Journal of Applied Sciences 4 (4): 568-574, 2004
ISSN 1607-8926


# Adaptive Wormhole Routing in Mesh-Hypercube Network

[1]Bassam Al-Mahadeen and [2]Mahmoud Omari
[1]Arab Academy for Banking and Financial Sciences, P.O. Box 09131, Amman 24911, Jordan
[2]Department of Information Technology, Mutah University, P.O. Box 7, Mu'tah, Al-Karak 61710, Jordan

**Abstract:** The aim of this research was the development of wormhole routing algorithms for the Mesh Hypercube (MH) network. MH has been introduced as a new interconnection network for parallel systems. The basic structure for this network is a combination of both mesh and hypercube networks. It combines the attractive features of both the hypercube and the mesh, while at the same time overcoming their disadvantages. Attractive features of the MH are high connectivity, simple message routing, fault-tolerance, scalability, constant node degree and small diameter and average distance. This research introduced two wormhole routing algorithms for unicast and multicast communication. These algorithms are based on labeling the nodes in the network to prevent deadlock and livelock problems. Messages are allowed to travel in ascending order or descending order of labels only to prevent any cyclic dependencies between nodes. The algorithms are shown to be deadlock-free and livelock-free.

**Key words:** Adaptive routing, mesh-hypercube, hypercube, mesh, deadlock, livelock, wormhole routing

## INTRODUCTION

The design of efficient routing algorithms has received significant attention due to its importance to high performance in multiprocessor systems[1-7]. Many applications that run on multiprocessor systems depend highly on communication time. Examples of such applications include sorting, matrix computations, parallel prefix and FFT.

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[8] .

Wormhole routing has become the switching technique of choice in modern distributed-memory multiprocessors such as Intel Paragon, the Cray T3D, the MIT J-machine, the Caltech MOSAIC and the nCUBE[9]. In wormhole routing, each message is divided into packets, which are then divided into flits. The header flits of a message contain the routing information and the data flits of the message follow the header flits through the network. When the header flits arrive at an intermediate router, the router immediately forwards the header to a neighboring router if an output channel the message can use is available. The data flits then follow the header flits in a pipelined fashion. If the header is unable to proceed because no appropriate output channel is free, the router buffers only a few flits, rather than the entire message[8,10]. The data flits contain no routing information, so messages cannot share channels. Hence, each channel in the path is reserved from the time the header flits acquire the channel until the last flit of the message has traversed the channel[10,11].

There are two factors account the popularity of wormhole routing for distributed memory multiprocessors. First, since the flits of a message are forwarded as soon as possible, the message latency is largely insensitive to the distance between the source and destination. Hence, wormhole routing has lower message latency when there is little or no channel contention. Second, wormhole routing requires only enough storage to buffer a few flits, rather than the entire packet[8,10].

Routing algorithms can be classified as deterministic or adaptive. In deterministic routing, a single path is defined between the source and destination. Adaptive routing algorithms, on the other hand, support multiple paths between the source and destination depending on dynamic network conditions, such as presence of faulty or congested channels. Adaptive routing algorithms are either minimal or non-minimal. Minimal routing algorithms allow only shortest paths to be chosen, while non-minimal routing algorithms do not require message to use only shortest paths[8,10]. Adaptive routing algorithms can be further differentiated by the number of shortest paths

**Corresponding Author:** Mahmoud Omari, Department of Information Technology, Mutah University, P.O. Box (7), Mu'tah, Al-Karak 61710, Jordan Tel: (+962) 3 237 2380 Fax: (+962) 3 237 5340 E-mail: omari@mutah.edu.jo

allowed. Partially adaptive routing algorithms do not allow all messages to use any shortest path. Fully adaptive routing algorithms allow messages to use any shortest path[8,11].

Messages in distributed-memory multiprocessors can be unicast (point-to-point), multicast, or broadcast. In unicast communication, a source node requires that a message to be delivered to a single destination. A multicast communication is one in which the same message is delivered from a source node to an arbitrary number of destination nodes. In a broadcast, a message is sent to all processors. Both unicast and broadcast are special cases of multicast[8,12].

