Indirect Adaptive Routing on Large Scale Interconnection Networks

Nan Jiang
Stanford University
Stanford, CA 94305
njiang37@stanford.edu

John Kim
KAIST
Daejeon, Korea
jjk12@cs.kaist.ac.kr

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

ABSTRACT

Recently proposed high-radix interconnection networks [10] require global adaptive routing to achieve optimum performance. Existing *direct* adaptive routing methods are slow to sense congestion remote from the source router and hence misroute many packets before such congestion is detected. This paper introduces *indirect* global adaptive routing (IAR) in which the adaptive routing decision uses information that is not directly available at the source router. We describe four IAR routing methods: credit round trip (CRT) [10], progressive adaptive routing (PAR), piggyback routing (PB), and reservation routing (RES). We evaluate each of these methods on the dragonfly topology under both steady-state and transient loads. Our results show that PB, PAR, and CRT all achieve good performance. PB provides the best absolute performance, with 2-7% lower latency on steady-state uniform random traffic at 70% load, while PAR provides the fastest response on transient loads. We also evaluate the implementation costs of the indirect adaptive routing methods and show that PB has the lowest implementation cost requiring <1% increase in the total storage of a typical high-radix router.

Categories and Subject Descriptors

B.4.3 [**Input/Output and Data Communications**]: Interconnections (subsystems); B.4.4 [**Input/Output and Data Communications**]: Performance Analysis and Design Aids; B.8.2 [**Performance and Reliability**]: Performance Analysis and Design Aids

General Terms

Algorithms, Performance

Keywords

Interconnection networks, routing, dragonfly

ISCA'09, June 20–24, 2009, Austin, Texas, USA.

1. INTRODUCTION

As processor and memory performance continues to improve, the interconnection network is becoming the critical component of a modern computer system. In scalable multiprocessors, the interconnection network plays a vital role in system performance as the remote memory access latency and bandwidth are established by the network. An efficient interconnection network is critical to achieve cost- and energy-efficiency in large-scale systems, such as supercomputers and more recently data centers [15], as the number of components continues to increase.

Previously, low-radix routers (i.e., routers with a small number of ports) were used to build low-radix topologies such as 2-D or 3-D mesh or torus networks [17]. However, recent work has shown that as router pin bandwidth increases, *high-radix* routers [12, 16] more efficiently use the additional pin bandwidth to reduce network cost and latency. Efficient high-radix topologies such as the flattened butterfly [9] and the dragonfly [10, 11] have been proposed to reduce latency and increase bandwidth for a given cost compared to a traditional Clos [3] network.

To realize the advantages of efficient high-radix topologies like the flattened butterfly and dragonfly, however, requires global adaptive routing [18]. Minimal routing on these networks can result in very low bandwidth for worst-case traffic patterns, and oblivious routing [20] gives performance on benign traffic patterns no better than that of a Clos network. With global adaptive routing, these topologies offer a 2× performance advantage over a Clos network on benign traffic while matching the performance of a Clos on worst-case traffic.

The original global adaptive routing proposals [19, 18] used local channel queues to sense global congestion. When congestion is sensed, some packets are routed non-minimally using Valiant's algorithm [20] to balance load. This approach works well in networks with *stiff* backpressure. In such networks, remote congestion is rapidly reflected in the length of local queues. In some recently proposed networks such as the dragonfly, however, there may be several deep queues between the point of congestion and the source router where the global routing decision is made. Many packets must be sent over the minimal route to fill these queues before congestion is reflected in the local queues. By the time congestion is sensed locally, too many packets have been routed minimally resulting in very high latency.

For networks with soft backpressure, *directly* sensing congestion at the source router is too slow to react to congestion. For such networks *indirect* adaptive routing, where conges-

tion is sensed using information not directly available at the source router is required. Several researchers have started experimenting with indirect adaptive routing. Credit-round-trip routing [10], for example, uses credit latency to signal remote congestion. Regional congestion awareness [7] sends explicit congestion signals in a 2D mesh on-chip network.

In this paper, we introduce three new indirect adaptive routing methods: progressive adaptive routing (PAR), piggyback routing (PB), and reservation routing (RES). We evaluate these methods along with credit-round-trip routing (CRT) on both steady-state and transient loads in a dragonfly network. Each of these four methods uses a different mechanism to sense non-local congestion and improve the accuracy of adaptive routing decisions

Specifically, the contributions of this work are:

- Introducing three new indirect adaptive routing methods.
- Introducing a methodology for evaluating routing methods on transient loads. Past work has considered only steady-state loads.
- Evaluation of four indirect adaptive routing methods on the dragonfly network for both steady-state and transient loads.
- Evaluating the effect of packet size, channel latency, and network dimension on the performance of indirect adaptive routing methods.
- Evaluating the cost of implementing four indirect adaptive routing methods.

Our results show that PB, PAR, and CRT all achieve good performance — within 7% of the lowest latency at 70% load on steady-state traffic — and respond to transient traffic changes within 100 cycles. PB provides the best absolute performance with 2-7% lower latency on steady-state uniform random traffic at 70% load, while PAR has the fastest response time during transient traffic changes. Our results also show that PB has the lowest implementation cost requiring <1% increase in the total storage of a typical high-radix router.

The remainder of the paper is organized as follows. In Section 2 we provide the background and motivation for this work. In Section 3 we describe the four indirect adaptive routing methods. Section 4 presents the evaluation setup and results are presented in Section 5. Insights into indirect adaptive routing are presented in Section 6. We end with related work and conclusion in Sections 7 and 8.

2. MOTIVATION

2.1 Dragonfly Networks

The hierarchical packaging of interconnection networks affects the cost of communication at each level. At the lowest level of the hierarchy, routers and nodes are connected via backplane printed circuit boards (PCBs). These components are often packaged within a cabinet, and multiple cabinets are interconnected through medium-length (2-10m), electrical links. To scale to a large number of nodes, long (>10m) optical cables are used to provide global channels. The cost per unit bandwidth of a channel increases with the level of hierarchy: short channels on a PCB offering the lowest cost and long optical links the highest cost.

The dragonfly topology [10] is designed to exploit the availability of high-radix routers and to match the packaging/interconnect constraints of a large-scale system. The dragonfly topology is a three-tiered network which matches the three levels of the packaging hierarchy. As shown in Figure 1(a), at the lowest level, each router is connected to p network nodes. These connections are typically made via backplane PCB links. At the second level, a *group* of a routers are connected using a local network. These connections are made using medium-length electrical links within a cabinet or a small group of cabinets. In this paper we consider primarily the dragonfly topology where the local network is a complete connection (a 1-dimensional flattened butterfly), that is where each router has $a-1$ local connections with other routers in the group. Larger groups can be built by using higher-dimensional flattened butterflies for the local network. At the highest level, each router has h global channels connecting to other groups using long optical interconnects. Therefore, the radix of each router is $p+a+h-1$, and each group has a total of ah global channels. There can be up to $ah+1$ groups in a global network with a diameter of one, as shown in Figure 1(b), giving a total network size of $ap(ah+1)$ nodes.

