On Multicast Wormhole Routing in Multicomputer Networks[*]

Rajendra V. Boppana
Div. of Math and CS
Univ. of Texas at San Antonio
San Antonio, TX 78249-0664
boppana@ringer.cs.utsa.edu

Suresh Chalasani
Dept. of ECE
Univ. of Wisconsin-Madison
Madison, WI 53706-1691
suresh@cauchy.ece.wisc.edu

C. S. Raghavendra
School of EECS
Washington State Univ.
Pullman, WA 99164-2752
raghu@eecs.wsu.edu

Abstract. *We show that deadlocks due to dependencies on consumption channels is a fundamental problem in multicast wormhole routing. This issue of deadlocks has not been addressed in many previously proposed multicast algorithms. We also show that deadlocks on consumption channels can be avoided by using multiple classes of consumption channels and restricting the use of consumption channels by multicast messages. In addition, we present a new multicast routing algorithm, column-path, which uses the well-known e-cube algorithm for multicast routing. Therefore, this algorithm could be implemented in the existing multicomputers with minimal hardware support. We present a simulation study of the performance of Hamilton-path based multicast algorithms with the proposed column-path algorithm. Our simulations indicate that the simplistic scheme of sending one copy of a multicast message to each of its destinations exhibits good performance and that the new column-path algorithm offers higher throughput compared to the Hamilton-path based algorithms.*

Keywords: consumption channels, deadlocks, e-cube routing, multicast routing, wormhole routing.

1 Introduction

In a multicomputer, routing algorithms provide mechanisms for communication between processors executing a parallel program. The efficiency of routing algorithms is important for achieving high performance in parallel computers. The interprocessor communication functions in a multicomputer are usually handled by a router which receives data from incoming channels and transmits data to outgoing channels using a suitable routing algorithm. The routing functions are often implemented in a distributed manner so that all routers work independently to accomplish the desired communications. Many commercially available parallel computers use hypercube and mesh networks for interprocessor communication with a processor and router module at each node.

Routing algorithms for hypercubes and k-ary n-cubes have been extensively studied in the context of *one-to-one* or *unicast* communication. In unicast communication, each source sends its message to precisely one destination. Routing algorithms for unicast communication are usually implemented as system calls in parallel machines. More powerful communication primitives that are useful in the execution of parallel programs are *broadcast* and *multicast* which allow data to be transferred from one source node to many destinations. Broadcasts and limited forms of multicasts are supported on some commercial machines; for example, nCUBE-2 supports global broadcast to all nodes and selective multicast when all destinations form a subcube. In other machines, such as the Intel Paragon [4], users can send a multicast to multiple destinations by sending a copy of the message to each destination node.

Multicast messages are useful for efficient execution of parallel programs. Such messages occur in several contexts; for example, when a master node dispatches information to several slave nodes, synchronization of multiple nodes, cache invalidations, etc. One important question is whether to support multicast communication in a parallel machine or implement multicast communication by other alternatives; these alternatives include using multiple unicast messages for multicast communication and using a broadcast message to accomplish multicasting.

There are several studies on the performance of multicast communication in multicomputer networks [8, 5, 6, 7]. Lin *et al.* [7, 6] address the issue of routing multicasts in *wormhole*-switched (see [3] for details) multicomputer networks. They show that the algorithm for broadcasting in nCUBE2 leads to deadlocks if more than one broadcast operation is in progress. Lin et al. [6] also present two algorithms—*dual-path* and *multipath*—for multicasts in wormhole networks. These two algorithms are based on the Hamilton paths in the multicomputer network. Lin et al. [6] show that the dual-path and the multipath algorithms, when used in hypercubes and meshes to route multiple multicast messages, do not cause deadlocks due to communication channel dependencies.

A common assumption made in literature in the context of unicast and multicast message routing is that each node has just one consumption channel capable of consuming messages that are destined to it in finite amount of time. To improve the performance of unicast routing, the use of two or more consumption channels has been considered [1]. When there are only unicast messages in the network, there may be contention for communication channels between nodes; however, there will be no contention for the consumption channels, since each unicast message, once it reaches the destination, is absorbed by the consumption channel in a finite amount of time. Contention for consumption channels also does not exist when there is a single multicast message in the network at any given

[*]Rajendra Boppana's research has been supported by NSF Grant CCR-9208784. Suresh Chalasani's research has been supported by a grant from the Graduate School of UW-Madison and by NSF grants CCR-9308966 & ECS-9216308. C.S. Raghavendra's research has been supported by NSF grant MIP-9296043.

time. It is unrealistic, however, to assume that there is at most one multicast message in the network at any given time.

When there are multicast messages 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. Thus there can be a circular wait among multiple multicast messages for consumption channels, which leads to deadlocks. For example, let us consider two multicast messages M_1 and M_2 from two different source nodes. Further, assume that both M_1 and M_2 need to be delivered to destinations A and B. In such a scenario, it is possible that M_1 reaches node A first (before M_2) and reserves the consumption channel in A. Similarly, M_2 may reach node B before M_1 and thus will reserve the consumption channel in B. Now, multicast message M_1 will wait for the consumption channel in node B, while M_2 waits for M_1 to release the consumption channel in A. This causes a deadlock between messages M_1 and M_2.

