The Turn Model for Adaptive Routing *

Christopher J. Glass and *Lionel M. Ni*

Advanced Computer Systems Laboratory
Department of Computer Science
Michigan State University
East Lansing, MI 48824-1027
{glass,ni}@cps.msu.edu

Abstract

We present a model for designing wormhole routing algorithms that are deadlock free, livelock free, minimal or nonminimal, and maximally adaptive. A unique feature of this model is that it is not based on adding physical or virtual channels to network topologies (though it can be applied to networks with extra channels). Instead, the model is based on analyzing the directions in which packets can turn in a network and the cycles that the turns can form. Prohibiting just enough turns to break all of the cycles produces routing algorithms that are deadlock free, livelock free, minimal or nonminimal, and maximally adaptive for the network. In this paper, we focus on the two most common network topologies for wormhole routing, n-dimensional meshes and k-ary n-cubes, without extra channels. In an n-dimensional mesh, just a quarter of the turns must be prohibited to prevent deadlock. The remaining three quarters of the turns permit partial adaptiveness in routing. Partially adaptive routing algorithms are described for 2D meshes, n-dimensional meshes, k-ary n-cubes, and hypercubes. Simulations of partially adaptive and nonadaptive routing algorithms for 2D meshes and hypercubes show that which algorithm has the lowest latencies and highest sustainable throughput depends on the pattern of message traffic. For nonuniform traffic, partially adaptive routing algorithms perform better than nonadaptive ones.

1 Introduction

Direct networks have become a popular interconnection architecture for constructing large-scale multiprocessors, such as multicomputers and scalable shared-memory multiprocessors. Direct networks offer massive parallelism [1, 2, 3, 4, 5] and are far more scalable than other approaches to multiprocessor interconnection. Systems based on direct networks are organized as ensembles of nodes, where each node has its own processor, local memory, and other supportive devices. The nodes communicate by sending messages. The network connects each node directly to only a few other nodes, its *neighbors*. Which nodes are neighbors is defined by the *topology* of the network. A node communicates with a node that is not its neighbor by sending a message through one of its neighbors. To handle the complexities of routing messages in the direct network, each node often has a *router*. The router controls local input and output channels, which connect it to local devices, and network input and output channels, which connect it to neighboring routers.

Wormhole routing [6] is becoming a popular switching technique in direct networks. Wormhole routing switches a message along a network by first dividing the message into packets and the packets into *flow control digits* or *flits*. It then routes the flits in each packet through the network in a pipeline fashion. *Header flits* contain all of the routing information for a packet and lead each packet through the network. When the header flits reach a router that has no suitable output channel available, all of the flits in the packet wait where they are for a suitable channel to become available. One of the attractions of wormhole routing is that each router requires just enough buffer space to store a few flits for each channel. The *store-and-forward* and *virtual cut-through* switching techniques [7] require enough buffer space to store an entire packet for each channel. Another attraction of wormhole routing is that its communication latencies are low. In the absence of contention, the latencies for store-and-forward are proportional to the *product* of packet length and distance to travel [8]. The latencies for wormhole routing, virtual cut-through, and *circuit switching*, on the other hand, are proportional to the *sum* of packet length and distance to travel. Wormhole routing also has some advantages over circuit switching. Channel reservation and release are an integral part of wormhole routing, but are separate phases in circuit switching. Wormhole routing also permits routers to replicate flits and output them over multiple channels, making multicast and broadcast communications possible [9, 1].

The majority of network topologies for wormhole routing are *n-dimensional meshes* and *k-ary n-cubes*, particularly low-dimensional meshes and *hypercubes*. A 2D mesh is used in the Intel Touchstone DELTA [2], the Intel Paragon, and the Symult 2010 [10]; a 3D mesh is used in the MIT J-machine [11], and the Caltech MOSAIC; and a hypercube is used in the nCUBE-2 [1]. Formally, an n-dimensional mesh has $k_0 \times k_1 \times \cdots \times k_{n-2} \times k_{n-1}$ nodes, k_i nodes along each dimension i, where $k_i \geq 2$. Each node $\mathbf{X}$ is identified by n coordinates, $(x_0, x_1, \ldots, x_{n-2}, x_{n-1})$, where $0 \leq x_i \leq k_i - 1$ for $0 \leq i \leq n-1$. Two nodes $\mathbf{X}$ and $\mathbf{Y}$ are neighbors if and only if $x_i = y_i$ for all i, $0 \leq i \leq n-1$, except one, j, where $y_j = x_j \pm 1$. Thus, nodes have from n to $2n$ neighbors, depending on their location in the mesh. In a k-ary n-cube [8], all nodes have the same number of neighbors. The definition of a k-ary n-cube differs from that of an n-dimensional mesh in that all of the k_i's are equal to k and two nodes $\mathbf{X}$ and $\mathbf{Y}$ are neighbors if and only if $x_i = y_i$ for all i, $0 \leq i \leq n-1$, except one, j, where $y_j = (x_j \pm 1) \bmod k$. The change to modular arithmetic in the definition adds wraparound channels to the k-ary n-cube, giving it symmetry. Every node has n neighbors if $k = 2$ and $2n$ neighbors if $k > 2$. Another topology with symmetry is the hypercube, which is a special case of both n-dimensional meshes and k-ary n-cubes. A hypercube is an n-dimensional mesh in which $k_i = 2$ for $0 \leq i \leq n-1$ or a 2-ary n-cube.

*This work was supported in part by the NSF grant ECS-88-14027.

Meshes and k-ary n-cubes are popular in part because their regular topologies simplify routing. In general, low-dimensional meshes are preferred over high-dimensional meshes and k-ary n-cubes. Low-dimensional meshes have low, fixed node degrees, which makes them more scalable than high-dimensional meshes and k-ary n-cubes. They also have fewer channels and higher channel bandwidth per bisection density and have lower contention and blocking latencies, which result in lower communication latencies and higher hot-spot throughputs [8]. On the other hand, high-dimensional meshes and k-ary n-cubes have some advantages that partially offset those of low-dimensional meshes. They tend to have lower diameters, which shortens path lengths. They also have more paths between pairs of nodes, which permits more fault tolerance. Finally, the symmetry of k-ary n-cubes can make it easier to spread packet traffic more evenly.

Algorithms for routing message packets through a network topology should have three characteristics: low communication latency, high network throughput, and ease of implementation in VLSI. Features contributing to ease of implementation are little hardware for channels, buffers, and control logic. Features contributing to low latency and high throughput are freedom from deadlock, freedom from indefinite postponement, freedom from livelock, fault tolerance, routing packets along short paths, spreading packet traffic evenly, and routing packets adaptively. Many of these terms require some definition (partly because they have been used in different ways by different authors). *Deadlock* occurs when a packet waits for an event that cannot happen. For example, a packet may wait for a network resource to be released by a packet that is, in turn, waiting for the first packet to release some resource. In wormhole routing, such a *circular wait* condition will cause deadlock, because a packet holds resources while waiting and excludes other packets from acquiring the held resources. *Indefinite postponement* is similar to deadlock, but occurs when a packet waits for an event that can happen but never does. For example, a packet may wait forever to acquire a network resource for which other packets are always competing successfully. The issue here is *fairness*. Both deadlock and indefinite postponement can stop a packet from being injected into a network or from moving once it is in the network. In contrast, livelock does not stop a packet's movement, but rather its progress toward the destination. *Livelock* occurs when the routing of a packet never leads it to its destination. Livelock is possible only when routing is *adaptive* (not along a predetermined path) and *nonminimal* (possibly away from or equidistant to the destination at times). *Minimal* routing, in contrast, restricts packets to shortest paths. Although minimal routing may initially sound more promising, nonminimal routing provides better fault tolerance.

