Adaptive Routing in High-Radix Clos Network

John Kim, William J. Dally, Dennis Abts†
Stanford University †Cray Inc.
{jjk12, billd}@cva.stanford.edu dabts@cray.com

Abstract

Recent increases in the pin bandwidth of integrated-circuits has motivated an increase in the degree or radix of interconnection network routers. The folded-Clos network can take advantage of these high-radix routers and this paper investigates adaptive routing in such networks. We show that adaptive routing, if done properly, outperforms oblivious routing by providing lower latency, lower latency variance, and higher throughput with limited buffering. Adaptive routing is particularly useful in load balancing around nonuniformities caused by deterministically routed traffic or the presence of faults in the network. We evaluate alternative allocation algorithms used in adaptive routing and compare their performance. The use of randomization in the allocation algorithms can simplify the implementation while sacrificing minimal performance. The cost of adaptive routing, in terms of router latency and area, is increased in high-radix routers. We show that the use of imprecise queue information reduces the implementation complexity and precomputation of the allocations minimizes the impact of adaptive routing on router latency.

1 Introduction

Interconnection networks are widely used to connect processors and memories in multiprocessors [20, 21], as switching fabrics for high-end routers and switches [9], and for connecting I/O devices [19]. As performance of processor and memory continue to increase in a multiprocessor computer system, the performance of the interconnection network plays a central role in determining the overall performance of the system. The latency and bandwidth of the network largely establish the remote memory access latency and bandwidth.

Recent advances in signaling technology have greatly increased the pin bandwidth available in a router chip. This bandwidth is most effectively used to reduce latency and cost by building high-radix routers — with a large number of *thin* (modest bandwidth) ports — rather than conventional routers — with a small number of *fat* (high bandwidth) ports [13].

With migration towards high-radix routers, the low-radix topologies such as the k-ary n-cubes [7] are no longer suitable. However, the folded-Clos [5] (or the fat-tree [14]) is a topology that can take advantage of high-radix routers. The Cray BlackWidow vector multiprocessor [20] uses radix-64 routers and implements a modified folded-Clos network. In this paper, we evaluate adaptive routing on a high-radix folded-Clos network and compare it to oblivious routing. We also evaluate different *allocation* algorithms that can be used in adaptive routing.

The main contributions of this paper include:

- Proper adaptive routing in a folded-Clos network reduces latency and provides less variance in the distribution of packet latency – which ultimately reduces the global synchronization time in a multiprocessor.
- We show how nonuniformities in the network traffic can be created in a folded-Clos topology by the presence of deterministic routing and faults in the network. By routing adaptively around the nonuniformities, adaptive routing can "smooth out" the traffic and provide significantly higher throughput compared to oblivious routing.
- We compare different allocation algorithms that can be used in adaptive routing on a high-radix network and compare their performance. We introduce randomization in the allocation algorithms to simplify the routing decision with minimal loss in performance.
- We evaluate the cost of implementing adaptive routing in a high-radix router. To minimize the implementation cost, we show how reduced precision simplifies the comparison logic and *precomputation* of the allocations minimizes the impact the on router pipeline delay.

The rest of the paper is organized as follows. In Section 2, we provide background and discuss related works on topology and routing of a folded-Clos network. We compare adaptive and oblivious routing on a high-radix folded Clos in Section 3 and discuss the benefits of adaptive routing. We discuss different allocation algorithms that can be used in adaptive routing in Section 4. A cost

SC2006 November 2006, Tampa, Florida, USA
0-7695-2700-0/06 $20.00 ©2006 IEEE

comparison and techniques to reduce the implementation cost of adaptive routing is presented in Section 5. Section 6 presents conclusion and future works.

2 Background and Related Works

2.1 Topology

A Clos network is a multi-stage non-blocking network with an odd number of stages [5]. The network is equivalent to two back-to-back *butterfly* networks — where the last stage of the input network is fused with the first stage of the output network. The input network can route from any input to any *middle-stage* switch. The output network can route from any middle-stage switch to any output. A 5-stage Clos network with 8 nodes, using radix-2 routers, is shown in Figure 1(a). Because of packaging constraints, a Clos network can be folded so that the input network and output network share switch modules. A folded-Clos network is sometimes called a fat-tree [14]. The corresponding folded-Clos of the Clos network shown in Figure 1(a) is shown in Figure 1(b).[1]

The folded-Clos topology has been used in various different networks including both circuit switching [5] and packet switching [6, 15] . Most folded Clos networks use low-radix routers such as the CM-5 network which uses radix-8 routers and the Cray XD1 which uses a radix-24 Mellanox [16] routers. By using *high-radix* routers in a folded Clos network, the latency and the cost of the network can be lowered [13]. The BlackWidow network [20] takes advantage of high-radix router and implements a modified high-radix folded Clos network.

2.2 Routing

For a given topology, an appropriate routing algorithm is needed to load-balance the traffic and minimize the latency. A routing algorithm can be classified as either *oblivious* where the routing decisions are made randomly or *adaptive* where the decisions are made based on the network state (e.g. queue depths along the route).

Routing a packet through a Clos network proceeds in two phases: input and output.[2] During the input phase, a middle-stage switch is selected and the packet is routed to that switch. For a folded-Clos topology, the packet need not route all the way to the middle stage but can stop as soon as a common ancestor of the source and destination nodes is reached. Any middle-stage switch (or common ancestor switch) can be selected during the input phase. The selection may be made using either oblivious or adaptive routing. During the output phase, the packet is routed from the selected middle-stage switch (or common ancestor) to its destination output port. This routing is deterministic as there exists only a single path to the destination.

