BLAM : A High-Performance Routing Algorithm for Virtual Cut-Through Networks

Mithuna Thottethodi* Alvin R. Lebeck† Shubhendu S. Mukherjee‡

*School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907
mithuna@purdue.edu

†Department of Computer Science
Duke University
Durham, NC 27708-0129
alvy@cs.duke.edu

‡Intel Corporation
Shrewsbury, MA
shubu.mukherjee@intel.com

Abstract

High performance, freedom from deadlocks, and freedom from livelocks are desirable properties of interconnection networks. Unfortunately, these can be conflicting goals because networks may either devote or under-utilize resources to avoid deadlocks and livelocks. These resources could otherwise be used to improve performance. For example, a minimal adaptive routing algorithm may forgo some routing options to ensure livelock-freedom but this hurts performance at high loads. In contrast, Chaotic routing achieves higher performance as it allows full-routing flexibility including misroutes (hops that take a packet farther from its destination) and it is deadlock-free. Unfortunately, Chaotic routing only provides probabilistic guarantees of livelock-freedom.

In this paper we propose a new routing algorithm called BLAM (Bypass Buffers with Limited Adaptive lazy Misroutes) which achieves Chaos-like performance, but guarantees freedom from both deadlocks and livelocks. BLAM achieves Chaos-like performance by allowing packets to be "lazily" misrouted outside the minimal rectangle. Lazy misrouting is critical to BLAM's performance because eager misrouting can misroute unnecessarily, thereby degrading performance. To avoid deadlocks, BLAM uses a logically separate deadlock-free network (like minimal, adaptive routing), virtual cut-through, and the packet exchange protocol (like Chaos). To avoid livelocks, unlike Chaos, BLAM limits the number of times a packet is misrouted to a predefined threshold. Beyond the threshold, stalled packets are routed by the deadlock-free network to their destinations. Simulations show that our BLAM implementation sustains high throughput at heavy loads for a variety of network configurations and communication patterns.

1 Introduction

The market for large-scale cache-coherent shared-memory machines with 16 or more processors has tripled in the past four years [4]. In 2001 this server market resulted in an annual revenue of $9 billion. Roughly half of this revenue resulted from large-scale machines with 32 or more processors. Today most major vendors, such as IBM, HP, SGI and Sun Microsystems offer machines that scale up to a large number of processors (usually between 24 and 512) [4].

Demanding server applications require low latency and high bandwidth communication from interconnection networks that connect processors and memory modules in such large-scale multiprocessors. Unfortunately, network packets are often delayed due to transient network congestion. Consequently, many interconnection networks employ virtual cut-through and adaptive routing algorithms. Virtual cut-through pipelines a packet among multiple routers and buffers it entirely at a router when the packet header is blocked due to congestion. This reduces congestion around a router by removing packets from the network links. Adaptive routing routes packets around congested spots in a network (by adapting to the network state) to achieve higher throughput from the network.

Unfortunately, fully adaptive routing algorithms, without additional safeguards, are either deadlock-prone, livelock-prone, or both. Existing solutions to make interconnection networks with adaptive routing livelock-free and deadlock-free reduce the performance benefits of adaptive routing algorithms. This paper demonstrates a new adaptive routing algorithm called BLAM–*Bypass buffers with Limited Adaptive lazy Misroutes*– that achieves the performance benefits of adaptive routing without deadlocks or livelocks.

Adaptive routing algorithms can be deadlock-prone in networks, such as k-ary n-cube networks, that allow packets to create a cyclic dependence. A deadlock occurs when packets in a cyclic dependence chain cannot make forward progress *and* hold on to resources, such as buffer space, that other packets in the chain require to make forward progress. Some solutions, such as partially adaptive routing [5, 13], trade performance for deadlock freedom by limiting the algorithm's routing options to eliminate the possibility of deadlock cycles. Other solutions, such as deadlock avoidance [10] and deadlock recovery [18], can be thought of as consisting of two logical networks: one fully adaptive network and another deadlock-free network. Deadlock is not possible because stalled packets can always make forward progress on the deadlock-free network.

Other solutions, such as the one used by Chaotic routing,

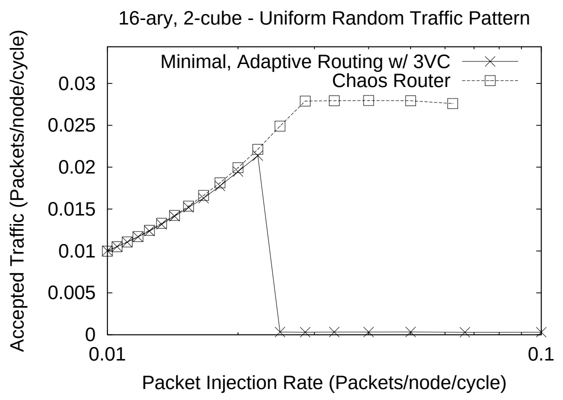


Figure 1. Minimal Adaptive Routing w/deadlock recovery and Chaotic Routing

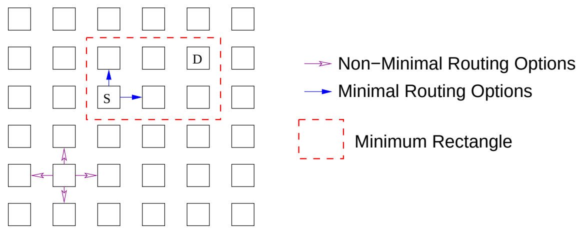


Figure 2. Routing Options Minimal vs. Non-minimal

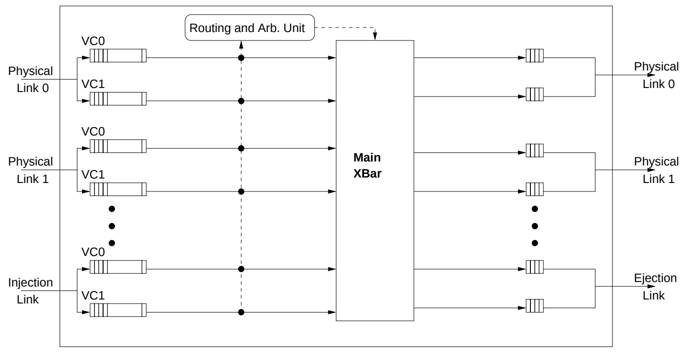


Figure 3. Base Minimal, Adaptive Router with Multiple Virtual Channels

rely on the deflection principle and the *packet exchange protocol* (see Section 2) to avoid deadlocks.

