http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 6 (5): 623-632, 2007
ISSN 1812-5638

Deadlock-Free Multicast Wormhole Algorithms in 3-D Mesh Multicomputers

Amnah El-Obaid and Wan-Li Zuo
Department of Computer Science and Technology, Jilin University, Changchun, China

Abstract: A mesh network is a popular architecture, which has been implemented, in many multicomputer systems. The essential pattern in new multicomputer generations is the multicast wormhole pattern, which corresponds to one-to-many communication in which one source sends the same message to multiple destination nodes. In wormhole routing, a message is divided into flits and flits of one message may be spread out among several nodes. Deadlock in the interconnection network occurs when there is a cyclic dependency for consumption channels. This study presents an efficient two algorithms, GTDBTPM, GTDMPM that implement multicast communication to find a deadlock-free wormhole routing in general three-dimensional networks without any restrictions in the number of rows or columns in the network. The introduced algorithm GTDBTPM is designed such that can send messages to any number of destinations within two start-up communication phases (in x- y- directions) using Binary search (in z direction); hence the name General Three-Dimension Binary Two-Phase Multicast. Another introduced algorithm GTDMPM is designed such that can send messages to any number of destinations within multiple start-up communication phases; hence the name General Tree-Dimension Multi-Phase Multicast. A simulation study has been conducted that compares the performance of these multicast algorithms under dynamic network traffic conditions in a 3-D mesh. The results indicate that the GTDMPM algorithm offers performance advantages over GTDBTPM algorithm.

Key words: Multicasting communication, Wormhole routing, multicomputers, 3-D mesh topology, deadlock-free algorithm

INTRODUCTION

In a multicomputer network, processors often need to communicate with each other for many reasons, such as data exchange and event synchronization. Collective communication patterns, such as broadcast and multicast, involve a group of intercommunicating nodes. Utilizing collective communication not only simplifies but also increases the functionality and efficiency of the parallel tasks. A performance analysis of direct networks has shown that with wormhole switching technique, lower dimensional networks offer improved latency and throughput results for the same network bandwidth (Agrawal, 1991; Dally, 1990). Recent interest in multicomputer systems is therefore concentrated on two or three-dimensional mesh and torus networks. Such technology has been adopted by the Intel Touchstone DELTA (Anonymous, 1990), MIT J-machine (Nuth and Dally, 1992), Intel Paragon (Foschia *et al*., 1997; Almasi and Gottlieb, 1994), Caltech MOSAIC (Athas and Seitz, 1988) and Cray T3D and T3E (Lessler and Schwazmeier, 1993; Anonymous, 1995).

In Wormhole-routed networks, packets are divided into flits. A flit is the smallest unit of information that a channel can accept or refuse. Wormhole routing operates by advancing the head of a packet directly from incoming to outgoing channels (Dally and Seitz, 1986). The transmission from the source node to the destination node is done through a sequence of routers. All flits in the same packet are transmitted in order as pipelined fashion. Only the header flit knows where the packet is going and the remaining data flits must follow the header flit. Once the header flit gains access to a channel, the current message owns that channel until the tail flit passes through it and resigns ownership of the channel. If the header encounters a channel already in use, it is blocked until the channel is freed. An important primitive among collective communication operations is multicast communication. Multicast is defined as sending a single message from a source node to a set of destination nodes. The performance of multicast communication is measured in terms of its latency in delivering a message to all destinations. In wormhole-routed networks, the communication latency consists of three parts, start-up latency network latency and blocking latency (Libeskind-Hadas *et al*., 1997). The start-up latency is the time required to start a message, which involves operation system overheads. The network latency consists of channel propagation and router latencies, i.e., the elapsed time after the head of a message has entered the network

Corresponding Author: Amnah El-Obaid, Department of Computer Science and Technology, Jilin University, Changchun, China

at the source until the tail of the message emerges from the network at the destination. The blocking latency accounts for all delays associated with contention for routing resources among the various worms in the network. In wormhole routing, contiguous flits in a packet are always contained in the same or adjacent nodes of the network. This can cause difficulties, as possibility of deadlock arises. Deadlock in the interconnection network occurs when a set of messages is blocked forever because each message in the set holds one or more resources needed by another message in this set (Hwang, 1993). No communication can occur over the deadlocked channels until exceptional action is taken to break the deadlock. Many deadlock-free routing algorithms have been developed for wormhole communications networks (Dally and Seitz, 1986; Kumar *et al*., 2001; Jianxi, 2002; Ahmed Al-Dubai *et al*., 2006; Moharam *et al*., 2000; Lin *et al*., 1994).

A generic architecture of a wormhole router is shown in Fig. 1. The router provides communication services to the host processor. The input/output internal channels to/from the router are usually referred as injection/consumption channels. The number of injection/consumption channels implies the number of messages that can be sent/received concurrently by the processor. Several pairs of external channels connect the router to neighboring routers. The interconnection of external channels among routers defines the network topology (Lin *et al*., 1994). A crossbar switch within the router allows simultaneous transmission of messages between different input and output channels.

The mesh-connected topology (Dally, 1989; Koeninger *et al*., 1994) is one of the most thoroughly investigated network topologies for multicomputer systems. Mesh-connected topologies also called k-ary-n dimensional meshes have an n-dimensional grid structure with k nodes in each dimension such that every node is connected to two other nodes in each dimension by a direct communication. Designing multicast protocols and routing algorithms to meet deadlock-free depends on the topology of the network used to interconnect nodes in the multicomputer. In this study, discussion is restricted to the 3-D mesh topology with bi-directional channels. An m (rows) x n (columns) x r (layers) 3-D mesh comprises mnr nodes interconnected in a grid fashion, with each node having at most six neighbors. The 3-D mesh topology is represented with a mesh graph M (V, E) in which each node in V (M) corresponds to a processor and each edge in E (M) corresponds to a communication channel.

