

Optimal Fully Adaptive Minimal Wormhole Routing for Meshes¹

Loren Schwiebert and D. N. Jayasimha
Department of Computer and Information Science
The Ohio State University
Columbus, Ohio 43210–1277
Email: {loren, jayasim}@cis.ohio-state.edu

¹To appear in the Journal of Parallel and Distributed Computing

Abstract

A deadlock-free fully adaptive minimal routing algorithm for meshes that is optimal in the number of virtual channels required and in the number of restrictions placed on the use of these virtual channels is presented. It is also proved that, ignoring symmetry, this routing algorithm is the *only* fully adaptive routing algorithm with uniform routers that achieves both of these goals. This new algorithm requires only $4n - 2$ virtual channels for an n -dimensional mesh. In addition, if more than the minimum number of virtual channels is available, the routing algorithm can use these additional channels with the fewest possible number of restrictions. The proofs are first presented for the 2D mesh and then generalized to n -dimensional meshes.

Keywords: wormhole routing, mesh architectures, routing algorithms, deadlock freedom, optimality.

1 Introduction

Wormhole routing [6] has become the switching technique of choice in modern distributed-memory multiprocessors such as the Intel Paragon, the Cray T3D, the MIT J-machine, the Caltech MOSAIC, and the nCUBE-2. Implementations of wormhole routing typically divide each message into packets, which are then divided into flits. Since the network treats each packet as a separate message, the term message is used for a single packet. The header flits of a message contain the routing information and the data flits of the message follow the header flits through the network. When the header flits arrive at an intermediate router, the router immediately forwards the header to a neighboring router if an output channel the message can use is available. The data flits then follow the header flits in a pipelined fashion. If the header is unable to proceed because no appropriate output channel is free, the router buffers only a few flits, rather than the entire message. The data flits contain no routing information, so messages cannot share channels. Hence, each channel in the path is reserved from the time the header flits acquire the channel until the last flit of the message has traversed the channel. Since the flits of a message are forwarded as soon as possible, the message latency is largely insensitive to the distance between the source and destination. This is especially important for multiprocessors with a high diameter, such as a mesh. On the other hand, packet switching buffers the entire message at every intermediate node before forwarding any part of the message. Hence, wormhole routing has lower message latency when there is little or no channel contention. In addition, wormhole routing requires only enough storage to buffer a few flits, rather than the entire packet. These two factors account for the popularity of wormhole routing for distributed-memory multiprocessors. See Ni and McKinley [13] for an in-depth discussion of wormhole routing.

The primary drawback to wormhole routing is the contention that can occur even with moderate network traffic, which leads to higher message latency. Since all the channels in the path from the source to the destination are held from the time they are acquired until the entire message has traversed the channel (which is after the entire path has been established except for very short messages that fit in the intermediate channel buffers), performance degradation due to contention can be severe and message latency can be unacceptably high. A message that requires several channels can block many messages while being transmitted. These blocked messages can in turn block other messages, which further increases the message latency. Providing additional *physical* channels between nodes in the network can reduce both latency and contention. This is an expensive solution, however. A more cost-effective method of reducing message latency, proposed by Dally [4], is to allow multiple *virtual* channels to share the same physical channel. Each virtual channel has a separate buffer, with multiple messages multiplexed over the same physical channel. Both latency and contention can be further reduced by using the multiple paths that exist in the network between the source and destination nodes. Dally and Seitz [6], however, have shown that since a message holds channels until the entire message has been transmitted, a routing algorithm with no restrictions on the use of virtual or physical channels can result in deadlock.

The simplest routing algorithms are *deterministic* and define a single path between the source and destination. Adaptive routing algorithms, on the other hand, support multiple paths between the source and destination. Adaptive routing algorithms are either minimal or non-minimal. Minimal routing algorithms allow only shortest paths to be chosen, while non-minimal routing algorithms do not require messages to use only shortest paths. Minimal routing algorithms provide higher

throughput for high message traffic and are generally simpler to implement. Non-minimal routing algorithms are useful for fault tolerance. Whether minimal or non-minimal, adaptive routing algorithms can be further differentiated by the number of shortest paths allowed. *Partially adaptive* routing algorithms do not allow *all* messages to use *any* shortest path. *Fully adaptive* routing algorithms *do* allow all messages to use any shortest path. Although all fully adaptive routing algorithms allow a message to use any *physical* channel that is part of a shortest path, different restrictions are placed on the choice of *virtual* channels on that physical channel. Hence, not all fully adaptive routing algorithms are equivalent. Some fully adaptive routing algorithms allow more adaptiveness than others by placing fewer restrictions on the choice of *virtual* channels. Gaughan and Yalamanchili [8] present a good overview of adaptive routing protocols.

Since each virtual channel needs a separate buffer and the virtual channels are multiplexed across the physical channel, the number of virtual channels required by an adaptive routing algorithm gives a good approximation of the hardware cost of the router. Routing algorithms that require more virtual channels need additional router control logic and are usually more complex. Multiplexing and scheduling virtual channels on a physical channel is more complicated with additional virtual channels. Router latency and cycle time also increase with the number of virtual channels [1], so fewer virtual channels is better. Reducing the number of virtual channels needed for a given degree of adaptiveness is accomplished by using a less restrictive routing algorithm. Conversely, when the same number of virtual channels is used, a less restrictive routing algorithm has better performance than a more restrictive routing algorithm.

Recent research on adaptive routing algorithms has partially addressed both of these issues by reducing the virtual channel requirements and imposing fewer restrictions on the virtual channels. In this paper, theoretical results for fully adaptive routing on meshes are presented which demonstrate tight lower bounds on the number of virtual channels and the number of restrictions on the virtual channels. We prove these claims for fully adaptive minimal routing on meshes with uniform routers. Besides providing a theoretical framework for addressing these two issues, the routing algorithm proposed in this paper improves on previous routing algorithms both in the number of virtual channels and in the number of restrictions on the use of the virtual channels.

The following conventions are used throughout the paper. Channels are assumed to be bidirectional. All channels, whether physical or virtual, are referred to as virtual channels. N, S, E, and W are used to indicate the appropriate direction, with N–W used to indicate that a message has switched from the North direction to the West direction, for example. The symbol VC_{nd} denotes virtual channel n in the d direction. For example, VC_{1N} is virtual channel one in the North direction. The channel number is omitted when there is a single virtual channel in that direction.

2 Previous Work

Adaptive routing algorithms have been proposed for mesh, torus, and hypercube topologies. Torus and hypercube topologies can be characterized as k -ary n -cubes, where k is the radix and n is the dimension. For example, an 8D hypercube is a 2-ary 8-cube and a 16×16 torus is a 16-ary 2-cube. A mesh is a torus without the wrap-around channels. Routing algorithms for only mesh topologies are reviewed here, because the focus of this paper is on such topologies.

Many adaptive routing algorithms for meshes have been proposed [2, 3, 5, 9, 10, 11, 12]. Table I

summarizes the main features of each algorithm. In Table I, VCs is used as an abbreviation for number of bidirectional virtual channels per router.

Table I: Overview of Adaptive Routing Algorithms for Meshes

| Author(s) | Fully Adaptive? | VCs for 2D Mesh | Comments |
|----------------------------------|-----------------|-----------------|--|
| Chien & Kim [2] | Yes | 6 | Partially Adaptive for Higher Dimensions |
| Dally [3] | Yes | 6 | 2D Mesh Only |
| Dally & Aoki [5] | Yes | k | 2D Mesh with $k \times k$ nodes |
| Dally & Aoki [5] | Yes | 8 | 'Dynamic' routing algorithm |
| Glass & Ni [9] | Yes | 6 | 2D Mesh Only |
| Glass & Ni [10] | No | 4 | Roughly Half the Adaptiveness of Fully Adaptive |
| Jesshope, Miller & Yantchev [11] | Yes | 8 | Number of Virtual Channels is Exponential in Dimension of Mesh |
| Linder & Harden [12] | Yes | 6 | |

Designing deadlock-free routing algorithms for wormhole routing was simplified by Dally and Seitz with a proof that an acyclic channel dependency graph guarantees deadlock freedom [6]. Each node of the channel dependency graph is a virtual channel. There is a directed edge from one virtual channel to another if a message is permitted to use the second virtual channel immediately after the first. Since the graph is acyclic, deadlock freedom can be shown by assigning a numbering to the edges of the graph, ensuring that virtual channels are used in always increasing or always decreasing order. A routing algorithm is connected if there is a path from any source to any destination.

Dally and Seitz proposed their proof technique for deterministic routing algorithms. Deterministic routing algorithms can be characterized by functions of the form $R : C \times N \times N \rightarrow C$, where the input channel, belonging to the set of channels C , and the current node and destination node, belonging to the set of nodes N , define an output channel on which to route the message. An acyclic channel dependency graph has also been used as a basis for developing adaptive routing algorithms defined by relations of the form $R : C \times N \times N \rightarrow C^p$, where a set of output channels, rather than a single channel, is defined on which to route the message.