In multicasting, we assume the multi-destination facility that allows the same message to be sent to several destinations located on the same path from the source of the message to its last destination. A destination gets a copy of the message while passing it on to the next destination. The number of message start-ups can be reduced compared to multicasting based on sending separate message to the destinations. Moreover, reducing the number of individual messages needed to accomplish a multicast can be expected to reduce the overall distance traversed, decreasing both contention and networks time[12].

There are two issues to be taken in consideration when designing wormhole routing algorithms. These are deadlock and livelock. A deadlock occurs when some packets cannot advance toward their destination because the buffers requested by them are full. All the packets involved in a deadlock configuration are blocked forever. Therefore, in deadlock configuration, a set of packets is blocked forever. Every packet is requesting resources held by other packet(s) while holding resources requested by other packets[11]. Livelock arises when some packets are not able to reach their destination, even if they never blocked permanently. A packet may be traveling around its destination node, never reaching it because the channels required to do so are occupied by other packets[11].

A common technique to avoid deadlock and livelock consists of establishing an order between resources and granting resources to each packet in descending or ascending order only. So, a label function can be used to give labels to each node in order to determine the path between source and destination.

In this study two algorithms were suggested for wormhole routing in Mesh-Hypercube networks. The first algorithm was for unicast communication, while the second was for the multicast communication. Both algorithms were shown to be deadlock-free and livelock-free.

**The mesh-hypercube network**: The MH introduced in[2] is an interconnection network characterized by a two-tuple (m, n), where m defines the number of nodes forming a one-dimension mesh and n defines the number of nodes in a hypercube. For an MH (m, n), where n is a power of 2, the total number of nodes is equal to m×n. Each node in the network is recognized by its row, L and its cube binary address, X, denoted by the pair (L, X), where, $1 \le L \le m$, $X=x_w...x_1x_0$, $0 \le w \le (\log n)-1$ and $x_i \in \{0, 1\}$. Two nodes ($L_1$, X) and ($L_2$, Y) are connected if:

X = Y and $|L_2-L_1| = 1$ or
$L_1 = L_2$ and X and Y differ in exactly one bit.

Figure 1 shows a MH (3,8) network, the dotted lines represent hypercube links and the solid lines represent 1-dimensional meshes.

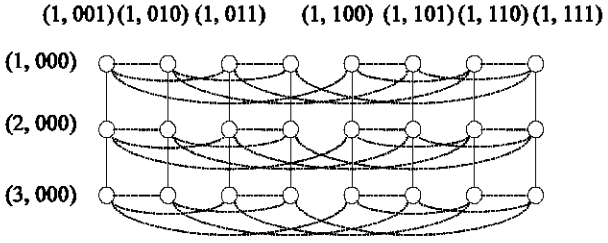


Fig. 1: MH (3, 8) network

An important feature in a multiprocessor system is the existence of multiple paths between every two processors[4,13-17]. The existence of multiple paths between every two processors enhances the fault tolerance of the system and the speedup of the transfer of data between pairs of processors[18,19]. When routing between processors, if there is a faulty link or node in the current path, the routing algorithm can choose a non-faulty path.

The length of a path between any two nodes is equal to the number of links contained in the path. The distance between two nodes X and Y in a hypercube is equal to the Hamming distance between their binary addresses, denoted by H(X, Y). That is, if H(X,Y) = d, then X's and Y's binary addresses differ in exactly d bit positions. A path between X and Y is called optimal path if its length is equal to the distance between the two nodes. A shortest path is a path of minimal length among all possible paths between the two nodes. Figure 2 shows all possible shortest paths in MH (3, 8).

**System model**: The present study supposed that the MH is characterized by two additional properties described hereunder. First, attached to each processor there is an

(1,000) → (1,100) → (1,110) → (2,110) → (3,110) → (3,111)
(1,000) → (1,001) → (1,101) → (2,101) → (3,101) → (3,111)
(1,000) → (1,010) → (1,011) → (1,111) → (3,111) → (3,111)
(1,000) → (2,000) → (3,000) → (3,001) → (3,011) → (3,111)

Fig. 2: Parallel paths between (1,000) and (3,111) in MH (3,8)

all-port router and communication between neighboring processors is handled by their routers. A router is connected to each neighbor using a separate full-duplex external channel and it can simultaneously relay or send out as many packets as there are outgoing channels. A router can also simultaneously receive as many packets as it has incoming channels. This implies that the number of internal channels (i.e., channels that connect a router to its processor) is equal to the number of external channels. Obviously, an outgoing channel must be free in order to transmit on it. This all-port assumption was made in several previous studies[7,20] although most commercial systems use a one-port architecture. The goal is to assess the potential benefits of this costlier architectural feature.

