

# The Impact of Output Selection Function Choice on the Performance of Adaptive Wormhole Routing\*

Loren Schwiebert and Renelius Bell  
Department of Electrical and Computer Engineering  
Wayne State University  
Detroit, MI 48202–3902

Email: loren@ece.eng.wayne.edu, bell@ernie.eng.wayne.edu

## Abstract

*Many adaptive routing algorithms have been proposed for wormhole-routed interconnection networks. Comparatively little work, however, has been done on determining how the output selection function (routing policy) affects the performance of an adaptive routing algorithm. In this paper, we present a detailed simulation study of various selection functions for a fully adaptive mesh routing algorithm. The simulation results show that the choice of selection function has a significant effect on the average message latency. Thus, a naive implementation of an adaptive routing algorithm may lead to poor performance. These selection functions are also compared with a theoretically optimal selection function [1]. We show that although theoretically optimal, the actual performance of the optimal selection function is not best. An explanation and interpretation of the results is provided.*

## 1 Introduction

Wormhole routing [5] has become the switching technique of choice in most modern distributed-memory and distributed-shared-memory multiprocessors. Wormhole routing moves messages through the network by dividing each message into flits, where a flit is typically a small multiple of the width of a physical channel. The header flit of a packet contains the routing information and the data flits of the packet follow the header flit through the network. The major advantage of wormhole routing is that when the header arrives at an intermediate router, the router forwards the message header to a neighboring router as soon as an output channel the message can use is available.

Since the flits of a message are forwarded as soon as possible, the message latency is largely insensitive to the distance between the source and destination. Virtual cut-through also allows messages to progress as soon as an

output channel is available and buffers packets only when the output channel is busy. Wormhole routing, however, requires only enough storage at each router to buffer a few flits, rather than the entire packet. The low minimum latency and modest buffering requirements account for the popularity of wormhole routing in distributed-memory multiprocessors. See Ni and McKinley [10] for a detailed explanation of wormhole routing.

Wormhole routing is susceptible to contention even with moderate traffic, which results in higher message latency. Because message headers are forwarded immediately, a message can span many channels simultaneously. Furthermore, a message that is blocked remains in the network and the message continues to hold all the channels it currently occupies. Thus, a message that traverses several channels could block many other messages.

Dally [4] proposes a cost-effective method of reducing contention by allowing multiple *virtual* channels to share the same physical channel. A virtual channel is really just a buffer on a physical channel. By providing multiple virtual channels on the same physical channel, multiple messages can be multiplexed over the same physical channel and thus bypass blocked messages. Contention can be further reduced by permitting a message to choose from among the multiple paths in the network between the source and destination, which also reduces the message latency and makes better use of the network channels. Dally and Seitz [5] have shown, however, that a deadlock configuration can be formed if no restrictions are placed on the use of channels.

Oblivious routing algorithms define a single path between a source and destination. Adaptive routing algorithms, on the other hand, support multiple paths between a source and destination. Adaptive routing algorithms are either minimal or nonminimal. Minimal routing algorithms allow only shortest paths to be chosen, while nonminimal routing algorithms do not require messages to use only shortest paths. Adaptive routing algorithms can be further differentiated by the number of shortest paths allowed.

---

\*This research was supported in part by National Science Foundation Grant #HRD-9450371.

*Partially adaptive* routing algorithms do not allow *all* messages to use *any* shortest path. *Fully adaptive* routing algorithms *do* allow all messages to use any shortest path. Although all fully adaptive routing algorithms allow a message to route to any neighboring router toward the destination, different restrictions are placed on the choice of channels that can be used to reach the next router.

When an adaptive routing algorithm is used, many messages can choose from multiple output channels. The output selection function chooses one of these output channels for the message.<sup>1</sup> The choice usually depends on the status of the output channels, since the objective is to select an available output channel rather than a busy channel. When multiple output channels are available, one of these available output channels is chosen based on the selection function. Some authors [1, 16] have referred to this decision process as the routing policy.

Previous research on routing algorithms for wormhole routing has produced many different adaptive routing algorithms. Adaptive routing algorithms have been compared with each other and with oblivious routing based on the average message latency. Little research has been done, however, on how different selection functions effect the performance of an adaptive routing algorithm. In this paper, we present a simulation study of different selection functions for the same fully adaptive routing algorithm. In addition, we compare these selection functions with a theoretically optimal selection function and show that although theoretically optimal, the performance is not the best and in some cases actually provides the worst performance.

