Performance Analysis of a Minimal Adaptive Router

Thu Duc Nguyen and Lawrence Snyder

Department of Computer Science and Engineering
University of Washington, Seattle, WA 98195

In **Proceedings of the 1994 Parallel Computer Routing and Communication Workshop**, May 1994, pp. 31-44.

Abstract. Two classes of adaptive routers, minimal and non-minimal, are emerging as possible replacements for the oblivious routers used in current multicomputer networks. In this paper, we compare the simulated performance of three routers, an oblivious, a minimal, and a non-minimal adaptive router, in a two-dimensional packet switching torus network. The non-minimal adaptive router is shown to give the best performance and the oblivious router the worst. Significantly, however, for many traffic patterns, the minimal adaptive router's performance degrades sharply as the network saturates. Based on an analysis made using several visualization tools, we argue that this performance drop results from non-uniformities introduced for deadlock prevention. Furthermore, this analysis has led us to believe that *network balance* is an important performance characteristic that has been largely overlooked by designers of adaptive routing algorithms.

1 Introduction

As processors and network interfaces become faster, many researchers have speculated that these components may soon outpace the oblivious routers used in today's multicomputer networks. Thus, researchers have proposed numerous adaptive routing algorithms, arguing that adaptive routers can outperform oblivious routers because they are more flexible. Adaptive routing can be partitioned into two classes, minimal and non-minimal. To date, the question of *"how much adaptivity?"* seems to present a classic complexity vs. flexibility/performance tradeoff.

Simulation studies have shown that non-minimal adaptive routers can significantly outperform oblivious routers in hypercube and torus topologies [11, 3, 9, 12, 7]. Boppana and Chalasani [4] and Berman et al. [1] have shown that several minimal adaptive routing algorithms outperform oblivious routing for wormhole routing networks. Comparatively little is known, however, about the performance of minimal adaptive routers in virtual cut-through packet switching networks. Two differences make packet switching networks interesting: (1) Packet switching routers have larger buffers than typically assumed for wormhole routers; (2) Stalled packets are completely buffered in routers, releasing physical channels to be used by other packets. These differences can cause routing algorithms' behavior to differ significantly.

In this paper, we compare the simulated performance of three routers, an oblivious, a minimal, and a non-minimal adaptive router, in a two-dimensional virtual cut-through packet switching torus network. We base our minimal adaptive router on a routing algorithm designed by Cypher and Gravano [5]. This algorithm is particularly attractive because it:

- provides *fully* minimal adaptive routing (that is, a packet may be routed along all minimal paths between the source and destination),
- is deadlock and livelock free, and
- requires a provably minimal amount of buffer space (i.e. virtual channels) for deadlock-prevention.

We compare the performance of this minimal adaptive router against simulation results previously documented by Fulgham and Snyder [9] for a state-of-the-art oblivious and a non-minimal adaptive router (Chaos router [11, 3, 2]).

Typically, it is difficult to compare simulation results because different simulation systems encompass different router designs and different timing assumptions (e.g. clock cycle length). In our study, however, performance results for all three routers have been obtained using the same simulator, allowing "apples-to-apples" comparisons. Furthermore, this simulator was designed and implemented at the University of Washington to mimic the behavior of UW chip designs at the system clock level. Router designs for components that are independent of the routing algorithms, e.g., channel arbitration, derive from a chip implementation and so are the same for all three routers. On the other hand, algorithm specific factors, e.g., the latency of a routing decision, are modeled using variable parameters of the simulator. Reported behavior of the simulated oblivious and Chaos routers match actual physical implementations.

Our simulation results show that the minimal adaptive router outperforms the oblivious router but is outperformed by the non-minimal adaptive router. Significantly, for many traffic patterns, the minimal adaptive router's performance degrades sharply as the network saturates. Based on an analysis made using several visualization tools, we argue that this performance drop results from non-uniformities introduced for deadlock prevention. Furthermore, this analysis has led us to believe that *network balance* is an important performance characteristic that has been largely overlooked by designers of adaptive routing algorithms.

This paper is organized as follows. Section 2 describes the three routers. Section 3 describes the simulation methodology. Section 4 presents our simulation results and compares the minimal adaptive router to the Chaos and oblivious routers. Section 5 discusses related work and other minimal adaptive routing algorithms. Section 6 presents our conclusions.

2 Router Descriptions

In this section, we first describe the simulated network and briefly review the oblivious and Chaos routers. We then describe in slightly more detail the minimal adaptive routing algorithm and our simulation models.

In this study, we focus on virtual cut-through packet switching in a two-dimensional torus network with 256 routing nodes. We shall refer to the two dimensions of the torus as the x and y dimensions. Each router is labeled by a pair of coordinates (x, y) given by its position with respect to some router designated to have coordinate $(0, 0)$.

