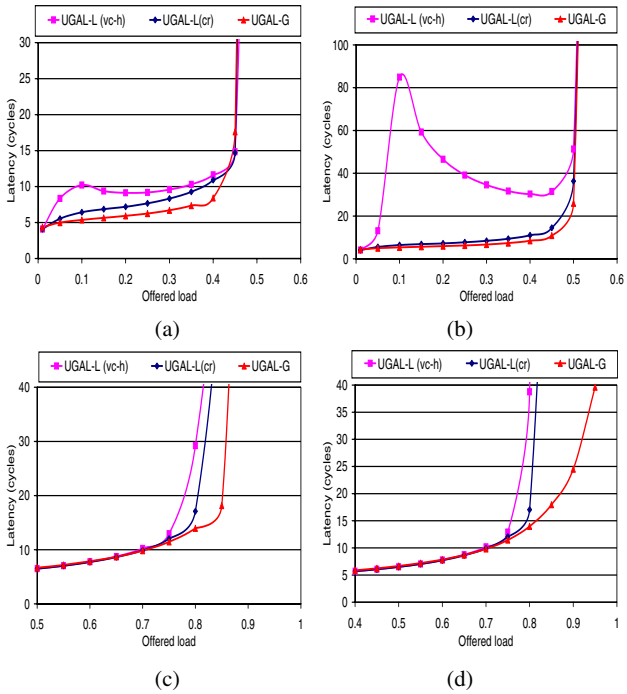


Figure 16. Performance comparison of UGAL-L$_{CR}$ with (a,b) WC traffic and (b,d) UR traffic. The buffer sizes are 16 for (a,c) and 256 for (b,d).

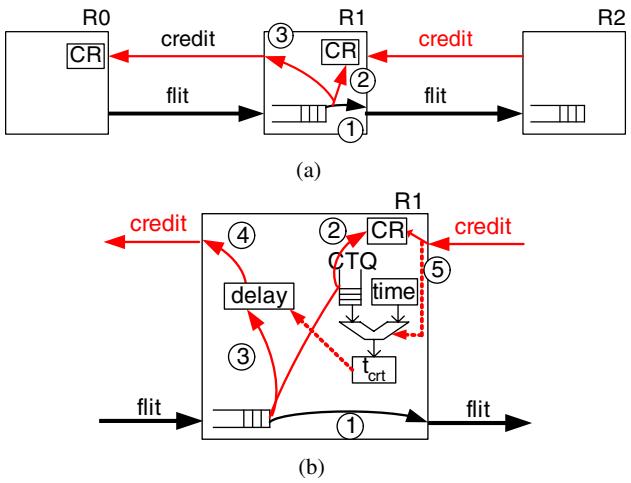


Figure 17. (a) Conventional credit flow control. As packets are sent downstream ①, the output credit count is decremented ② and credits are sent back upstream ③. (b) Modification to the flow control to use credit round trip latency to estimate congestion. In addition to the output credit count being decremented ②, the time stamp is pushed into the credit time queue (CTQ). Before sending the credit back upstream ④, the credit is delayed ③. When downstream credits are received ⑤, credit count (CR) is updated as well as the credit round trip latency (t_{crt}).

UR traffic using buffers of depth 16 and 256. UGAL-L$_{CR}$ leads to significant reduction in latency compared to UGAL-L and approaches the latency of UGAL-G. For WC traffic, UGAL-L$_{CR}$ reduces latency by up to 35% with 16 buffers and up to over 20× reduction in intermediate latency with 256 buffers compared to UGAL-L. Unlike UGAL-L, the intermediate latency with UGAL-L$_{CR}$ is independent of buffer size as shown in Figure 16(a,b). For UR traffic, UGAL-L$_{CR}$ provides up to 50% latency reduction near saturation compared to UGAL-L$_{VC_H}$. However, both UGAL-L$_{CR}$ and UGAL-L$_{VC_H}$ fall short of the throughput of UGAL-G with UR traffic because their imprecise local information results in some packets being routed non-minimally.