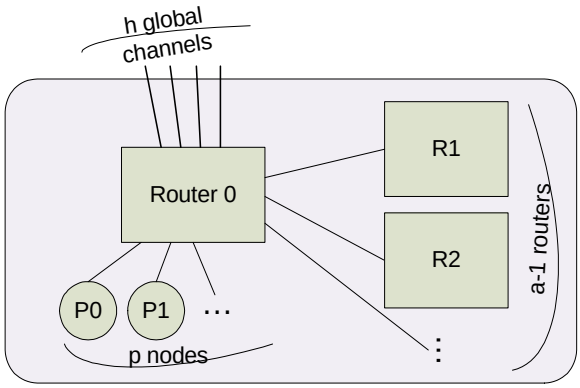
In a maximum-sized dragonfly network, a single global channel connects each pair of groups. A minimally routed packet takes a local hop in the source group, a global hop across an inter-group channel, and a local hop in the destination group, resulting in a 2:1 ratio of local hops to global hops. To ensure that the expensive global channels are fully utilized, twice as much local bandwidth as global bandwidth must be provided out of each router: $ab_l \geq 2hb_g$ (where b_l is the bandwidth of a local channel and b_g is the bandwidth of a global channel). Overprovisioning of local bandwidth contributes to the soft backpressure exhibited by these networks.

2.2 Dragonfly Routing

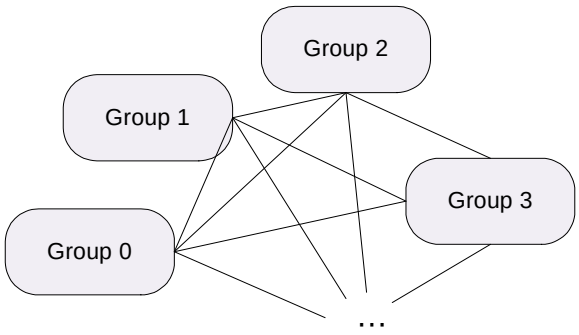
For benign traffic patterns — which themselves balance load across network channels — a dragonfly network is best routed using minimal routing (MIN). MIN routing proceeds in three steps. First, a packet routes locally from the source node to the global channel leading to the destination group. Next, the packet traverses the global channel. Finally, the packet routes in the destination group to the destination node. With minimal routing, each packet traverses exactly one global channel, and, given one-dimensional local networks, exactly two local channels.

While minimal routing yields maximum throughput and minimum latency for benign traffic, it can result in load imbalance on adversarial traffic. Consider, for example the case where every node in group A sends traffic to a corresponding node in group B. Traffic from all pa nodes in A attempt to traverse the single global channel from A to B, leading to significant congestion.

To avoid congestion on such adversarial traffic, one can use an oblivious non-minimal routing algorithm such as Valiant's algorithm [20] (VAL). With VAL, a packet first routes to a randomly selected intermediate group and then to the final destination. In the general case, a packet routed with VAL traverses exactly two global channels and three local channels (with one-dimensional local networks). By ran-



(a) Local network within a group



(b) Global network with a diameter of one

Figure 1: The dragonfly topology

domizing the intermediate destination, VAL gives optimal performance on worst-case traffic patterns but halves the throughput and doubles the latency on benign traffic patterns.

The Universal Globally-Adaptive Load-balanced routing (UGAL) [18] chooses between MIN and VAL on a packet by packet basis depending on the state of the network to minimize the delay of the packet. The delay of a route is estimated by the product of path queue length (Q) and hop count (H). UGAL routes a packet minimally if the following inequality is satisfied:

$$Q_{min} \times H_{min} \leq Q_{val} \times H_{val} + T \quad (1)$$

where T is a routing threshold constant used to filter out transient load imbalances on the minimal path or to bias UGAL's preference towards MIN or VAL. The threshold constant is not part of the original UGAL algorithm and was added to our implementation to balance performance between benign and adversarial traffic patterns. It is described in detail in section 6.1. In the dragonfly network, since the global channels limit the network bandwidth and dominate network latency, we can simplify the original UGAL decision (Equation 1) by using only the number of global hops: one for MIN and two for VAL making the inequality:

$$Q_{min} \leq 2Q_{val} + T \quad (2)$$

Since global channels limit network performance, ideally we would use the global channel queue length in the routing equation. However, the source router does not always have direct access to the global channel queue length and hence uses the length of its local queues as proxies for the global channels. This approximation works well in networks with stiff backpressure but leads to high latency in networks with soft backpressure. This scenario is shown in Figure 2, where $q2$ and $q3$ are used to approximate $q0$ and $q1$, respectively. In order for the source router to sense congestion on the minimal global channel, $q0$ must be filled before $q2$ reflects this congestion. The packets used to fill $q0$ experience very high latency resulting in high average latency on adversarial patterns.

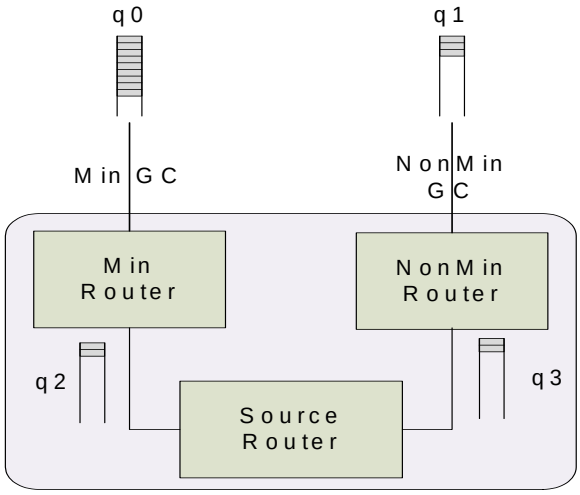


Figure 2: Congestion on the global channels not directly connected to the source router ($q0$,$q1$) must be inferred through local channels ($q2$,$q3$). $q2$ does not reflect congestion until $q0$ is filled.

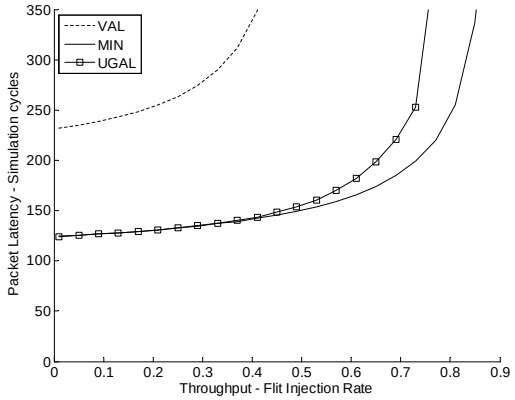
Figure 3 compares the performance of MIN, VAL, and UGAL on both uniform random (UR) traffic, a benign traffic pattern, and worst-case[1] (WC) traffic, an adversarial traffic pattern, on a dragonfly network with $p = h = 4$ and $a = 8$. Channel latency is 10 cycles for local channels and 100 cycles for global channels, and router input buffers are 256 flits deep. Figure 3(a) shows that MIN and UGAL both give nearly optimal performance on benign traffic, while using VAL doubles the latency and halves the saturation throughput. Figure 3(b) shows that VAL gives optimal performance on worst-case traffic, while MIN gives very poor throughput. UGAL matches the throughput of VAL on worst-case traffic but results in substantially higher latency because of the packets that are sacrificed to fill the minimal queues before congestion can be sensed.

The indirect adaptive routing methods described in this paper enable routers to sense congestion quickly — without requiring intermediate queues to back up to the source — and hence avoid the high latency shown in Figure 3(b) for UGAL.