In this paper, we show that all the previously proposed multicast wormhole algorithms lead to deadlocks due to dependencies on consumption channels. An example of such a deadlock is shown in Figure 5, with the presence of two multicast messages and a cyclic dependency for consumption channels. This example is explained in more detail in a later section. To solve this problem of deadlocks on consumption channels, we propose to provide multiple consumption channels in each node. Multiple consumption channels can be implemented using a single physical consumption channel on which several *virtual* consumption channels are time-multiplexed.

Another issue we address in this paper is the compatibility between unicast and multicast routing algorithms. A unicast routing algorithm and a multicast routing algorithm are said to be compatible with each other if there are no deadlocks among the unicast and multicast messages routed by them. Certain unicast routing algorithms are incompatible with some multicast routing algorithms. For example, the e-cube algorithm [3] for unicasts is incompatible with the dual-path and multipath algorithms for multicasts by Lin *et al.* [6]. One approach to handle this problem is to use the same multicast algorithm for unicast routing also. Another approach is to design multicast algorithms that are compatible with the existing unicast algorithm. Since the e-cube is a popular routing algorithm used in several recent parallel computers such as Intel Paragon [4] and Cray T3D [10], designing multicast routing algorithms that are compatible with the e-cube algorithm is important. In this paper, we present one such multicast routing algorithm, called *column-path*.

The *column-path* method uses at most two messages for each column in the mesh that contains one or more destinations of a multicast message. The column-path algorithm uses the row-column or e-cube routing method and, hence, is compatible with the e-cube routing mechanism. In addition, we show that the column-path algorithm avoids deadlocks on consumption channels using at most two consumption channels per node.

Another simple multicast routing algorithm that has been considered by researchers before is the *individual* multicast routing method. This method uses one unicast message (called *copy* to distinguish it from a true unicast message) for each destination of the multicast message. This method is simple to implement since routing a multicast message is equivalent to routing multiple unicast messages. Further, it does not suffer from deadlocks on consumption channels, and is compatible with the underlying unicast routing algorithm. Despite these advantages, the individual multicast scheme was discarded by many researchers before, since it uses far too many messages for a multicast operation. The column-path algorithm, which generates at most two messages per column, is thus a compromise between the individual multicast algorithm and the dual-path/multipath techniques.

Using simulations, we study the performance of various multicast algorithms with multiple multicasts and with a mixture of multicasts and unicasts. We also compare the performance of our column-path routing scheme with dual-path and multipath algorithms. Our results show that the performance of column-path routing is comparable to that of dual and multipath algorithms for the traffic consisting only of multicast messages and is superior to them when the traffic has a mixture of unicast and multicast messages. Though we use a two-dimensional mesh as the multicomputer network in this paper, our results extend to multi-dimensional meshes and tori in a straight forward manner.

The rest of this paper is organized as follows. Section 2 describes the various multicast algorithms in detail. Section 3 shows several situations in which deadlocks can occur when previously proposed multicast algorithms are used; this section also presents possible solutions for avoiding these deadlocks. Section 4 presents performance results of various algorithms. Section 5 summarizes the results reported in this paper.

2 Multicast routing algorithms

In this paper we consider k-radix, 2-dimensional meshes with wormhole switching technique. But all the results and discussions can be applied to multidimensional tori and meshes with suitable modifications.

The two dimensions of the mesh are denoted as DIM_1 and DIM_0. The rows of a 2D mesh are numbered from top to bottom $0, 1, \ldots, k-1$, and the columns are numbered from left to right $0, 1, \ldots, k-1$. Node x, $0 \leq x < k^2$, in a 2D mesh may be represented by a two-tuple (x_1, x_0), where x_1 is the node's row number and x_0 the node's column number in the 2D grid. The hops taken by a message in a row correspond to hops through processors in DIM_0 and hops in a column correspond to hops in DIM_1. In addition, a hop may be a '+' or a '−' hop depending on the indices of the current node and the next node in the dimension of travel. For example, DIM_{1+} hops correspond to column hops from top to bottom. A communication channel from node x to y is denoted by $[x, y]$.

Since a multicast is a complex form of communication, many researchers have addressed the issue of deadlock-free multicast routing for mesh and other networks. The main emphasis of these earlier works has been to devise multicast schemes which ensure that dependencies on communication channels are acyclic. For this purpose, sometimes multiple *virtual* channels are multiplexed on a single physical channel to create many directed virtual networks. In this paper, we are interested in algorithms that are deadlock free even without virtual channels.

In this section we describe a few important multicast algorithms that have been recently proposed, and a new

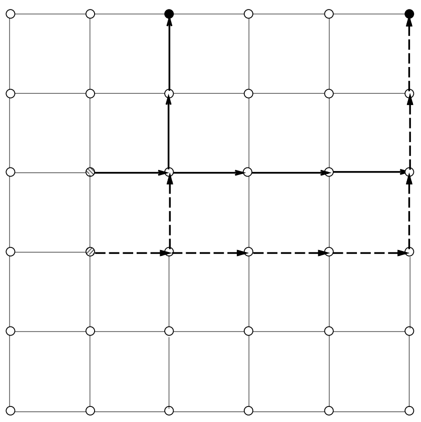


Figure 1: Example of deadlocks with the double-channel tree scheme.

algorithm, which we call the *column-path* algorithm. Our discussion in this section is primarily on the avoidance of deadlocks due to dependencies on communication channels. The issue of deadlocks due to dependencies on consumption channels is addressed in the next section.