Each router has five bi-directional channels, with four channels connecting the router to its neighbors and one connecting the router to its processor. Each channel is connected to two sets of buffers at a router, the *input frames* and the *output frames*.[1] Each channel can only transmit in one direction at a time. Note that because all routers in the network are attached to processors, traffic can be and is introduced into the network from all nodes.

Each packet is composed of 20 flits (flow control units), with the first flit being large enough to hold the destination address. This implies that each flit must contain at least 8 bits, since our network has 256 nodes. All three routers can send or receive one flit over each of its five channels in one routing time unit (cycle). All three routers are virtual cut-through routers [10] – that is, once the header flit of a packet has been received and decoded, the router can direct that packet to a free channel before the packet has entirely arrived at that node. Input frames are connected to output frames via a full crossbar, allowing multiple packets to be in transit across a router concurrently.

2.1 Oblivious Router

The oblivious router uses dimension order routing; each packet is first routed in the x dimension until it reaches a router with the same x coordinate as the destination, then routed along the y dimension until it reaches the destination. This means that, at any time, each packet has only one permissible path through a router that it has reached. If the needed out channel is busy, the packet must wait. Deadlock is avoided using virtual channels as described by Dally and Seitz [6].

In the simulated model, the oblivious router requires two cycles to process a header flit and one cycle to route a packet across the wires of the appropriate channel, resulting in a minimum routing latency of three cycles. Headers of incoming packets are processed concurrently but only one routing decision can be made per cycle. That is, only one input to output frame setup can be done per cycle. To route packets, the oblivious router services its outframes in a round-robin fashion, moving to the next *interesting* output frame on each cycle. An output frame is interesting if it is available for use and is desired by some packet. If more than one packet wants a particular output frame, then one is chosen at random.

2.2 Chaos Router

Under normal routing conditions, the Chaos router routes packets from its input frames to its output frames similar to a minimal adaptive router. The Chaos router has a central multiqueue used to avoid head-of-line blocking and for deadlock prevention. A packet is moved into the multiqueue if it is potentially in a deadlock loop or if it has stalled

[1] For the oblivious and minimal adaptive routers, each end of a channel is connected to two sets of buffers rather than just two buffers because of the use of virtual channels for deadlock-prevention. For the Chaos router, each end of the channel is connected to just one input frame and one output frame.

at an input frame. When an output frame becomes available, packets in the multiqueue desiring that output frame have priority over packets in input frames. If more than one packet in the multiqueue desires the channel, the output frame is allocated using FIFO ordering. If a packet must be moved into the multiqueue when it is full, a packet in the queue is chosen at random and is *derouted* to the next available output frame; such a deroute may take the packet further away from its destination. Derouting is the only mechanism that can cause packets to be routed along non-minimal paths.

Like the oblivious router, the Chaos router services its channels using a round-robin policy. Because of its relative complexity, the Chaos router is charged with three cycles to process a header flit, resulting in a minimum routing latency of four cycles. Furthermore, implementation considerations [2] specify that a cut-through of a packet from the multiqueue to an output frame stalls the router's cut-through function for two cycles. In our simulation, the multiqueue can hold up to five packets and has crossbar connections to the input and output frames, allowing multiple packets to be transmitted into and out of the multiqueue. Like the oblivious router, however, the Chaos router can only process one packet header at a time.

2.3 Minimal Adaptive Router

The minimal adaptive routing algorithm specified by Cypher and Gravano [5] works as follows. Each router consists of three central FIFO queues, called the A, B, and C queues, representing three virtual channels. Each incoming channel is connected to all three queues. Similarly, each queue is connected to all outgoing channels. Packets are initially injected into the A queue. When a packet is in the A queue, the router examines the set of all possible minimal paths. If the packet can be routed in the positive direction using an internal edge, it is moved to the A queue of any neighbor which lies on a minimal path. Otherwise, the packet is moved into the B queue. For a packet in the B queue, the router likewise examines the set of all possible minimal paths. If the packet can be routed along a minimal path in the negative direction using an internal edge or in the positive direction using a wrap edge, then it is moved to the B queue of any neighbor which lies on a minimal path. Otherwise, the packet is moved into the C queue. For a packet in the C queue, the router checks to see whether it is the packet's destination. If so, it delivers the packet to the attached node. If not, then the packet is moved to the C queue of any neighbor which lies on a minimal path.

For our simulation, we have adapted the above algorithm to two simulation models. Figure 1 depicts a possible minimal adaptive router design that emphasizes simplicity, differing from the oblivious router only in that it routes packets adaptively. Figure 2 depicts a second possible minimal adaptive router that leverages the design of the Chaos router's central multiqueue to prevent stalled packets from holding on to valuable channel resources. We shall refer to these two simulation models as MA and MA-Q, respectively.