Duato [7] proved that requiring an acyclic channel dependency graph is too restrictive for routing algorithms defined by relations of the form $R : N \times N \rightarrow C^p$, where, independent of the input channel, the current node and the destination node define the set of channels on which the message can be routed. Cycles are permitted in the channel dependency graph if some subset of channels defines a connected routing subfunction with an acyclic *extended* channel dependency graph. An extended channel dependency graph contains both the direct and the indirect dependencies. Each edge in the channel dependency subgraph defines a *direct* dependency. An *indirect* dependency is a dependency between two channels in the subgraph that exists only because of the intermediate use of one or more channels not in the subgraph.

A method of analyzing routing algorithms based on the permitted and prohibited dependencies

from one virtual channel to another has been proposed by Glass and Ni [10]. These dependencies are characterized as turns. The set of possible turns is defined by the topology. For meshes, the turns can be 90° turns (when switching from a virtual channel in one dimension to a virtual channel in a different dimension), 0° turns (when switching from one virtual channel to another virtual channel in the same direction), and 180° turns (when switching from one virtual channel to a virtual channel in the opposite direction). Clearly, 180° turns are not used for minimal routing. The *turn model* groups the turns into cycles and breaks all cycles by prohibiting some turns. It is then necessary to show that the remaining turns cannot form any cycles. The turn model is an effective tool for describing routing algorithms and indicating the routing restrictions. It also provides a straightforward method of comparing the restrictions imposed by different routing algorithms.

For the 2D mesh, the routing algorithms proposed in [2, 3, 12] all produce an equivalent fully adaptive minimal routing algorithm called double-y. This routing algorithm requires two virtual channels in each Y direction (positive and negative) and one virtual channel in each X direction. This set of virtual channels has a total of sixteen 90° turns and four 0° turns. Double-y allows only eight of the sixteen 90° turns and prohibits all 0° turns.

Glass and Ni used the turn model to improve on double-y. They showed that double-y, although fully adaptive, imposes unneeded restrictions on the routing. They proposed the Maximally Fully Adaptive (mad-y) routing algorithm, which makes better use of the virtual channels and hence improves adaptiveness. It was also shown that a fully adaptive routing algorithm with fewer virtual channels or fewer routing restrictions is not possible without allowing cycles in the channel dependency graph. Thus, it was argued that mad-y is the least restrictive fully adaptive routing algorithm for 2D meshes. In the next section, we propose a routing algorithm that is more adaptive than mad-y, because our routing algorithm permits cycles in the channel dependency graph.

Dally and Aoki [5] have recently proposed two adaptive routing algorithms for meshes. The first, called static, labels each virtual channel on a physical channel with a dimension reversal number. Every time a message switches from a higher dimension to a lower dimension, the message must move to a virtual channel with a higher dimension reversal number. Once a message has reached a virtual channel with the highest dimension reversal number, the message must use dimension order routing. Fully adaptive routing on a $k \times k$ mesh requires as many as k dimension reversals, so k virtual channels per physical channel are required for fully adaptive routing using the static algorithm. The second algorithm, called dynamic, requires only two virtual channels per physical channel for fully adaptive routing. One of the virtual channels is used for adaptive routing and the other for dimension order routing. Once a message has used a virtual channel reserved for dimension order routing, the message is no longer permitted to use the adaptive channels. With two virtual channels per physical channel, there are a total of thirty-two 90° turns and eight 0° turns. The dynamic routing algorithm allows only twenty of the thirty-two 90° turns and only four of the eight 0° turns.

The partially adaptive routing algorithms proposed by Glass and Ni [10], although requiring no additional virtual channels, allow only about $1/2^{n-1}$ of the paths provided by a fully adaptive routing algorithm for an n -dimensional mesh. Three virtual channels per physical channel are used by Chien and Kim [2] to support partially adaptive routing for arbitrary dimension meshes. Jesshope, Miller, and Yantchev [11] propose a routing algorithm that is fully adaptive by dividing an n -dimensional mesh into 2^n regions and using separate virtual channels for each region. Fully adaptive routing on n -dimensional meshes requires 2^{n-1} subnetworks with $n + 1$ levels per subnetwork and

one virtual channel per level for each router, using the routing algorithm proposed by Linder and Harden [12]. Table II summarizes the virtual channel requirements per router for routing algorithms defined for n -dimensional meshes. The table includes the routing algorithm proposed in this paper, called the Optimal Fully Adaptive routing algorithm.

Table II: Comparison of the Number of Virtual Channels per Router

| Topology | Deterministic | Partially Adaptive | | Fully Adaptive | | | |
|------------|-----------------|--------------------|-------------|-----------------------------|------------------|--------------|------------------------|
| | Dimension Order | Glass & Ni | Chien & Kim | Jesshope, Miller & Yantchev | Linder & Harden | Dally & Aoki | Optimal Fully Adaptive |
| 2D Mesh | 4 | 4 | 6 | 8 | 6 | 8 | 6 |
| 3D Mesh | 6 | 6 | 12 | 24 | 16 | 12 | 10 |
| 4D Mesh | 8 | 8 | 18 | 64 | 40 | 16 | 14 |
| n D Mesh | $2n$ | $2n$ | $6n - 6$ | $n2^n$ | $(n + 1)2^{n-1}$ | $4n$ | $4n - 2$ |

Table II illustrates the dramatic reduction in the number of virtual channels required by the Optimal Fully Adaptive routing algorithm compared to the fully adaptive routing algorithms proposed in [11, 12]. The routing algorithm recently proposed by Dally and Aoki [5] requires two additional virtual channels and imposes substantially more routing restrictions than required. Note that fully adaptive routing is possible using about *two-thirds* the virtual channels needed for the partially adaptive routing algorithm proposed by Chien and Kim [2]. The only adaptive routing algorithms that require fewer virtual channels are the partially adaptive algorithms proposed by Glass and Ni [10]. With less than twice as many virtual channels, however, the Optimal Fully Adaptive routing algorithm allows an average of 2^{n-1} times as many paths.

3 Optimal Minimal Routing

Requiring an acyclic channel dependency graph places unnecessary restrictions on the routing algorithm. Since mad-y has this requirement, it is not the least restrictive routing algorithm for 2D meshes. A new routing algorithm, the Optimal Fully Adaptive routing algorithm, which has substantially fewer restrictions, is proposed. Since there are cycles in the channel dependency graph, the turn model cannot be used to prove deadlock freedom. An extended version of the turn model notation is used, however, to depict the routing restrictions. This new routing algorithm is first presented for the 2D mesh and then generalized to n -dimensional meshes.

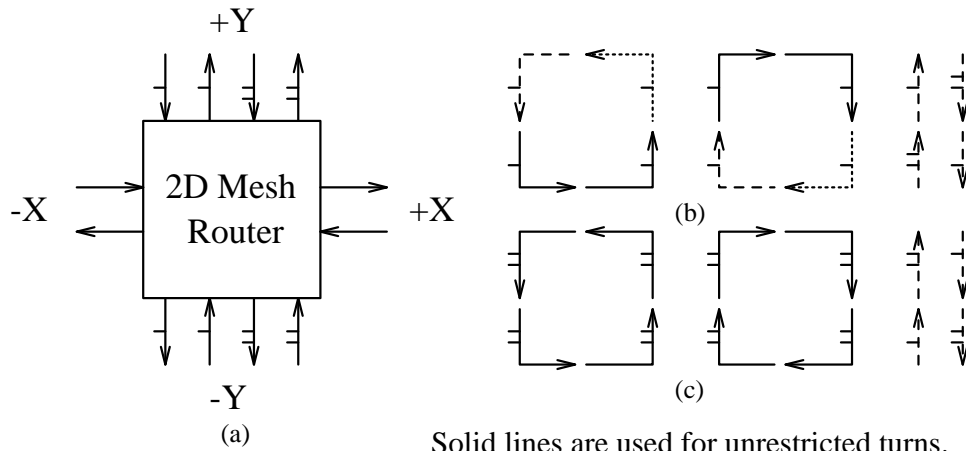
A block diagram of the router with two virtual channels in the North and South directions is shown in Figure 1(a). The virtual channels in the North and South directions are differentiated by marking VC_{1N} and VC_{1S} with a single dash and VC_{2N} and VC_{2S} with a double dash. This configuration has sixteen 90° turns and four 0° turns. *Unrestricted* turns are indicated by solid lines, *restricted* turns by dashed lines, and *prohibited* turns by dotted lines. The term *permitted* turns is used for the combination of restricted and unrestricted turns. The constraints imposed by the Optimal Fully Adaptive routing algorithm, referred to as opt-y, can be summarized succinctly: a mes-

sage that needs to route further in the West direction cannot use VC_{1N} or VC_{1S} . Opt-y has the following two sets of constraints (See Figures 1(b) and 1(c)):

- Two 90° turns, N–W using VC_{1N} and S–W using VC_{1S} , are prohibited.
- The 0° turns from VC_{2N} to VC_{1N} and from VC_{2S} to VC_{1S} are allowed only when the message does not need to route further West.

The following restrictions arise solely from the previous constraints:

- The 90° turns W–S using VC_{1S} and W–N using VC_{1N} are allowed only when the message does not need to route further West. These turns are restricted only because the N–W turn from VC_{1N} and S–W turn from VC_{1S} are prohibited.
- The 0° turns from VC_{1N} to VC_{2N} and from VC_{1S} to VC_{2S} are allowed only when the message does not need to route further West, because VC_{1N} and VC_{1S} are not used until the message has completed routing West.
- VC_{1N} and VC_{1S} cannot be used by the router at the source if the message needs to route West.