Adaptive routing algorithms can be livelock-prone if the routing algorithm does not guarantee delivery of a packet from source to destination within a finite number of hops. Livelock can never occur in minimal adaptive routing, as packets always reach their destination within a finite number of hops because every hop takes a packet closer to its destination. In contrast, non-minimal adaptive routing, such as the Chaos routing algorithm [17], is not provably livelock-free, because it allows packets to be "misrouted" outside the minimal rectangle at every hop. That is, it allows hops that takes a packet farther from its destination and, hence, does not guarantee delivery within a finite number of hops.

Interestingly, however, the Chaos routing algorithm performs significantly better than a minimal adaptive routing algorithm (Figure 1) at high offered loads. This is because the Chaos routing algorithm offers greater routing freedom compared to a minimal adaptive routing algorithm (Figure 2.) The minimal adaptive routing algorithm we simulated (Figure 1) saturates and, thereby, causes the performance to degrade rapidly beyond a certain load.

Unfortunately, in spite of its high performance, to the best of our knowledge no commercially available interconnection network uses the Chaos routing algorithm, even though it has been over a decade since the design was proposed. The presence of livelocks–however low its probability may be–causes network designers to shy away from using such algorithms in real products. The challenge is to develop a solution that has the benefits of each of the two routing algorithms (minimal adaptive and Chaos) without either technique's pitfalls.

This paper proposes a new adaptive routing algorithm called BLAM that achieves Chaos-like performance without livelocks or deadlocks. BLAM has four salient features. First, like Chaos, BLAM allows packets to be misrouted outside the minimal rectangle, thereby giving packets greater routing freedom. Second, to avoid livelocks, BLAM limits the number of times a packet is misrouted to a predefined threshold. Third, BLAM uses "lazy" misrouting in which packets are misrouted only after they fail consistently, over a period of time, to route within the minimal rectangle. Finally, BLAM uses "bypass buffers" at input ports to make sure that packets that fail to route profitably do not block the paths of other packets that follow.

The remainder of this paper is organized as follows. Section 2 provides background information and discusses related work. Section 3 studies the design space of routing techniques and advocates one point in this space as an attractive routing algorithm called BLAM. Section 4 describes a BLAM implementation. Section 5 and Section 6 present our experimental methodology and simulation results, respectively. Section 7 summarizes this paper.

2 Background and Related Work

High performance interconnection networks in tightly coupled multiprocessors can be achieved by using wormhole [8, 9] or virtual cut-through switching [15], adaptive routing [12], and multiple virtual channels [7]. Many commercial machines [21, 24, 19] use a combination of these techniques for their interconnection networks. In these systems communication occurs by sending packets of information that are routed independently through the network. Each packet is composed of flits (flow control units) that are transferred between network nodes.[1]

We can classify routers as minimal or non-minimal. A minimal router offers only profitable routing options (hops

[1]For ease of exposition we assume each network node contains a processor, memory and a network router.

that take a packet closer to its destination) whereas a non-minimal router may offer misrouting options (hops that take a packet farther from its destination) as well.

Figure 3 shows the minimal, adaptive router we use as our base case router in the rest of the paper. Each node has several incoming and outgoing network physical links and injection and ejection channels through which packets enter and exit the network. The physical channels are logically split into multiple virtual channels, each with its own buffers. The routing and arbitration unit sets up the crossbar connections linking input buffers to output buffers. Flits deposited in the output buffer are transferred across the physical link into the corresponding input buffer at the neighboring node.

Our base router and the Chaotic router may be prone to deadlocks because they use adaptive routing. Below, we discuss the two different approaches to deadlock handling that differ in their performance under heavy load and in their livelock-freedom guarantees.

One approach to handling deadlocks in adaptive channels is to guarantee that packets are able to make forward progress on a logically separate subnetwork. Deadlock avoidance [10] and deadlock recovery [18] are examples of this approach. The only difference between deadlock avoidance and deadlock recovery is that deadlock avoidance reserves per-channel resources for the deadlock-free subnetwork whereas deadlock recovery uses a per-node central flit buffer which is used only when packets are stalled.

In either deadlock avoidance or recovery, the frequency of deadlocks in the adaptive channels increases dramatically when the network reaches saturation [18]. When this occurs, packets are delivered over the relatively limited escape bandwidth available on the deadlock-free paths. This causes a sudden, severe drop in throughput and corresponding increase in packet latency. Several proposed schemes prevent the performance degradation that accompanies saturation by throttling packet injection when saturation is imminent [1, 16, 20, 25, 30]. Our approach goes further as it increases the applied load at which saturation occurs *and* avoids throughput degradation at saturation without placing limits on packet injection beyond those imposed by simple back-pressure.

Deflection routing is another class of routing algorithms that avoid deadlocks in virtual cut-through networks by ensuring that no packets are blocked indefinitely. This technique works for networks where the number of input network channels is equal to the number of output channels at each node. This property makes it possible to match every incoming packet to an output channel. However, such a matching cannot guarantee that all matched pairs correspond to profitable routes. In fact, using this approach to prevent deadlocks requires an unlimited number of misroutes. This approach eliminates the need for deadlock-free escape paths but guarantees of livelock freedom are either weaker (probabilistic) or they come at the cost of added complexity to implement timestamping and router-wide priorities.

Synchronous deflection routers [14, 26] assume that all packets arrive at an input port at the same time and they are routed to the output ports in a single step. Non-synchronous routers use the same principle of deflection routing but relax the constraint of synchronous operation by adding buffers that can hold incoming packets while waiting for output channels to become free [22, 17]. Such routers need additional mechanisms like the *packet exchange protocol* [22] to prevent deadlocks. This protocol demands that if a node a sends a packet on a link to a neighboring node b, node a should also be prepared to accept a packet from node b.

In the above discussion, we outlined a design space of routers with minimal adaptive routers – which disallow misroutes and thus eliminate livelocks – on one end, and chaotic routing – which achieves high performance and deadlock freedom by allowing unlimited, lazy misroutes but offers only probabilistic guarantees of livelock-freedom – on the other end. In the next section, we analyze this design space of routers with respect to network performance, livelock-freedom and deadlock-freedom guarantees. This analysis provides the insight necessary to develop a routing algorithm that has the best features of both classes of routers. In Section 4, we discuss various implementation options for this new design.