In the first model (Figure 1), each set of input and output frames connected to a channel consists of three buffers, one for each of the three virtual channels, A, B, and C. Each buffer is capable of storing one packet. Movements between queues as described above (e.g. from A to B) happen during the cut-through. Each router in the network

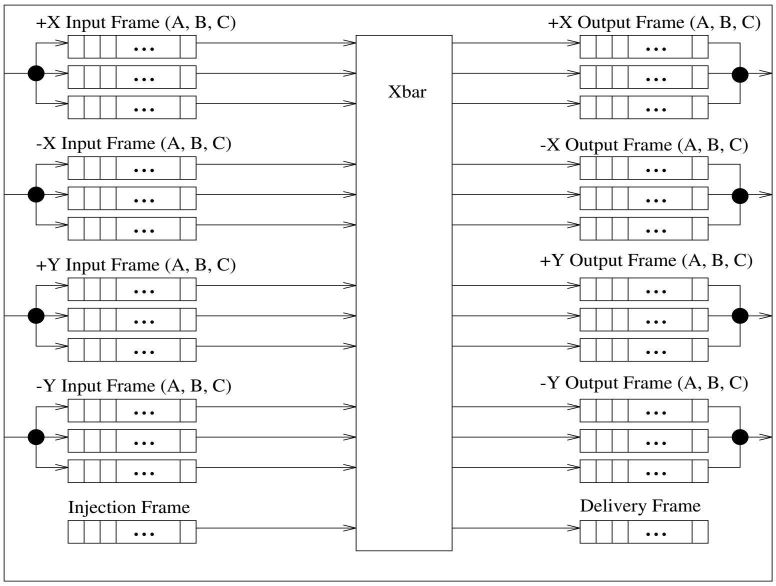

Fig. 1. Model for Simulated Minimal Adaptive Router (MA)

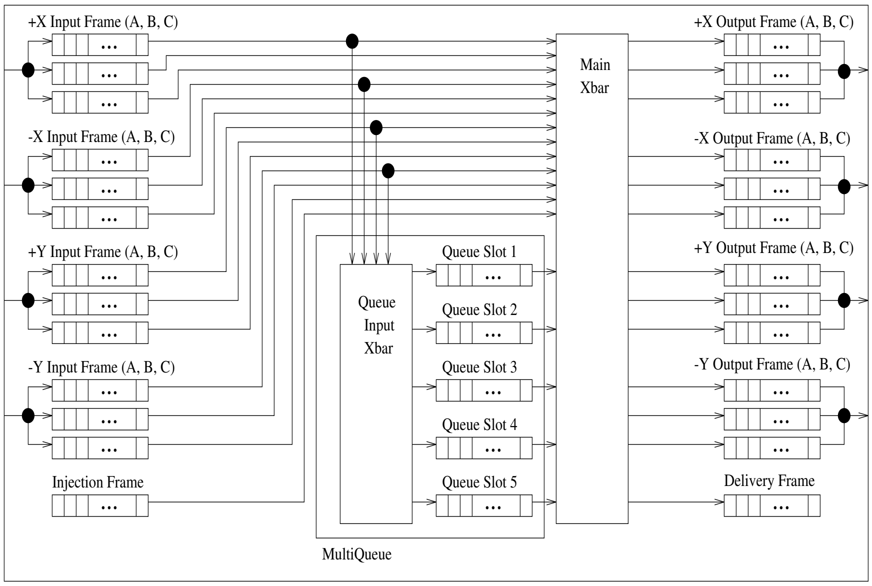

Fig. 2. Model for Simulated Minimal Adaptive Router With Central Queue (MA-Q)

maintains a bit-mask that describes whether the router's coordinate is 0, K,[2] or neither in each dimension. Using this mask and packets' headers containing the number of hops that the packet must still traverse in each direction, the above conditions can be tested efficiently. Suppose a packet is to be moved from the A queue to the B queue, when the router directs the packet from an input frame to an output frame, it simply directs the packet from the A buffer of the inframe to the B buffer of the outframe.

In the second model (Figure 2), in addition to the input and output frames, each router has a multiqueue as in the Chaos router. Each buffer in this model can hold exactly one packet and the multiqueue can hold five packets. As will be discussed below, early simulations show that performance limiting congestion occurs only in the A virtual channels. Thus, all buffer space in the central queue is reserved for packets traversing A virtual channels. When a packet traversing an A virtual channel stalls, it is moved into the multiqueue.

Both minimal adaptive routers are charged with three cycles for processing the header of a packet, resulting in a minimum routing latency of four cycles. Usage of the central queue incurs exactly the same penalty as for the Chaos router; that is, the routing of a packet from the multiqueue to an output frame stalls the router's cut-through function for two cycles. Output frames are serviced using a round-robin policy similar to the oblivious and Chaos routers. Only one packet header can be processed per cycle.