Solid lines are used for unrestricted turns.
Dashed lines are used for restricted turns.
Dotted lines are used for prohibited turns.

Figure 1: **Optimal Fully Adaptive Routing**

Several important properties are proved for opt-y:

- Opt-y is fully adaptive.
- Opt-y is deadlock-free.
- Opt-y requires only the minimum number of virtual channels.

- Opt-y imposes only the minimum number of routing restrictions on these virtual channels.
- Opt-y is the only algorithm to satisfy all of the previous properties (except for algorithms symmetric to opt-y).

Theorem 1 *Opt-y is fully adaptive.*

Proof. With minimal routing, a message always routes closer to the destination. When the source and destination differ in only one dimension, the message can use the virtual channel(s) in the appropriate direction; the single virtual channel in the East and West directions or either virtual channel in the North and South directions. For a destination that is Northeast (Southeast) of the source, the message can route adaptively along the North (South) and East virtual channels. For a destination that is Northwest (Southwest) of the source, the message can route adaptively along VC_{2N} (VC_{2S}) and the West virtual channel. Once the message has completed routing West, it can use either of the North (South) virtual channels and switch between them. \square

A message can use VC_{1N} or VC_{1S} only when the destination is not West of the current node. Otherwise, it must choose one of the other virtual channels. This restriction is independent of the input channel, so opt-y has a routing relation of the form $R : N \times N \rightarrow C^p$. Therefore, Duato's result [7] can be used to show that opt-y is deadlock-free.

Using Duato's terminology [7], the routing algorithm is denoted R and the set of channels used by R is denoted C . There is some subset of channels, $C_1 \subseteq C$, that defines a routing algorithm $R_1 \subseteq R$, i.e., R_1 is the routing algorithm R restricted to using the set of channels C_1 . When there are cycles in the channel dependency graph, it is not possible to provide a numbering of the channels that guarantees the channels are used in always increasing or always decreasing order. Instead, Duato has shown [7] that any routing algorithm of the form $R : N \times N \rightarrow C^p$ is deadlock-free if it satisfies the following three conditions:

- Routing algorithm R_1 is connected.
- Routing algorithm R_1 is acyclic and therefore deadlock-free.
- The additional channels ($C - C_1$) of routing algorithm R do not introduce cycles in the extended channel dependency graph of C_1 .

The idea behind the proof is that cycles are permitted in the channel dependency graph if every message always has the possibility of switching to an acyclic path. The set of output channels, C^p , always includes at least one channel in C_1 , so a deadlock-free path is always available. In order for deadlock to be avoided, header flits cannot enter a virtual channel queue until the queue is empty. Whenever all channels in C^p are busy, the output selection policy must either defer selecting a channel or choose a busy channel in $C_1 \cap C^p$. For opt-y, C_1 consists of all the virtual channels except VC_{2N} and VC_{2S} . Consequently, R_1 uses all the permitted 90° turns shown in Figure 1(b). R_1 is the West-First routing algorithm by Glass and Ni [10].

Lemma 1 *Routing algorithm R_1 is connected.*

Proof. The only restriction imposed by R_1 (other than minimal routing) is that a message cannot use a West channel after using a North or South channel. Therefore, XY routing is a subset of R_1 and the lemma follows immediately from the fact that XY routing is connected. \square

Lemma 2 *Routing algorithm R_1 is acyclic and therefore deadlock-free.*

Proof. Messages must route West before routing North or South. Since there are no dependencies from a channel in the Y dimension to a channel in the West direction, R_1 is acyclic. \square

Lemma 3 *The additional channels of routing algorithm R do not introduce cycles in the extended channel dependency graph of C_1 .*

Proof. The only possibility for a cycle in C_1 is for an *indirect* dependency to be introduced by using VC_{2N} or VC_{2S} . First notice that R_1 is fully adaptive for all messages except messages with a destination West of the source. Indirect dependencies that allow a use of the channels different from R_1 could arise in only two ways: (1) A message uses VC_W after using VC_{1N} or VC_{1S} . This cannot occur, since the routing algorithm, R , prohibits the use of VC_{1N} or VC_{1S} by any message that needs to route further West. (2) A message uses some VC_W and later uses another VC_W in a different row of the mesh. This is possible only because a message that is routed West can use VC_{2N} or VC_{2S} and later route West again. However, this indirect dependency does not cause a deadlock. The channel dependency graph for C_1 has no dependencies from a channel in the East, North, or South directions to a channel in the West direction, so the West channels are always used before any other channel in C_1 . Hence, these indirect dependencies create new dependencies among only the West virtual channels. Since a mesh has no wrap-around channels and minimal routing is used, there are no cycles using only the West virtual channels. Therefore, the extended channel dependency graph of C_1 is acyclic. \square

Theorem 2 *Opt-y is deadlock-free.*

Proof. The proof follows immediately by lemmas 1 – 3. \square

4 Proofs of Optimality

Opt-y is optimal for fully adaptive minimal routing in two ways: (1) It requires only the minimum number of virtual channels per router. (2) It imposes only the minimum number of routing restrictions on the virtual channels. Two reasonable and standard assumptions are made in connection with proving the optimality of opt-y. First, optimality holds for the number of virtual channels at only the interior nodes of the mesh. Nodes on the edges of the mesh need to be connected to other nodes in only two or three directions, so fewer virtual channels could be used for these routers. Second, all the nodes use identical routers and thus the same routing algorithm.

The optimality proofs are greatly simplified by proving that *all* configurations with fewer than six virtual channels and *most* configurations with six virtual channels are either not deadlock-free or not fully adaptive. (This was proved by Glass and Ni [9] only for routing algorithms with an acyclic channel dependency graph.)

Note: The following conventions are used for figures showing deadlock configurations. In each figure, the routers are represented as circles and the channels as edges connecting the routers. The source and destination of message i are labeled S_i and D_i , respectively. Virtual channels acquired by message i are labeled M_i . Bidirectional channels are assumed, so arrows are used to indicate the

direction of each message. Messages on the left side of a vertical channel use VC_1 and messages on the right side use VC_2 . Similarly, messages above a horizontal channel use VC_1 and messages below use VC_2 .

Theorem 3 *Deadlock-free fully adaptive wormhole routing on a 2D mesh requires a second virtual channel in two opposite directions, either North and South or East and West.*

Proof. Consider the potential deadlock configuration shown in Figure 2(a). All four messages in the cycle are waiting for a virtual channel in the North or West direction. If there is only a single virtual channel in the North and West directions, then the routing algorithm is not deadlock-free. This applies to *any* fully adaptive minimal routing algorithm. Similarly, Figure 2(b) shows a deadlock configuration if there is a single virtual channel in the North and East directions. Figure 2(c) shows a deadlock configuration if there is a single virtual channel in the South and West directions. Figure 2(d) shows a deadlock configuration if there is a single virtual channel in the South and East directions. At least two more virtual channels are required to resolve all four deadlock cases, and the two virtual channels must be added in opposite directions. For example, the top two deadlock configurations can be avoided by adding a virtual channel in the North direction and the bottom two deadlock configurations can be avoided by adding a virtual channel in the South direction. \square

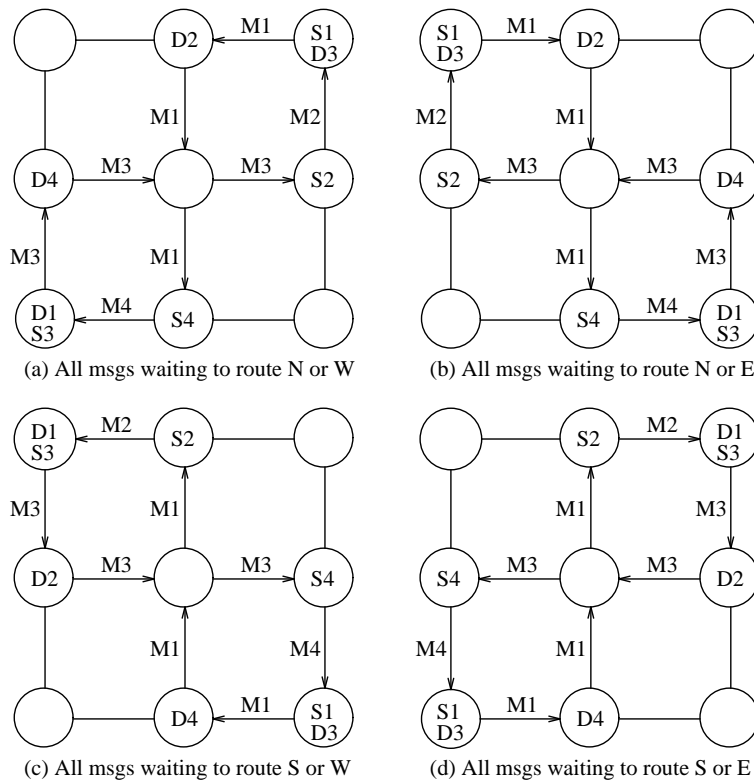


Figure 2: Potential Deadlock Configurations

By theorem 3, opt-y requires only the minimum number of virtual channels for deadlock-free fully adaptive routing. The remainder of this section is used to show that opt-y has the fewest restric-

tions of *any* fully adaptive routing algorithm with six virtual channels, and that, ignoring symmetry, *all* other fully adaptive routing algorithms have more restrictions than opt-y.

