Proc. Fifth IEEE Symposium on Parallel and Distributed Computing, pp. 72–79, Dallas, Texas, December 1993.

Adaptive Wormhole Routing in Hypercube Multicomputers*

X. Lin, A-H. Esfahanian, P. K. McKinley, and A. Burago

Department of Computer Science
Michigan State University
East Lansing, Michigan 48824

Abstract

In this paper, we propose a uniform adaptive routing strategy for wormhole-routed hypercube networks that accommodates both unicast and multicast communication. Based on a node labeling method, the resultant routing algorithms are shown to be deadlock-free without requiring virtual channels. The order in which the destinations are visited is important to efficiency. We present an ordering algorithm, quadratic in the number of destinations, which is optimal in that it minimizes the traffic generated under the proposed paradigm. A greedy algorithm is also proposed for ordering the destinations, which requires less time and space to execute but creates more traffic than the optimal algorithm. Simulation results that evaluate the performance of the proposed routing algorithms are presented.

1 Introduction

Efficient communication among nodes is critical to the performance of multicomputers. A *routing algorithm* determines the path traversed by a message (or packet) in order to reach its destination. In most systems, each node contains a separate *router* to handle such communication-related tasks. Typically, the first part of a packet, called the *header*, contains information used in routing.

Routing can be classified as *deterministic* or *adaptive*. In deterministic routing, the path is completely determined by the source and destination addresses. A routing technique is adaptive if, for a given source and destination, which path is taken by a particular packet depends on dynamic network conditions, such as the presence of faulty or congested channels. Further, a routing algorithm is said to be *minimal* if the path selected is one of the shortest paths between the source and destination pair. Using a minimal routing algorithm, every channel visited will bring the packet closer to the destination. A *nonminimal* routing algorithm allows packets to follow a longer path.

In some cases, a source node may require that a message be delivered to more than one destination. A *multicast* communication service is one in which the same message is delivered from a source node to an arbitrary number of destination nodes. Both *unicast*, which involves a single destination, and *broadcast*, in which a message is sent to all nodes in the network, are special cases of multicast. General multicast services have been shown to be very useful in large-scale multiprocessors [1, 2, 3].

Formally, a hypercube, or n-cube, consists of 2^n nodes, each of which has a unique n-bit binary address. For each node v, let v also denote its address, and let $\| v \|$ represent the number of 1's in v. Two nodes u and v in an n-cube are connected by a channel (u, v) if and only if $\| u \oplus v \| = 1$, where $\oplus$ is the bitwise exclusive-or operation on binary numbers. For any pair of nodes x and y in a multicomputer, the distance between these nodes, denoted $dis(u, v)$, is defined as the length (in number of edges) of a shortest path between the two nodes. In an n-cube, it turns out that $dis(u, v) = \| u \oplus v \|$.

Wormhole routing [4] is a switching technology used in many current parallel machines. In this approach, a packet is divided into a number of *flits* for transmission. The header flit(s) of a packet governs the route, and the remaining flits follow in pipeline fashion. When a packet is blocked due to the unavailability of a channel, the packet is not buffered at the router preceding that channel, but rather remains *in the network*, specifically, in small flit buffers at the routers along the established path. The two salient characteristics of wormhole routing are that the network latency is relatively distance-

*This work was supported in part by the NSF grants MIP-9204066, CDA-9121641, CDA9222901, by DOE grant DE-FG02-93ER25167, and by an Ameritech Faculty Fellowship.

insensitive when there is no channel contention and that only very small buffers are required at routers [5].

Because blocked messages hold some channels while waiting for others, wormhole routing is particularly susceptible to deadlock. Although several deadlock-free adaptive unicast routing algorithms and deadlock-free deterministic multicast routing algorithms have been proposed recently for wormhole-routed networks, these techniques are not compatible with one another.

The contribution of this research is to propose a deadlock-free adaptive routing scheme that handles uniformly both unicast and multicast communication in wormhole-routed hypercube networks. In Section 2, we discuss the deadlock issue as related to adaptive routing and describe the difficulties encountered in developing deadlock-free adaptive multicast routing algorithms. Section 3 presents an adaptive unicast routing algorithm that can coexist with the proposed multicast routing method, which itself is presented in Section 4. A path-based approach to multicast communication is used, whereby a message is pipelined through the network, visiting each destination in turn. The order in which the destinations are visited is important to efficiency, that is, minimizing the total amount of traffic produced. We present both a suboptimal greedy ordering algorithm and an optimal polynomial-time ordering algorithm. Simulation results of the proposed adaptive routing algorithms are presented in Section 5, and Section 6 contains concluding remarks.