The implementation of this scheme results in minimal complexity overhead as the following three features are needed at each router:

- tracking credits individually to measure t_{crt}
- registers to store t_d values
- a delay mechanism in returning credits

The amount of storage required for t_d is minimal as only $O(k)$ registers are required. The credits are often returned by piggybacking on data flits and delaying credits to wait for the transmission of the next data flit upstream is required. The proposed mechanism only requires adding additional delay.

As for tracking individual credits, credits are conventionally tracked as a pool of credits in credit flow control – i.e., a single credit counter is maintained for each output VC and increments when a credit is received. The implementation of UGAL-L$_{CR}$ requires tracking each credit individually. This can be done by pushing a timestamp on the tail of a queue each time a flit is sent, as shown in Figure 17(b) with the use of a credit timestamp queue (CTQ), and popping the timestamp off the head of the queue when the corresponding credit arrives. Because flits and credits are 1:1 and maintain ordering, the simple queue suffices to measure round-trip credit latency. The depth of the queue needs to be proportional to the depth of the data buffers but the queue size can be reduced to utilize *imprecise* information [13] to measure congestion – e.g., by having a queue which is only 1/4 of the data buffer size, only one of four credits are tracked to measure the congestion.

5 Cost Comparison

In this section, we provide a comparison of the dragonfly topology to a flattened butterfly. We also provide a cost comparison of the dragonfly to alternative topologies using the cost model presented in Section 2.

The flattened butterfly topology reduces network cost by removing intermediate routers and channels [14]. As a result, the flattened butterfly reduces cost by approximately 50% compared to a folded-Clos [4, 20] on balanced traffic. The dragonfly topology extends the flattened butterfly by increasing the effective radix of the routers to further reduce the cost and increase the scalability of the network.

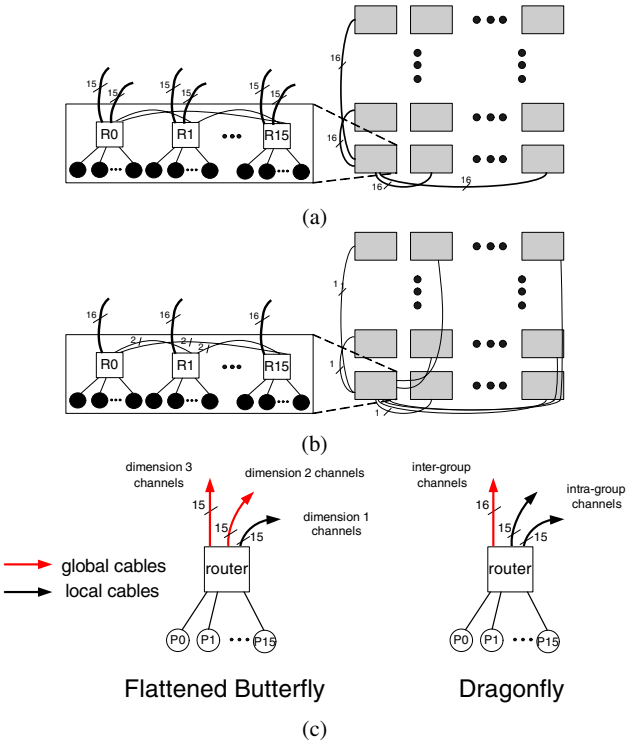


Figure 18. Topology comparison of a 64K network for (a) 3-D flattened butterfly and (b) dragonfly. The squares represents a small number of cabinets required to connect the 256 nodes and the global channels are only shown for lower-left corner group. The routers used in the topologies are compared in (c).