2.1 Routing algorithms based on multicast trees

The double-channel scheme [6] is an example of multicast routing algorithms based on multicast trees. This scheme uses two virtual channels for each physical channel. Therefore, the network can be divided into four disjoint directed subnetworks: N_{++}, N_{+-}, N_{-+}, and N_{--}. The network N_{++} consists of virtual channels of the form $[(i,j),(i+1,j)]$ and $[(i,j),(i,j+1)]$, and N_{+-} consists of virtual channels of the form $[(i,j),(i+1,j)]$ and $[(i,j),(i,j-1)]$, etc. It can be easily verified that the four subnetworks use disjoint sets of virtual channels. To apply the double-channel tree scheme, Lin *et al.* partition the set of destinations, D, of a multicast message into four subsets: $D_{++}, D_{+-}, D_{-+}, D_{--}$. If (s_1, s_0) is the source of a multicast message, then its destination set is partitioned into four subsets as follows.

$$\left.\begin{array}{lcl} D_{++} & = & \{(x_1,x_0) | x_1 > s_1, x_0 \geq s_0\} \\ D_{+-} & = & \{(x_1,x_0) | x_1 > s_1, x_0 < s_0\} \\ D_{-+} & = & \{(x_1,x_0) | x_1 \leq s_1, x_0 \geq s_0\} \\ D_{--} & = & \{(x_1,x_0) | x_1 \leq s_1, x_0 < s_0\} \end{array}\right\} \quad (1)$$

After partitioning the destinations, the multicast message is sent to the destinations in the set D_{++} using the subnetwork N_{++}. Similarly, destinations in D_{+-} receive their messages via the network N_{+-}, and so on. In each subnetwork, the multicast message is sent to the destinations using a tree that routes the message along the row (in DIM$_0$) and then along the column (in DIM$_1$). The path traced by a multicast message is a tree in the shape of a comb. For an example, consider the routing of a multicast message from node $(3,1)$ to destinations $(2,2)$ and $(0,5)$. The path that is to be used by the tree scheme to route the message from $(3,1)$ is indicated by thick-dashed lines with arrows in Figure 1.

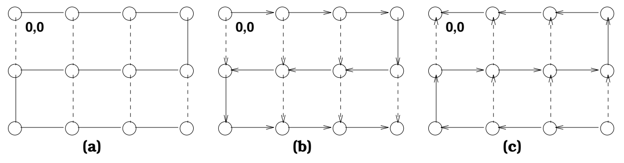


Figure 2: Example of (a) an undirected Hamilton path and the corresponding (b) H_u and (c) H_l directed networks of a mesh. The solid lines indicates the Hamilton path and dashed lines indicate the links that could be used to reduce path lengths in message routing.

Deadlocks with the double-channel tree scheme. The double-channel tree scheme is simple in concept, but it does not give good performance and has deadlocks. To show this, we now construct an example that leads to deadlocks under the double-channel scheme. Consider the scenario in which two multicast messages m_1 and m_2 are generated with sources $(2,1)$ and $(3,1)$ respectively. The destination set for both multicast messages is given by $\{(0,2),(0,5)\}$. In this case, both m_1 and m_2 will be routed entirely in the N_{-+} network. Message m_1 has acquired all channels to the destination $(0,2)$ and message m_2 has acquired all channels to the destination $(0,5)$. Figure 1 illustrates this using solid lines with arrows for the channels acquired by m_1 and dashed lines with arrows for the channels acquired by m_2. Now, message m_1 is waiting for m_2 to release the channel $[(2,5),(1,5)]$ (at node $(2,5)$) and message m_2 is waiting for m_1 to release the channel $[(2,2),(1,2)]$ at node $(2,2)$. Thus, there is a circular wait between messages m_1 and m_2 which causes a deadlock. □

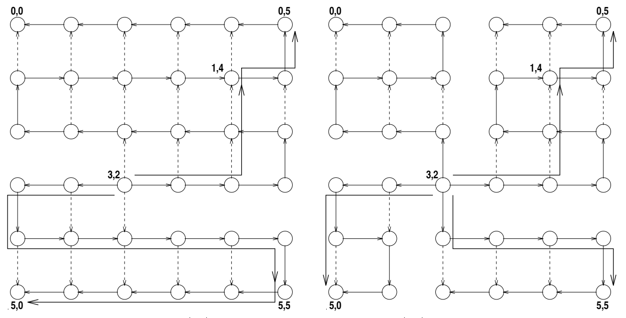
For the remainder of the paper, we consider routing algorithms that avoid deadlocks on communication channels without using multiple virtual channels.

2.2 Multicast routing based on Hamilton paths

First an undirected Hamilton path, which goes through each node exactly once, of the network is constructed. An example of an undirected Hamilton path, with node $(0,0)$ as an end node, is given in Figure 2. From this two directed hamiltonian paths can be constructed: one starts at $(0,0)$, the H_u network, and another ends at $(0,0)$, the H_l network, as shown in Figure 2. The links that are not part of the hamiltonian path may be used with appropriate direction to reduce path length.