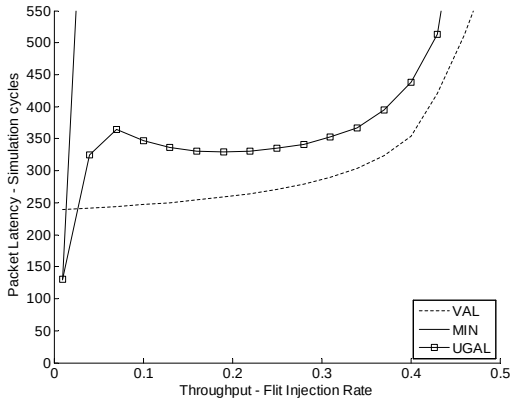
3. INDIRECT ADAPTIVE ROUTING (IAR)

This section describes the four IAR methods. Each method decides whether to route a packet minimally or non-minimally

[1]WC traffic pattern for the dragonfly is when all nodes in group g_i send traffic to group g_{i+1}.



(a) Basic routing method on UR traffic



(b) Basic routing method on WC traffic

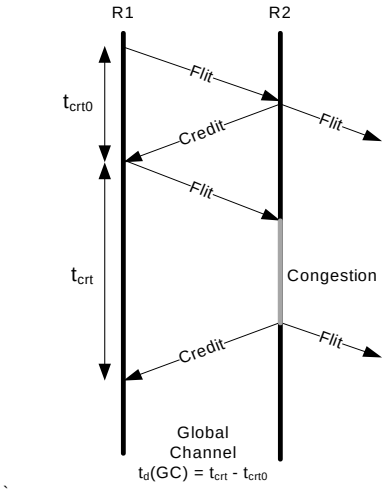
Figure 3: Three basic routing methods on the dragonfly network

using information not directly available at the source router.

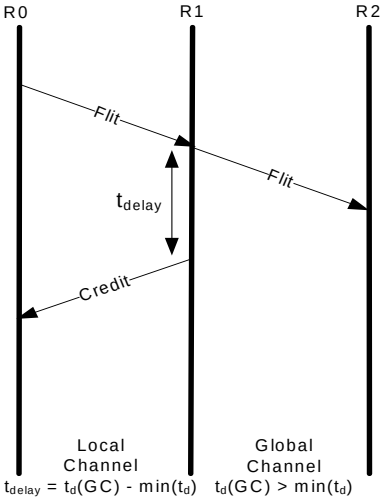
3.1 Credit Round Trip (CRT)

Kim et al. introduced the use of credit round trip (CRT) delay to reduce the latency observed when using UGAL on a dragonfly network [10]. They showed that delaying when credits are returned over local channels can be used to signal congestion on global channels by stiffening backpressure. Figure 4(a) shows the timing of the typical credit-based flow control. A flit departs router $R1$ and once it arrives at $R2$, advances immediately at router $R2$. The credit for this flit returns to $R1$ in t_{crt0}, the zero-load credit round trip latency. If a flit is delayed by t_d because of congestion at $R2$, the credit round trip latency for this flit (t_{crt}) is increased by t_d, and hence the congestion, can be measured by subtracting t_{crt0} from the observed t_{crt}.

Based on the congestion sensed, credit round trip latency can be used to signal congestion as shown in Figure 4(b). For each global channel GC, we measure $t_d(GC)$ as described above to quantify the congestion on GC. Each time a flit is released to GC, the credit corresponding to this flit is delayed by $t_d(GC) - \min(t_d)$, where $\min(t_d)$ is the minimum delay measured for global channels on that router. Delaying the return credit creates the illusion of shallower buffers and stiffer backpressure to the upstream router. As a consequence, the UGAL algorithm in the upstream router is able to adapt properly in making its routing decision. We use $\min(t_d)$ as the delay threshold to ensure the credit delay is adjusted dynamically based on the load of the network. Credit delay is only applied to credits traveling over local channels. Credits returning on the global channels are not delayed to avoid degrading the utilization of global channels.



(a) Definition of t_{crt}, t_{crt0} and t_d



(b) Local credits are delayed based on t_d

Figure 4: Credit round-trip timeline

3.2 Progressive Adaptive Routing (PAR)

Progressive adaptive routing (PAR) operates by re-evaluating the decision to route minimally at each hop in the source group using Equation 2. Any decision to route non-minimally, at the source router or at a subsequent hop, is permanent and is not revisited again. PAR handles the case where a global link is congested but this information has not yet propagated back to the source router. The packet will first route minimally until it encounters the conges-

tion. Then, once the congestion is encountered, the packet is routed non-minimally. Since the routing decision is not re-evaluated after a packet leaves the source group, the maximum number of route re-evaluations is equal to the hop count in the source group of the network. Because decisions to route non-minimally are not re-evaluated, there is no danger of oscillations between minimal and non-minimal routing decisions.

PAR wastes network resources when a minimal routing decision is later reversed. If a packet routes one or more hops minimally, then encounters a congestion and starts routing non-minimally, the resources expended on the minimal hops are wasted — increasing load on the local channels in the source group. However, because local channels are typically overprovisioned, this effect should not impact overall performance.

PAR also requires that each router supports one additional virtual channel to avoid deadlock. This is because a channel dependency is created between channels used to route minimally before congestion is detected and channels used to route non-minimally after congestion is detected.

3.3 Piggyback (PB)

The piggyback (PB) method broadcasts link state information of the global channels to all adjacent routers. A link state bit vector is *piggybacked* on every packet and broadcast on idle channels. Each router maintains the most recent link state information for every global channel in its group. The routing decision is made using both this global state information and the local queue depth. Only if the global state information indicates the minimal global channel is uncongested *and* Equation 2 is true will a packet be routed minimally. Otherwise the packet is routed non-minimally. The UGAL equation must be checked before routing a packet minimally to avoid congestion in the local group — which cannot be detected with the global channel link state.

To reduce the size of the broadcast link state, the congestion level of each global channel is compressed into a single bit (S_{GC})according to the following equation:

$$S_{GC} = Q_{GC} > 2 \times \bar{Q} + T \tag{3}$$

If S_{GC} is true, global channel GC is congested, otherwise it is uncongested. To compute congestion we compare Q_{GC} to $\bar{Q}$, the average queue length of other global channels on the same router. Similarly to Equation 1, a routing threshold T is added to the comparison to smooth out transient load imbalances on the minimal channels.

3.4 Reservation (RES)

The reservation (RES) method operates by first attempting to reserve bandwidth on the minimal global channel and then, if the reservation is successful, sending the packet over that channel. For every global channel GC, a reservation count R is maintained for the number of flits that have reserved GC but not yet traversed it. To make the reservation, the source router sends a *reservation flit* to the minimal global channel router. The packet being routed is buffered at the source router until the reservation flit returns with a routing indication. The reservation flit carries the average of the reservation counters ($\bar{R}$) of the source router. The minimal global channel's reservation level is compared with the average using following inequality:

$$R_{GC} \leq 2 \times \bar{R} + T \tag{4}$$

If the inequality is true, R_{GC} is incremented by the number of flits in the packet, and a positive indication is returned to the source router causing the packet to route minimally. If the inequality is false, R is left unchanged, and a negative indication is returned to the source router causing the packet to route non-minimally. When a minimally routed flit traverses a global channel, the reservation counter is decremented by 1.

RES makes very accurate routing decisions because the reservation counts reflect exactly the minimal routing commitment of each global channel. There is no time delay on the link state information as with the other methods. However, the overhead of RES is considerable. A sizeable buffer is needed in the source router to hold packets awaiting reservation, packet latency is increased by the round-trip time of the reservation flit, and the reservation flits themselves can create a significant load on the local channels in the source group. The channel load of the reservation flits becomes less significant as packet size is increased.

