Information Technology Journal 3 (3): 283-289, 2004
ISSN 1682-6027

A Broadcast Algorithm for All-port Wormhole-routed Mesh-hypercube Network

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

Abstract: The aim of this research was the development of a Broadcast wormhole routing algorithm 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. In this research it is introduced a wormhole routing algorithm for broadcast communication. The algorithm takes advantage of the distance insensitivity of wormhole-routing and the presence of multiple ports between processors and their routers. The algorithm is based on dividing the network into mesh and hypercube subnetworks. The hypercube subnetwork at each level is also divided into subcubes of dimension two where each subcube consists of four nodes. Broadcast transmission in MH takes into consideration two issues: deadlock-free and transmissions terminate when the message reaches to all nodes in the network.

Key words: Wormhole routing, mesh-hypercube, broadcast, hypercube, mesh, deadlock, all-port

INTRODUCTION

The design of efficient routing algorithms has received significant attention due to its importance to high performance in multiprocessor systems[1-5]. 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[5-7].

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[8]. In wormhole routing, a message is divided elementary units called flits, each of a few bytes for transmission and flow control. The header flits of a message contain the routing information and the data flits of the message follow the header flits through the network in a pipelined fashion. 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. 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[6,9]. The data flits contain no routing information, so if a channel transmits the header of a message, it must transmit all the remaining flits of the same message before transmitting flits of another message. When the header is blocked the data flits are blocked in-situ. 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[5,7,10,11].

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[5,6,9,11].

Messages in distributed-memory multiprocessors can be unicast (Point-to-point), multicast, or broadcast. In unicast communication, a source node requires that a

Corresponding Author: Bassam Al-Mahadeen, Arab Academy for Banking and Financial Sciences, P.O. Box 09131, Amman 24911, Jordan Tel: (+962) 6 550 2900 Fax: (+962) 6 523 7834 E-mail: bassam@mutah.edu.jo

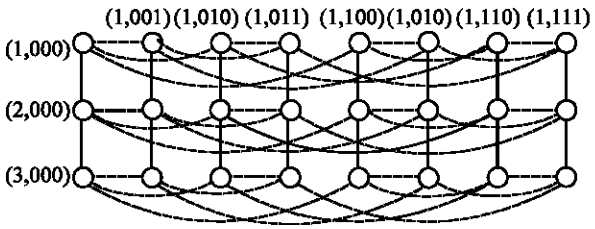

Fig. 1: MH (3,8) Network

(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)→(1,111)→(1,111)→(3,111)→(3,111)
(1,000)→(2,000)→(3,001)→(3,001)→(3,011)→(3,111)

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

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. Broadcast, refers to the delivery of the same message originating from a given source to all network nodes[7,11-13]. Broadcasting is important in many real-world parallel applications found in the areas of Science and Engineering. Both unicast and broadcast are special cases of multicast[6,12,14].

There are two issues to be taken in consideration when designing wormhole routing for broadcasting transmission: deadlock and termination of transmission. A deadlock occurs when some packets cannot advance toward their destination because the channels 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[5,10]. Termination of transmission must take place when the broadcast message reaches to all nodes in the network. This can be satisfied by preventing intermediate nodes from sending replicated messages to other nodes that have received a copy of the message; otherwise the network will be heavily loaded with the same message.

In this study a broadcast algorithms is suggested for wormhole routing in Mesh-Hypercube networks. The algorithm tries to reduce the broadcast latency by reducing the number of message-passing steps, i.e. the number of exchanges required to perform a broadcast operation. The algorithm benefits from the all-port assumption, in which a node can send several copies of the same message along all its out-ports simultaneously in order to reduce the start-up latency and communication latency. The algorithm is shown to be deadlock-free and the termination condition is satisfied for the same message.

The Mesh-Hypercube Network: The MH introduced is an interconnection network characterized by a two-tuple (m, n), where, m defines the number of nodes forming a mesh and n defines the number of nodes in a hypercube[1]. 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, $1 \le L \le m$, $X = x_w \ldots x_1 x_0$, $0 \le w \le (\log n)-1$ 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 solid lines represent 1-dimensional meshes.