2 The Deadlock Problem in Adaptive Wormhole Routing

The deadlock problem in wormhole-routed networks has been extensively studied. Perhaps the most popular method of avoiding deadlock is to use a deterministic routing algorithm, such as *dimension-ordered routing* [6], where freedom from deadlock is guaranteed by enforcing a strictly monotonic order on the dimensions of the network traversed by each message.

Deadlock-free adaptive unicast routing methods have also been proposed. One class of algorithms requires additional (virtual) channels to support the adaptive routing [7] [8]. In this method, the multicomputer network is partitioned into several disjoint acyclic subnetworks, each subnetwork containing channels that form all of the shortest paths from one node to some other nodes. Another method for adaptive unicast wormhole routing is the *turn model*, which involves analysis of the cycles that can be formed when messages change direction. All potential cycles are avoided by prohibiting certain "turns," producing a partially adaptive routing algorithm.

Lin *et al* [9] previously developed a deadlock-free approach to deterministic multicast routing, called *path-based* routing. A *multicast path* for a source and a set of destinations consists of a set of consecutive channels, starting from the source node and traversing each destination in the set. A multicast path can be represented by an ordered list of addresses $(s, d_1, d_2, \ldots, d_m)$, where s is the source node and the d_i's are destinations in the order they are reached by the worm. Path-based multicasting may be implemented by placing an ordered list of destinations in the header of the message. Each destination address occupies one or more flits of the message header. When the header arrives at the router of destination d_i, the address d_i is removed from the message header and the subsequent flits are forwarded both to the local host and towards destination d_{i+1}. If the header arrives at the router of a node that is not a destination, the router simply forwards the message towards the next destination to be visited. Eventually, the data component of the message will arrive at all the destinations.

Figure 1 gives an example of path-based multicast in a 4-cube (routers are not explicitly shown). The source node 0111 sends a message to five destinations: 0011, 0100, 1000, 1100, and 1111. The single path visits the destinations destinations in the following order: 1111, 1000, 1100, 0100, and 0011. Why this particular order was chosen will become clear in Section 4. Please note that routers at four non-destination nodes, namely, 1011, 1001, 0000, and 0001, are required to forward the message.

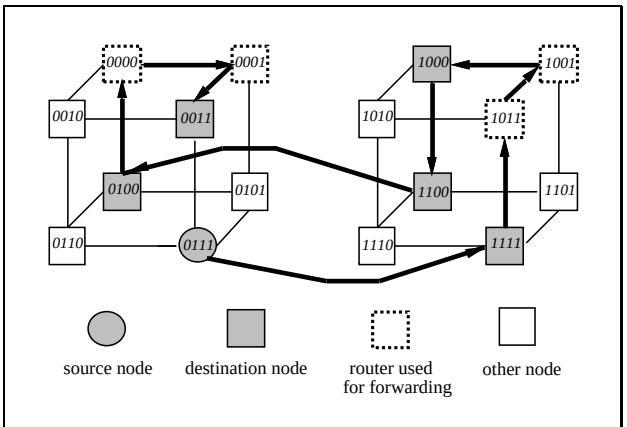


Figure 1. An example of path-based multicast in a 4-cube.

It is possible to use more than one multicast path to deliver a message to a set of destinations. Multiple-path, deadlock-free routing algorithms have been devised for many topologies, including hypercubes and meshes [9]. In this paper, we consider only single-path approaches, which are better-suited for so called one-port architectures [10], in which a node can send/receive only one message at a time. Furthermore, we consider only minimal multicast routing. Specifically, given a multicast path represented by $(s, d_1, d_2, \ldots, d_m)$, the path from s to d_1 and the path from d_i to d_{i+1}, $1 \leq i < m$, must be a shortest path.

In order to develop an adaptive path-based multicast routing algorithm, two important issues must be addressed. First, as with deterministic multicast communication, the degenerate cases of unicast and broadcast must use the same routing strategy in order to guarantee freedom from deadlock. The second issue involves the ordering of destinations in the path. Because of the pipelining characteristic of wormhole routing, it is not sufficient to simply order the destinations randomly and perform adaptive unicast routing between each pair, as deadlock may occur. Rather, the destinations must be ordered in such a way as to allow deadlock-free adaptive routing between the source and the first destination and between subsequent pairs of destinations, while using as few number of the total channels (the traffic) as possible.