A comparison of dragonfly and flattened butterfly networks of 64K nodes is shown in Figure 18(a,b). The group size of the dragonfly is 16 routers (256 terminals) and this is also the size of each dimension of the flattened butterfly. To scale to 64K nodes, the flattened butterfly requires two additional dimensions, each of size 16, while the dragonfly, with its effective radix of 256, connects all of the groups in a single, large dimension. Thus, while both topologies provide the same amount of global bisection bandwidth, the dragonfly requires only half the number of global cables compared to the flattened butterfly. As shown in Figure 18(c)), the flattened butterfly uses 50% of the router ports for global channels while the dragonfly uses only 25% of the ports for global channels. In addition, the dragonfly provides better scalability because the group size can be increased to scale the network whereas scaling the flattened butterfly requires adding additional dimensions. The two topology comparisons are summarized in Table 2. With the hop count nearly identical, the dragonfly trades off longer global cables for smaller number of global cables required to provide a more cost-efficient topology better matched to emerging signaling technologies.

Figure 19 compares the cost of the dragonfly, flattened butterfly, Clos, and 3D-torus networks as a function of the number of terminal nodes. For short cables (<8m) we use the electrical cable cost model (from [14]). For cables longer than 8m we use the active optical cable cost model (Section 2). We assume the use of radix-64 routers for the high-radix networks and adjust the cost of the router appropriately for the low-radix 3-D torus network. For the dragonfly network we use a group size of 512 nodes.

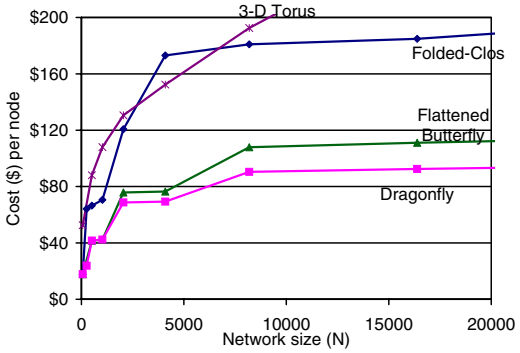


Figure 19. Cost comparison of the dragonfly topology to alternative topologies.

For networks up to 1K nodes, all routers are fully connected and the dragonfly is identical to a 1-D flattened butterfly – thus, the cost of the two networks are identical. For a topology that is fully connected, there is no cost benefit of attempting to using virtual routers as it will only increase the cost. For larger networks, the dragonfly is more scalable not only because it has higher effective radix but the group size is twice as large as the dimension size for the flattened butterfly which leads to lower cost. For networks up to 4K nodes, the dragonfly provides approximately 10% savings. This is because the dragonfly has a shorter average cable length than the flattened butterfly at these small sizes. For larger networks (>4K nodes), the dragonfly provides approximately 20% cost savings over a flattened butterfly since the dragonfly has fewer long, global cables.

The 3-D torus network results in short cables and does not require the use of optical signalling. However, as shown in Figure 19, the cost of the network is significantly higher than the other topologies because of the larger number of cables needed to support the high network diameter. For a network of size 1K, the dragonfly reduces cost by approximately 62% while at a network of size 8K, the dragonfly provides a cost savings of only 47% as the dragonfly topology requires the use of the more expensive optical cables. However, as the network size increases, the cost benefits of dragonfly exceeds 60%. Compared to the folded-Clos, the dragonfly provides over 50% cost savings. The reduction of network cost in the dragonfly also translates to reduction of power as shown in prior work [14].

topology	diameter		cable length	
	minimal	nonminimal	avg	max
flattened butterfly	$h_l + 2h_g$	$2h_l + 4h_g$	E/3	E
dragonfly	$2h_l + h_g$	$3h_l + 2h_g$	2E/3	$2E^{\dagger}$

Table 2. Topology comparison of the dragonfly and the flattened butterfly topology. h_l is a local hop, h_g is a global hop and E corresponds to the length of a dimension of the system layout. †The maximum length cable for dragonfly can be reduced to $\sqrt{2}E$ if cables are connected diagonally.

6 Related Work

Many topologies have been previously proposed and we compare the dragonfly to some relevant topologies in this section. The Scalable Opto-Electronic Network (SOENet) [11] was proposed to exploit emerging optical technology. SOENet is constructed by forming subnetworks and connecting multiple subnetworks through global switches. The proposed dragonfly topology shares a similar goal of exploiting optical technology and the structure of creating subnetworks, or *groups*. However, the dragonfly topology extends the previous work by exploiting high-radix routers and packaging locality to create very high-radix virtual routers from groups of routers. In addition, the dragonfly topology is *flat* in hierarchy compared to SOENet since there are no intermediate routers. Unlike SOENet, in the dragonfly topology all routers are directly connected to end terminals, resulting in a reduced network diameter and network cost.

