Adaptive Algorithms for Wormhole-Routed Single-Port Mesh-Hypercube Network

Mahmoud Omari

Department of Computer Science
Applied Science University
Amman, 11921, Jordan

Abstract

In this research, adaptive wormhole routing algorithms are proposed. The algorithms presented for both unicast and multicast communication for single-port Mesh Hypercube (MH) network. MH introduced as a new interconnection network for parallel systems. The basic structure for it is a blend of both mesh and hypercube networks. It combines the desirable features of both the hypercube and the mesh, while at the same time overcoming their disadvantages. The proposed algorithms are based on node labeling technique, which is shown to prevent deadlock problem without requiring virtual channels. The multicast routing algorithm efficiency affected by the order in which the destinations are visited. The presented algorithm capable of ordering m destinations in a multicast message in O(m log m) time complexity. Both unicast and multicast routing algorithms make use of the node labeling technique to route the messages through the network without creating cycle, and hence, avoid any cyclic dependencies between nodes that may create deadlock.

Keywords: *Wormhole, Adaptive, Routing, Multi-path, Single-port, Mesh-Hypercube.*

1. Introduction

The efficient communication between nodes is critical to multicomputer systems performance. Routing algorithms are very important to high performance multicomputer systems, therefore, it has received substantial consideration [1][2][9][10][13][15][16][18]. Many of the applications that run on multicomputers depend greatly on communication time. Any routing algorithm states the path that a message must go through so that it can reach specific destination. A dedicated routed is attached to each node in most multicomputer systems that takes care of all communication tasks. Usually, all necessary information needed for routing are contained in the message header.

Routing algorithms can be either deterministic or adaptive. In deterministic routing, a single path is decided between the source node and destination node and that path is determined by the source and destination addresses. In adaptive routing, in the other hand, the algorithm finds multiple paths that can exist between the source and the destination. These alternative paths are decided depending on the network conditions. Most computer networks face congestion in network channels or the existence of nodes that are faulty and/or faulty channels [8]. In addition, routing algorithms are classified as either minimal or non-minimal. Minimal routing algorithm selects one shortest path between the source-destination pair. When the packet traverses the network channels designated by the shortest path will bring it closer to the destination. A non-minimal routing algorithm does not necessarily need that the packet follow only shortest paths [3] [4] [17].

Messages can be unicast (point-to-point), multicast, or broadcast. In unicast communication, the source node needs to send a message to a single node destination. In multicast communication, in the other hand, the message will delivered to number of destination nodes. In broadcast, a message is sent to all nodes in the network. Both unicast and broadcast are special cases of multicast [18]. In multicasting, the routing algorithm 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 each destination.

In modern multicomputer systems, it has been shown that the most used routing technique is the wormhole routing [15]. Some of the modern systems that use wormhole routing are the NCUBE, Cray T3D, Intel Paragon, Caltech MOSAIC, and MIT J-machine [15]. In this technique, every message is divided into a number of packets that are also divided into some flits. The routing information are usually contained in the header flits of the packet that are followed by the remaining flits of the message throughout the network. The router at an intermediate node forwards the coming flits to a neighboring node if there is an available output channel. The remaining flits follow in a pipeline fashion.

One of the main advantages of the wormhole switching becomes clear from the pipeline fashion that the message follows and the time it takes a message to reach its destination is not affected by the distance between the source and the destination nodes. In addition, every nodes in the network needs only one flit-size buffer since the message flits move one by one through the network. In the other hand, to improve routing performance some systems employ several buffers to store several flits [17]. The use of this small size storage at each node has a major effect on the reduction of both the cost and the size of the system. One disadvantage of such switching technique comes from the fact that if there exist some channel contention in the network then the header flit cannot advance further through the network which results in blocking all remaining flits of the message along the path. In wormhole routing, a blocked message may lead to blocking other messages in the network. If the blocked messages form a chain, in which each message is waiting for another message in the chain, then those messages are said to be in a deadlock. One of the main problems in wormhole switching is to prevent the occurrence of deadlocks. Therefore, some techniques prevents some messages from using all the paths that are available to avoid creating cycle of waiting messages in the system [6] [17]. Therefore, a very important issue in wormhole switching is the choice of a routing algorithm.

Dimension ordering routing is one of the deterministic routing techniques that is mostly used to avoid deadlocks [5] [11] [19]. This technique accomplishes its task by imposing a strict order on the network dimensions traversed by each message. Some adaptive routing techniques can avoid deadlocks by maintaining additional channel in the network that are virtual [6] [12]. This is accomplished by dividing the network into several virtual subnetworks that are disjoint, and each subnetwork contains the channels that constitute all the paths with minimum cost between a node and some other nodes. One more adaptive routing technique is called the *turn model*. This technique involves analyzing the cycles that are possible every time a message change its direction. Hence, by prohibiting certain "turns" all the possible cycles can be avoided, resulting in an algorithm that is partially adaptive.