3 Adaptive Unicast Routing

The adaptive routing algorithms presented in this paper are based on labeling all the nodes in the system; deadlock is avoided by restricting the order in which nodes may be visited. We define a one-to-one mapping ℓ from the nodes of an n-cube to a set of labels $[0, 2^n - 1]$. The label for a node with address $d_{n-1}d_{n-2}\ldots d_0$ is $\ell(d_{n-1}d_{n-2}\ldots d_0) = \sum_{i=0}^{n-1}(c_i\overline{d_i}2^i + \overline{c_i}d_i2^i)$, where $c_{n-1} = 0, c_{n-j} = d_{n-1} \oplus d_{n-2} \oplus \ldots \oplus d_{n-j+1}$, for $1 < j \leq n$, and $\overline{c_i}$ is the complement of c_i. It is straightforward to check that for any pair of nodes u and v in n-cube, $\ell(u) \neq \ell(v)$ if $u \neq v$. In fact, this particular labeling follows a Hamiltonian path of the graph representing the hypercube [9].

Figure 2 shows an example of label assignment ℓ for a 3-cube. Notice that adjacent nodes are actually connected by to two unidirectional channels in opposite directions. We identify two classes of unidirectional channels: a channel from a node with a lower label to a node with a higher label is referred to as high-channel (H-channel); the remaining channels are referred to as low-channels (L-channels). The paths followed by our routing algorithms are based on the following definitions.

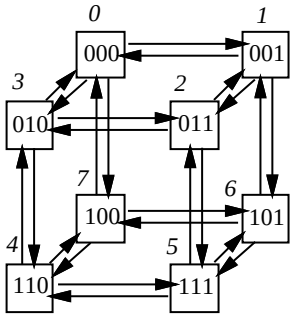


Figure 2. Label assignment for a 3-cube.

Definition 1 *A path $(v_1, v_2, \ldots, v_k)$ is an* up path (U-path) *if $\ell(v_i) < \ell(v_{i+1})$ for $1 \leq i < k$. A path $(v_1, v_2, \ldots, v_k)$ is a* down path (D-path) *if $\ell(v_i) > \ell(v_{i+1})$ for $1 \leq i < k$.*

Definition 2 *A path $(v_1, v_2, \ldots, v_k)$ is an* up-down path (UD-path) *if there exists an integer j, $1 \leq j \leq k$, such that the path $(v_1, v_2, \ldots, v_j)$ is a U-path and $(v_j, v_{j+1}, \ldots, v_k)$ is a D-path. If a UD-path is a shortest path from v_1 to v_k, the path is also referred to as a shortest UD-path.*

In Figure 2, for example, the path $(011, 111, 101, 100)$ is a U-path, the path $(100, 110, 010)$ is a D-path, and the path $(011, 111, 101, 100, 110, 010)$ is a UD-path. Since either the "up part" or the "down part" of a UD-path may be empty, both U-paths and D-paths are also UD-paths. It is easy to see that if a path $(s, v_1, v_2, \ldots, v_k, d)$ is a UD-path from s to d, then the path $(d, v_k, v_{k-1}, \ldots, v_1, s)$ is also a UD-path from d to s. Thus, given any two nodes s and d, if there exist m different shortest UD-paths from s to d, then there must exist m different shortest UD-paths from d to s. The following lemma is important for the labeling function to support the adaptive routing. Due to space limitations, the proofs of all lemmas and theorems are omitted here, but can be found in [11].

Lemma 1 *Given the labeling function ℓ defined above, for any two nodes u and v, there is a shortest U-path from u to v if $\ell(u) < \ell(v)$. Similarly, there is a shortest D-path from u to v if $\ell(u) > \ell(v)$.*

Clearly, a U-path consists solely of H-channels and a D-path contains only L-channels. A message routed

along a UD-path travels first along H-channels (if necessary) then along L-channels (if necessary). As will be shown, requiring that all messages follow UD-paths results in a routing strategy that is deadlock-free.

The *degree of adaptivity* of a routing algorithm based on UD-paths depends on the number of different UD-paths that exist between each pair of nodes. Let $p(u,v)$ denote the number of distinct, but not necessarily disjoint, UD-paths from u to v. Given an integer k, we define P_k to be the minimum number of shortest UD-paths between any pair of nodes that are distance k apart. We define E_k to be the average number of shortest UD-paths between pairs of nodes that are distance k apart. Formally, $P_k = \min\{p(u,v) | u,v \in V(H) \text{ and } dis(u,v) = k\}$, and $E_k = \sum_{u,v,dis(u,v)=k} p(u,v)/m$, where m is the total number of pairs u, v with $dis(u,v) = k$.