A cycle in the channel dependency graph is not a sufficient condition for deadlock in an adaptive routing algorithm. However, a *knot* is a sufficient condition for deadlock, so the proofs of deadlock are given based on the existence of knots (rather than cycles) in the channel dependency graph. A knot exists when a set of channels forms a cycle and no channel in the set has a path to a channel outside the set. In other words, a knot is a cycle with the additional requirement that *all* channels in the knot have paths to only channels in the knot. This set of channels is determined by the routing algorithm and the destination of the message currently using the channel.

Lemma 4 *At least two 90° turns must be prohibited to make the routing algorithm deadlock-free.*

Proof. Knots can occur when a set of messages routes in the clockwise or counter-clockwise directions, so it is necessary to prevent both of these knots. At least one 90° turn must be prohibited to prevent each knot, so at least two 90° turns must be prohibited. \square

Since at least two 90° turns must be prohibited by lemma 4 and opt-y prohibits only two, the only other routing algorithms that need to be considered are those that prohibit only two 90° turns. Ignoring symmetry, there are only three such routing algorithms: North-Last, Negative-First, and West-First [10]. In addition, by theorem 3, only configurations with a second virtual channel in either the North and South or the East and West directions need to be considered. Therefore, there are six different routing algorithms to consider. Opt-y, which uses West-First, is deadlock-free. A deadlock configuration is shown for each of the five remaining possibilities. The deadlock configurations occur whether or not 0° turns are permitted. **Note:** In some of the examples, deadlock can be avoided by prohibiting additional 90° turns. Such modifications are irrelevant, since the goal is to minimize the number of restrictions.

Lemma 5 *Opt-y is not deadlock-free if the routing restrictions shown in Figure 1 are still applied to C_1 , but the virtual channels are added in the East and West directions.*

Proof. The routing algorithm is fully adaptive only if the 90° turns N–W and S–W using VC_{2W} are unrestricted. A deadlock configuration is shown in Figure 3. \square

Lemma 6 *The North-Last routing algorithm is not deadlock-free with only six virtual channels and two prohibited 90° turns.*

Proof. The turn model representation of the North-Last routing algorithm with the second virtual channel in the East and West directions is shown in Figure 4(a). Only the 90° turns are shown, since the 0° turns are not used for showing deadlock. The deadlock configuration for this routing algorithm is shown in Figure 4(b). The turn model representation with the second virtual channel in the North and South directions is shown in Figure 5(a). The deadlock configuration for this routing algorithm is shown in Figure 5(b). \square

Lemma 7 *The Negative-First routing algorithm is not deadlock-free with only six virtual channels and two prohibited 90° turns.*

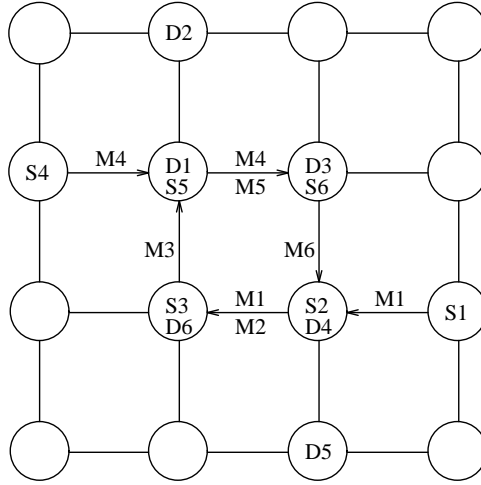


Figure 3: **West-First with E and W Virtual Channels**

Proof. The turn model representation of the Negative-First routing algorithm with the second virtual channel in the East and West directions is shown in Figure 6(a). Again, only the 90° turns are shown, since the 0° turns are not used for showing deadlock. The deadlock configuration for this routing algorithm is shown in Figure 6(b). The turn model representation with the second virtual channel in the North and South directions is shown in Figure 7(a). The deadlock configuration for this routing algorithm is shown in Figure 7(b). \square

The only remaining possibility is to divide the prohibited turns between the virtual channels so that one is applied to VC_1 and the other to VC_2 . However, the deadlock configurations shown in Figures 3 – 7 are still possible with at most minor modifications. This leads to theorem 4, which is proved for *symmetric* routing algorithms. Symmetry is preserved if the virtual channels are added in the X dimension with a corresponding set of restrictions and/or the restrictions are applied to VC_2 instead of VC_1 .

Theorem 4 *Ignoring symmetry, opt-y prohibits fewer 90° turns than any other deadlock-free fully adaptive routing algorithm that uses only six virtual channels.*

Proof. By lemma 4, at least two 90° turns must be prohibited. Ignoring symmetry, there are only three such routing algorithms [10]. Opt-y uses one of these. However, by lemma 5, it is not deadlock-free if the virtual channels are added in the East and West directions instead of the North and South directions. The other possibilities, North-Last and Negative-First, are both not deadlock-free (by lemma 6 and lemma 7, respectively). Since there are no other possibilities to consider, opt-y is the only deadlock-free fully adaptive routing algorithm that requires only six virtual channels and prohibits only two 90° turns. \square

Theorem 5 *No routing algorithm using only six virtual channels per router can have fewer restrictions than those imposed by opt-y.*

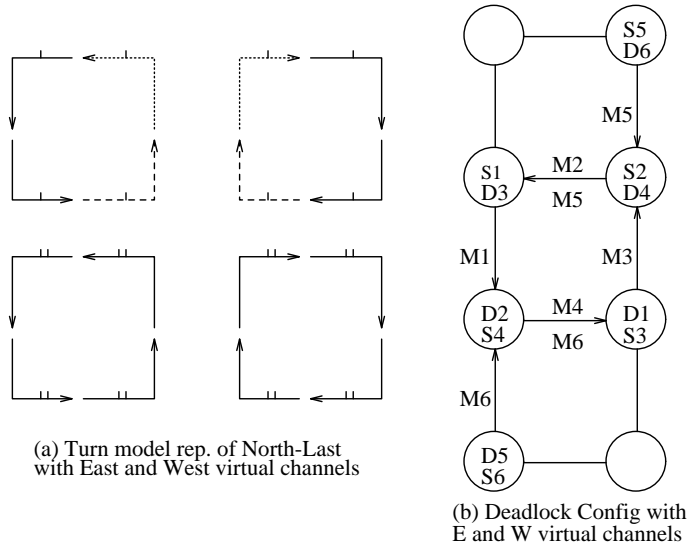


Figure 4: **North-Last with E and W Virtual Channels**

Proof. By lemma 4 and theorem 4, it is clear that only the 0° turns need to be considered. Opt-y restricts the use of the 0° turns from VC_{2N} to VC_{1N} and from VC_{2S} to VC_{1S} to messages that do not need to route further West. Figure 8 gives an example where deadlock occurs when the restriction on the 0° turn from VC_{2N} to VC_{1N} is removed. A similar deadlock configuration can be constructed if the restriction on the 0° turn from VC_{2S} to VC_{1S} is removed. Since the restrictions on the other two 0° turns are a direct consequence of the prohibited 90° turns, opt-y cannot be made more adaptive. \square

5 Comparison

Opt-y is compared with mad-y, the least restrictive of any previously proposed routing algorithm. The comparison is limited to minimal routing. The virtual channel requirements are the same for both routing algorithms. The turn model representation of mad-y is shown in Figure 9. Mad-y has the following two sets of constraints:

- Four 90° turns, N–W using VC_{2N} , S–W using VC_{2S} , E–N using VC_{1N} , and E–S using VC_{1S} , are prohibited.
- The 0° turns from VC_{2N} to VC_{1N} and from VC_{2S} to VC_{1S} are prohibited.

The following four restrictions arise solely from the previous constraints:

- The 0° turns from VC_{1N} to VC_{2N} and from VC_{1S} to VC_{2S} are allowed only when a message does not need to route further West and has not already routed East. The West-bound restriction exists because a message cannot route West from VC_{2N} or VC_{2S} . The East-bound restriction exists because a message never uses VC_{1N} or VC_{1S} after routing East.

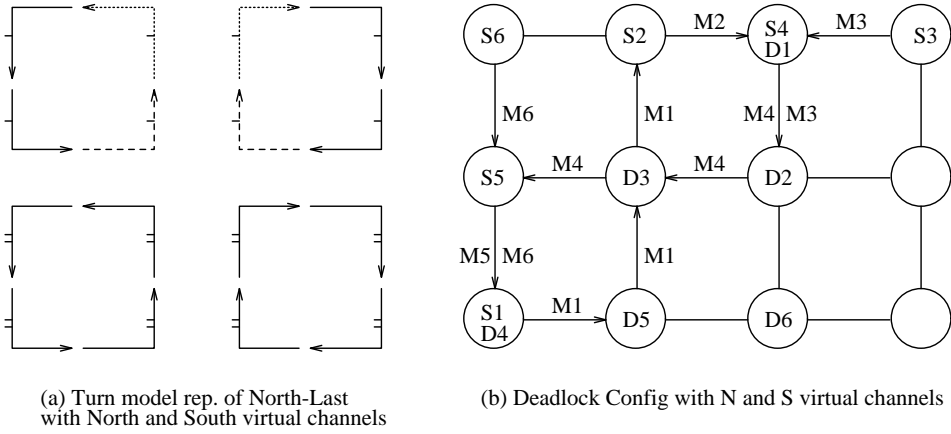


Figure 5: **North-Last with N and S Virtual Channels**

- Two 90° turns, N–E using VC_{1N} and S–E using VC_{1S} , are allowed only when the message has not routed East. These turns are restricted only because a message cannot use VC_{1N} or VC_{1S} after VC_E .
- Two 90° turns, W–N using VC_{2N} and W–S using VC_{2S} , are allowed only when the message is not routed further West. These turns are restricted only because the turns from VC_{2N} or VC_{2S} to VC_W are prohibited.
- VC_{2N} and VC_{2S} cannot be used by the router at the source if the message needs to route West.