A deterministic multicast algorithm for hypercube networks that is deadlock-free have been previously proposed in [14]. The algorithm is based on a *path-based* approach, in which a multicast path is established between the source and all the destinations. The path consists of all the successive channels starting at the source and go across every destination. A path in a multicast message is denoted by a list of addresses ($s, d_1, d_2, \ldots, d_m$), where s is the source and d_i's are the destinations in the order they are to be reached. Path-based multicast is implemented by placing the ordered list of destinations in the header of the packet. Once the header of the packet arrives at d_i's router, the router removes d_i from the list and forwards the successive flits to local node and in the direction of destination d_{i+1}. If the header arrives at the router of a non-destination node, the router only forwards the packet in the direction of the next destination in the list. In all cases, the original message eventually arrives at all destination nodes in the network.

In this research, deadlock-free adaptive wormhole routing algorithms are proposed for Mesh-Hypercube networks. Specifically, the algorithms are designed for unicast and multicast messaging. Section 2 gives a brief description of the Mesh-Hypercube networks. The system model is described in section 3. Section 4 introduces the unicast routing algorithm. The multicast routing algorithm presented in section 5. Finally, section 6 contains the conclusion.

2. The Mesh-Hypercube Network

The MH introduced in [16] as an interconnection network characterized by a two-tuple (m, n), where m defines the number hypercube subnetworks each of which consists of 2^n nodes. The corresponding nodes in each hypercube subnetwork are connected together forming one-dimensional meshes of m nodes each. For an MH(m, n), the total number of nodes is equal to $m \times 2^n$. Each node in the network is recognized by its row number, L, and its cube binary address, X, denoted by the pair (L, X), where $0 \le L \le m-1$, $X = x_{n-1} \ldots x_1 x_0$, and $x_i \in \{0, 1\}$. Assume $\|w\|$ denote the number of 1's in a binary address w. Two nodes (L_1, X) and (L_2, Y) are connected if and only if X = Y and $|L_2 - L_1| = 1$, or $L_1 = L_2$ and $\|X \oplus Y\| = 1$, where $\oplus$ is the bitwise exclusive-or operation on binary numbers. Figure 1 shows a MH(3, 2) network, the dashed lines represent hypercube links, and the solid lines represent 1-dimensional meshes.

An n-cube, or a hypercube, consists of 2^n nodes each of which has a unique address consists of n-bit binary. Two nodes x and y are said to be adjacent in a hypercube of n nodes if and only if $\|x \oplus y\| = 1$, that is, their addresses representation in binary differ in exactly one bit. The length of a path between any two nodes in a multicomputer is equal to the number of edges contained in the path. For any pair of nodes x and y, the distance between them, denoted *dist*(x, y), represents the shortest-path length between x and y. The distance between two nodes x and y in a hypercube is represented by the Hamming distance between them. The Hamming distance is denoted by $H(x, y) = \|x \oplus y\|$. That is, if $H(x, y) = d$, then x′s and y′s binary addresses differ in exactly d bit positions. The distance between two nodes (L1, X) and (L2, Y) of a MH, is defined as $|L2 - L1| + H(X, Y)$.

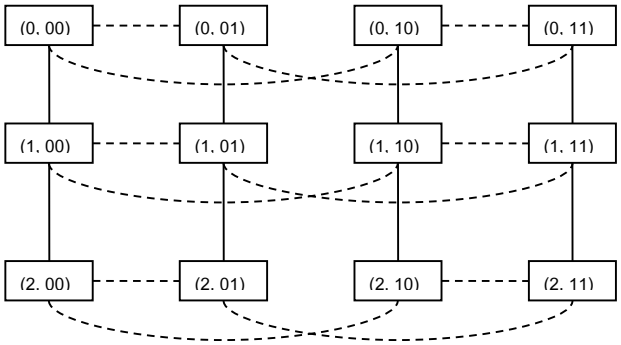

Figure 1. MH(3, 2) Network.