Of these features, the most important is freedom from deadlock. Deadlock can keep many or all packets from reaching their destinations and occurs readily unless a routing algorithm includes preventive measures. Indefinite postponement and livelock are less likely to occur and are generally easier to prevent. Adaptiveness is also an important feature, because it contributes to several of the other features [12]. It provides alternative paths for packets that encounter continuously blocked channels, faulty hardware, or hot spots in traffic patterns.

The drawback to previous routing algorithms for meshes and k-ary n-cubes is that they either sacrifice adaptiveness for deadlock freedom or achieve adaptiveness and deadlock freedom at the expense of adding physical or virtual channels. The popular *xy routing algorithm* for 2D meshes [10, 2] routes a packet first along the x dimension (dimension 0) and then along the y dimension (dimension 1). The *e-cube routing algorithm* for hypercubes [13] routes a packet first along the lowest dimension and then along higher and higher dimensions. Ordering the dimensions in this way ensures that the xy and e-cube algorithms avoid deadlock, but it also ensures nonadaptiveness. Another popular way to avoid deadlock, and possibly to provide adaptiveness, is to add *virtual channels* to networks. Dally and Seitz [14] introduced the idea for nonadaptive routing, and several researchers [15, 12, 16] have extended it to adaptive routing. Adding a virtual channel to a physical channel is less expensive than adding a new physical channel, but it is not free. It involves adding buffer space and control logic to the two routers at the ends of the physical channel so that the virtual channels can share the physical channel and routers. It also reduces the bandwidths of the virtual channels already sharing the physical channel. An advantage of adding virtual or physical channels, however, is that they can support routing algorithms with a high degree of adaptiveness. A minimal, *fully adaptive* algorithm can route packets along any of the shortest paths in the topology. Dally [17] and Linder and Harden [16] describe such an algorithm for 2D meshes. A *partially adaptive* algorithm cannot route packets along every shortest path.

This paper presents a model for designing wormhole routing algorithms that are deadlock free, livelock free, minimal or nonminimal, and maximally adaptive for networks. A unique feature of this model is that it is not based on adding physical or virtual channels to network topologies (though it can be applied to networks with extra channels). Instead, the model is based on analyzing the directions in which packets can turn in a network and the cycles that the turns can form. Section 2 presents the model in general terms and applies it to n-dimensional meshes without extra channels. Sections 3, 4, and 5 describe many of the partially adaptive routing algorithms derived for 2D meshes, n-dimensional meshes and k-ary n-cubes, and hypercubes respectively. Simulations studying the performance of partially adaptive and nonadaptive algorithms in 2D meshes and hypercubes for different patterns of message traffic are described in Section 6. Section 7 concludes the paper.

2 Turn Model

Deadlock in wormhole routing is caused by packets waiting on each other in a cycle. Figure 1 shows one way that deadlock can occur in a 2D mesh. Four packets travelling in different directions try to turn left and wind up in a circular wait. If only one of the packets had not turned, this deadlock would have been avoided. This suggests that, by prohibiting certain turns in a network, a routing algorithm might prevent deadlock altogether. The routing algorithm would have to prohibit at least one turn in each of the many possible cycles. At the same time, the algorithm would have to leave a path between every pair of nodes. In addition, it should not prohibit more turns than necessary; otherwise, the adaptiveness of the algorithm would be reduced.

To solve this problem of designing wormhole routing algorithms that are deadlock free and maximally adaptive for a network, we propose the *turn model*. The model involves analyzing the directions in which packets can turn in the network and the cycles that the turns can form. It prohibits just enough turns to break all of the cycles. Routing algorithms that employ the remaining turns are deadlock free, livelock free, minimal or nonminimal, and maximally adaptive for the network. The model produces partially adaptive routing algorithms for such basic topologies as mesh-connected, k-ary n-cube, hexagonal, octagonal, and cube-connected cycle networks. Adding extra physical or virtual channels to the topologies allows the model to produce fully adaptive routing algorithms, the topic of a forthcoming paper [18]. The current paper focuses on improving performance without the expense of extra channels, as might be done by changing the routing algorithms in existing routers.

The steps of the turn model are given in more detail below. Unless specified otherwise, the term "channel" will be used to indicate either a virtual or physical channel.

1. Partition the channels in the network into sets according to the directions in which they route packets. If each node has v channels in a physical direction, treat these channels as being in v distinct *virtual directions* and divide them into v distinct sets accordingly. Put any wraparound channels (for tori) in a separate set to be incorporated during Step 5.

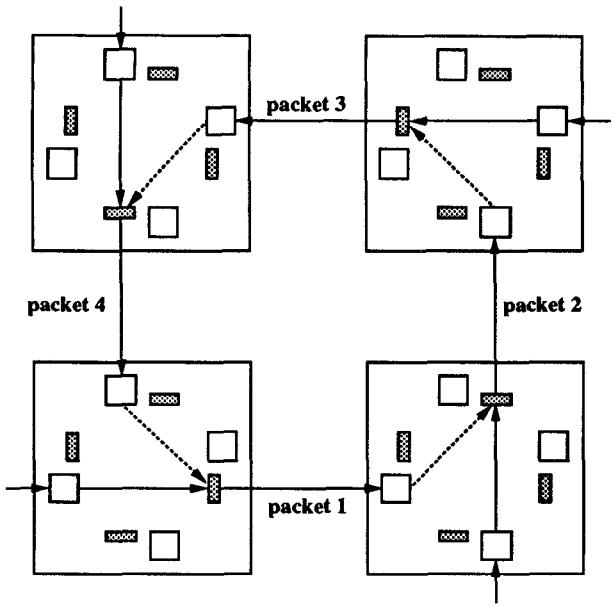


Figure 1. A wormhole deadlock involving four routers and four packets in a 2D mesh.

2. Identify the possible turns from one virtual direction to another, ignoring 180-degree and 0-degree turns. A 0-degree turn is only possible when there are multiple channels in one direction. It represents a transition from one set of channels to another when the two sets route packets in the same physical direction, but different virtual directions.

3. Identify the cycles that these abstract turns can form. Generally, identifying the simplest cycles in each plane of the topology is adequate.

4. Prohibit one turn in each abstract cycle so as to prevent deadlock. The turns must be chosen carefully in order to break every possible cycle, including complex cycles not identified in Step 3. A useful approach is first to break the cycles in each plane and then to check whether this allows more complex cycles.

