

A Foundation for Designing Deadlock-free Routing Algorithms in Wormhole Networks

D. N. Jayasimha*
Intel Corporation
2200 Mission College Boulevard
Santa Clara, CA 95052-8119
djayasim@mipos2.intel.com

Loren Schwiebert
Department of Electrical and Computer Engineering
Wayne State University
Detroit, MI 48202-3902
loren@ece.eng.wayne.edu

D. Manivannan Jeff A. May
Department of Computer and Information Science
The Ohio State University
Columbus, OH 43210-1277
{manivann, may-j}@cis.ohio-state.edu

Stephen L. Hary
Department of Electrical Engineering
The Ohio State University
Columbus, OH 43210-1277
hary@ee.eng.ohio-state.edu

Abstract

This paper provides necessary and sufficient conditions for deadlock-free unicast and multicast routing with the path-based routing model in interconnection networks which use the wormhole switching technique. The theory is developed around three central concepts: channel waiting, False Resource Cycles, and valid destination sets. The first two concepts are suitable extensions to those developed for unicast routing by two authors of this paper; the third concept has been developed by Lin and Ni. The necessary and sufficient conditions can be applied in a straightforward manner to prove deadlock freedom and to find more adaptive routing algorithms for collective communication. The latter point is illustrated by developing two routing algorithms for multicast communication in 2-D mesh architectures. The first algorithm uses fewer resources (channels) than an algorithm proposed in the literature but achieves the same adaptivity. The second achieves full adaptivity for multicast routing.

1. Introduction

Collective communication routines such as broadcast, scatter, gather, reductions, transpose, prefix computations (scan), etc. are very important for developing parallel programs that are both efficient and portable. Although there is a large body of research that has addressed the development of efficient collective communication algorithms (Kumar *et al.* [8] contains a good survey; a more recent survey dealing with wormhole-routed architectures appears in [12]), this research has invariably assumed a simple underlying hardware model with non-adaptive (dimension-ordered) routing of point-to-point messages. This has been in large part because that model reflects the characteristics of most present-day commercial multicomputers. In a position paper, Ni ar-

gues that supporting multicast at the router level is critical to the efficient performance of message-based parallel computers [13]. There have been a number of recent research advances in adaptive routing and router models which permit multicasting in hardware. There are a number of theoretical and systems-related issues that arise with respect to deadlock freedom, design of routing algorithms, and design of collective communication algorithms in such systems. In this paper, we provide a *general* set of necessary and sufficient conditions for deadlock freedom for a large class of routing algorithms and router models which use wormhole routing (for a survey on wormhole routing, see [14]). We illustrate the applicability of these conditions by developing two adaptive multicast routing algorithms for 2-D mesh architectures.

2. Previous work

Dally and Seitz have shown that for nonadaptive routing algorithms, the existence of an acyclic channel dependency graph is necessary and sufficient for deadlock freedom [3]. Much of the early research on wormhole routing has focused on the design of deadlock-free routing algorithms using this result. The idea of channel waiting was introduced independently by Lin, McKinley, and Ni [10]. Nonadaptive 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 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 functions of the form $R : C \times N \times N \rightarrow P(C)$ ($P(C)$ is the *power set* of C), where a *set* of output channels is defined on which to route the message. Since a set of output channels is provided, a selection function is then used to choose one of these output channels.

Duato showed that requiring an acyclic channel depen-

*This work was done while the author was at The Ohio State University.

dependency graph is too restrictive for routing algorithms defined by relations of the form $R : N \times N \rightarrow P(C)$ [4, 5]. 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. He also designed routing algorithms based on this relaxed condition. Schwiebert and Jayasimha have used Duato's sufficiency condition and the mesh topological properties to propose an optimal fully adaptive routing algorithm for arbitrary dimension mesh networks [16, 18]. This algorithm is optimal in the number of virtual channels and in the number of restrictions they place on the use of virtual channels.

A natural question that arises is: Exactly how restrictive must the routing algorithm be to guarantee deadlock freedom? In other words, what is a necessary and sufficient condition for deadlock-free routing? This question has been independently answered by several researchers for classes of routing algorithms and particular flow control mechanisms. Previous work done by two authors of this paper provides the most general solution proposed so far. The solution, which works for routing functions of the form $R : C \times N \times N \rightarrow P(C)$, introduces two new notions: the *channel waiting graph* (*CWG*) and *False Resource Cycles* [19]. The *CWG*, which is derived on the basis of channel *waiting* instead of channel *usage*, omits most channel dependencies that cannot lead to a deadlock configuration. Consequently, the proofs of deadlock freedom become natural and straightforward. The *CWG* is a static graph, however, and the dependencies that arise among channels are dynamic. The notion of False Resource Cycles is used to capture these dynamic dependencies. For a routing algorithm that requires a blocked message to wait for a specific channel, a *CWG* with no *True Cycles* is shown to be necessary and sufficient condition for deadlock freedom (the *CWG* could contain False Resource Cycles, however). When a blocked message can wait on multiple channels, even the presence of a True Cycle does not imply a deadlock. In such a case, it is shown that the presence of a certain subgraph of the *CWG* is a necessary and sufficient condition for deadlock freedom.

The authors have extended this work to include any flow control mechanism (that is not inherently deadlock-free) [17]. They also comment that the proof technique applies to any routing relation that depends only on local information available at a router to make routing decisions. Thus a *single* set of necessary and sufficient conditions apply to all practical classes of routing algorithms and to wormhole routing, store-and-forward routing, and virtual cut-through.