Second, a multi-destination facility is available. In multicast communication, the header of a packet consists of an ordered list of destination addresses D= ($n_0$, $n_1$, …, $n_{d-1}$), where, d is the number of destinations the packet is to be sent to as a group. The packet is sent first to the processor whose address is $n_0$, where, $n_0$ is removed from the header and the packet is forwarded to the next destination $n_1$. This process is repeated until the final destination $n_{d-1}$ is reached. The router of each destination processor copies the data of the packet to the processor[12].

As wormhole routing is assumed the routing algorithms use the first flit of a packet to determine the next router on the path to the next destination. Subsequent flits follow in a pipeline fashion. The time to communicate a message from a source to a final destination is called the message latency or communication time. It is composed of three components: start-up time, network latency and blocking time. The network latency is the time, assuming no interconnection contention, from when the first flit leaves the source to when it arrives at the final destination. The network latency for a packet of length l flits when it traverses d routers equals $\tau_s$*d+l*$\tau_t$, where, $\tau_s$ is the switching delay of one router and $\tau_t$ is the time needed by a flit to traverse one direct link. The message latency $T_c$ for such packet is defined by the following equation when $t_s$ is the start-up time:

$$T_c = t_s+\tau_s*d+l*\tau_t + \text{blocking time}$$

The blocking time is the time the first flit spends waiting for the next outgoing channel it will take to its next destination to become free. This time depends on the state of the interconnection network, which is difficult to describe analytically. The start-up time is the time needed to prepare the message at the source and to receive it at a destination[11,12].

**Adaptive unicast routing algorithm:** The adaptive routing algorithms presented in this study which based on labeling all nodes in the system, so deadlock and livelock are avoided by restricting the order in which nodes may be visited. This study define a mapping function, Label, from nodes in each hypercube subnetwork of a MH (m,n) to a set of labels [0..n-1]. The label of a node with address (L, O) is computed using the following function:
Function Label (O: binary label ): return a label ∈ [0 …n-1]

O = $o_{k-1}$ …..$o_0$, where, k = log n, f =0, w =1
for I =0 to k-2 do
    $e_i = o_{k-1} \oplus \ldots\ldots \oplus o_{i+1}$  //the symbol ⊕ means the XOR bit operation
    $f = f + (\bar{e}_i o_i w + e_i \bar{o}_i w)$
    w = w * 2
endfor
    $e_{k-1} = 0$
    $f = f + o_{k-1} * w$
return f
end Label

It is straight forward to see that for any pair of nodes on the same hypercube (L, X) and (L, Y) in an MH(m, n), Label(X) ≠ Label(Y) if X ≠ Y. This labeling function follows a Hamiltonian path of the graph representing each hypercube subnetwork. Figure 3 shows an example of node labels for an MH (3, 8).

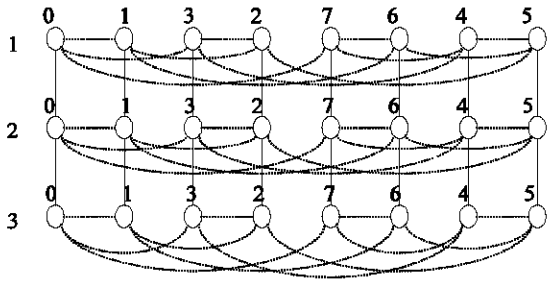


Fig. 3: Labeling nodes for MH (3, 8) network

It is assumed that each physical link constitute of two unidirectional channels in opposite directions. This assumption identify four classes of unidirectional channels: a channel from a node with a lower label to a

node with a higher label in the same hypercube is referred to as cube-high-channel (CH-channel), a channel from a node with a higher label to a node with a lower label in the same hypercube is referred to as cube-low-channel (CL-channel). A mesh channel from a node on a lower level to a node on a higher level is referred to as mesh-high-channel (MH-channel), while a mesh channel from a node on a higher level to a node on a lower level is referred to as mesh-low-channel (ML-channel).