5. Incorporate as many turns as possible from the set of wraparound channels, without reintroducing cycles. At least one turn for each wraparound channel can always be incorporated.

6. Incorporate as many 180-degree and 0-degree turns as possible, without reintroducing cycles.

Routing algorithms that route packets along the sets of channels identified in Step 1 and use only the turns from one set to another allowed by Steps 4, 5, and 6 are deadlock free, livelock free, minimal or nonminimal, and maximally adaptive for the network. They are deadlock free because breaking all of the cycles prevents circular waits. (We demonstrate deadlock freedom for individual routing algorithms later.) Preventing circular waits in this way means that it is possible to number the channels in the network so that each algorithm routes every packet along channels in strictly decreasing (or increasing) order [14]. This, together with the fact that a network contains a finite number of channels, means that a packet will reach its destination after a limited number of hops. Thus, routing is livelock free. Routing is maximally adaptive for the network because the model prohibits the minimum number of turns from one direction to another. The algorithms the model produces will also be nonminimal, but can easily be made minimal by modifying them to use channels only when they lead toward the destination. Nonminimal routing is more adaptive and fault tolerant, though.

To make the steps of the turn model clearer, the remainder of this section applies them to n-dimensional meshes, starting with 2D meshes. For 2D meshes, we first simplify the terminology used in the definition of n-dimensional meshes. Dimensions 0 and 1 become x and y. The lengths of the dimensions, k_0 and k_1, become m and n. The directions $-x$, $+x$, $-y$, and $+y$ become west, east, south, and north. From these four directions, eight 90-degree turns can be formed: left and right turns when travelling west, east, south, and north. The eight turns form two abstract cycles as shown in Figure 2. The xy routing algorithm prevents deadlock by prohibiting four of the turns (Figure 3). The remaining four turns cannot form a cycle; they also do not allow any adaptiveness. Deadlock can be prevented by prohibiting fewer than four turns, however. In fact, only two turns need to be prohibited, one from each abstract cycle, allowing for some adaptiveness in routing. Prohibiting any two turns will not prevent deadlock, however, as Figure 4 illustrates. The three left turns allowed in Figure 4a are equivalent to the prohibited right turn in Figure 4b, and the three right turns allowed in Figure 4b are equivalent to the prohibited left turn in Figure 4a. Thus, both cycles still exist and deadlock is possible (Figure 4c). Section 3 describes which pairs of turns can be prohibited to prevent deadlock and describes the partially adaptive routing algorithms based on these turns.

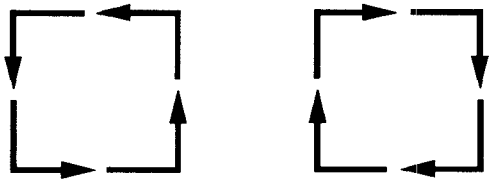


Figure 2. The possible abstract cycles and turns in a 2D mesh.

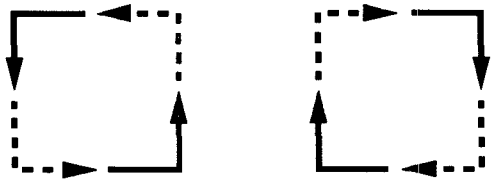


Figure 3. Only four turns (solid lines) are allowed in the xy routing algorithm.

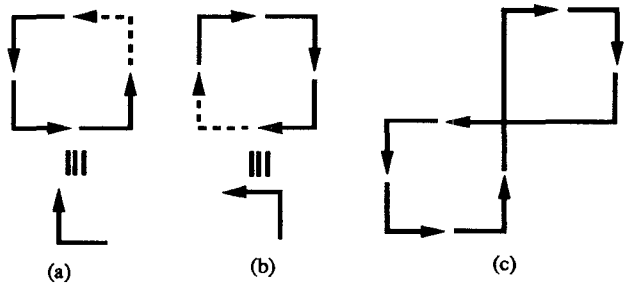


Figure 4. Six turns that complete the abstract cycles and allow deadlock.

In an n-dimensional mesh, each node has up to $2n$ input channels and $2n$ output channels. For each of these $2n$ directions, there are $2n - 2$ possible 90-degree turns to a different direction, making $4n(n-1)$ total turns. These turns form two abstract cy-

cles in each of the $n(n-1)/2$ planes, making $n(n-1)$ total cycles of four turns.

Theorem 1 *The minimum number of turns that must be prohibited to prevent deadlock in an n-dimensional mesh is $n(n-1)$, or a quarter of the possible turns.*

Proof: The $4n(n-1)$ turns in an n-dimensional mesh can be partitioned into $n(n-1)$ abstract cycles of four turns each. Each of the $n(n-1)/2$ planes contains two of these cycles. At least one of the four turns in each cycle must be prohibited in order to prevent these cycles. Therefore, prohibiting a quarter of the turns is necessary to prevent deadlock. □

In the following sections, we introduce specific partially adaptive routing algorithms based on the turn model for 2D mesh, n-dimensional mesh, k-ary n-cube, and hypercube topologies. A detailed study for the 3D mesh can be found in [19].

3 Partially Adaptive Routing in 2D Meshes

In a 2D mesh, there are four directions, eight 90-degree turns, and two abstract cycles of four turns. One turn from each cycle must be prohibited to prevent deadlock. Of the 16 different ways to prohibit these two turns, 12 prevent deadlock (see Figure 4 for a deadlock case) and three are unique if symmetry is taken into account.

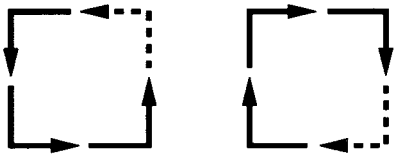
3.1 West-First Routing Algorithm

Figure 5a shows one way to prohibit two turns in a 2D mesh. The prohibited turns are the two to the west. Therefore, to travel west, a packet must start out in that direction. This suggests the *west-first routing algorithm*: route a packet first west, if necessary, and then adaptively south, east, and north. Example paths for the west-first algorithm are shown in Figure 5b. In this figure, black squares represent nodes, and gray bars indicate blocked channels requiring that packets wait (dashed lines) or take alternative paths. Note that both minimal and nonminimal paths are shown.

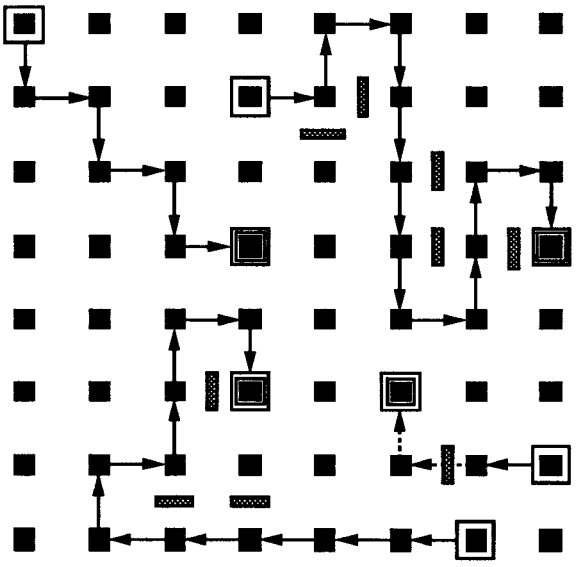
Proof that the west-first partially adaptive algorithm is deadlock free is based on the work of Dally and Seitz [14], who show that a routing algorithm is deadlock free if the channels in the interconnection network can be numbered so that the algorithm routes every packet along channels with strictly decreasing (or increasing) numbers. Because the west-first algorithm routes packets first west and then in the other directions, we assign lower numbers to westward channels the farther west they are, and assign still lower numbers to eastward, northward, and southward channels the farther east they are.