The dual-path and multipath algorithms by Lin *et al.* [6, 7] are based on the acyclic directed networks thus formed. The destinations of a multicast message are partitioned into a small number of subsets (at most four), and a copy of the multicast message is sent to each subset of destinations. Each copy of the message visits its destinations in a predefined order, which is determined by the positions of the destinations in the Hamilton path. Different copies of a multicast message use disjoint sets of physical channels and are routed independently of one another. The path-based schemes attempt to alleviate the congestion and deadlocks introduced by tree-based multicast algorithms, albeit by using longer paths.

Lin *et al.* [6, 7] present two path-based algorithms: the dual-path and multipath algorithms.



(a) (b)

Figure 3: The subnetworks of H_u and H_l used by multicast messages from node (3,2) in (a) dual-path routing and (b) multipath routing. Also shown is the path of a multicast message from (3,2). For clarity, the links unused by messages from (3,2) are not indicated.

Dual-path algorithm. Due to the construction of the Hamilton paths H_u and H_l, the paths from any node to any other node are acyclic. In particular, some nodes are reached from a given node via H_u network only and the rest via H_l network only. For example, the nodes that can be reached from $(3,2)$ using H_l are the nodes that are between $(3,2)$ and $(0,0)$ in the undirected Hamilton path. The remaining nodes can be reached from $(3,2)$ using H_u.

In the dual-path algorithm, multicast messages from a node are transmitted on appropriate parts of the H_u and H_l networks. Figure 3(a) illustrates the portions of H_u and H_l networks used by node $(3,2)$ to send its multicast messages. Therefore, the destinations of a multicast message are placed into two groups. One group has all the destinations that can be reached from the source node using the H_u network, and the other has the remaining destinations, which can be reached using the H_l network.

Thus each source of a multicast message, depending on its location and the locations of the destinations, transmits either one or two copies of the message. For example, if $(3,2)$ needs to send a message to destinations $(0,5)$ and $(5,0)$, it will send two copies in opposite directions—one to $(0,5)$ and another to $(5,0)$. This is because, nodes $(0,5)$ and $(5,0)$ lie on the opposite sides of node $(3,2)$ in the undirected Hamilton path. On the other hand, a multicast message from $(3,2)$ to destinations $(5,5)$ and $(5,0)$ will be sent as a single message, since nodes $(5,5)$ and $(5,0)$ lie on the same side of node $(3,2)$ in the undirected Hamilton path. For shorter paths, vertical channels that are not part of the Hamilton path may used appropriately. The routing of a multicast message from $(3,2)$ to (0,5), (1,4), (5,0), and (5,5) is indicated in Figure 3(a).

Multipath algorithm. The dual-path algorithm uses at most two copies of the message for a multicast communication. This may increase the latency for some multicast messages. The multipath algorithm attempts to reduce long latencies by using up to four copies ($2n$ for n-dimensional meshes) of the original multicast message. As per the multipath routing algorithm, all the destinations of the multicast message are grouped into four disjoint subsets using rules similar to those in (1). The copies of the message are routed using the dual-path routing rules. For a complete description, see [6].

As an example, let us consider a multicast message with source $(3,2)$. For example, if the destinations are $(1,4)$, $(0,5)$, $(5,0)$ and $(5,5)$, then four copies of the multicast message (one to each destination) will be sent along disjoint paths from source $(3,2)$. Figure 3(b) indicates the routing of a multicast message from (3,2) to (0,5), (1,4), (5,0), and (5,5) using three copies.

The dual-path and multipath schemes provide deadlock-free routing of multicast messages. Further, they also provide minimal routing of unicast messages, since vertical links are used for shortcuts. Therefore, either scheme can be used to route unicast and multicast messages simultaneously in a common framework.

2.3 Other algorithms for multicast routing

The dual-path and multipath schemes are not compatible with the well-known *e*-cube routing algorithm. In this paper, we are interested in designing multicast algorithms that are compatible with the *e*-cube routing. This led us to investigate other algorithms that use the *e*-cube as the base technique for unicast routing.

Multicast routing using multiple unicast messages. One naive algorithm for multicast routing is to generate several unicast messages, one for each destination of the multicast message, and route these unicast messages independently through the network. This algorithm, called *individual*, has the potential to perform well when the average number of destinations for each multicast message is from small to medium (see Section 4). In addition, routing of each individual message can take place using unicast routing and hence there cannot be additional deadlocks (for example, due to consumption channels) in this solution.

Column-path algorithm for multicast routing. The column-path algorithm partitions the set of destinations of a multicast message into at most $2k$ subsets (k is the number of columns in the mesh), such that there are at most 2 messages directed to each column. Only one message is sent to a column if all destinations in that column are either below or above the source node; otherwise, two messages are sent to that column. In the example of Figure 4, the destinations for the multicast message with source $(2,2)$ are $(1,4)$, $(3,3)$, $(3,4)$, $(4,4)$, $(1,5)$ and $(0,5)$. In all four copies of the message are sent; one copy to $(3,3)$, one to $(1,4)$, another to $(3,4)$ and $(4,4)$, and yet another to $(1,5)$ and $(0,5)$. Each of these messages is routed using the *e*-cube (or, row-column) routing algorithm. Therefore, the column-path routing is compatible with the unicast routing method used in the current parallel computers. A similar but more general method has been independently developed by Panda *et al.* [11].

