Analysis of Buffer Design for Adaptive Routing in Direct Networks [*]

Annette Lagman
College of Integrated Science and Technology
James Madison University
Harrisonburg, VA 22807
lagmanag@cs.jmu.edu

Walid A. Najjar
Department of Computer Science
Colorado State University
Ft. Collins, CO 80523
najjar@cs.colostate.edu

Abstract

The performance of a massively parallel computing system is often limited by the speed of its interconnection network. One strategy that has been proposed for improving network efficiency is the use of adaptive routing, in which network state information can be used in determining message paths. The design of an adaptive routing system involves several parameters, and in order to build high speed scalable computing systems, it is important to understand the costs and performance benefits of these parameters. In this paper, we investigate the effect of buffer design on communication latency. Four message storage models and their related route selection algorithms are analyzed. A comparison of their performance is presented, and the features of buffer design which are found to significantly impact network efficiency are discussed.

1 Introduction

Recent research has focused on the use of adaptive routing algorithms in interconnection networks for massively parallel machines. In adaptive routing, messages are not restricted to a single path when traveling from source to destination. Moreover, the choice of path can be made in response to current network conditions. Such schemes are more flexible, can minimize unnecessary waiting, and can provide fault-tolerance. This suggests that adaptive routing has the potential, at least in theory, to provide lower communication latency, but at the cost of increased complexity. Most commercial systems currently available (e.g. Intel Paragon and Cray T3D) use deterministic strategies, such as dimension-ordered routing, in order to limit system cost and complexity.

[*]This work is supported in part by NSF Grant MIP-9113268

Various strategies for preventing deadlock have been proposed, these include the Turn model [12], variations on the use of virtual channels [9, 22, 4, 11, 25, 24], and others involving restrictions on the choice of paths or involving the use of additional hardware [5, 10, 18].In most of these studies, however, the emphasis is on algorithm design and performance analysis of the algorithms is secondary. Some studies have addressed the effects of other design parameters on adaptive routing performance, including [3, 16, 4]. More extensive performance studies on routing strategy have been done for deterministic schemes, in papers such as [6, 7, 2, 15, 8, 29, 1]. Several of these studies have presented analytic models of deterministic routing schemes as well as simulation experiments.

The work presented here is a performance evaluation study of one of the design parameters for a network routing system, namely the the choice of buffer structure used to store messages while they wait at a node. Such buffers are necessary because several messages arriving at a node may compete for the same system resources. Depending on the flow control technique used, some messages may be made to wait while others receive service. This delay is one of the major components of communication latency, and thus the efficiency of the storage mechanism can have a significant impact on network performance. This paper presents the results of extensive simulation experiments on four buffer systems for adaptive routing in direct networks, comparing their performance to each other and to that of dimension ordered deterministic routing.

2 Network and Message Model

The interconnection network model used in this study is that of a k-ary n-cube. Virtual cut-through switching [14] is used: message advancement is similar to wormhole routing, except that the body of a message can continue to progress even while the message head is blocked, and the entire message can be buffered at a single node. For simplicity all messages are assumed to have the same length, and source and destination nodes are uniformly distributed. Communication channels are unidirectional, and thus every node will have n input channels and n output channels connecting it to the rest of the network. In addition, since nodes can generate messages and absorb messages, each node will also have two “channels” designated as source and sink.

A message is composed of one or more *flits*, a single flit occupies one unit of storage in a node, and it is assumed that one flit can travel along a communication channel in one time unit (cycle). The first or *header* flit of a message is distinguished, and contains all the routing information for that message. All the flits of a message proceed along the same route as the header in train-like fashion.

The routing information consists of a record containing n fields, corresponding to each of the n dimensions in the network. At any time, each field in this routing tag will contain the number of hops the message still needs to take along that dimension in order to reach its destination. Note that although there may be several paths from the current location of a message to its destination node, the entries in the routing tag will always be unique. At every hop taken by the header flit, the value in the appropriate field of its routing tag is decremented. In dimension ordered routing, the routing tag fields are decremented in a fixed order. In adaptive routing, any order is allowed.

3 Selection and Buffering Strategies

The impact of buffer design on communication latency has been investigated previously in studies such as [26, 13] and [27], but in the context of multistage interconnection networks. One study that has addressed the impact of the storage mechanism on performance of direct networks is [17], which describes the Chaos router.

Closely associated with the choice of buffer structure is the selection policies used to manage waiting messages and to determine the paths they take. The *input selection policy* is used to choose among messages competing for the same output channel. Additionally, in adaptive routing the *output selection policy* assigns an output channel to a message when it has a choice of paths to take.