3 Design Space

This section examines the design space between minimal adaptive routers and chaotic routers and isolates the the key components of these routers. Section 3.1 discusses misroutes and Section 3.2 discusses how we use bypass buffers to implement lazy misrouting.

3.1 Misroutes

Minimal adaptive routers do not allow misroutes by definition because this eliminates the possibility of livelock. A packet is guaranteed to move closer to the destination in each hop. Minimal routing, combined with deadlock-freedom, guarantees that a packet will be delivered to its destination. Another disincentive for the use of misroutes is that they may waste network bandwidth since packets move farther from their destinations.

However, there are three motivations to use misroutes. First, misroutes can be used to avoid deadlocks in the adaptive channels. Chaos uses misroutes, in addition to the packet exchange protocol, to avoid deadlocks. Chaos can, thereby, avoid the use of a separate, logical deadlock-free network. Second, by allowing non-minimal routing, they can provide fault-tolerance by routing around faulty links. Third, again, by non-minimal routing, misrouting can provide higher network throughput by routing around congested areas in the network.

To use misroutes, policies that answer the following questions must be in place.

• *Should there be a limit on the number of misroutes?* Unlimited misroutes are fundamental to ensure deadlock-freedom in chaotic routing, but this results in probabilistic (i.e., not deterministic) livelock-freedom. Since our goal is to have deterministic guarantees of livelock-freedom, we place a limit on the number of misroutes a packet may take. The decision to limit misroutes removes the guarantee of deadlock-freedom on the adaptive channels. Therefore, we must include a deadlock handling mechanism.

• *What should the misroute limit be?* It is difficult (or even impossible) to recommend a single number for this limit without any information about the workload. Instead, we examine the tradeoffs involved if the limit is too high or too low. The idea is to set the limit high enough to ensure that most packets get delivered before they use all their misroutes. This reduces the latency of packet delivery from the source to the destination. In networks that have low-bandwidth deadlock-free paths, a high enough limit also ensures that these paths do not get unduly congested.

• *When is a packet misrouted? Eager* misrouting is a policy that lets packets misroute on a free channel if they cannot obtain a free profitable channel. *Lazy* misrouting policies impose some other condition that delays using misroutes.

In general, the choice of when packets are misrouted depends on the motivation for misroutes. When the purpose for misroutes is fault tolerance, an *eager* misrouting strategy may be sufficient. (If the only profitable channels for a given packet are known to be faulty, there is no point in delaying the misroute.)

Our purpose is to achieve high performance and to avoid deadlocks in adaptive channels. Anjan and Pinkston [25] have shown that eager misrouting can hurt performance for uniform traffic. As such, we want a *lazy* misrouting strategy that postpones misrouting until it is either hurting performance because of packets stalling behind it or because misrouting is mandated by the packet exchange protocol to avoid deadlocks in adaptive channels. This lazy misrouting strategy minimizes wasted network bandwidth. Note, Chaos also uses lazy misrouting.

With a *lazy* misrouting strategy, it is important to ensure that a blocked packet waiting to be misrouted does not block other packets that could otherwise make forward progress. We achieve this by using bypass buffers. Our experiments show that lazy misrouting without bypass buffers decreases performance.

3.2 Bypass buffers

We use bypass buffers to facilitate lazy misrouting. A bypass buffer allows a blocked packet to "step aside" from the critical path of other packets by buffering the blocked packet and releasing the input buffer. Once a packet enters a bypass buffer, packets behind it can use the free input buffer and bypass the blocked packet if they find profitable channels. Packets resident in these bypass buffers are candidates for lazy misrouting.

Bypass buffers have a secondary effect of increasing the total amount of buffer space at an input port. This may help improve performance. However, in our experience, if the number of input buffers are chosen appropriately (perhaps using Little's Law), then additional buffering provides no or marginal improvement in performance. We demonstrate this effect in Section 6.3 by adding bypass buffers (but with no misrouting) to a minimal adaptive routing algorithm.

Below, we examine the policies needed to manage bypass buffers for lazy misrouting.

• *When do packets enter the bypass buffers?* Packets in input buffers that are unable to make progress on profitable channels move to bypass buffers when they have waited "sufficiently long" or for implementing the packet exchange protocol. The Chaos router considers a packet to have waited sufficiently long at a node if the whole packet (including the tail flit) has arrived at that node. We use the same definition.

The packet exchange protocol dictates that if a node sends a packet to a neighboring node, it should also be prepared to accept a packet from that node. To this end, when a packet is sent out on an output channel, an input channel in the reverse direction (on the same physical link) should be made free in anticipation of an incoming packet. To do so, we move any packet that is in the input buffer to a bypass buffer.

• *Should packets in the bypass buffers have priority?* If both the bypass buffer and its corresponding input buffer have packets to nominate to a particular virtual channel, then the routing algorithm must decide which packet to pick. One policy is to use a fair-mechanism like round robin among all input and bypass buffers. Another option is to give priority to packets in the bypass buffers, since these packets are older. Routing older packets first is a good heuristic to achieve better performance. The *Rotary Rule* mode of the Alpha 21364 network uses a similar heuristic to assign higher priority to packets arriving from a network link than to new packets trying to enter the network [21].

Priorities for broad classes of packets (e.g., priority packets in bypass buffers over packets in input buffers, packets arriving from network link over packets arriving from node, coherence replies over requests, etc.) can be implemented in simple and scalable ways. However, router-wide priorities (e.g., priority for "oldest" packet) are not only more complex designs, they also require central structures that can become clock-scaling bottlenecks.

3.3 Summary

From the above discussion, we see that there exists a potential design point between a minimal adaptive router and the chaos router. This router uses **B**ypass buffers, with **L**imited, **A**daptive, lazy **M**isroutes and deadlock handling (**BLAM**). Limited misroutes gives BLAM three advantages: livelock-freedom (compared to chaos), more routing flexibility and reduced frequency of use of deadlock-free escape paths (compared to the base router). Lazy misrouting with bypassing is an important feature of BLAM that enables chaos-like high performance.

4 A BLAM Implementation

In this section, we describe an implementation of BLAM that uses distributed bypass buffers. We first examine the implementation used in chaos and then suggest our own.