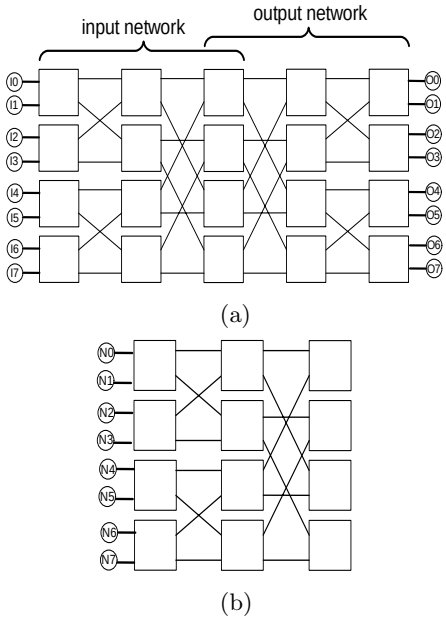


Figure 1: A block diagram of (a) Clos topology and (b) folded Clos or a fat-tree topology. The channels in (a) represent unidirectional channels while channels in (b) represent bidirectional channels.

Both adaptive routing [15] and oblivious routing [23] have been implemented in a folded-Clos topology. Oblivious routing is simple to implement as each source independently selects a random middle-stage switch for each packet. Adaptive routing is more complex, requiring a decision at each input-network router to select the output port based on network state. Despite the higher complexity, adaptive routing is often used as it gives better performance.

Adaptive routing algorithms have been studied in depth on the torus network [3, 22]. Because the torus network contains non-minimal paths, the routing algorithm can provide proper load-balancing by taking non-minimal paths and lead to higher performance. However, a folded-Clos network has multiple paths between a source and a destination with all of the paths being minimal. Therefore, the difference between oblivious routing and adaptive routing is not clear on this topology.

Aydogan et al. showed that an adaptive routing provides better performance than oblivious routing on the SP2 network [1]. However, their study was focused on a low-radix routers and limited analysis was provided on the benefits of adaptive routing over oblivious routing.

[1]The 5-stage folded-Clos requires radix-4 routers. However, the middle stage routers are logically radix-2 routers. A radix-4 router can be partitioned into two radix-2 *virtual routers* [20] and used in the middle stages.

[2]The input phase routing will be referred to as *uprouting* and the output phase routing will be referred to as *downrouting*.

Petrini and Vanneschi [18] evaluated a family of fat-trees and studied the benefits of virtual channels [8] with adaptive routing but do not compare oblivious routing with adaptive routing.

We extend these work by providing a better understanding on the difference between adaptive and oblivious routing on a folded-Clos network. Because of the complexity of implementing adaptive routing in a high-radix router, we discuss different implementations and compare their performance and implementation cost.

Many studies have been done with regard to the impact of randomization. Mitzenmacher [17] studied the impact of two choices in load balancing and Azar et al. [2] studied balanced allocation and how it can be applied to on-line load balancing. The uprouting in folded-Clos network can also be viewed as an *allocation* problem since the packet can traverse using any uplinks. Therefore, the routing needs to make an assignment of the outputs to the input ports. We apply randomization to the different allocation algorithms and show that most of the gain can be realized with only two choices [2]. However, continuing to increase the number of random samples can actually *reduce* the performance in some allocation algorithm.

3 Adaptive vs. Oblivious routing

In this section, we compare adaptive and oblivious routing on a high-radix folded-Clos network and discuss the benefits of adaptive routing. We show how adaptive routing is particularly beneficial with limited buffering and nonuniformities. We assume an ideal adaptive routing in this section using the *sequential* allocation algorithm. The different allocation algorithms will be discussed in Section 4.

3.1 Benefits of Adaptive Routing in High-radix Network

The goal of adaptive routing in a folded-Clos network is to load balance across the different physical links during the uprouting to the common ancestor. Efficient adaptive routing will minimize packet *collisions* that occur when multiple packets request the same output. These collisions will create *congestion* in the network and result in higher latency and lower throughput [11].

To evaluate the benefit of adaptive routing, we calculate the ratio of network latency between adaptive routing (T_{adapt}) and oblivious routing (T_{ob}). Ratio of 1 represents no benefit of adaptive compared to oblivious routing while a lower ratio represents a higher benefit of adaptive routing. The odd number of stages in a folded-Clos network can be expressed as $2x+1$ where $x > 0$, x stages in the uprouting of the folded-Clos that is routed either adaptively or obliviously and $x+1$ stages in the downrouting that must be routed deterministically. The latency through the network can then be expressed as

$$\begin{aligned} T_{ob} &= (2x+1)Q_{ob} \\ T_{adapt} &= xQ_{adapt} + (x+1)Q_{ob} \end{aligned}$$

where Q_{ob} and Q_{adapt} are the per router delay (including the queuing delay and the router pipeline delay) of the router that is routed oblivious and adaptively respectively. At high offered loads where the queuing delay dominates the total delay, $Q_{adapt} \ll Q_{ob}$ since adaptive routing attempts to remove congestion and minimize any queuing delay. Thus, the ratio T_{adapt}/T_{ob} can be simplified to $\frac{x+1}{2x+1}$.

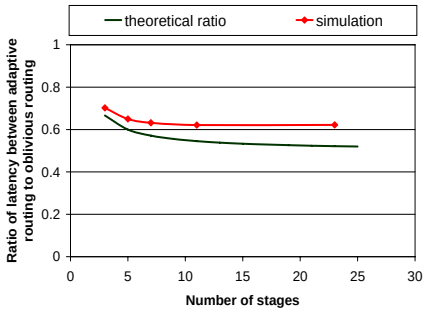


Figure 2: Ratio of latency using adaptive and oblivious routing in a 4K node folded-Clos network as the number of stages in the network is varied. For larger number of stages, the radix of the router decreases and the smaller ratio represents higher benefit of adaptive routing.

We plot this ratio in Figure 2 and the ratio from simulations [3] of a 4K node folded-Clos network at an offered load of 0.95. We vary the radix of the routers such that with radix-128, only 3 stages (x=1) are required and with radix-4, 23 stages (x=11) are needed.