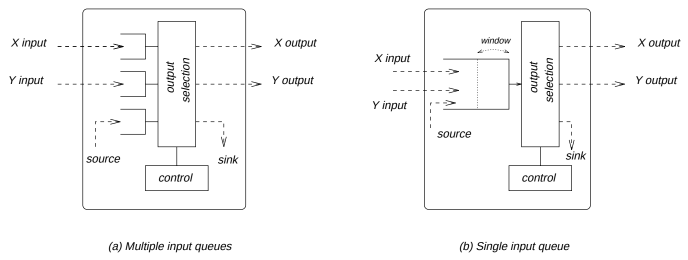


Figure 1: Input based buffer structures

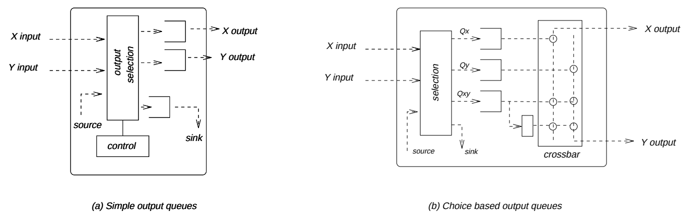


Figure 2: Output based buffer structures

One goal of this study is to determine the best possible performance gain that could be achieved by using the different structures, the selection policies are therefore designed for optimal performance. The basic strategy used by all systems presented here is that messages are dynamically mapped to output ports such that: (1) the maximum number of messages are routed out of a node at every cycle, and (2) each message stays on a shortest path from its source to destination. Routing the maximum number of messages minimizes unnecessary waiting, and can therefore improve network performance.

The buffering schemes are associated either with input queues or output queues. In input queuing, messages are stored before the output channel on which they will proceed is determined. In output queuing, messages are buffered after the the choice of output channel has been made. In both cases, freedom from deadlock is ensured by allowing the message buffers to be as large as needed, or in essence to be of infinite size. This allows us to determine optimal performance improvements, as well as to focus on the effects of the buffer design alone. Experiments on finite buffer structures in combination with deadlock prevention algorithms are described in [21].

3.1 Routing with input queues

Two simple buffer structures based on input queues are considered. Analytical models of these schemes were previously derived in [20] and [23]:

A *multiple input queue model:* In this model, a queue is associated with each input channel, to store messages that arrive on that channel (in a two dimensional network each node would have three input queues: one for each dimension and one for the node itself as a source). At each cycle, the flits at the head of the queues are mapped to the output channels in such a way that the maximum possible number of messages are allowed to proceed.

A *single input multiqueue model:* In this model all messages are buffered in a single, centralized queue as they arrive. At every cycle, the router looks at messages within a given "window" at the top of the queue, and maps the messages to the output channels choosing from among those in the window. This window can be as large as the entire queue.

The buffer structures are shown in Figure 1 for the case of a two-dimensional network ($n = 2$). Assignment of messages to output channels is based on the routing tag of the messages under consideration (either at the head of the multiple input queues, or within a window on the single queue). A *bipartite graph matching algorithm* [28] was used in the simulator to determine the mapping of messages to output channels which allows the maximum number of messages to proceed.

The optimal assignment may be unique or there may be several mappings which allow the same number of messages to proceed out of a node. The selection policies used should therefore include some criteria for choosing among these mappings. Possible input selection criteria may include giving priority to older messages or to messages which are close to their destination. Possible output selection criteria include biasing messages towards or away from dimension ordered paths or simply a random choice.

These two input buffer based models have the potential to provide message latencies that are lower than in dimension ordered routing. However, they suffer from two major drawbacks associated with input buffering: (1) Only messages at the head or within a window at a queue are considered at any time. Thus, if these messages are not routed, they may block some other message on the queue which would have been able to proceed. (2) The algorithm used to find an optimal mapping of messages to output channels is extremely complex in the general case. Graph matching is an expensive operation which cannot be easily implemented in a router.

3.2 Routing with output queues

Some of the deficiencies associated with input buffering can be overcome by using output queues. In this case, messages are first assigned to output channels, and then they are stored on queues associated with the chosen channel. The schemes under consideration are:

A *simple output queue* model, in which a queue is associated with each output channel. As messages arrive, they are assigned to some queue. The assignment can be made in a variety of ways, such as by dimension order as in deterministic routing, or at random. In line with the basic strategy of allowing the maximum number of messages to proceed at every cycle, in our experiments the assignment was done as follows: When a message arrives, it looks for an empty output queue. If none is found, an output queue is chosen at random.