3 Methodology

All simulation experiments were performed using the Chaos project simulator. This simulator was designed and implemented to mimic the behavior of router chip designs at the system clock level. Simulated traffic is generated as follows. Each node generates packets independently. Packets are generated according to a Poisson distribution, with the mean arrival rate specified by the presented load. Furthermore, packets' destinations are chosen according to the following traffic patterns. To describe the patterns, let each node (x, y) also be labeled with a number xy, where xy is the value resulting from the concatenation of x and y. xy's binary representation is $a_{n-1}a_{n-2}...a_0$.

- *Random:* Destinations are chosen at random with equal probability.
- *Transpose:* Each node sends only to the destination given by $a_{(n/2)-1}a_{(n/2)-2}...a_0a_{n-1}a_{n-2}...a_{n/2}$.
- *Bit Reversal:* Each node sends only to the destination given by $a_0a_1...a_{n-1}$.
- *4X Hot Spots:* Ten randomly chosen nodes are distinguished. Destinations are chosen at random with the probability of choosing a distinguished node being 4 times the probability of choosing a normal node.

Various quantities such as throughput, source queueing time, latency, delay, number of hops, etc., can be measured using our simulator. Results are presented in terms of loads normalized to the bisection bandwidth. That is, loads are normalized with respect to the maximum theoretical load for uniform random traffic, where network bisection saturates. For a torus network with 256 nodes, the maximum injection rate, corresponding to a load of 100%, is one message every 80 cycles.

[2] In a k-ary n-cube network, $K = k - 1$. For a 256 node torus, $K = 15$.

Table 1. Minimum normalized load when saturation is detected (5% increments)

	Saturation			
Traffic Pattern	Obliv	Chaos	MA	MA-Q
Random	65	95	80	80
Hot Spot	50	90	70	70
Bit Reversal	40	85	65	65
Transpose	55	55	55	55

4 Simulation Results

Typically, router performance is assessed by two quantitative parameters, throughput and latency. Aside from throughput and latency, however, we believe that qualitative characteristics such as whether bandwidth and latency are location dependent may be important as well. In this section, we first discuss the throughput and latency characteristics of the minimal adaptive router and compare them with those of the Chaos and oblivious routers. We then discuss more qualitative characteristics of the minimal adaptive router observed using visualization tools.

4.1 Throughput

Throughput is defined as the sustained data delivery rate given some applied load. Two important throughput characteristics are saturation loads and sustained delivery rates after saturation. Saturation is defined as the first applied load when the network throughput is lower than the aggregate packet creation rate at source nodes. Note that throughput before saturation is not particularly interesting. By definition, before saturation, the throughput is simply equal to the applied load because all created packets are successfully injected into, routed, and delivered by the network.

In practical situations, one does not expect a network to operate close to or above saturation for extended periods of time. Network traffic, however, is known to be bursty, and peak transmission periods can often cause network saturation. Thus, saturation and throughput after saturation are important parameters that determine how often a network might reach saturation and whether it can recover once saturation has been reached.

Saturation. Table 1 gives the saturation loads for the simulated routers. Results are given for only one representative Hot-Spots configuration although we simulated six different configurations.

We observe the following:

- The MA router saturates at the same or higher load than the oblivious router in all cases.
- The Chaos router saturates at the same or higher load than the MA router in all cases.
- The MA-Q router saturates at the same load as the MA router in all cases.

