Department of

Computer Science

Analytical Models of Adaptive Routing Strategies

W. A. Najjar, A. Lagman, and S. Sur

and P. K. Srimani

Technical Report CS-93-130

August 10, 1994

Colorado State University

Analytical Models of Adaptive Routing Strategies

W. A. Najjar, A. Lagman, and S. Sur and P. K. Srimani

Department of Computer Science
Colorado State University
Ft. Collins, CO 80523 USA

This work is in part supported by NSF Grants CCR-9010240, CCR-9148225 and MIP-9113268

Modeling Adaptive Routing

W. A. Najjar, A. Lagman, S. Sur and P. K. Srimani

Abstract

Adaptive routing algorithms have been frequently suggested as a means of improving communication performance in multicomputers. These algorithms, unlike deterministic routing, can utilize network state information to exploit the presence of multiple paths. Adaptive routing, however, is complex and expensive. Before such schemes can be successfully incorporated in multiprocessor systems, it is necessary to have a clear understanding of the factors which affect their performance potential. In this paper we present a simple and efficient scheme to model the performance of idealized adaptive routing. We evaluate a basic, high-performance adaptive system using an analytic queueing model which approximates its behavior. This analytic model predicts the performance of networks with varying parameters, and provides insight into the nature of message traffic. We have also conducted extensive simulation experiments, the results of which are used to validate the analytic model and to identify some of the conditions which promote high performance of adaptive routing in a communication network.

Keywords: Multicomputers, Interconnection networks, Performance evaluation, Adaptive Routing, Modeling, k-ary n- cubes.

Address for Correspondence:

Walid A. Najjar
Department of Computer Science
Colorado State University
Ft. Collins, CO 80525
Tel: (303) 491-7026
Fax: (303) 491-2466
Email: najjar@CS.ColoState.Edu

1 Introduction

The success of large scale message-passing multicomputers is highly dependent on the efficiency of their underlying interconnection networks. In the past few years there have been tremendous improvements in the performance of multicomputer systems, but these have come primarily from advances in processor technology. Communication systems have not kept up, and the increasing disparity between processor and network speeds has become a major hindrance to further performance gains.

One approach that has often been suggested as a solution to the communication latency problem is the use of adaptive routing techniques. At present most commercial machines use deterministic routing, in which a message path is fully determined by the addresses of its source and destination. Thus, messages with the same source and destination would use exactly the same paths. While these schemes are simple and easily implemented, they are susceptible to unnecessary blocking of messages and do not respond to changing conditions within the network. Adaptive routing algorithms, on the other hand, allow message paths to be determined in response to variations in network state. This provides greater freedom in choosing message paths, which can potentially lead to reduced communication overhead.

There are, however, a number of problems associated with adaptive routing. First, the increased flexibility of adaptive algorithms make them more susceptible to deadlock than deterministic schemes. In order to avoid the circular resource dependencies which lead to deadlock, it is necessary to either restrict the choice of paths or to include extra hardware in the system. The second problem associated with adaptive routing is that it may be difficult to implement, and may either require expensive hardware or suffer from high node switch delay.

In order to accurately assess the tradeoff and benefits of adaptive routing, we must be able to evaluate the performance potential of these systems. Recent research studies have begun to address this issue, but as yet the problem has not been fully explored. In [GN92a], Glass and Ni compared partially adaptive deadlock free routing schemes to dimension ordered schemes in mesh networks. In [LDT93], Liu *et al* evaluated expected channel utilization for a class of deadlock-free algorithms, again in mesh networks. Kim and Chien [KC93] assessed the effects of packetization on adaptive routing performance, Nowatzyk [Now89] has investigated a communication architecture using an adaptive algorithm, and recently Boppana and Chalasani [BC93] presented a comparative study of several deadlock free adaptive routing schemes. Almost all of the performance evaluation studies of direct networks, however, have made use solely of simulation experiments. In contrast to studies of deterministic routing, very little analytic work has been done. Moreover, most studies have looked at adaptive routing in isolation, without considering the relationship of other network parameters to routing performance.

The study presented in this paper is a performance evaluation, both analytic and experimental, of a basic adaptive routing system on k-ary n-cube networks. Preliminary versions of this research were presented in [LNSS93, NLSS94]. The goals of the study are to evaluate the performance potential of adaptive routing and to understand the effects of various network parameters on routing performance. Thus, we have chosen to investigate an idealized system with a general routing strategy which can achieve near-optimal performance. The basic model is flexible, however, and can be modified for future research on specific network design parameters as well as on cost rather than performance. The details of the system are described in Sections 2 and 3.

We have derived a general analytic queueing model which approximates the behavior of adaptive routing. When messages are routed non-deterministically, they will usually have a choice of communication channels to use at every hop. Our analytic model is based on the probability distribution of the number of such choices. The general model can be applied to different adaptive routing systems by choosing the appropriate queueing paradigm and by calculating the appropriate arrival and departure probabilities. We present the analysis for two examples which make use of birth-death queueing. The analytic model is derived and discussed in Section 4. In addition to the theoretical analysis, extensive simulation experiments were performed. The results of these experiments were used to validate the analytic model, as well as to investigate the effects of various network parameters on adaptive routing performance. These results are presented and analyzed in Section 5. The paper ends with a brief summary and discussion of future work.

2 Background

There are three general parameters in the design of an interconnection network: topology, routing algorithm and flow control strategy. Topology refers to the underlying graph of the network which determines the connections between processing elements. The routing algorithm is the technique used to determine the path taken by a message from its source to its destination, while flow control refers to the system used to manage messages as they travel around the network. Some of the general issues associated with these parameters are discussed in [Fen81] and [Fea91].

2.1 Topology

The topology we have chosen to investigate is the k-ary n-cube [DS87]. This is a general family of graphs which has become increasingly popular among researchers and has been implemented in a number of existing machines [Dal89, Com85, Hay86, Res93]. A k-ary n-cube is a direct network with $N = k^n$ nodes; k is called the radix and n the dimension. A k-ary 1-cube is a ring, and a k-ary n-cube can be built be connecting k copies of k-ary $(n-1)$-cubes in a ring. There are many variations and special cases of this network. When $k = 2$, for instance, the network is a hypercube. In this case there are no wrap-around edges. Another popular variation is the mesh, which is an n-cube without wrap-around edges. The mesh topology is popular because it simplifies implementation, and the absence of wrap-around edges makes the deadlock problem less difficult. However, such a network is not symmetric and has longer average distance and diameter.

2.2 Flow Control

Flow control refers to the management of messages as they traverse the network. Issues addressed by the flow control strategy include the system used to acquire and relinquish communication channels, the procedure for arbitrating among messages that contend for the same resource, and the scheme used to manage messages that are not allowed access to a channel. Effective flow control techniques should allow messages to progress while managing system resources efficiently.

Older multicomputers and networks for distributed systems have controlled communication channels either using circuit switching or store-and-forward policies. In circuit switching, all the channels in a message path are reserved for that message and are relinquished only after the entire message has arrived at its destination. In store-and-forward, channels are only acquired as messages go from one node to the next, but the head of a message does not advance until after its tail has arrived at the same node.

Neither of these two techniques meet the high-speed demands of present day multicomputers. Instead, many now use a pipeline system introduced in [KK79] which combines features from both. There are two techniques that do this: wormhole and virtual cut-through. In these systems, messages are divided into flow control units or flits [DS87] with the head flit being distinguished. Only the first flit contains the routing information, and messages traverse the network in train-like fashion with all flits using the same channels as the header flit. Channels are reserved for the message from the time the header flit uses it and are released when the last flit has passed through the channel. Message latency times in this system are less sensitive to the distance traveled and at the same time channels are not held unnecessarily.

When several messages contend for the same channel, one message is allowed to proceed while the rest are made to wait. The first task of the flow control system is to determine which message is allowed to proceed, arbitrating on the basis of an input selection policy. Input selection can be random, use schemes such as round-robin, or use state information such as the distance of a message from its destination. See [GN92a] for a study which addresses input selection policy.