Theorem 2 *The west-first routing algorithm is deadlock free.*

Proof: Assign each channel in the $m \times n$ mesh a two-digit number a, b in base r, where r is the greater of $3m-2$ and $n-1$. Figure 6 shows the particular numbers to assign to the channels leaving node (x, y). Figure 7 illustrates the numbering of a 4×4 mesh. To show that west-first routes every packet along channels with strictly decreasing numbers, it is sufficient to show that, for each channel into an arbitrary node, the algorithm can only route the packet out along a channel with a lower number. Figure 8 shows the four possible cases. Examination shows that, in each case, the channels used to route a packet out of a node have lower numbers than the input channel. Note that in part (c) a packet can make a 180-degree turn. This turn is only useful for nonminimal routing. □



(a) The six turns allowed (solid lines) by the west-first algorithm.



(b) Examples of the west-first algorithm in an 8×8 mesh.

Figure 5. The west-first routing algorithm for 2D meshes.

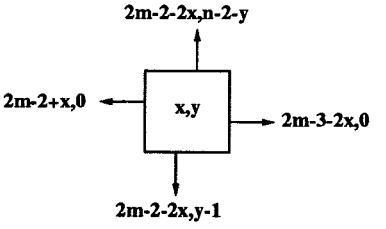


Figure 6. Numbering of the channels leaving each node (x, y) for the west-first routing algorithm.

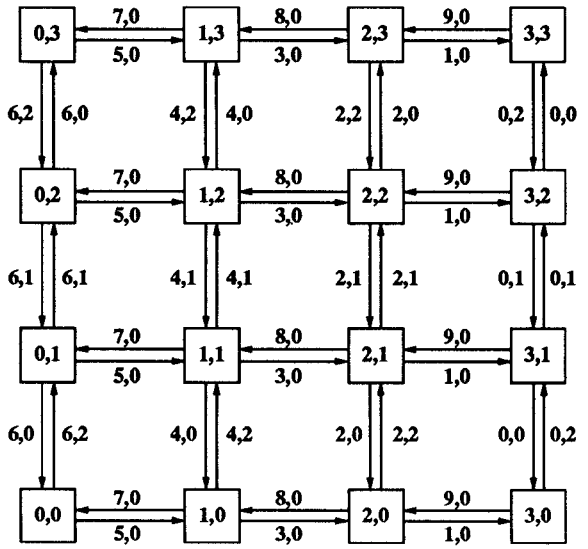


Figure 7. Numbering of a 4 × 4 mesh for the west-first routing algorithm.

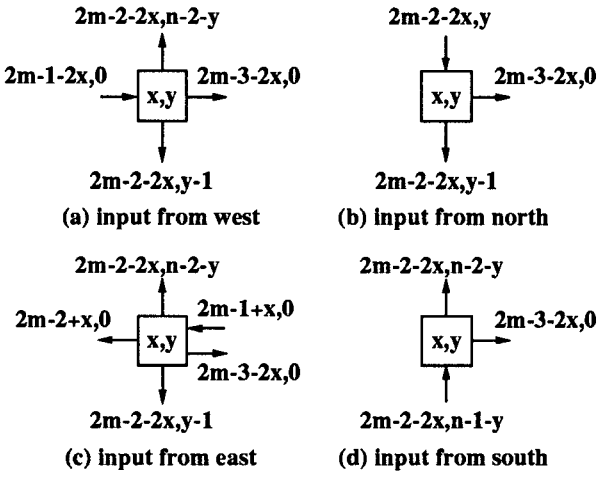
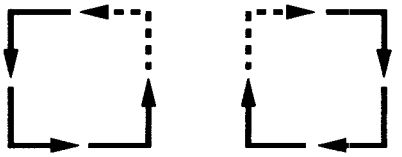


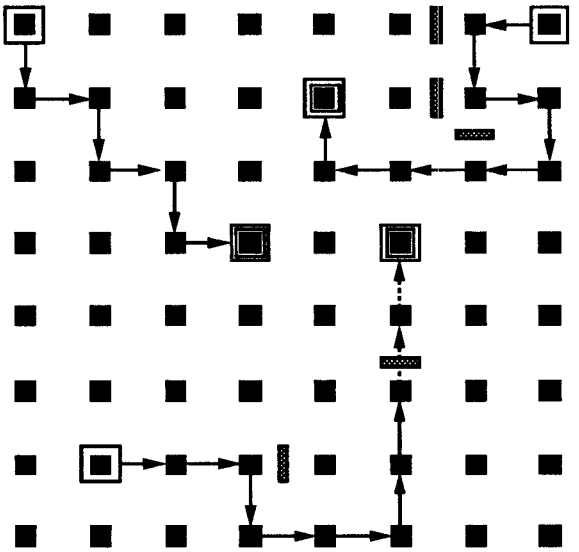
Figure 8. For each input channel, the output channels allowed by the west-first routing algorithm have lower numbers.

3.2 North-Last Routing Algorithm

Figure 9a shows another way to prohibit two turns in a 2D mesh. The prohibited turns are the two when travelling north. Therefore, a packet should only travel north when that is the last direction it needs to travel. This suggests the *north-last routing algorithm*: route a packet first adaptively west, south, and east, and then north. Example paths for the north-last algorithm are shown in Figure 9b.



(a) The six turns allowed (solid lines) by the north-last algorithm.



(b) Examples of the north-last algorithm in an 8 × 8 mesh.

Figure 9. The north-last routing algorithm for 2D meshes.