4. EVALUATION SETUP

4.1 Simulator

We evaluate the four indirect adaptive routing methods on a cycle-accurate simulation of a dragonfly network (Section 2.1) with $p = h = 4$, $a = 8$, and one-dimensional local networks. We simulate a single-cycle input queued router and add 10 cycles of latency for local channels (to model realistic router pipeline delay) and 100 cycles of latency for global channels (to model time of flight over long optical fibers). To avoid deadlock, we configure the routers for CRT, PB, RES with three virtual channels [5], PAR routers with 4 virtual channels, and MIN routers with 2 virtual channels. Each global virtual channel has a 256-flit buffer, while each local virtual channel has a 32-flit buffer. Simulations showed that input buffer depth does not significantly impact IAR performance as long as there is sufficient buffering to cover the channel round-trip latency. Unless otherwise stated, each network packet contains 10 flits. Packets are injected into the network using a Bernoulli process. We empirically adjusted the routing threshold for each routing method: 3 packets for PAR and RES, 5 packets for PB, and 1 packet for CRT. Unless otherwise stated, these constants are used for all simulations.

In the steady-state simulations, a single traffic pattern is used. For each latency-throughput data point, 10,000 cycles are run to warm-up the network before collecting network statistics for 50,000 cycles to capture the network steady state. In the transient evaluations, the network is warmed up using the first traffic pattern for 10,000 cycles then switched to the second traffic pattern for the remainder of the simulation. Transient statistics are collected on a cycle-by-cycle basis when the traffic switches.

4.2 Traffic Patterns

We evaluate the IAR methods using uniform random (UR) and worst case (WCn) traffic patterns. With UR, each node is equally likely to send to any other node. With WCn, each node in group i sends all of its traffic to nodes in group $i+n \bmod g$, where $g = ah+1$ is the number of groups in the network. In addition to UR and WCn, 1000 randomly generated permutations are used to test the general reliability of the IAR methods and to find pathological traffic patterns.

5. RESULTS

5.1 Steady State Traffic Performance

Figures 5(a) and 5(b) show that the IAR methods all give good performance on steady-state traffic. On uniform traffic, PB and PAR closely match the performance of minimal routing until injection rates become very high. The CRT method begins to deviate from the ideal performance much earlier, but still maintains stable performance. By using 10-flit packets, RES performance is comparable to that of other IAR methods. At low injection rates, RES's latency is approximately 20 cycles higher due to the reservation flit round-trip latency (two 10-cycle local channels). On worst-case traffic, all four methods start with a lower latency than VAL — because they can route traffic minimally at low loads — and then converge to the performance of VAL as load is increased.

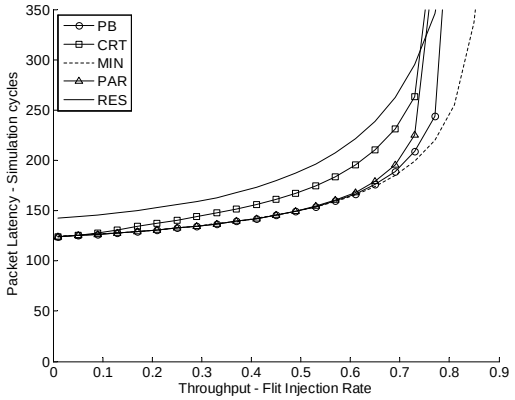
On both traffic patterns, PB has the lowest latency under load because the broadcast link state information results in more accurate routing decisions. For PAR on UR traffic at high injection rates, the transient load imbalance triggers reversals of the minimal routing decisions. This causes additional load on the local channels and leads PAR to saturate earlier than PB. On WC traffic, PAR wastes resources as some packets take a minimal hop (consuming bandwidth and adding latency) before congestion is sensed and the packets are re-routed non-minimally. For both traffic patterns, CRT makes much less accurate decisions than PB and PAR, resulting in higher latency and earlier saturation.

The WC performance of each IAR methods can be slightly improved at the expense of UR performance by *decreasing* the routing threshold constants in the routing equations. We discuss threshold tuning in more detail in Section 6.1. Despite the fact that IAR performance can be *tuned* by the routing threshold, PB has the best performance for either traffic pattern, indicating that it will always outperform CRT or PAR in this network setup.

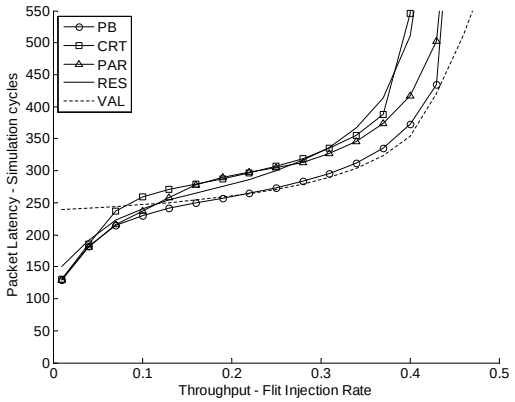
Figure 6 shows a histogram of packet latency of PB, PAR, CRT, RES, and VAL on 1000 randomly generated permutation traffic patterns at a 40% load. We also simulated MIN with the permutation traffic patterns, but all simulations ended in saturation. The IAR methods completed all random permutations without saturating. Three IAR methods, PB, PAR, and CRT, have very similar performance; there is only a 5% difference in their average packet latencies. All three methods were significantly better than VAL. The best IAR method, PB, has a 43% lower average latency than VAL. The variation in latency across the traffic patterns is very small, indicating that these three IAR methods are stable under a variety of traffic patterns. The RES method also has a lower average latency than VAL, but its average latency is higher than other IAR methods due to reservation flit latency. RES's latency variation is much higher due to the reservation flits. The minimal routing of the reservation flits within the source group causes congestion and results in higher average packet latency on some permutation traffic patterns.

5.2 Transient Traffic Response

Because real traffic patterns are characterized by transients, in this section we measure the performance of PB, PAR, and CRT on a cycle-by-cycle basis as traffic is switched from one traffic pattern to another at 35% load.



(a) Indirect adaptive routing on UR traffic



(b) Indirect adaptive routing on WC traffic

Figure 5: Four indirect adaptive routing methods in a dragonfly network with 1D local networks

5.2.1 Case 1: Uniform Random to Worst Case

Figure 7 shows the cycle-by-cycle packet latency (top panel) and fraction of packets routing non-minimally (bottom panel) when the traffic pattern is switched from UR to WC1 at time zero.

Both PAR and CRT transition very smoothly to their WC steady state latency within 40 cycles. PB takes about 80 cycles to settle, and there is a significant latency overshoot in the interim period. From the bottom panel we see that PAR has the fastest response time. It is able to sense congestion at the minimal global channels on the fly and reverse the initial minimal routing decisions. The bandwidth and latency penalty of these decision reversals cause PAR to have slightly higher latency than CRT during the switch. CRT has a fast response time and a smooth transition because of its low routing threshold (1 packet). It can immediately respond to any imbalances between the queue level of the channels. Because PB's routing threshold (5 packets) is much higher than CRT, it cannot quickly differentiate the load imbalance caused by WC traffic. Furthermore, it takes time for piggybacked global congestion information to arrive at the routers due to local channel latency. If we reduced PB's routing threshold, its transient results would be nearly identical to that of CRT.