The simulation and theoretical ratio follow the same trend but the simulation ratio is higher because of the various assumptions made in the analysis. There is more benefit of adaptive routing as the number of stages increases, resulting in up to 40% savings in latency. High-radix networks have fewer stages but there is still approximately 30% reduction in latency with adaptive routing. This result and analysis assumed infinite buffers at each router but we show in the next section that when buffering is limited or in the presence of nonuniformities, the benefit of adaptive routing is much greater in a high-radix network.

3.2 Performance Evaluation

In this section, we provide additional simulation results to compare adaptive and oblivious routing. We use a

[3]The simulation setup is described in Section 3.2

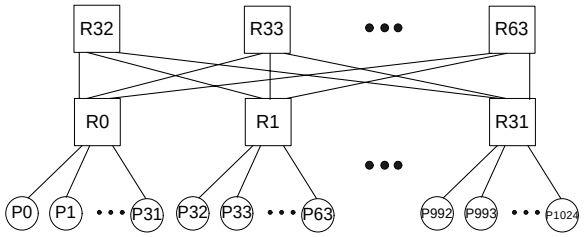


Figure 3: Block diagram of a 1K node high-radix folded-Clos network with radix-64 routers. P0-P1023 represents the terminals, R0-R31 represents the first level routers, and R32-R63 represents the second level routers.

cycle-accurate simulator to evaluate the performance of adaptive and oblivious routing in a folded Clos network. We perform open-loop simulations [10] to evaluate the network and assume that the system has enough latency hiding so that the offered load is not affected by message latency.[4] We simulate a single-cycle input-queued router and the 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 are labeled during a measurement interval. The sample size was chosen such that the measurements are accurate to within 3% with 99% confidence. The simulation is run until all measurement packets are delivered.

The design of input-queued routers has been shown to be problematic as the number of ports increase in high-radix routers [13]. However, in order to generalize the results, we use input-queued routers and provide sufficient switch speedup so that the routers do not become the bottleneck in the network. We use radix-64 routers and create a 3-stage folded Clos network with 1K nodes as shown in Figure 3. We evaluate the network performance using the worst-case uniform random (wc-UR) traffic pattern – each source sends traffic to a random destination whose common ancestor is the root of the network – as well as permutation traffic patterns. The credit-based flow control is used between routers to maintain the buffer information of downstream routers. Since we are focusing on the routing of the network, we assume only a single virtual channel [8] and use a packet size of 1 flit. Longer packet sizes generally follow the same trend in the comparison but when necessary to clarify some of the results, we vary the packet size.

3.2.1 Infinite Buffers

The simulation results comparing adaptive and oblivious routing with infinite buffers is shown in Figure 4(a).

[4]As an example, the processors in the BlackWidow system [20] can support thousands of outstanding global memory references and provide latency hiding.

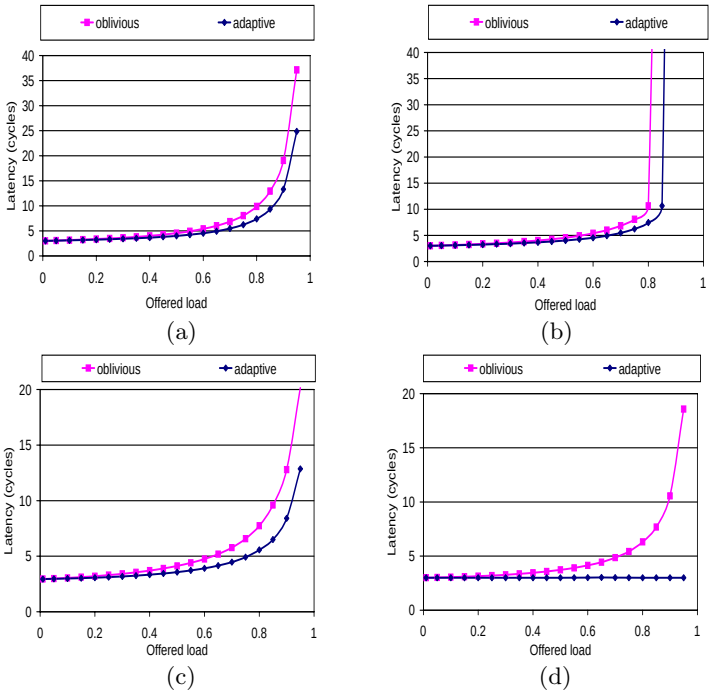
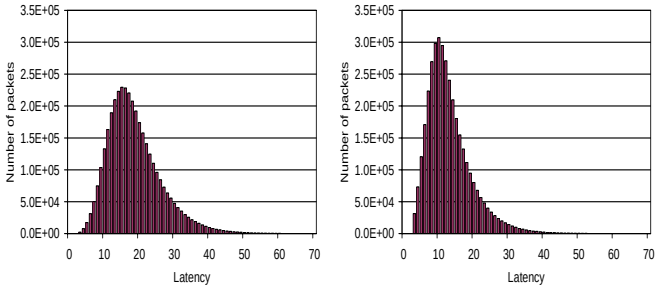


Figure 4: Adaptive and oblivious routing comparison of the latency vs. offered load in the folded Clos network with (a) wc-UR traffic with infinite buffers (b) wc-UR traffic with 16 buffers (c) bit reverse traffic pattern and (d) bit complement traffic pattern.

The throughput is identical as both routing algorithms achieve nearly 100% throughput with adaptive routing resulting in lower latency at higher loads. For oblivious routing, the middle stages are chosen randomly – thus, with sufficient buffering, the *expected* or the average load across all the outputs will be the same. Thus, oblivious routing results in the same throughput as adaptive routing with infinite buffers.