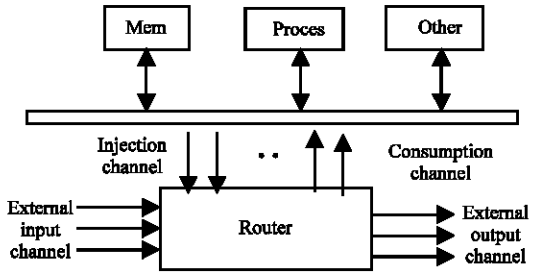


Fig. 1: The basic node

Definition 1: An m×n×r non wraparound 3-D mesh graph is a directed graph M (V, E), where the following conditions exist:

$$V(M)=\{(x,y,z)\mid 0\le x<n,0\le y<m,0\le z<r\}\text{ and}$$

$$E(M)=\{[(x_i,y_i,z_i),(x_j,y_j,z_j)]\mid (x_i,y_i,z_i),(x_j,y_j,z_j)\in V(G),\ \text{and}\ |x_i-x_j|+|y_i-y_j|+|z_i-z_j|=1\}$$

In this study, deadlock-free multicast communication in 3-D mesh multicomputer networks that use wormhole routing is addressed. In the proposed algorithms two heuristic multicast routing algorithms based on a path-based model are presented. These algorithms are shown to be deadlock-free. In order to compare these routing algorithms, their operation was simulated under a variety of network conditions.

RELATED WORKS

Dual-path algorithm: A network partitioning strategy based on Hamiltonian paths is fundamental to the deadlock-free routing schemes. A Hamiltonian path visits every node in a graph exactly once (Lin *et al*., 1994), a 2-D mesh has many Hamiltonian paths. In this algorithm, each node u in a multicomputer is assigned a label, L (u). In a network with N nodes, dual-path assigns a label for each node based on the position of that node in a Hamiltonian path, where the first node in the path is labeled 0 and the last node in the path is labeled N-1. Figure 2 shows such a labeling in a 4×4 mesh. The label assignment function L for an m×n mesh can be expressed in terms of the x-, y-coordinates of nodes as follows:

$$L(x,y)=\begin{cases} y*n+x, & \text{if } y \text{ is even}\\ y*n+n-x-1 & \text{if } y \text{ is odd}\end{cases}$$

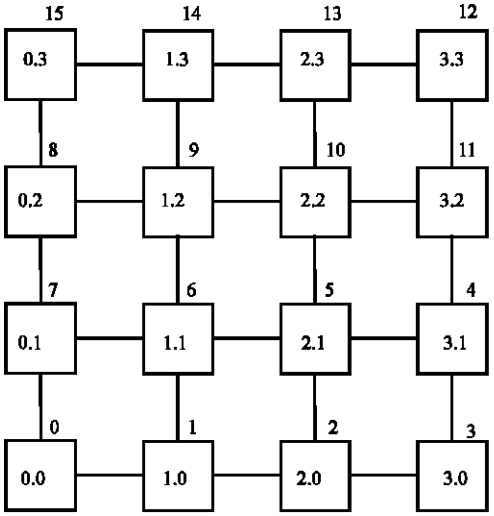


Fig. 2: The labeling of 4×4 mesh

The labeling effectively divides the network into two subnetworks. The high-channel subnetwork contains all of the channels whose direction is from lower-labeled nodes to higher-labeled nodes and the low-channel network contains all of the channels whose direction is from higher-labeled nodes to lower-labeled nodes. At the source node, dual-path algorithm divides the network into two subnetworks, N_U and N_L, where every node in N_U has a higher label than that of the source node and every node in N_L has a lower label than that of the source node. The message transmission in dual-path is made according to the equation that presented by Lin *et al.* (1994). One such routing function, defined for a source node u and destination node v, is defined as R (u, v) = w, such that w is a neighboring node of u and, if L (u) < L (v), then we have the following equation:

$$L(w) = \max \{L(z) \mid L(z) \le L(v) \text{ and } z \text{ is a neighboring node of } u\}$$

or, if $L(u) > L(v)$, then we have the following equation:

$$L(w) = \min \{L(z) \mid L(z) \ge L(v) \text{ and } z \text{ is a neighboring node of } u\}$$

In dual-path algorithm, a source node divides the destination set D into two subsets, D_U and D_L, where D_U contain the destination nodes in N_U and D_L contain the destination nodes in N_L. The messages will be sent from the source node to the nodes in D_U using the high-channel network and to the destination nodes in D_L using the low-channel network.

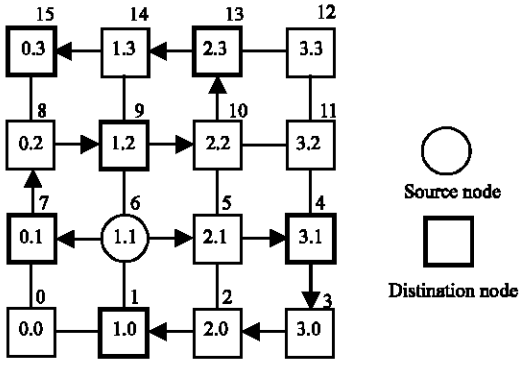


Fig. 3: The routing pattern of dual-path algorithm

Sort the destination nodes in D_U, using the L value as the key, in ascending order. Sort the destination nodes in D_L, using the L value as the key, in descending order. Construct two messages, one containing D_U as part of the header and the other containing D_L as part of the header. The source sends two messages into tow disjoint subnetworks N_U and N_L.

The dual-path routing algorithm uses a distributed routing method in which the routing decision is made at each intermediate node. Upon receiving the message, each intermediate node determines whether its address matches that of the first destination node in the message header. If so the address is removed from the message header, the message is copied and sent together with its header to the above (below) neighboring using the routing function R. In case where the intermediate node is not a destination, it sends the message together with its header to the above (below) neighboring using the routing function R. If the sets of the destination nodes are not empty, the algorithm continues according to the previous method. As proved by Lin *et al.* (1994) dual-path algorithm is deadlock-free.

Consider the example shown in Fig. 3 for a 4×4 mesh topology labeling using a Hamiltonian path. The source node labeled 6 initiates a multicast to the destination set D = {15, 7, 1, 9, 13, 4}. The dual-path algorithm splits and sorts, D in two subsets D_L = {4, 1} and D_U = {7, 9, 13, 15}. The routing pattern is shown with bold lines in Fig. 3.

THE PROPOSED ALGORITHMS

In this study two algorithms are presented. First algorithm GTDBTPM is based on splitting the 3-D mesh network into set of layers. Second algorithm GTDMPM is based on splitting the destination set in subsets (here groups of columns, rows and diagonals).