## 2 Previous Work

Many adaptive routing algorithms have been proposed, including [3, 6, 8, 14, 15]. Comparatively little work, however, has been done on output selection function design and performance analysis. We briefly review the existing work on selection functions as well as related topics.

Some *theoretical* work has been done on the design of output selection functions. The *zigzag* selection function was shown by Badr and Podar [1] to be *optimal* for meshes. *Zigzag* is optimal in that it maximizes the probability of reaching the destination without any delays. Wu [16] has shown that for 2D meshes, this is equivalent to maximizing the number of remaining paths to the destination.

Glass and Ni [8] present several partially adaptive routing algorithms and simulate their performance with three different output selection functions. The comparisons of the selection functions use uniform message traffic and

---

<sup>1</sup>An input selection function is used to choose among multiple messages simultaneously requesting the same output channel. Because of the focus of this paper, references to selection functions should be understood to mean output selection functions.

minimal routing. The authors suggest that reducing the number of turns that a message takes may reduce blocking and hence improve performance and *zigzag* does poorly because it also tends to concentrate traffic in the center of the mesh. These simulations are done for partially adaptive routing algorithms, so over half the source-destination pairs have no adaptiveness. This suggests that more dramatic results may be possible with fully adaptive routing.

Duato [7] presents simulation results for a time-dependent selection function. The idea is to avoid virtual channel multiplexing in the network unless the network traffic load is relatively high. This is achieved by preventing a message from using certain virtual channels unless the time a message waits exceeds a threshold value.

Rao [11] studies the imbalance in the channel utilization among different channels in the mesh. Rao proposes assigning different weights to channels based on their relative expected utilizations. Thus, channels that are expected to be heavily utilized are waited on less frequently.

Although the techniques proposed by Duato and by Rao improve performance, there are drawbacks and limitations with both ideas. Rao's approach is based on adjusting the use of waiting channels to balance expected utilization, but the relative utilization of channels can vary with the traffic pattern. Duato's approach requires additional virtual channels or allows less adaptiveness. The complexity of the router and the network cycle time increase with the number of virtual channels [2]. Furthermore, both approaches require a counter for each virtual channel. These counters add to the complexity of the router and if placed on the critical path will increase the network cycle time.

In contrast, the selection functions suggested in this paper are simple to implement and add no overhead. In addition, a time-dependent selection function or adjusting the choice of waiting channel to even the utilization could be combined with our result, if desired.

Upadhyay, *et al.* [15] compare their adaptive routing algorithm with several others and claim better performance, despite being less adaptive, because their routing algorithm makes more balanced use of the virtual channels. Similarly, we show that changing the selection function can change the balance of traffic in the network and improve performance, even without changing the routing algorithm.

Our simulations use a fully adaptive routing algorithm, called *opt-y*, proposed by Schwiebert and Jayasimha [13, 14]. For  $n$ -dimensional meshes, *opt-y* has the fewest restrictions of any fully adaptive minimal routing algorithm for meshes, so *opt-y* provides the maximum number of virtual channels for each message. By using *opt-y* as the routing algorithm in our simulations, we maximize the flexibility of the selection function.

### 3 Simulation Methodology

In order to model the interconnection network, a modular simulator [9] was written in C++ using the CSIM17 [12] simulation package. The simulator is structured so that modules, such as the routing algorithm, message traffic, or multiplexing scheme, can be easily changed without any modifications required to the remaining components. This allows the testing of different selection functions by slightly changing only the routing algorithm.

For accuracy, a flit-level simulation using demand multiplexing is performed. To prevent starvation, the FCFS input selection function is used. We record both the average message latency and channel utilization. A  $16 \times 16$  mesh is used for all simulations, with message sizes of 16 flits or 128 flits. Both uniform and transpose traffic patterns are simulated. With uniform traffic, a node sends a message to any other node with equal probability. With transpose traffic, a node at  $(x, y)$  sends a message to the node at  $(k - y - 1, k - x - 1)$ , where  $k$  is the arity of the mesh. For these simulations,  $k = 16$ , because a  $16 \times 16$  mesh is used.

The simulation consists of three phases: start-up, steady-state, and termination. The start-up phase is used to ensure the network is in steady-state before beginning to measure the message latency. When the number of active messages is nearly constant over a sufficiently long period of time, the network is judged to be in steady-state. On the other hand, if the number of active messages consistently increases, the network is saturated and the simulation is halted. Once the network is judged to be in steady-state, the simulation continues while additional messages are generated. The average message latency and channel utilization are recorded during this time period. Enough messages are generated to achieve a 95% confidence interval of well under  $\pm 1\%$ . After generating this many messages, the simulation enters the termination phase. During the termination phase, messages continue to be generated, so that the network traffic remains at the same rate. Messages generated during the termination phase are *not* included in the results. The termination phase continues until *all* the messages generated during the steady-state phase are delivered.