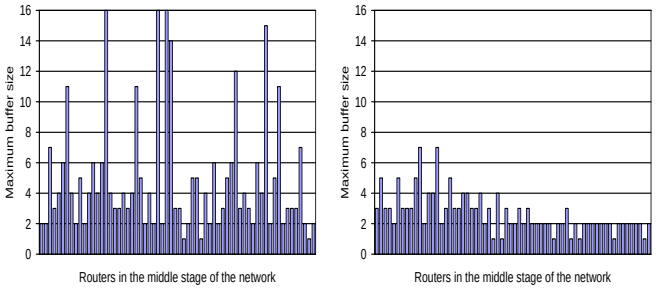
However, adaptive routing provides lower latency at higher offered loads since the routing decisions are made adaptively each cycle to load balance across the uplinks. To illustrate the benefit of adaptive routing, we plot the distribution of packet latency in Figure 5 for oblivious and adaptive routing at an offered load of 0.9. The average latency of oblivious routing is approximately 38% higher than adaptive. In addition, adaptive routing has a tighter latency distribution with a standard deviation that is 20% less than that of oblivious routing. As a result, the poor instantaneous decisions of oblivious routing lead to higher average latency and a larger latency distribution of the packets compared to adaptive routing. Reducing the variance in the packet latency is significant as it will improve the global synchronization time across the whole system.

Other traffic patterns such as bit reverse permutation [10] follows the same trend as the wc-UR traffic



(a) (b)

Figure 5: Latency distribution of packets with an offered load of 0.9 with (a) oblivious routing and (b) adaptive routing.



(a) (b)

Figure 6: A *snapshot* of the maximum buffer size of the middle stage routers in a 1K node folded-Clos network. The distribution is shown for (a) oblivious and (b) adaptive routing at an offered load of 0.8 and the buffer depth of the routers are 16 entries.

pattern (Figure 4(c)). With a congestion-free traffic pattern [11] such as the bit complement permutation, adaptive routing results in a constant delay regardless of the offered load [1] while congestion increases latency at higher offered load with oblivious routing (Figure 4(d)).

3.2.2 Finite Buffers

When buffering is limited, the throughput of oblivious routing suffers compared to adaptive routing as shown in Figure 4(b) when the input buffers are limited to 16 entries.[5] Adaptive routing provides approximately 10% higher throughput as oblivious routing does not consider the state of network – i.e. the randomly selected output maybe not be available because of lack of buffer space.

The poor instantaneous load-balancing of oblivious routing can be shown by a *snapshot* of the buffer utilization of the middle stage routers during simulation. For each middle stage router, we plot the size of the input buffer with the highest occupancy in Figure 6 at an offered load of 0.8. The average queue utilization is under 1 for both routing (0.78 for oblivious and 0.33 for adaptive) but the distribution of the maximum queue occupancy is different. With oblivious routing, some of the buffers are filled to capacity (16 entries) and average of the maximum buffer depth is approximately 5 entries. In contrast, the average of the maximum buffer depth is only 3 entries with adaptive routing and the maximum value is only 7, or roughly 50% of the capacity. Because of this imbalance in buffer utilization, oblivious routing results in not only higher latency but also lower throughput with limited buffering.

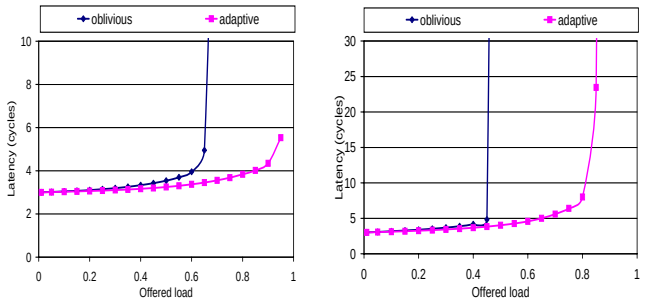
3.2.3 Nonuniformity - Presence of Deterministic Traffic

Additional benefits of adaptive routing can be observed in the presence of nonuniformity. Because of the symmetry in the topology, nonuniformity can not be created by the traffic pattern itself since the traffic can be distributed across all the middle stages.

The nonuniformity can result from deterministic routing being used in the network. In a shared-memory multiprocessor, it is often necessary to ensure ordering of requests to a given cache-line memory address because of the memory consistency model. Therefore, deterministic routing such as that used by the Cray BlackWidow network [20], can be used to provide in-order delivery of all request packets at a cache-line address granularity for a source-destination pair. Since response packets in the network do not require ordering they can be routed by using either oblivious or adaptive routing. By taking into account the congestive state of the network, adaptive routing avoids any nonuniformities that may be introduced as a result of deterministic routing of the request traffic. Oblivious routing does not take into account these potential nonuniformities and may randomly select an output port which has a substantial amount of deterministic request traffic en route. In effect, the adaptive routing will "smooth out" any nonuniformities in the traffic pattern to load balance the set of available links.

To illustrate this nonuniformity, we simulate a traffic pattern where each node i sends traffic to $(i+k)\ mod\ N$ where k is the radix of the routers and N is the total number of nodes. Using this traffic pattern, 50% of the traffic is routed using deterministic routing and the remaining 50% of the traffic is routed either adaptively or obliviously. The simulation result is shown in Figure 7(a). With oblivious routing, the throughput is limited to under 70% but with adaptive routing, 100% throughput can be achieved. The deterministically routed traffic cause nonuniformities in the various different middle stages as they need to route through the same middle stage. Adaptive routing allows the

[5] 16 buffer entries are sufficient to cover the credit latency.



(a) (b)

Figure 7: Routing comparison with nonuniformity in the traffic pattern. The latency vs. offered load comparison of adaptive and oblivious routing in the folded Clos network is shown for when (a) half of the traffic is routed using deterministic routing and (b) network with faults.

other traffic to be routed around these nonuniformities while oblivious routing can not avoid the nonuniformity, leading to lower throughput.

3.2.4 Nonuniformity - Faults in the Network

Nonuniformity can also be created by oblivious routing in the presence of faults in the network. An example is shown in Figure 8. Assume that the downlink from R4→R0 is faulty[6] and we observe the wc-UR traffic generated from nodes connected to R1. Any traffic generated for R0 can not be routed through R4 since they can not reach its destination, resulting in the traffic being load balanced across the other three routers (R5-R7). However, traffic destined for R2 and R3 will be equally distributed across all four uplinks. As a result, the uplink from R1→R4 will be underutilized while the other three uplinks will be over utilized – limiting the throughput of the network. To load balance appropriately, the traffic for R2 and R3 should utilize the R1→R4 uplink more so that the traffic is balanced.