General three-dimension binary tow-phase multicast (GTDBTPM): Here, first algorithm is based on splitting

the 3-D mesh network into set of layers. The 3-D mesh network can be shown as set of layers; each layer represents a 2-D mesh network. Figure 5 represents 3×3×3 mesh (each node in the mesh is represented by its integer coordinate (x, y, z), we can show that there are three layers of 3×3 2-D mesh, the z coordinate for the first mesh is 0, for the second mesh is 1 and for the third mesh is 2. In fact, if we have m×n×r 3-D mesh, then we have r layers of m x n 2-D mesh. The z coordinate for the first layer is 0, for the second layer is 1 and for the last layer is r-1. The GTDBTPM algorithm divides network in z coordinate into two subnetworks N_{+z} and N_{-z}. Subnetwork N_{+z} contains all upper diagonal channels of source with addresses [(x, y, z), (x, y, z+1)] and subnetwork N_{-z} contains all lower diagonal channels of source with addresses [(x, y, z), (x, y, z - 1)]. Suppose that the coordinate of the source node u_0 is represented by (x_0, y_0, z_0) and D represent the destination-set. The simple idea of this algorithm is as follow: -

Step 1: The source split the D into two subsets D_{zs} and D_{zd}, where D_{zs} contain the destination nodes which their z coordinates are equal z_0 and D_{zd} contain the remaining destination nodes (which their z coordinates are not equal z_0). Formally, the subsets are described by the following expression:

$$D_{xs} = \{(x,y,z) \mid (x,y,z) \in D, (0 \le x < n), (0 \le y < m), (z = z_0)\},$$

and

$$D_{xd} = \{(x,y,z) \mid (x,y,z) \in D, (0 \le x < n), (0 \le y < m), (z <> z_0)\}$$

Step 2: The source sends the message to destination nodes in D_{zs} using the dual-path multicast algorithm for 2-D mesh, which is explained earlier.

Step 3: The algorithm save the z coordinates of D in one-dimensional array and sort it in ascending order.

Step 4: Here, proposed algorithm is based on splitting array into sub-arrays. First (at level 0) the algorithm locates the middle element of the array and compares it with z_0, if the middle element of the array is greater than z_0, the source split D_{zd} into, s_{zdr} and s_0, where s_0 contain the destination nodes, which their z coordinates are grater than or equal middle element and s_{zdr} contain the remaining destination nodes of D_{zd}. Otherwise the source split D_{zd} into, s_{zdr} and s_0, where s_0 contain the destination nodes, which their z coordinates, are smaller than or equal middle element and s_{zdr} contain the remaining destination nodes of D_{zd}. The source sends subset s_0 to middle node, which is represented by its integer coordinate (x_0, y_0, middle). Steps 1 and 2 are repeated (let D = s_0 and source = middle node). (At level 1) the algorithm locates the middle element of the above two sub-arrays (two half of array)

Step 5: Step 4 is repeated for each quarter of the original array. Search of middle element continue until each sub-array consists of one element. The performance of the GTDBTPM algorithm depends on the number of middles for each sub-array. This number can be found via the binary search, where a binary search with n elements requires only log n steps.

To present the main idea of this algorithm, let us consider this example. Suppose we have 6×6×6 3-D mesh, the source is (2, 1, 0) and D = {(0, 1, 2), (1, 3, 3), (3, 2, 1), (0, 2, 0), (3, 3, 2), (0, 2, 3), (2, 1, 0), (5, 1, 5), (5, 3, 4)}. The z coordinate of a source is 0 and z coordinates of D are {1, 2, 3, 4, 5}, which are saved in array in ascending order. The GTDBTPM algorithm would perform in z coordinate as shown in Fig. 4

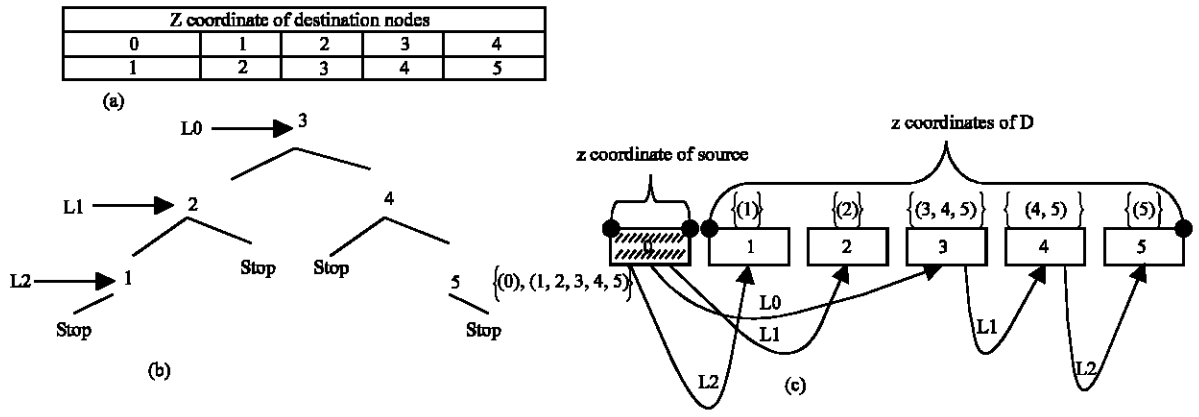


Fig. 4: The performance of GTDBTPM algorithm for 6×6×6 mesh in z coordinates, (a) array sorted, (b) binary search represented as tree and (c) binary search represented as linear array

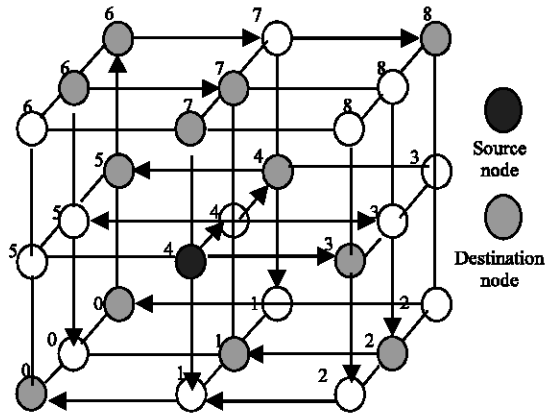


Fig. 5: The routing pattern of GTDBTPM algorithm

The following example shown the operation of the algorithm. Consider the 3×3×3 mesh given in Fig. 5, where each 2-D mesh is labeled independently by Hamiltonian paths. The source node is (1,1,0) and destination set is: D = {(0,0,0), (1,0,1), (2,0,1), (0,0,2), (2,1,0), (0,1,2), (1,1,2), (1,2,0), (0,2,1), (1,2,1), (0,2,2), (2,2,2)}