3 Deadlocks due to consumption channel dependencies

A common assumption, called *consumption assumption* in [9] and is used here, is that when a flit of a message reaches its destination, it is consumed in finite time. With this assumption, the proof of deadlock-free routing of unicast messages by a wormhole routing algorithm is reduced to showing that there do not exist cyclic dependencies on

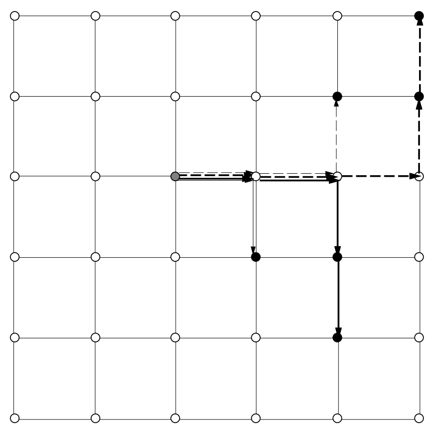


Figure 4: Example of column path routing.

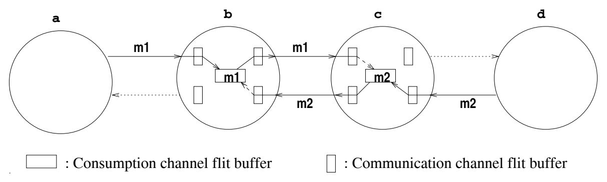

a
b
c
d
m1
m1
m1
m2
m2
m2
: Consumption channel flit buffer
: Communication channel flit buffer

Figure 5: Example of deadlocks on consumption channels.

the communication channels during the routing process. There are some recent studies on the performance effects of consumption channels for wormhole routing of unicast messages [1]. However, little attention has been paid to the issue of consumption channels in multicast wormhole routing. We now show that wormhole multicast routing algorithms cause cyclic dependencies on the consumption channels.

3.1 Deadlocks on consumption channels

To see the deadlocks on consumption channels, consider two-dimensional meshes, in which each node has one consumption channel satisfying the consumption assumption. Assume that each router has buffer space to hold only a few (typically, 1 or 2) flits of each message that is destined to its processor or passing through it. A multicast message that reaches one of its destinations and has more destinations to be visited can be handled in two ways: (a) the message reserves the consumption channel in the current destination and then reserves/awaits the communication channel in the path to its next destination; (b) the message is consumed (absorbed and removed from the network) by the current destination processor and then retransmitted to its next destination. The latter strategy is called absorb-and-retransmit. We show that deadlocks occur in both cases. First, we consider method (a) of handling messages at intermediate destinations.

Figure 5 illustrates the routing of two multicast messages in a row of a two-dimensional mesh. The first message, m_1, originates at node a and has destinations b, c; the second message, m_2, also originates at node d and has destinations b, c. Furthermore, nodes a, b, c are left neighbors to b, c, d, respectively. The following scenario is shown in Figure 5. The message m_1 obtains the communication channel from a to b, denoted $[a, b]$, consumption channel in b, denoted $Cons_b$, and communication channel $[b, c]$; m_2 obtains the communication channel $[d, c]$, consumption channel in c, $Cons_c$, and communication channel $[c, b]$. The reservation of the consumption channel is shown by labeling the (flit) buffer associated to it with the name of the message that has reserved it. It is noteworthy that each of the multicast schemes, the dual-channel tree, fixed-path, and Hamilton-path based dual-path and multipath algorithms [6], route $m1$ and $m2$ the same way as shown in the figure.

Due to flit-level flow control in wormhole routing, node b can accept only the header flit (and may be a few data flits if more than one flit is sent between nodes at a time) of $m1$. Though the consumption channel in b is free, $m1$ cannot be consumed at b until it acquires the consumption channel in c also, since messages are not buffered at intermediate nodes during routing. A similar condition occurs with the consumption channel in c and message $m2$. Therefore, $m1$ waits for the consumption channel occupied by $m2$, and vice versa. This is a circular wait on consumption channels by $m1$ and $m2$, which leads to deadlock. This observation is summarized in the following lemma.

Lemma 1 *Let M be a mesh such that each node has a consumption channel satisfying the consumption assumption. Also let F be a multicast algorithm on M such that a message can wait for a communication channel after reserving a consumption channel. Then the algorithm F can lead to deadlocks on consumption channels when it routes two or more multicast messages simultaneously.*

The proof is similar to the discussion given for the deadlock in Figure 5. The modification is to show that, if two multicasts have a common set of destinations and visit them in different orders, then cyclic dependencies on consumption channels occur.

Corollary 1 *The dual-path, multipath, and column-path algorithms can cause deadlocks on consumption channels if two or more multicast messages are to be routed simultaneously. The individual scheme does not cause deadlocks.*

We now show that deadlocks still occur if a message is absorbed and retransmitted at intermediate destinations. We still assume one consumption channel per node satisfying the consumption assumption. Let us consider Figure 5 once again. A plausible approach to avoid deadlocks on consumption channels may be to let a multicast message be absorbed by its destinations. If an absorbed message has more destinations to visit, then it may be retransmitted by the last destination node that absorbed the message. With the absorb and retransmit strategy for the situation in Figure 5, $m1$ is absorbed at b and retransmitted to c later, when $m1$ obtains the consumption channel in c; similarly, $m2$ is absorbed at c and retransmitted to b later.