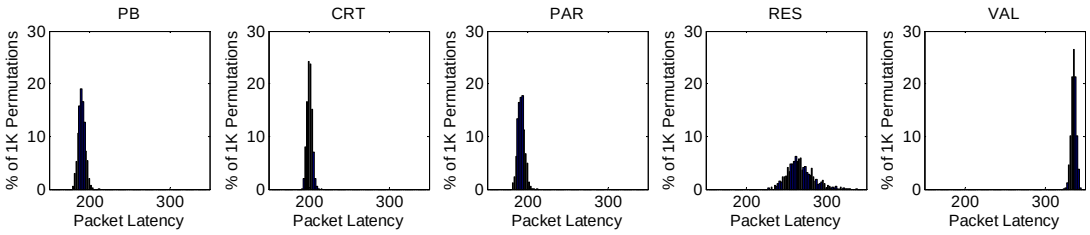


Figure 6: Histogram of average packet latency from 1000 randomly generated permutation traffic patterns at 40% injection rate

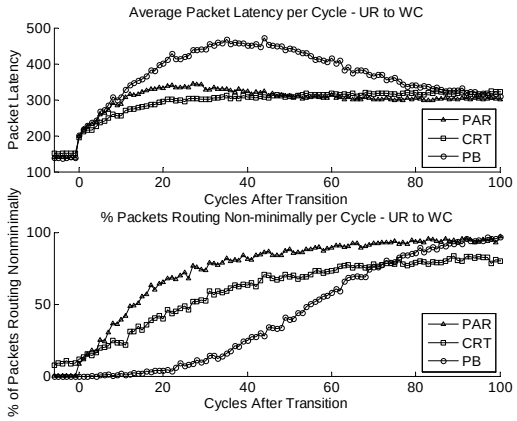


Figure 7: Transient effect of traffic pattern change on indirect adaptive routing at high load (35% injection rate). Uniform Random to Worst Case

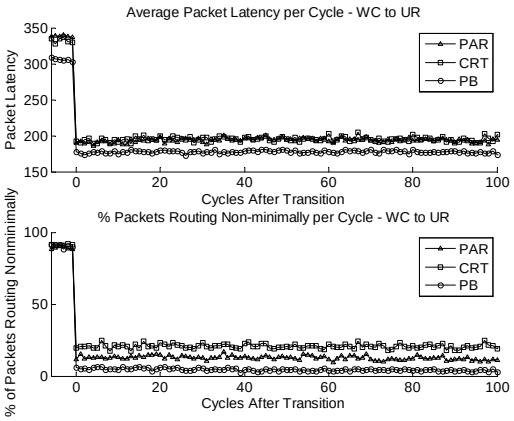


Figure 8: Transient effect of traffic pattern change on indirect adaptive routing at high load (35% injection rate). Worst Case to Uniform Random

5.2.2 Case 2: Worst Case to Uniform Random

Figure 8 shows the cycle-by-cycle transient response when the traffic pattern is switched from WC1 to UR. Under WC1, only few channels in the network are congested. When traffic is switched to UR, most packets choose other, uncongested channels and are routed minimally. A small fraction, $\frac{1}{ah+1}$, of packets, whose minimal channels were congested under WC1, continue to route non-minimally. As the queue of the congested channels drain, nearly all packets route minimally.

5.2.3 Case 3: Worst Case 1 to Worst Case 10

Figure 9 shows the transient response when switching from WC1 to WC10. The minimal global channels used by these two WC traffic patterns are on different routers. In cycle 0, when the traffic pattern switches to WC10, the minimal channels used by WC10 are uncongested and most packets route minimally. These packets quickly congest the minimal channels, and the adaptive routing methods respond by resuming non-minimal routing. From the latency plot, we see similar initial response from all three methods. PB has a smaller latency overshoot than case 1, while CRT results in a higher latency that does not settle after 100 cycles.

The latency behavior of the routing methods can be explained by the bottom panel. PAR again has the fastest response time, achieving steady state after 20-30 cycles. Compared to PB's response time in case 1, PB shows a slightly faster reaction in case 3. This is because the network is more loaded due to the non-minimal routing caused by the initial WC1 traffic. Higher network load is able to trigger the higher PB routing threshold more quickly. PB's quicker response results in the smaller latency overshoot. In contrast, the CRT response time is slower than case 1 despite having a lower threshold than PB. After the traffic pattern switch, the minimal global channel routers of the first traffic pattern will continue to delay returning credits until its t_d returns to normal. This creates the illusion that these routers are still congested, causing some packets to route minimally instead. During case 1, under UR traffic, no router in the network is delaying its credits, allowing CRT to respond faster to the second traffic pattern.

6. DISCUSSION

6.1 Routing Threshold

The routing equations in Sections 2.2 and Section 3 all contain a routing threshold constant. As mentioned previously, we have chosen this constant in our simulations to balance the IAR performance on UR and WC traffic patterns. We determined the value of this constant for each method empirically through numerous simulations. Figure 10 shows the effect of different threshold values on PB routing for different traffic patterns. When the threshold constant is decreased to zero, the network throughput for UR is reduced by 10% and the average packet latency begins to deviate from ideal routing at a much lower injection rate. This is caused by too many packets routing non-minimally. However, for WC traffic, where most of the packets need to be

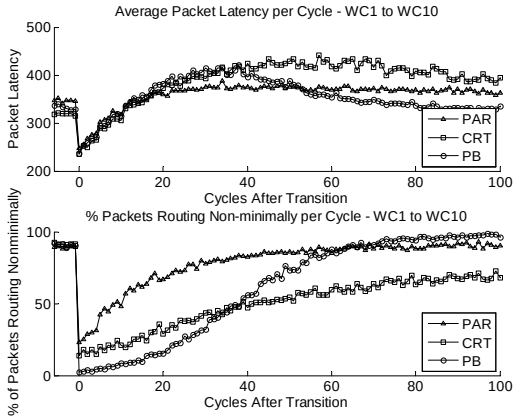
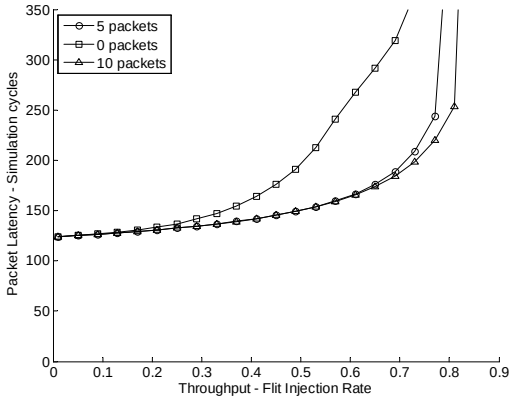
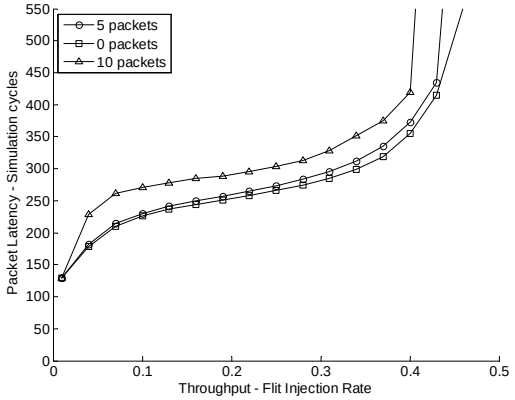


Figure 9: Transient effect of traffic pattern change on indirect adaptive routing at high load (35% injection rate). Worst Case 1 to Worst Case 10



(a) Threshold induced variations – PB on UR traffic



(b) Threshold induced variations – PB on WC traffic

Figure 10: Effect of different routing threshold on PB routing

sent non-mimally, there is a slight gain in performance as latency is reduced by 5% near saturation. In contrast, the high threshold of 10 packets improves UR performance at the cost of WC performance. Thus, a threshold of 5 packets was used to provide a compromise. The routing threshold has similar effects on all other routing methods.