### 4 Simulation Results

The *opt-y* routing algorithm is used for all the simulations. For a 2D mesh, *opt-y* uses one virtual channel in the  $x$  dimension and two virtual channels in the  $y$  dimension. We refer to the first virtual channel in the  $y$  dimension as Y1 and the second as Y2. Fully adaptive routing is supported by *opt-y*; the only restriction on routing is that messages that need to route further West from their current position cannot use Y1.

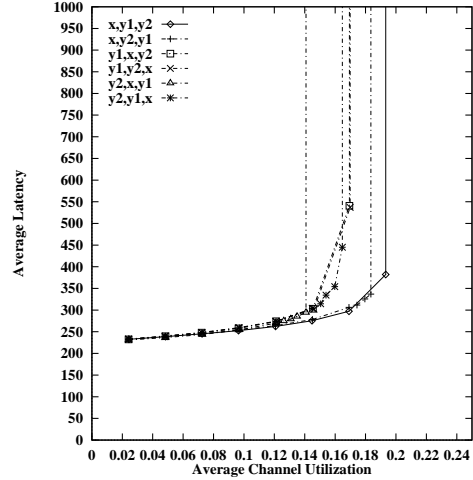


Figure 1: 16-flit Messages with Uniform Traffic

Eight different selection functions are simulated. The selection functions affect the choice of output channel only when *multiple* output channels are free. When no output channel is free, the simulator randomly picks one of the waiting channels and the message must wait for that channel to become free. The waiting channels are chosen to prevent deadlock and avoid concentrating messages on a single path. Thus, when a message needs to route in two dimensions, it has a waiting channel in each direction: Y1 for East and Y2 for West. A message that needs to route in only one dimension can wait for a channel in that direction.

Since the selection function controls the routing only when multiple output channels are free, the choice of selection function may seem relatively unimportant. As the simulation results show, however, the choice has a dramatic effect on both message latency and saturation behavior.

The decision was made to choose selection functions that do not require counters and other sophisticated mechanisms. Six different selection functions are simulated; the difference among the selection functions is the priority placed on the virtual channels. For example, when multiple output channels are available for a message, the  $x, y1, y2$  selection function uses the X channel if possible, followed by Y1, and uses Y2 only when necessary.

Figure 1 shows the average message latency and point of saturation for these six selection functions when using 16-flit messages and uniform traffic. Figure 2 depicts the results with 128-flit messages and uniform traffic. The x-axis shows the average channel utilization and the y-axis is the average message latency experienced by the messages. In both cases, the  $x, y1, y2$  selection function shows the best average message latency with higher utilization rates and sustains the highest traffic load before saturation. One reason that the  $x, y1, y2$  selection function might perform best is that as long as a message use only the X channel,

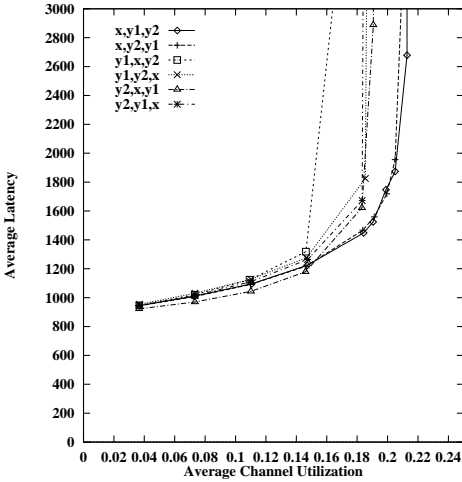


Figure 2: 128-flit Messages with Uniform Traffic

that message receives the entire bandwidth of the physical channel. A channel in the Y-dimension may be multiplexing two messages, so each message receives only half the channel bandwidth. Since there is only one channel in the X-dimension, a message that needs to route only in the X-dimension may be blocked longer than a message that needs to route only in the Y-dimension. Better performance can be expected when Y1 is preferred over Y2, because this encourages East-bound messages to use a channel that West-bound messages cannot use.