Mad-y allows four more 90° turns than double-y, but a message can use these additional turns at most once. Additionally, a message can make at most one 0° turn, since a message cannot use VC_{1N} or VC_{1S} after switching to VC_{2N} or VC_{2S} . Finally, a message cannot make both a restricted 90° turn and a 0° turn. A message that makes a restricted W–N or W–S turn is no longer in VC_{1N} or VC_{1S} , so it cannot make a 0° turn. Similarly, a message that makes a restricted N–E or S–E turn never uses VC_{1N} or VC_{1S} again, so it cannot make a 0° turn. A message that makes a 0° turn cannot make a restricted turn to the East, because the message never uses VC_{1N} or VC_{1S} again. Similarly, a message can make a 0° turn only after routing completely in the West direction, so it cannot later make a restricted turn from the West direction.

Opt-y prohibits only two 90° turns and restricts only two 90° turns. Mad-y prohibits four 90° turns and restricts four 90° turns. Mad-y also prohibits half the 0° turns and restricts the other half. For opt-y, all the 0° turns are restricted, so none of the 0° turns are prohibited.

Opt-y also makes much better use of the restricted turns. The 0° turns are restricted to messages that no longer need to route West, but messages can switch from either virtual channel to the other. This allows a message to make a 0° turn *more than once*. A message also has the possibility of making a restricted 90° turn *and* some 0° turns.

Not only does opt-y have fewer restrictions, but the router control logic is also simpler. Mad-y has a routing relation of the form $R : C \times N \times N \rightarrow C^p$, so the set of output channels from which

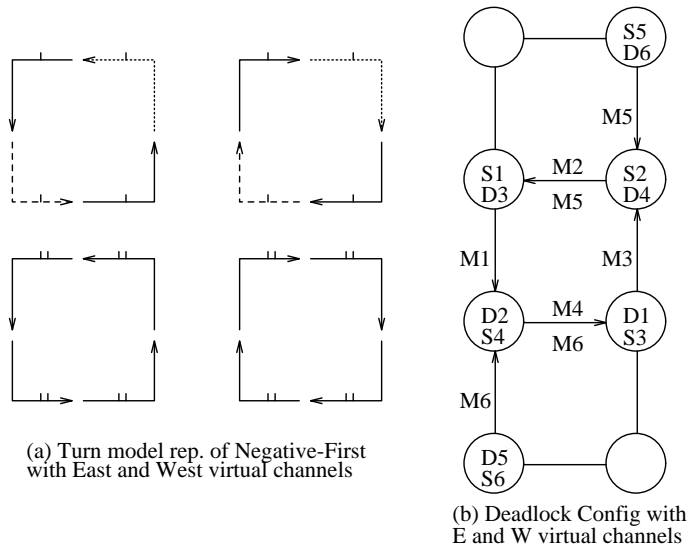


Figure 6: **Negative-First with E and W Virtual Channels**

a message can choose depends not only on the current node and destination but also on the input channel. Opt-y does not differentiate based on the input channel, because the routing relation is of the form $R : N \times N \rightarrow C^p$. This simplifies the hardware required to choose the output channel on which to route the message, since the choice is independent of the input channel.

6 Generalization

The Optimal Fully Adaptive routing algorithm can be extended to n -dimensional meshes in a straightforward manner. Minimal routing is still assumed. The routing algorithm needs only the minimum number of virtual channels and imposes only the minimum number of routing restrictions needed to avoid deadlock. Two different generalizations from the 2D mesh are provided. The first is proved using Duato's proof technique and is used to prove the lower bound on the number of virtual channels. However, a slightly different generalization is required to prove the minimum number of routing restrictions. The routing algorithm is first generalized using the following steps:

- Assign a channel to both directions of each dimension.
- Number the dimensions in some order and add a second virtual channel to both directions of all dimensions except the first dimension.
- Allow a message to route along the second virtual channel at any time.
- For each dimension except the last, select one of the two directions as the chosen direction of that dimension. Prohibit a message from routing on the first virtual channel of any direction until it has completed routing in the chosen direction of *all* lower dimensions.

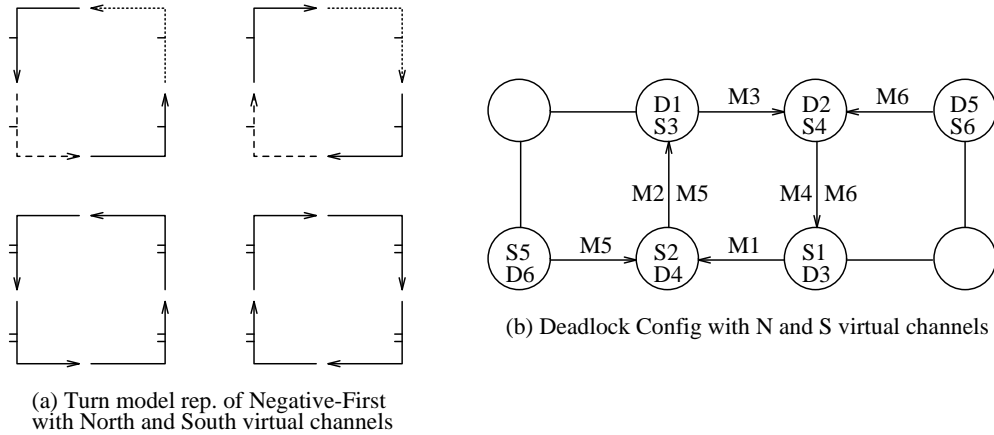


Figure 7: **Negative-First with N and S Virtual Channels**

- Allow a message to make a 0° turn between the two virtual channels of a direction only after the message has completed routing in the chosen direction of *all* lower dimensions.

For example, consider a 4D mesh with dimensions $XYZW$ and number the dimensions 1, 2, 3, and 4, respectively. The positive and negative directions in the Z dimension are called Up and Down, respectively. The positive and negative directions in the W dimension are called In and Out, respectively. Let the negative directions in the X , Y and Z dimensions be the chosen directions. A second virtual channel is added in both directions of the Y , Z and W dimensions. A message can route adaptively in any dimension using virtual channel one, once it has completed routing in the chosen direction of all lower dimensions. For example, a message can use VC_{1N} and VC_{1S} only after it has completed routing West. Similarly, a message can use VC_{1U} and VC_{1D} only after it has completed routing West and South. A message can use VC_{1I} and VC_{1O} only after it has completed routing West, South, and Down. A message can route, without any restrictions, in the X dimension and in any direction using virtual channel two.

A block diagram of the YZ plane of an n -dimensional router is shown in Figure 10(a). The virtual channels are differentiated by marking the first virtual channel in each direction with a single dash and the second virtual channel with a double dash. This configuration has thirty-two 90° turns and eight 0° turns. As before, *unrestricted* turns are indicated by solid lines, *restricted* turns by dashed lines, and *prohibited* turns by dotted lines. For simplicity, only 90° turns are shown, since all 0° turns are restricted. The YZ plane of the Optimal Fully Adaptive routing algorithm has the following two sets of constraints (See Figure 10(b)):

- Four of the thirty-two 90° turns, namely U–S from VC_{1U} to VC_{1S} or VC_{2S} and D–S from VC_{1D} to VC_{1S} or VC_{2S} , are prohibited.
- The 0° turns from VC_{2U} to VC_{1U} and from VC_{2D} to VC_{1D} are not allowed when the message needs to route further West or South.

The following restrictions arise solely from the previous constraints:

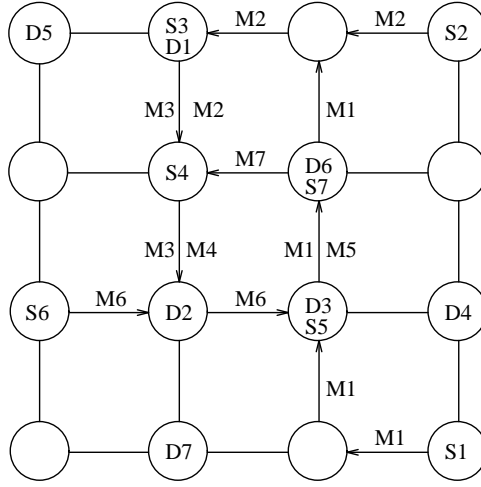


Figure 8: **Deadlock due to Fewer Restrictions**

- The four 90° turns S–U from VC_{1S} or VC_{2S} to VC_{1U} and S–D from VC_{1S} or VC_{2S} to VC_{1D} are not allowed when the message needs to route further West or South. These turns are restricted only because the U–S turns from VC_{1U} and D–S turns from VC_{1D} are prohibited.
- The 0° turns from VC_{1U} to VC_{2U} and from VC_{1D} to VC_{2D} are not allowed when the message needs to route further West or South, because VC_{1U} and VC_{1D} are not used until the message has completed routing West and South.
- VC_{1U} and VC_{1D} cannot be used by the router at the source if the message needs to route West or South.