**Definition 1:** A path that consists of H-channels linking the nodes $u_1$, $u_2$, …,$u_k$ is called an up path (U-path) if Label ($u_1$) < Label ($u_2$) < …..< Label ($u_k$).

**Definition 2:** A path that consists of L-channel linking the nodes $u_1$, $u_2$, …,$u_k$ is called down path (D-path) if Label ($u_1$) > Label ($u_2$) > …> Label ($u_k$).

From definition 1 and definition 2, four paths can be defined as follows: CU-path, CD-path, MU-path and MD-path. Figure 4 shows that the path that links the nodes in the hypercube at level 1 (0, 2, 5, 7) is an Up-path (U-path) and the path that links the nodes in the hypercube at level 3 (6, 5, 2, 1) is down-path (D-path). It can be easily seen that if a path (s, $u_1$, $u_2$, …, $u_k$, d) is a H-path from s to d, then the path (d, $u_k$, $u_{k-1}$, …, $u_1$,s) is a D-path. Thus, for any two nodes u and v, if there exist m different shortest U-paths from u to v, then there must exist m different shortest D-paths.

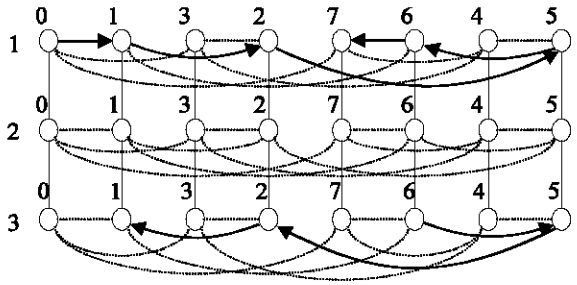


Fig. 4: An up-path and a d-path example

**Proposition 1:** Using the labeling function, Label, described above, for any two nodes ($L_i$, X) and ($L_i$, Y) there is a shortest CU-path from ($L_i$, X) to ($L_i$, Y) if Label (X) < Label (Y). Similarly, there is a shortest CD-path from ($L_i$, Y) to ($L_i$,X) if Label (Y) > Label (X).

Due to the regularity of the mesh-hypercube structure, a distributed routing scheme can be implemented without global information. At the source node, the message contains the source address, the destination address and message length. The interprocessor message traffic of a node gets redistributed into two categories, the hypercube communication and the mesh communication. If the source and the destination of the message are within the same hypercube subnetwork of the MH network, the routing procedure follows a CU-path or a CD-path in the hypercube-interconnected network. Similarly, if the source and the destination of the message are within the same mesh subnetwork of the MH network, the routing procedure follows a MU-path or a MD-path in the mesh-interconnected network.

If neither of the previous two cases is true, the source and the destination are on different hypercubes and on different meshes subnetworks of the MH network. A routing strategy for this case is first to use the hypercube routing scheme until the message arrives at the same mesh where the destination resides and then to use the mesh routing scheme for the message to arrive at the destination. Or the mesh routing scheme can first be applied to forward the message to the same hypercube where the destination resides and then the message can reach the destination using the hypercube routing scheme. A third routing strategy would route messages by mixing the hypercube and the mesh routing until the message is forwarded to the same hypercube or to the same mesh where the destination resides and then forward the message to the destination using the hypercube or the mesh routing scheme, respectively. A specification of the unicast routing algorithm that uses the second strategy is given bellow.

Procedure route-in-MH( (L, X), ($L_d$, D) )
// X and D denote the labels computed using Label function
begin
case 1: (L, X) = ($L_d$, D):
Send message to local processor and exit the algorithm.
case 2: X = D: // on the same mesh
route-in-M((L,X), ($L_d$,D))
case 3: L = Ld: // i.e. on the same hypercube
route-in-H((L,X), ($L_d$,D))
default: // L ≠ Ld and X ≠ D. i.e. different mesh and different hypercube
route-M-then-H((L,X), ($L_d$,D))
end route-in-MH.
Procedure route-in-H( (L, X), ($L_d$, D) )
begin

**step 1:** if (X < D) then
{find a node Y where, Y > X and Y≤ D and an available cube link on the shortest path between X and Y then send header to (L, Y),
if cube link not available wait for a predefined time then go to step 1}

**step 2:** if (X > D)
{find a node Y where Y < X and Y ≥ D and an available cube link on the shortest path between X and Y then send Header to (L, Y),
if cube link not available wait for a predefined time then go to step 2}
end route-in-H.
Procedure route-in-M ( (L, X), ($L_d$, D) )
begin
step 1: if L > $L_d$ then Send header to (L-1, X);
step 2: if L < $L_d$ then Send header to (L+1, X);

end route-in-M.