An important feature in a multiprocessor system is the existence of multiple paths between every two processors[1,2,4,5,15-17]. 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[5,18]. 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[4,5]. Figure 2 shows all possible shortest paths between node (1,000) and node (3,111) in MH(3,8).

System model: It is supposed that the MH is characterized by an additional property described hereunder. Attached to each processor there is an 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[5,11-13,19,20] although most commercial systems use a one-port architecture. The goal is to assess the potential benefits of this costlier architectural feature. The all-port architecture allows considerable overlap among messages sent in succession from a given node, even though the start-up latency is relatively high[20].

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 send 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 to $t_p*d+l*t_f$ where, t_p is the switching delay of one router and t_f 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 formula when t_s is the start-up time:

$$T_c = t_s+(t_p*d+l*t_f)+\text{blocking time}$$

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 required to handle the message at both the source and destination nodes[5,11,13,14]. The broadcast latency is defined as the time from when the source processor begins to send the first copy of the message until the last destination receives its copy of the message[12,13,20].

The broadcast routing algorithm: The routing algorithm presented in this paper is based on partitioning the MH network into mesh and hypercube networks. The source node of the broadcast message, sends two copies of the message simultaneously a long its mesh connections, one to the up-middle level and the other to the down-middle level. The source node also sends (n/4)-1 copies of the messages along its hypercube connections. When an intermediate node receives the message it performs the following: First, if the message is received along a mesh connection, the node will do the same works as the source node by sending two copies of the message simultaneously along the mesh connections, one to the new up-middle level and the other to the new down-middle level. It also sends (n/4)-1 copies along its hypercube connections in the same manner as the source node. Second, if the message received along a hypercube connection, the node is concerned only in sending the message to its three neighbors.

An attractive feature of the algorithm is that each node will receive only one copy of the message. Hence, the transmission of the message terminates when all nodes receive the message. Therefore, the algorithm will not form a heavy load on the network. Detailed descriptions of the source node algorithm and of the intermediate node algorithm are given below.

Algorithm 1: Source-node-algorithm:

// This algorithm is executed only by the source node
Input: Source node address (Ls, S) in MH(m, n)
// where, Ls: Level number of the source, S: Cube address of the source
Output: sends n/4+4 messages simultaneously with format:
(Source Level No., Source Cube Address, Current Node level No., Level bound for intermediate node, Data)
Ld = Ls div 2
// Where, Ld is the level number of the next intermediate node between Ls and 1
Lu = (m+Ls+1) div 2
// Where, Lu is the level number of the next intermediate node between Ls and m
If Ls ≠ 1 then send (Ls, S, Ls, 1, data) to node (Ld, S)
If Ls ≠ m then send (Ls, S, Ls, m, data) to node (Lu, S)
// Source node sends (n/4 –1) messages simultaneously to adjacent subcubes on the same level
// The * in the message format means that the value is not important while
// broadcasting in the cube
For i = 1 to n/4 – 1
    Send message (Ls, S, Ls, *, data) to node (Ls, (S+4 * i) mod n))
Endfor
// Source node sends 2 messages to adjacent nodes simultaneously in the local subcube
For i = 1 to 2
    Send message (Ls, S, Ls, *, data) to node (Ls, S+i)
Endfor
// send one message to the last node in the local subcube in a separate step
Send message (Ls, S, Ls, *, data) to node (Ls, R+3)
End Source-Node-Algorithm

Algorithm 2: Intermediate-node-algorithm:

// Received message of the format (Ls, S, Lx, X, Lb, data) from node (Lx, S)
// Where, Lx is the level number of the node from which the message is received
// Where, X is the cube address of the node from which the message is received
// and Lb is a lower bound or an upper bound value used by the current node to compute
// the level number of the next intermediate nodes
Let Lr be the level number of the current node
Let R be the cube address of the current node
Copy the message to local processor
If Lr ≠ Lx then Call Cube-Broadcast //message received from node on different level
If Lr = Lx and $R_{\log n-1} \ldots R_2 \neq X_{\log n-1} \ldots X_2$ then
    Call Local-Cube-Broadcast // message received from node on the same level
                                // and on different subcube
If Lr = Lx and $R_{\log n-1} \ldots R_2 = X_{\log n-1} \ldots X_2$ then
Terminate and exit // message received from node on the same level
                                // and on the same subcube; no further transmission
If Lr < Lx then Call Send-To-Low-Level // message received from node on higher level
If Lr > Lx then Call Send-To-High-Level // message received from node on lower level
End Intermediate-Node-Algorithm