Lemma 2 *The minimum and average number of UD-paths for a distance k, namely P_k and E_k, respectively, are independent of the size of the hypercube.*

Table 1 shows the values of P_k and E_k for $1 \leq k \leq 10$. The proposed adaptive unicast routing algorithm forwards a message along any shortest UD-path from the source to the destination.

Table 1. Number of Distance k Shortest UD-Paths

k	1	2	3	4	5
P_k	1	1	2	4	12
E_k	1	1.5	3.0	7.5	22.5

k	6	7	8	9	10
P_k	36	144	512	1500	4650
E_k	78.75	315	1417.5	7087.5	28750.5

Figure 3 shows the algorithm as executed at each node along the path, including the source node. The parameter TAG can take on values H, L, or S, indicating whether the message arrived at the router via an H-channel, an L-channel, or the source node s, respectively. The message is routed along H-channels until the label of the current node v is greater than that of the destination node d. After this point, the message may continue to be routed along H-channels, as long as they lie on a shortest path from s to d. Upon arrival at any node whose label is greater than that of the destination, however, the message may begin to follow L-channels, which will continue until it reaches the destination.

Figure 4 gives an example of the operation of Algorithm 1. Suppose that a source node 110 (label 4) wants to send a message to node 001 (label 1). For simplicity, we refer the nodes by their labels. By Step 2 of Algorithm 1, at source node 4, any H-channel that lies on a shortest path from node 4 to node 1 may be selected. Such H-channels are (4, 7) and (4, 5). Alternatively, L-channel (4, 3) may be selected by Step 3. If channel (4, 5) were selected, then upon arrival of the message at node 5, channel (5, 6) could be selected by Step 2, or channel (5, 2) could be selected by Step 3. Figure 4 shows the four shortest UD-paths from source node 4 to destination node 1 that may be selected by the routing algorithm.

Algorithm 1: Adaptive Unicast Routing.
Input: current node v, destination node d, and message TAG.
Output: An outgoing channel (v,w) to be used to forward the message towards d.
Procedure:

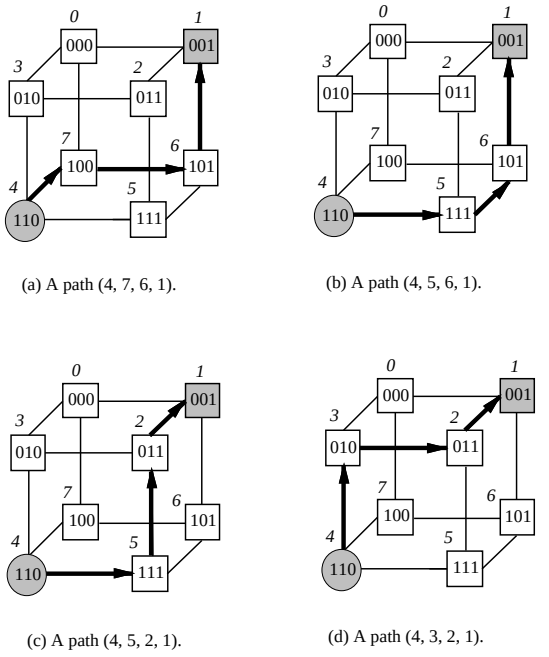
1. If $v = d$, send the message to the local processor of d, STOP.
2. If TAG=S or TAG=H, select any available H-channel (v,w), such that w is in a shortest path from v to d, STOP. If no such H-channel is available, go to the next step.
3. If $\ell(v) > \ell(d)$, select any available L-channel (v,w), such that w is in a shortest path from v to d, STOP. If no such L-channel is available, go to the next step.
4. Wait for a predetermined period of time, go to Step 2.

Figure 3. Adaptive unicast routing algorithm.

Theorem 1 *The path whose channels are selected by Algorithm 1 is a UD-path from the source to destination. Any shortest UD-path from the source to the destination can be selected by the routing algorithm. The routing algorithm is deadlock-free.*

4 Adaptive Multicast Routing

If all message routes are UD-paths under a given labeling of the nodes and a given routing algorithm, then the routing algorithm is deadlock-free. In this section, we show how to extend this strategy to multicast communication. Recall that a multicast path can be represented by a list that begins with the



(a) A path (4, 7, 6, 1).