The motivation for absorb and retransmit strategy is that, while intermediate nodes may not have space to store and retransmit a message, it might be feasible to do this at the message's destinations. For example, a multicast message may be absorbed at a destination through its consumption channel and retransmitted later via its injection channel.

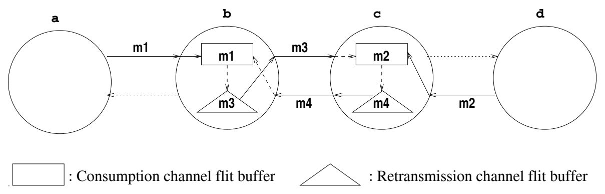

a
b
c
d
m1
m3
m1
m2
m3
m4
m4
m2
: Consumption channel flit buffer
: Retransmission channel flit buffer

Figure 6: Example of deadlocks with absorb-and-retransmit strategy. Communication channel flit buffers are not shown for clarity.

Though the absorb and retransmit strategy avoids deadlocks for the example in Figure 5, it does not work in general. Now, deadlocks occur on consumption and retransmission channels. We illustrate this in Figure 6. There are four multicast messages that need to be routed. The information on source, destination, resources acquired, and resources requested by these messages are given in the following table.

Msg.	Source	Dest. List	Resources Obtained	Resources Awaiting
$m1$	a	b, c	$[a, b], Cons_b$	$Retrans_b$
$m3$	a	b, c, d	$Retrans_b, [b, c]$	$Cons_c$
$m2$	d	b, c	$[d, c], Cons_c$	$Retrans_c$
$m4$	d	a, b, c	$Retrans_c, [c, b]$	$Cons_b$

The notations $Cons_x, Retrans_x$ are used to indicate, respectively, the consumption and retransmission channels in node x. The rectangle indicates the flit buffer associated with a consumption channel and the triangle indicates the flit buffer associated with the retransmission channel. Figure 6 shows the following scenario: $m1$ has acquired the communication channel $[a, b]$ and consumption channel in b, and waits for the retransmission channel in b; $m3$ has acquired the retransmission channel in b, communication channel $[b, c]$, and waits for the consumption channel in c; $m2$ has acquired the communication channel $[d, c]$, consumption channel in c, and waits for the retransmission channel in c; $m4$ has acquired the retransmission channel in c, communication channel $[c, b]$, and waits for the consumption channel in b. There is a circular wait for resources by these four messages, and deadlock occurs.

Lemma 2 *Let the absorb and retransmit strategy be used in routing multicast messages by algorithm F in a 2D mesh. Let m be maximum number of multicasts a node can absorb (or reside in the node) at a time. Then deadlocks can occur if* $2m+2$ *or more multicasts are in progress simultaneously.*

The above discussion, which proves the lemma for $m = 1$, can be generalized easily.

Deadlocks due to consumption resource dependencies also occur with packet or store-and-forward switching, when care is not taken. Suppose, a common pool of consumption buffers are maintained separately from the communication buffers and, when a message reaches its destination, it is placed in a consumption buffer. Such an implementation is modeled by the absorb and retransmit method described above, which leads to deadlocks. The situation is more complicated with wormhole switching, however, since a blocked multicast message holds consumption and communication resources indefinitely. One way to solve deadlocks on consumption channels is to provide as many or more consumption channels than the maximum number of messages that can be in a node at any time instant. By partitioning and systematically allocating consumption channels, however, deadlocks can be avoided used fewer consumption channels.

3.2 Prevention of deadlocks on consumption channels

The above results show that the cyclic dependencies caused by multicast messages on consumption channels is a fundamental limitation for deadlock free routing. As in the case of unicast messages, where deadlocks on communication channels are resolved by multiple virtual channels, multiple (virtual) consumption channels can be used to resolve deadlocks on consumption channels in multicast routing algorithms that are otherwise deadlock free. In particular, the dual-path, multipath, and column-path algorithms may be used for deadlock free routing if each node has a minimum of two consumption channels.

Lemma 3 *Let each node of a two-dimensional mesh have at least two consumption channels. Then, the dual-path algorithm may be used for deadlock free routing of multicast messages.*

Proof. When the dual-path algorithm is used to route a message, a copy of the message may be sent on each of the directed H_u and H_l networks. The copy of the message on H_u is routed independent of that on H_l, and vice versa. Consider the H_u network. Since it is acyclic, the physical channels in H_u can be ranked such that each and every message is routed in H_u by acquiring physical channels of increasing ranks [7]. Then, the consumption channels in all the nodes can be ranked such that the sequence of communication channels and consumption channels acquired by a message have monotonically increasing ranks. A similar observation holds for the messages on H_l. Thus, the messages on H_u (or H_l) alone cannot cause deadlocks.

Therefore, if a message is involved in a deadlock on consumption channels, then messages from both H_u and H_l are involved. Because of the construction of H_u and H_l, the only resource shared by the messages in H_u with the messages in H_l are the consumption channels. Since there exist two or more consumption channels in a node, partition them into two nonempty sets C_u and C_l. Now, messages on their H_u paths are allowed to use only the consumption channels of class C_u when they reach destinations. Similarly, messages on their H_l paths are allowed to use only the consumption channels of class C_l when they reach destinations. This avoids the cyclic dependencies and deadlocks on consumption channels. ∎

Corollary 2 *Let each node of a two-dimensional mesh have at least two consumption channels. Then, the multipath algorithm may be used for deadlock free routing of multicasts.*

Proof. The main difference between the dual-path and the multipath algorithms is that the latter algorithm may inject up to two independent copies on each of H_u and H_l networks. During the routing process, the two copies of a message on, for example, H_l, behave as though they are two independent messages generated by the same source.