3. System Model

In the system, as shown in figure 2, there is a separate router handles communication among nodes. The router is connected to the neighboring routers through several input/output *external channels*. The topology of the network is defined by the connection pattern formed by the external channels. In this configuration, any two neighboring routers can send messages to each other simultaneously. When several messages arrive at a router then the router can simultaneously relay those messages through the output channels if each messages needs to go on different output channel. There are one or more pair of *internal* input/output channels that connect the router it the local processor or memory. This paper assumes the existence of a single pair of internal channels, one for input and the other is for output. This class of architecture that is shown in figure 2 is called "one-port communicator architecture" [7]. This architecture requires that the processer sends and/or receive messages in a sequential fashion. Many systems exist that employ this type of architecture [15]. Some systems may increase their capabilities in communication by having more internal channels.

Several metrics can be used to evaluated a multicomputer system. One important metric is the communication latency that is equals to the sum of several delay factors [15]. The first factor is the *start-up latency*, which is the required time by the system for handling the message at the source node and at the destination node. The network latency is the period that starts when the message enters the network until the last piece of the message enters the destination node. Finally, the *blocking time* is the sum of all delays faced by the message during its trip through the network, such as delays at intermediate routers due to contentions in network channels. Another metric used to measure the performance of multicast communication is the *multicast latency*, which is the total time needed to deliver the message to all the destinations. That is, the time from sending the message to its first destination until it is received by last destination. The multicast latency can be greatly reduced by employing a system architecture called "multi-port communicator architecture", in which the router is connected to its local processor/memory through several pairs of internal input/output channels.

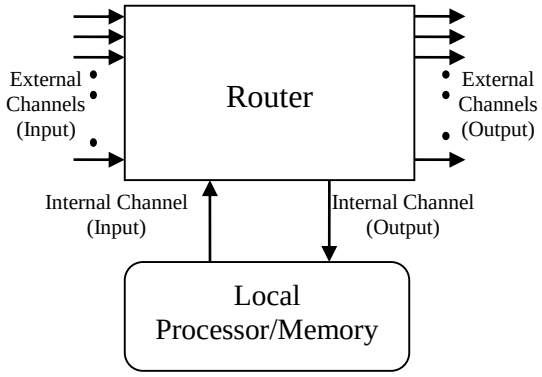

Figure 2. A generic node architecture.

To deliver a message to multiple destinations, a routing algorithm may use more than one multicast path. Multipath routing algorithms that are deadlock-free have been devised for Mesh-Hypercubes in [3] and [4]. In this paper, we are interested in single-path multicast routing. This type of routing requires that a single message at time can be sent/received by a node. Moreover, minimal unicast routing will be considered in this paper, in which a message is routed throughout a shortest path. While in multicast routing, the path found by the algorithm may not follow a shortest path that covers all the destinations of a message, therefore, multicast routing algorithm is non-minimal. Two issues must be considered when developing an adaptive path-based multicast routing algorithm. First, to prevent deadlocks then the algorithm used for unicast and the one used for multicast should both use the same routing technique. The second issue is related to problem of ordering the addresses of the destination node that lie on the path. Since the wormhole routing has a pipeline characteristic, randomly ordering the destinations and perform adaptive unicast routing between each pair may result in deadlock. Therefore, to have a deadlock-free adaptive routing between each pair of subsequent nodes in a multicast message, the destinations must be ordered in such a way that achieves this goal, and at the same time using as few channels as possible.

4. Unicast Routing

The adaptive unicast routing algorithm presented here based on a strategy adapted from [14] in which a labelling function used to label the Hypercube nodes in a hypercube-based multicomputer system. The labelling technique guarantees freedom from deadlock. The labelling enforces the order in which nodes are visited during communication. This strategy labels all the nodes in the system in such a way to avoid cycles between any two nodes, and hence, preventing deadlock [14]. A labeling function, *l*, is defined that maps the nodes of a MH(*m*, *n*) to a set of labels [0, $m(2^n-1)$]. A node with address (*r*, *x*) has a label defined by the function $l(r, x) = r2^n + \sum_{i=0}^{n-1} (c_i \bar{x}_i + \bar{c}_i x_i 2^i)$, where $c_{n-1}=0$, $c_{n-j} = x_{n-1} \oplus x_{n-2} \oplus \ldots \oplus x_{n-j+1}$, for $1<j\leq n$, and $\bar{x}_i$ is the complement of x_i. Notice that, for any pair of nodes (*r1*, *x*) and (*r2*, *y*) of a MH, $l(r1, x) \neq l(r1, y)$ if $(r1, x) \neq (r2, y)$. In effect, this labeling method follows a Hamiltonian path of the graph that represents the hypercube subnetwork of MH [14].