The original *Chaos* router design for two dimensional networks augments a basic router with an additional central *multiqueue* to provide a central pool of bypass buffers (see Figure 4). This central *multiqueue* requires additional routing logic and a cross bar on the input side of the queue. Note,

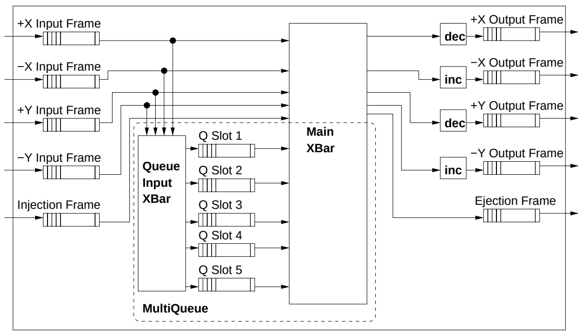


Figure 4. Central Bypass Buffers: Chaos

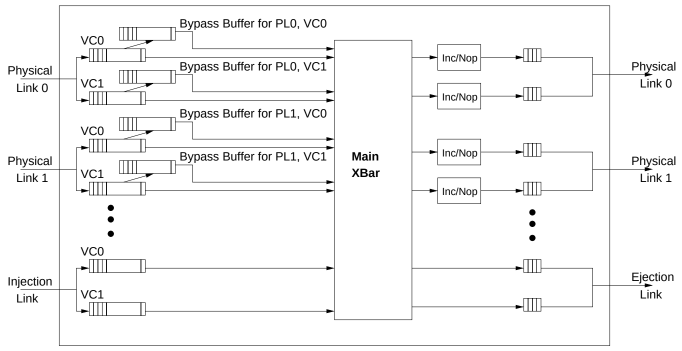


Figure 5. Distributed Bypass Buffers

Figure 4 shows only the datapath of the two-dimensional chaotic router. The routing and arbitration units for the two crossbars are omitted. The *Chaos* router does not use multiple virtual channels per physical channel. As such, the input buffers (or frames) are associated with the physical channel. The four network physical channels that connect to neighboring nodes are marked with labels that indicate the dimension (X or Y) and direction (positive or negative) they traverse. Each header contains the number of hops required in each dimension to reach the destination. This header information is modified (incremented or decremented) appropriately depending on its output channel.

For our implementation (Figure 5), we use distributed bypass buffers rather than the centralized pool approach of the *Chaos* router. Previous research [23, 27] has recommended centralized buffer-pools over distributed buffers arguing that dynamic sharing of the central buffer pool leads to more efficient use of buffers. However, buffer efficiency is not a critical concern when we consider on-chip routers (such as the Alpha 21364 router [21]) where additional buffers are cheap. With enough buffers, a shared buffer pool and distributed buffers should be similar in performance. Further, central structures (queues, implementation of router-wide priorities, etc.) become bottlenecks for clock scaling. Consequently, routers with distributed buffers are attractive design points [21] and distributed bypass buffers are a natural fit for such designs.

Figure 5 shows the datapath of the BLAM router. Since BLAM permits a finite number of misroutes for each packet, the header maintains a count of the number of misroutes taken. This is incremented (in the *Inc/Nop* block) each time the packet is misrouted. No change is required for profitable hops. We have an additional bypass-buffer associated with each input buffer belonging to an adaptive virtual channel. Bypass buffers are not associated with injection channels. Note, Figure 5 assumes fully adaptive routing with the deadlock recovery scheme. Therefore, all virtual channels are adaptive channels with associated bypass buffers. If deadlock avoidance is used, there are no bypass buffers associated with the deadlock-free channels.

The operation of bypass buffers is similar to the chaotic router implementation in some respects. Packets move from the input buffers to the bypass buffers when the whole packet has arrived at the node or when such a transfer is necessary due to the packet exchange protocol. In the common case, when packets are making forward progress, the bypass buffers are not on the critical path and packets go directly from the input buffers to the output buffers. Packets in the bypass buffers have priority over packets in the input buffers when they compete for the same output channel. A packet in a bypass buffer is misrouted when the corresponding input buffer entry wants to enter the bypass buffer (either due to stalling or due to the packet exchange mechanism.)

Unlike Chaos, BLAM's one-to-one correspondence between adaptive virtual channel input buffers and bypass buffers eliminates the need for the queue input cross bar and associated routing logic. However, this approach removes the element of randomization present in chaotic routing. In chaotic routing, when a packet needs to enter the "multiqueue" (either due to stalling or due to the packet exchange protocol) and the "multiqueue" is full, one entry is selected at random to be misrouted. This is fundamental to the probabilistic guarantees of livelock-freedom. Since our implementation allows a packet to move to only one possible bypass-buffer, the packet in that bypass buffer must be selected for misrouting and there is no scope for randomization. However, our design provides deterministic guarantees of livelock-freedom without using randomization because of the limit on the number of misroutes.

5 Evaluation Methodology

To evaluate the various routing schemes we use the `flexsim` simulator [28] and the chaos simulator available from the University of Washington [3]. The Chaos router is simulated on the chaos simulator. We use the `flexsim` simulator for all other configurations. All simulations execute for 60,000 cycles. However, we ignore the first 10,000 cycles to eliminate warm-up transients. We have verified, for a subset of experiments, that longer simulations of 300,000 cycles do not change the results significantly. (Bandwidth is within 2% and latency is within 4% of the results from a 60,000 cycle simulation.) We use an extension of the *Chaos Normal Form (CNF)* [2] standard for presenting our results. We present two graphs (throughput vs. applied load and latency vs. applied load) for each configuration. The injection/delivery rate axes use absolute throughput values but there are additional lines to show CNF's definition of 100% throughput as well.

The offered load consists of each node generating 16-flit packets at the same fixed rate for the duration of the simulation. Our goal is to show that our results are valid for a variety of loads. Ideally, we would like to measure interconnect performance by using real communication workloads from parallel applications. Unfortunately, due to inadequate simulation infrastructure and problems associated with trace-based simulation [6], it is the state-of-the-practice to evaluate interconnection network performance with synthetic communication workloads that are "difficult" (i.e., they increase contention for resources, resulting in sub-optimal/worst-case performance [31]) and/or "useful" (i.e., they correspond to the communication pattern for various parallel numerical algorithms [11]).

Apart from the widely used, *uniform random* traffic pattern, we consider three synthetic communication patterns, *bit-reversal*, *perfect-shuffle* and *complement* to stress the network in non-uniform ways. The communication patterns differ in the way a destination node is chosen for a given source node with bit co-ordinates $(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)$. The bit co-ordinates for the destination nodes are $(a_{n-2}, a_{n-3}, \ldots, a_0, a_{n-1})$ for *perfect shuffle*, $\overline{(a_{n-1}, a_{n-2}, \ldots, a_1, a_0)}$ for *complement* and $(a_0, a_1, \ldots, a_{n-2}, a_{n-1})$ for *bit-reversal*.