Procedure route-in-M-then-H ( (L, X), ($L_d$, D) )
// algorithm is called when L≠$L_d$ and X≠D,
// i.e. source and destination are on different meshes and different hypercubes
begin
route-in-M ( (L, X), ($L_d$, D) )
route-in-H ( (L, X), ($L_d$, D) )
end route-in-mesh-then-hypercube.

**Proposition 2:** The unicast algorithm is deadlock free.
Since the path of the message is U-path or D-path only and since there are two unidirectional channels in opposite directions, there will not be any cyclic dependencies between nodes. So, the deadlock will never occur.

**Proposition 3:** The unicast algorithm is livelock free.
Sine a message follows only an U-path or D-path, so in every routing step the distance between the current node and the destination node will be reduced. So the message always travels toward its destination and finally it reaches its destination.

**Adaptive multicast routing in algorithm:** If all message routes are U-paths or D-paths under a given labeling of the nodes and a given routing algorithm, then the routing algorithm is deadlock-free. Multicast path can be represented by a list of addresses (s, $d_1$, $d_2$, …, $d_m$), where s is the source node and the $d_i$'s are destinations in the header of the message. So, our first goal is to order the destinations such as the resultant multicast paths are U-paths and/or D-paths such that the paths minimize the total number of channels traversed by 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 $d_{i+1}$.

**Definition 3:** Given a labeling function, Label, as defined before and a node set $D_k$, $D_k = \{d_0, d_1, \ldots, d_k\}$, a permutation ($d_{i0}$, $d_{i1}$, ….., $d_{ik}$) of $D_k$ is an U-permutation if Label ($d_{i0}$) < Label ($d_{i1}$) < … < Label ($d_{ik}$). A permutation ($d_{i0}$, $d_{i1}$, ….., $d_{ik}$) is a D-permutation if Label ($d_{i0}$) > Label ($d_{i1}$) > …> Label ($d_{ik}$). In the following procedure this study present deadlock free and livelock free multicast routing algorithm.

**Procedure multicast-routing**

**Step 1:** Before transmission, the source node, s, sorts the destination addresses in ascending order according to their level number and label number.

**Step 2:** s, prepares 4-messages (depending on its label in the network) as follows:

- the 1st message contains the destination addresses of all nodes $d_i$'s on the same cube such that Label($d_i$) > Label (s). This message will follow the CU-path.
- the 2nd message contains the destination addresses of all nodes $d_j$'s on the same cube such that the Label ($d_j$) < Label (s). This message will follow the CD-path.
- the 3rd message contains the destination addresses of all nodes on levels > the level of s. This message will follow the MU-path.
- the 4th message contains the destination addresses of all nodes on levels < the level of s. This message will follow the MD-path.
  The 4 messages are sent simultaneously along their paths.

**Step 3:** When a node (v) receives a multicast message it compare its label with the addresses in the message header, if so, it receives a copy of the message and removes its address from the header. If not, the current node sends three copies of the received message as follows:

- the 1st message contains the destination addresses of all nodes $d_i$'s on the same cube such that Label ($d_i$) > Label (v). This message will follow the CU-path.
- the 2nd message contains the destination addresses of all nodes $d_j$'s on the same cube such that the Label ($d_j$) < Label (v). This message will follow the CD-path.
- the 3rd message contains the destination addresses of all nodes on levels > level (v), if level(v) > level(s) This message will follow the MU-path. Or contains

the destination addresses of all nodes on levels < level (v), if level (v) < level(s). This message will follow the MD-path

- The 3 messages are sent simultaneously along their paths.