(At level 0): At the source node (1, 1, 0), the following destination set: D = {(0,0,0), (1,0,1), (2,0,1), (0,0,2), (2,1,0), (0,1,2), (1,1,2), (1,2,0), (0,2,1), (1,2,1), (0,2,2), (2,2,2)}
Is divided into the following two subsets:

D_{zs} = {(0,0,0), (2,1,0), ((1,2,0)}
D_{zd} = {(1,0,1), (2,0,1), (0,2,1), (1,2,1), (0,0,2), (0,1,2), (1,1,2), (0,2,2), (2,2,2)}

The message will be sent to the destination nodes in D_{zs} using dual-path algorithm, the destination nodes in D_{zd} will be sent to middle node, which is represented by coordinates (1,1,1) using N_{+z}.

(At level 1): At the intermediate node (1,1,1) (which is a source on its 2-D mesh), the following destination set: D = {(1,0,1), (2,0,1), (0,2,1), (1,2,1), (0,0,2), (0,1,2), (1,1,2), (0,2,2), (2,2,2)}
Is divided into the following two subsets:

D_{zs} = {(1,0,1), (2,0,1), (0,2,1), (1,2,1)}
D_{zd} = {(0,0,2), (0,1,2), (1,1,2), (0,2,2), (2,2,2)}

The message will be sent to the destination nodes in D_{zs} using dual-path algorithm, the destination nodes in D_{zd} will be sent to middle node, which is represented by coordinates (1,1,2) using N_{+z}.

(At level 2): At the intermediate node (1,1,2) (which is a source on its 2-D mesh), the following destination set: {(0,0,2), (0,1,2), (1,1,2), (0,2,2), (2,2,2)}
Is divided into the following two subsets:

D_{zs} = {(0,0,2), (0,1,2), (1,1,2), (0,2,2), (2,2,2)}
D_{zd} = φ

The message will be sent to the destination nodes in D_{zs} using dual-path algorithm. The routing pattern is shown with bold lines in Fig. 5.

Theorem 1: GTDBTPM is deadlock-free.

Proof: Because a cyclic dependency among resources is a necessary condition for deadlock, the GTDBTPM routing algorithm may be proven deadlock-free by showing that there cannot existed such a dependency among the channels. The performance of the GTDBTPM algorithm is dependent on the distribution of destination nodes in z- and then in x- y- coordinates. The GTDBTPM algorithm uses dual-path routing algorithm to distribute destination nodes in x- and y- coordinates. Lin *et al.* (1994) dual-path algorithm is deadlock-free. In z coordinate the GTDBTPM algorithm divides the network into two disjoint subnetworks N_{+z} and N_{-z}. By the definition of channel-partitioning scheme in z direction, $N_{+z} \cap N_{-z} = \phi$. Now, we will prove that there are no dependencies within each subnetwork. In N_{+z} a message entering a node represented by (x, y, z) always leaves on a node represented by (x, y, z + 1), (always leaves on a node with z- coordinate greater than z- coordinate of entered node); therefore, no cyclic dependency can exist among the channels. In N_{-z} a message entering a node represented by (x, y, z) always leaves on a node represented by (x, y, z - 1), (always leaves on a node with z- coordinate smaller than z- coordinate of entered node); therefore, no cyclic dependency can exist among the channels. Hence GTDBTPM algorithm is deadlock-free.

General three-dimension multi-phase multicast (GTDMPM): Our second proposed algorithm based on splitting the network into six subnetworks, N_{xR}, N_{xL}, $N_{xs,yu}$, $N_{xs,yL}$, $N_{xs,ys,zu}$ and $N_{xs,ys,zL}$. Subnetwork N_{xR} contains all right horizontal channels of source with addresses [(x, y, z), (x + 1, y, z]; Subnetwork N_{xL} contains all Left horizontal channels of source with addresses [(x, y, z), (x - 1, y, z)]; Subnetwork $N_{xs,yu}$ contains all upper vertical channels of source with addresses [(x, y, z), (x, y + 1, z]; Subnetworks $N_{xs,yL}$ contains all Lower vertical channels of source with addresses [(x, y, z), (x, y - 1, z]; Subnetworks $N_{xs,ys,zu}$ contains all upper diagonal channels of source with addresses [(x, y, z), (x, y, z + 1] and Subnetworks $N_{xs,ys,zL}$ contains all lower diagonal channels of source with addresses [(x, y, z), (x, y, z - 1]. Figure 6 shows the

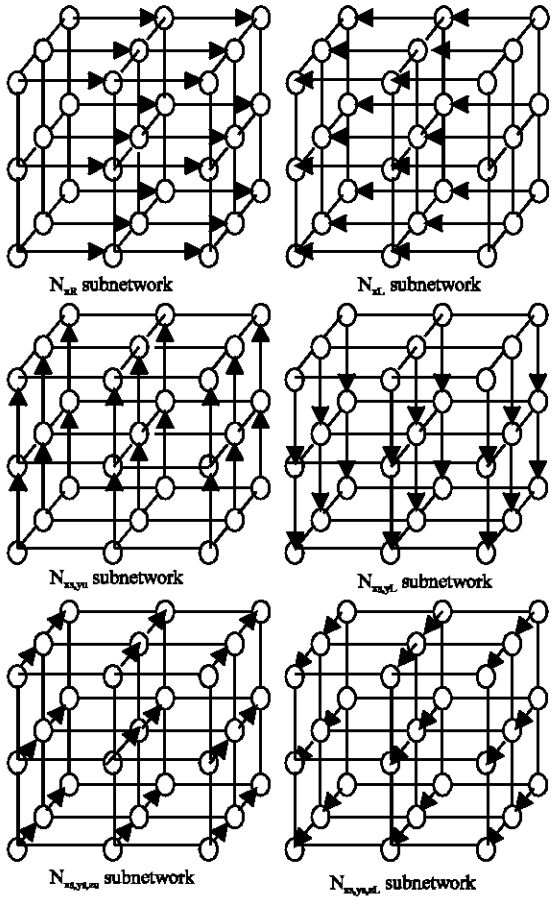


Fig. 6: Network partitioning for 3×3×3 mesh

partitioning of 3×3×3 mesh into six subnetworks. Suppose that the coordinate of the source node u_0 is represented by (x_0, y_0, z_0) and D represents the destination-set, the simple idea of this algorithm is as follow: -