For an n -dimensional mesh, the dynamic routing algorithm proposed by Dally and Aoki [5] imposes considerably more routing restrictions than necessary. For each 2D plane, the Optimal Fully Adaptive routing algorithm prohibits only four of the thirty-two 90° turns and none of the eight 0° turns, compared to the twelve 90° turns and four 0° turns prohibited by the dynamic routing algorithm.

Theorem 6 *The Optimal Fully Adaptive routing algorithm is fully adaptive for n -dimensional meshes.*

Proof. Messages can use the second virtual channel at any time. Messages can also route in either direction of the lowest dimension at any time, since there are no restrictions on using the virtual channels in the lowest dimension. This allows the message to route in any direction that moves the message closer to the destination. \square

Since the use of the first virtual channels depends on in which of the chosen directions the message still needs to route, the routing relation is of the form $R : N \times N \rightarrow C^p$. The proof that the routing algorithm is deadlock-free relies on Duato’s proof of a sufficient condition for deadlock freedom [7]. C_1 consists of only virtual channel one in each direction, so R_1 uses only the permitted 90° turns among the first virtual channels.

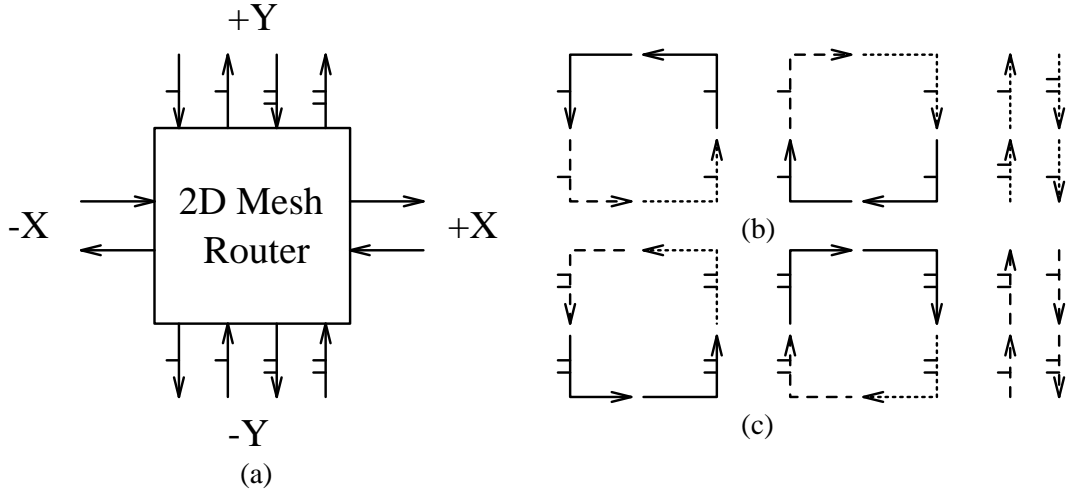


Figure 9: **Maximally Fully Adaptive Routing**

Lemma 8 *Routing algorithm R_1 is connected.*

Proof. A message can reach any destination from any source. R_1 prevents only a channel in a higher dimension from using a channel in a lower chosen direction, hence dimension order routing is a subset of R_1 . The lemma follows immediately from the fact that dimension order routing is connected. \square

Lemma 9 *Routing algorithm R_1 is acyclic and therefore deadlock-free.*

Proof. Since 180° turns are not permitted for minimal routing, any cycle requires that there be dependencies across multiple dimensions. In addition, *both* directions of each dimension in the cycle must be used to form the cycle. For a cycle to exist, there must be a dependency from some higher dimension(s) of the cycle to both directions in the lowest dimension of the cycle. However, R_1 requires that the chosen direction of each dimension be used before any direction of a higher dimension, so no cycle is possible. \square

Lemma 10 *The additional channels of routing algorithm R do not introduce cycles in the extended channel dependency graph of C_1 .*

Proof. The only possibility for a cycle in C_1 is for an *indirect* dependency to be introduced by using one or more of the second virtual channels. First, note that R_1 prevents only a channel in a higher dimension from using a channel in a lower chosen direction. Indirect dependencies that allow a use of the channels different from R_1 could arise in only three ways: (1) A message uses virtual channel one in a chosen direction after using virtual channel one in a higher dimension. This cannot occur, since the routing algorithm, R , explicitly prohibits the use of the virtual channels in this order. (2) A message uses virtual channel one in one chosen direction and later uses virtual channel one in another chosen direction. This is a special case of (1) if the first chosen direction is in a higher dimension than the second chosen direction. The routing algorithm does not allow this possibility. If

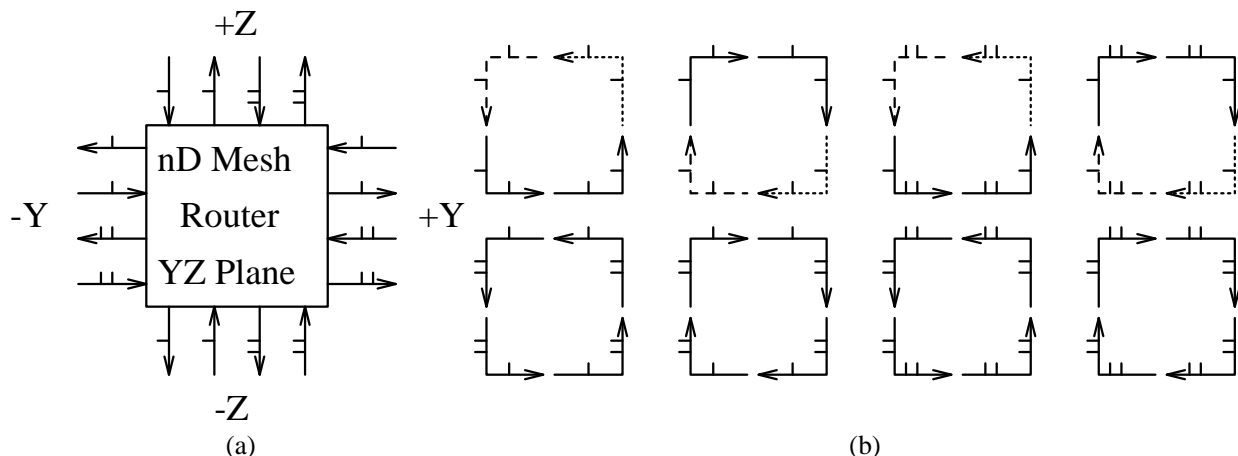


Figure 10: **YZ Plane of an n -dimensional Router**

the first chosen direction is in a lower dimension than the second chosen direction, then this possibility does not cause a problem, since this is allowed by R_1 . (Note that this turn is allowed by R_1 and R only when the message has completed routing in the lower of the two dimensions.) (3) A message uses virtual channel one in a chosen direction, routes in another dimension, and then uses virtual channel one in the same chosen direction. Routing algorithm R allows this possibility, but it does not cause a deadlock. The channel dependency graph for C_1 has no dependencies from a channel in a higher dimension to the channel in the chosen direction of a lower dimension. There are also no dependencies from this channel to the channel in any chosen direction in a lower dimension. Therefore, these indirect dependencies create new dependencies only among channels in this chosen direction and channels in unchosen directions of lower dimensions. However, these dependencies do not create a cycle. Since a mesh has no wrap-around channels and minimal routing is used, there are no cycles among channels in the same chosen direction and R_1 already allows the use of a channel in an unchosen direction after the use of a channel in a higher chosen direction. Hence the extended channel dependency graph of C_1 is acyclic. \square

Theorem 7 *The Optimal Fully Adaptive routing algorithm is deadlock-free for n -dimensional meshes.*

Proof. The proof follows immediately by lemmas 8 – 10. \square

Theorem 8 *The Optimal Fully Adaptive routing algorithm uses the minimum number of virtual channels for deadlock-free fully adaptive wormhole routing on a mesh.*

Proof. Assume that a virtual channel is removed. Removing the virtual channel in either direction of the lowest dimension makes the mesh disconnected. If the virtual channel is removed from one of the other dimensions, then there are only five virtual channels in the plane formed by this dimension and the lowest dimension. It is possible to generate a set of messages that need to route in only these two dimensions. By theorem 3 it is known that this configuration is not both fully adaptive and deadlock-free. \square

By theorem 8, the Optimal Fully Adaptive routing algorithm requires only the minimum number of virtual channels for deadlock-free fully adaptive routing. The remaining proofs are used to show that the Optimal Fully Adaptive routing algorithm has the fewest restrictions of any fully adaptive routing algorithm with $4n - 2$ virtual channels, and that, ignoring symmetry, all other fully adaptive routing algorithms have more restrictions. The 2D case has been used to prove optimality when there are six virtual channels between the two dimensions, which occurs with the lowest dimension and any other dimension. There are eight virtual channels, two virtual channels per physical channel, for every pair of dimensions, except when the lowest dimension is one of the two dimensions. The following lemmas are used to prove that the routing algorithm is optimal with two virtual channels in all four directions.