The flow control policy then determines how these waiting messages are managed. In wormhole flow control, the non-header flits of a message are not allowed to advance, and each flit is buffered on a different node. In virtual cut-through, the non-header flits are allowed to proceed until they reach the node occupied by the head flit, where some or all of the flits may be buffered if necessary. In a traffic free network, the message latency using either method would be the same. The advantage of virtual cut-through over wormhole is that channels can be released earlier, resulting in less blocking. However, this is done at the cost of additional buffer space needed on each node. The two methods also differ in the deadlock avoidance techniques that can be employed. In virtual cut-through, deadlock can be avoided by using ordered buffers, while in wormhole, buffer management as well as channel management is necessary in order to avoid deadlock. Most research on network performance assumes wormhole flow-control, an excellent survey of which can be found in [NM93].

The flow control mechanism must also include the decision on buffer placement. In general buffers can be associated with the input channels, the output channels, or they can be intermediate. Associating buffers with input channels means that messages are buffered before the routing algorithm is applied. Similarly, output channel buffers mean that the routing algorithm has been applied before the flits are stored. The flow control mechanism also determines the structure of the buffers. For instance, buffers may be queues processed in FCFS order, or they may be pools which use some other criteria to prioritize departure. Studies which have investigated this issue include [TF88] and [KHM87].

The flow control mechanism used in the study presented here is virtual cut-through, with buffers associated with input channels. We examine two variations, one in which a set of queues is used and messages are processed in FIFO order, and another in which all messages are buffered in a single modified queue. Since input queues are used, messages are buffered before the routing algorithm is applied. Such a system complements adaptive routing, because by delaying channel assignment the algorithm can make use of the most current network state information. The input selection policy used is described in Section 3.

2.3 Routing Algorithms

Routing algorithms for direct networks are generally classified as being either deterministic or adaptive. In deterministic routing, message paths are chosen based solely on the addresses of the source and destination nodes. In adaptive routing, a path can be determined by other factors, including the state of the network as the message travels. The performance of deterministic routing schemes have been widely studied, and unlike adaptive routing several analytic models of these systems have been developed [AV92, KD91, Dal90, CE91, Dal92]. In our comparisons of adaptive versus deterministic routing, we use the model presented and analyzed in [Aga91], in which infinite queues associated with output channels are used.

The most general adaptive routing schemes can make use of any path in the network, regardless of length, the presence of loops, or any other factors. Often a message will have a choice of channels on which to exit a node, and arbitrating among choices is made on the basis of an output selection policy. For instance, one scheme that has been investigated extensively is the use of random routing [Val82]. Other researchers such as [LH91, CK92, DG92] have investigated adaptive routing as a means of adding fault-tolerance. In this study, we will focus on adaptive routing which makes use only of minimal distance paths. This reduces the number of choices, but in a k-ary n-cube there are enough minimal distance paths available so as not to constrict the routing. Moreover, this eliminates the problem of livelock, in which messages travel around the network without progressing towards their destinations.

Deterministic routing is simple and easily implemented, with minimal overhead. It has been implemented in various machines such as the N-cube [Com85] and the Cray T3D [Res93]. Adaptive routing, on the other hand, allows for more flexibility at the cost of additional complexity in the algorithm and its implementation. The study presented here will focus on evaluating the potential improvement of adaptive routing over deterministic routing. Thus, we choose to compare the two in an idealized situation, where routing overhead is ignored.

2.4 Deadlock

Deadlock is another important consideration in designing a network system, and one that has been the subject of numerous studies. Various avoidance techniques have been proposed, such as in

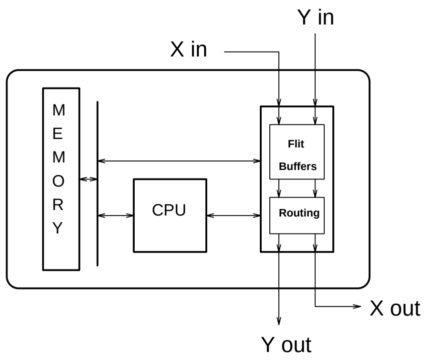

Figure 1: Structure of a Node

[LMN93, SS93, CK92, GN92b, SJ93, LH91, LGS92]. In general, deadlock avoidance is achieved either by restricting routing or by the addition of hardware such as virtual channels, as proposed in [DS87]. A recent study describing a new model of deadlock freedom can be found in [Dua93].

The issue of deadlock, however, is orthogonal to that of performance in the sense that, unlike the three parameters described previously, avoidance mechanisms are not usually designed with the goal of improving communication time. Thus, although deadlock avoidance schemes will indirectly affect performance, they have been excluded from the scope of the study presented here. Instead, as in [Aga91], deadlock is avoided in our model by the use of infinite buffer stores.

3 System Model

The communication network being modeled is composed of processing nodes which send messages over unidirectional communication channels. Each node in our k-ary n-cube network consists of a processing element and a communication unit which is responsible for receiving and transmitting messages from neighboring nodes. Each node is connected to n input channels and n output channels, can generate its own messages, and can be the final destination for messages originated at other nodes. Thus, each node can receive input from $(n+1)$ different logical sources and can transmit messages to $(n+1)$ different logical destinations. Figure 1 illustrates a node in a k-ary 2-cube.

Each message consists of one or more flits, with the head flit being distinguished. Routing decisions are made only for the head flit of a message, which must contain the routing information. The system makes use of virtual cut-through flow control, meaning once the head of a message is routed, the rest of the message must also advance and use the same sequence of channels. The routing model assumes that even when the header flit of a message is blocked at a node, the node

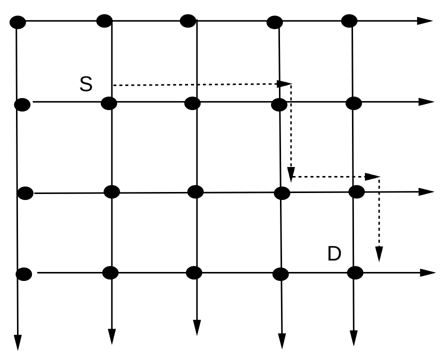

Figure 2: A Minimal Path from S to D

still continues to receive the subsequent flits of the message and stores them in that node's buffers.

In our model we assume that messages are generated at source nodes at a uniform rate, and all pairs of nodes have equal probability of being source/destination pairs.

3.1 Routing Strategy

When a message is first generated at a source node S, it is possible to determine not only its (minimal) distance from the destination node D but also the number of hops the message must make in *each dimension* on its way to D. Although there may be several minimal distance paths from S to D, the number of hops in each dimension is unique. This information can be assembled into a routing tag containing n fields, with each field holding the number of hops to be made in one of the dimensions. As an example, a message from node S to node D in the 2 dimensional network illustrated in Figure 2 will have a tag consisting of the values $< 3, 2 >$, where 3 is the number of hops in the first (horizontal) dimension and 2 is the number of hops in the other dimension. When the header flit of a message traverses a a channel, the value of the tag field corresponding to the channel's dimension is decremented, until at the destination all fields are zero. This tag is the only information needed to route a message adaptively along minimal paths, since a message can proceed on a channel in every dimension whose tag entry is non-zero.

In the basic routing strategy which we model, the communication unit looks at a set of header flits that are candidates for routing, and attempts to route as many as possible at every cycle. In a deterministic routing scheme there is only one possible output channel that can be taken by a message at each node, and if that channel is busy, then the message must stall. In order to eliminate this unnecessary delay, the solution proposed here is to allow a message to take *any free output channel that would route it on one of its shortest paths.* Thus, in a situation where two messages are waiting to be routed through a node with two output channels, if one message can go on both channels while the other can only proceed on one of the channels, then the channels will be assigned such that both messages will be allowed to proceed. The optimal mapping is determined by a matching routine. This routine first determines which output channels are free to be used by a new message. It then examines every message that is a candidate for routing, and uses the message tags to determine which output channels can possibly be used by which message. An assignment which maximizes the number of messages routed is then made based on this information. The secondary flits (whose headers have already moved) are routed on their predetermined channels.