The above discussion has been limited to unicast routing, i.e., the transmission of messages from one source node to a single destination node. Collective operations, defined on a group of processes, require the participation of multiple source/destination nodes. Improvement in the performance of collective operations could arise by enhancing the unicast router model. While several router models of multicasting are possible, the *path-based* model [9, 11] seems the most feasible. In the path-based model, a "single-head" worm visits the destinations in sequence. The router hardware then must have the capability to *copy and forward* a message (as opposed to just forwarding with unicast messages). Hence, the condition for deadlock freedom with path-based routing is different from unicast routing because of additional dependencies introduced into the channel dependency graph. Typical deadlock-free routing algorithms in such a model are based on the use of Hamiltonian paths

and disjoint destination sets [9, 11] or based on using only the paths of the underlying unicast routing algorithm [15]. It is possible to design multicast routing algorithms with more adaptivity by relaxing the conditions used in previous approaches. Recently, Duato has provided a sufficiency condition for deadlock-free path-based routing [6]. As a result, he has been able to design the most adaptive multicast routing algorithm to date. His theory, based on an extension of the *channel dependency graph*, introduces additional *direct* and *indirect* multicast dependencies. The approach is complicated to apply to design algorithms or to check if routing algorithms are deadlock-free.

The necessary and sufficient conditions for deadlock-free routing, using sophisticated router models such as *path-based routing*, has been an open problem. In this paper, we solve this problem.

3. Assumptions and definitions

To be precise, the channels referred to so far are the *communication* channels of the network. In addition, a processor at each node has one or more *injection* channels through which messages are injected into the network. The processor consumes or absorbs messages from the network through one or more *consumption* channels. We will continue to refer to communication channels as just channels and explicitly refer to consumption or injection channels.

The following assumptions are standard and correspond to the way (path-based multicast) wormhole routing is implemented.

1. A node can generate messages of arbitrary length destined for any other node at any rate.
2. A unicast message arriving at its destination is eventually consumed. A multicast message arriving at a destination is eventually consumed if all the remaining headers of the message can make progress.
3. Once a channel queue accepts the header flit of a message, it must accept all the flits of the message before accepting any flits from another message (this is a characteristic of wormhole routing).
4. A channel queue cannot contain flits belonging to more than one message at a time. The channel must transmit the tail flit of the current message before the channel queue accepts the header flit of the next message.
5. A node arbitrates among messages which simultaneously request the same output channel. Messages already waiting for a channel are chosen in an order that prevents starvation.

Definition 1 An *interconnection network* I is a strongly connected directed multigraph, $I = G(N, C)$, where the vertices, $n_i \in N$, are the processors and the directed edges, $c_i \in C$, are channels that connect neighboring processors. Each channel, c_i , can transmit messages from one processor, denoted s_i , to a neighboring processor, denoted d_i .

If D is an ordered subset of N , then n_1 precedes n_2 in the set D is denoted by $n_1 \prec_D n_2$. If D_1 and D_2 are two ordered subsets of N such that $\{n_1, n_2\} \subset D_1$ and $\{n_1, n_2\} \subset D_2$, then it is possible that $n_1 \prec_{D_1} n_2$ and $n_2 \prec_{D_2} n_1$.

Definition 2 Let U be the set of all ordered subsets of N . Let $V : N \rightarrow P(U)$ be a function, called valid destination set function, that assigns for each node n_s a set of ordered sets $V(n_s)$ satisfying the following property (the sets belonging to $V(n_s)$ are called the *valid destination sets* for node n_s):

- If $D \subset N$ and $n_s \notin D$, then there exists a set $A \subseteq V(n_s)$ such that $D = \bigcup_{D_i \in A} D_i$, $D_i \cap D_j = \emptyset \forall i \neq j$. In other words, any subset D of N not containing a node n_s can be partitioned as a disjoint union of valid destination sets of node n_s . We call $VDS = \bigcup_{n_s \in N} V(n_s)$, the set of valid destination sets determined by the function V .

Definition 3 A *routing relation* R is a function $R : C \times N \times P(N) \rightarrow P(C)$ and specifies a set of output channels based on the input channel, the current node, and the destination set of the message.

Definition 4 A *selection function* $S : C \times P(C \times F) \rightarrow C$ gives a single output channel based on the input channel, the set of output channels given by the routing relation, and the status of the output channels. F represents the possible states of an output channel. Typically, $F = \{\text{free, busy}\}$.

Definition 5 A *routing algorithm* corresponding to a valid destination set function V is a function $R_A : N \times P(N) \rightarrow \text{Paths_of_}I$ on interconnection network I . For each $n_s \in N$ and $D_{n_s} \in V(n_s)$, R_A assigns a set of paths $R_A(n_s, D_{n_s})$ that are available to a message from the source n_s to $\text{last}(D_{n_s})$ through the nodes of D_{n_s} in order (here $\text{last}(D_{n_s})$ is the last element in the ordered set D_{n_s}). For each destination node that is visited in order, the path also includes one or more consumption channels available to the message. $R_A(n_s, D_{n_s}) = \emptyset$ if $D_{n_s} \notin V(n_s)$. The routing is accomplished by application of a routing relation and then a selection function at each router in an intermediate node.

Note: If the valid destination set function $V : N \rightarrow P(U)$ is defined as $\forall n_s \in N, V(n_s) =$ the set of all the singleton subsets of $N - \{n_s\}$, then VDS is the set of all one element subsets of N , and R_A defines a routing algorithm for unicast. Otherwise, it defines a routing algorithm for multicast.

Definition 6 A *waiting channel* is a channel for which the message waits when the message is unable to proceed because every channel the message can use is unavailable. A message may have multiple waiting channels at a node.