We use a minimal, fully-adaptive router (Figure 3) as the base configuration. In this paper, we evaluate the base (minimal, adaptive) and BLAM network configurations with the Disha [18] progressive deadlock recovery scheme with a time-out of 25 cycles. Also, in this paper, we present detailed results and analysis for a 16-ary, 2-cube only. Results for other configurations (deadlock avoidance) and network sizes (8-ary, 3-cube and 32-ary, 2-cube) are not included in this paper due to space limitations and have been reported separately [29].

Each router has one injection channel (through which packets sent by that node enter the network) and one delivery channel (through which packets sent to that node exit the network). The router's source queue holds upto 1024 packets. We use edge-buffers (buffers associated with virtual channels) that can hold an entire packet. There is a one cycle arbitration delay and a one cycle routing delay per packet. It takes one cycle per flit to traverse the cross-bar switch and one cycle per flit to traverse a physical link. Each physical link is full duplex. (Note, our configuration with full-duplex links (i.e., two unidirectional links) between neighbors is not the same configuration reported in some Chaos papers. They use bi-directional links and hence the results are not comparable.)

6 Simulation Results

This section presents our simulation results. We begin by examining the overall performance of the BLAM router. This is followed by a comparison of BLAM with three virtual channels (i.e., three input and bypass buffers each per physical channel) against a minimal, adaptive router with six virtual channels. We call this a "resource-neutral" comparison, because these two configurations have the same number of buffers and crossbar inputs. Next, we evaluate the effects of varying the maximum number of allowed misroutes. Finally, we dissect BLAM performance by examining the effects of adding *eager* misroutes, *lazy misroutes* and bypass buffers, in isolation, to the base router.

The primary conclusions from our simulations are: (1)The BLAM router sustains near-*Chaos* throughput for all considered communication patterns at high offered load levels where the base case suffers a drop in throughput. (2)The combined use of lazy misroutes and bypass buffers are necessary for high performance. (3) We isolate the effects of misroutes to validate and extend previous research findings that misroutes alone, whether *eager* or *lazy*, decreases performance.

6.1 Overall Performance

This section examines the performance of a complete BLAM router implementation with a limit of 16 misroutes on a 16-ary, 2-cube with 16-flit packets. Figure 6 shows the bandwidth (left graph) and latency (right graph) for the four communication patterns for base case and BLAM configurations with deadlock-recovery. Note the logarithmic scale used on the y-axis for the latency graphs.

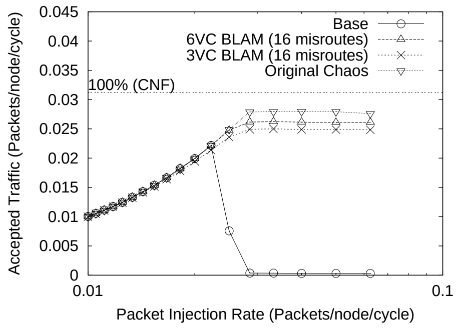
The curve for the base router ($\circ$) in Figure 6 illustrates the network saturation problem as described earlier. There are two curves corresponding to the BLAM router: one for the configuration with three virtual channels (3VC) per physical channel ($\times$ in Figure 6) and another for the configuration with six virtual channels (6VC) per physical channel ($\triangle$ in Figure 6). Both curves show that our design is able to prevent the drop in performance that occurs at high loads due to deadlocks in adaptive channels. Furthermore, BLAM achieves performance comparable or superior to chaos while providing deterministic guarantees.

Similarly, simulations with deadlock avoidance configurations show that BLAM (on top of deadlock avoidance) outperforms the base deadlock avoidance configuration [29].

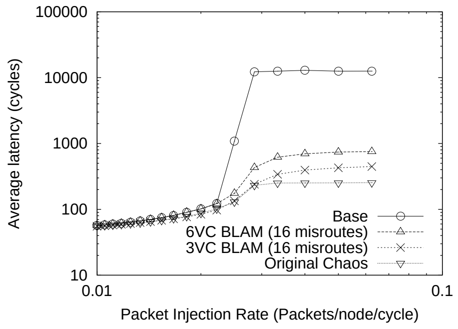
6.1.1 Throughput/Latency Tradeoff

Figure 6 also demonstrates a throughput/latency tradeoff based on the number of virtual channels. The 6VC configuration shows higher throughput than the 3VC configuration, however this comes at the cost of higher latency. The reason is the 6VC BLAM has a higher number of bypass-buffers, and a packet is able to stay in a buffer for a longer period of time. Recall, a packet is misrouted only when another packet needs to enter the same bypass buffer. We verified this phenomenon by measuring the amount of time a packet spends in the bypass-buffers. For the six heaviest applied loads in Figure 6, where the difference in latencies becomes obvious, packets stay in the bypass buffers 35%, 77%, 55% and 48% longer, on average, in the 6VC configuration than in the 3VC configuration for the *uniform random, perfect shuffle, complement* and *bit reversal* traffic patterns, respectively. To better understand this throughput/latency tradeoff, we compare the bandwidth and latency of 6VC BLAM and 3VC BLAM to the throughput and latency of Chaos for the six heaviest applied loads.