The routing strategy always chooses a minimal path between the source and the destination and considers all possible minimal paths. It is *minimally fully adaptive.* Aside from eliminating unnecessary waits, this allows communication load to be shared more equally throughout the system, and contributes to the fault tolerance of the system. It uses only *current and local information*, which saves on time and space requirements: resources are not needlessly idled, and message path selection is done in a distributed manner. More importantly, because it has been designed to meet the objectives of minimal waiting time and maximal throughput, the strategy has a very good potential for high performance.

Optimizing the mapping of messages to channels at a node is analogous to the bipartite graph matching problem [Tuc84] by means of the following association: Represent each message and each output channel by a vertex in the graph, with an edge between two vertices whenever the associated message can proceed along the corresponding output channel (*i.e.*, the route tag field for that channel is non-zero). The bipartite matching problem is a well-known combinatorial problem, and various algorithms exist to solve it. The implementation we have chosen in our simulator is based on the Ford-Fulkerson maximum flow algorithm, adapted to the case of an unweighted bipartite matching [Tuc84]. The complexity of this algorithm is $O(|V|^{\frac{1}{2}} \cdot |E|)$, where $|V|$ = number of vertices and $|E|$ = number of edges [PS82]. If n is the dimension of the network, then $V \leq 2(n+1)$ and $E \leq (n+1)^2$. While this may be seen as a limitation of the routing schemes, in practice, the general matching algorithm need not be used. In particular, for very low dimension networks, the matching can be performed by a simple search.

3.2 Buffering Strategy

In our adaptive routing system we associate buffers with input channels, meaning that messages are assigned to storage locations before the routing algorithm is applied. In addition, we assume

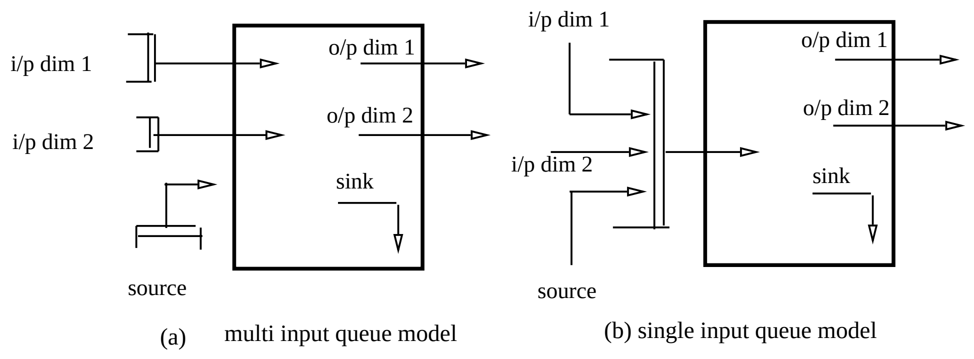

Figure 3: Node Buffer Structures

that these buffers can hold infinitely many flits. This assumption is made in order to simplify the analytical models developed in Section 4. The results presented in Section 5.2 on measures of maximum queue length for both dimension ordered and adaptive routing strategies indicate, however, that adaptive routing requires much smaller size buffers.
We consider two buffering schemes:

- In the *multiple queue* scheme a separate input queue is used for each of the $n+1$ input channels (shown in Figure 3-a). Only the head of each input queue is considered for routing at every cycle. If several messages can proceed on a channel, priority is given to older messages.

- In the *single queue* scheme, all incoming flits are are buffered into one shared modified queue as shown in Figure 3-b. At every cycle, the flits in a given "window" at the the top of the queue are candidates for routing. If several messages can proceed on a channel, an arbitrary choice is made.

The performance of these two buffering schemes are compared to each other and to a dimension ordered model which uses output buffering.

4 Analytic Models

In this section, we present mathematical models that approximate the behavior of communication systems using each of the two buffering schemes. The main component of the analysis is the derivation of the probability distribution of the number of output channel choices for messages in the network. In an adaptive routing system, the number of path possibilities influences channel contention among messages, and thus their probability distribution can be used to model the arrivals and departures of messages at a node. Moreover, in our adaptive routing system the message queues at one node can be decoupled from those in neighboring nodes, and thus it is possible to model the entire network by looking at the behavior of queues locally rather than by analyzing networks of queues. This general scheme can be used to describe adaptive routing systems with different design parameters by using the appropriate queueing paradigm. We apply the model here to systems with the two buffering schemes given previously, using the birth-death model of queue behavior.

Many simplifying assumptions were made in deriving the models, and thus they are only approximations. However, as was verified by comparison to simulation data, the errors resulting from the simplifications were often small and the analytic models can be used to predict the general pattern of behavior of the systems.

The analysis we present here applies to the k-ary 2-cube, also known as a torus or wrap-around mesh, with unidirectional channels. It was limited to the 2-cube primarily because this network is one of the most frequently encountered in the literature and in practice. In addition, the computational complexity of analyzing higher dimension cubes quickly becomes intractable.

Let G be a k-ary n-cube network, with N nodes and $E = nN$ (unidirectional) edges. We use the following notation throughout the paper:

ℓ :	Length of a message in flits.
m :	Uniform message generation rate at any node.
c :	Average channel utilization = average number of busy channels in any clock cycle.
Δ :	Average distance between any two nodes in the network.

For a k-ary n-cube with unidirectional channels where a node can be both source and destination of the same message, Δ is given by

$$\Delta = n(k-1)/2$$

If the source and destination or a message must always be distinct, which is assumed in our model, then the expression is:

$$\Delta = n\frac{(k-1)}{2}\frac{k^n}{(k^n-1)} \approx n\frac{(k-1)}{2}$$

The relation between the channel utilization and other network parameters, as in [Aga91], is given by:

$$c = \frac{m\Delta\ell}{n}$$

In order to analytically derive the expected latency of a message by modeling the behavior of the queues at each node, we first determine the probabilities governing the arrival and routing of messages at the queues, and then use these probabilities to predict their expected lengths.

4.1 Message State Probability Distribution

The rate at which a waiting message gets routed out of a node is influenced by the number of channels it can proceed on, which we designate as the message state. In a 2 dimensional network, any message at a node can be in one of three possible states: (1) it can traverse further in both the X or the Y dimension, (2) it can traverse in exactly one dimension (either X or Y), or (3) it cannot traverse any further (this node is the sink). As noted earlier, a message can proceed on a channel in every dimension whose routing tag entry is non-zero, and thus the state of a message can be determined by counting the number of fields in the tag containing zeroes.

Let σ_i be the probability that a message flit has i zero fields in its routing tag. Thus, σ_0 denotes the average probability, at any node, that the message can traverse along either of the two output dimensions, σ_1 denotes the probability that it can go in only one dimension, and σ_2 denotes the probability that the message is destined for the sink at that node. It is to be noted that in practice at any given time at any given node the input flits in the waiting queue will have different probabilities of routing restriction, but we characterize them with the same average probabilities for ease of computation in the model. Because source-destination pairs are randomly distributed, the parameters σ_0, σ_1 and σ_2 depend only on the network topology and can be computed as follows.

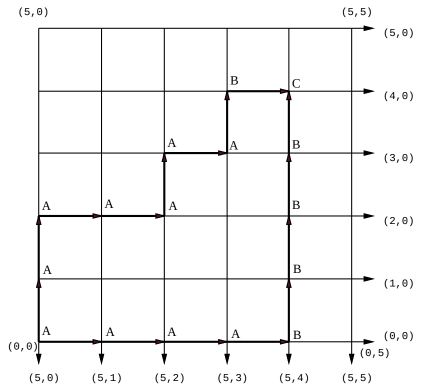

Figure 4: Computation of σ_0, σ_1, σ_2 in a 6-ary 2-cube

We fix an arbitrary source node s. Since the network is symmetric and regular, all nodes are equivalent and hence σ_i computed with respect to one node would be the same for all others. By considering all possible destination nodes from s and all possible paths to these destinations, and by examining the various states of a message as it traverses one of these paths, we can determine the probability distribution of σ_i.

Consider a destination node d, which is i hops away in one dimension and j hops away in the other dimension. The distance between s and d is $i + j$, and it can be shown that there are $(i + j)!/(i!)(j!)$ different minimal paths between s and d. For any such path, we designate as type A those nodes on the path from which the destination can be reached by continuing in either of the two directions. Type B nodes are those from which a message must exit using one particular dimension, and the destination node is a type C node. Figure 4 illustrates the type A, B and C nodes for two different paths between the source node (0,0) and the destination node (4,4).