(b) A path (4, 5, 6, 1).

(c) A path (4, 5, 2, 1).

(d) A path (4, 3, 2, 1).

Figure 4. Four shortest UD-paths from 110 to 001.

source node and contains all the destination nodes. Our goal is to find an ordering of the destinations such that the resultant multicast path is a UD-path and such that the path minimizes the total number of channels traversed by the message. The following definitions will be used towards this end.

Definition 3 *Given a labeling function ℓ as defined before and a node set $D_k, D_k = \{d_0, d_1, \ldots, d_k\}$, a permutation $(d_{i_0}, d_{i_1}, \ldots, d_{i_k})$ of D_k is an* up-down permutation (UD-permutation) *if there exists an index t, $0 \leq t \leq k$, such that, for $0 \leq j < t$, $\ell(d_{i_j}) < \ell(d_{i_{j+1}})$; and for $t \leq j < k$, $\ell(d_{i_j}) > \ell(d_{i_{j+1}})$. The* length *of the UD-permutation $(d_{i_0}, d_{i_1}, \ldots, d_{i_k})$ is $\sum_{j=0}^{k-1} dis(d_{i_j}, d_{i_{j+1}})$. An* optimal UD-permutation *of D_k is a UD-permutation of minimum length.*

Note that if a permutation $(d_{i_0}, d_{i_1}, \ldots, d_{i_k})$ is a UD-permutation, then the permutation $(d_{i_k}, d_{i_{k-1}}, \ldots, d_{i_0})$ is also a UD-permutation. Clearly, the list of destinations associated with a multicast UD-path must be a UD-permutation.

We first present a greedy algorithm that can be used to obtain a UD-permutation of the destinations of a multicast. The algorithm does not always find an optimal UD-permutation, but it is simple and easy to implement. We then propose an optimal ordering algorithm whose complexity is quadratic in the number of destinations.

4.1 Greedy UD Ordering Algorithm

Given a multicast with source node s, let $d_0 = s$, and let $d_1, d_2, \ldots, d_m$ be the destinations. For the present, let us assume that $\ell(d_0) < \ell(d_i)$ for $1 \leq i \leq m$; we will later consider the case in which some destinations have labels less than that of the source. Note that any UD-permutation for the given multicast must start with d_0.

Figure 5 gives a greedy algorithm for ordering the destination nodes and placing them in the message header for a path-based multicast operation. The first step is to sort the source and destinations in ascending order using their labels as keys. Without loss of generality, suppose after the sorting, we have the ordered list $(d_0, d_1, d_2, \ldots, d_m)$.

Algorithm 2: Greedy Header Construction.
Input: Source node d_0 and destination set D.
Output: Ordered list of destination nodes, M_H.
Procedure:

1. Sort d_0 and the elements of D by their labels. Suppose that after sorting, the ordered set is $(d_0, d_1, \ldots d_m)$.
2. Set $P = (d_m)$.
3. For $k = m$ down to 0 do:

   Suppose that the first node of P is $d_{i_{k+1}}$ and the last one is d_{i_m}. If $dis(d_k, d_{i_{k+1}}) < dis(d_{i_m}, d_k)$, then set $P = (d_k P)$; else set $P = (P d_k)$.
4. Assume that $P = (d_{j_0}, d_{j_1}, \ldots, d_{j_m})$. If $d_{j_0} = d_0$, then set $M_H = P$; else (i.e., $d_{j_m} = d_0$) set $M_H = (d_{j_m} d_{j_{m-1}} \ldots d_{j_1}, d_{j_0})$
5. Construct one message, containing M_H as part of the header of the message.

Figure 5. Greedy destination ordering algorithm.

Step 2 begins the construction of a partial UD-permutation, initially containing only one destination, d_m. In the first iteration of the loop in Step 3, node d_{m-1} is added to the partial UD-permutation either before or after d_m, depending on which case results in a shorter permutation length. In the next iteration of the loop in Step 3, d_{m-2} is added to the partial UD-permutation (either (d_m, d_{m-1}) or (d_{m-1}, d_m)). This

process continues until the last node d_0 is added to the UD-permutation.