Definition 7 The *channel waiting graph* (CWG) for a given routing algorithm R_A and interconnection network I is a directed graph, $CWG = G(C, E)$. The vertices of CWG are the channels of I and the consumption channels at each processor. The directed edges of CWG are ordered pairs of channels, (c_i, c_j) , where c_j is a waiting (consumption or communication) channel for a message that occupies c_i . Formally, $E = \{(c_i, c_j) \mid \exists n_a \in N \text{ and } D \in V(n_a) \text{ such that } \{\dots, c_i, \dots, c_j, \dots\} \in R_A(n_a, D) \text{ and } c_j \text{ is a waiting (communication or consumption) channel for } R_A(n_a, D) \text{ on this path}\}$

Note: There is no requirement that the message waits for c_j immediately after using c_i , only that the message is long enough to fill the channel queues from c_i to c_j . Since arbitrary message lengths are permitted, this imposes no restrictions under our system model.

Definition 8 Routing algorithm R_A is *wait-connected* if a) for every input channel on a path, there exists a waiting channel through which the message can be routed and b) on reaching a destination, a message can wait on one or more consumption channels at that node. Formally, $\forall n_a \in N$ and $D \in V(n_a), \forall c_i \in C$ such that $\{\dots, c_i, \dots\} \in R_A(n_a, D), \exists c_j \in C$ such that $\{\dots, c_i, c_j, \dots\} \in R_A(n_a, D)$ and c_j is a waiting channel for $R_A(n_a, D)$ after using c_i .

Note: A message must be able to reach all the nodes in its destination set. Hence, a blocked message must wait for at least one output channel. Otherwise, it is never delivered if it reaches an intermediate node where all the output channels and the consumption channel(s), if the message is destined to the corresponding node, are busy. Therefore, any deadlock-free routing algorithm must be wait-connected.

In the following two definitions only, “channels” refer to both communication and consumption channels.

Definition 9 A *configuration* is an assignment of messages to channels. The headers and data flits of each message are stored in the channel queues and each channel queue holds flits from at most one message. The leading channel is the communication channel the message has most recently acquired and its channel queue contains the message header. Any other communication channels occupied by this message contain only data flits. Message headers or data flits could be held in consumption channel queues of the destinations that a (multicast) message has already visited. A configuration is *legal* if each message in the configuration occupies one or more consecutive channels; the message header is stored at the head of the leading channel queue that the message occupies; each message occupies only channels the routing algorithm permits the message to use; and the storage capacity of each occupied channel has not been exceeded.

The notion of a configuration is used to define deadlocks. Note that if a message header does not occupy a communication channel, that message will eventually be consumed since every header of that message in the network occupies a consumption channel—such a message cannot give rise to a deadlock and we do not need to consider such a message in the configuration.

Definition 10 A *deadlock configuration* for routing algorithm R_A on interconnection network I is a non-empty legal configuration consisting of a set of messages, $m_1, m_2, \dots, m_n, n \geq 1$, where each message, m_i , in the set has acquired at least one channel. A header flit of m_i unable to proceed because every output channel for m_i is unavailable. Moreover, every waiting channel for m_i is occupied by either data flits of m_i or the header or data flits of another message in the set. The data flits at the head of any other channel queue held by m_i are unable to proceed because the next channel queue occupied by m_i is full. Thus, each message is blocked and must wait for an unavailable waiting channel held by another message in the set. Alternatively, if $n = 1$, m_1 waits for a channel already occupied by itself, otherwise, we can order the messages such that m_i waits for a channel occupied by $m_{i+1} \forall i < n$ and m_n waits for a channel occupied by m_1 .

4. Sufficient condition

Theorem 1 If routing algorithm R_A is wait-connected and the CWG for R_A is acyclic, then R_A is deadlock-free.

Proof. R_A is wait-connected, so every message always has a waiting channel when all output channels are busy. Assume there is a deadlock configuration involving n messages. If $n = 1$, then there is a cycle in the CWG from a channel to itself, which is not possible since the CWG is acyclic. Otherwise, $(\forall i < n)$ there is an edge in the CWG from every channel occupied by m_i to the channel occupied

by m_{i+1} for which m_i is waiting (call this channel c'_{i+1}). There is also an edge in the CWG from every channel occupied by m_n to the channel occupied by m_1 for which m_n is waiting (call this channel c'_1). Hence, there is an edge in the CWG from c'_i to c'_{i+1} ($\forall i < n$) and from c'_n to c'_1 . The CWG for R_A is acyclic, however, so no such set of edges is possible. Therefore, no deadlock configuration exists and R_A is deadlock-free. \square

An edge in the CWG requires only the existence of a path from some channel to a waiting channel. The specific intermediate channels used between this channel and the waiting channel are not considered when creating the CWG . Hence, it is possible that a cycle in the CWG exists only if two or more messages occupy the same channel. For this reason, we divide cycles in the CWG into two classes: False Resource Cycles and True Cycles. A False Resource Cycle is a cycle in the CWG that *requires* at least one channel to be occupied simultaneously by more than one message in order to create the cycle. Note that this shared channel is not necessarily within the cycle. Obviously, a False Resource Cycle cannot occur, since this is physically impossible. (Even though the configuration is legal, it is not a reachable configuration [2].) Therefore, a False Resource Cycle cannot be used to create a deadlock configuration. A True Cycle is a cycle in the CWG that permits every message in the cycle to occupy different channels.