Transient load imbalance is the fluctuation of the channel queue levels caused by the random nature of the injection process. Even with the uniform random traffic pattern, as the network load increases, the amount of transient load imbalance increases in the network. As a general principle, the routing threshold must be high enough to filter out the transient load imbalances while low enough to quickly respond to the real load imbalance created by adversarial traffic. Therefore, the IAR methods always saturate earlier than the ideal on UR traffic unless a very high threshold is set. Under adversarial traffic, the accuracy of the IAR method determines the appropriate threshold. The more accurate the method, the higher the threshold can be set and still achieve good adversarial performance. In the results we have shown thus far, PAR's and PB's thresholds are higher than CRT's because their indirect adaptive mechanisms are more accurate.

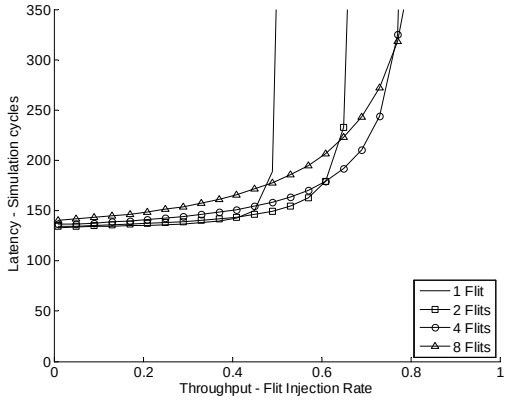
The routing threshold is measured in units of packets, while the queue level used in the routing equations are in units of flits. This is because, while the credit count used to estimate the queue levels is measured in flits, routing decisions are made on a packet by packet basis. The magnitude of the transient imbalance grows linearly with the length of the packet. Therefore, if the network uses longer packets, the threshold in the routing equation will grow proportionately compared to the rest of the equation.

6.2 Effect of Packet Size On IAR

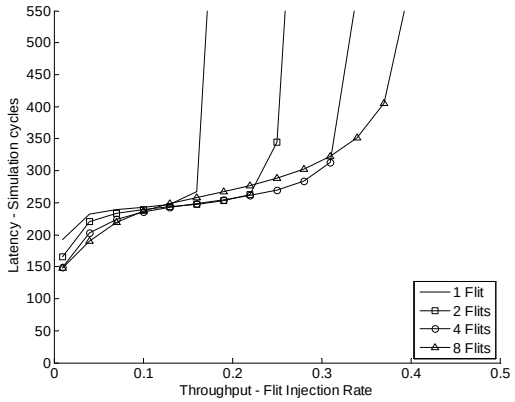
The simulation results shown in this paper have used 10-flit packets. As a result, the RES performance is competitive relative to other IAR methods. However, the performance and the throughput of RES is highly dependent on the packet size as the cost of the reservation flit is amortized across the length of the packet (Figure 11). With smaller packets, the throughput of RES is significantly degraded by up to 50% with single flit packets as the reservation flits consume a significant amount of bandwidth – thus, RES performance is only comparable to other IAR methods with long packets. Regardless of the packet length, RES will always result in higher latency due to the latency added by the reservation flit.

In contrast, using longer packets has a slightly negative effect on CRT, PAR, and PB. In the previous simulations (in Figure 5), these three IAR methods deviate from ideal routing at higher injection rates due to incorrect decisions caused by transient load imbalance. With longer packets, each bad decision has a higher penalty since more flits will route through the undesirable path.

In addition, these three methods rely on channel-queue level by using credits to evaluate adaptive routing. However, with large packets, using only credits to estimate channel-queue levels can be misleading as credits are updated with flit granularity while routing decisions are made with packet granularity. For example, with 10-flit packets, assume packet A decides to route minimally. In the following cycle, if packet B in the same router needs to make an adaptive routing decision, packet B will only observe the head flit of A that was transmitted minimally and not the 9 other body flits that are scheduled to be sent minimally as well – thus, possibly creating further congestion by deciding to route minimally. To overcome this problem, a counter was



(a) RES using different packet sizes – UR traffic



(b) RES using different packet sizes – WC traffic

Figure 11: Effect of packet sizes on RES routing method

added at each output port that is the sum of the credits for the downstream buffer minus the number of flits already committed to be routed through the port but not yet transmitted. All of the simulations shown in the paper used this feature to help with long packets.

6.3 Effect of Local Network Dimensionality

As the dimensionality of the local networks used to realize each group of a dragonfly increases, indirect adaptive routing becomes even more necessary. In an n-dimensional flattened butterfly local network, the minimal global channel may be n hops away from the source router. As n increases, the amount of intermediate queueing increases and the backpressure becomes softer – causing direct adaptive routing to respond even more slowly to remote congestion. However, IAR methods are also adversely affected by high dimensionality with the increase in indirectness.

In multi-dimension local networks, the credit round trip delay, $t_d(LC)$, of each local channel LC must also be measured in addition to the global channels. Routers two or more hops away from a global channel must infer congestion using $t_d(LC)$. However, inferring global congestion through $t_d(LC)$ adds a lot of noise and uncertainty. The CRT method becomes much less accurate as dimensionality is increased.

With more local dimensions, RES will add more overhead to the local network. Reservation flits need to travel more hops to acquire reservations. With higher dimensional local networks, it is even more important that the cost of reservations be amortized over long packets. Alternatively, multiple reservation requests to the same global channel can be combined to reduce reservation flit overhead. However, this will require additional hardware to combine and split reservation flits.

The complexity of PB routing also increases with n-dimension intra-group networks as the global congestion information needs to be sent to routers n-hops away. Each router not only needs to transmit its own global channel congestion information but also transmit global channel information that it receives from its neighbors to ensure all routers have the congestion information for all global channels. This results in an increase in the amount of piggybacked data. The overhead of PB can be reduced by combining multiple global channel congestion information into a single bit while reducing the accuracy of PB.

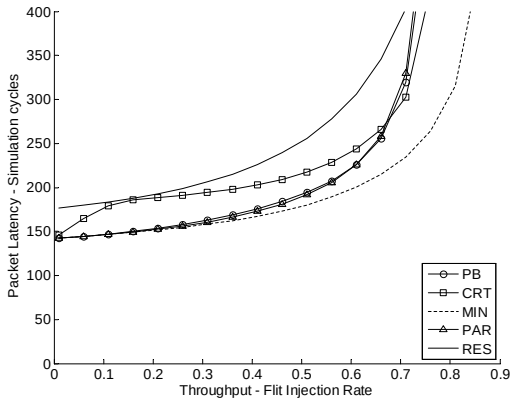
Under PAR, the number of re-evaluations increases with the number of local dimensions. A packet could go through several minimal local hops before ultimately being routed non-minimally. This increases packet latency and the load on the local networks. PAR should still maintain good accuracy at multiple dimensions since it can avoid congestion dynamically.