Step 1: The source split the D into three subsets, D_{xs}, D_{xR} and D_{xL}, where D_{xs} contains the destination nodes which their x coordinates are same of x_0, D_{xR} contains the destination nodes to the right of u_0 and D_{xL} contains the destination nodes to the left of u_0 Formally, the subsets are described by the following expression:

$$D_{xs} = \{(x,y,z) \mid (x,y,z) \in D, x = x_0, (0 \le y < m), (0 \le z < r)\},$$

$$D_{xR} = \{(x,y,z) \mid (x,y,z) \in D, x > x_0, (0 \le y < m), (0 \le z < r)\},$$

$$D_{xL} = \{(x,y,z) \mid (x,y,z) \in D, x < x_0, (0 \le y < m), (0 \le z < r)\}$$

Step 2: The source split D_{xs} into four subsets $D_{xs,ys,zu}$ $D_{xs,ys,zL}$, $D_{xs,yu}$ and $D_{xs,yL}$, where $D_{xs,ys,zu}$ contains the destination nodes to the upper diagonal of u_0, $D_{xs,ys,zL}$ contains the destination nodes to the lower diagonal of u_0, $D_{xs,yu}$ contains the destination nodes to the upper vertical of u_0 and $D_{xs,yL}$ contains the destination nodes to the lower vertical of u_0. Formally, the subsets are described by the following expression:

$$D_{xs,ys,zu} = \{(x,y,z) \mid (x,y,z) \in D, x = x_0, y = y_0, z > z_0\},$$

$$D_{xs,ys,zL} = \{(x,y,z) \mid (x,y,z) \in D, x = x_0, y = y_0, z < z_0\},$$

$$D_{xs,yu} = \{(x,y,z) \mid (x,y,z) \in D, x = x_0, y > y_0, (0 \le z < r)\},$$

$$D_{xs,yL} = \{(x,y,z) \mid (x,y,z) \in D, x = x_0, y < y_0, (0 \le z < r)\}$$

Step 3: Sort destination subset $D_{xs,ys,zu}$ using the z coordinate as the key in ascending order, sort subset $D_{xs,ys,zL}$ using the z coordinate as the key in descending order, sort subset $D_{xs,yu}$ using the y coordinate as the key in ascending order, sort subset $D_{xs,yL}$ using the y coordinate as the key in descending order.

Step 4: The source sends the destination subset D_{xR} through subnetwork N_{xR} to next right node of u_0, which is represented by integer coordinate (x_0+1, y_0, z_0), sends destination subset D_{xL} through subnetwork N_{xL} to next left node of u_0, which is represented by integer coordinate (x_0-1, y_0, z_0), sends destination subset $D_{xs,yu}$ through subnetwork $N_{xs,yu}$ to next upper vertical node of u_0, which is represented by integer coordinate (x_0, y_0+1, z_0), sends destination subset $D_{xs,yL}$ through subnetwork $N_{xs,yL}$ to next lower vertical node of u_0, which is represented by integer coordinate (x_0, y_0-1, z_0), sends destination subset $D_{xs,ys,zu}$ through subnetwork $N_{xs,ys,zu}$ to next upper diagonal node of u_0, which is represented by integer coordinate (x_0, y_0, z_0+1) and sends destination subset $D_{xs,ys,zL}$ through subnetwork $N_{xs,ys,zL}$ to next lower diagonal node of u_0, which is represented by integer coordinate (x_0, y_0, z_0-1). The source would simultaneously send these messages by six ports (GTDMPM algorithm is all ports). In this step all diagonal destination subsets of source will be reached.

Step 5: The intermediate node that represented by (x_0, y_0+1, z_0) will be act as a source on it's surface, when it receives the destination subset $D_{xs,yu}$, it will repeat the previous steps from 2 to 4 (let D_{xs} = $D_{xs,yu}$) except that in

step 4, there is only 3 subsets ($D_{xs,ys,zu}$, $D_{xs,ys,zL}$, $D_{xs,yu}$, $D_{xs,yL} = \varphi$, $D_{xR} = \varphi$ and $D_{xL} = \varphi$). The intermediate node that represented by (x_0, y_0-1, z_0) will be act as a source on it's surface, when it receives the destination subset $D_{xs,yL}$, it will repeat the previous steps from 2 to 4 (let $D_{xs} = D_{xs,yL}$) except that in step 4, there is only 3 subsets ($D_{xs,ys,zu}$, $D_{xs,ys,zL}$, $D_{xs,yL}$, $D_{xs,yu} = \varphi$, $D_{xR} = \varphi$ and $D_{xL} = \varphi$).

Step 6: The intermediate node that represented by (x_0+1, y_0, z_0) will be act as a source on it's surface, when it receives the destination subset D_{xR}, it will repeat the previous steps from 1 to 5 (let $D = D_{xR}$) except that in step 4 there is only 5 subsets ($D_{xs,ys,zu}$, $D_{xs,ys,zL}$, $D_{xs,yu}$, $D_{xs,yL}$, D_{xR} and $D_{xL} = \varphi$). The intermediate node that represented by (x_0-1, y_0, z_0) will be act as a source on it's surface, when it receives the destination subsets D_{xL}, it will repeat the previous steps from 1 to 5 (let $D = D_{xL}$) except that in step 4 there is only 5 subsets ($D_{xs,ys,zu}$, $D_{xs,ys,zL}$, $D_{xs,yu}$, $D_{xs,yL}$, D_{xL} and $D_{xR} = \varphi$). If the destination subsets are not empty, the algorithm continues according to the previous method.