Uniform Random Traffic

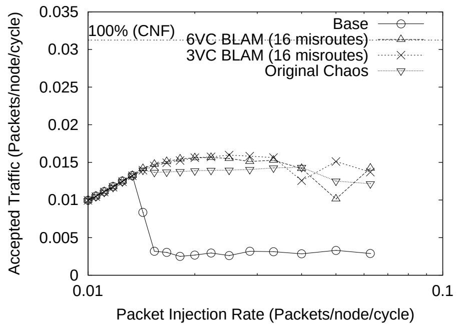


(a) Delivered Throughput vs. Offered Load

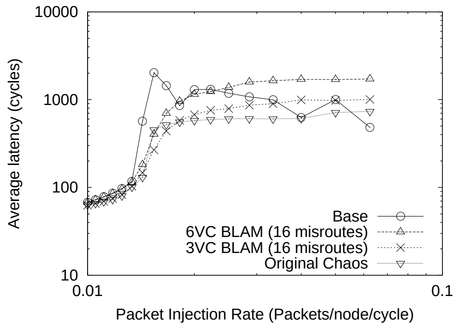


(b) Average Latency vs. Offered Load

Perfect Shuffle Communication Pattern

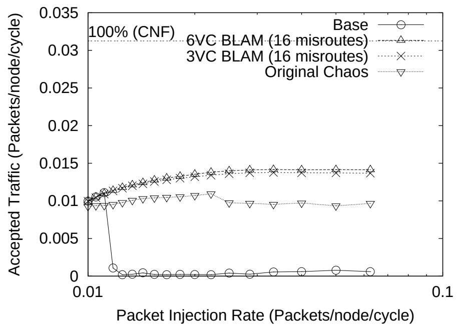


(c) Delivered Throughput vs. Offered Load

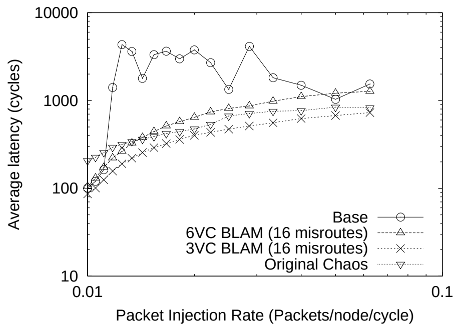


(d) Average Latency vs. Offered Load

Complement Communication Pattern

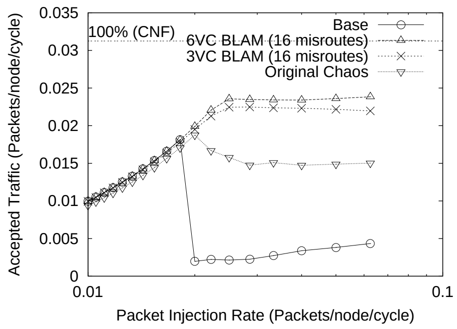


(e) Delivered Throughput vs. Offered Load

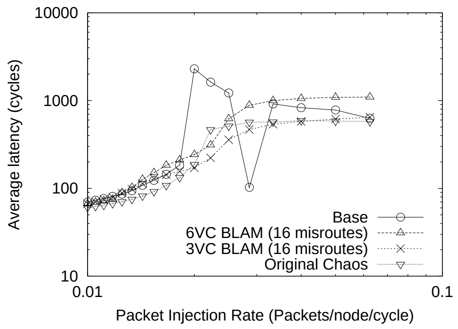


(f) Average Latency vs. Offered Load

Bit Reversal Communication Pattern



(g) Delivered Throughput vs. Offered Load



(h) Average Latency vs. Offered Load

Figure 6. Overall Performance w/ Deadlock Recovery

6VC BLAM achieves better throughputs than *chaos* for three of the four communication patterns. The throughput improves by as much as 56% (for *bit-reversal*) and is 5% lower only in the case of *uniform random* traffic. The behavior of 3VC BLAM is qualitatively similar with slightly lower throughputs. For 3VC BLAM, the throughputs vary from 10% lower to 48% higher than *chaos*.

The latencies for 6VC BLAM are between 35% (for *bit-reversal*) and 151% higher (for *perfect shuffle*) than Chaos latencies, on average. A similar latency comparison for 3VC BLAM shows that it achieves latencies varying from 22% lower (for *complement*) to 42% higher (for *perfect shuffle*) than Chaos latencies. In conclusion, we see that 6VC BLAM suffers from increased latency to achieve better-than-*chaos* throughput. In contrast, 3VC BLAM achieves much better latencies than 6VC BLAM for a small penalty on throughput.

6.1.2 Resource-Neutral Comparison

Comparing the six virtual channel base case against the BLAM router with six virtual channels is not a resource-neutral comparison. This is because the BLAM requires additional bypass buffers for each virtual channel, in this case doubling the number of buffers. BLAM also requires a larger crossbar with twice as many inputs: the base virtual channel inputs and the bypass buffer inputs. A resource neutral comparison can be made between the six virtual channel minimal adaptive router and the BLAM router with three virtual channels. From Figure 6 this resource neutral comparison reveals that the 3VC BLAM router ($\times$) outperforms the base router ($\circ$) for all traffic patterns. The 3VC BLAM router maintains high throughput at higher load levels whereas the 6VC base router saturates.

Due to space limitations, in the remainder of this paper we present results only for the *uniform random* and *perfect shuffle* communication patterns. Results for the other two patterns reveal similar conclusions.

6.2 Varying the Misroute Limit

The previous results use a BLAM router with up to 16 misroutes. In this subsection, we examine the trade-offs involved in varying this limit. Figure 7 shows the performance of 6VC BLAM as the misroute limit is increased progressively from 3 to 8 to 16. In general, we see that increasing the number of misroutes postpones saturation as packets can use the adaptive channels for a longer period of time.

For *uniform random* traffic, a limit of three misroutes is enough to prevent saturation. Increasing the number of misroutes beyond three does not change the behavior in any significant manner. In contrast, the *perfect-shuffle* communication pattern (see Figures 7c & 7d) shows well-separated performance curves as we vary the misroute limit. This facilitates explanation of network behavior in response to variations in the misroute limit.

As the misroute limit increases, network saturation (and the corresponding drop in performance) occurs at higher and higher loads. We see that 3 misroutes prevents saturation at certain loads. But at higher loads, the packets use up all the allowed misroutes and are then constrained to route only within the minimum rectangle. This increases the frequency of deadlock cycles forming in the adaptive channels and hence increases the frequency of deadlock-free channel usage. Similarly, we see that 8 misroutes, while better than 3 misroutes, is unable to prevent saturation at the higher loads.

Note, 16 misroutes is high enough for all the loads and communication patterns we considered. In general, the number of misroutes should be set to a value that is high enough to reduce the probability of using the deadlock-free escape paths for any workload that the network may handle. However, for some workloads the network could saturate but the resulting behavior will be no worse than that of a minimal adaptive router. Theoretically we may see high latencies if the limit on misroutes is very high, because our implementation does not have the randomization property of the *Chaos* router which increases the probability of packet delivery with increasing time. However, we have not seen this in practice.

6.3 Effect of Adding Bypass Buffers

The bypass buffers create additional buffering capacity in each node which can break some hold-and-wait cycles, thus improving performance over the base case. The curve for zero misroutes (∇) in Figure 7 isolates the effect of only adding bypass buffers to the base router. We observe that while there is some marginal improvement over the base case, increasing network load eventually saturates the additional buffers.