Procedure cube-broadcast:

// node sends (n/4 –1) messages simultaneously to adjacent (n/4 – 1) subcubes on the same level
For j = 1 to n/4 – 1
    Send message (Ls, S, Lr, *, data) to node (Lr, (R+4 * j) mod n))
Endfor
Call Local-Cube-Broadcast
End Cube-Broadcast
Procedure Local-Cube-Broadcast
// node sends 2 messages to adjacent nodes simultaneously in the local subcube
For j = 1 to 2
    Send message (Ls, S, Lr, *, data) to node (Lr, R+j)
Endfor
// send one message to the last node in the local cube in a separate step
Send message (Ls, S, Lr, *, data) to node (Lr, R+3)

End local-cube-broadcast:

Procedure Send-To-Low-Level
H = Lx -1
Ld = (Lb+Lr - 1) div 2
Lu = (H+Lr+1) div 2
If Lr ≠Lb then send message (Ls, S, Lr, Lb, data) to node (Ld, S)
If Lr ≠H then send message (Ls, S, Lr, H, data) to node (Lu, S)
End Send-To-Low-Level
Procedure Send-To-High-Level
L = Lx+1
Ld = (L+Lr - 1) div 2
Lu = (Lb+Lr+1) div 2
If Lr ≠L then send message (Ls, S, Lr, L, data) to node (Ld, S)
If Lr ≠Lb then send message (Ls, S, Lr, Lb, data) to node (Lu, S)
End Send-To-High-Level

Due to the regularity of the mesh-hypercube structure, a distributed routing scheme can be implemented without global information. The broadcast message contains, in addition to data, the source address, the current node level number, message length and Lb, where Lb is a lower bound or an upper bound value used by the current node to compute the level number of the next intermediate nodes. The interprocessor message traffic of a node gets redistributed into two categories, the mesh communication and the hypercube communication. At each intermediate node, the value of Lb is used to compute the values of Lu and Ld to decide the next intermediate nodes. The message may be sent to other node(s) on the same mesh at different levels and/or to nodes on the same hypercube. The message is sent along mesh connections only if the computed value of Lu or Ld is different from the current node level. The message is sent to nodes on the same hypercube only if the intermediate node receives the message from a different level or from a node located on different subcube in the same level. The value of Lb is updated at each intermediate node to guarantee that the message will not be sent again to nodes that have received a copy of the same message.

Proposition 1: The algorithm is deadlock free for the same message.

Since each node sends a copy of the message only after receiving it and each node in the next transmission step sends the message to its neighbors that did not receive a copy of the message, no link in the network used by more than one message in the same communication step. So, there will not be any cyclic dependencies between nodes and the deadlock will never occur for the same message.