This does not introduce any new deadlock possibilities. Then, the deadlocks on consumption channels is avoided by having two nonempty classes of consumption channels in each node. ∎

A similar argument can be constructed for the column-path algorithm using the fact that the paths used by the e-cube are acyclic and directed. Specifically, deadlocks on consumption channels are avoided by using C_u channels for the multicast message copies that are sent to the destinations in rows below the row of the source, and C_l for the remaining copies of the message.

4 Simulation study

To study the performance issues, we have developed a flit-level simulator. This simulator can be used for wormhole routing in meshes and tori for unicast and multicast traffic. We present performance results for dual-path, multipath, individual, and column-path algorithms.

We have simulated an 8×8 mesh with uniform traffic pattern and 20-flit messages. We have used 8 virtual lanes per physical channel to improve performance [2]. The virtual channels on a physical channel are demand time-multiplexed, and it takes one cycle to transfer a flit on a physical channel. The message interarrival times are geometrically distributed, and the uniform traffic model is used. To avoid deadlocks and improve performance, we have used eight consumption channels with suitable partitioning. (As explained in Section 3, a minimum of two consumption channels are necessary with e-cube or Hamilton-path based routing.)

We use throughput and average message latency as the performance metrics. The throughput is defined as the number of messages delivered per cycle by the network. For example, if the network throughput is 0.1, then the mean interarrival time of messages at each node is $\frac{64}{0.1} =$ 640 cycles, where 64 is the number of nodes in the network.

The message latency is defined as the number of cycles elapsed from the time a message is injected to the earliest time when each and every destination of the message has received a copy.

Each latency and throughput value reported here is obtained from the average of the values from at least four simulation samples, each with distinct random number sequences. The 90% confidence interval widths for latencies prior to saturation are within 10% of the reported mean values. The 90% confidence interval widths for throughputs values are within 5% of the reported values.

Performance under multicast traffic. In the first set of simulations, we considered multicast traffic only. The number of destinations of a message is a uniform discrete random variable with values in the closed interval [1,19]; so, a message has an average of 10 destinations. This traffic pattern is not representative of any realistic communication in multiprocessors. The purpose is to see how well a routing algorithm performs under intensive multicast traffic. Another reason for this set of experiments is to ensure that the relative performances of dual-path and multipath algorithms confirm those given by Lin *et al.* [6, Figure 9]. The results are given in Figure 7.

The individual algorithm has higher latencies than the other three algorithms. Compared to the dual-path algorithm, however, the individual exhibits higher throughput.

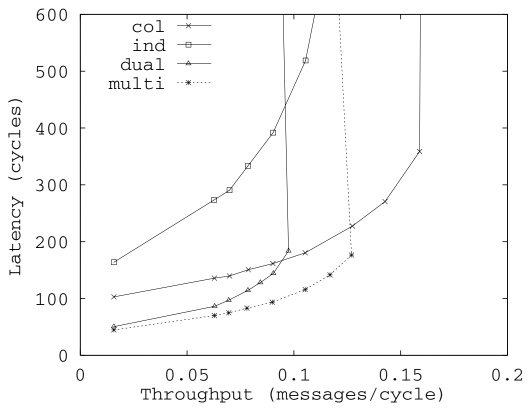


Figure 7: Performance for all multicast traffic.

At low traffic loads, the column-path algorithm has higher latencies compared to both dual-path and multipath algorithms. However, it offers much higher throughput: 25% higher throughput compared to the multipath and 62% higher throughput compared to the dual-path algorithm.

With column-path routing, the resources are held for shorter times by multicast messages (and their copies). In contrast, the dual-path and multi-path algorithms have longer message paths and hold resources (communication and consumption channels) for longer times. The higher latencies by column-path are mainly due to the limited bandwidth available to disperse multiple copies of a multicast message. With some form of adaptivity and more virtual channels, the column-path is likely to perform even better.

Performance under unicast and multicast traffic. In the second set of simulations, we considered a mixture of unicast and multicast traffic; 90% of all messages injected are unicast messages and the rest 10% are multicast messages. The number of destinations of a multicast message is a uniform discrete random variable with values in [1,7]; so, a multicast message has an average of 4 destinations. This traffic pattern could be representative of communication in cache-coherent shared memory multiprocessors; the majority of traffic is due to remote fetches of cache blocks and the rest is a mixture of traffic for invalidations of shared cache blocks and synchronizations.

For individual and column-path, we have used the standard e-cube algorithm. For dual- and multi-path algorithms, we have used the same Hamilton-path algorithm for unicast and multicast messages. (Using e-cube for unicast and dual- or multi-path for multicast messages leads to deadlocks.) The results are given in Figure 8.