Now suppose that some destinations have labels lower than that of the source node s. For example, after the sorting in Step 1 of Algorithm 2, we may have $(d_{i_1}, d_{i_2}, \ldots, d_{i_k}, s, d_{i_{k+1}}, \ldots, d_{i_m})$, with $\ell(d_{i_1}) < \ell(d_{i_2}) < \ldots < \ell(d_{i_k}) < \ell(s) < \ell(d_{i_{k+1}}) < \ldots < \ell(d_{i_m})$. Clearly, these nodes will need to be placed in decreasing order at the end of any UD-permutation that results from Algorithm 2.

Theorem 2 *The greedy algorithm in Figure 5 generates a UD-permutation for the given destination nodes. The time complexity is $O(m \log m)$, where m is the number of the destination nodes.*

We emphasize that, in many cases, Algorithm 2 can actually be executed at compile time, which can save time if a node multicasts to a given set of destinations more than once during execution of the application. The multicast routing algorithm that is executed at each node along the path, including the source node, is given in Figure 6.

Algorithm 3: Adaptive Multicast Routing
Input: Message with $M_H = (d_i, \ldots, d_m)$, local address v.
Output: Message forwarding decision.
Procedure:

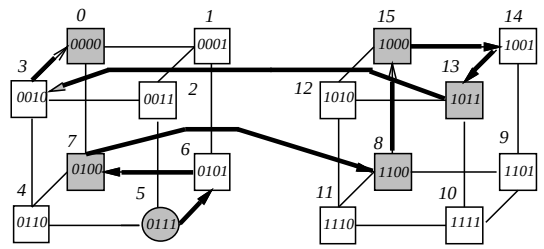
1. If $v = d_i$, then set $M_{H'} = M_H - \{d_i\}$ and forward a copy of the message to the local node; otherwise, set $M_{H'} = M_H$.
2. If $M_{H'} = \emptyset$, then terminate the message transmission.
3. Let d be the first node in $M_{H'}$. If $\ell(v) < \ell(d_i)$, then select any available channel (v, v'), such that $\ell(v) < \ell(v') \leq \ell(d_i)$. If $\ell(v) > \ell(d_i)$, then select any available channel (v, v'), such that $\ell(v) > \ell(v') \geq \ell(d_i)$.
4. Forward the message to node v' with address destination list $M_{H'}$ in its header.

Figure 6. Adaptive multicast routing algorithm.

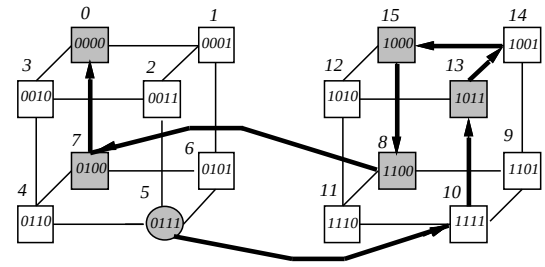
Starting from the source node, the algorithm selects an H-channel if the label of the first destination in the message header is greater than that of the current node; otherwise, it uses an L-channel. The multicast path whose channels are selected by Algorithm 3 is clearly a UD-path, and the time complexity of Algorithm 3 is $O(n)$ for an n-cube.

Theorem 3 *The routing algorithm in Figure 6 is deadlock-free.*

Figure 7(a) shows an example of a routing path selected by Algorithm 3. In this example, the source node is 5 and the multicast destination list is $\{0, 7, 8, 13, 15\}$. The UD-permutation resulting from application of the greedy sorting algorithm is $(5, 7, 8, 15, 13, 0)$, which has total length 9. However, it is easy to see that the UD-permutation $(5, 13, 15, 8, 7, 0)$ has total length 7; in fact, the UD-permutation is optimal. Figure 7(b) shows the routing path associated with the optimal ordering.



(a) An example of a multicast for greedy algorithm.



(b) A multicast path with optimal UD−sequence.

Figure 7. Multicast paths on a 4-cube.

4.2 Optimal UD Ordering Algorithm

We now show how to order the destination nodes of a given multicast so as to obtain a minimum length UD-path. We again assume without loss of generality that, for a multicast with source s and destinations $d_1, d_2, \ldots, d_m$, $s = d_0$, and $\ell(d_i) < \ell(d_{i+1})$ for $0 \leq i \leq m$. The proposed method uses dynamic programming to find an optimal UD-permutation of the node addresses, starting with d_0.