Many hierarchical topologies have been previously proposed [9, 18, 19]. The dragonfly topology, with an intra- and an inter-group network, can also be referred to as a hierarchical topology. However, the dragonfly topology is fundamentally different from previously proposed hierarchical networks in that the radix of the network is increased, thereby providing more global bandwidth, while also reducing network diameter. Previously proposed hierarchical networks have been built as tree-structures. This approach introduces a bandwidth bottleneck and increases hop count and latency as the packets traverse up the hierarchy. Other product-form networks, such as the cube-connected cycles [25], have also been proposed, but these networks do not exploit the benefits of increasing the effective radix of the networks.

The significance of signalling technology for optimal topology choice was demonstrated through a topology optimization tool [10]. The availability of economical optical signaling, as described in Section 2, significantly changes the cost model and enables a topology such as the dragonfly with longer channels. Optical interconnects have the potential to replace electrical interconnects due to their higher bandwidth and lower latency [24]. To exploit optical technology, complete optical networks have been proposed [2, 16, 28]. However, because of the difficulty of buffering, it becomes very costly to implement a purely optical network. In addition, many optical networks utilize very low-radix networks to simplify the switch and can not exploit the benefits of high-radix routers. The RAPID architecture (Reconfigurable and scalable All-Photonic Interconnect for Distributed-shared memory) [16] is a hierarchical optical network that uses passive components. RAPID networks require longer latency to communicate between different clusters, and the use of passive network limits the scalability of the topology.

Routing has been well studied on a k-ary n-cube network. On such networks, when an adaptive routing decision is made based on the injection queues, Singh [29] showed that although optimal throughput is achieved, the routing algorithm results in high latency at intermediate loads. This is similar to what was observed in Section 4.3.2. To overcome this, Singh proposed the use of channel queues to make adaptive routing decisions [31]. However, the indirect nature of the dragonfly topology prevents the use of channel queues to reduce latency. The approach taken in this work demonstrates how to make adaptive decisions using credit round-trip latency to provide a faster mechanism for sensing congestion and reducing latency.

7 Conclusion

This paper has introduced the dragonfly topology which uses a group of routers as a virtual router to increase the effective radix of the network, and hence reduce network diameter, cost, and latency. Because it reduces the number global cables in a network, while at the same time increasing their length, the dragonfly topology is particularly well suited for implementations using emerging active optical cables — which have a high fixed cost but a low cost per unit length compared to electrical cables. Using active optical cables for the global channels, a dragonfly network reduces cost by 20% compared to a flattened butterfly and by 52% compared to a folded Clos network of the same bandwidth.

This paper has also introduced two new variants of global adaptive routing that overcome the challenge of *indirect* adaptive routing presented by the dragonfly. A dragonfly router must make a routing decision based on the state of a global channel attached to a different router in the same group. Conventional global adaptive routing algorithms that use local queue occupancies to infer the state of this remote channel give degraded throughput and latency. We introduce the selective use of virtual channel discrimination to overcome the bandwidth degradation. We also introduce the use of credit round-trip latency to both sense and signal channel congestion. The combination of these two techniques gives a global

adaptive routing algorithm that approaches the performance of an *ideal* algorithm with perfect knowledge of remote channel state.

Acknowledgments

The authors would like to thank the anonymous reviewers for their insightful comments and David Black-Schaffer for his feedback on the paper. This work has been supported in part by the National Science Foundation under Contract CCF0702341, in part by Cray, and in part by the Semiconductor Research Corporation under Contract SRC2007-HJ-1591.

References

[1] D. Abts, A. Bataineh, S. Scott, G. Faanes, J. Schwarzmeier, E. Lundberg, T. Johnson, M. Bye, and G. Schwoerer. The Cray BlackWidow: A Highly Scalable Vector Multiprocessor. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis (SC'07)*, Reno, NV, Nov. 2007.