The performances of all algorithms are similar for low to moderate traffic loads. The message latencies and throughputs of dual-path and multipath are indistinguishable. The message latencies of individual are only slightly higher than those for the other three algorithms. The column-path has virtually the same message latencies as those of dual-path and multipath algorithms. The individual has 8% higher throughput and the column-path 11% higher throughput than dual-path and multipath algorithms. These results also indicate that even the simple individual scheme performs well compared to more sophisticated algorithms in

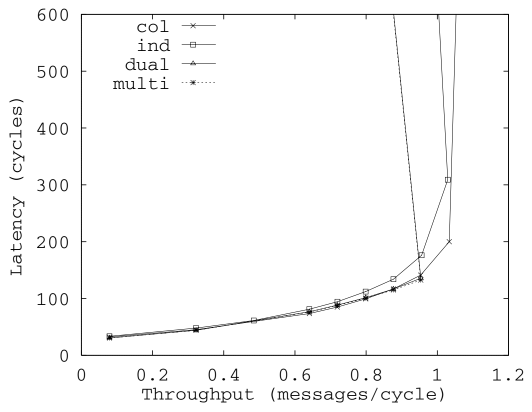


Figure 8: Performance for '90% unicast and 10% multicast' traffic.

practical situations where the majority of the traffic is unicast.

5 Summary and concluding remarks

Recently, much attention has been focused on designing deadlock-free routing algorithms for multicast communication. Thus far only deadlocks due to dependencies on communication channels and buffers has been addressed. In this paper, we have shown that the dependencies on consumption channels is a fundamental issue in the design of multicast algorithms for wormhole networks. Whenever a multicast algorithm allows a message to hold consumption channels and reserve additional communication channels, deadlocks can occur.

We have also shown that deadlocks on consumption channels may be resolved using two or more *virtual* consumption channels time-multiplexed on a single physical consumption channel. This issue is similar to that of avoiding deadlocks on communication channels with multiple virtual channels.

Another issue in the design of special purpose multicast algorithms is their compatibility with popular unicast routing algorithms. If e-cube routing is used for unicast messages, then using dual-path and multipath algorithms [6] to route multicast messages could lead to deadlocks on communication channels.

In view of this, we have discussed two simple multicast routing algorithms for mesh and torus networks that are *deadlock free* when two or more virtual consumption channels are used. One of them, called *individual*, uses one unicast message for each destination of a multicast message. It has been considered and discarded by many researchers before, since it uses far too many messages for a multicast operation. In order to contain the problem of the number of copies generated for a multicast operation, we have proposed a new approach called the *column-path* routing. Both individual and column-path follow row-column or e-cube routing method and, hence, are compatible with the routing mechanism in, for example, the Intel Paragon and Cray T3D parallel computers.

We have presented a simulation study of the performance of these algorithms with multiple multicasts and a mixture of multicasts and unicasts using simulations. We have compared the performance of our column-path routing scheme with dual-path and multipath algorithms. Our results show that the column-path routing offers superior performance compared to the previously proposed dual and multipath algorithms. In terms of metrics such as average additional traffic (used in [7]), both individual and column-path are worse than dual-path and multipath algorithms. But in terms of metrics such as network throughput, the column-path performs better than dual-path and multipath algorithms.

Specialized multicast routing algorithms tend to be incompatible with e-cube routing and cannot take advantage of the results and techniques developed for unicast routing. Our approach in this paper has been to use simple techniques that are compatible with e-cube to provide multicast routing. Our results can be easily extended to multidimensional meshes and tori. The column-path algorithm is based on a form of multicast primitive: column broadcast. In future, we will consider other multicast primitives such as subcube broadcasting to design better multicast algorithms.

Acknowledgement

The authors are grateful to Professor D.K. Panda of Ohio State University for his comments on an earlier version of this paper.

References

[1] S. Balakrishnan and D. K. Panda. Impact of multiple consumption channels on wormhole routed k-ary n-cube networks. In *Proc. 7th Int. Parallel Processing Symposium*, April 1993.

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

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

[4] Intel Corporation. *Paragon XP/S Product Overview*, 1991.

[5] Y. Lan, A.-H. Esfhanian, and L. M. Ni. Multicast in hypercube multiprocessors. *J. of Parallel and Distributed Computing*, 8:30–41, 1990.

[6] X. Lin, P. K. McKinley, and L. M. Ni. Performance evaluation of multicast wormhole routing in 2d-mesh multicomputers. In *Proc. 1991 Int. Conf. on Parallel Processing*, pages I435–I442, 1991.

[7] X. Lin and L. M. Ni. Deadlock-free multicast wormhole routing in multicomputer networks. In *Proc. 18th Ann. Int. Symp. on Comput. Arch.*, pages 116–125, 1991.

[8] P. K. McKinley and J. W. Liu. Multicast tree construction in bus-based networks. *Comm. of Assoc. for Comput. Machinery*, 33:29–42, 1990.

[9] J. Y. Ngai and C. L. Seitz. A framework for adaptive routing in multicomputer networks. In *Proc. First Symp. on Parallel Algorithms and Architectures*, pages 1–9, 1989.

[10] W. Oed. The cray research massively parallel processor system, CRAY T3D. Technical report, Cray Research Inc., Nov. 1993.

[11] D. K. Panda, S. Singhal, and P. Prabhakaran. Multidestination message passing mechanism conforming to base wormhole routing scheme. In *Proc. of Parallel Routing and Communication Workshop*, May 1994.