Theorem 3 *The north-last routing algorithm is deadlock free.*

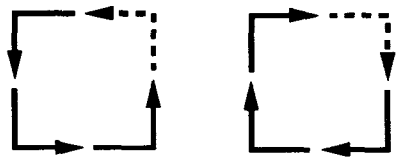
Proof: The proof is similar to that for Theorem 2. Rotate Figures 6 and 7 counterclockwise 90 degrees, and reverse the directions of the channels. The figures now show that north-last routes every packet along channels with strictly increasing numbers. □

3.3 Negative-First Routing Algorithm

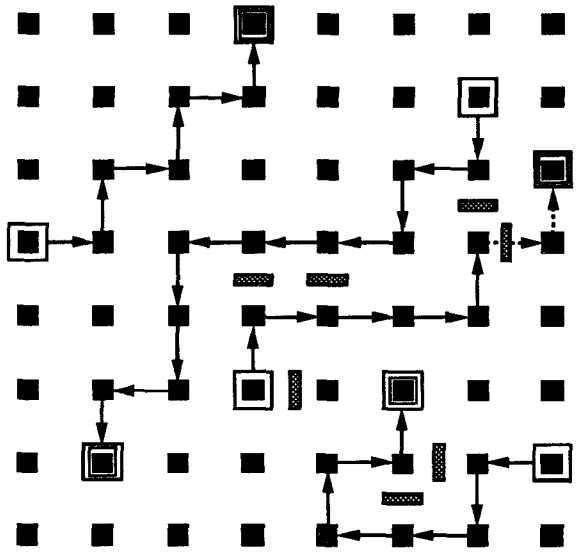
Figure 10a shows a third way to prohibit two turns in a 2D mesh. The prohibited turns are the two from a positive direction to a negative direction. Therefore, to travel in a negative direction, a packet must start out in a negative direction. This suggests the *negative-first routing algorithm*: route a packet first adaptively west and south, and then adaptively east and north. Yantchev and Jesshope propose a similar algorithm [15], but only for minimal routing. The negative-first algorithm can be either minimal or nonminimal, the nonminimal version being more adaptive and fault tolerant. Example paths for the negative-first algorithm are shown in Figure 10b.

Theorem 4 *The negative-first routing algorithm is deadlock free.*

The proof is a special case of the one given for n-dimensional meshes in Section 4.



(a) The six turns allowed (solid lines) by the negative-first algorithm.



(b) Examples of the negative-first algorithm in an 8 × 8 mesh.

Figure 10. The negative-first routing algorithm for 2D meshes.

3.4 Degree of Adaptiveness

How adaptive are these partially adaptive routing algorithms? Let $S_{algorithm}$ be the number of shortest paths the algorithm allows from source node (s_x, s_y) to destination node (d_x, d_y). Also, let f be a fully adaptive algorithm, p be one of the three partially adaptive algorithms, $\Delta x = |d_x - s_x|$, and $\Delta y = |d_y - s_y|$. Then,

$$S_f = \frac{(\Delta x + \Delta y)!}{\Delta x! \Delta y!}$$

$$S_{west-first} = \begin{cases} \frac{(\Delta x+\Delta y)!}{\Delta x!\Delta y!} & \text{if } d_x \geq s_x \\ 1 & \text{otherwise} \end{cases}$$

$$S_{north-last} = \begin{cases} \frac{(\Delta x+\Delta y)!}{\Delta x!\Delta y!} & \text{if } d_y \leq s_y \\ 1 & \text{otherwise} \end{cases}$$

$$S_{negative-first} = \begin{cases} \frac{(\Delta x+\Delta y)!}{\Delta x!\Delta y!} & \text{if } (d_x \leq s_x \text{ and } d_y \leq s_y) \\ & \text{or } (d_x \geq s_x \text{ and } d_y \geq s_y) \\ 1 & \text{otherwise} \end{cases}$$

For minimal routing algorithms, the larger $S_{algorithm}$ is, the more adaptive the algorithm is. For the three partially adaptive routing algorithms, however, $S_p = 1$ for at least half of the source-destination pairs. Another measure of the degree to which a minimal routing algorithm is adaptive is the ratio $S_{algorithm}/S_f$. S_p/S_f ranges from 0 to 1, but averaged across all source-destination pairs, $S_p/S_f > 1/2$, which indicates a significant degree of adaptiveness. Note that $S_p = 1$ for a source-destination pair does not imply that algorithm p is nonadaptive for that pair. The partially adaptive algorithms all permit nonminimal routing. In the case of the negative-first algorithm, this means that routing can be adaptive even when $d_x \leq s_x$ and $d_y \geq s_y$ or when $d_x \geq s_x$ and $d_y \leq s_y$. For an illustration of this, see the bottom path in Figure 10b.

4 Partially Adaptive Routing in n-Dimensional Networks

This section describes the adaptive routing algorithms resulting from the application of the turn model to n-dimensional meshes and k-ary n-cubes. In general, these topologies are not practical unless n is small or the k's equal 2.

4.1 n-Dimensional Meshes

Of the many routing algorithms formed by prohibiting one turn per cycle, three are noteworthy for their simplicity. They are analogs of the west-first, north-last, and negative-first routing algorithms for 2D meshes. The analog of the west-first routing algorithm is the *all-but-one-negative-first routing algorithm*: route a packet first adaptively in the negative directions of all but one dimension $(n-1)$ and then adaptively in the other directions. The analog of the north-last routing algorithm is the *all-but-one-positive-last routing algorithm*: route a packet first adaptively in the negative directions and the positive direction of one dimension (0) and then adaptively in the other directions. The analog of the negative-first routing algorithm is also called the *negative-first routing algorithm*: route a packet first adaptively in the negative directions and then adaptively in the positive directions. All of these algorithms are deadlock free, but the only proof we present is for the negative-first algorithm.

Theorem 5 *The negative-first routing algorithm for n-dimensional meshes is deadlock free.*

Proof: Let K be the sum of the k_i for an n-dimensional mesh, and let X be the sum of the x_i for any node $(x_0, x_1, \ldots, x_{n-2}, x_{n-1})$. Number each channel leaving a node in a positive direction $K - n + X$ and each channel leaving a node in a negative direction $K - n - X$. Then, if a packet enters a node when travelling in a negative direction, it enters along a channel numbered $K - n - X - 1$, which is less than $K - n - X$ and $K - n + X$, the numbers of the channels leaving the node in the negative and positive directions. If a packet enters a node when travelling in a positive direction, it enters along a channel numbered $K - n + X - 1$, which is less than $K - n + X$, the number of the channels leaving the node in the positive directions. Therefore, the negative-first algorithm routes every packet along channels with strictly increasing numbers and is deadlock free. □

The negative-first algorithm is deadlock free as a result of prohibiting just one turn per abstract cycle, the turn from a positive direction to a negative direction. Therefore, prohibiting some quarter of the turns is sufficient to prevent deadlock in an n-dimensional mesh. Combining this with Theorem 1, we have the following theorem, which supports our claim that the turn model produces maximally adaptive routing algorithms.

Theorem 6 *Prohibiting some quarter of the turns (that is, $n(n-1)$) in an n-dimensional mesh is necessary and sufficient to prevent deadlock.*

As the number of dimensions increases, the minimal partially adaptive algorithms are more likely to be able to route messages adaptively. $S_p = 1$ less often, especially when the k_i are large. But averaged across all source-destination pairs, $S_p/S_f > 1/2^{n-1}$, indicating that the degree of adaptiveness relative to fully adaptive algorithms decreases as n increases. Again, adaptiveness can be increased by nonminimal routing.

4.2 k-ary n-cubes

The partially adaptive routing algorithms for meshes can be extended to use the wraparound channels of k-ary n-cubes. One way is to allow a packet to be routed along a wraparound channel only on its first hop. To prove that the routing is still deadlock free, number the mesh channels as before and assign the wraparound channels a number that is either greater than or less than those of the mesh channels, depending on whether packets are routed along channels in strictly decreasing or increasing order in the proof. The negative-first algorithm can be extended to k-ary n-cubes in another way: classify each wraparound channel according to the direction in which it routes packets and then apply the negative-first algorithm. Thus, a node at the east edge of the mesh channels will have two channels to the west: a mesh channel to the node immediately to its west and a wraparound channel to a node at the west edge of the mesh channels. Again, the proof of deadlock freedom is a simple modification of the proof for meshes.