Figure 3 shows an example of labelling a MH(3, 3). Remember that there are two unidirectional channels in opposite directions connect every two adjacent nodes. These unidirectional channels are classified into four classes. The first class referred to as cube-high-channel (cH-channel) that includes all channels that goes from a node with a label of lower value to a node with label of higher value on the same hypercube subnetwork. The second class includes all other channels in the hypercube and referred to as cube-low-channel (cL-channel). The third class referred to as mesh-high-channel (mH-channel) which includes all channels that goes from a node with a label of higher value to a node with label of lower value on the same mesh subnetwork ; the fourth class includes the other channels on the same mesh and referred to as mesh-low-channel (mL-channel). The following definitions explain how the routing algorithm establishes the routing paths.

Definition 1: A path between two nodes *x* and *y*, *p*(*x*, *y*), is called an up path (U-path) if $l(x) < l(u_1) < l(u_2) < \ldots < l(u_k) < l(y)$, where u_i's are intermediate nodes in the path between *x* and *y*. A path, *p*(*y*, *x*), is a down path (D-path) if $l(y) > l(u_k) > l(u_{k-1}) > \ldots > l(u_1) > l(x)$.

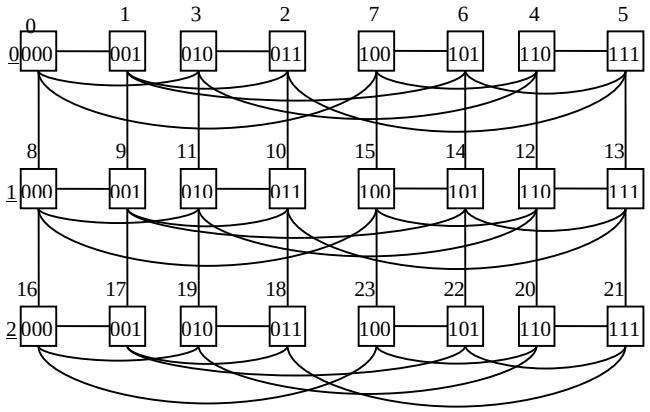

Figure 3. Node Labeling of a MH(3, 3).

Definition 2: A path between two nodes *x* and *y*, *p*(*x*, *y*), is called an up-down path (UD-path) if there exist a node *v* such that the path *p*(*x*, *v*) is a U-path and the path *p*(*v*, *y*) is D-path.

From the above two definitions, we identify three classes of paths: U-path, D-path, and UD-path. In some cases a UD-path could have an empty U-path or an empty D-path. Therefore, any U-path and D-path can also considered UD-paths. Figure 4 shows examples of three paths, the path *p*(1, 5) is a U-path, the path *p*(12, 0) is a D-path, and the path *p*(17, 6) is a UD-path. It can be seen that if a path *p*(*x*, *y*) is a UD-path from *x* to *y*, then the path *p*(*y*, *x*) is also a UD-path. Clearly, if there are *n* different shortest paths between any node *x* and any other node *y*, then there must exist the same number of shortest paths from *y* to *x*.

It can be seen that using the labeling function *l*, there will be a shortest U-path from *x* to *y* if $l(x)<l(y)$. Similarly, there is a shortest D-path from *y* to *x* if $l(y)>l(x)$.

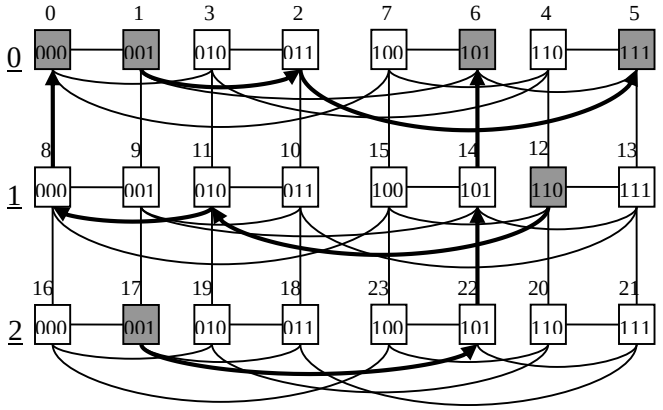

Figure 4. Example of paths in a MH(3, 3).

Recall that there exist four classes of unidirectional channels, a U-path consists of cH-channels and/or mH-channels and a D-path consists of cL-channels and/or mL-channels. Therefore, a message that follows a UD-path moves first on a cH-channels (if needed) and over cL-channels (if needed), then along mH-channels or mL-channels. It will be apparent that we will have a deadlock-free routing approach if all messages go through UD-paths.