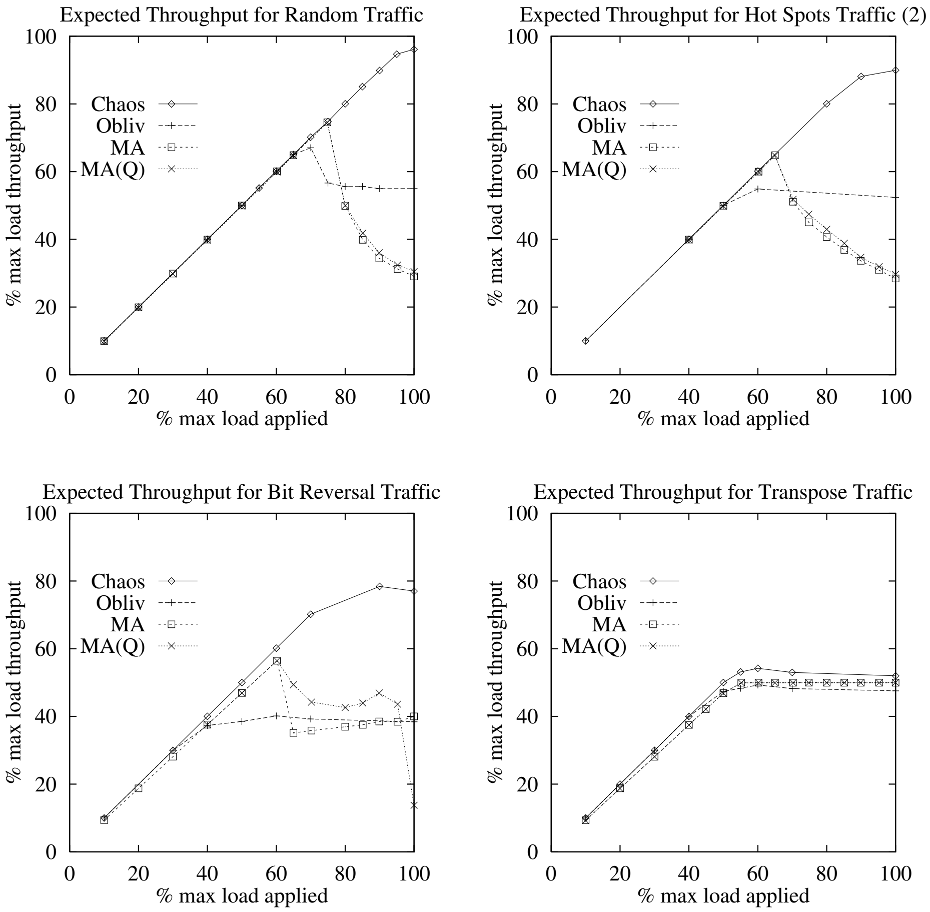

Fig. 3. Expected Throughput on a 256 Node Torus.

For Random, Bit Reversal, and Hot Spots traffic, the minimal adaptive routers saturate at loads that are 15 to 25% higher than the oblivious router, while saturating at loads 15 to 20% lower than the Chaos router.

The Transpose traffic pattern seems to be universally hard, causing all routers to saturate at a normalized load of 55%. This is because under the transpose pattern on the torus, the destination of each packet is a reflection of the source about the line $y = -x$, forcing all packets to go through the center of the network [9]. This results in a continuous area of congestion along this diagonal for all routers.

Post-Saturation. Figure 3 gives the throughput vs. normalized load plots for the simulated routers. We observe the following post-saturation characteristics:

– The Chaos router either maintains the throughput reached at saturation or experiences only slight throughput degradation in all cases.

– The oblivious router's throughput degrades significantly for Random traffic after saturation. Bolding and Snyder postulate that this throughput degradation results from network asymmetries introduced by the use of virtual channels for deadlock prevention [3].
– The MA router's throughput degrades significantly for Random, Hot Spots, and Bit Reversal traffic after saturation. We believe that this also results from network asymmetries introduced for deadlock prevention. We explain this behavior below.
– The MA-Q router's throughput does not degrade as badly as the MA router for Bit Reversal.

Two visualization tools developed in conjunction with the simulator assist in explaining the minimal adaptive router's post-saturation behavior. The first visualizer displays graphically the packet population at each node for A, B, and C buffers during the simulation. This visualizer shows that, as the applied load increases, heavy congestion develops in the A virtual channels at the center of the network[3]. Contention for B and C virtual channels is relatively light at all loads. To understand this phenomenon, we note that in general, packets can move in four directions, $++$, $+-$, $-+$, and $--$. Furthermore, recall that packets traveling in the $++$, $+-$, and $-+$ must be routed using A virtual channels until all positive moves have been exhausted. Intuitively, one can see that, at the center of the network, a majority of packets are injected in A buffers and must initially be routed using A virtual channels.

The second visualizer allows us to observe routing decisions and packet movements within a router during a simulation run. This visualizer shows that, at network saturation, packets that can be routed immediately using B channels are routed and delivered with minimal delay once they have been injected. Also, there is greater contention for A virtual channels leading in positive directions than for A virtual channels leading in negative directions. This is consistent with the routing algorithm because all packets moving in $++$, $+-$, and $-+$ directions have positive moves but not all have negative moves. Thus, this nonuniformity seems to cause negative moves to often be used first in routing a packet that have both positive and negative moves. This forces packets that could have been partly routed using B virtual channels to be entirely routed using A virtual channels.