Lemma 11 *At least four 90° turns must be prohibited to make a routing algorithm deadlock-free when two virtual channels are used for each physical channel.*

Proof. Figure 11 shows a deadlock configuration that arises when no 90° turns are prohibited. A similar deadlock configuration can be constructed for a set of messages routing in the clockwise direction. The only way to prevent each deadlock configuration is to prohibit at least one *channel* from being used before *any* channel in at least one other direction. Otherwise, the deadlock configuration shown in Figure 11 cannot be avoided. Each message could still use the virtual channel(s) shown in Figure 11 and wait for at least one of the two channels in the other direction. There are two virtual channels in each direction, so at least two 90° turns must be prohibited for each of the two orientations (CW and CCW). Therefore, a total of at least four 90° turns must be prohibited. \square

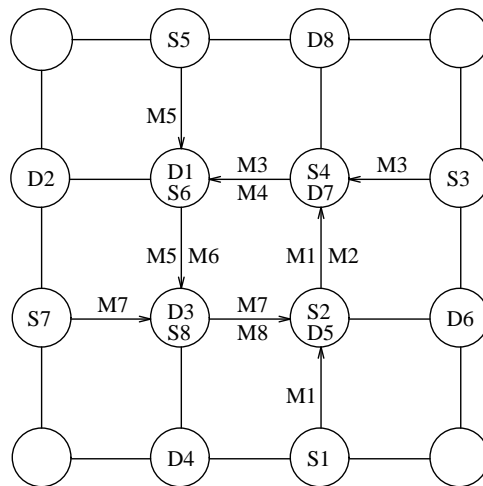


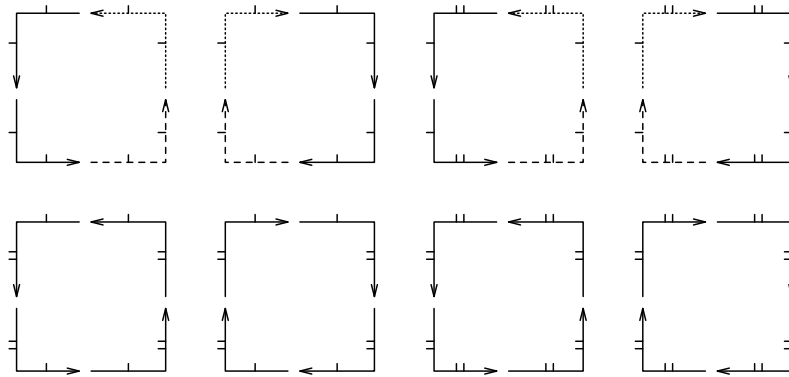
Figure 11: **Deadlock with 2 VCs per Physical Channel**

By lemma 11, at least four 90° turns must be prohibited between each pair of dimensions. Also, the prohibited turns must be from one virtual channel to either virtual channel in another direction. The Optimal Fully Adaptive routing algorithm, which uses an extension of West-First, prohibits exactly four 90° turns. Hence, ignoring symmetry, the only possibilities to consider are extensions of the North-Last and Negative-First routing algorithms. A deadlock configuration is shown for

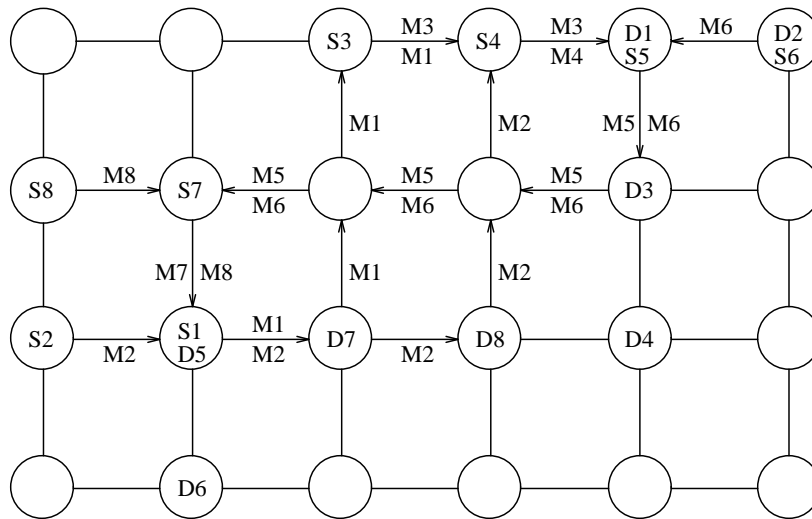
these two possibilities when only four 90° turns are prohibited. Both deadlock configurations occur whether or not 0° turns are permitted.

Lemma 12 *The North-Last routing algorithm is not deadlock-free with two virtual channels per physical channel and four prohibited 90° turns.*

Proof. The turn model representation of the North-Last routing algorithm with two virtual channels per physical channel is shown in Figure 12(a). Only the 90° turns are shown, since deadlock occurs with or without 0° turns. A deadlock configuration is shown in Figure 12(b). \square



(a) Turn model rep. of North-Last with two virtual channels per physical channel

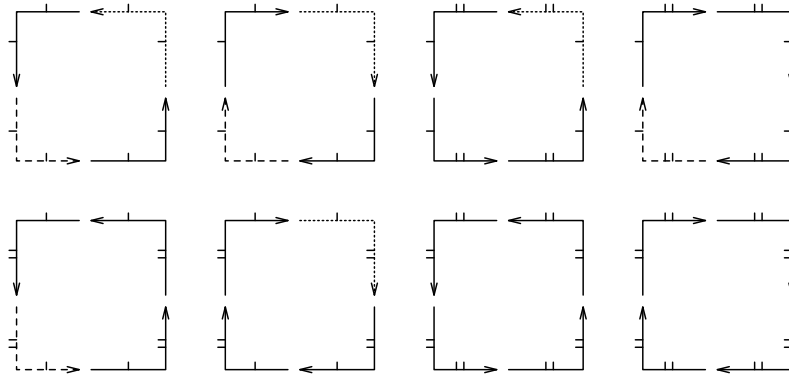


(b) Deadlock Config with two virtual channels per physical channel

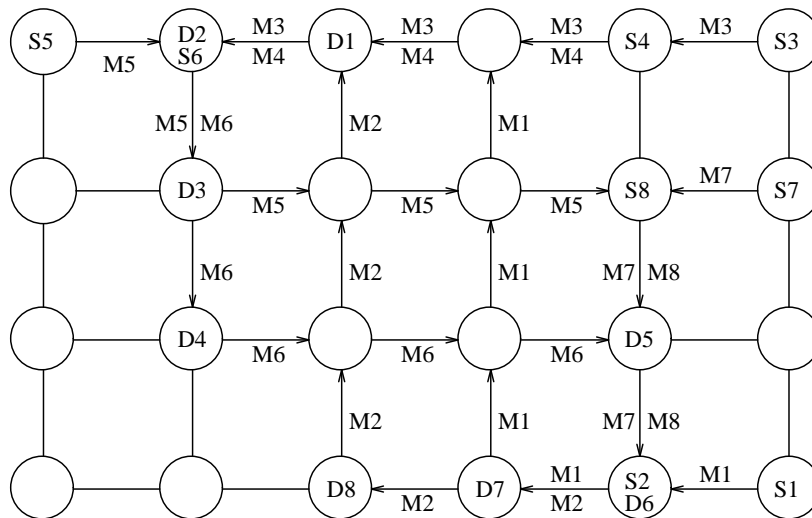
Figure 12: North-Last with 2 VCs per Physical Channel

Lemma 13 *The Negative-First routing algorithm is not deadlock-free with two virtual channels per physical channel and four prohibited 90° turns.*

Proof. The turn model representation of the Negative-First routing algorithm with two virtual channels per physical channel is shown in Figure 13(a). Only the 90° turns are shown, since deadlock occurs with or without 0° turns. A deadlock configuration is shown in Figure 13(b). \square



(a) Turn model rep. of Negative-First with two virtual channels per physical channel



(b) Deadlock Config with two virtual channels per physical channel

Figure 13: **Negative-First with 2 VCs per Physical Channel**

Theorem 9 *Ignoring symmetry, the Optimal Fully Adaptive routing algorithm prohibits fewer 90° turns than any other fully adaptive deadlock-free routing algorithm with two virtual channels per physical channel.*

Proof. By lemma 11, at least four 90° turns must be prohibited. Ignoring symmetry, there are only three routing algorithms to consider. The Optimal Fully Adaptive routing algorithm is one of these. By lemma 12 and lemma 13, the other two possibilities are not deadlock-free. Therefore, the Optimal Fully Adaptive routing algorithm is the only deadlock-free fully adaptive routing algorithm with two virtual channels per physical channel which prohibits only four 90° turns. \square

Theorem 10 *No routing algorithm using two virtual channels per physical channel can have fewer restrictions than those imposed by the Optimal Fully Adaptive routing algorithm.*