The unicast routing algorithm, shown in figure 5, will be executed at the source node and at every intermediate node on the path. The *T* parameter is assigned one of the following values: cH, cL, mH, mL, or S. These values indicate that if the message has arrived on a cH-channel, an cL-channel, mH-channel, mL-channel, or the source node s, respectively. The algorithm first routes the message along cH-channels and/or mH-channels up until the current-node's label value is greater than the label of destination node. At this stage, if the shortest path to the destination node consists of cH-channels and/or mH-channels then the message can continue along those channels. When the message arrives at any node with a label greater than that of the destination and does not have cH-channels or mH-channels that lie on the shortest path to *d*, the message may begin to follow cL-channels and/or mL-channels, which will continue until it reaches the destination.

Procedure Adaptive-Unicast-Routing(*x*, *d*, *T*)
x: current node,
d: destination node,
T: tag
Returns: output channel (*x*, *y*) used in forwarding the message
Begin
Step1: if x=d then direct message into local processor and exit.
Step2: if *T* = S or *T* = cH or mH then
Find any available cH-channel (*x*, *y*) or mH-channel (*x*, *y*) that lies on a shortest path to the destination.
If a channel found then send message to node y and stop, else go to Step3.
Step3: if *l*(*x*) > *l*(*d*) then
Find any available cL-channel (*x*, *y*) or mL-channel (*x*, *y*) that lies on a shortest path to destination.
If a channel found then send message to node y and stop, else go to Step4.
Step4: message cannot advance, therefore, wait for a period of time then goto Step2.
End Adaptive-Unicast-Routing

Figure 5. Adaptive unicast routing algorithm.

Figure 6 gives examples of the operations of the Adaptive-Unicast-Routing algorithm. Assume that a source node (1, 110) of label 12 wants to send a message to node (0, 001) of the label 1. According to Step2 of the algorithm and at the source node 12, the algorithm selects any cH-channel, such as channel (12, 13), that is on a shortest path between the source and the destination. Alternatively, by Step3, the algorithm may select either the cL-channel(12, 11) or the mL-channel(12,4). Assume the cH-channel(12, 13) was chosen by Step2, then the message will be sent to node 13. When the message arrives at node 13, then Step3 may select either the cL-channel (13, 10) or the mL-channel (13, 5). To show all alternative paths that may be selected by the algorithm for this example, figure 6 shows all four possible shortest paths between source node label 12 and destination node 1.

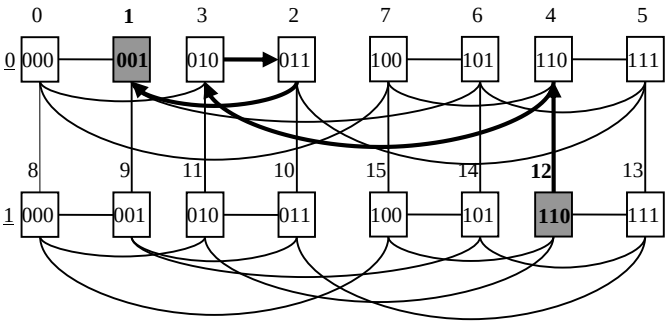

Figure 6 (a) A shortest UD-path from (1, 110) to (0, 001). path (12, 4, 3, 2, 1).

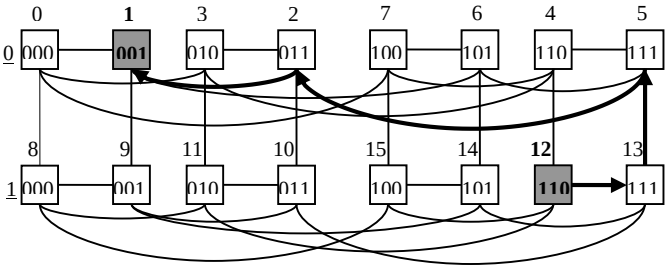

Figure 6 (b) A shortest UD-path from (1, 110) to (0, 001). path (12, 13, 5, 2, 1)

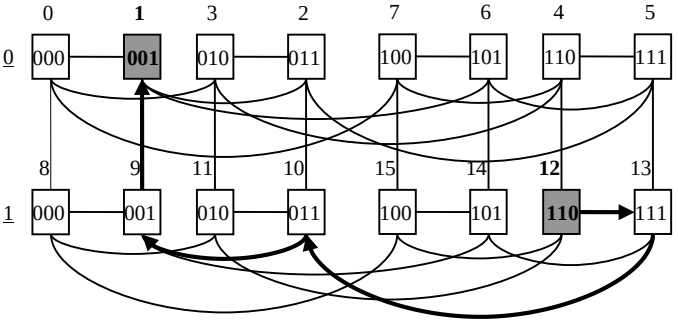

Figure 6 (c) A shortest UD-path from (1, 110) to (0, 001). path (12, 13, 10, 9, 1)