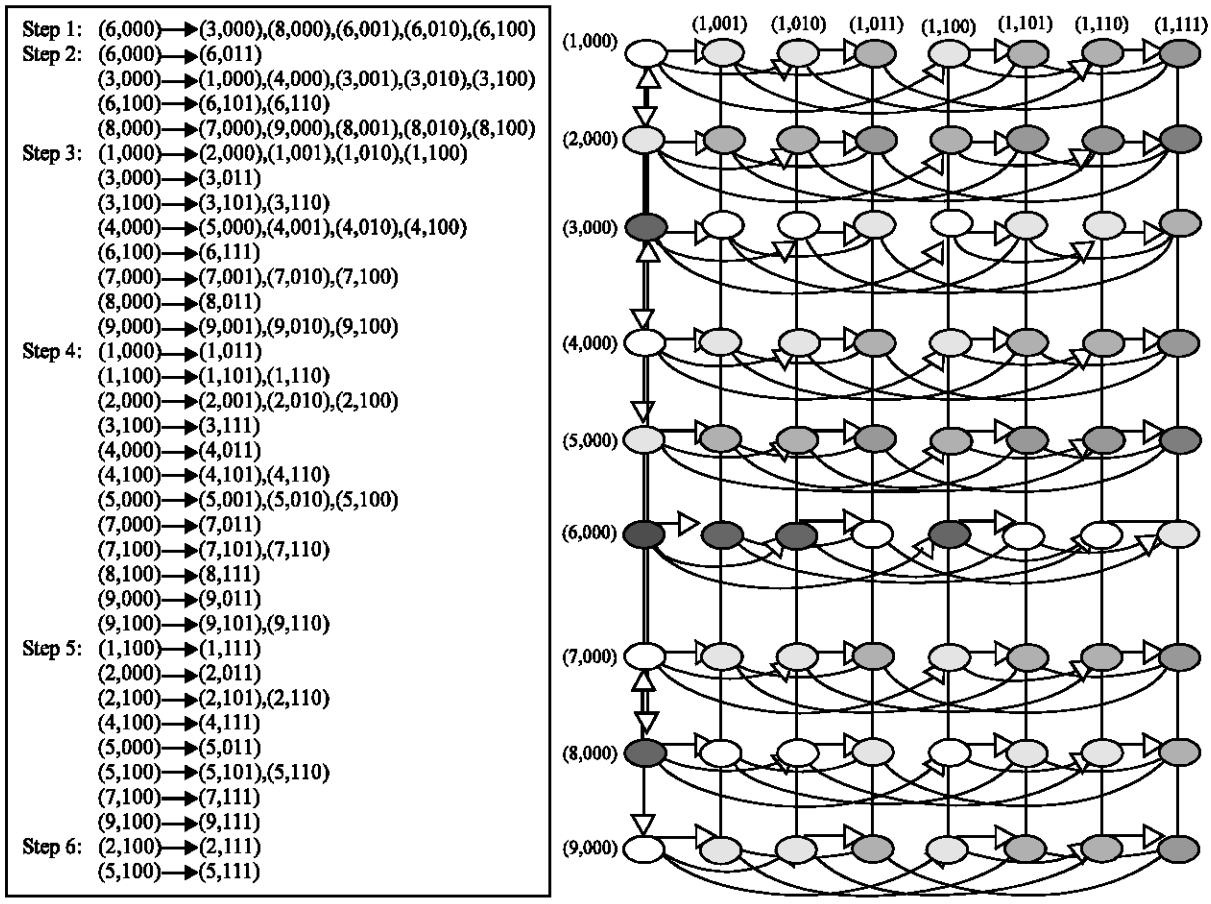

Fig. 3: Broadcasting example initiated by node (6,000)

Proposition 2: The Broadcast routing algorithm satisfies the termination condition.

In MH the transmission of a message occurs along mesh connections and/or hypercube connections. In the algorithm the hypercube on each level is divided into n/4 subcubes of dimension 2. An intermediate node sends the message received through a mesh connection to nodes on other mesh levels only if the current mesh level differs from the computed value of Lu or Ld. If an intermediate node receives the message along a hypercube connection it sends a copy of the message to its neighbors only if the message is received from a node on different subcube. If the above conditions do not hold then the intermediate node consumes the message and terminates transmission.

An illustrative example: Suppose that the node (6,000) sends a broadcast message to all the nodes in MH(9,8) network. The following steps along with Fig. 3 show how the algorithm performs the transmission of the message.

First: when the source node executes the Source-Node-Algorithm, it computes the down-lower middle level (Ld) and the up-middle level (Lu) as follows: Ld = (5+1)/2 = 3, Lu = (7+9)/2 = 8. Then the source node sends 5 messages simultaneously in the first transmission step to nodes: (3, 000), (8, 000), (6, 001), (6, 010) and (6, 100), then it sends a message to node (6, 011) in the second transmission step.