(2) A *choice-based output queue* model, in which messages are buffered in queues such that each queue holds messages which can be routed on the same set of output channels. Thus, in a two dimensional network, one queue would be assigned to each dimension (an X-only and a Y-only queue) to hold those messages which can proceed on only one of these dimensions. A third queue (an XY queue) would hold the messages that have a choice of output. This is depicted in Figure 2. Note that enqueuing messages would be done by identifying the non-zero entries in a message routing tag. Routing is done first by dequeuing the X and Y queues. If either one of these queues is empty, then the first one or two message at the head of the XY queue are routed instead. An analytical model of this scheme is described in [19].

The use of output queues eliminates the situation associated with input queues in which the message at the head of a queue may unnecessarily block the messages behind it. However, when output queues are used, it may be possible that early assignment of messages to channels may result in an uneven distribution of messages among queues, and again messages may be made to wait unnecessarily. The selection scheme for simple output queues described above may suffer from this, and perhaps a better policy would have been to assign a message to the shortest queue at the time of arrival at a node. This operation, however, is expensive, and the algorithm described above was chosen as a reasonable compromise. It is worth noting that output queues provide optimal performance for deterministic routing. Since there is no choice of output channels, a message can be enqueued at the proper location without the need to worry about the early assignment problem.

There are several other drawbacks associated with output queuing. First, messages arriving at a node may do so simultaneously, and thus the routing hardware must be fast enough to handle this situation. And second, since several messages may be arriving at a queue (not just at a node) simultaneously, the management of physical storage is more complex.

Finally, one issue that should be mentioned is that of starvation. In both input queues and in the choice based output queue, a decision is made at some point. This policy may not always be fair, as in the case of random choice, and a starvation prevention mechanism is needed. The simplest technique is to assign priority to messages following some aging scheme. This has been implemented, and experiments have shown that it does not significantly affect performance.

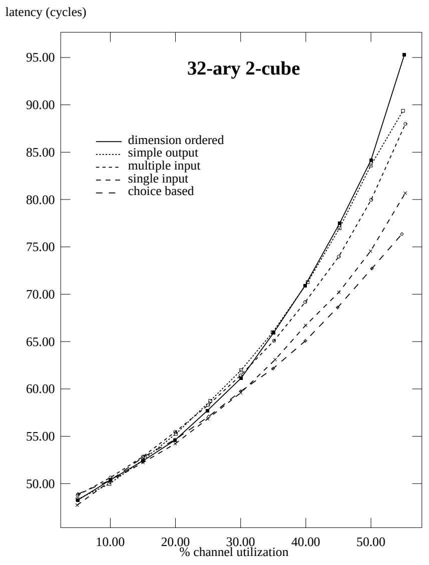
4 Performance Results

A discrete event simulator was developed in order to evaluate the performance of the various buffer models and selection policies. Figures 3 illustrates the results for two example networks. Latency is the average number of clock cycles between the time a message was generated and the time its tail flit arrived at its destination. Channel utilization is the mean percentage of communication channels that are busy at any cycle. In general, the order in which the buffer systems are listed in the legend at the top left of each graph corresponds to the order in which the curves appear (top to bottom).

In general, the data indicates that:

- Use of adaptive routing with any of the buffer systems resulted in some performance gain over dimension ordered routing (the exception was the use of simple output queues in the 32-ary torus, which exhibited the same average latency as dimension ordered routing). The performance improvement increases as the number of dimensions in the network increases, as well as with increased channel utilization levels. In a two dimensional network with low traffic ($< 30\%$), there was no visible improvement due to adaptive routing using any of the buffer structures. In the three dimensional torus, the performance improvement even at medium traffic levels was significant.

- In all cases, the best performance (lowest latency) was displayed by the *choice based* output queuing model. This model is characterized by two



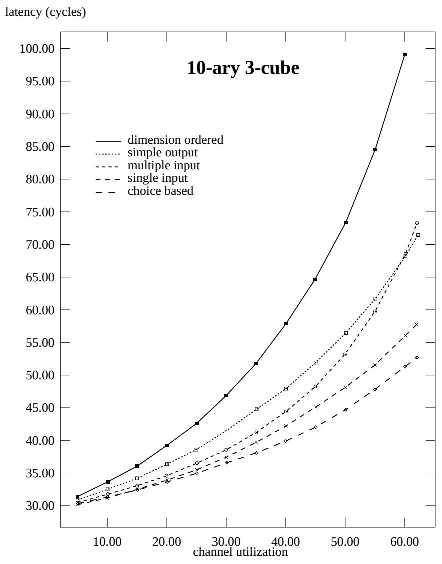


Figure 3: Buffer designs, 16 flit messages

things: (1) Messages stored in a queue can all use the same output channels and thus a message will never block another in the same queue. (2) In the two dimensional network, messages are routed out of the X-only or Y-only queues before considering the XY queue. This illustrates that a built-in priority is given to messages which have less choice of output channels on which to proceed. In addition, this model eliminates the need for a complex matching routine.