5. Necessary and sufficient conditions

A message is unable to proceed when all output channels the message is permitted to use are busy. This situation can be resolved in one of two different ways: (1) The message could wait for a specific output channel to become free or (2) The message could wait until *any* permitted output channel becomes free. For case (1), once a waiting channel is chosen from a set of possible waiting channels, the message must then wait for that specific channel to become free. For case (2), the message also has the possibility of waiting on a subset of more than one output channel. In fact, case (2) includes any routing algorithm that does not conform to case (1). We first prove a necessary and sufficient condition for routing algorithms that belong to case (1), followed by one for case (2).

Theorem 2 *A routing algorithm, R_A , that requires a message to wait for a specific output channel is deadlock-free iff R_A is wait-connected and the CWG for R_A has no True Cycles.*

Proof. First note that R_A is wait-connected by definition. By Theorem 1, an acyclic CWG is a sufficient condition for deadlock freedom. A False Resource Cycle cannot result in deadlock, so any False Resource Cycles can be ignored. Since there are no True Cycles, the routing algorithm is deadlock-free.

To prove necessity, assume that a True Cycle with n messages exists. A deadlock configuration can be created from this True Cycle. For each $i < n$, allow message m_i to occupy channel c'_i , some additional channels if necessary, and then wait to acquire channel c'_{i+1} occupied by message m_{i+1} . (Assume that m_i and c'_i are defined as before.) Similarly, message m_n occupies channel c'_n and waits for channel c'_1 . Since this is a True Cycle, it is possible to generate a set of messages that are able to occupy the appropriate channel(s) and then wait for the appropriate channel. To

force m_i to wait for the appropriate channel, it is necessary to guarantee that every output channel m_i could use at this node is busy. For any output channel available to m_i that is also available to the source, assume the source has injected a message that is occupying this channel. If R_A is not suffix-closed (the notions of *suffix-closure* and *prefix-closure* are defined in [19]), however, it is possible that some of the output channels available to m_i can be used only by messages arriving on the input channel used by m_i . For these output channels, assume that a previous message, m_j , used this input channel and was forwarded on one of the output channels. In addition, the length of m_j is assumed to be short enough that it releases the input channel that m_i uses, however, m_j is long enough that it occupies the output channel at this node. By Assumption 2, m_j is not necessarily removed from the network immediately, so it is possible that m_j occupies this output channel for a short amount of time. Hence, it is always possible to force m_i to wait for c'_{i+1} . Clearly, each message in the set is waiting for a channel occupied by another message in the set and none of the messages can make progress. Therefore, a deadlock configuration can always be constructed from a True Cycle. \square

For routing algorithms that permit a message to wait for *any* of the output channels to become free, an acyclic CWG is not a necessary condition. Since a blocked message may have multiple waiting channels, messages may be able to avoid channels that form cycles in the CWG by using an alternative channel outside the cycle. Deadlock can be avoided, however, only if at least one of the waiting channels is *guaranteed* to become free. For this reason, we selectively remove edges from the CWG to resolve all True Cycles, as long as the routing algorithm for the resulting graph, CWG' , remains wait-connected. We next prove that if no such CWG' exists, then the routing algorithm is not deadlock-free. If such a CWG' does exist, however, then the following theorem can be used to prove deadlock freedom.

Theorem 3 *A routing algorithm, R_A , that permits a message to wait for any output channel is deadlock-free iff R_A is wait-connected for some subgraph of the CWG , called CWG' , and this CWG' has no True Cycles.*

Proof. If R_A is wait-connected for the CWG and the CWG has no True Cycles, then the result follows immediately from Theorem 2, with $CWG = CWG'$. Assume the CWG contains True Cycles. In this case, R_A must be wait-connected for some CWG' without True Cycles.

We first prove sufficiency. Consider a potential deadlock configuration for R_A , involving a cycle of n messages ($n > 0$). This requires that every message in the configuration is waiting for channels occupied by itself or another message in the cycle. Since R_A is wait-connected for CWG' , at least one of the waiting channels for each message is in CWG' . Because CWG' has no True Cycles, an output channel in CWG' eventually becomes free and some message in the set is forwarded. There is no guarantee, however, that the output channel the message, m_i , eventually acquires a channel in CWG' . (It is possible that m_i is forwarded along a different channel before an output channel in CWG' becomes free.) If m_i has reached the last node in its destination set, then the cycle has been resolved. Otherwise, whether or not m_i acquires a channel in CWG' , m_i can acquire an output channel in CWG' at the next node because R_A is wait-connected for CWG' . Hence, one of the messages can always be routed and a deadlock configuration cannot occur.

We now prove necessity by showing that the routing algorithm is not deadlock-free if every wait-connected CWG' has True Cycles. Assume that every wait-connected CWG' has True Cycles. Hence, it is possible to generate a set of messages, each of which has no waiting channel guaranteed to become available.¹ Furthermore, these messages are all blocking each other, since otherwise it would be possible to guarantee that a waiting channel becomes free. Therefore, a wait-connected CWG' without True Cycles must exist for every deadlock-free routing algorithm. \square

6. Design examples

We first present a minimal adaptive routing algorithm for a 2-dimensional mesh and prove it is deadlock-free. The routing algorithm has the same degree of adaptivity as the one described by Duato [6], since both algorithms provide the same set of routing paths. Our algorithm requires fewer resources, however — 5 virtual channels per node as opposed to 6 required by Duato’s algorithm. We then extend this routing algorithm to make it fully adaptive by adding two virtual channels at each node.

Suppose the address of each node in a 2-dimensional mesh is represented by its integer coordinates (x, y) , where the lowest left node has coordinates $(0, 0)$. Then the label assignment function L for a $k \times k$ mesh defined below assigns an integer label for each node.

$$L(x, y) = \begin{cases} (ky + x) & \text{if } y \text{ is even} \\ k(y + 1) - x - 1 & \text{if } y \text{ is odd} \end{cases}$$

This labeling induces an ordering among the nodes which defines a Hamiltonian path in the network [11]. Hereafter, we denote a node by its associated label.