6.4 M-misroute, Adaptive router

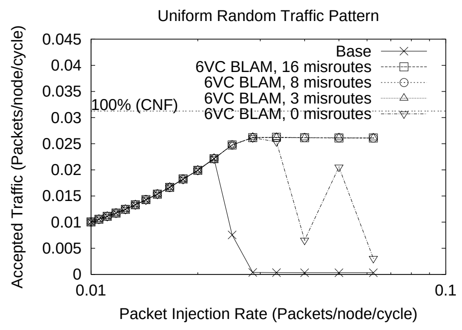
We now evaluate the effects of adding misroutes (both *eager* and *lazy*) without bypass buffers to the base router. While we present simulation results for the *eager* misroutes, we qualitatively describe our experiment and results for *lazy* misroutes.

Eager Misroutes: In this scheme misroutes are initiated eagerly whenever a packet is unable to find a profitable route. Figure 8 shows our results. From these simulations we see that eager misrouting without bypass buffers is insufficient to prevent saturation. In fact, eager misrouting consistently performs worse than the base router, and increasing the misroute limit further exacerbates saturation. This behavior matches our expectations.

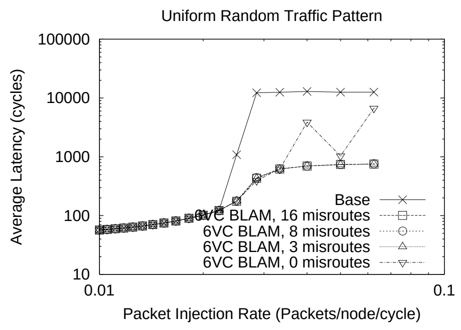
Previous research [18] shows that starting out with a minimal adaptive router (w/ wormhole routing and Disha deadlock recovery) and enabling a limited number of misroutes does not improve performance for uniform random traffic. Our experiments reproduce this result for virtual cut-through switching and other traffic patterns. We do note, that the previous work did show that misroutes can be helpful in the case of hot-spot traffic pattern (where one node is the destination for packets coming from many source nodes.) We do not model that specific pattern.

Lazy misroutes: Our BLAM results combine the effects of bypassing and *lazy* misrouting. To separate the effects of bypassing from *lazy* misrouting we modified the eager misrouting technique to delay misrouting for a set number of

Uniform Random

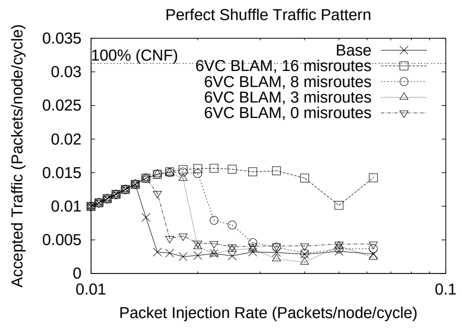


(a) Delivered Throughput vs. Offered Load

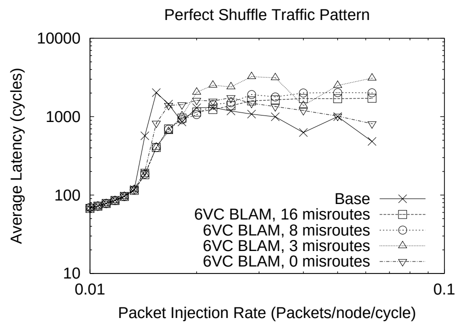


(b) Average Latency vs. Offered Load

Perfect Shuffle



(c) Delivered Throughput vs. Offered Load



(d) Average Latency vs. Offered Load

Figure 7. Effects of Varying the Misroute Limit of BLAM

cycles—called spin-cycles. During this time the packet continues to try and obtain a profitable route but packets that follow cannot bypass the stalled packet. Our results show that while *lazy* misrouting is better than *eager* misrouting, it is still worse than the base case [29].

7 Conclusion

High performance, deadlock-freedom, and livelock-freedom are all important for multiprocessor interconnection networks. Unfortunately, existing routing algorithms trade off one property for others. Minimal, adaptive routing algorithms compromise performance at high loads to guarantee deadlock-freedom and livelock-freedom. In contrast, Chaotic routing algorithms accept weaker, probabilistic livelock-freedom guarantees to achieve high performance and deadlock-freedom.

This paper proposes a new routing algorithm—called BLAM (Bypass buffers with Limited Adaptive lazy Misrouting)—which combines the best of minimal adaptive routing and Chaotic routing to provide high performance without sacrificing deadlock- or livelock-freedom. Like minimal adaptive routing algorithms, BLAM provides deadlock-freedom by the use of a deadlock-free escape paths. Like Chaos, it provides high performance via the use of lazy misrouting and the *packet exchange protocol*. BLAM implements lazy misrouting via bypass buffers at the input ports. However, unlike Chaos, BLAM limits the number of misroutes, thereby, providing livelock-freedom.

Using simulation of a variety of configurations and communication patterns, we demonstrate that BLAM achieves very high network performance, which is comparable to what Chaos can achieve. Additionally, we demonstrate that components of BLAM–bypass buffers and lazy misrouting–may not necessarily improve performance individually. However, when combined in BLAM, these techniques provide performance similar to Chaotic routing algorithms, but with guaranteed livelock- and deadlock-freedom.

References

[1] E. Baydal, P. Lopez, and J. Duato. A Simple and Efficient Mechanism to Prevent Saturation in Wormhole Networks. In *Proceedings. 14th International Parallel and Distributed Processing Symposium*, pages 617–622, 2000.

[2] The Chaotic Routing Project, Computer Science and Engineering Department, University of Washington. *Standard for Presentation of Results*. http://www.cs.washington.edu/research/projects/lis/chaos/www/presentation.html.

[3] The Chaotic Routing Project, Computer Science and Engineering Department, University of Washington, Seattle. *The Chaos Router Simulator*. http://www.cs.washington.edu/research/projects/lis/chaos/ www/simulator.html.

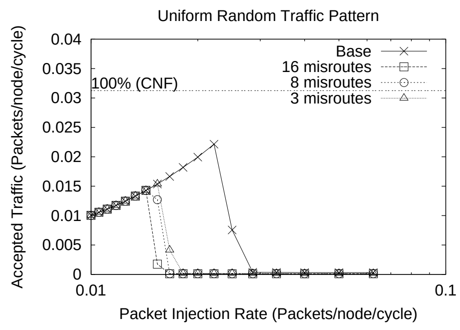
[4] A. Charlesworth. The Sun Fireplane Interconnect. *IEEE Micro*, 22(1):36–45, January/February 2002.