The contribution to σ_0 of a path is the number of type A nodes in the path. Similarly, contributions to σ_1 and σ_2 can be obtained from the number of type B and C nodes in the path. For example, in Figure 4 the lower path (through (0,4)) contributes 4 towards σ_0, 4 towards σ_1 and 1 towards σ_2.

However, since a path is built sequentially as a message traverses the network, all possible paths between a source and destination are not equally probable. We use the example of Figure 4 to explain how the probability of taking a particular path is computed. Consider $(0, 0)$ as the source and $(4, 4)$ as the destination node. Then the lower path (through (0,4)) has the probability of 1/16 of being taken, since the probability that the message reaches node $(0, 1)$ is 1/2, from which the probability that the message reaches $(0, 2)$ is 1/2 of that, etc. Once the message reaches $(0, 4)$, it would reach the destination with probability 1. Hence the probability of taking this path is $(1/2)^4 = 1/16$. Similarly the other path in Figure 4 has probability $(1/2)^6 = 1/64$ of being taken.

The contributions of each path (taking account the probability that the path is used) for each destination are accumulated, and the results are then averaged. Table I shows the values of σ_i for different size k-ary 2-cubes using this algorithm. Note that these probabilities give an indication of how much freedom messages have to choose a route, which is lost when deterministic schemes are used.

4.2 Single Queue Model

In this section we describe an analytic model of the behavior of the single queue buffering scheme. First, we derive expressions which describe the arrivals and departures of flits on the queue in terms of message generation rate, channel utilization, and the message state probabilities given in

k-ary 2-cubes	σ_0	σ_1	σ_2
10-ary 2-cube	0.5016	0.3993	0.0991
20-ary 2-cube	0.5780	0.3721	0.0499
32-ary 2-cube	0.6095	0.3593	0.0312

Table I: σ_0, σ_1 and σ_2 for different k-ary 2-cubes

the previous section. We use the birth-death stochastic queueing model to approximate the single queue behavior, and apply the flit arrival and departure probabilities to derive the expected queue length and message waiting time. This enables us to determine the expected message latency for this buffering scheme.

Arrival Probabilities At any clock cycle the number of flits that can arrive at a node of a k-ary 2-cube can vary between 0 to 3. Let α_i denote the probability that i flits arrive at a node at one clock cycle, $0 \leq i \leq 3$. The probability that a flit arrives at a node via one of the input channels is equal to the channel utilization, c, and the probability that a message is generated at the node is equal to the message generation rate, m.

Lemma 1 *The arrival probabilities* α_i, $\quad 0 \leq i \leq 3$ *are given by,*

$$\alpha_0 = (1-c)^2(1-m)$$

$$\alpha_1 = m(1-c)^2 + 2c(1-m)(1-c)$$

$$\alpha_2 = 2mc(1-c) + c^2(1-m)$$

$$\alpha_3 = c^2 m$$

Proof : Zero arrival occurs when there is no arrival through either of the input channels and no message is generated at the node. The probability of no arrival via the input channels is $(1-c)^2$ and the probability of no message generation at the node is $(1-m)$. Hence $\alpha_0 = (1-c)^2(1-m)$. The expressions for α_1, α_2 and α_3 can be similarly derived. $\Box$

Departure Probabilities At any clock cycle in our model at most 3 flits can leave a node Also, if the input queue is not empty, there is at least one departure from the node. Let $\delta_{i,j}$ denote the probability of i $(1 \leq i \leq 3)$ departures in a clock cycle when there are j $(j \geq 1)$ flits in the queue.

Clearly, $\delta_{1,1} = 1$, $\delta_{2,1} = 0$ and $\delta_{3,1} = 0$. When the input queue size is 2, there cannot be more than 2 departures and hence $\delta_{3,2} = 0$. There can be only one departure if both the input flits contend for the same channel. The probability of both flits contending for the same output dimension is $(\sigma_1/2)^2$, while the probability of both flits contending for the sink is σ_2^2. Thus, the probability of one departure is $2(\sigma_1/2)^2 + \sigma_2^2 = \delta_{1,2}$. In all other cases both flits can be routed, which implies $\delta_{2,2} = 1 - 2(\sigma_1/2)^2 - \sigma_2^2$. The probabilities of departures for queue size greater than 2 are given by the following lemma.

Lemma 2 *When the input queue size is bigger than 2 (*i.e. $i \geq 3$*), the departure probabilities are given by:*

$$\delta_{1,i} = 2\left(\frac{\sigma_1}{2}\right)^i + \sigma_2^i$$

$$\delta_{2,i} = (\sigma_0 + \sigma_1)^i + 2\left(\frac{\sigma_1}{2} + \sigma_2\right)^i - \left(2\left(\frac{\sigma_1}{2}\right)^i + \sigma_2^i\right)$$

$$\delta_{3,i} = 1 - (\sigma_0 + \sigma_1)^i - 2\left(\frac{\sigma_1}{2} + \sigma_2\right)^i$$

Proof : The proof of the expression for $\delta_{1,i}$ follows from the same line of argument given for queue size 2. For exactly 2 departures out of a queue size i ($i \geq 3$), each of the input flits must go in at most 2 of the 3 outputs. The probability that an input flit may go in any of the 2 output dimensions is $\sigma_0 + \sigma_1$. Hence the probability that all the i inputs may go in the same two output dimensions is $(\sigma_0 + \sigma_1)^i$. Similarly, all the inputs may also contend for an output channel and the sink. This probability is given by $2(\frac{\sigma_1}{2} + \sigma_2)^i$, since the event of choosing 1 output dimension and the sink can happen in 2 ways. This total probability also includes the probability that all the i flits contend for the same output, and thus $\delta_{1,i}$ must be subtracted, yielding the expression for $\delta_{2,i}$. The expression for $\delta_{3,i}$ follows from the fact that $\delta_{1,i} + \delta_{2,i} + \delta_{3,i} = 1$. □

Expected Waiting Time and Message Latency We use the birth-death queueing model as a first order approximation of the single queue, ignoring transitions to states other than the neighboring ones. In Figure 5, S_i ($i \geq 0$) represents the system state when the input queue length is i. In order to use such a model, we have also assumed that our queues have the the Markov property, that changes in state depend only on the present state, and that the birth-death chain is aperiodic and recurrent non-null. The forward transitions λ_i and the backward transitions μ_i denote the probability of increasing or decreasing the waiting queue size by unity. The waiting

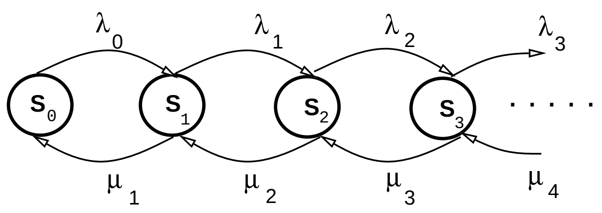

Figure 5: Birth and Death Queuing Model

queue size can increase by 1 if the number of departures is one less than that of the arrivals, and can decrease by one if the number of arrivals is one less than that of the departures. Thus,

$$\lambda_i = \alpha_2\delta_{1,i+2} + \alpha_3\delta_{2,i+3}$$

$$\mu_i = \alpha_0\delta_{1,i} + \alpha_1\delta_{2,i+1} + \alpha_2\delta_{3,i+2}.$$

We solve this birth-death system in steady state [Kle75] to determine the individual state probabilities. If p_i denotes the probability of the system being in state i (the waiting queue size is i), then

$$p_0 = \left(1 + \sum_{k=1}^{\infty}\prod_{i=0}^{k-1}\frac{\lambda_i}{\mu_{i+1}}\right)^{-1}$$

and

$$p_{i+1} = \frac{\lambda_i}{\mu_{i+1}}p_i.$$

Although the chain has an infinite number of states, with high probability only finitely many such states will in reality be attained. *Thus, we can approximate the infinite chain by a finite chain with an arbitrarily large number of states, and the infinite summation above can be computed to the point when the remaining terms add no significant value to the sum.*