Let N be the set of all the labels associated with the nodes (in the case of a $k \times k$ mesh, $N = \{0, 1, 2, \dots, k^2 - 1\}$). We define U to be the set of all ordered subsets of N where the ordering of elements is in ascending or descending order. For example, in the case of a 4×4 mesh, the ordered sets $\{2, 4, 7, 10\}$ and $\{14, 10, 5\}$ are members of U . However, $\{2, 8, 6\}$ is not a member of U since its elements are in neither ascending nor descending order. We can write U as the union of two disjoint sets U^a and U^d , where U^a contains all those ordered sets that are in ascending order and U^d contains all those ordered sets that are in descending order. The valid destination set function $V : N \rightarrow P(U)$, is defined as follows: For each $n_s \in N$, $V(n_s) = \{S \in U \mid (S \in U^a \text{ and } \forall n \in S, n > n_s) \text{ or } (S \in U^d \text{ and } \forall n \in S, n < n_s)\}$. In other words, $V(n_s)$ consists of all those ordered sets whose elements are all greater than n_s and are ordered in ascending order and the ordered sets whose elements are all less than n_s and are ordered in descending order. For any $n_s \in N$, and a destination set S for the node n_s , let $S^a = \{n \in S \mid n > n_s\}$ and $S^d = \{n \in S \mid n < n_s\}$ and sort S^a in ascending order and S^d in descending order. Then, it is clear that $S = S^a \cup S^d$ and that both S^a and S^d belong to $V(n_s)$. Thus, any destination set for a node n_s can be written as the union of two disjoint sets belonging to $V(n_s)$.

¹In fact, if even one of these messages, m_i , has a waiting channel that becomes free, then either all the messages have a waiting channel or the remaining messages in the set (without m_i) form a deadlock configuration.

Notation: For each $n_s \in N$, let $V(n_s)^{up} = \{S \in V(n_s) \mid n > n_s \forall n \in S\}$ and let $V(n_s)^{down} = \{S \in V(n_s) \mid n < n_s \forall n \in S\}$. In other words, $V(n_s)^{up}$ consists of all the valid destination sets whose elements are sorted in ascending order and $V(n_s)^{down}$ consists of all the valid destination sets whose elements are sorted in descending order.

6.1. Routing algorithm R_{AA}

We define a routing algorithm R_{AA} for a $k \times k$ mesh with respect to the valid destination sets provided by the function V defined above. Figure 1 shows an example of a 4×4 mesh. In the horizontal direction, there are three channels connecting the neighboring nodes. In the vertical direction, there are two channels connecting the neighboring nodes, one in the upward direction and one in the downward direction. The channels are colored blue, black, or green as shown in figure 1. We denote the blue channels originating from node i in the horizontal and vertical directions by $C_{i,h}^{blue}$ and $C_{i,v}^{blue}$, respectively and use similar notations for channels of other colors. When multiple messages are in the network, it is possible that a multicast message can reserve a consumption channel in one node and wait for a consumption channel in another node [1]. Thus, a circular wait among multicast messages for consumption channels can lead to a deadlock. To avoid deadlock due to consumption channels, each node has two consumption channels—the “UMC-channel” (Upward bound Message Consumption channel) and the “DMC-channel” (Downward bound Message Consumption channel). The UMC-channel at a node is used to consume a message originating from a lower labeled node and the DMC-channel for a message originating from a higher labeled node. The consumption channels at each node are not shown in figure 1. We denote the UMC-channel at node i as C_i^{UMC} and the DMC-channel at node i as C_i^{DMC} .

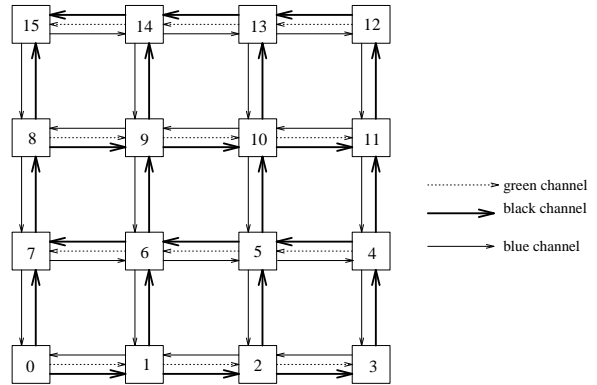


Figure 1. Label Assignment for nodes and channels of a mesh for the algorithm R_{AA} .

We define the adaptive routing algorithm $R_{AA} : N \times P(N) \rightarrow Paths_of_I$ as follows. For each $n_s \in N$, $S \in V(n_s)$,

- if $S \in V(n_s)^{up}$, then $R_{AA}(n_s, S)$ consists of all the minimal paths consisting of black, green, and/or blue channels passing through the nodes in S in ascending order such that the path never passes through a node that has a higher label than the next destination node in S . When the message arrives at a destination, it must

use the UMC-channel. Every black channel on a path $\in R_{AA}(n_s, S)$ is a waiting channel for every channel preceding that channel in the path; for each node $i \in S$, the UMC-channel at node i is a waiting channel for every channel preceding node i that lies on the path; no other channel on the path is a waiting channel.