The following example shown the operation of the algorithm. Consider the 3 x 3 x 3 mesh given in Fig. 7. At the source node (1, 1,0), the following destination set D = {(0,0,0), (1,0,1), (2,0,1), (2,0,2), (2,1,0), (0,1,2), (1,1,2), (1,2,0), (1,2,1), (2,2,1), (0,2,2)} is divided into the following six subsets: D_{xL} = {(0,0,0), (0,1,2), (0,2,2)}, D_{xR} = {(2,0,1), (2,0,2), (2,1,0), (2,2,1)}, $D_{xs,yu}$ = {(1,2,0), (1,2,1)}, $D_{xs,yL}$ = {(1,0,1)} $D_{xs,ys,zu}$ = {(1,1,2)} and $D_{xs,ys,zL} = \varphi$.

The message will be sent to the destination nodes in D_{xR} through subnetwork N_{xR}, to the nodes in D_{xL} through subnetwork N_{xL}, to the nodes in $D_{xs,yu}$ through subnetwork $N_{xs,yu}$, to the nodes in $D_{xs,yL}$ through subnetwork $N_{xs,yL}$ and to the nodes in $D_{xs,ys,zu}$ through subnetwork $N_{xs,ys,zu}$. The routing pattern is shown with bold lines in Fig. 7.

Theorem 2: The GTDMPM algorithm is deadlock-free.

Proof: At the source node, GTDMPM algorithm divides the network into six disjoint subnetworks. This is obvious since, $N_{xs,ys,zu} \cap N_{xs,ys,zL} \cap N_{xs,yu} \cap N_{xs,yL} \cap N_{xR} \cap N_{xL} = \varphi$. Then GTDMPM algorithm is deadlock-free at the six subnetworks. Now, we will prove that there are no dependencies within each subnetwork. First, the nodes are labeled such that node (0, 0, 0) has label 0. The label assignment function L for an m×n×r mesh can be expressed in terms of the x-, y- and z-coordinates of nodes as follows:

$$L(x,y,z)=\{m*n*z+n*y+x\}$$

Figure 8a shows such a labeling in a 3×3×3 mesh. Next all channels entering a node j are labeled j; Fig. 8b

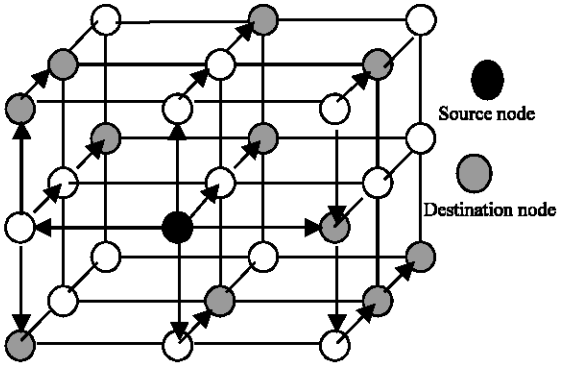
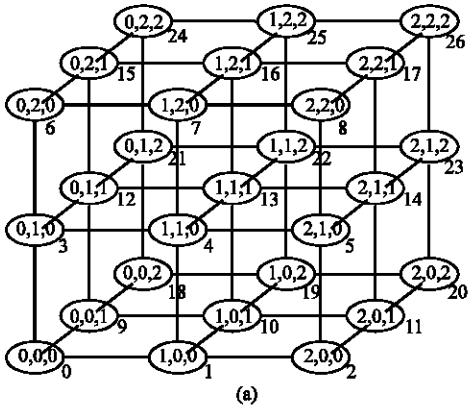
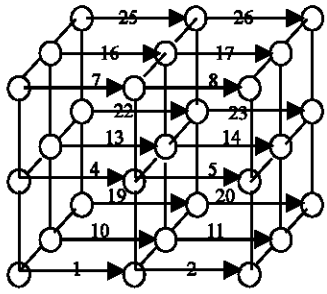


Fig. 7: The routing pattern of GTDMPM





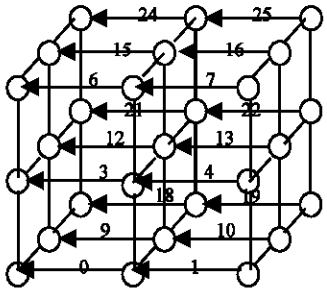


Fig. 8: The label assignment for 3×3×3 mesh, (a) nodes labeling and (b) channels labeling

shows such a channel labeling for the N_{xR} and N_{xL} subnetworks (the channels labeling for the other subnetworks are similar). In subnetwork N_{xR}, using the GTDMPM algorithm, a message entering a node on a channel labeled j always leaves on a channel labeled with a number greater than j; therefore, no cyclic dependency can exist among the channels. In subnetwork N_{xL}, using the GTDMPM algorithm, a message entering a node on a channel labeled j always leaves on a channel labeled with a number smaller than j; therefore, no cyclic dependency can exist among the channels. In subnetwork $N_{xs,yu}$, using the GTDMPM algorithm, a message entering a node on a channel labeled j always leaves on a channel labeled with a number greater than j, therefore, no cyclic dependency can exist among the channels. In subnetwork $N_{xs,yL}$ using the GTDMPM algorithm, a message entering a node on a channel labeled j always leaves on a channel labeled with a number smaller than j, therefore, no cyclic dependency can exist among the channels. In subnetwork $N_{xs,ys,zu}$, using the GTDMPM algorithm, a message entering a node on a channel labeled j always leaves on a channel labeled with a number greater than j, therefore, no cyclic dependency can exist among the channels. In subnetwork $N_{xs,ys,zL}$, using the GTDMPM algorithm, a message entering a node on a channel labeled j always leaves on a channel labeled with a number smaller than j, therefore, no cyclic dependency can exist among the channels. Hence GTDMPM algorithm is deadlock-free.

SIMULATION

To compare the performance of our proposed multicast routing algorithms, the simulation program used to model multicast communication in 3-D mesh networks is written in C++ and uses an event-driven simulation package, CSIM (Schwetman, 1985). CSIM allows multiple processes to execute in a quasiparallel fashion and provides a very convenient interface for writing modular simulation programs. The simulation program for multicast communication is part of a larger simulator, called MultiSim (McKinley and Trefftz, 1993), which is designed to study large-scale multiprocessors. MultiSim consists of several components, all of which run within the CSIM package. This section describes the program and results obtained from it. All simulations were executed until the confidence interval was smaller than 5% of the mean, using 95% confidence intervals, which are not shown in the results. To compare the performance of GTDBTPM and GTDMPM, 3-D mesh network that contained single channels is used. We have considered $\beta = 10$, where β denotes the ratio of the startup time over the propagation time of a flit from one router to a neighboring router. Also we have studied four-message lengths 1 flit, 100 flits, 1000 flits and 10000 flits. We have studied two sections of experiments, first section study the effects of average injection rate (interarrival time) and second section study the effects of average destination numbers.