[2] R. D. Chamberlain, M. A. Franklin, and C. S. Baw. Gemini: An Optical Interconnection Network for Parallel Processing. *IEEE Transactions on Parallel and Distributed Systems*, 13(10):1038–1055, 2002.

[3] D. Chiou, L. R. Dennison, and W. J. Dally. Adaptive source routing and packet processing. United States Patent 20050100035, May 2005.

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

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

[6] W. J. Dally and J. W. Poulton. *Digital systems engineering*. Cambridge University Press, New York, NY, 1998.

[7] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, 36(5):547–553, 1987.

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

[9] S. Dandamudi and D. Eager. Hierarchical Interconnection Networks for Multicomputer Systems. *IEEE Transactions on Computers*, 39(6):786–797, 1990.

[10] A. K. Gupta and W. J. Dally. Topology optimization of interconnection networks. *IEEE Computer Architecture Letters*, 5(1), 2006.

[11] A. K. Gupta, W. J. Dally, A. Singh, and B. Towles. Scalable Opto-Electronic Network (SOENet). In *Proc. of Hot Interconnects*, pages 71–75, Stanford, CA, Aug. 2002.

[12] Intel Connects Cables. http://www.intel.com/design/network/ products/optical/cables/index.htm/.

[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, Nov. 2006.

[14] J. Kim, W. J. Dally, and D. Abts. Flattened Butterfly : A Cost-Efficient Topology for High-Radix Networks. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 126–137, San Diego, CA, June 2007.

[15] 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, June 2005.

[16] A. K. Kodi and A. Louri. Design of a High-Speed Optical Interconnect for Scalable Shared-Memory Multiprocessors. *IEEE Micro*, 25(1):41–49, 2005.

[17] P. Kongetira, K. Aingaran, and K. Olukotun. Niagara: A 32-Way Multithreaded Sparc Processor. *IEEE Micro*, 25(2):21–29, 2005.

[18] J. M. Kumar and L. M. Patnaik. Extended hypercube: A hierarchical interconnection network of hypercubes. *IEEE Trans. Parallel Distrib. Syst.*, 3(1):45–57, 1992.

[19] 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.

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

[21] Luxtera Blazar LUX5010. http://www.luxtera.com/ products_blazar.htm.

[22] Luxtera Inc. White Paper: Fiber will displace copper sooner than you think, Nov. 2005.

[23] R. Palmer, J. Poulton, W. J. Dally, J. Eyles, A. M. Fuller, T. Greer, M. Horowitz, M. Kellam, F. Quan, and F. Zarkeshvari. A 14mW 6.25Gb/s Transceiver in 90nm CMOS for Serial Chip-to-Chip Communications. In *IEEE Int'l Solid-State Circuits Conf., Digest of Tech. Papers (ISSCC)*, pages 440–441, 2007.

[24] T. Pinkston. Design considerations for optical interconnects in parallel computers. In *Massively Parallel Processing Using Optical Interconnections*, pages 306–322, Cancun, Mexico, 1994.

[25] F. P. Preparata and J. Vuillemin. The cube-connected cycles: a versatile network for parallel computation. *Commun. ACM*, 24(5):300–309, 1981.

[26] 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)*, pages 16–28, Boston, MA, June 2006.

[27] 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.

[28] A. Shacham and K. Bergman. Building Ultralow-Latency Interconnection Networks Using Photonic Integration. *IEEE Micro*, 27(4):6–20, 2007.

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

[30] 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.

[31] A. Singh, W. J. Dally, A. K. Gupta, and B. Towles. Adaptive channel queue routing on k-ary n-cubes. In *SPAA '04: Proceedings of the sixteenth annual ACM symposium on Parallelism in algorithms and architectures*, pages 11–19, New York, NY, USA, 2004. ACM Press.

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

[33] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. B. III, and A. Agarwal. On-Chip Interconnection Architecture of the Tile Processor. *IEEE Micro*, 27(5):15–31, 2007.