- if $S \in V(n_s)^{down}$, then $R_{AA}(n_s, S)$ consists of all the minimal paths consisting of blue and/or green channels passing through the nodes in S in descending order such that the path never passes through a node that has a lower label than the next destination node in S . When the message arrives at a destination, it must use the DMC-channel. Every blue channel in a path provided by $R_{AA}(n_s, S)$ is a waiting channel for every channel preceding that channel in the path; for each node $i \in S$, the DMC-channel at node i is a waiting channel for every channel preceding node i that lies on the path; no other channel in the path is a waiting channel.
- $R_{AA}(n_s, S) = \emptyset$ if $S \notin V(n_s)$.

For example, in Figure 1, if a multicast message is sent from node 0 to the destination set $S = \{6, 10, 12\}$, then the path $(0 \rightarrow 1 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12) \in R_{AA}(0, S)$; however, $(0 \rightarrow 7 \rightarrow 6 \rightarrow 9 \rightarrow 10 \rightarrow 11 \rightarrow 12) \notin R_{AA}(0, S)$, since it passes through the node 7 which is larger than the next destination node, 6.

Lemma 1 *The routing algorithm R_{AA} is deadlock-free.*

Proof: By Theorem 1, it is sufficient to prove that the channel waiting graph $CWG(R_{AA})$ of the routing algorithm R_{AA} is wait-connected and acyclic. Every message from a source node n_s that is bound for a destination set $S \in V(n_s)^{up}$ has a black channel to wait on, at each node on its path; and at each node in the destination set, the message has the UMC-consumption channel to wait on. Every message from a source node n_s that is bound for a destination set $S \in V(n_s)^{down}$ has a blue channel to wait on, at each node on its path; and at each node in the destination set, the message has the DMC-consumption channel to wait on. Thus, $CWG(R_{AA})$ is wait-connected. All upward bound messages wait only on black channels and UMC-channels and all the downward bound messages wait only on blue channels and DMC-channels. Thus, all possible edges in $CWG(R_{AA})$ arise as follows:

- Edges that result from waiting dependencies among upward bound messages are :
 - edges from green, blue, black, and the UMC-channels to the black channels, and
 - edges from green, blue, black, and the UMC-channels to the UMC-channels.
- Edges that result from waiting dependencies among for downward bound messages are:
 - edges from green, blue, and DMC-channels to the blue channels, and
 - edges from green, blue, and DMC-channels to the DMC-channels.

Figure 2 shows the condensed form of $CWG(R_{AA})$. In figure 2, nodes represent groups of channels. For example, the node labeled “green channels” represents the set of all the green channels in the network. An edge from node A to node B in this graph implies that there is at least one edge in $CWG(R_{AA})$ from a channel belonging to group A to a

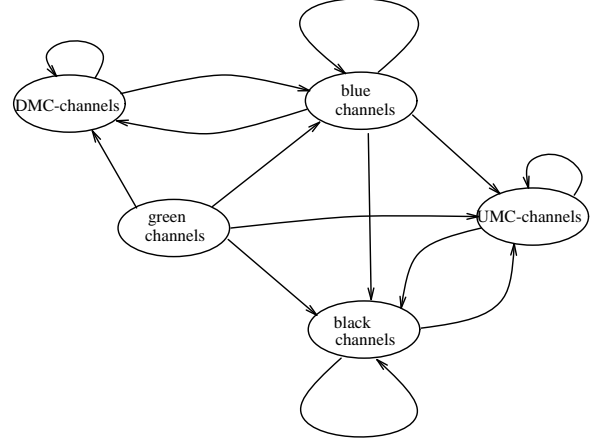


Figure 2. The condensed CWG for R_{AA}

channel belonging to group B . To prove that $CWG(R_{AA})$ is acyclic, we consider all possible cases a cycle can arise and prove that none of those cases arise.

Case (i): $CWG(R_{AA})$ contains a cycle involving a DMC-channel.

Since a waiting dependency from a DMC-channel C_i^{DMC} to a DMC-channel C_j^{DMC} implies $i > j$, a cycle involving only DMC-channels does not exist. Thus, a cycle involving a DMC-channel must contain at least one channel which is not a DMC-channel. From figure 2, it is clear that such a cycle must consist of blue channels and DMC-channels only. Dependencies among blue channels arise from the paths supplied for the upward bound messages. Since upward bound messages traverse the nodes in ascending order, if there is an edge from a blue channel $C_{i,*}^{blue}$ to another blue channel $C_{j,*}^{blue}$, then $i > j$. Similarly an edge from a blue channel $C_{i,*}^{blue}$ to a DMC-channel C_j^{DMC} implies $i > j$; and an edge from a DMC-channel C_i^{DMC} to a blue channel $C_{j,*}^{blue}$ implies $i > j$. Therefore, no cycle involving only DMC-channels and blue channels is possible and hence the DMC-channel cannot be part of a cycle. Thus proving Case (i) does not arise.

Case (ii): $CWG(R_{AA})$ contains a cycle involving a UMC-channel.

The proof is similar to Case (i).

Case (iii): $CWG(R_{AA})$ contains a cycle involving black channels only.

As observed above, if there is an edge from a black channel $C_{i,*}^{black}$ to another black channel $C_{j,*}^{black}$, then $i < j$. Thus, all the vertices in a cycle cannot be black channels.

Case (iv): $CWG(R_{AA})$ contains a cycle involving blue channels only.

Dependencies among blue channels arise from the paths supplied for the downward bound messages. Such messages traverse the nodes in descending order, and hence if there is an edge from a blue channel $C_{i,*}^{blue}$ to another blue channel $C_{j,*}^{blue}$, then $i > j$. Thus, all the nodes in a cycle cannot be blue channels.