Figure 12 shows simulation results of the IAR methods on a dragonfly network using 2D flattened butterflies for intra-group networks. Simulation is setup with $p = h = 3$, $a = 36$. The CRT routing threshold has been lowered to 0 packets to help with its performance. The other methods' thresholds are unchanged. Under UR traffic, the IAR methods saturate 10% earlier than the ideal MIN routing. PB and PAR have similar levels of performance while the higher RES latency is caused by the reservation flits. At lower injection rates, CRT has a higher latency than PB and PAR because of its lower routing threshold. Under WC traffic, CRT performs poorly, as it is unable to infer accurate global channel information through the credit round trip. The RES performance is also much less than optimal. The 1 to 10 reservation flit to data flit ratio is too high for this new network setup. PB and PAR provide the best performance under WC traffic. PAR has a higher packet latency at lower injection rates because of wasted hops. However, its ability to adjust for congestion en route to destination results in higher saturation throughput. In general, the IAR methods do not perform as well with 2D flattened butterfly intra-group networks.

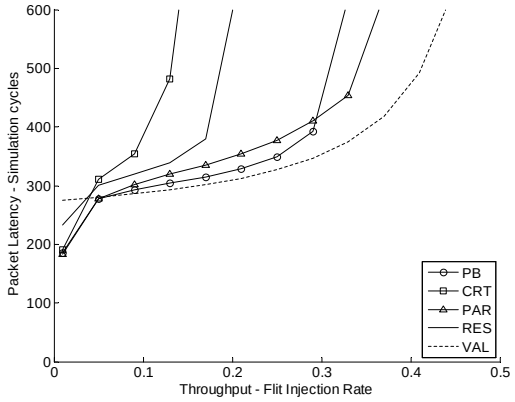
6.4 Effect of Channel Latency

The channel latency in all simulations is set to 10 cycles for local channels and 100 cycles for global channels based on previously published large scale systems [16]. All IAR methods can be used in networks with arbitrary channel latencies, but some aspects of IAR are affected by channel latency.

CRT uses the zero-load credit round-trip latency, t_{crt0}, in estimating channel congestion and is impacted by the channel latency. The congestion information distributed through piggybacked data in the PB method is also dependent on the intra-group channel latency. The reservation latency and the size of the waiting buffer for RES, as well as the extra hops



(a) Indirect adaptive routing on UR traffic with 2D local networks



(b) Indirect adaptive routing on WC traffic with 2D local networks

Figure 12: Four indirect adaptive routing methods in a dragonfly network with 2D local networks

caused by PAR, are also directly proportional to the local channel latency.

Considering the results shown in previous sections, we predict that an increase in channel latency will impact RES and PAR more than CRT and PB. For steady state simulations, channel latency does not affect CRT and PB because the congestion information stays relatively constant. The steady state network latency of RES and PAR are affected by longer channels due to reservation latency and extra hop latency.

6.5 Piggyback Rate and Channel Congestion Compression

In the default PB setup, the global channel congestion level is piggybacked on every packet, and when a channel is idle, an information flit carrying the congestion data is sent. To reduce the overhead of the piggybacked data, we reduced the rate of piggybacking when the channel is utilized but still send information flits when a channel is idle. We found that even when the rate of piggybacking is reduced to once per 100 packets, the steady state plots were nearly identical to the default PB setup. At low injection rates, the network channels are largely idle, and there are many information flits exchanging between the routers. At higher injection rates, fewer information packets are sent. But when the network has reached a steady state, the congestion level of the global channels does not change. Only a handful of information flits, and piggybacked data are necessary to keep the network under saturation at higher injection rates.

The other aspect of PB is the compression of the global channel congestion level. By default, the congestion information is compressed to one bit using Equation 3. We have demonstrated in the previous sections that the single bit representation is sufficient for different traffic loads and traffic patterns. Since Equation 3 uses the average of global channel queue levels as the basis of comparison, the routing algorithm dynamically adjusts with the network load.

However, a side effect of single bit representation is the decision granularity. Under WC traffic, if a minimal global channel's congestion bit transitions from 1 to 0, other routers, when they receive the transition, will route all their packets minimally. Due to channel latency, the minimal global channel does not see the wave of incoming minimal packets for many cycles. It cannot inform the other routers to stop sending minimally. PB's local decision mitigates the sudden switch to minimal routing, but its response time can be slow. Additional bits of accuracy on the global channel congestion level would provide more granularity on the global decision making, but will cost additional piggybacking overhead.

6.6 Implementation Cost

Finally, we compare the cost of implementation of the different IAR methods for the dragonfly network described in Section 4. Since the allocators and the crossbars are unaffected by IAR implementations, our cost metric is based on the amount of storage needed per router for the different IAR implementations. We assume that UGAL route computation is implemented in the baseline router. The storage required for the baseline router is 264K bits per router, as each router has 15 ports, 3 VCs per input port, and buffer size of 256 flits per global VC and 32 flits per local VC. We assume the flit size is 64 bits and each packet contains 10 flits. A summary of the estimated cost is shown in Table 1. PB is the most cost-effective IAR method, while PAR has the highest cost because it requires an additional VC per input port.

The cost of implementing CRT was first presented in [10]. If credit round trip information is represented by k bits, a router with h global channels requires kh registers to store $t_d(GC)$. Each credit needs to be tracked individually to measure the t_{crt}, but since there is a 1:1 correspondence between flits and credits, a timestamp queue [10] can be used. Each time a flit is transmitted on a global channel, a timestamp is added to the queue. When a credit returns from the channel, a timestamp is popped from the queue, and the difference between the timestamp and the current time is t_{crt}. The queue size is equal to the total amount of buffering, F, at the input of the downstream router. The credit queues will require khF registers per router. The cost is also directly proportional to the bit representation of the timestamp. The number of bits must be sufficient to cover t_{crt0} plus any additional delay. Therefore, we use $log_2(t_{crt0}) + 1$, or 9 bits, to represent t_{crt}. Under this setup, CRT requires an additional 27K bits of storage per router. Since CRT's cost depends on the timestamp queues, its router overhead can be reduced by sacrificing timestamp accuracy.

Table 1: Cost Summary

	Total cost per router (bits)	Increase from baseline
Baseline	264K	N/A
CRT	291K	10%
PAR	352K	33%
PB	264K	<1%
RES	274K	4%

For the RES method, the entire packet must be stored in a waiting buffer at the injection node until the reservation flit has returned. Since each node can inject a maximum of one flit per cycle into the router, and the reservation flit can be sent when the head flit is injected, the size of the waiting buffer is proportional to the zero-load reservation round trip latency τ. We increase τ by a safety factor of two to account for any additional network delays experienced by the reservation flit. A reservation counter is needed at each global channel to track the reservation level, requiring and additional $hlog_2(F)$ registers per router. The total RES overhead is 10K bits of storage per router. RES's cost can be reduced by combining the waiting buffer with the input buffer at the injection node.

The PB method requires a lookup table in each router that stores the congestion level of all global channels within the group. With a routers per group and h global channels per router, ah 1-bit registers are needed per router. Although data packets are extended by h bits to carry piggybacked data, the additional h bits are removed before the packet is stored in the input buffer. Thus, the total storage increase is only 32 bits or <1% increase. To ensure this lookup table is not the bottleneck of the router, the lookup table should be replicated. However, with the small amount of storage required, the impact of replicating the lookup table is minimal. The PB method also incurs additional cost for the bandwidth used by the piggybacked data, which is $\frac{h}{length_{packet}}$. For our network configuration, this results in only a 0.6% bandwidth overhead.