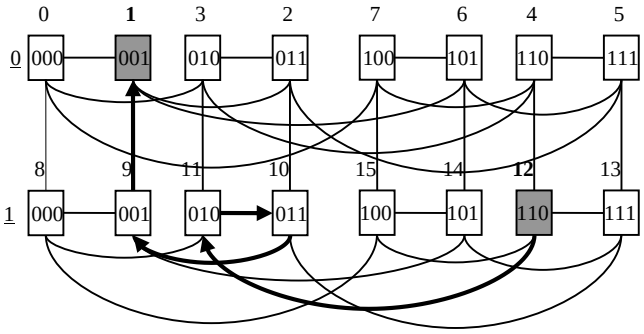

Figure 6 (d) A shortest UD-path from (1, 110) to (0, 001). path (12, 11, 10, 9, 1)

5. Multicast Routing

As shown in the previous section, the routing algorithm is said to be deadlock-free if the algorithm routes all the message through UD-paths. The routing strategy explained in the previous section will be extended to develop a multicast routing algorithm. In multicasting, the multicast path is usually represented by a list that contains all the destination nodes and the source node is at the beginning of the list. The challenge in multicast routing is to find a suitable order the destinations according to their labels so the resulted routing path will be a UD-path.

Procedure Multicast-Message-Header-Construction(*s*, *D*)
s: source node,
D: destination set.
The algorithm constructs message header that contains the ordered list of destinations, *L*, as part of the header of the message.

Begin
Step1: Add the source node *s* to *D* then sort the node in D by their labels. Assume that the set *D* after sorting contains the elements (s, d_1, …, d_m).
Step2: Set $L = [d_m]$.
Step3: For $k = m$ downto *0* do:
Let the first node in *L* be l_{k+1} and let the last node in *L* be l_m .
if $dist(d_k, l_{k+1}) < dist(l_m, d_k)$ then set $L = [\, d_k\, L]$; else set $L = [L\ d_k]$
End for.
Step4: Assume that $L = [l_0, l_1, \ldots, l_m\ ]$.
if $l_m = s$ then set $L = [\ l_m, \ldots, l_1, l_0\ ]$. Return a message header containing *L*.
End Multicast-Message-Header-Construction

Figure 7.Multicast message header construction algorithm.

Definition 3: Given the aforementioned labeling function and a set D_k of nodes, where $D_k=\{d_0, d_1, \ldots, d_k\}$, then a permutation $[p_0, p_1, \ldots, p_k\ ]$ of the set D_k is called an up-down list (UD-list) if there exists an index e, $0 \le e \le k$, such that $l(p_j) < l(p_{j+1})$, for $0 \le j < e$, and for $e \le j < k$, $l(p_j) > l(p_{j+1})$. The length of the UD-list is the sum of $dist(p_j, p_{j+1})$ for $0 \le j < k$.

It can be seen that, if a permutation $[p_0, p_1, \ldots, p_k\ ]$ is a UD-list, then the permutation $[p_k, \ldots, p_1, p_0]$ is also a UD-list. It is clear that the list that contains all of the destinations that included in a multicast UD-path should be a UD-list. One algorithm, presented in this section, will be used to build a list of the destinations that are included in a multicast message and that list is UD-list. The algorithm is very simple and can be implemented easily and its running time is in the order of ($m \log m$), where m is the number of destinations in a multicast message. One drawback of the algorithm is that, it cannot always find an optimal path.

Given a multicast message with source node s and the destinations $d_1, d_2, \ldots, d_m$. For simplicity, assume that $l(s) < l(d_i)$ for $0 \le i \le m$; the case when the label of the source is greater than the destinations will be considered later. Note that the source node, s, must be the first element in any UD-list for any multicast message. The algorithm given in figure 7, is responsible of arranging the destinations in the list and then place list in the header of the message to accomplish the operation of multicasting that is path-based multicasting. First, the algorithm sorts the source and destinations in a nondecreasing order of their labels as keys. Assume that the resulted list after sorting is the list (s, d_1, …, d_m). The algorithm at Step2 starts by constructing a partial list that only contains one destination, which is d_m. In Step3 and during the first loop-iteration the node d_{m-1} is added either before or after d_m that is already in the UD-list. The choice of adding before or after a certain node label depends on whether either way will result in a permutation of shortest length. In the next loop-iteration, d_{m-2} is added either at the beginning or at the end of the UD-list that contains d_m and d_{m-1} in some order. The algorithm continues in this manner until adding the last node, which is the source node s. Now we will consider the case when the source node label is greater that the destination labels. Assume that in Step1 that sorts the nodes we got the sorted list $D = (d_1, d_2, \ldots, d_k, s, d_{k+1}, \ldots, d_m)$ with $l(d_1) < \ldots\ l(d_k) < l(s) < l(d_{k+1}) < \ldots < l(d_m)$. Clearly, the algorithm will place these nodes in decreasing order at the end of the list.