First Section: We have first run simulations on a 5×5×5 mesh. The aim of this first set of experiments is to study the effects of average injection rate (average interarrival time) on our proposed algorithms. For our first set of simulations, we have fixed the number of destination nodes as 10% of the total number of nodes of the mesh, we have studied four message lengths 1 flit, 100 flits, 1000 flits and 10000 flits and we have studied start-up time ($\beta = 10$).

Figure 9 gives the plot of average network latency for various network loads. The average number of destinations for a multicast is 12, the message length is 1 flit and β is 10. Two algorithms exhibit good same performance at low load. Because message size is very small and start-up time is small (1 flit, $\beta = 10$), there is no contention in the network due to other multicasts, so two algorithms exhibit good same performance without effect the loads.

Figure 10 compares two algorithms again. The message length is 100 flits. The other parameters are the same as for the previous figure. The GTDMPM algorithm obtains the best performance. The GTDBTPM algorithm saturates when load is smaller than 50 and GTDMPM algorithm saturates when load is smaller than 20.

The GTDMPM algorithm, however, is less sensitive to increased load than the GTDBTPM algorithm. The disadvantage of GTDBTPM algorithm increases with the message lengths as shown in Fig. 11 and 12. Because the destinations are divided into p sets (number of columns,

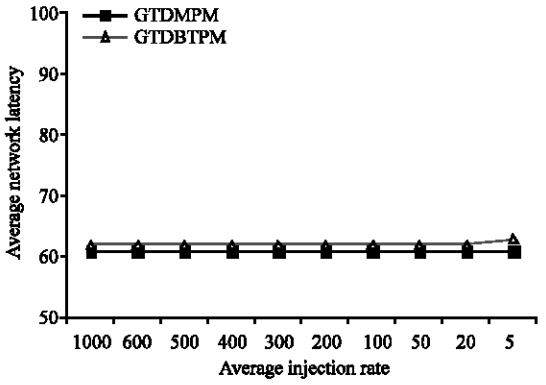


Fig. 9: Performance under different loads. $\beta = 10$, Message length = 1 flit and number of destinations = 12

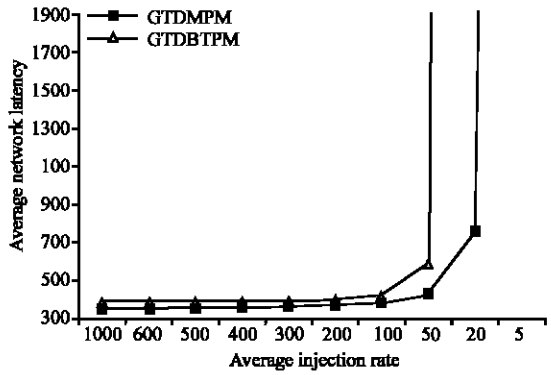


Fig. 10: Performance under different loads. β = 10, message length = 100 flits and number of destinations = 12

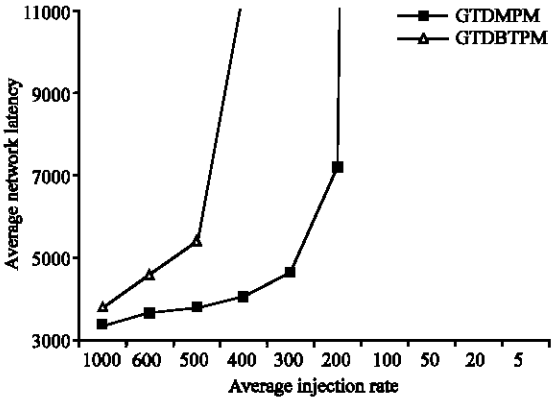


Fig. 11: Performance under different loads. β = 10, message length = 1000 flits and number of destinations = 12

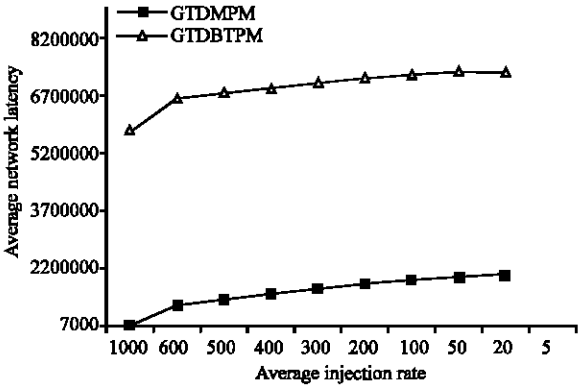


Fig. 12: Performance under different loads. β = 10, message length = 10000 flits and number of destinations = 12

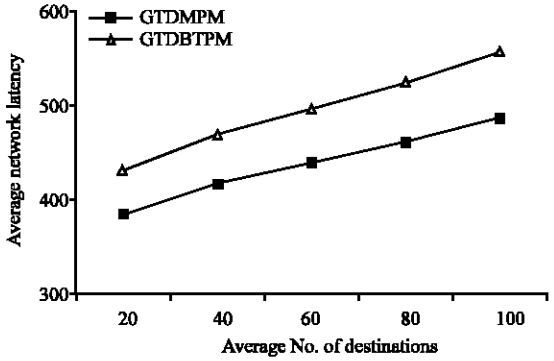


Fig. 13: Performance of different number of destinations. β = 10, message length = 100 flits and mean interarrival time = 300 μ sec

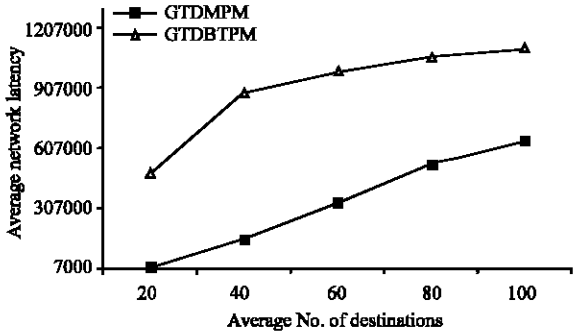


Fig. 14: Performance of different number of destinations. β = 10, message length = 1000 flits and mean interarrival time = 300 μ sec

rows and diagonals) in GTDMPM rather than two (use dual-path algorithm for each 2-D mesh) in GTDBTPM, they are reached more efficiently from the source, which is approximately centrally located among the sets. This allows decreasing the lengths of the paths used to reach the destinations. Paths of GTDMPM tend to be usually shorter than paths of GTDBTPM. Hence, the GTDMPM algorithm will not saturate as quickly.