The above observations lead us to believe that the following takes place at network saturation. The bottom right corner of the network (the white nodes in Figure 4(a)) remains clear of congestion and act as a "drain" area for the network. Congestion does not develop in this area because all injected packets quickly become routable using B virtual channels. As the applied load increases, congestion first develops at the center of the network. Packets with positive moves injected by nodes far away from the drain area (the gray nodes in Figure 4(a)) must enter or pass through the congested middle before they can be delivered. In fact, congestion becomes so bad that once a gray node injects a packet with a positive move, this packet stalls in the injection frame, preventing the node from further traffic injection. Thus, a buffer waiting pattern develops as shown in Figure 4(a) where all packets at gray-shaded nodes can only be routed using A virtual channels in the directions shown. When this waiting pattern develops, the sustained

[3] Locations within the network (e.g. center or bottom right hand corner) are interpreted with respect to a flattened torus with origin (0, 0), looking like a mesh with long wrap edges.

throughput of all gray nodes is effective determined by the throughput of the black nodes in Figure 4(a).

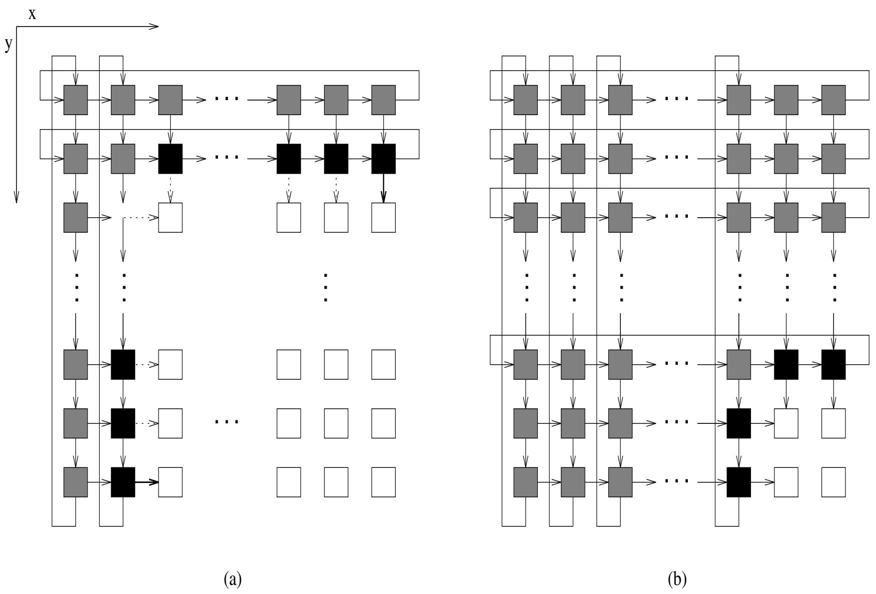

Fig. 4. Interlocking Buffer Waiting Pattern at High Loads: (a) shows interlocking buffer waiting forming at high loads causing black nodes to be the throughput bottleneck; (b) shows interlocking buffer waiting spreads as load increases, causing the congested area to have an even smaller bottleneck.

As the applied load increases beyond the saturation load, the “stalled” area shown in Figure 4(a) spreads to a larger portion of the network, further decreasing the number of nodes at the edge of the congested area as shown in Figure 4(b), causing throughput to degrade further. To validate this theory, we modified the simulator to stop generating traffic after network saturation. We then watched the network as it drained. This draining process can be observed to be the exact inverse of Figures 4(a) and 4(b).

4.2 Latency

Latency is defined as the average delay that a packet will experience in the network once it has been successfully injected by the source. Successful injection of a packet is defined as the successful transmission of that packet from the processor into the injection input frame at the router. Thus, latency does not include the time that processors must wait for input frames to become free but does include any stalling time at the injection input frame. Latency is defined in this manner so that saturated and supersaturated loads can be studied. Under saturated and supersaturated loads, the average delay time between packet creation and delivery rises to infinity and becomes a meaningless parameter.