Once the individual state probabilities are known, the average waiting time of a single flit can be computed in the following way. Let w_i denote the expected waiting time of a flit when the queue size is i. Clearly, $w_0 = 0$, and $w_1 = 1$, since when waiting queue size is 1 the flit can always be routed in the next clock cycle. When there are two flits in the queue, each of them waits for the next clock cycle when one flit is routed with probability $\delta_{1,2}$ and both of them are routed with probability $\delta_{2,2}$. Also, if one flit is routed the queue size decreases to 1 and the remaining flit would then require w_1 time to get routed. Hence, $w_2 = 1 + \delta_{1,2}w_1$. By extending this argument ($i > 2$), we get

$$w_i = 1 + \delta_{1,i} w_{i-1} + \delta_{2,i} w_{i-2} + \delta_{3,i} w_{i-3}.$$

Given the values of w_i, the average waiting time W of a flit before it reaches its destination is given by

$$W = \sum_{i=1}^{\infty} p_i w_i. \tag{1}$$

And finally, the average latency of a message of length ℓ flits is computed as:

$$T = (1 + \ell W)\Delta + \ell. \tag{2}$$

4.3 Multiple Queue Model

There are two types of queues at each node in this model: the queue that a message enters when it is generated at the source node and the queues which hold messages arriving from the network. The two queues differ in the rate at which messages enter the queue, as well as in the choices of output channels a waiting message can be routed on. A message on a source queue can only be routed out of the node, whereas a message on a non-source queue has the potential to be routed or to be absorbed by that node. In order to evaluate total message latency we must consider that each message will incur a delay in a queue of type s (source) once and delay in queues of type n (network) Δ times, corresponding to the Δ hops it must take.

The message state probabilities derived earlier must also be modified. Let $\sigma_s(i)$ be the probability that a message on a source queue has i zeroes in its routing tag. A message on a source queue can be in either of two states ($i = 0$ or $i = 1$) depending on the destination of the generated message. In particular, when $i = 1$ then the destination is either on the same row or on the same column as the source, viewing the network as a mesh. Since there are $k - 1$ nodes on a row or on a column, and $k^2 - 1$ possible destination nodes from a specified source, therefore

$$\sigma_s(1) = \frac{2(k-1)}{k^2-1} = \frac{2}{k+1}$$

and

$$\sigma_s(0) = 1 - \sigma_s(1) = \frac{k-1}{k+1}.$$

A message on a non-source queue, on the other hand, can be in one of three states. The probability distribution in this case can be computed by an algorithm very similar to that described in Section 4.1, except that the state of a message at the source node does not contribute to the computation.

Arrival and Departure Probabilities We first consider the case when $\ell = 1$. When multiple queues are used, at every clock cycle at most one message is allowed to enter or to depart from a queue. Thus, in this case the arrival rate at a network queue is simply c and the arrival rate at a source queue is m.

The departure rate in this model is equivalent to the probability of routing a message at the head of a queue. Unlike in the previous model, however, this probability is affected by factors external to the queue, namely contention from other queues. The probability of a message being routed out of a given queue is therefore dependent on the absence or presence of messages in the other queues.

When messages are only 1 flit long, all output channels at a node will be free to carry a message at every clock cycle. Thus, contention only occurs when several messages vie for the same output channel. Let R_n denote the probability of routing the head flit of a message which lies on a network queue. R_n depends on the states of the other queues on the node, and there are several several cases which must be considered. Let $C_n(j)$ denote the probability of being in Case j, and let $R_n(j)$ denote the probability of routing when in Case j.

When messages consist of only a single flit, the channel utilization c and the message generation rate m closely approximate the probability that a queue is busy. These values have therefore been used to compute $C(j)$. The probabilities $R(j)$ on the other hand are computed by (1) enumerating all possible combinations of states for the messages in the given case, (2) determining the probability of routing given each combination, (3) weighing this by the probability that such a combination of states occurs, and (4) summing all the corresponding values.

Case 1: All other queues are empty. In this case the message can always be routed, $C_n(1) = (1-c)(1-m)$, and $R_n(1) = 1$.

Case 2: One other network queue is busy and the source queue is empty. Contention occurs in this case when both messages are in state $i = 1$ and need the same output channel, or both are in state $i = 2$ (have reached the destination).
$C_n(2) = c(1-m)$, and $R_n(2) = \sigma_n(0) \ + \ \sigma_n(1) \ \ [\ 1 \ - \ \frac{1}{4}\sigma_n(1)\ ] \ + \ \sigma_n(2) \ \ [\ 1 \ - \ \frac{1}{2}\sigma_n(2)\ ]$.

Case 3: One network queue is empty and the source queue is busy. As in the previous case, contention occurs iff both messages contend for the same output channel. $C_n(3) = (1-c)m$
$R_n(3) = \sigma_n(0) \ + \ \sigma_n(1) \ \ [\ (\sigma_s(0) \ + \ \frac{3}{4}\sigma_s(1)\ ] \ + \ \sigma_n(2)$.

Case 4: All other queues are busy. $C_n(4) = cm$, and $R_n(4) = \sigma_n(0)\ [\ \frac{2}{3}\sigma_n(0)\sigma_s(0) + \frac{3}{4}\sigma_n(0)\sigma_s(1) + \frac{5}{4}\sigma_n(1)\sigma_s(0) + \frac{7}{5}\sigma_n(1)\sigma_s(1) + \sigma_n(2)\ ] + \sigma_n(1)\ [\ \frac{3}{5}\sigma_n(1)\sigma_s(0) + \frac{5}{9}\sigma_n(1)\sigma_s(1) + \sigma_n(2)\sigma_s(0) + \frac{3}{4}\sigma_n(2)\sigma_s(1)] + \sigma_n(2)\ [\ 1 - \frac{1}{2}\sigma_n(2)\ ]$.

The corresponding probabilities for a source queue are determined as follows:

Case 1: All other queues are empty. $C_s(1) = (1-c)(1-c)$, and $R_s(1) = 1$.

Case 2: One other network queue is busy and the source queue is empty. $C_s(2) = 0$.

Case 3: One network queue is empty and the source queue is busy. $C_s(3) = c(1-c)$, and $R_s(3) = 2(\sigma_s(0) + \sigma_s(1)\ [\ 1 - \frac{1}{4}\sigma_n(1)\ ])$.

Case 4: All queues are busy. $C_s(4) = c^2$, and $R_s(4) = \sigma_s(0)\ [\ \frac{2}{3}\sigma_n(0)^2 + \frac{3}{2}\sigma_n(0)\sigma_n(1) + 2\sigma_n(0)\sigma_n(2) + \frac{4}{5}\sigma_n(1)^2 + 2\sigma_n(1)\sigma_n(2) + \sigma_n(2)^2\ ] + \sigma_s(1)\ [\ \frac{1}{2}\sigma_n(0)^2 + \frac{6}{5}\sigma_n(0)\sigma_n(1) + 2\sigma_n(0)\sigma_n(2) + \frac{5}{9}\sigma_n(1)^2 + \frac{3}{2}\sigma_n(1)\sigma_n(2) + \sigma_n(2)^2\ ]$.

Finally, we see that

$$R_n = \sum_{j=1}^{4} R_n(j)C_n(j) \quad \text{and} \quad R_s = \sum_{j=1}^{4} R_s(j)C_s(j).$$

Expected Waiting Time and Message Latency Using the the probabilities R_n and R_s, the arrivals and departures from the channel queues can be modeled as a Markov chain as in Section 4.2. For a queue of type n, the transition probabilities λ and μ are given by:

$\lambda_0 = c$, the channel utilization rate

$\lambda_j,\ \ j \geq 1 =$ the probability of 0 departure and 1 arrival $= (1 - R_n)c$.

$\mu =$ the probability of 1 departure and 0 arrivals $= (1-c)R_n$.

Similar expressions can be derived for a queue of type s.