To evaluate the impact of faults on adaptive and oblivious routing, we simulate a faulty network with the 1K network shown in Figure 3 with approximately 1.5% of the links between stage1 and stage2 routers (16 of the 1024 links) assumed to be faulty. The simulation results on this network with the wc-UR traffic pattern is shown in Figure 7(b). By load balancing appropriately across the healthy links, adaptive routing leads to approximately 2× improvement in throughput.[7]

[6]The corresponding uplink will also need to be disabled.

[7]In the simulation setup, we assumed 16 of the 32 links connected to R32 and R33 of Figure 3 are faulty. Different simulation setup will result in different amount of benefit using adaptive routing.

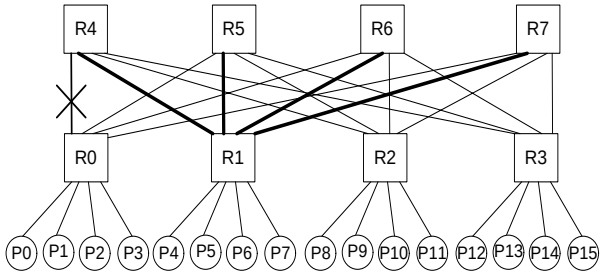


Figure 8: Block diagram of a radix-8 3-stage folded-Clos network with a fault. By using oblivious routing, the uplinks to the middle stages are not load balanced.

4 Allocation Algorithms in Adaptive Routing

Adaptive routing in a folded-Clos requires an *allocation* algorithm since the outputs (middle stages) need to be appropriately assigned to the inputs. We discuss the different allocation algorithms and compare their performances in this section.

4.1 Algorithm Description

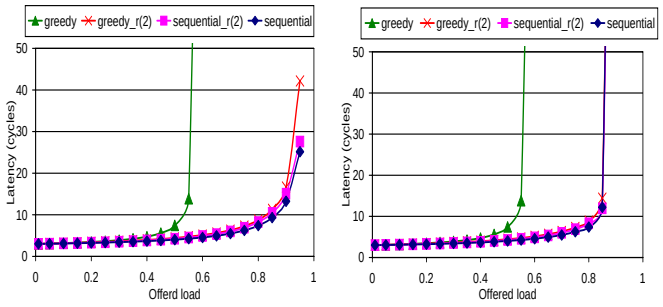
To load balance in a folded-Clos network, adaptive routing selects the middle stage with the least amount of congestion – i.e. middle stage with the largest amount of buffering available. Since simulation is done with input-queued routers, *credits* from the downstream routers are used to load-balance. Thus, middle stages with larger credits (more buffer space) is preferred over those with lower credits.

We evaluate the following four allocation algorithms. Unless stated otherwise, ties (equal credit counts) are broken randomly.

- *sequential* : Each input i makes its adaptive decision after inputs 0 through $i-1$ have made their decisions and updated the state of the network. The allocation algorithm assigns outputs to each input one at a time, taking into account the previous allocations of this cycle. To provide fairness, starting input is selected randomly each cycle.

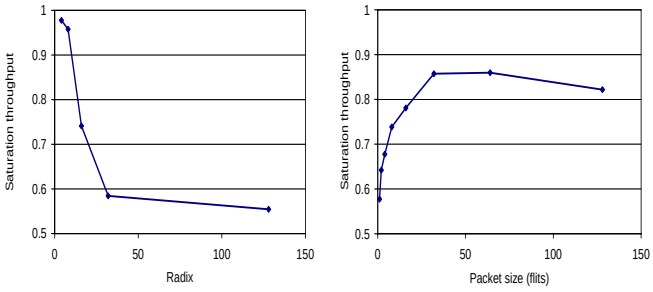
- *greedy* : Each input adaptively selects an output independently. As a result, each input does not take into account the routing decisions made by the other inputs in the same cycle.

- *sequential_r*(n) : A randomized version of *sequential* with n samples. Each input i selects n random outputs and adaptively selects among the n random outputs after inputs 0 through $i-1$ have made their routing decision. When $n = 1$, *sequntial_r*(1) is similar to oblivious routing and



(a) (b)

Figure 9: Adaptive routing comparisons with (a) infinite buffers and (b) 16 buffers using wc-UR traffic pattern.



(a) (b)

Figure 10: The impact on the saturation throughput as (a) radix and (b) packet size is varied using the *greedy* routing algorithm. Higher radix and smaller packet size limits the throughput of the *greedy* algorithm.

sequntial_r(k) is similar to the *sequential* algorithm where k is the radix of the router.

- *greedy_r*(n) : A randomized version of *greedy* with n samples. Each input selects n random outputs and adaptively selects among them. *greedy_r*(k) is identical to *greedy* and *greedy_r*(1) is identical to oblivious routing.

In the *sequential* and the *sequential_r*(n) allocation algorithms, if multiple outputs have the same credit count, the ties are broken randomly. However, if one of the outputs with the same credit count has already been selected by a previous input in the same cycle, we provide priority to the other outputs. For example, if four outputs {O_0,O_1,O_2,O_3} have a credit count {2,2,2,3}, the first input (I_0) would select O_3. The credit count for O_3 would be decremented by 1, resulting in a credit count of {2,2,2,2}. For the next input (I_1), all of the credits are equal and it can select any output to load balance. However, since O_3 has been selected by another input, we provide preference to the other three outputs (O_0,O_1,O_2). Using this policy to break ties simplifies the switch scheduling by not overloading an output and reduces congestion.

4.2 Algorithm Comparison

We simulate the different allocation algorithms using the simulation setup described in Section 3. For the randomized allocation algorithms, we use $n = 2$. The allocation algorithms are compared in Figure 9 using the wc-UR traffic pattern. The results for other traffic patterns follow the same trend.