It is easy to infer from Figure 2 that any cycle in $CWG(R_{AA})$ must be one of the four types discussed above. Hence, the $CWG(R_{AA})$ is acyclic and hence R_{AA} is

deadlock-free. \square

As we noted earlier, the routing algorithm R_{AA} defined above is not fully adaptive, because it does not allow the message to use all the shortest paths that lead to the next destination node in the destination set of a message. To make it fully adaptive, we add two additional virtual channels in the horizontal direction- thus each interior node now has seven virtual channels. These additional virtual channels are shown as red and yellow channels in Figure 3. A red channel from node i is denoted as C_i^{red} and a yellow channel from node i as C_i^{yellow} . Next, we modify the routing algorithm R_{AA} to define a new routing algorithm R_{FAA} which is minimal and fully adaptive.

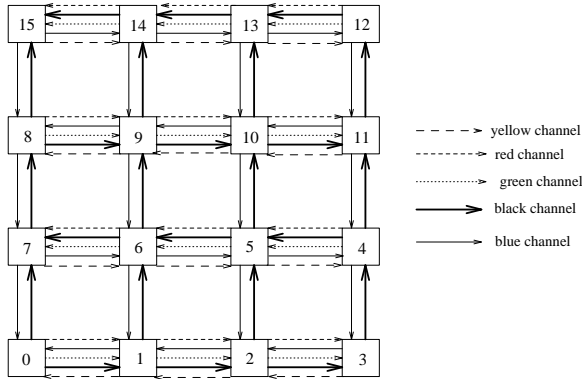


Figure 3. Label Assignment for nodes and channels of a mesh for the algorithm R_{FAA} .

6.2. Fully adaptive routing algorithm R_{FAA}

The fully adaptive routing algorithm $R_{FAA} : N \times P(N) \rightarrow Paths_of_I$ is defined as follows. For each $n_s \in N$ and for each $S \in V(n_s)$,

- if $S \in V(n_s)^{up}$, then $R_{FAA}(n_s, S)$ consists of all the minimal paths consisting of black, green, yellow, and/or blue channels traversing the nodes in S in ascending order with the exception that the path supplied by $R_{FAA}(n_s, S)$ does not allow waiting on a yellow channel when the next destination node has a higher label than the current node. Every black channel on a path $\in R_{FAA}(n_s, S)$ is a waiting channel for every channel preceding that channel in the path; every yellow channel on a path $\in R_{FAA}(n_s, S)$, except as noted above, is also a waiting channel for every channel preceding that channel in the path; for each node $i \in S$, the UMC-channel at node i is a waiting channel for every channel preceding node i that lies on the path.

For example, suppose node 1 sends a multicast message to the destination set $S_1 = \{5, 6, 10\}$; two of the possible paths supplied by $R_{FAA}(1, S_1)$ for such a message are $1 \rightarrow 6 \rightarrow 5 \rightarrow 6 \rightarrow 5 \rightarrow 10$ and the path $1 \rightarrow 6 \rightarrow 5 \rightarrow 6 \rightarrow 9 \rightarrow 10$. Here, after visiting node 6 for a second time, the message can use the blue or yellow channel to go to node 5 if one of them is available, otherwise, it should wait on the black channel, go to node 9, and then to node 10.

- if $S \in V(n_s)^{down}$, then $R_{FAA}(n_s, S)$ consists of all the minimal paths consisting of blue, red, and/or

green channels traversing the nodes in S in descending order with the exception that the path supplied by $R_{FAA}(n_s, S)$ does not allow waiting on a red channel when the next destination node has a lower label than the current node. Every blue channel in a path provided by $R_{FAA}(n_s, S)$ is a waiting channel for every channel preceding that channel in the path; every red channel on a path $\in R_{FAA}(n_s, S)$, except as noted above, is also a waiting channel for every channel preceding that channel in the path; for each node $i \in S$, the DMC-channel at node i is a waiting channel for every channel preceding node i that lies on the path; no other channel on the path is a waiting channel.

- $R_{FAA}(n_s, S) = \emptyset$ if $S \notin V(n_s)$.

Lemma 2 *The routing algorithm R_{FAA} is deadlock-free.*

Proof: By Theorem 1, it is sufficient to prove that the associated channel waiting graph $CWG(R_{FAA})$ is wait-connected and acyclic. That $CWG(R_{FAA})$ is wait-connected follows from the fact that at each node in any path provided by the routing algorithm R_{FAA} , the message can wait on a yellow or a black channel if the message is bound for destinations with a higher label than the source, and can wait on a blue or red channel if the message is bound for destinations with a label lower than the source; also, at each node in the destination set, a downward bound message can wait on the DMC-channel and an upward bound message can wait on the UMC-channel.

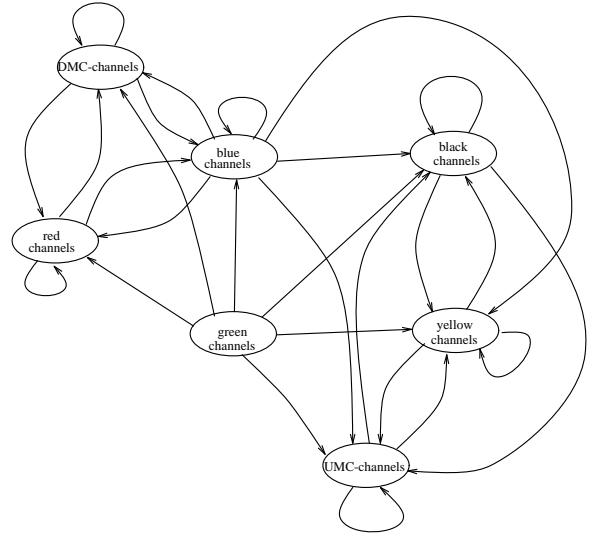


Figure 4. The condensed CWG for R_{FAA}

Thus all possible edges in $CWG(R_{FAA})$ arise as follows:

- Edges that result from waiting dependencies among upward bound messages are:
 - edges from green, yellow, blue, black, and UMC-channels to black channels,
 - edges from green, yellow, blue, black, and UMC-channels to yellow channels, and
 - edges from green, yellow, blue, black, and UMC-channels to UMC-channels.
- Edges that result from waiting dependencies among downward bound messages are:

- (a) edges from green, red, blue, and DMC-channels to red channels,
- (b) edges from green, red, blue, and DMC-channels to blue channels, and
- (c) edges from green, red, blue, and DMC-channels to DMC-channels.