The state probabilities p_i are computed as in Section 4.1. The expected queue length E (for either a source or a network queue) is determined from the equation

$$E = \sum_{j=0}^{j=\infty} jp_j,$$

where the p_j for the appropriate queue type is used, and Little's result is used to determine the waiting times

$$W_n = \frac{E_n}{c} \quad \text{and} \quad W_s = \frac{E_s}{m}.$$

Finally, given the waiting times above and the average message distance Δ, the average message latency is computed as

$$L = \Delta W_n + W_s.$$

Multi Flit Messages When the message length ℓ is greater than one flit, the model must be modified in several ways:

1. Additional cases must be considered, to take into account the situations when a channel is busy because a previously routed message has control of that channel.
2. The values c and m can no longer be used to approximate the probability that a network queue or a source queue will be busy, except when channel utilization is very low. Instead, values of the Markov chain state probabilities were used. The analytic computations were run iteratively on increasing values of the channel utilization, using a very small step size. The values of p_i in one iteration were then used to approximate the channel state probabilities (busy or non-busy) at the succeeding iteration.
3. The transition probabilities in the Markov chain were modified to account for the new cases, when queues are non-empty but the first flit in a queue is not the head of a message.
4. Message latency was computed as $L = (\Delta + \ell - 1)W_n + W_s$, to include the time it takes to absorb the non-head flits of a message.

It should also be pointed out that when modeling multi flit messages, the flit arrival probabilities are no longer strictly independent, and thus the analytic model must again be viewed only as an approximation.

5 Experimental Results

A discrete event simulator was developed in order to verify the mathematical analysis and to further understand and evaluate the performance of the adaptive routing system. The parameters of a simulation run include: the network dimensions k and n, the message length, the number of "warm-up" cycles simulated, and the message generation rate. At every clock cycle, the simulator does the following: (1) generate new messages at random nodes with random destinations, (2) route and forward existing messages, and (3) absorb messages which have arrived at their destination. The simulator can be configured to use either of the adaptive buffering schemes, as well as dimension ordered routing. All measurements were taken only after the system had reached steady state, as determined by the stabilization of the channel utilization. Moreover, in order to minimize sampling error, the latencies were generated by averaging the results of multiple runs.

5.1 Model Validation

The average latencies obtained from the simulation and analytic results are plotted in Figures 6 and 7 for both buffering schemes with 1 and 8 flit messages respectively. Both graphs display average latency, which is the average number of clock cycles it takes from the time a message is generated to the time its last flit is absorbed, over various levels of channel utilization. Channel utilization is taken as the percentage of network channels that are busy at every cycle. Channel utilization is a function of message length, average distance, and message generation rate, and as such allows us to arrive at a fair comparison between varying sets of parameters.

For the shorter messages, the graphs show a very close match between the values predicted by the model and those obtained from simulation. For longer messages, the analytic model tends to underestimate latencies slightly. Computation of percentage errors for the simulation and analytic models show that at channel utilizations up to 60%, the error in latency is well below 8%. Thus, while the analytic model does not describe the system exactly, it gives a very good approximation with little computational effort.

Aside from measuring latency, the simulator also tabulated the various states of the messages going through the system. This was done in order to verify the routing probabilities σ_i which were derived as part of the analytic model. The data generated by simulation show that the distribution of the message states is very close to that predicted analytically. Moreover, the state probability distributions were minimally affected by channel utilization, and remained almost constant throughout the life of the experiments. The simulation experiments also generated data on queue length distribution, channel utilization, and waiting times. Again, the simulation results closely matched those predicted analytically, validating the assumptions made in developing the models.

5.2 Performance Results

In this section we present and discuss the result of the performance evaluation of adaptive and deterministic routing schemes. The objective of this evaluation is not to demonstrate the superiority of one scheme over another, rather to assess the impact of the various parameters affecting their

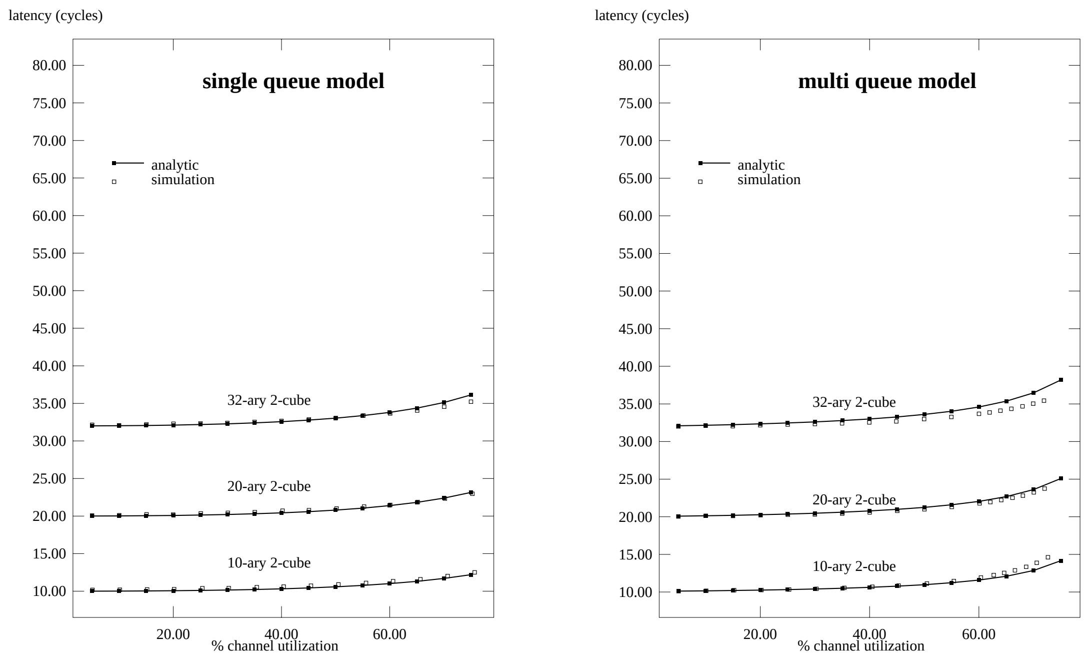

Figure 6: Comparison Between Model and Simulation for 1 Flit Messages

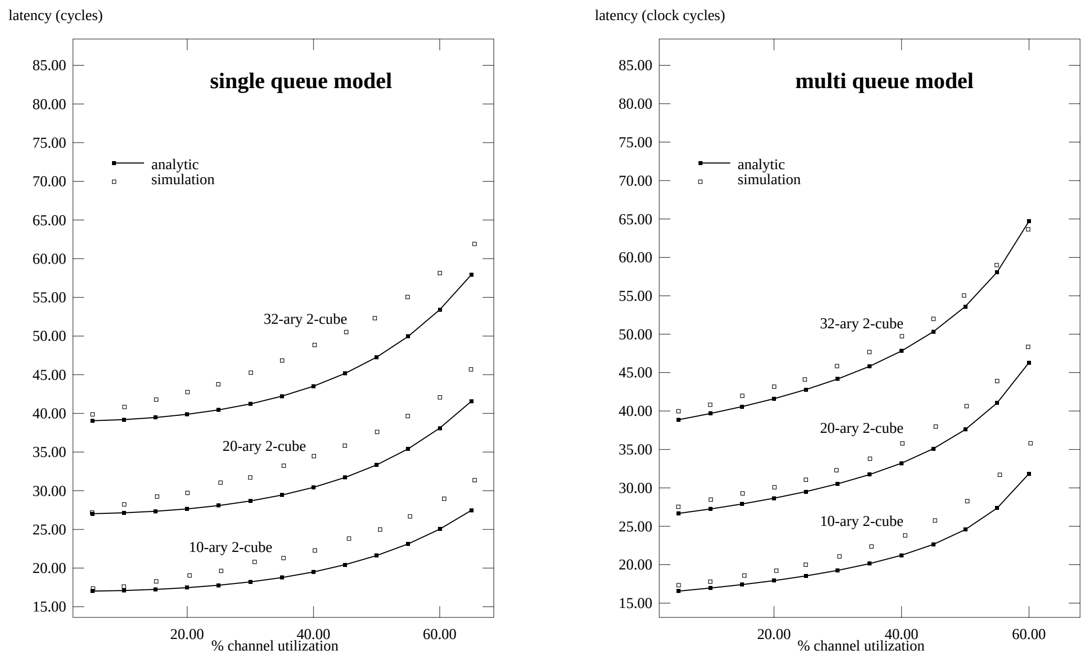

Figure 7: Comparison Between Model and Simulation for 8 Flit Messages

performance in order to gain a better understanding of the advantages and drawbacks of each scheme.