With unlimited buffering, *sequential* provides the best performance as it leads to the lowest latency (Figure 9(a)). The *sequential_r*(2) performs comparable to *sequential* but leads to slightly higher latency near saturation (approximately 10% higher at an offered load of 0.95). The *greedy_r*(2) also provides the same throughput but leads to higher latency, approximately 60% higher at an offered load of 0.95.

The difference in latency between *greedy_r*(2) and *sequential* is minimized with limited buffering (Figure 9(b)). With unlimited buffers, *greedy_r*(2) might randomly select a *bad* output – i.e. an output which have a lot of packets. However, with limited buffering, if bad outputs are selected which correspond to outputs that are full, the packet will be stalled and allocation is re-attempted in the next cycle and can avoid the *bad* outputs. Thus, the latency difference is less than 20% at an offered load of 0.85 near saturation.

Regardless of the amount of buffering, the *greedy* algorithm performs poorly as the throughput is limited to under 60%. With the *greedy* algorithm, each input makes its routing decision independent of the other inputs. Thus, the routing decision might be an optimal *local* decision but could be a poor *global* decision in attempting to load balance. To illustrate this behavior, we use the same example from Section 4.1. If the credit count was {2,2,2,3} for the four outputs {O_0,O_1,O_2,O_3} and if a new packet arrives at all four inputs, all of the inputs would select O_3. This allocation would be an optimal local decision for the four inputs but does not globally load balance across all the outputs and leads to a poor allocation. As a result, congestion is created at an output and the poor allocation decision creates a head-of-line blocking effect [12] and limits the throughput of the router.

It is worth noting that the *greedy* algorithm is not problematic with low-radix routers but becomes problematic with high-radix routers. The throughput of the *greedy* algorithm is compared in Figure 10(a) on a 4K network as the radix of the routers is varied between radix-4 and radix-128 with a single flit packet. Lower radix networks achieves almost 100% throughput but the throughput drops to under 80% with radix-16 and beyond radix-32, the throughput is under 60%. The packet size also impacts the performance of *greedy* al-

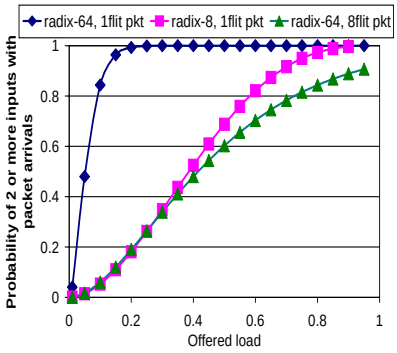


Figure 11: Probability of two or more packets arriving in the same cycle as the radix and the packet size is varied.

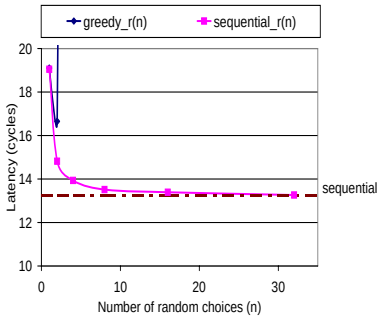


Figure 12: Randomized adaptive allocation algorithm comparison as n is varied for $sequential\_r(n)$ and $greedy\_r(n)$. The lower bound of the algorithm is shown by the *sequential* line.

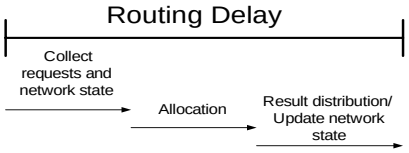


Figure 13: Timeline of delay in adaptive routing in a high-radix folded-Clos network.

gorithm as large packet size increases the throughput (Figure 10(b)). Since the routing decision is made only for the head flit of a packet, large packet size decreases the probability of having 2 or more new packets arriving in the same cycle – reducing the chance of output collision from the poor routing decision. In Figure 11, we plot this probability as a function of offered load. The probability approaches 1 very quickly for radix-64 router which explains the poor performance of high-radix routers for the *greedy* algorithm using 1-flit packets. However, the probability gradually approaches 1 for radix-8 router, With a packet size of 8-flits in a radix-64 router, the probability is reduced and behaves very similar to a radix-8 router with 1-flit packets. Thus, it is essentially the ratio between the radix and the packet length that determines the performance of *greedy* algorithm.

To evaluate the impact of the parameter n, we vary n in the randomized allocation algorithms ($sequential\_r(n)$ and $greedy\_r(n)$) and plot the latency at an offered load of 0.9 in Figure 12. When $n = 1$, the randomized allocation algorithms are identical to oblivious routing since only 1 randomly selected output is used. As n increases from 1 to 2, there is approximately 10% reduction in latency for $greedy\_r(n)$. However, as n is increased further, the latency *increases* significantly and is beyond the scale of the plot. As n approaches 32, the allocation algorithm behaves like *greedy* and the network is no longer stable as the offered load of 0.9 exceeds the throughput.[8] Similar to $greedy\_r(1)$, $sequential\_r(1)$ behaves identical to oblivious routing. By increase n from 1 to 2, there is over 20% reduction in latency with $sequential\_r(1)$. However, increasing n from 2 to 32 results in less than 10% reduction in latency. Thus, most of the performance gain can be achieved by using only two samples.

[8]Although radix-64 routers is used in the simulation, only 32 output selections are possible in routing upstream in the folded-Clos. Thus, the maximum size of n is 32.

The randomized algorithm were implemented assuming that the random choices are not necessarily unique - e.g. for $sequential\_r(2)$, the two randomly selected outputs can be the same output. Simulations show that the uniqueness of the random choices has minimal impact on the latency of the allocation algorithm. As a result, $sequential\_r(32)$ latency is slightly higher than the *sequential* but by less than 1%.