Note that all of these routing algorithms are strictly nonminimal. For k-ary n-cubes with $k > 4$, it is impossible to construct deadlock-free routing algorithms that are minimal without adding extra channels. This is a result of the many cycles that do not involve turns in the topology. By adding channels to a k-ary n-cube, Linder and Harden [16] construct a routing algorithm that is deadlock free, minimal, and fully adaptive. They add, however, enough channels to partition the k-ary n-cube into 2^{n-1} subnetworks with $n+1$ levels per subnetwork and k^n channels per level.

5 p-cube Routing in Hypercubes

Hypercubes are a special case of both n-dimensional meshes and k-ary n-cubes. Consequently, the partially adaptive routing algorithms for hypercubes are special cases of the algorithms for n-dimensional meshes and k-ary n-cubes. Proofs that the routing algorithms are deadlock free are corollaries of the proofs for the more general cases.

The special case of the negative-first algorithm has a particularly compact expression, the *p-cube routing algorithm*. Let $\mathbf{S}$ be the binary address of the source node for a packet, $\mathbf{C}$ be the binary address of the node the header flits currently occupy, and $\mathbf{D}$ be the binary address of the destination node. The p-cube routing algorithm has two phases. In the case of minimal routing, the first phase routes the packet along a dimension i for which $c_i = 1$ and $d_i = 0$. When there is no such dimension, the second phase routes the packet along a dimension i for which $c_i = 0$ and $d_i = 1$. These steps are easily computed using bitwise logic operations as shown in Figure 11. If nonminimal routing is desired, because of its increased adaptiveness and fault tolerance, the first phase can also route the packet along any dimension i for which $c_i = 1$ and $d_i = 1$. Then, the steps can be computed as shown in Figure 12. In both of these algorithms, the only input transmitted in the header flits is $\mathbf{D}$. $\mathbf{C}$ is a unique constant for each router, and p depends on which input buffer the header flits occupy in the router. Konstantinidou proposes an algorithm similar to p-cube [20], but only for minimal routing.

The number of shortest paths from $\mathbf{S}$ to $\mathbf{D}$, S_{p-cube}, is $h_1!h_0!$, where $|\mathbf{X}|$ represents the number of 1's in the binary number $\mathbf{X}$, $h_1 = |(\mathbf{S} \wedge \bar{\mathbf{D}})|$, and $h_0 = |(\bar{\mathbf{S}} \wedge \mathbf{D})|$. For a fully adaptive routing algorithm, f, $S_f = h!$, where $h = h_1 + h_0 = |(\mathbf{S} \oplus \mathbf{D})|$ is the Hamming distance between $\mathbf{S}$ and $\mathbf{D}$. The other measure of the degree of adaptiveness for the p-cube routing is $S_{p-cube}/S_f = 1/\binom{h}{h_1}$.

Algorithm: Minimal p-cube routing for hypercubes.
Input: Current address, $\mathbf{C}$, and destination address, $\mathbf{D}$.
Procedure:

1. If $\mathbf{C} = \mathbf{D}$, route the packet to the local processor and exit.
2. $\mathbf{R} = \mathbf{C} \wedge \bar{\mathbf{D}}$.
3. If $\mathbf{R} = \mathbf{0}$, then $\mathbf{R} = \bar{\mathbf{C}} \wedge \mathbf{D}$.
4. Route the packet along any available channel in a dimension i for which $r_i = 1$.

Figure 11. The minimal p-cube routing algorithm for hypercubes.

Overall, the p-cube routing algorithm offers a choice of many shortest paths, especially when compared to the nonadaptive e-cube routing algorithm. The following table illustrates this for a binary 10-cube where the source node (1011010100) sends a message to the destination node (0010111001). For this example, $h = 6$, $h_0 = 3$, and $h_1 = 3$. One of the 36 possible shortest paths is shown. For each node transmitting the message, the number of choices based on the p-cube routing is also shown. The number of choices in parentheses indicates the additional choices available with nonminimal routing.

Algorithm: Nonminimal p-cube routing for hypercubes.
Input: Current address, $\mathbf{C}$; destination address, $\mathbf{D}$; and whether the last hop was in a positive direction, p.
Procedure:

1. If $\mathbf{C} = \mathbf{D}$, route the packet to the local processor and exit.
2. If $p = 1$, then $\mathbf{R} = \bar{\mathbf{C}} \wedge \mathbf{D}$.
3. Else if $\mathbf{C} \wedge \bar{\mathbf{D}} = \mathbf{0}$, then $\mathbf{R} = \mathbf{C} \vee (\bar{\mathbf{C}} \wedge \mathbf{D})$.
4. Else $\mathbf{R} = \mathbf{C}$.
5. Route the packet along any available channel in a dimension i for which $r_i = 1$.

Figure 12. The minimal p-cube routing algorithm for hypercubes.

address	choices	dimension taken	comment
1011010100	3(+2)	2	source
1011010000	2(+2)	9	phase 1
0011010000	1(+2)	6	phase 1
0010010000	3	5	phase 2
0010110000	2	0	phase 2
0010110001	1	3	phase 2
0010111001			destination

6 Simulation Experiments

To compare the partially adaptive routing algorithms all-but-one-negative-first (ABONF), all-but-one-positive-last (ABOPL), and negative-first (NF) with the nonadaptive routing algorithms xy and e-cube, we simulate a 16×16 mesh and a binary 8-cube for three different traffic patterns. Each of these topologies contains 256 nodes. A pair of unidirectional channels connects each pair of neighboring routers and each router to its local processor. All

of the channels have the same bandwidth, 20 flits/μsec. Each input channel into a router has a buffer the size of a single flit. The routers operate asynchronously and synchronize to simultaneously transmit the flits in a packet.

The processors generate messages at time intervals chosen from a negative exponential distribution. Each message has an equal probability of being one packet of 10 or 200 flits. Messages that are blocked from immediately entering the network are queued at the source processor. Messages that arrive at a destination processor are immediately consumed. When multiple input channels contain header flits waiting for the same available output channel, an *input selection policy* must arbitrate. The policy used is called *local first-come-first-served* and decides in favor of the header flits that arrived in the router first. This policy is fair and therefore prevents indefinite postponement. When a header flit in an input channel has multiple output channels available to it, an *output selection policy* must arbitrate. The policy used is called xy and decides in favor of the output channel along the lowest dimension. All routing is minimal.

For each simulation, two characteristics of network performance are measured: average communication latency (in μsec) and average sustainable network throughput (in flits delivered per μsec). The throughput is sustainable when the number of packets queued at their source processors is small and bounded.