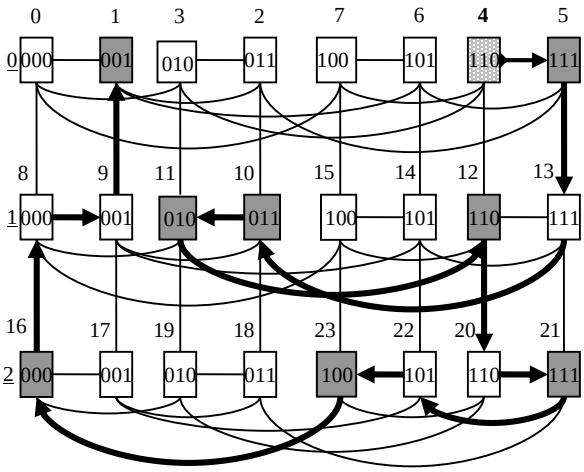

Figure 9. A multicast on a MH(3, 3), source node is 4.

Procedure Adaptive-Multicast-Routing (x, M)
x: current node address,
M: Message with destinations list $DL = (d_i, .., d_m)$.
The procedure forwards the message to next destination in DL.

Begin
Step1: if $x = d_i$ then remove d_i from DL and send a copy of the message to local node.
Step2: if destination list DL is empty then this concludes message transmission and exit.
Step3: Let d_i be the first node in DL.
if $l(x)<l(d_i)$ then select any available channel (x, y), such that $l(x)<l(y)\leq l(d_i)$.
if $l(x)>l(d_i)$ then select any available channel (x, y), such that $l(x)>l(y)\geq l(d_i)$.
Step4: Forward the message to node y with the list DL in its header.
End Adaptive-Multicast-Routing

Figure 8.Adaptive multicast routing algorithm.

Figure 8 gives the algorithm that is executed at the source node and at every node on the path. If the current node label is less than that of the first destination in the list of destinations then the algorithm selects an available cH-channel or mH-channel; otherwise, it uses a cL-channel or a mL-channel. Notice that, the set channels that are selected by the algorithm form a UD-path. The algorithm worst-case running time is in the order of ($m2^n$) for a MH(m, n). Notice that this worst-case running time is linear with respect to the total number of nodes in the network. An example of routing path found by the algorithm is given in Figure 9. The example assumes that node label 4 is the source and the destinations in the multicast message are {1, 5, 10, 11, 12, 16, 21, 23}. The UD-list resulted from applying the header construction procedure is [**4**, 5, 10, 11, 12, 21, 23, 16, 1], which has a total path length of 13.

It is clear that for multicast communication, the multicast path should be a UD-path in order to avoid any deadlocks. Therefore, all paths that starts from the source to the first destination must be a UD-path (that is, a U-path or a D-path) and all other paths between successive destinations must be UD-paths.

6. Conclusion

The paper presented two adaptive wormhole routing algorithms for both unicast and multicast communication for single-port mesh-hypercube networks. It was shown that both algorithms are deadlock-free. We have presented an algorithm, that takes $O(m \log m)$ time complexity, which finds a suitable ordering of m destinations included in a multicast message. The single path that is found by the algorithm follows a UD-path fashion. This algorithm is very appropriate for one-port architecture, in which, the router at each node is connected to its local processor/memory by a pair of input/output channels. This paper is the first to describe adaptive routing algorithms for both unicast and multicast communication for single-port wormhole-routed mesh-hypercube network.

Acknowledgments

The author is grateful to the Applied Science Private University, Amman, Jordan for the financial support granted to cover the publication fee of this research article.

References

[1] Adhikari, N., Sethi, K., Mohanta, A., and Tripathy, C.R., "Metastar: A massive large scale parallel interconnection network," in 2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC), 2011, pp. 359-364.

[2] Aggarwal, Rinkle Rani, "Design and Performance Evaluation of a New Irregular Fault-Tolerant Multistage Interconnection Network," International Journal of Computer Science Issues, Vol. 9, No. 2, 2012, pp. 108-113.

[3] Al-Mahadeen, Bassam and Mahmoud Omari, "Adaptive Wormhole Routing in Mesh-Hypercube Network", Journal of Applied Sciences, Vol. 4, No. 4, 2004, pp. 568-574.