As figures 1 and 2 show, the selection function has a dramatic effect on the average message latency and saturation behavior. The performance differs because the message traffic is spread differently in the network. A straightforward analysis suggests that with uniform traffic the channel utilization should form a bell shape with the highest utilization at the center. As the traffic concentrates in the center of the mesh, however, adaptive routing tends to avoid this region of heavy congestion and to spread the traffic more evenly. The analytical result for a  $10 \times 10$  mesh, shown in figure 3, is reproduced from Rao's thesis [11] and hence the values on the axes do not correspond directly to the units used in this paper. The z-axis represents the expected channel utilization and the x-axis and y-axis show the relative position of each channel in the network.

The analytical model suggests that zigzag will perform poorly, because it moves traffic toward the center. On the other hand, intentionally routing away from the center could actually improve performance. We simulate the performance of zigzag and a selection function, called *outside*, that attempts to route messages away from the center of the mesh. Note that the *outside* selection function still uses only minimal routing. Essentially, outside chooses the direction that keeps the message as far from the center as possible.

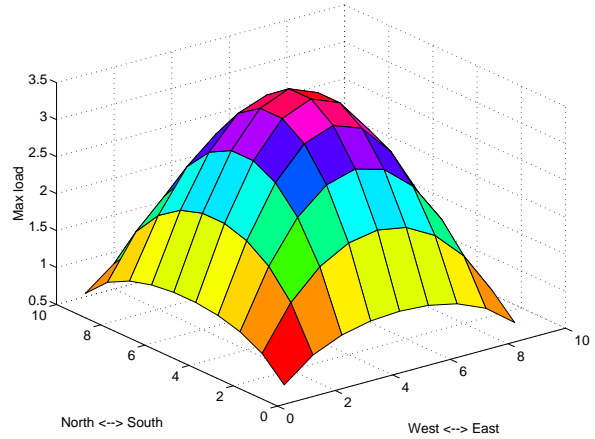


Figure 3: Expected Channel Util. with Uniform Traffic

Although zigzag and outside are more complicated and require additional resources, the simulations are done as if the overhead for all selection functions is equal. In practice, some additional router complexity and resulting increase in router cycle time are to be expected with the zigzag and outside selection functions, so our comparison gives a relative advantage to these two selection functions.

In figures 4 and 5, these two selection functions are compared with  $x, y1, y2$ , which has the best performance of the original six selection functions. For both message sizes,  $x, y1, y2$  provides a clear advantage both in terms of average message latency and saturation behavior.

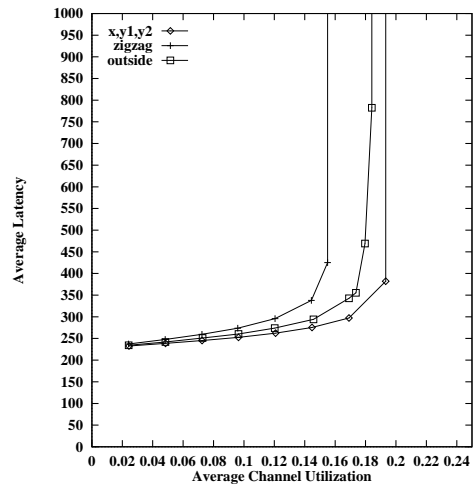


Figure 4: 16-flit Messages with Uniform Traffic

Figures 6 – 8 present the channel utilization characteristics for these three selection functions. An examination of the 3D graphs explains the superior performance of the  $x, y1, y2$  selection function. These graphs depict the case where the *average* channel utilization was approximately

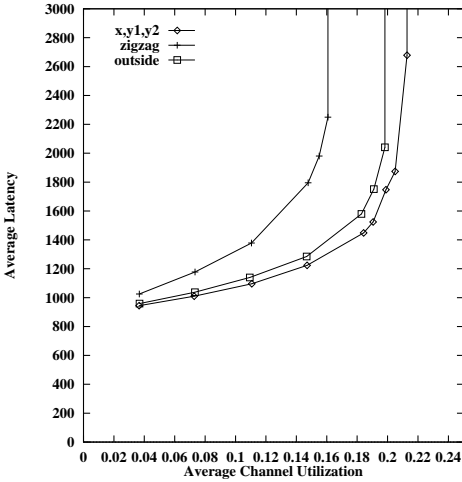


Figure 5: 128-flit Messages with Uniform Traffic

12% and the message length was 16 flits, so the network is not nearing saturation. The channel utilization for the South-bound channels is shown, because the results are similar for all four directions.