Obviously, the network performance is largely determined by the message traffic pattern, which is application dependent. Three network workloads are considered: *uniform*, *matrix-transpose*, and *reverse-flip*. The uniform pattern sends each message to any of the other processors with equal probability. In the mesh, the matrix-transpose pattern sends each message from the processor at row i and column j to the one at row j and column i. In the hypercube, a matrix-transpose pattern is derived by mapping a 16 x 16 mesh to the hypercube so that neighbors in the mesh are neighbors in the hypercube. Messages are then sent to the nodes dictated by the matrix transpose in the mesh. The resulting pattern in the hypercube sends each message from the processor at $(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ to the one at $(\bar{x}_4, x_5, x_6, x_7, \bar{x}_0, x_1, x_2, x_3)$. The reverse-flip pattern sends each message from the processor at $(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7)$ to the one at $(\bar{x}_7, \bar{x}_6, \bar{x}_5, \bar{x}_4, \bar{x}_3, \bar{x}_2, \bar{x}_1, \bar{x}_0)$.

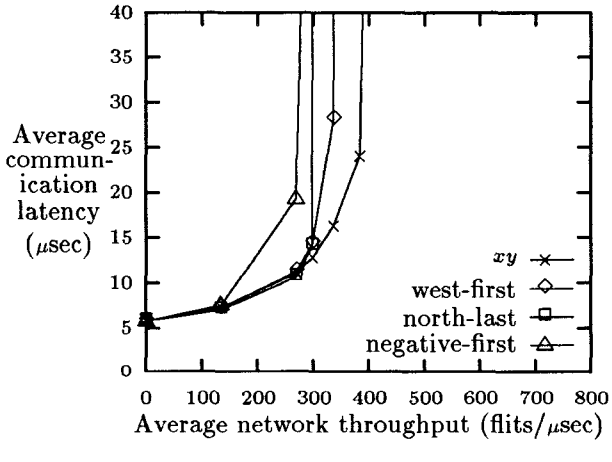


Figure 13. Comparison of routing algorithms for uniform traffic in a 16 x 16 mesh.

The simulations indicate that, for uniform traffic in the mesh and hypercube, the nonadaptive routing algorithms have lower latencies at high throughputs than the partially adaptive algorithms (Figure 13). At low throughputs, the algorithms perform about the same. For the nonuniform traffic patterns, the partially adaptive routing algorithms have the lower latencies, especially at high throughputs (Figures 14, 15, and 16). For matrix-transpose traffic in both the mesh and hypercube, the maximum sustainable throughput of the partially adaptive algorithms is twice that of the nonadaptive algorithms. For reverse-flip traffic, the maximum sustainable throughput of the partially adaptive algorithms is four times that of the nonadaptive e-cube algorithm.

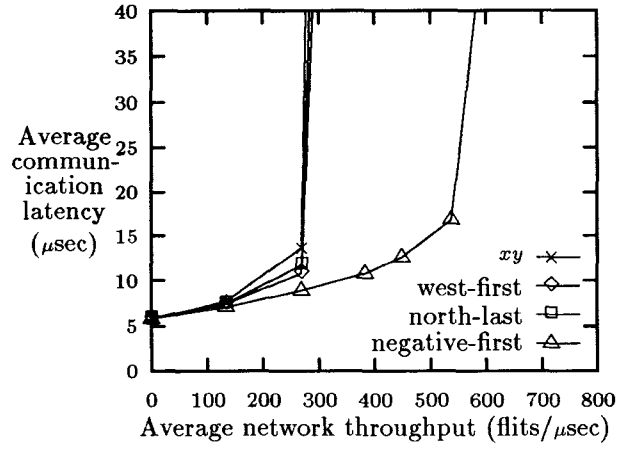


Figure 14. Comparison of routing algorithms for matrix-transpose traffic in a 16 x 16 mesh.

Overall in the hypercube, the highest sustainable throughputs are for the partially adaptive algorithms and reverse-flip traffic. These throughputs are 50% higher than the next highest sustainable throughput in the hypercube, which occurs for the e-cube algorithm and uniform traffic. This improvement in throughput is not due to shorter path lengths for reverse-flip traffic. The average path length for reverse-flip traffic is 4.27 hops, versus 4.01 hops for uniform traffic. Overall in the mesh, the highest sustainable throughput is for the negative-first algorithm and matrix-transpose traffic. This throughput is 30% higher than the second highest sustainable throughput in the mesh, which occurs for the xy algorithm and uniform traffic. Again, average path length is longer for matrix-transpose traffic (11.34 hops) than for uniform traffic (10.61 hops).

The reason the nonadaptive routing algorithms perform better than the partially adaptive routing algorithms for uniform traffic is that they happen to embody global, long-term information about this traffic pattern. From a global, long-term point of view, the uniform traffic pattern starts with message traffic spread evenly across the mesh or hypercube, and the xy and e-cube algorithms maintain that evenness. The adaptive algorithms, on the other hand, select channels based on local, short-term information. These selections tend to benefit just the routed packet and only for the immediate future and tend to interfere with other packets. The result is that the evenness of uniform traffic is not maintained as well as when global information is used.

Despite the superior performance of the nonadaptive routing algorithms for uniform traffic, the partially adaptive algorithms probably provide better performance in real systems. Uniform traffic has been used in many previous simulation studies, but we know of no real applications that generate uniform traffic. A traffic pattern is determined by the application and how its processes are mapped to the nodes of the network. For most applications, each node will communicate with some nodes much

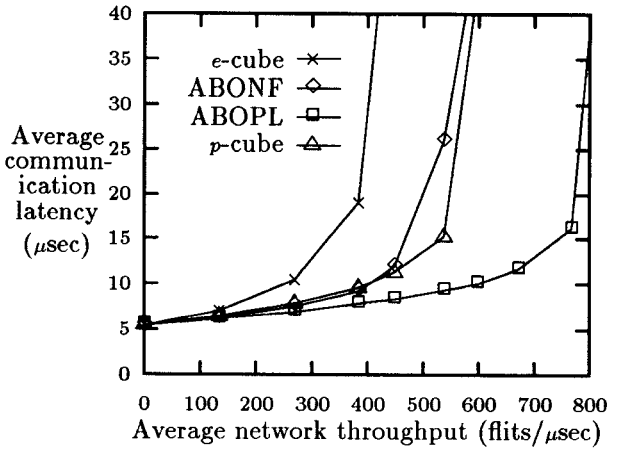


Figure 15. Comparison of routing algorithms for matrix-transpose traffic in an 8-cube.

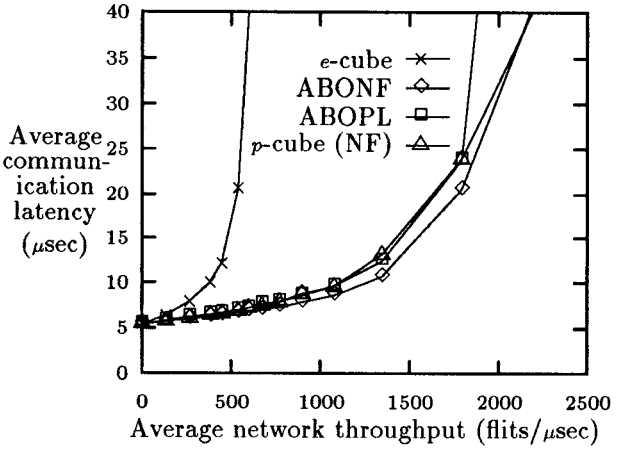


Figure 16. Comparison of routing algorithms for reverse-flip traffic in an 8-cube.

more than others. Nonuniform traffic presents a problem for the xy and e-cube algorithms because they are nonadaptive. Just as they maintain the evenness of uniform traffic, they blindly maintain the unevenness of nonuniform traffic. The result, as the figures illustrate, is often poor performance.