Proof. By lemma 11 and theorem 9, it is clear that only the 0° turns need to be considered. The Optimal Fully Adaptive routing algorithm restricts the 0° turns from the second virtual channel to the first virtual channel in the higher dimension to messages that do not need to route further in the chosen direction of the lower dimension. Figure 8 is easily modified to give an example where deadlock occurs when the restriction on the 0° turn in the positive direction of the higher dimension is removed. A similar deadlock configuration can be constructed if the restriction on the 0° turn in the negative direction is removed. Since the restrictions on the other two 0° turns (from the first virtual channel to the second) are a direct consequence of the prohibited 90° turns, the Optimal Fully Adaptive routing algorithm cannot be made more adaptive. \square

The Optimal Fully Adaptive routing algorithm proposed at the beginning of section 6 requires only the minimum number of restrictions for messages that route in any two dimensions. For messages that traverse more than two dimensions the routing restrictions can be relaxed, although the routing restrictions for any 2D plane of the mesh remain unchanged. The relaxed restrictions permit a message to use virtual channel one in any dimension when the message does not need to route in the chosen direction of the *lowest* dimension in which the message needs to route. This allows a message to use the first virtual channel in a higher dimension when the message has not completed routing in the chosen direction of *all* lower dimensions. To maintain deadlock freedom, a message waits for the first virtual channel in only the lowest dimension in which the message still needs to route. We next prove that the routing algorithm is still deadlock-free unless the routing restrictions are relaxed further.

Theorem 11 *The Optimal Fully Adaptive routing algorithm is deadlock-free even with the aforementioned relaxation of the routing restrictions.*

Proof. Any cycle requires that there be dependencies across multiple dimensions, with both directions of each dimension in the cycle used to form the cycle. Notice that for a deadlock configuration to exist, every message in the cycle must be waiting for a channel held by some other messages in the cycle. Without loss of generality, assume that the Y dimension is the lowest dimension in the cycle, with South as the chosen direction. Consequently, no message in the cycle still needs to route in the X dimension. Divide the cycle into two planes along the Y dimension. The cycle includes some message, m_i , that has used at least one channel in the North plane and is either waiting to route South or has routed South. The cycle requires that some other message waits for a channel that m_i has used in the North plane. Otherwise, there is either a message in the North plane that is not blocked, which resolves the cycle, or there is a cycle in just the North plane, which is a contradiction of the assumption that the Y dimension is used to form the cycle. However, the routing algorithm does not permit a message to use virtual channel one when the message needs to route in the chosen direction of the lowest dimension in which the message needs to route. Further, a message waits for only the first virtual channel in the lowest dimension in which the message needs to route. Since m_i has not used any virtual channel one in the North plane, m_i cannot block a message in the North plane. Therefore, a cycle is not possible and the routing algorithm is deadlock-free. \square

Theorem 12 *The Optimal Fully Adaptive routing algorithm imposes the minimum number of restrictions for deadlock freedom on an n -dimensional mesh.*

Proof. By theorems 5 and 10, the Optimal Fully Adaptive routing algorithm imposes the minimum number of restrictions for each 2D plane of the mesh and precisely this set of restrictions is required. The only routing restriction is that a message cannot use virtual channel one in a higher dimension when the message needs to route in the chosen direction of the lowest dimension in which the message needs to route. Deadlock can occur if this restriction is relaxed. Without loss of generality, assume that a message, m_1 , used VC_{1U} even though m_1 needs to route South. (Let South be the chosen direction in the Y dimension and assume that m_1 does not need to route in the X dimension.) There are two messages that route South (and block m_1), use virtual channels in the appropriate dimensions, and wait for VC_{1D} in the same 2D plane where m_1 used VC_{1U} . Use additional messages to form a deadlock configuration similar to the one shown in Figure 11. The routing algorithm is no longer deadlock-free, so the routing restrictions cannot be relaxed. \square

7 Extra Virtual Channels

The Optimal Fully Adaptive routing algorithm requires only the minimum number of virtual channels. Previous routing algorithms proposed for n -dimensional meshes require more virtual channels. It may appear that adaptiveness can be increased by using one of these previous routing algorithms. However, the Optimal Fully Adaptive routing algorithm can be extended to use additional virtual channels with only the minimum number of routing restrictions.

Consider a router with more than two virtual channels per physical channel. Generalizing lemma 11, the routing algorithm is not deadlock-free unless, for each pair of dimensions, one channel in both directions of the higher dimension is not used by messages that need to route in the chosen direction of the lower dimension. These restrictions depend on the ordering of the dimensions. For example, the chosen direction in the lowest dimension (based on the ordering) cannot be used after the first virtual channel in any other dimension.

Minimizing the number of restrictions requires minimizing the number of prohibited 90° turns, since the 0° turns are restricted for all directions except in the lowest dimension. The first virtual channel of both directions of every other dimension cannot be used by a message that needs to route further in the chosen direction of the lowest dimension, so the direction with the fewest virtual channels should be used as the chosen direction of the lowest dimension. Similarly, the direction with the fewest virtual channels of the remaining dimensions should be used as the chosen direction of the next lowest dimension and so on. By selecting the dimension order and chosen directions in this way, the number of prohibited 90° turns can be minimized. The number of 90° turns that must be prohibited for an n -dimensional mesh is

$$2 \sum_{i=1}^{n-1} VC_i(n-i)$$

where VC_i is the number of virtual channels in the chosen direction of dimension i .

For example, consider a 3D mesh with the following set of virtual channels: East = 2, West = 3, North = 6, South = 5, Up = 5, and Down = 3. There are 24 different ways to order the dimensions and choose the directions, since there are six dimension orders with two of four directions chosen. The minimum number of prohibited 90° turns is achieved by using a dimension order of XZY and making East and Down the chosen directions. This configuration imposes fourteen prohibited 90°

turns, since the 90° turns from VC_{1N} , VC_{1S} , VC_{1U} , and VC_{1D} to either East virtual channel are prohibited and the 90° turns from VC_{1N} and VC_{1S} to any of the three Down virtual channels are prohibited. Any other ordering of the dimensions or any different choice of chosen directions requires that more than fourteen 90° turns be prohibited to prevent deadlock.

8 Conclusion

A deadlock-free fully adaptive routing algorithm has been proposed for wormhole routing on n -dimensional meshes. It has been proved that the routing algorithm requires only the minimum number of virtual channels and imposes only the minimum number of routing restrictions on those virtual channels. Ignoring symmetry, the Optimal Fully Adaptive routing algorithm is the *only* fully adaptive routing algorithm for meshes that satisfies both of these properties. Furthermore, if additional virtual channels are available, only the minimum number of routing restrictions are imposed on these additional channels.

Hypercube and torus topologies have also been used for commercial wormhole-routed multiprocessors. Although optimal routing algorithms have not been proposed for these topologies, the proof techniques presented in this paper can be extended to these topologies. Using these techniques could result in more adaptive routing algorithms for these topologies. This issue merits further investigation.

Acknowledgments

The authors thank Shobana Balakrishnan, Dave Lutz, Jeff May, D. K. Panda, and Kant Patel for valuable discussions and the anonymous referees for their suggestions, which have improved the content and presentation of this paper.

References

- [1] A. A. Chien. A Cost and Speed Model for k -ary n -cube Wormhole Routers. In *Hot Interconnects '93*, August 1993.
- [2] A. A. Chien and J. H. Kim. Planar-Adaptive Routing: Low-cost Adaptive Networks for Multiprocessors. In *19th Annual International Symposium on Computer Architecture*, pages 268–277, 1992.
- [3] W. J. Dally. Fine-Grain Message-Passing Concurrent Computers. In *Proceedings of the Third Conference on Hypercube Concurrent Computers*, volume 1, pages 2–12, 1988.
- [4] W. J. Dally. Virtual-Channel Flow Control. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):194–205, March 1992.
- [5] W. J. Dally and H. Aoki. Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels. *IEEE Transactions on Parallel and Distributed Systems*, 4(4):466–475, April 1993.

- [6] W. J. Dally and C. L. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.
- [7] J. Duato. On the Design of Deadlock-Free Adaptive Routing Algorithms for Multicomputers: Design Methodologies. In *Parallel Architectures and Languages Europe 91*, volume I, pages 390–405, 1991.
- [8] P. T. Gaughan and S. Yalamanchili. Adaptive Routing Protocols for Hypercube Interconnection Networks. *IEEE Computer*, 26(5):12–23, May 1993.
- [9] C. Glass and L. M. Ni. Maximally Fully Adaptive Routing in 2D Meshes. In *International Conference on Parallel Processing*, volume I, pages 101–104, 1992.
- [10] C. Glass and L. M. Ni. The Turn Model for Adaptive Routing. In *19th Annual International Symposium on Computer Architecture*, pages 278–287, 1992.
- [11] C. R. Jesshope, P. R. Miller, and J. T. Yantchev. High Performance Communications in Processor Networks. In *16th Annual International Symposium on Computer Architecture*, pages 150–157, 1989.
- [12] D. H. Linder and J. C. Harden. An Adaptive and Fault Tolerant Wormhole Routing Strategy for k -ary n -cubes. *IEEE Transactions on Computers*, 40(1):2–12, January 1991.
- [13] L. M. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, 26(2):62–76, February 1993.