5 Cost Analysis

Although adaptive routing provides performance benefits, the cost of implementation complexity, in terms of router latency and area, need to be considered. For deterministic routing or source routing where only bit manipulation is required or oblivious routing where only a random number needs to be generated, the routing pipeline delay will be minimal. However, earlier work showed that introducing adaptive routing can increase the complexity and the cycle time of a router [4]. The larger number of ports in a high-radix router can further increase the routing complexity. For example, the YARC router requires 4 clock cycles for the routing decision [20].

A timeline of adaptive routing in a high-radix network is shown in Figure 13. The three main components to the latency of adaptive routing in high-radix routers are the following:

1. Collecting the network state (e.g. availability of the

outputs and the output credit information) as well as *requests* from each input.

2. Allocation based on the network state and the *requests*.

3. Updating the network state and distributing the outputs assigned to each input.

In this section, we provide a qualitative comparison of the different adaptive allocation algorithms discussed in Section 4.1. Then, we evaluate two different techniques that reduce the complexity of adaptive routing in a high-radix folded-Clos network. By using imprecise queue information, the complexity of the route allocation can be reduced. The precomputation of the allocations can effectively hide the router latency of adaptive routing. We evaluate their impact on performance and show that there is minimal performance loss. The use of imprecision can be used for all four of the algorithms but the precomputation can only be used for *greedy* and *greedy_r*(n) algorithm since they are distributed algorithms.

5.1 Algorithm Cost Comparison

Among the adaptive algorithms discussed in Section 4.1, *sequential* requires a centralized routing structure that collects all of the requests, performs the allocation, and distribute the results. The complexity of such routing structure grows as $O(k^2)$ and becomes prohibitively expensive to implement. The *sequential_r*(n) routing algorithm, even with $n = 2$, still requires a central routing structure since each input is routed sequentially.

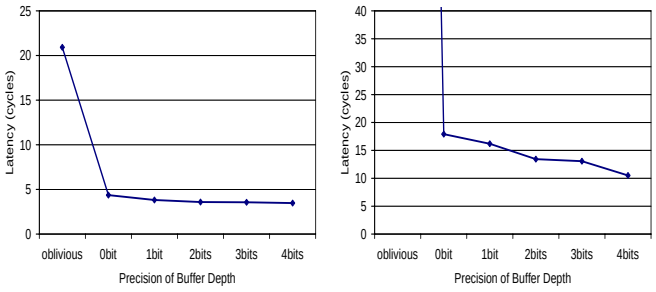
The *greedy* algorithm does not require a centralized structure as the routing logic can be duplicated at each input and be distributed. However, the routing algorithm still requires the distribution of the output information to all of the inputs. In addition, the comparison logic at each input needs to compare all the outputs, requiring a significant amount of logic. The *greedy_r*(n) algorithm can simplify the implementation as only n comparisons need to be made. Simulation earlier showed that $n = 2$ performs well – thus, only a single comparison is needed.

5.2 Precision in Adaptive Routing

The buffer depth (credits) is used to load-balance properly in adaptive routing and results so far assumed that the *full* information was available – i.e. the exact buffer depth was available from the credits. For a buffer with q entries, $log_2 q$ bits are needed to obtain the exact buffer information. However, $log_2 q$ bit comparators can be costly with the larger number of ports in a high-radix router. As a result, the YARC router only uses 1 or 2 bits of information to make its adaptive decision [20].[9]

number of bits	Description
0	no buffer depth information used – only whether an output is available or not
1	only the MSB is used
2	two MSBs is used
3	three MSBs is used
4	full buffer information is used

Table 1: Different values of precisions used to evaluate the impact of precision in adaptive routing. Buffer depth of 16 entries is assumed.



(a) (b)

Figure 14: Latency comparsion near saturation for network simulations with wc-UR traffic using (a) 1 flit packets and (b) 10 flit packets as the precision of allocation is varied . The network shown in Figure 3 was used for the simulations.

Table 1 describes the different values of precisions that are used to evaluate the impact of the precision. We assume a buffer depth of 16 and use the *sequential* algorithm in our evaluation. Using 0 bits of information corresponds to an allocation which does not consider the queue depth but considers only whether an output is available or not.[10] An output is not available if another input with a multi-flit packet is transmitting to the output or if the downstream buffer is full. By using only the most significant bit (MSB), the adaptive information will be used to differentiate whether the buffer has more or less than 8 entries. By using 2 bits, the buffer information is defined in granularity of 4 entries – e.g. 00 corresponds to less than 4 entries, 01 corresponds to 4 or more entries but less than 8 entries, and so forth. Using 3 bits results in a finer granularity and 4 bits allows the exact queue information to be used.

[9]The routing decision at the input buffers of YARC use 1 bit to select the row buffer and 2 bits are used to select the output within the column.

[10]This adaptive allocation is similar to the adaptive routing used in CM-5 [15].

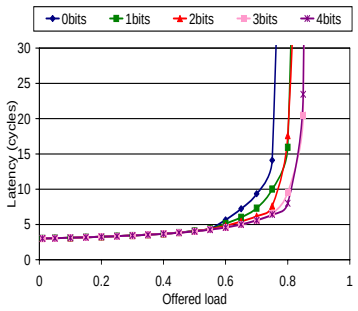


Figure 15: Impact of precision with nonuniform traffic pattern.

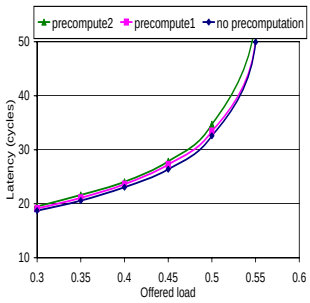


Figure 16: Performance comparison with the use of precomputation.

We plot the latency near saturation throughput as the precision is varied in Figure 14. With single flit packets, there is a significant difference between oblivious and adaptive routing but only a small difference between any of the different precision schemes – the difference between 0 bits and 4 bits is less than 10% (Figure 14(a)). With longer packets, the precision has more significant impact on the latency as 4 bits of precision can reduce the latency by over 60%, compared to using 0 bits (Figure 14(b)). However, by using only 2 or 3 bits of precision, the latency can still be reduced by over 40% compared to using 0 bits.