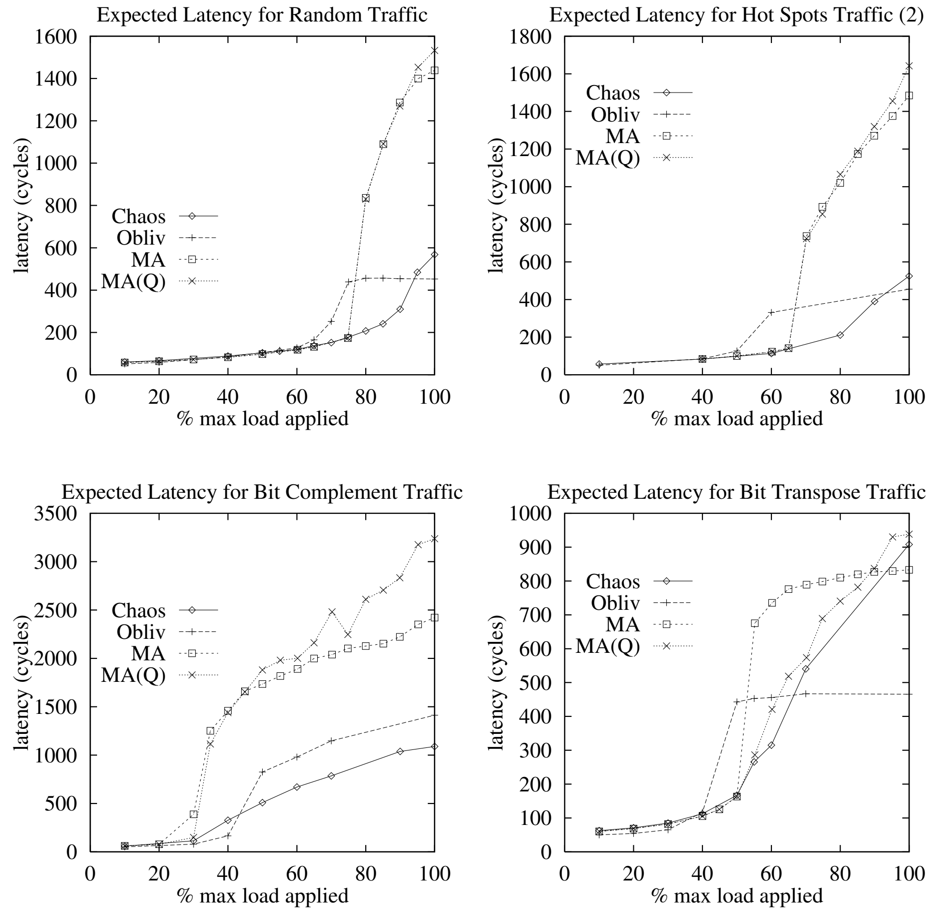

Fig. 5. Expected Latency on a 256 Node Torus.

Figure 5 shows that, before saturation, the latencies of all three routers are comparable. At low loads, latency is slightly lower for the oblivious router than the Chaos and minimal adaptive routers. This is as expected given that the oblivious router has a lower minimum cross-chip latency (3 vs. 4 cycles). For all traffic patterns, however, the oblivious router's latency soon matches that of the other three routers (at most by loads of 40%).

For all traffic patterns, latency for the oblivious and MA routers jumps sharply at network saturation. Note that, except for the Transpose traffic pattern,[4] this is also true for the MA-Q router. This is consistent with the sudden change of slope in these routers' throughput curves. Under Random and Hot Spots traffic, the minimal adaptive routers' latencies continue to rise sharply after saturation. This is consistent with the throughput

[4] Interestingly, this is the only pattern where the MA-Q router's behavior differs significantly from that of the MA router.

degradations shown by these routers' throughput curves.

For all traffic patterns, latency for the Chaos router starts to rise somewhat before saturation and continues to rise after saturation. The rise in latency corresponds to deroutes that begin to happen as the network becomes congested. In all but one case, the Chaos router's latency eventually rises above that of the oblivious router. In only one case does the Chaos router's latency rise above that of the minimal adaptive router – $> 90\%$ load under the Transpose traffic pattern.

4.3 Routing Imbalances

We have already noted that significant non-uniformities are introduced by the Cypher-Gravano algorithm. Simulation results and visualization tools have shown that such non-uniformities limit achievable throughput. Aside from this throughput limitation, however, these non-uniformities are further undersirable because they cause perceived bandwidth and latency to differ depending on nodes' locations in the network. For Random traffic, even before network saturation, packets originating from the bottom right hand corner of the torus can be expected to have lower latency than packets originating elsewhere in the network. Furthermore, even at extreme network saturation, packets injected at nodes at the bottom right hand corner of the network experience little or no network contention. This is because these packets typically require very little routing using A buffers. On the other hand, as the network approaches saturation, congestion is acute for nodes at the upper left hand corner and along the top and left edge of the network, throttling traffic injection at these nodes. As noted, at network saturation, we have observed packets stalling in injection frames for tens of thousands of cycles. One unrealistic assumption built into our simulator is the assumption that traffic injection at each node is independent of traffic injections at other nodes. In practice, processors in a multicomputer are cooperating in running parallel applications such that the above assumption of injection independence cannot hold true. In fact, unless sophisticated load-balancing techniques are applied (and perhaps even if they are applied) the progress of a parallel computation will be limited by the throughput of the congested area, suggesting that achievable throughput may be much lower the simulation results given above.

5 Related Work

Felperin et al. have reported results for a minimal adaptive routing algorithm in packet switching mesh networks [8]. In fact, the algorithm is similar to the algorithm studied here but only works for mesh and hypercube networks. Simulation results show that this algorithm achieves throughput of 75% for Random traffic and 35% for Transpose traffic before network saturation. Unfortunately, this study makes two unrealistic assumptions: delivery queues are infinitely large and cross-node latency is only one cycle. This makes it difficult to compare the results of this study to other simulation studies of adaptive routing algorithms.