[5] A. A. Chien and J. H. Kim. Planar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors. In *Proceedings of the 19th Annual International Symposium on Computer Architecture*, pages 268–277, May 1992.

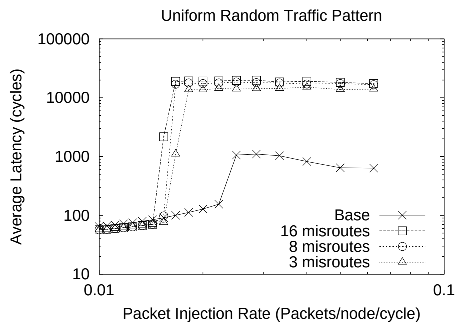
[6] A. A. Chien and M. Konstantinidou. Workloads and Performance Metrics for Evaluating Parallel Interconnects. *IEEE Computer Architecture Technical Committee Newsletter*, pages 23–27, Summer-Fall 1994.

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

Uniform Random

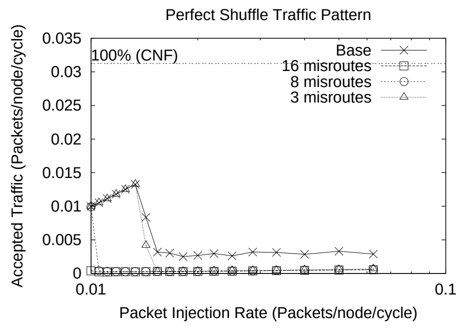


(a) Delivered Throughput vs. Offered Load

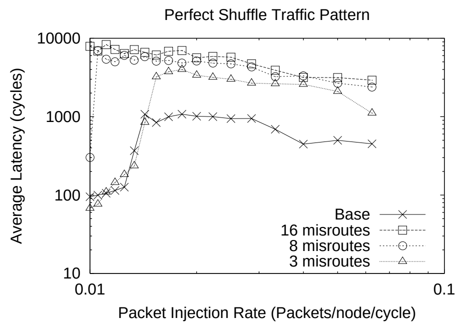


(b) Average Latency vs. Offered Load

Perfect Shuffle



(c) Delivered Throughput vs. Offered Load



(d) Average Latency vs. Offered Load

Figure 8. M-misroute, Adaptive Router

[8] W. J. Dally and C. L. Seitz. The TORUS routing chip. *Journal of Distributed Computing*, 1(3):187–196, October 1986.

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

[10] J. Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, December 1993.

[11] J. Duato, S. Yalamanchili, and L. Ni. *Interconnection Networks: An Engineering Approach*, page 405. IEEE Computer Society Press, 1997.

[12] P. T. Gaughan and S. Yalamanchili. Adaptive Routing Protocols for hypercube Interconnection Networks. *IEEE Computer*, 46(2):12–22, 1997.

[13] C. J. Glass and L. M. Ni. The Turn Model for Adaptive Routing. In *Proceedings of the 19th Annual International Symposium on Computer Architecture*, pages 278–287, May 1992.

[14] A. G. Greenberg and B. Hajek. Deflection routing in hypercube networks. *IEEE Transactions on Communications*, COM-40(6):1070–1081, June 1992.

[15] P. Kermani and L. Kleinrock. Virtual Cut-Through : A New Computer Communication Switching technique. *Computer Networks*, 3:267–286, 1979.

[16] J. H. Kim, Z. Liu, and A. A. Chien. Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing. In *Proceedings of the 21st International Symposium on Computer Architecture*, April 1994.

[17] S. Konstantinidou and L. Snyder. The Chaos Router. *IEEE Transactions on Computers*, 43(12):1386–1397, December 1994.

[18] Anjan K.V. and T.M. Pinkston. An Efficient, Fully Adaptive Deadlock Recovery Scheme : Disha. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 201–210, June 1995.

[19] J. Laudon and D. Lenoski. The SGI Origin: A ccNUMA Highly Scalable Server. In *Proceedings of the 24th International Symposium on Computer Architecture*, pages 241–251, June 1997.

[20] P. Lopez, J. M. Martinez, J. Duato, and F. Petrini. On the Reduction of Deadlock Frequency by Limiting Message Injection in Wormhole Networks. In *Proceedings of Parallel Computer Routing and Communication Workshop*, June 1997.

[21] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb. The 21364 Network Architecture. *IEEE Micro*, 22(1):26–35, January/February 2002.

[22] J. Y. Ngai and C. L. Seitz. A Framework for Adaptive Routing in Multicomputer Networks. In *Proceedings of the 1st Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 1–9, June 1989.

[23] A. G. Nowatzyk, M. C. Browne, E. J. Kelly, and M. Parkin. S-Connect: from Networks of Workstations to Supercomputer Performance. In *Proceedings of the 22nd Annual International Symposium on Computer Architecture*, pages 71–82, June 1995.

[24] S. L. Scott. Synchronization and Communication in the T3E Multiprocessor. In *Proceedings of the Seventh Internation Conference on Architectural Support for Programming Languages and Operating Systems*, pages 26–36, October 1996.

[25] A. Smai and L. Thorelli. Global Reactive Congestion Control in Multicomputer Networks. In *5th International Conference on High Performance Computing*, pages 179–186, 1998.

[26] B. J. Smith. Architecture and Applications of the HEP Multiprocessor Computer System. In *Proceedings of SPIE*, pages 241–248, 1981.

[27] C. B. Stunkel, D. G. Shea, and B. Abali et al. The SP2 High Performance Switch. *IBM Systems Journal*, 34(2):185–204, 1995.

[28] The Superior Multiprocessor ARchiTecture (SMART) Interconnects Group, Electrical Engineering - Systems Department, University of Southern California. *FlexSim*. http://www.usc.edu/dept/ceng/pinkston/tools.html.

[29] M. Thottethodi. *Techniques for High Bandwidth, Low Latency Interconnection Network Operation at High Offered Loads.* PhD thesis, Duke University, December 2002.

[30] M. Thottethodi, A. R. Lebeck, and S. S. Mukherjee. Self-Tuned Congestion Control for Multiprocessor Networks. In *Proceedings of the Seventh International Symposium on High Performance Computer Architecture (HPCA-7)*, pages 107–118, January 2001.

[31] B. Towles and W. J. Dally. Worst-case Traffic for Oblivious Routing Functions. *Computer Architecture Letters*, 1, February 2002.