Second: when the Intermediate-Node-Algorithm is performed at an intermediate node, the following communication steps take place:

1. Communication step 2: Node (3, 000) sends 5 messages simultaneously to nodes: (1, 000), (4, 000), (3, 001), (3, 010) and (3, 100). Node (6, 100) sends two messages to nodes: (6, 101) and (6, 110). Node (8, 000) sends also 5 messages to nodes: (7, 000), (9, 000), (8, 001), (8, 010) and (8, 100).

2. Communication step 3: Node (1, 000) sends 4 messages to nodes: (2, 000), (1, 001), (1, 010) and (1, 100). Node (3, 000) sends one message to node (3, 011). Node (3, 100) sends 2 messages to the nodes (3, 101) and (3, 110). Node (4, 000) sends 4 messages to the nodes: (5, 000), (4, 001), (4, 010) and (4, 100). Node (6, 100) sends one message to node (6, 111). Node (7, 000) sends 3 messages to nodes: (7, 001), (7, 010) and (7, 100). Node (8, 000) sends one message to node (8, 011). Node (8, 100) sends 2 messages to nodes (8, 101) and (8, 110). Node (9, 000) sends 3 messages to the nodes: (9, 001), (9, 010) and (9, 100).
3. Communication step 4: Node (1, 000) sends one message to node (1, 011). Node (1, 100) sends 2 messages to nodes: (1, 101) and (1, 110). Node (2, 000) sends 3 messages to nodes: (2, 001), (2, 010) and (2, 100). Node (3, 100) sends one message to node (3, 111). Node (4, 000) sends one message to node (4, 011). Node (4, 100) sends two messages to nodes: (4, 101) and (4, 110). Node (5, 000) sends 3 messages to nodes: (5, 001), (5, 010) and (5, 100). Node (7, 000) sends one message to node (7, 011). Node (7, 100) sends 2 messages to nodes: (7, 101) and (7, 110). Node (8, 100) sends one message to node (8, 111). Node (9, 000) sends one message to node (9, 011). Node (9, 100) sends two messages to nodes: (9, 101) and (9, 110).
4. Communication step 5: Node (1, 100) sends one message to node (1, 111). Node (2, 000) sends one message to node (2, 011). Node (2, 100) sends 2 messages to the nodes: (2, 101) and (2, 110). Node (4, 100) sends one message to node (4, 111). Node (5, 000) sends one message to node (5, 011). Node (5, 100) sends 2 messages to nodes: (5, 101) and (5, 110). Node (7, 100) sends one message to node (7, 111). Node (9, 100) sends one message to node (9, 111).
5. Communication step 6: Node (2, 100) sends one message to node (2, 111). Node (5, 100) sends one message to node (5, 111).

After the sixth communication step the message reaches all nodes and transmission terminates.

Analysis of the algorithms: As mentioned earlier the broadcast algorithm works in a distributed fashion where each intermediate node executes the algorithm when it receives a message. The only information needed by the intermediate node to decide the next intermediate nodes on the same mesh are the values of Lu and Ld. Therefore, the algorithm is optimal in space. Every step in the algorithm requires only a few (constant) computations and bitwise operations. Therefore, the time complexity of the algorithm is constant and hence it's optimal in time.

It can be seen in the Source-Node-Algorithm that the source node benefits from the all-port assumptions by sending as many messages as there are outgoing links simultaneously in the first transmission step. In later transmission steps as seen in the Intermediate-Node-Algorithm that each intermediate node can send as many messages as there are outgoing links simultaneously to nodes that did not receive a copy of the message. The transmission of the message to its destinations is done by modifying the value of Lb that is included in the message header and/or modifying the intermediate node hypercube address bit by bit. Therefore, the broadcast of a message takes at most O(log m) transmission steps to guarantee that the message reaches to all the network levels. It takes also O(log n/4) extra transmission steps to deliver the message to all hypercube nodes in the final levels which are computed by Lu and/or Ld. Hence, the algorithm takes O(log m+log(n/4)) transmission steps.