The PAR router requires an additional virtual channel to avoid deadlock, thus increases the router storage by one third compared to the baseline router.

7. RELATED WORK

There has been much research in the area of routing for interconnection networks. Routing can be classified as either oblivious routing or adaptive routing [4]. Oblivious routing methods, such as Valiant's Algorithm [20] can be used to load-balance adversarial traffic patterns. Deterministic algorithms such as dimension-order routing provide a fixed path from source to destination that simplifies the route computation while providing high performance on benign traffic patterns. However, these oblivious routing algorithms are unable to adapt to different traffic patterns. Adaptive routing is needed to provide high performance across a wide range of traffic patterns.

Various adaptive routing algorithms have been previously proposed [1, 8]. These adaptive routing algorithms focus on providing higher performance by increasing path diversity and exploiting non-minimal routing to load-balance the network channels and increase network utilization. However, most prior work on adaptive routing focuses on low-radix networks [14, 6]. The migration to high-radix topologies presents new challenges in adaptive routing. This work addresses these challenges by presenting alternative routing algorithms to load-balance the global channels in the high-radix dragonfly topology.

Gratz et al. [7] present a taxonomy of routing policies based on the adaptiveness of a routing method. Under the category of adaptive routing algorithms that sense network congestions, in most cases only congestion information local to a router is used in the routing decisions [18, 19], and thus these methods only require *direct* adaptive routing. Recently, there has been work on using global congestion information for route computation in on-chip networks. Gratz et al. proposed the Regional Congestion Awareness method (RCA) [7] to implement adaptive routing for on-chip mesh networks. Token Flow Control (TFC) proposed in [13] also provides information sharing between routers of an on-chip mesh network. Routers send their resource availability information to adjacent routers. The knowledge of resource availability allows some network packets to bypass router pipelines and reduce network latency. Like RCA and TFC, the IAR method proposed here uses remote information in making routing decisions. However, IAR goes beyond these methods tailored for on-chip networks with low delays between routers to address the challenges of high-radix, large-scale networks with high channel delays and large queues. The constraints of an on-chip network are dramatically different from the constraints and requirements of a large-scale network, and hence different solutions are required.

Previous work [10] showed that CRT performs well when compared to UGAL because it eliminates the high intermediate latency. It showed results for CRT approaching the performance of ideal routing. However, it did not address the impact of packet size or evaluate how CRT reacts to transients. In addition, the CRT evaluation of [10] did not consider the impact of realistic channel latency. The high latency of long global channels reduces the accuracy and timeliness of the CRT information and hence makes it difficult to quickly respond to bottlenecks. In this work, we modify the CRT mechanism to include the impact of channel latency as the congestion information, $t_d(GC)$, is delayed by the global channel latency and compare CRT against other proposed IAR routing algorithms.

The idea of indirect adaptive routing has been used in other network contexts. Routing methods similar to PAR have been previously proposed for ad hoc wireless networks. Choi and Das [2] describe a routing method that provides rerouting mechanisms to packets encountering congestion or broken links on their primary path. However, the constraints of wireless networks are very different from high-radix, large-scale networks, and thus the results presented in [2] do not apply to a large system-area network.

8. CONCLUSION

Global adaptive routing is required to load-balance the global channels in some large-scale interconnection networks to achieve reasonable performance. In networks, such as a dragonfly, with soft backpressure between the point of congestion and the source router, *direct* adaptive routing methods, that rely on network state observable directly at the source router, are slow to react to congestion — leading to high latency. This paper introduced *indirect* adaptive routing methods that make routing decisions using network state not directly observable at the source router.

We presented three novel indirect adaptive routing methods: progressive adaptive routing (PAR), piggygback routing (PB), and reservation routing (RES). We evaluated the steady-state performance of these three methods, along with the previously published credit-round trip (CRT) method on a dragonfly network. We introduced transient network evaluation and evaluated the transient response of PB, PAR, and CRT. Our results show that PB, PAR, and CRT all achieve good performance, with PB having the best performance under steady state traffic pattern. PAR has slightly higher latency than PB because it wastes network resources routing packets minimally that later are re-routed non-minimally, but it has the fastest response time to transient traffic. CRT is less accurate than other methods because the remote congestion is inferred through credit delays. The RES method is not suitable for use with short packets because of its high overhead. We also compared the implementation complexity of these routing algorithms. PB has the lowest implementation cost and results in less than 1% area overhead in a high-radix router.

9. ACKNOWLEDGMENTS

The authors would like to thank Steve Scott for suggesting the idea of progressive adaptive routing, students from the Concurrent VLSI Architecture group at Stanford for feedback on the paper, and the anonymous referees for many helpful suggestions. This work has been supported by the National Science Foundation under grant CCF-0701341 and by gifts from Cray, Inc. and Google, Inc.

10. REFERENCES

[1] R. V. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 351–360, San Diego, CA, 1993. ACM.

[2] W. Choi and S. K. Das. Design and performance analysis of a proxy-based indirect routing scheme in ad hoc wireless networks. *Mob. Netw. Appl.*, 8(5):499–515, 2003.

[3] C. Clos. A study of non-blocking switching networks. *Bell System Technical Journal*, 32:406–424, 1953.

[4] W. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2003.

[5] W. J. Dally. Virtual-channel flow control. In *ISCA '90: Proceedings of the 17th annual international symposium on Computer Architecture*, pages 60–68, New York, NY, USA, 1990. ACM.

[6] P. T. Gaughan and S. Yalamanchili. Adaptive routing protocols for hypercube interconnection networks. *Computer*, 26(5):12–23, 1993.

[7] P. Gratz, B. Grot, and S. W. Keckler. Regional congestion awareness for load balance in networks-on-chip. *High Performance Computer Architecture, 2008. HPCA 2008. IEEE 14th International Symposium on*, pages 203–214, Feb 2008.

[8] L. Gravano, G. D. Pifarré, P. E. Berman, and J. L. C. Sanz. Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks. *IEEE Trans. Parallel Distrib. Syst.*, 5(12):1233–1251, 1994.

[9] 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, 2007.

[10] J. Kim, W. J. Dally, S. Scott, and D. Abts. Technology-driven, highly-scalable dragonfly network. In *Proc. of the International Symposium on Computer Architecture (ISCA)*, pages 77–88, Beijing, China, 2008.

[11] J. Kim, W. J. Dally, S. Scott, and D. Abts. Cost-efficient dragonfly topology for large-scale system. *IEEE Micro Top Picks*, 29(1):33–40, Jan/Feb 2009.

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

[13] A. Kumar, L.-S. Peh, and N. Jha. Token flow control. In *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 342–353, Nov. 2008.

[14] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Trans. Comput.*, 40(1):2–12, 1991.

[15] O. Lysne, S.-A. Reinemo, T. Skeie, Å. G. Solheim, T. Sødring, L. P. Huse, and B. D. Johnsen. The interconnection network - architectural challenges for utility computing data centres. *IEEE Computer*, 41(9):62–69, 2008.

[16] 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, 2006.

[17] S. L. Scott and G. M. Thorson. The cray T3E network: Adaptive routing in a high performance 3D torus. In *Hot Interconnects*, pages 147–156, August 1996.

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

[19] 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.

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