Let $D_k = \{d_k, d_{k+1}, \ldots, d_m\}$, and $u, v \in D_k$. We will denote by $M_k(u, v)$ the length of a least length UD-permutation of $d_k, d_{k+1}, \ldots, d_m$ that starts with

u and ends with v. Since any UD-permutation of $d_k, d_{k+1}, \ldots, d_m$ either starts or ends with d_k, for $M_k(u,v)$ to be defined, we must have either $u = d_k$ or $v = d_k$. To be complete, we will let $M_k(u,v) = \infty$ if either $u \notin D_k$ or $v \notin D_k$, and $M_k(u,u) = 0$, $\forall u$ and $\forall k$. Our objective is to find the least possible $M_0(d_0, v)$ or $M_0(u, d_0)$. The following theorem establishes that the *principle of optimality* holds for the problem of finding an optimal UD-permutation.

Theorem 4

$$M_k(d_k, y) = \min_{\text{all possible } x} \{dis(d_k, x) + M_{k+1}(x, y)\} \quad (1)$$

$$M_k(x, d_k) = \min_{\text{all possible } y} \{M_{k+1}(x, y) + dis(y, d_k)\} \quad (2)$$

Figure 8 gives the algorithm for an optimal UD-permutation based on the formulae in Theorem 4. As with Algorithm 2, if some destinations have lower labels than the source node s, then these must be placed at the end of the UD-permutation in decreasing order.

Algorithm 4: Optimal Ordering Algorithm.
Input: Source node d_0 and destination set D.
Output: Optimally ordered destination list, M_H.
Procedure:

1. Sort D using their labels as keys. Suppose that after sorting, the ordered set is $(d_0, d_1, \ldots d_m)$.
2. Set $M_m(d_m, d_m) = 0$.
3. For all $k = m$ down to 0 do:

   Calculate $M_k(d_k, y)$ and $M_k(x, d_k)$ for all possible x, y, using the fomulas in the above claim. Record the permutation corresponding to $M_k(d_k, y)$ or $M_k(x, d_k)$.
4. Choose the minimum value from $M_0(d_0, y)$ and $M_0(x, d_0)$ for all possible x or y. Assume that P is the permutation corresponding to the minimum value, $P = (d_{i_0}, d_{i_1}, \ldots, d_{i_m})$.
5. If $d_{i_0} = d_0$, then set $M_H = P$. Otherwise (i.e., $d_{i_m} = d_0$) set $M_H = (d_{i_m} d_{i_{m-1}} \ldots d_{i_1} d_{i_0})$. Construct one message, containing M_H in the message header.

Figure 8. Optimal header construction.

Theorem 5 *For a give multicast with source s, $s = d_0$, and destination $d_1, d_2, \ldots, d_m$, the permutation generated by Algorithm 4 is an optimal UD-permutation starting from s. The time complexity of Algorithm 4 is $O(m^2)$.*

5 Performance Evaluation

We have conducted a simulation study of the performance of the proposed adaptive routing algorithms for a 6-cube. Our performance metrics are the average traffic created and the number of the alternative UD-paths for different numbers of destination nodes. Each unit of traffic represents the transmission of a message over a channel. Destination set sizes ranged from 1 to 40. For each size, a large number (more than 1,000) of random multicast sets were generated and used as input to the two ordering algorithms.

Figure 9 plots the average amount of traffic generated by each of the two algorithms. When the number of the destinations is relatively small, the traffic created by the greedy algorithm is only slightly higher than that created by the optimal algorithm. However, when the number of destination is large, the optimal algorithm demonstrates much better performance. However, this advantage must be weighed against the greater time complexity of the optimal algorithm.

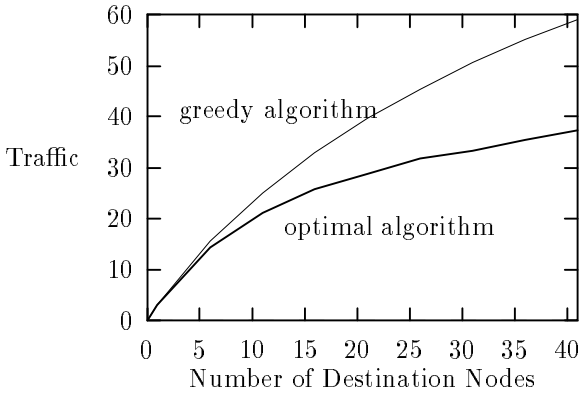


Figure 9. Average traffic generated.

For multicast communication, the multicast path should be a UD-path in order to be deadlock-free. In a multicast path, the path from the source node to the first destination node, as well as any path from a destination to next destination, must be a U-path or a D-path. Table 2 gives the average number of the U-paths (also the number of D-paths in the reverse direction) between any two nodes at various distances from one another. Again, as shown

in Lemma 2, the number of such paths between two nodes is independent of the size of the hypercube.