[4] Al-Mahadeen, Bassam and Mahmoud Omari, "A Broadcast Algorithm for All-Port Wormhole-Routed Mesh-Hypercube Network", Information Technology Journal, Vol. 3, No. 3, 2004, pp. 283-289.

[5] Chiu, Ge-Ming, "The Odd-Even Turn Model for Adaptive Routing, "*IEEE Transactions on Parallel and Distributed Systems*, Vol. 11, No. 7, July 2000, pp. 729-738.

[6] Duato J., "A Necessary and Sufficient Condition for Deadlock-free Adaptive routing in Wormhole Networks," In *Proceedings of the International Conference on Parallel Processing*, Vol. 1, 1994, pp. 142-149.

[7] Fleury, E., and P. Fraigniaud, "Strategies for Path-Based Multicasting in Wormhole-Routed Meshes," *Journal of Parallel and Distributed Computing*, Vol. 53, No. 53, 1998, pp. 26-62.

[8] Gaughan. P., B. Dao, S. Yalamanchili, and D. Schimmel, "Distributed, Deadlock-Free Routing in Faulty, Pipelined, Directed Interconnection Networks," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 45, No. 6, 1996, pp. 651-665.

[9] Ghandriz, Zahra and Esmaeil Khasraghi, "A New Routing Algorithm for a Three-Stage Clos Interconnection Networks," International Journal of Computer Science Issues, Vol. 8, No. 5, 2011, pp. 309-313.

[10] Huyn, S., O. Jae-Chul and L. Hyeong-Ok, "Multiple Reduced Hypercube (MRH): A New Interconnection Network Reducing Both Diameter and Edge of Hypercube," International Journal of Grid and Distributed Computing, Vol. 3, No. 1, 2010, pp. 19-30.

[11] Iwama, K., Y. Kambayashi, and E. Miyano, "New Bounds for Oblivious Mesh Routing," *Journal of Graph Algorithms and Applications*, Vol. 5, No. 5, 2001, pp. 17-38.

[12] Jayasimha, D. N., L. Schwiebert, D. Manivannan, and J. A. May, "A Foundation for designing deadlock-free routing algorithms in wormhole networks," *Journal of the ACM (JACM)*, Vol. 50, No. 2, March 2003, pp. 250-275.

[13] Li, Yamin, Shietung Peng, and Wanming Chu, "Hierarchical Dual-Net: A Flexible Interconnection Network and its Routing Algorithm," International Journal of Networking and Computing, Vol. 2, No. 2, 2012, pp. 234–250.

[14] Lin, X., A-H. Esfahanian, P. K. Mckinely, and A. Burago, "Adaptive Wormhole Routing in Hypercube Multicpmputers," In *Proceedings of the Fifth IEEE Symposium on Parallel and Distributed Computing*, Dallas, Texas, December 1993, pp. 72-79.

[15] Mohapatra, P., "Wormhole Routing Techniques in Multicomputer Systems," *ACM Computing Surveys*, Vol. 30, No. 3, September 1998.

[16] Omari, M., "Mesh-Hypercube: A Network Topology for Parallel Systems," *Mu'tah Lil-Buhuth wad-Dirasat (Natural and Applied Sciences Series), Mu'tah University*, Vol. 18, No. 1, 2003, pp. 37-60.

[17] Schwiebert, L. and D. N. Jayasimha, "Optimal Fully Adaptive Minimal Wormhole Routing for Meshes," Journal of Parallel and Distributed Computing, Vol. 27, No. 1, 1995, pp. 56-70.

[18] Tsai, Y.-J., and P. McKinley, "A Broadcast Algorithm for All-Port Wormhole-Routed Torus Networks," IEEE Transactions on Parallel and Distributed Systems, Vol. 7, No. 8, 1996, pp. 876-885.

[19] Wu, Jie, "A deterministic fault-tolerant and deadlock-free routing protocol in 2-D meshes based on odd-even turn model," In Proceedings of the 16th international conference on Supercomputing, New York, New York, USA, 2002, pp. 67-76.

Mahmoud Omari received his BSc. degree in Computer Science from Yarmouk University, Irbid, Jordan in 1984, MSc. and Ph.D. degrees in Computer Science from George Washington University, Washington, DC and Illinois Institute of Technology, Chicago, IL, in 1988 and 1992, respectively. He is currently an Associate Professor at the Department of Computer Science of the Applied Science Private University, Amman, Jordan. His research interests including Interconnection Networks, Wormhole Routing, Interval Routing, Mobile Agents, Instance-Based Learning, and Information Retrieval.