Latency Figures 8 and 9 compare the average latencies using deterministic routing to those of the two adaptive routing schemes. In all cases, deterministic routing exhibited the worst performance, and the single queue buffer model had the best performance. The graphs show a slight improvement in latency for two dimensional networks, and a significant improvement in three dimensions. The trend suggests that the performance gains will be even more significant with higher dimension networks. Such networks, however, pose serious realizability problems.
The experimental data reveals the following:

- In the two dimensional networks, at low channel utilization dimension ordered routing performed as well as adaptive routing. This suggests that in this case, when messages are blocked adaptive routing cannot take advantage of the multiple paths. This will occur when either messages have only one choice of channels or when there are three messages competing for only two output channels. In higher dimensions networks messages will in general have more choices of channels on which to proceed, for a same average distance, and similar situations are less likely to occur. Thus, in the three dimensional networks, there is a distinct difference in average latency between adaptive and dimension-ordered routing even at low channel utilization levels.

- One of the implicit assumptions in using deterministic routing is that if messages are suitably distributed in a network and channel utilization is not excessive, then the amount of traffic contention would be minimal, and messages would not need alternative routes. The graphs for deterministic routing show, however, that even at low channel utilizations the message latencies are greater than the minimum required, and that latencies increase rapidly. This indicates that even when there is little traffic on the network and messages are uniformly distributed, contention and blocking still occur.

- As channel utilization increases, so does channel contention, and at 50% or 60% channel utilization the difference in average latency between adaptive and deterministic routing is quite large in both two and three dimensional networks.

- As message length increases, so too does waiting time at a queue. As a consequence, performance improvement of adaptive over dimension-ordered routing also increases with message length.

Effect of Buffer Organization The two buffering strategies for adaptive routing both used queues that were associated with input channels. This delays the assignment of messages to output channels, and allows the algorithm to use the most current state information in optimizing the assignments. The drawback of this is that in the multi queue model, messages at the head of a queue may block other messages which arrived on the same input channel, but which may have a free output channel on which to proceed. This situation does not occur in the single queue model when the window size is large, and thus this model had lower average latencies. This is similar to the use of virtual channels for performance improvement as proposed in [Dal92], where the virtual channels are used as passing lanes to overtake blocked messages. Another approach would be to devise a buffer organization scheme which incorporates the best features of input and output channel buffering. Note, however, that in the three dimensional networks, the two models had very close latencies at low channel utilization levels, and it was only at about 40% channel utilization that this intra-queue blocking becomes a factor.

Maximum Queue Length The maximum queue length was also measured for each simulation run. Table II compares lengths for the multi queue and dimension ordered routing models. The data indicates that maximum queue length is greater in dimension ordered routing than in adaptive routing. At higher channel utilizations the ratio of maximum queue length is often a factor of two.

Note that the deterministic model makes use of output queues (since output channels are statically known when the message arrives). Output queueing strategies are known to have shorter queue sizes [KHM87]. This result suggests that an adaptive routing scheme with an output queue strategy would have even smaller maximum queue sizes and that adaptive routing is better at balancing communication load among channels.

The data generated from simulation also indicates that for both buffering schemes the most frequent queue size encountered was either 0 or 1 flit, even at high c or with long messages. This suggests that the choice of virtual cut-through over wormhole routing may not necessarily require much larger buffer space, specially if virtual channels for deadlock avoidance are already in use.

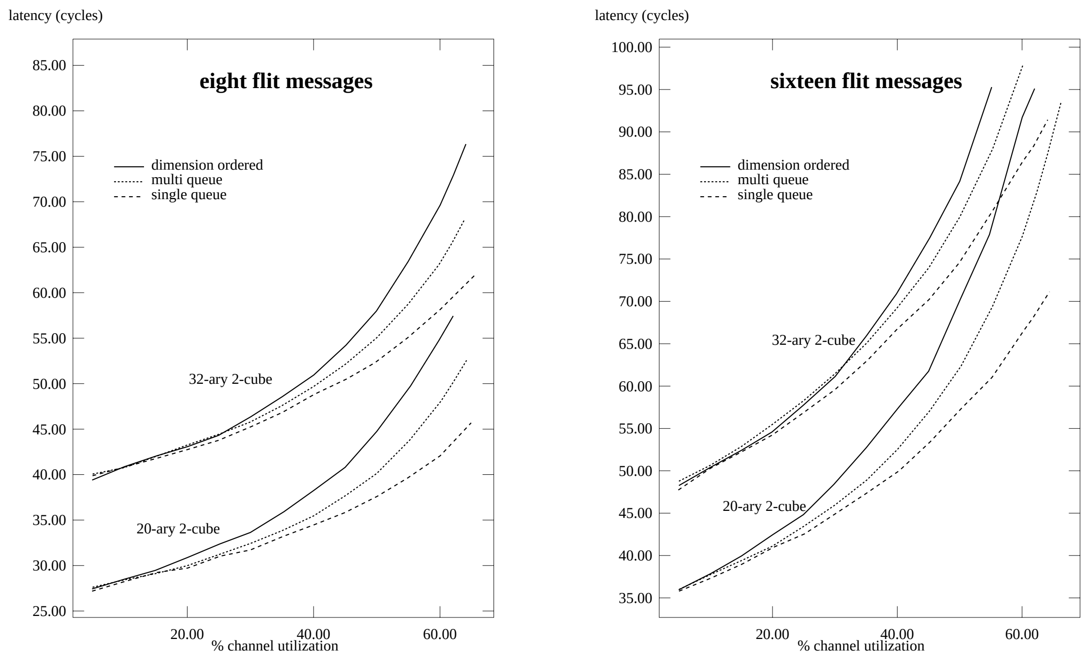

Figure 8: Average Latency, 2 Dimensional Networks

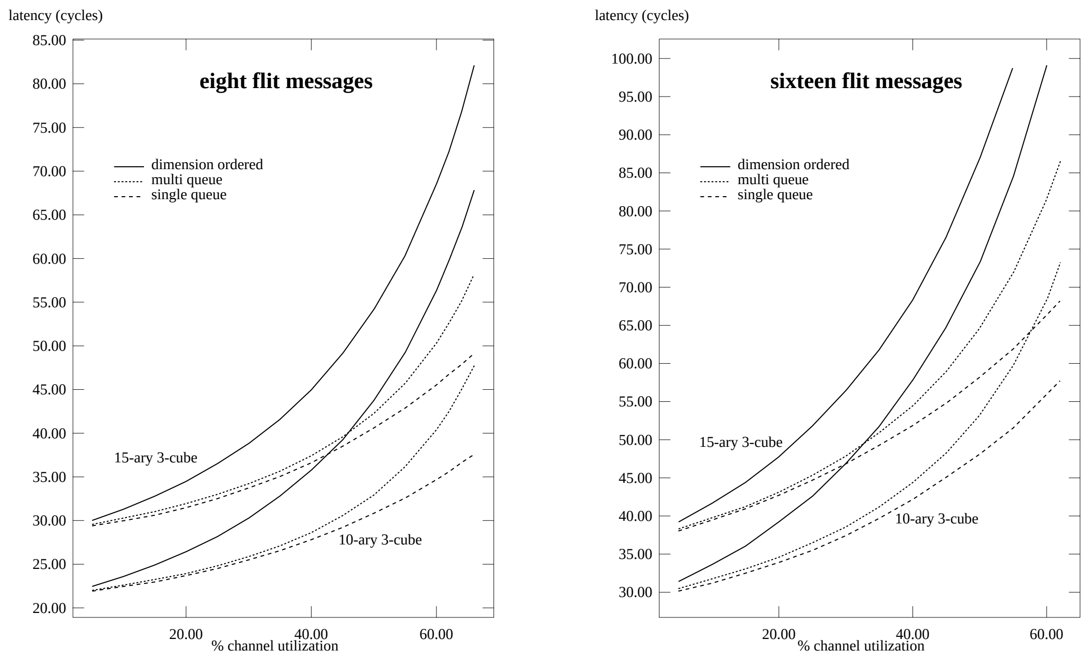

Figure 9: Average Latency, 3 Dimensional Networks

	adaptive multi queue			dimension ordered		
$c =$	30%	50%	70%	30%	50%	70%
$k = 20, n = 2$	16	19	30	28	41	73
$k = 32, n = 2$	16	20	32	25	33	65
$k = 10, n = 3$	17	25	43	40	63	106
$k = 15, n = 3$	16	24	35	39	49	89