The  $x, y1, y2$  selection function does an acceptable job of balancing the traffic. With uniform traffic, the traffic crossing the middle of the mesh is higher than the traffic on the edges of the mesh, but the traffic is not concentrated in the center. The zigzag selection function concentrates traffic in the center, and hence saturates relatively quickly. The outside selection function is overly successful in avoiding the center of the mesh. In fact, the network actually saturates at the edges of the mesh.

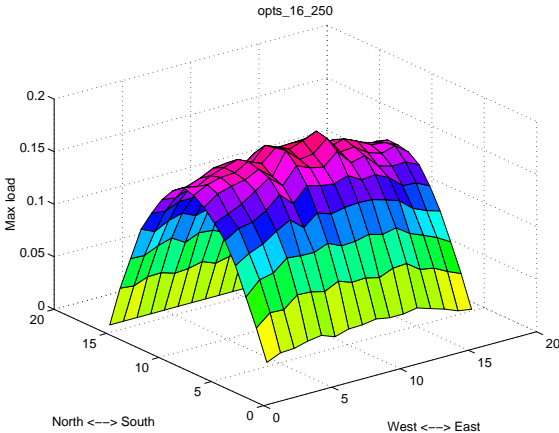


Figure 6: Channel Util. for  $x,y1,y2$  with Uniform Traffic

To provide a more complete comparison transpose traffic is also simulated. The simulations are unchanged except for the traffic pattern. The simulation results with the original simple selection functions are not shown, but

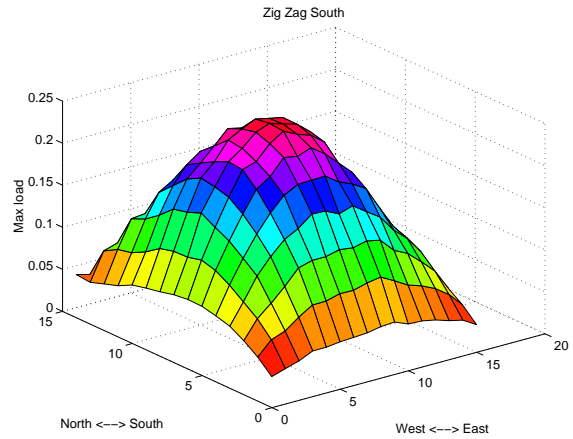


Figure 7: Channel Util. for Zigzag with Uniform Traffic

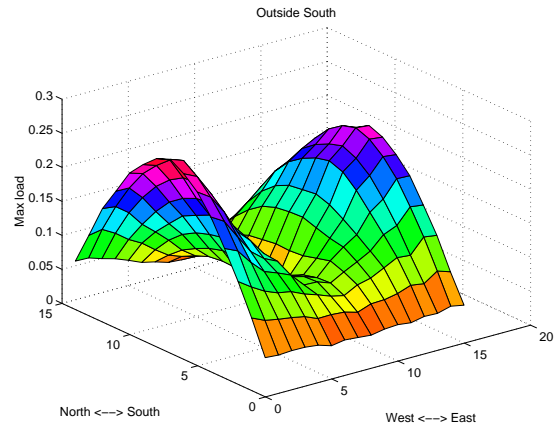


Figure 8: Channel Util. for Outside with Uniform Traffic

$x, y1, y2$  is again the best. Figures 9 and 10 depict the simulation results for  $x, y1, y2$ , outside, and zigzag. The results again show that the  $x, y1, y2$  selection function has the best saturation behavior. Although zigzag shows better performance than before,  $x, y1, y2$  is superior. Initially, zigzag has better performance because of its path-preserving property. Eventually, however, the network saturates due to the congestion in the center of the mesh. The 3D channel utilization graphs for transpose traffic are similar to the results for uniform traffic.

## 5 Conclusion

We have presented a detailed simulation of various selection functions with an adaptive routing algorithm. The simulations clearly support our hypothesis that the choice of selection function can have a substantial impact on the average message latency and saturation behavior. The 3D graphs facilitate a clear understanding of the reasons that