Table 2. Average number of U-paths between nodes.

distance	1	2	3	4	5
U-paths	1	1	1.5	3.0	7.5

distance	6	7	8	9	10
U-paths	22.5	78.75	315	1417.5	7087.5

We also measured the potential adaptivity of the proposed algorithms by computing the number of available multicast paths from the source node to the last destination. The optimal ordering algorithm was executed on destination sets. Given such an ordering, the number of UD-paths is the product of the number of permissible paths between each pair of successive nodes. The results are shown in Figure 10. Although the number of the multicast UD-paths for a given multicast is not always large, the major advantage of using this method is that it is compatible with the deadlock-free unicast routing strategy proposed in the previous section.

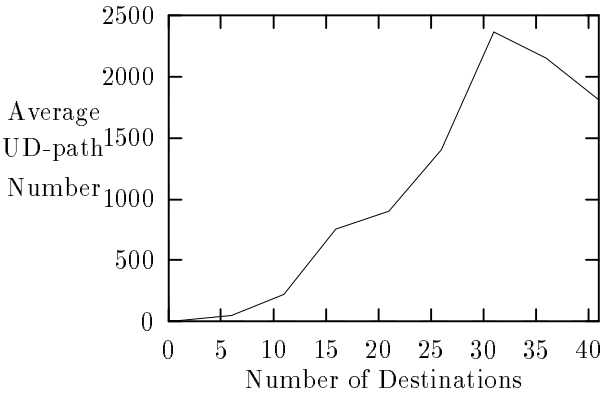


Figure 10. Available number of paths.

6 Conclusions

We have proposed adaptive unicast and multicast wormhole routing algorithms for hypercube multicomputers, all of which are deadlock-free. We have presented both a suboptimal greedy algorithm and an optimal algorithm for ordering the destinations in a multicast UD-path. The use of a single multicast path makes these algorithms well-suited for one-port architectures, where each router is connected to its local processor by a single pair of input/output channels. A simulation study was conducted to measure the performance of the proposed algorithms. The most important contribution of this work is that it is the first to describe a routing algorithm that accommodates both adaptive unicast and adaptive multicast communication in wormhole-routed networks.

References

1. P. K. McKinley, H. Xu, A.-H. Esfahanian, and L. M. Ni, "Unicast-based multicast communication in wormhole-routed networks," in *Proc. of the 1992 International Conference on Parallel Processing*, vol. II, pp. 10–19, Aug. 1992.
2. P. K. McKinley, H. Xu, E. Kalns, and L. M. Ni, "ComPaSS: Efficient communication services for scalable architectures," in *Proceedings of Supercomputing'92*, pp. 478 – 487, Nov. 1992.
3. K. Li and R. Schaefer, "A hypercube shared virtual memory," in *Proc. of the 1989 International Conference on Parallel Processing*, vol. I, pp. 125 – 132, Aug. 1989.
4. W. J. Dally and C. L. Seitz, "The torus routing chip," *Journal of Distributed Computing*, vol. 1, no. 3, pp. 187–196, 1986.
5. L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *IEEE Computer*, vol. 26, pp. 62–76, Feb. 1993.
6. W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Transactions on Computers*, vol. C-36, pp. 547–553, May 1987.
7. C. R. Jesshope, P. R. Miller, and J. T. Yantchev, "High Performance Communications in Processor Networks," in *Proceedings of IEEE 16th Annual International Symposium on Computer Architecture*, pp. 150–157, 1989.
8. D. H. Linder and J. C. Harden, "An adaptive and fault tolerant wormhole routing strategy for $k$ary n-cubes," *IEEE Transactions on Computers*, vol. 40, pp. 2–12, Jan. 1991.
9. X. Lin, P. K. McKinley, and L. M. Ni, "Deadlock-free multicast wormhole routing in 2D mesh multicomputers." accepted to appear in *IEEE Transactions on Parallel and Distributed Systems*.
10. S. L. Johnsson and C.-T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Transactions on Computers*, vol. C-38, pp. 1249–1268, Sept. 1989.
11. X. Lin, A.-H. Esfahanian, A. Burago, and P. K. McKinley, "Adaptive multicast wormhole routing in hypercube multicomputers," Tech. Rep. MSU-CPS-93-10, Department of Computer Science, Michigan State University, East Lansing, Michigan, Apr. 1991.