**Step 4:** Repeat step 3 until the message reaches to all of its destinations.
end Multicast-routing.

**An illustrative example:** Suppose the source node (2, 4), sends a multicast message to {(2, 5), (2, 6), (1, 3), (1, 4), (1, 1), (1, 0), (1, 5), (3, 7)}. The source node prepares 3 multicast messages with the following header addresses: {(2, 5), (2, 6)} along the CU-path, {(3, 7)} along the MU-path and {(1, 0), (1, 1), (1, 3), (1, 4), (1, 5)} along MD-path. These messages are sent simultaneously along their paths. When the message header reaches (2, 5) along CU-path with addresses {(2, 5), (2, 6)}, it removes its address from the message header and makes a copy of the data then sends the header {(2, 6)} along the CU-path. When (2, 6) receives the header {(2, 6)} it copies the data and removes its address from the message header. Since the header message becomes empty the transmission along CU-path is terminated.

When the node (1,4) receives the header with the addresses {(1, 0), (1, 1), (1, 3), (1, 4), (1, 5)} it copies the data and removes its address from the message header and sends one message with header {(1, 5)} along CU-path and sends another message with header {(1, 3), (1, 1), (1, 0)} along CD-path. When the node (1, 5) receives the message header with {(1, 5)} it copies the data and terminates the transmission along the CU-path. The message with header {(1, 3), (1, 1), (1, 0)} along CD-path is sent to (1, 3) first and then to (1, 2) because there is no direct link between (1, 3) and (1, 1), finally, the message header is sent to (1, 0) and the message terminates. The node (3, 4) receives a message with header (3, 7) and sends it along CU-path to node (3, 7) directly (Fig. 5).

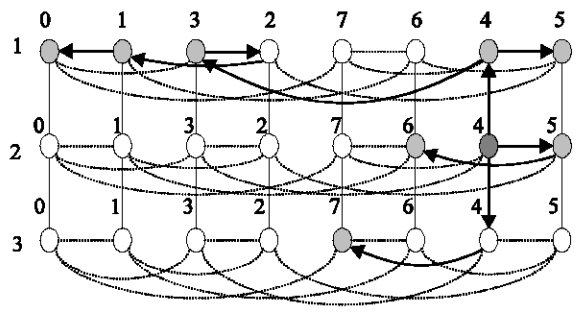


Fig. 5: Multicast routing example

**Analysis of the algorithms:** As it mentioned earlier the unicast routing algorithm works in a distributed fashion where each intermediate node on the shortest path executes the algorithm when it receives a message header. The only information needed by the intermediate node to decide the next neighbor falling on the shortest path is the address of the destination node. Therefore, the algorithm is optimal in space. Every step in the algorithm requires only a few (constant) clock cycles to execute (mostly bitwise operations). Therefore, the time complexity of the algorithm is O(1) and hence it's optimal in time.

It can be seen from the algorithm that the process to propagate the message to its destination is done by modifying the source node level number and/or modifying the source node hypercube address bit by bit at the source node and at each intermediate node. Therefore, if the source and the destination are at the same hypercube then the message takes at most O (log n) steps, since in the worst case the address of the source node will be transformed bit by bit to agree with the address of the destination node and the number of bits in the address is log n bits where n is the number of nodes in the hypercube of MH(m, n). If the source and destination nodes are at the same mesh then the message takes at most O(m) steps. Otherwise, the source and destination are on different meshes and on different hypercubes, which is the worst case, the message takes O(m+log n) steps. It should be noticed that (m+(log n)-1) also corresponds to the diameter of MH(m, n) which represents the maximum distance between any two nodes[6] and hence no algorithm can route a message between any two nodes in MH(m, n) in less than O(m+log n) steps.

The multicast routing algorithm also works in a distributed fashion where each intermediate node on the shortest path executes the algorithm when it receives a message header. The source node sorts the list of destinations addresses of size p, where p is the maximum number of destinations in a multicast message, which takes an order of O(p log p). The intermediate node performs a split operation of the list of the remaining destination addresses into a maximum of 3 subsets. Each subset is a message header containing destination addresses along the CU-path and CD-path and MD-path or MU-path. The split operation takes a maximum of 3 steps each takes an order of O(1). Therefore, the time complexity of the algorithm at each intermediate node is constant and hence it's optimal in time.

If the source and the destinations are at the same hypercube then the message takes at most O(log n) steps to reach all destinations, since in the worst case the maximum distance between any two nodes on the same

hypercube is log n, where n is the number of nodes in the hypercube of MH(m, n). If the source and destination nodes are at the same mesh then the message takes at most O(m) steps. Otherwise, the source and destinations are on different meshes and on different hypercubes, which is the worst case, the message takes O(m+log n) steps.