Table II: Maximum Queue Length (in flits), Message Length = 8

6 Conclusion

In this paper we have analyzed the performance of a basic idealized adaptive routing scheme. We developed analytic models to describe the system's behavior and verified the models against simulation data. The analytic models make use of message state probabilities, and can be modified to apply to other system designs. Our routing algorithm decouples queues in one node from those in its neighboring nodes in the sense that messages leaving a queue in one node are distributed to the neighboring nodes non-deterministically. This enables us to model queue behavior locally, which greatly simplifies the computation. The analytical model was validated experimentally using a detailed simulation. Results show a very close match between the simulation and analytically derived performance measures.

The analytic and simulation data indicate that the idealized adaptive router can exploit multiple minimal path alternatives even in networks with dimensions as low as three. It can therefore outperform a dimension ordered router when the number of alternative paths is large as in large dimension networks or when the cost of message congestion is high as in large size messages. The data also indicates that routing performance is strongly affected by the buffer placement and organization scheme used in the system.

Adaptive routing is an expensive and complicated procedure, but it can potentially deliver high performance in interconnection networks. As shown in this paper, however, its performance is not dependent on one single factor, but rather on the interactions between various network and message parameters. Future work will address the impact of some of these parameters, such as the use of wormhole flow control, changes in selection policies, and the addition of deadlock avoidance mechanisms. It is important to understand the effects of these and other design choices on adaptive routing, in order that they may be efficiently utilized in future multicomputer interconnection networks.

References

[Aga91] A. Agarwal. Limits on interconnection network performance. *IEEE Trans. on Parallel and Distributed Systems*, 2(4):398–412, October 1991.

[AV92] Vikram S. Adve and Mary K. Vernon. Performance analysis of multiprocessor mesh interconnection networks. Technical Report Computer Sciences Tech Report 1001a, University of Wisconsin-Madison, 1992.

[BC93] R. V. Boppana and S. Chalasani. A comparison of adaptive wormhole routing algorithms. *Proc. 20th Int. Symp. Comp. Arch.*, pages 351–360, 1993.

[CE91] S. Chittor and R. Enbody. Predicting the effect of mapping on the communication performance of large multicomputers. *1991 Int. Conf. Par. Proc.*, pages II–1 – II–4, 1991.

[CK92] A. A. Chien and J. H. Kim. Planar-adaptive routing: low cost adaptive networks for multiprocessors. *Proc. 19th Ann. Symp. Comp. Arch.*, pages 268–277, may 1992.

[Com85] Intel Scientific Computers. ipsc user's guide. Order No. 175455-001, Santa Clara, CA, August 1985.

[Dal89] W. J. Dally. The J-machine: system support for actors. In Hewitt and Agha, editors, *Actors: Knowledge Based Concurrent Computing*. MIT Press, 1989.

[Dal90] W. J. Dally. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. on Computers*, 39(6):775–785, June 1990.

[Dal92] W. J. Dally. Virtual-channel flow control. *IEEE Trans. on Parallel and Distributed Systems*, 3(2):194–205, March 1992.

[DG92] J. T. Draper and J. Ghosh. Multipath e-cube algorithms (MECA) for adaptive wormhole routing and broadcasting in k-ary n-cubes. In *Fifth Int. Par. Proc. Symp.*, pages 407–410, Beverly Hills, CA, 1992.

[DS87] 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.

[Dua93] 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.

[Fea91] S. Felperin and L. Gravano et al. Routing techniques for massively parallel communication. *Proc. IEEE*, 79(4):488–503, April 1991.

[Fen81] T. Feng. A survey of interconnection networks. *IEEE Computer*, pages 12 – 27, December 1981.

[GN92a] C. J. Glass and L. M. Ni. Adaptive routing in mesh-connected networks. In *Proc.Int. Conf. on Distributed Computing Systems*, pages 12–19, Yokohama, Japan, June 9-12 1992.

[GN92b] C. J. Glass and L. M. Ni. Maximally fully adaptive routing in 2D meshes. In *Proc. 1992 Int. Conf. Par. Proc.*, August 1992.

[Hay86] J. P. Hayes. Architecture of a hypercube supercomputer. *Proc. 1986 Int. Conf. Par. Proc.*, pages 653–660, 1986.

[KC93] J. H. Kim and A. A. Chien. The impact of packetization in wormhole-routed networks. In *Proc. PARLE '93*, pages 242–253, 1993.

[KD91] J. Kim and C. R. Das. Modeling wormhole routing in a hypercube. In *Proc. Int. Conf. Dist. Comp. Sys.*, pages 386–393, May 1991.

[KHM87] M. J. Karol, M. G. Hluchy, and S. P. Morgan. Input versus output queueing on a space-division packet switch. *IEEE Trans. Communications*, COM-35(12):1347–1356, December 1987.

[KK79] P. Kermani and L. Kleinrock. Virtual cut-through: a new computer communication switching technique. *Computer Networks*, 3:267 – 286, 1979.

[Kle75] L. Kleinrock. *Queueing Systems.* John Wiley & Sons, 1975.

[LDT93] Z. Liu, J. Duato, and L. Thorelli. Grouping virtual channels for deadlock-free adaptive wormhole routing. In *Proc. PARLE '93*, pages 254–265, 1993.

[LGS92] G. Denicolay L. Gravano, G. D. Pifarre and J. L. C. Sanz. Adaptive deadlock-free wormhole routing in hypercubes. In *Fifth Int. Par. Proc. Symp.*, pages 512–517, Beverly Hills, CA, 1992.

[LH91] 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.

[LMN93] X. Lin, P. K. McKinley, and L. M. Ni. The message flow model for routing in wormhole-routed networks. In *Proc. Int. Conf. on Parallel Processing*, pages I294–I297, 1993.

[LNSS93] A. Lagman, W. A. Najjar, S. Sur, and P. K. Srimani. Evaluation of idealized adaptive routing on k-ary n-cube. In *Proc. IEEE Symposium on Parallel and Distributed Processing*, pages 166–169, 1993.

[NLSS94] W. A. Najjar, A. Lagman, S. Sur, and P. K. Srimani. Modeling adaptive routing in k-ary n-cube interconnection networks. In *Proc. International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pages 120–125, 1994.

[NM93] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, pages 62–76, February 1993.

[Now89] A. Nowatzyk. A communication architecture for multiprocessor networks. Technical Report CMU-C-89-181S-*, Cargnegie Mellon University, 1989.

[PS82] C. Papadimitrou and K. Steiglitz. *Combinatorial Optimization*, pages 225–226. Prentice-Hall, Inc., 1982.

[Res93] CRAY Research. *CRAY T3D System Architecture Overview*, 1993.

[SJ93] L. Schwiebert and D. N. Jayasimha. Optimal fully adaptive wormhole routing for meshes. In *Proc. Supercomputing '93*, pages 782–791, November 1993.

[SS93] C-C. Su and K. G. Shin. Adaptive deadlock-free routing in multicomputers using only one extra virtual channel. *icpp*, 1993.

[TF88] Y. Tamir and G. L. Frazier. High performance multi-queue buffers for vlsi communication swithces. *Proc. 1988 International Symposium on Computer Architecture*, 1988.

[Tuc84] A. Tucker. *Applied Combinatorics.* John Wiley & Sons, second edition, 1984.

[Val82] L. G. Valiant. A scheme for fast parallel computation. *SIAM J. Comp.*, 11:350–361, 1982.