- The *single multiqueue* model with a routing window equal to the entire queue showed the second best performance. In this model, because the buffer structure is not a strict FIFO queue, messages at the head of the multiqueue cannot block those behind it. The difference between this and the choice based queues is that no priority is given to any message by the selection algorithm.

- The *multiple input queue* model showed some improvement over dimension ordered routing, but the presence of intra queue blocking in this case limited the performance gain. In two dimensions in particular, the improvement was minimal.

- The use of *simple output queues* showed good performance improvement in three dimensions. In two dimensions, there was no improvement at channel utilization below 50%. Moreover, other experimental results ([21]) indicate that there were great variations in performance when different selection policies were used with simple output queuing. In some cases, the latencies were even higher than in dimension ordered routing.

- In all adaptive schemes, the maximum buffer size measured is less than when the dimension ordered routing is used.

As indicated by the graphs, the choice based queue model can improve performance significantly by eliminating intra queue blocking, without the need for a complex matching algorithm. However, the number of queues needed for an n dimensional network is 2^n, and for dimensions greater than two or three this may not be acceptable. Variations on this concept are therefore investigated. One variation is presented here for the three dimensional networks. In this partially adaptive model, messages are first routed deterministically on the X dimension, and then adaptively using the full choice based queuing model on the remaining two dimensions. Thus, for a three dimensional network, instead of requiring eight queues, this system will only

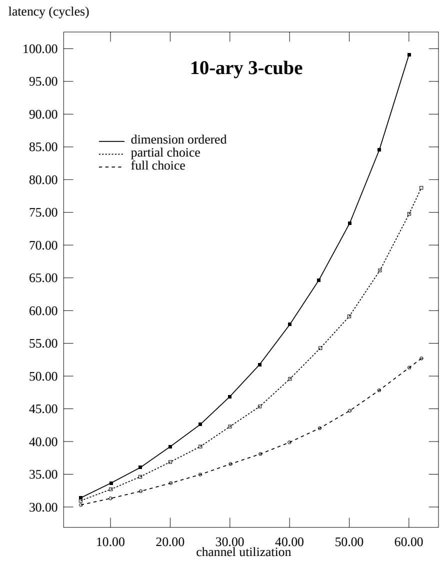


Figure 4: Partially vs fully adaptive routings, 16 flit messages

require five. However, by requiring that a message travel on one dimension initially, the choice of paths becomes more limited.

The latencies using the full and partial systems are shown in Figure 4. The graphs show that the partial system shows some improvement over dimension ordered routing, but that this was much less than that of the full system. In fact, the latencies for the partially adaptive system are comparable to those of the multiple input queue model. This suggests that limiting the choice of paths, and increasing contention for channels in the first dimension, has a significant effect on performance.

5 Conclusion

This paper discusses the effect of buffer system design on network performance by evaluating communication latencies using four message storage models. In general, the results indicate that a queuing structure which does not impose artificial blocking dependencies among messages will exhibit a significant improvement in network performance. This can be achieved using input based queues, but at the cost of more complex routing algorithms. It can also be achieved by using a large number of output based queues.

The results for the simple output queue model illustrate that designing a buffer system to meet this objective is not always simple. The wrong choice of mapping algorithm can result in a system which adds artificial dependencies, which can lead to performance which is worse even than deterministic routing. However, it should be noted that even in the presence of some dependencies, as in the case of the multiple input queue model, adaptive routing still performs better than dimension ordered routing.

The impact of the various selection policies used to arbitrate when there are several (optimal) choices for mapping messages to output channels criteria (random, priority based, etc.) does not appear to be significant for the simple network and message model used.

References

[1] Vikram S. Adve and Mary K. Vernon. Performance analysis of multiprocessor mesh interconnection networks. *IEEE Trans. on Parallel and Distributed Systems*, 5(3):225–246, March 1994.