For the uniform traffic patterns, the throughput is very similar regardless of the precision used and difference in latency near saturation was compared. However, with the nonuniformity such as the one discussed in Section 3.2.4, the different precision results in different throughput as shown in Figure 15. Using 0 bit of information still outperforms oblivious routing (see Figure 7(b)) but results in approximately 15% reduction in throughput, compared to using all 4 bits. By using only 2 bits of precision, the difference can be reduce by half and 3 bits of precision performs nearly identical to using all 4 bits of precision.

5.3 Precomputation

To reduce the impact of adaptive routing on the router latency, the allocation can be *precomputed*. By using the queue information in the previous cycle, the routing decision can be precomputed and be available when a new packet arrives. The precomputation of allocations can be utilized with minimal loss in performance for the following reasons.

- The queue depth will change minimally from cycle to cycle.
- When full precision is not used, the change in the credit will have minimal impact – e.g. if only the 2 bits are used for adaptive decisions, the change in the lower 2 bits will have minimal impact.
- With the use of randomization, even if some of the data is *stale*, it might not impact the results.

The performance comparison with precomputation is shown in Figure 16 using $greedy\_r(2)$ algorithm. We compare the performance without precomputation to precompute1, where the output is calculated in the previous cycle, and precompute2 where the output is calculated in 2 cycle advance. The comparison is shown for the nonuniform traffic from Section 3.2.4 with a 10 flit packets using only 2 bits of precision. Both precompute1 and precompute2 perform comparable to no precomputation, with precompute2 resulting in approximately 10% higher latency near saturation. With minimal loss in performance, precompute2 will allow an extra cycle to distribute the routing information – further minimizing the impact of wires and reduce the router pipeline delay as the routing results can be computed in advance.

6 Conclusion and Future Work

A folded-Clos network is a topology that exploits the recent developments in high-radix routers to reduce the latency and the cost of the network. In this paper, we evaluate adaptive routing on a high-radix folded-Clos network and compare it to oblivious routing. We show that appropriate adaptive routing lowers latency and provides less variance in the packet latency compared to oblivious routing. With limited buffering, adaptive routing achieves better buffer utilization and results in higher throughput. Nonuniformity can be created in the folded-Clos topology with the presence of deterministically routed traffic and faults in the network. In the presence of such nonuniformity, adaptive routing provides significant advantages as it will "smooth out" the traffic and provide higher throughput.

We propose different allocation algorithms that can be used for adaptive routing and compare their performances. The *sequential* algorithm provides the best performance but is expensive to implement. By using

randomization with the $greedy\_r(2)$ algorithm, we show that the implementation can be simplified with minimal performance loss. We show that the implementation complexity can be further reduced by using imprecise queue information and the latency of adaptive routing can be hidden by *precomputing* the adaptive routing results in a previous cycle.

The migration toward high-radix networks open other opportunities for future work. Adaptive routing has been shown to provide better buffer utilization in this paper. However, it still remains to be seen what is the most effective way to partition the buffers available – e.g. partition the buffers into virtual channels, shared buffering scheme, etc. In addition, an appropriate flow control for high-radix networks needs to be evaluated. The diameter of the network is reduced with high-radix routers and it remains to be seen how this impacts the flow control.

Acknowledgments

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

References

[1] Y. Aydogan, C. B. Stunkel, C. Aykanat, and B. Abali. Adaptive source routing in multistage interconnection networks. In *IPPS '96: Proceedings of the 10th International Parallel Processing Symposium*, pages 258–267, Honolulu, HW, 1996. IEEE Computer Society.

[2] Y. Azar, A. Z. Broder, A. R. Karlin, and E. Upfal. Balanced allocations. *SIAM Journal on Computing*, 29(1):180–200, 2000.

[3] 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, California, 1993.

[4] A. A. Chien. A cost and speed model for k-ary n-cube wormhole routers. *IEEE Transactions on Parallel and Distributed Systems*, 9(2):150–162, 1998.

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

[6] Cray XD1. http://www.cray.com/xd1.

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

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

[9] W. J. Dally, P. Carvey, and L. Dennison. Architecture of the Avici terabit switch/router. In *Proceedings of Hot Interconnects Symposium VI, August 1998*, pages 41–50, 1998.

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

[11] S. Heller. Congestion-Free Routing on the CM-5 Data Router. In *Parallel Computer Routing and Communication Workshop*, pages 176–184, Seattle, WA, 1994.

[12] M. J. Karol, M. G. Hluchyj, and S. P. Morgan. Input versus Output Queueing on a Space-division Packet Switch. *IEEE Transactions on Communications*, COM-35(12):1347–1356, 1987.

[13] 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.

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

[15] C. E. Leiserson, Z. S. Abuhamdeh, D. C. Douglas, C. R. Feynman, M. N. Ganmukhi, J. V. Hill, W. D. Hillis, B. C. Kuszmaul, M. A. S. Pierre, D. S. Wells, M. C. Wong-Chan, S.-W. Yang, and R. Zak. The Network Architecture of the Connection Machine CM-5. *J. Parallel Distrib. Comput.*, 33(2):145–158, 1996.

[16] Mellanox. http://www.mellanox.com/.

[17] M. Mitzenmacher. The power of two choices in randomized load balancing. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1094–1104, 2001.

[18] F. Petrini and M. Vanneschi. k -ary n -trees: High performance networks for massively parallel architectures. In *IPPS '97: Proceedings of the 11th International Symposium on Parallel Processing*, page 87, Geneva, Switzerland, 1997. IEEE Computer Society.

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

[20] 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.

[21] 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.

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

[23] C. B. Stunkel, D. G. Shea, B. Aball, M. G. Atkins, C. A. Bender, D. G. Grice, P. Hochschild, D. J. Joseph, B. J. Nathanson, R. A. Swetz, R. F. Stucke, M. Tsao, and P. R. Varker. The SP2 High-performance Switch. *IBM Syst. J.*, 34(2):185–204, 1995.