We have developed adaptive wormhole routing algorithms for unicast and multicast transmission in MH network. Both algorithms are deadlock-free and livelock-free. The algorithms are based on labeling the nodes in the MH in such a way to prevent deadlock and livelock problems. The use of multi-path makes the multicast algorithm suitable for all-port architecture, in which a router is connected to the local processor by multi input/output channels.

## REFERENCES


1. Bokhari, S., 1990. Communication Overhead on the Intel ipsc-860 Hypercube. Icase Interim Report 10, ICASE.
2. Ho, C.T. and L. Johnsson, 1986. Distributed Routing Algorithms for Broadcasting and Personalized Communication in Hypercubes. Proceedings of the 1986 Internatinal Conference on Parallel Processing, pp: 640-648.
3. Omari, M., 2003. Mesh-Hypercube: A Network Topology for Parallel Systems. Mu'tah Lil-Buhuth wad-Dirasat (Natural and Applied Sciences Series), Mu'tah University, 18: 37-60.
4. Omari, M. and H. Abu-Salem, 2001. Routing Strategies for Binary Tree-Hypercube Network. Al-Manarah Res., AL al-bayt University, 7: 27-48.
5. Raghavendra, C.S., A. Avizienis and M.D. Ercegovac, 1984. Fault tolerance in binary tree architectures. IEEE Transactions on Computers, C-33: 568-572.
6. Saad, Y. and M. H. Schultz, 1987. Data communication in hyperubes. Parallel and Distributed Computing, 9: 115-153.
7. Tsai, Y.-J. and P. McKinley, 1996. A broadcast algorithm for all-port wormhole-routed torus networks. IEEE Transactions on Parallel and Distributed Sys., 7: 876-885.
8. Lin, X., A-H. Esfahanian, P.K. Mckinely and A. Burago, 1993. Adaptive Wormhole Routing in Hypercube Multicomputers. Proc. 5th IEEE symposium on Parallel and Distributed Computing, Dallas, Texas, December 1993, pp: 72-79.
9. Stout, Q.F. and B. Wagar, 1990. Intensive hypercube communication. Parallel and Distributed Computing, 10: 167-181.
10. Schwiebert, L. and D.N. Jayasimha, 1995. Optimal fully adaptive minimal wormhole routing for meshes. Parallel and Distributed Computing, 27: 56-70.
11. Mohapatra, P., 1998. Wormhole Routing Techniques in Multicomputer Systems. ACM Computing Surveys, September 1998.
12. Ababneh, I. and A. Al-Obeid, 2001. Wormhole-Routed Multicast in 3D Mesh Architecture. Al-Manara J., Al al-Bayt University.
13. Bhuyan, L. N. and D. P. Agrawal, 1984. Generalized hypercube and hyperbus structures for a computer network. IEEE Transactions on Computers, C-32: 323-333.
14. Lan, Y., 1995. An adaptive fault-tolerant routing algorithm for hypercube multicomputers. IEEE Transactions on Parallel and Distributed Sys., 6: 1147-1152.
15. Al-Mahadeen, B. and M. Omari, 2003. Fault-Free Point-to-Point Routing in Mesh-Hypercube Networks. Proceedings of International Conference on Information Technology and Natural Sciences, Al-Zaytoonah University, Amman, Jordan, October 19-21.
16. Omari, M., 2000. Routing in Quad Tree-Hypercubes. The 15th ACM Symposium on Applied Computing (SAC 2000), Como, Italy, March, 2000, pp: 677-681.
17. Padmanbhan, K. and D. Lawrie, 1983. A class of redundant path multistage interconnection networks. IEEE Transactions on Computers, C-32: 1099-1108.
18. Yeung, K.H. and T.S. Yu, 1997. Selective broadcast data distribution systems. IEEE Transactions on Computers, 46: 100-104.
19. Gaughan, P., B. Dao, S. Yalamanchili and D. Schimmel, 1996. Distributed, deadlock-free routing in faulty, pipelined, directed interconnection networks. IEEE Transactions on Parallel and Distributed Sys., 45: 651-665.
20. Fleury, E. and P. Fraigniaud, 1998. Strategies for path-based multicasting in wormhole-routed meshes. Parallel and Distributed Computing, 53: 26-62.