[2] A. Agarwal. Limits on interconnection network performance. *IEEE Trans. on Parallel and Distributed Systems*, 2(4):398–412, October 1991.

[3] D. Badouel, C. A. Wuthrich, and E. L. Fiume. Routing strategies and message contention on low-dimesional interconnection networks. Technical report, Computer System Research Institute, University of Toronto, 1991.

[4] R. V. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. In *Int. Symp. on Computer Architecture*, pages 351–360, 1993.

[5] A. A. Chien and J. H. Kim. Planar-adaptive routing: low cost adaptive networks for multiprocessors. *Int. Symp. on Computer Architecture*, pages 268–277, May 1992.

[6] W. J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. on Computers*, 39(6):775–785, June 1990.

[7] W. J. Dally. Express cubes: improving the performance of k-ary n-cube interconnection networks. *IEEE Trans. on Computers*, 40(9):1016–1023, 1991.

[8] W. J. Dally. Virtual-channel flow control. *IEEE Trans. on Computers*, 3(2):194–205, March 1992.

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

[10] J. T. Draper and J. Ghosh. Multipath e-cube algorithms (MECA) for adaptive wormhole routing and broadcasting in k-ary n-cubes. In *Int. Parallel Processing Symp.*, pages 407–410, Beverly Hills, CA, 1992.

[11] J. Duato. A new theory of deadlock-free adaptive routing in wormhole networks. *IEEE Trans. on Parallel and Distributed Systems*, 4(12):1320–1331, December 1993.

[12] C. L. Glass and L. M. Ni. The turn model for adaptive routing. In *Int. Symp. on Computer Architecture*, pages 278–287, 1992.

[13] M. J. Karol, M. G. Hluchy, and S. P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Trans. on Computers*, COM-35(12):1347–1356, December 1987.

[14] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267 – 286, 1979.

[15] J. Kim and C. R. Das. Modeling wormhole routing in a hypercube. In *Int. Conf. on Distributed Computing Systems*, pages 386–393, May 1991.

[16] J. H. Kim and A. A. Chien. The impact of packetization in wormhole-routed networks. In *PARLE '93*, pages 242–253, 1993.

[17] S. Konstantinidou and L. Snyder. Chaos router: architecture and performance. *Int. Symp. on Computer Architecture*, pages 212–221, 1991.

[18] G. Denicolay L. Gravano, G. D. Pifarre and J. L. C. Sanz. Adaptive deadlock-free wormhole routing in hypercubes. In *Fifth Int'l. Parallel Processing Symp.*, pages 512–517, Beverly Hills, CA, 1992.

[19] A. Lagman and W. A. Najjar. On the design of optimal adaptive routers for direct networks. In *Proc. 1994 Scalable High Performance Computing Conference*, pages 642–649, 1994.

[20] A. Lagman, W. A. Najjar, S. Sur, and P. K. Srimani. Evaluation of idealized adaptive routing on k-ary n-cube. In *IEEE Symp. Parallel and Distributed Processing*, pages 166–169, 1993.

[21] Annette Lagman. *Modelling, Analysis and Evaluation of Adaptive Routing Strategies.* PhD thesis, Colorado State University, Computer Science Department, November 1994.

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

[23] W. A. Najjar, A. Lagman, S. Sur, and P. K. Srimani. Modeling adaptive routing in k-ary n-cube interconnection networks. In *MASCOTS'94*, pages 120–125, Durham, NC, 1994.

[24] G. D. Pifarre, L. Gravano, S. A. Felperin, and J. L. Sanz. Fully adaptive minimal deadlock-free packet routing in hypercubes, meshes, and other networks: algorithms and simulations. *IEEE Trans. on Parallel and Distributed Systems*, pages 247–263, 1994.

[25] C-C. Su and K. G. Shin. Adaptive deadlock-free routing in multicomputers using only one extra virtual channel. In *Int. Conf. on Parallel Processing*, 1993.

[26] Y. Tamir and G. L. Frazier. High performance multiqueue buffers for VLSI communication swithces. In *Int. Symp. on Computer Architecture*, 1988.

[27] Y. Tamir and G. L. Frazier. Hardware support for high priority traffic in VLSI communication switches. Technical Report CSD-900051, UCLA Computer Science Department, 1990.

[28] A. Tucker. *Applied Combinatorics*. John Wiley & Sons, second edition, 1984.

[29] N. Tzeng. Empirical evaluation of incomplete hypercube systems. In *Int. Conf. on Parallel Processing*, pages I–96 – I–99, 1993.