7 Conclusions and Future Work

Our goal has been to make the best use of the channels in wormhole-routed interconnection networks. Analyzing the directions in which packets can turn in a network and prohibiting the minimum number of turns that break all of the cycles produces routing algorithms that are deadlock free, livelock free, minimal or nonminimal, and maximally adaptive. Deadlock freedom and livelock freedom are essential for routing algorithms. Adaptiveness increases the chances that packets can avoid hot spots and faulty hardware and decreases the chances of indefinite postponement and livelock. Nonminimal routing allows even greater hot-spot avoidance and fault tolerance. The turn model, unlike other approaches to designing adaptive routing algorithms, is applicable to networks with only the channels required by the network topologies (as well as to networks with extra physical or virtual channels). Applied to n-dimensional meshes and k-ary n-cubes without extra channels, the turn model produces several new, partially adaptive routing algorithms. Simulations of these partially adaptive routing algorithms indicate that they can perform better than nonadaptive algorithms for nonuniform patterns of message traffic.

While the turn model has many advantages, it also has some disadvantages. Adaptive routing can require more complex control logic for route selection than does nonadaptive routing, and this may increase node delay. Part of the complexity is due to the need for a router to decide between multiple output channels, all of which lead to the destination. Another part of the complexity is due to the need for a router to base the route selection on more header information. For dimension-order routing, a router typically bases a selection on the distance remaining in one of the dimensions. For adaptive routing, a router must base a selection on the distance remaining in more than one, or all, dimensions. Every extra bit of header information that is required for the router to select an output channel increases router storage requirements and makes communication latencies more like those of store and forward.

There are many directions for future work. In [19], we investigate the effects of different input and output selection policies on network performance. In [18], we illustrate the application of the turn model to networks that include extra virtual or physical channels. Other models for designing adaptive routing algorithms are based on adding extra channels to networks, but do not produce routing algorithms that are maximally adaptive for the enhanced networks. Another obvious extension of our work is to apply the turn model to other topologies, such as hexagonal, octagonal, and cube-connected cycle networks, all of which permit adaptive routing without the addition of channels. In such topologies, the turns are not necessarily 90-degrees and the abstract cycles are not necessarily formed by four turns. A final important task is the identification of realistic workload distributions, so that the results of future simulations can be more meaningful.

Acknowledgments

The authors wish to thank Dr. Philip K. McKinley for helping us develop the simulator.

References

1. NCUBE Company, *NCUBE 6400 Processor Manual*, 1990.

2. Intel Corporation, *A Touchstone DELTA System Description*, 1991.

3. S. B. Borkar, R. Cohn, G. Cox, S. Gleason, T. Gross, H. T. Kung, M. Lam, B. Moore, C. Peterson, J. Pieper, L. Rankin, P. S. Tseng, J. Sutton, J. Urbanski, and J. Webb, "iWarp: An integrated solution to high-speed parallel computing," in *Proceedings of Supercomputing'88*, pp. 330–339, Nov. 1988.

4. D. Lenoski, J. Laudon, K. Gharachorloo, A. Gupta, and J. Hennessy, "The directory-based cache coherence protocol for the DASH multiprocessor," in *Proc. of the 17th International Symposium on Computer Architecture*, pp. 148–159, May 1990.

5. A. Agarwal, B.-H. Lim, D. Kranz, and J. Kubiatowicz, "APRIL: A processor architecture for multiprocessing multiprocessor," in *Proc. of the 17th International Symposium on Computer Architecture*, pp. 104–114, May 1990.

6. W. J. Dally and C. L. Seitz, "The torus routing chip," *Journal of Distributed Computing*, vol. 1, no. 3, pp. 187–196, 1986.

7. P. Kermani and L. Kleinrock, "Virtual cut-through: A new computer communication switching technique," *Computer Networks*, vol. 3, no. 4, pp. 267–286, 1979.

8. W. J. Dally, "Performance analysis of k-ary n-cube interconnection networks," *IEEE Transactions on Computers*, vol. C-39, pp. 775–785, June 1990.

9. X. Lin and L. M. Ni, "Deadlock-free multicast wormhole routing in in multicomputer networks," in *Proceedings of the 18th Annual International Symposium on Computer Architecture*, pp. 116–125, May 1991.

10. C. L. Seitz, W. C. Athas, C. M. Flaig, A. J. Martin, J. Seizovic, C. S. Steele, and W.-K. Su, "The architecture and programming of the Ametek Series 2010 multicomputer," in *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications, Volume I*, (Pasadena, CA), pp. 33–36, Association for Computing Machinery, Jan. 1988.

11. W. J. Dally, "The J-machine: System support for Actors," in *Actors: Knowledge-Based Concurrent Computing* (Hewitt and Agha, eds.), MIT Press, 1989.

12. W. J. Dally and H. Aoki, "Adaptive routing using virtual channels," tech. rep., Massachusetts Institute of Technology, Laboratory for Computer Science, Sept. 1990.

13. H. Sullivan and T. R. Bashkow, "A large scale, homogeneous, fully distributed parallel machine," in *Proceedings of the 4th Annu. Symp. Comput. Architecture*, vol. 5, pp. 105–124, Mar. 1977.

14. W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, pp. 547–553, May 1987.

15. J. T. Yantchev and C. R. Jesshope, "Adaptive, low latency, deadlock-free packet routing for networks of processors," in *IEE Proceedings, Pt. E*, vol. 136(3), pp. 178–186, May 1988.

16. D. H. Linder and J. C. Harden, "An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes," *IEEE Transactions on Computers*, vol. 40, pp. 2–12, Jan. 1991.

17. W. J. Dally, "Fine-grain message passing concurrent computers," in *Proc. of the Third Conference on Hypercube Concurrent Computers*, vol. 1, (Pasadena, CA.), pp. 2–12, Jan. 1988.

18. C. J. Glass and L. M. Ni, "Maximally fully adaptive routing in 2d meshes," Tech. Rep. MSU-CPS-ACS-51, Dept. of Computer Science, Michigan State University, East Lansing, Michigan, Jan. 1992.

19. C. J. Glass and L. M. Ni, "Adaptive routing in mesh-connected networks," in *Proceedings of the 12th International Conference on Distributed Computing Systems*, June 1992.

20. S. Konstantinidou, "Adaptive, minimal routing in hypercubes," in *Proc. of the 6th MIT Conference: Advanced Research in VLSI*, pp. 139–153, 1990.