The condensed form of $CWG(R_{FAA})$ is shown in Figure 4. To prove that $CWG(R_{FAA})$ is acyclic, we analyze all possible types of cycles that can exist. It is easy to see from Figure 4, that a cycle in $CWG(R_{FAA})$ must be one of the following types: (i) a cycle involving a DMC-channel, (ii) a cycle involving a UMC-channel, (iii) a cycle involving red channels only, (iv) a cycle involving blue channels only, (v) a cycle involving red and blue channels only, (vi) a cycle involving black channels only, (vii) a cycle involving yellow channels only, or (viii) a cycle involving yellow and black channels only. We can show by arguments similar to the ones given in the proof of Lemma 1 that none of these eight types of cycles exist. We omit the details of the proof for lack of space. However, a detailed proof can be found in [7]. \square

7. Conclusion

We have solved an open problem by providing necessary and sufficient conditions for deadlock-free wormhole routing with the path-based multicast router model. The theory also holds for unicast routing and thus provides a unified approach to deadlock-freedom. This is particularly important since both unicast and collective communication messages will coexist in a network. The theory can also be used as a foundation to design deadlock-free routing algorithms. We have illustrated this by proposing two deadlock-free adaptive multicast routing algorithms. The first algorithm requires fewer virtual channels than a similar one proposed by Duato but has the same adaptivity. The second algorithm is fully adaptive and requires only 7 virtual channels per node. We are not aware of any other multicast routing algorithms that are as adaptive and yet have such modest virtual channel requirements.

References

- [1] R. V. Boppana, S. Chalasani, and C. S. Raghavendra. On Multicast Wormhole Routing in Multicomputer Networks. In *Symposium on Parallel and Distributed Processing*, pages 722–729, 1994.
- [2] R. Cypher and L. Gravano. Requirements for Deadlock-Free, Adaptive Packet Routing. *SIAM Journal on Computing*, 23(6):1266–1274, December 1994.
- [3] 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.
- [4] 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.
- [5] J. Duato. A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 4(12):1320–1331, December 1993.
- [6] J. Duato. A Theory of Deadlock-Free Adaptive Multicast Routing in Wormhole Networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(9):976–987, September 1995.
- [7] D. N. Jayasimha, D. Manivannan, L. S. Jeff A. May, and S. L. Hary. “A Foundation for Designing Deadlock-free Routing Algorithms in Wormhole Networks”. Technical Report OSU-CISRC-4/96-TR20), The Ohio State University, Department of Computer and Information Science, 1996.
- [8] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing*. Benjamin/Cummings Publishing Company, Redwood City, California, 1994.
- [9] X. Lin, P. K. McKinley, and L. M. Ni. Deadlock-free Wormhole Routing in 2-D Mesh Multicomputers. *IEEE Transactions on Parallel and Distributed Systems*, 5(8):793–804, August 1994.
- [10] X. Lin, P. K. McKinley, and L. M. Ni. The Message Flow Model for Routing in Wormhole-Routed Networks. *IEEE Transactions on Parallel and Distributed Systems*, 6(7):755–760, July 1995.
- [11] X. Lin and L. Ni. Deadlock-Free Multicast Wormhole Routing in Multicomputer Networks. In *18th Annual International Symposium on Computer Architecture*, pages 116–125, 1991.
- [12] P. K. McKinley, Y.-J. Tsai, and D. F. Robinson. Collective Communication in Wormhole-Routed Massively Parallel Computers. *IEEE Computer*, 28(12):39–50, December 1995.
- [13] L. M. Ni. Should Scalable Parallel Computers Support Efficient Hardware Multicasting? In *ICPP Workshop on Challenges for Parallel Processing*, pages 2–7, August 1995.
- [14] L. M. Ni and P. K. McKinley. A Survey of Wormhole Routing Techniques in Direct Networks. *IEEE Computer*, 26(2):62–76, February 1993.
- [15] D. K. Panda, S. Singal, and P. Prabhakaran. Multidestination Message Passing Conforming to Base Wormhole Routing Scheme. In *Parallel Computer Routing and Communication Workshop*, pages 131–145, 1994.
- [16] L. Schwiebert and D. N. Jayasimha. Optimal Fully Adaptive Wormhole Routing for Meshes. In *Supercomputing '93*, pages 782–791, 1993.
- [17] L. Schwiebert and D. N. Jayasimha. A Universal Proof Technique for Deadlock-Free Routing in Interconnection Networks. In *7th Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 175–184, July 1995.
- [18] L. Schwiebert and D. N. Jayasimha. Optimal Fully Adaptive Minimal Wormhole Routing for Meshes. *Journal of Parallel and Distributed Computing*, 27(1):56–70, May 1995.
- [19] L. Schwiebert and D. N. Jayasimha. A Necessary and Sufficient Condition for Deadlock-Free Wormhole Routing. *Journal of Parallel and Distributed Computing*, 32(1):103–117, January 1996.