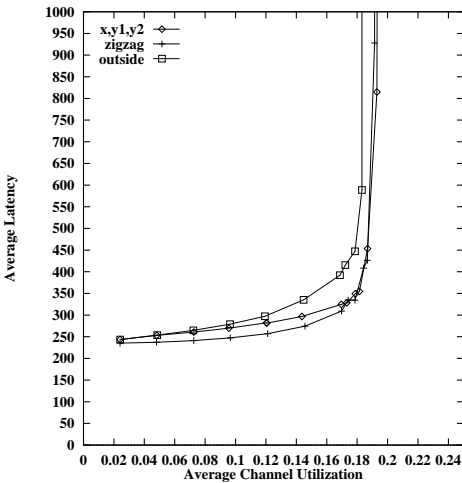


Figure 9: 16-flit Messages with Transpose Traffic

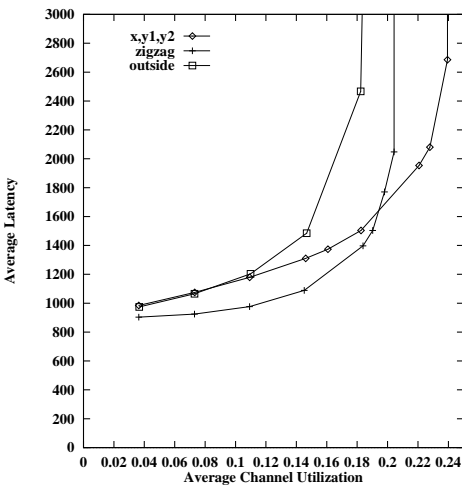


Figure 10: 128-flit Messages with Transpose Traffic

some selection functions perform better than others.

The results also show that the theoretically optimal selection function, *zigzag*, does not perform well in practice. This discrepancy between theory and practice exists because the underlying assumption for the design of *zigzag* is that the utilization of all channels is equal. Because traffic tends to concentrate in the center of a mesh, this assumption does not hold in practice and the benefit of *zigzag*, namely preserving routing flexibility, is out-weighted by the disadvantage of moving messages toward the center of the mesh.

We plan to extend these results to include additional message traffic patterns. We may also evaluate a few more carefully chosen selection functions. One of our objectives is to consider selection functions that can be implemented without substantial complexity. This would also be a requirement for any additional selection functions.

## References

- [1] S. Badr and P. Podar. An Optimal Shortest-Path Routing Policy for Network Computers with Regular Mesh-Connected Topologies. *IEEE Transactions on Computers*, 38(10):1362–1371, October 1989.
- [2] A. A. Chien. A Cost and Speed Model for k-ary n-cube Wormhole Routers. In *Proceedings of the Hot Interconnects Workshop*, August 1993.
- [3] A. A. Chien and J. H. Kim. Planar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors. In *19<sup>th</sup> Annual International Symposium on Computer Architecture*, pages 268–277, 1992.
- [4] W. J. Dally. Virtual-Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.
- [5] 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.
- [6] J. Duato. On the Design of Deadlock-Free Adaptive Routing Algorithms for Multicomputers: Design Methodologies. In *Parallel Architectures and Languages Europe 91*, volume I, pages 390–405, 1991.
- [7] J. Duato. Improving the Efficiency of Virtual Channels with Time-Dependent Selection Functions. In *Parallel Architectures and Languages Europe 92*, pages 635–650, 1992.
- [8] C. Glass and L. M. Ni. The Turn Model for Adaptive Routing. *Journal of the Association for Computing Machinery*, 41(5):874–902, September 1994.
- [9] D. N. Jayasimha, J. May, K. Patel, R. Rao, and L. Schwiebert. *A Simulation Environment for Multiprocessor Architectures*, 1995. Unpublished Manuscript.
- [10] L. M. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, 26(2):62–76, February 1993.
- [11] R. Rao. Utilization Imbalance in Wormhole Routed Networks. Master’s thesis, Ohio State University, 1994.
- [12] H. Schwetman. CSIM: A C-Based, Process-Oriented Simulation Language. In *Proceedings of the 1986 Winter Simulation Conference*, pages 387–396, 1986.
- [13] L. Schwiebert and D. N. Jayasimha. Optimal Fully Adaptive Wormhole Routing for Meshes. In *Supercomputing ’93*, pages 782–791, 1993.
- [14] L. Schwiebert and D. N. Jayasimha. Optimal Fully Adaptive Minimal Wormhole Routing for Meshes. *Journal of Parallel and Distributed Computing*, 27(1):56–70, May 1995.
- [15] J. Upadhyay, V. Varavithya, and P. Mohapatra. A Traffic-Balanced Adaptive Wormhole-Routing Scheme for Two-Dimensional Meshes. *IEEE Transactions on Computers*, 46(2):190–197, February 1997.
- [16] J. Wu. An Optimal Routing Policy for Mesh-Connected Topologies. In *International Conference on Parallel Processing*, volume I, pages 267–270, 1996.