In their paper, Felperin et al. also describes a fully-adaptive minimal routing algorithm, "4-classes", for torus networks. This algorithm is interesting because it requires one more virtual channel than the Cypher-Gravano algorithm but, for Random traffic, divides packets uniformly into four classes, those with only positive moves, only negative moves, positive moves in the x dimension and negative moves in the y dimension, negative moves in the x dimension and positive moves in the y dimension (i.e. x^+y^+, x^-y^-, x^+y^-, and x^-y^+). Unfortunately, Felperin et al. did not report any simulation results for this algorithm.

A number of other researchers have reported simulation results for minimal adaptive routing algorithms in wormhole routing networks [4, 1]. Unfortunately, no meaningful comparisons can be made because buffer availability and resource consumption, especially in the presence of network congestion, is so different for wormhole routing than for virtual cut-through packet-switching routing.

6 Conclusions

Although many minimal adaptive routing algorithms have been published in the last several years, relatively little is known about their performance in virtual cut-through packet-switching networks. Furthermore, because of differences in simulation assumptions, it has been difficult to compare reported performance of minimal and non-minimal adaptive routing algorithms. In this study, we have presented simulated performance results for a minimal adaptive router in a virtual cut-through packet-switching torus network and compared these results to performance data previously reported for the oblivious and Chaos routers. More importantly, all performance data cited here for the three routers were obtained using a single simulator that was designed and implemented to reflect practical router implementations.

Our data show that a minimal adaptive router based on the Cypher-Gravano algorithm outperforms the oblivious router but falls short of the Chaos router. In some cases, the minimal adaptive router saturates before the Chaos router because of its inability to make use of non-minimal paths to avoid local congestion. In other cases, our visualization tools show that, the minimal adaptive router saturates before the Chaos router because the Cypher-Gravano routing algorithm introduces significant non-uniformities in buffer usage.

We also believe that system throughput and average latency do not offer the full story about routers' performance. Our visualization efforts show that the Cypher-Gravano algorithm causes significant network imbalances. Such imbalances are undesirable in the design and implementation of real multicomputer systems. We believe that designers of adaptive routing algorithms should pay careful attention to asymmetries introduced by their algorithm. Efforts to reduce asymmetric use of network resources can pay off with higher performance and better applicability.

7 Acknowledgements

We would like to thank Kevin Bolding and Melanie Fulgham for their generous help in all phases of the project. We would also like to thank the Chaos group for helpful discussions and review. Donald Chinn first advanced the idea for our X-based visualization tools. Sung-Eun Choi made the first visualization tool a reality and setup the framework for further visualization work.

References

1. P. Berman, L. Gravano, G. Pifarré, and J. Sanz. Adaptive deadlock- and livelock-free routing with all minimal paths in torus networks. In *Proceedings of the Symposium on Parallel Algorithms and Architectures*, 1992.
2. K. Bolding, S.-C. Cheung, S.-E. Choi, C. Ebeling, S. Hassoun, T. A. Ngo, and R. Wille. The chaos router chip: Design and implementation of an adaptive router. In *Proceedings of VLSI '93*, Sept. 1993.
3. K. Bolding and L. Snyder. Mesh and torus chaotic routing. In *Proceedings of the Advanced Research in VLSI and Parallel Systems Conference*, Mar. 1992.
4. R. V. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. In *Proceedings of the International Symposium on Computer Architecture*, May 1993.
5. R. Cypher and L. Gravano. Adaptive, deadlock-free packet routing in torus networks with minimal storage. In *International Conference on Parallel Processing*, 1992.
6. W. Dally and C. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.
7. W. J. Dally and H. Aoki. Deadlock-free adaptive routing in multicomputer networks using virtual channels. *IEEE Transactions on Parallel and Distributed Systems*, 1992.
8. S. A. Felperin, L. Gravano, G. D. Pifarré, and J. L. C. Sanz. Fully-adaptive routing: Packet switching performance and wormhole algorithms. In *Supercomputing '91 Proceedings*, Nov. 1991.
9. M. L. Fulgham and L. Snyder. Performance of chaos and oblivious routers under non-uniform traffic. Technical Report 93-06-01, Department of Computer Science and Engineering, University of Washington, June 1993.
10. P. Kermani and L. Kleinrock. Virtual cut-through: A new computer communication switching technique. *Computer Networks*, 3:267–286, 1979.
11. S. Konstantinidou and L. Snyder. Chaos router: Architecture and performance. In *Proceedings of the 18th International Symposium on Computer Architecture*, pages 212–221, May 1991.
12. J. Y. Ngai and C. L. Seitz. A framework for adaptive routing in multicomputer networks. In *Proceedings of the Symposium on Parallel Algorithms and Architechtures*, pages 1–9, 1989.