Conclusions: We have developed wormhole routing algorithm for broadcast transmission in MH network. It was shown that the algorithm is deadlock-free and satisfies the termination condition. The algorithm based on dividing the network into mesh and hypercube subnetworks. The hypercube at each level is also divided into 4-nodes subcubes. Broadcast transmission in MH takes O(log m+log(n/4)), where, (log m) is the transmissions steps needed to deliver the message to all nodes on the mesh subnetwork of the source node and log(n/4) is the transmission steps needed to deliver the message to all nodes in the same hypercube subnetwork.

REFERENCES

1. Omari, M., 2003. Mesh-Hypercube: A network topology for parallel systems. Mu'tah Lil-Buhuth wad-Dirasat Natural and Applied Sciences Series, Mu'tah Univ., 18: 37-60.
2. Omari, M. and H. Abu-Salem, 2001. Routing strategies for binary tree-hypercube network. Al-Manarah Res. J., AL al-bayt Univ., 7: 27-48.
3. Saad, Y. and M.H. Schultz, 1987. Data Communication in Hyperubes. J. Parallel and Distributed Computing, 9: 115-153.
4. Al-Mahadeen, B. and M. Omari, 2004. Fault-free point-to-point routing in mesh-hypercube networks. Proc. Intl. Conf. on Inform. Technol. Natural Sci., Al-Zaytoonah University, Amman, Jordan, October, pp: 19-21.

5. Al-Mahadeen, B. and M. Omari, 2004. Adaptive wormhole routing in mesh-hypercube network. Applied Sciences. (In press).
6. 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, pp: 72-79.
7. Al-Dubai, A.Y. and M. Ould-Khaoua, 2001. Coded path routing: A new approach to broadcasting in 3-D meshe. Proc. 20th IEEE Int. Performance, Comput. Comm. Conf. (IEEE-IPCCC'2001). IEEE, Phoenix North, Phoenix, Arizona, USA, April, pp: 155-162.
8. Stout, Q.F. and B. Wagar, 1990. Intensive hypercube communication. J. Parallel and Distributed Computing, 10: 167-181.
9. Schwiebert, L. and D.N. Jayasimha, 1995. Optimal fully adaptive minimal wormhole routing for meshes. J. Parallel and Distributed Computing, 27:56-70.
10. Mohapatra, P., 1998. Wormhole routing techniques for directly connected multicomputer systems. ACM Computing Surveys, 30: 374-410.
11. McKinley, P.K., Y. Tsai and D.F. Robinson, 1995. Collective communication in wormhole–routed massively parallel computers. IEEE Computer, 28: 39-50.
12. Tsai, Y.J. and P. McKinley, 1996. A broadcast algorithm for all-port wormhole-routed torus networks. IEEE Tran. Parallel and Distributed Sys., 7: 876-885.
13. McKinley, P.K. and C. Trefftz, 1993. Efficient broadcast in all-port wormhole routed hypercubes. Intl. Conf. Parallel Processing, St. Charles, Illinois, August, 1993, 2: 288-291.
14. Ababneh, I. and A. Al-Obeid, 2003. A wormhole-routed multicast in 3D mesh architecture. Al-Manarah J., Al al-Bayt Univ., 9: 23-47.
15. Lan, Y., 1995. An adaptive fault-tolerant routing algorithm for hypercube multicomputers. IEEE Tran. Parallel and Distributed Sys., 6: 1147-1152.
16. Omari, M., 2000. Routing in quad tree-hypercubes. The 15th ACM Symposium on Applied Computing (SAC2000), Como, Italy, March, pp: 677-681.
17. Padmanbhan, K. and D. Lawrie, 1983. A class of redundant path multistage interconnection networks. IEEE Transactions on Computers, 32: 1099-1108.
18. Yeung, K.H. and T.S. Yum, 1997. Selective broadcast data distribution systems. IEEE Transactions on Computers, 46: 100-104.
19. Fleury, E. and P. Fraigniaud, 1998. Strategies for path-based multicasting in wormhole-routed meshes. J. Parallel and Distributed Computing, 53: 26-62.
20. Tsai, Y.J. and P.K. McKinkey, 1994. An extended dominating node approach to collective communication in all–port wormhole–routed 2d meshes. In Proc. Scalable High–Performance Computing Conf., October, pp: 199-206.