Second section: We have run simulations on a 5×5×5 mesh. The aim of this second set of experiments is to study the effects of average destination nodes on our proposed algorithms. In this set of tests, every node generates multicast messages with an average time between messages of 300 μ sec. We have studied two message lengths 100 flits, 1000 flits and we have studied β = 10.

Figure 13 plots the network latency obtained by the two algorithms versus various values of number of destinations, ranging from 20 to 100. In this set of tests,

every node generates multicast messages with an average time between messages of 300 μ sec; the message length is 100 flits and small start-up time (β = 10). Notice that the GTDMPM algorithm results in lower latency than the GTDBTPM algorithm for large destination sets. This is likely due to the fact that GTDMPM algorithm introduces shortest path to the network. For a large enough number of destinations, however, the GTDMPM algorithm is less sensitive rather than GTDBTPM algorithm. However, the disadvantage of GTDBTPM algorithm increases with the message lengths.

Figure 14 compares two algorithms, again. The message length is 1000 flits. The other parameters are the same as for the previous figure. GTDMPM algorithm exhibits lower latency than GTDBTPM algorithm.

CONCLUSIONS

In this study, a new two deadlock-free multicast wormhole algorithms in 3-D mesh parallel machines using a path-based facility was presented. These algorithms are shown to be deadlock-free. The behavior of GTDMPM algorithm was compared with behavior of GTDBTPM algorithm using simulation.

All the experimental results show that the best performances are obtained by the GTDMPM algorithm over different traffic loads and destination set sizes. The disadvantage of GTDBTPM algorithm is that the paths from source to destinations are longer rather than the pats in GTDMPM algorithm.

REFERENCES

Agrawal, A., 1991. Limits on interconnection network performance. IEEE Trans. Parallel Dist. Syst., 2: 398-412.

Ahmed Al-Dubai, Mohamed Ould-Khaoua and Lewis Mackenzie, 2006. On balancing network traffic in path-based multicast communication. Fut. Gen. Comput. Syst., 22: 805-811.

Almasi, G.S. and A. Gottlieb, 1994. Highly Parallel Computing Benjamin/Cummings.

Anonymous, 1990. A Touchstone DELTA system description. Intel Corporation. Intel Supercomputing Systems Division.

Anonymous, 1995. CRAY T3E scalable parallel processing system. Cray Research Inc., http://www.cray.com/products/systems/crayt3e/.

Athas, W.C. and C.L. Seitz, 1988. Multicomputers: Message passing concurrent computers. IEEE Comput., 21: 9-24.

Dally, W.J. and C.L. Seitz, 1986. The tours routingchip. Dist. Comput., 1: 187-196.

Dally, W.J., 1989. The J-machine: System Support for Actors. Actors: Knowledge-Based Concurrent Computing. Hewitt and Agha (Eds.) MIT Press.

Dally, W.J., 1990. Performance Analysis of K-ary N-cube Interconnection Networks. IEEE Trans. Comput., 39: 775-785.

Foschia Rauber, R.T. and G. Runger, 1997. Modeling the Communication Behavior of the Intel Paragon. In: Modeling, Analysis and Simulation of Computer and Telecommunication Systems. IEEE Comput. Soc. Press, pp: 117-124.

Hwang, K., 1993. Advanced Computer Architecture: Parallelism, Scalability, Programmability. McGraw-Hill, New York.

Jianxi, F., 2002. Hamilton-connectivity and cycle-embedding of the Mobius cubes. Inform. Processing Lett., 82: 113-117.

Koeninger, R.K., M. Furtney and M. Walker, 1994. A shared memory MPP from Cray research. Digital Tech. J., 6: 8-21.

Kumar, D.R., W.A. Najjar and P.K. Srimani, 2001. A new adaptive hardware tree-based multicast routing in K-ary N-cubes. IEEE Trans. Comput., 50: 647-659.

Lessler, R.E. and J.L. Schwazmeier, 1993. CRAY T3D: A new dimension for Cray Research. In: COMPCON. IEEE Comput. Soc. Press, pp: 176-182.

Libeskind-Hadas, R., T. Hehre, A. Hutchings, M. Reyes and K. Watkins, 1997. Adaptive multicast routing in wormhole networks. In: Proceedings of the 9th IASTED International Conference on Parallel and Distributed Computing and Systems. Washington DC, 13-16, pp: 513-522.

Lin, X., P.K. McKinley and L.M. Ni, 1994. Deadlock-free multicast wormhole routing in 2-D mesh multicomputers. IEEE Trans. Parallel Distrib. Syst., 5: 793-804.

McKinley, P.K. and C. Trefftz, 1993, MultiSim: A tool for the study of large-scale multiprocessors. In: Proceeding 1993 International Workshop on Modeling, Analysis and Simulation of Computer and Telecommun. Nehvorks (MASCOTS 93), pp: 57-62.

Moharam, H., M.A. Abd El-Bakay and S.M.M. Nasssar, 2000. Yomana-an efficient deadlock-free multicast wormhole algorithm in 2-D mesh multicomputers. J. Syst. Architect., 46: 1073-1091.

Nuth, P.R. and W.J. Dally, 1992. The J-Machine Network. In: Proc. IEEE Int. Conf. on Computer Design: VLSI in Computer and Processors. IEEE Computer Society Press, pp: 420-423.

Schwetman, H.D., 1985, CSIM: A C-based, process-oriented simulation language. Technical Report, Microelectronics and Computer Technology Corp, pp: 80-85.