



Journal of Applied Sciences

ISSN 1812-5654

science
alert

ANSI*net*
an open access publisher
<http://ansinet.com>

Hamiltonian Paths for Designing Deadlock-Free Multicasting Wormhole-Routing Algorithms in 3-D Meshes

El-Obaid Amnah and Wan-Li Zuo

College of Computer Science and Technology, Jilin University, Changchun, China

Abstract: Multicasting is an information dissemination problem which consists, for a processor of a distributed memory parallel computer, to send a same message to a subset of processors. In this study, we propose two new multicast algorithms for a mesh network using wormhole routing with the path-based facility. The main feature of the proposed algorithms is its ability to handle multicast operations with a fixed number of message-passing steps irrespective of the network size. Results from extensive comparative analysis reveal that our algorithms exhibit superior performance characteristics over the well-known GTDMPM algorithm.

Key words: Multicasting communication, wormhole routing, Hamiltonian model, 3-D mesh, deadlock-free

INTRODUCTION

Optimizing the performance of message-passing multicomputers requires matching inter-processor communication algorithms and application characteristics to a suitable underlying interconnection network. The mesh has been one of the most common networks for existing multicomputers due to its desirable properties, such as scalability, ease of implementation, recursive structure and ability to exploit communication locality found in many parallel applications to reduce message latency. 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).

The switching method determines the way messages visit intermediate nodes. Wormhole switching has been widely used in practice due firstly to its low buffering requirements, allowing for efficient router implementation. Secondly and more importantly, it makes latency almost independent of the message distance in the absence of blocking (Al-Dubai *et al.*, 2006). 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, 1987). 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 (Moharam *et al.*, 2000).

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. In general, the multicasting problem can be modeled by three routing schemes: Tree-based, unicast-based and path-based routing (Nen-Chung *et al.*, 2002). The tree-based multicasting relies on finding a tree from the underlied network architecture and the source messages are sent to each destination along the paths on the constructed tree. The unicast-based multicasting sends the messages from source node to destination nodes via intermediate nodes recursively. In path-based facility (Lin *et al.*, 1994) header consists of a list of destination addresses that must be reached in the specified order. More precisely, a header consists of an ordered sequence of addresses $@(v_1) @(v_2) \dots @(v_k)$ meaning that the message must go first to v_1 , next to v_2 and so on. When the flits of a message reach an intermediate destination v_i , the address $@(v_i)$ is removed from the header and they can be copied to the local memory while they continues in order to reach the next destination specified by the header, namely $@(v_{i+1})$. A message is removed from the network when it reaches its last destination. In this way, a message can be delivered to

several destinations with the same startup latency as a message sent to a single destination (Fleury and Fraigniaud, 1998).

The performance of multicast communication is measured in terms of its latency in delivering a message to all destinations. Multicast latency consists of three parts, start-up latency network latency and blocking latency (McKinley *et al.*, 1995; Duato *et al.*, 2003; McKinley *et al.*, 1994; Panda *et al.*, 1999; Malumbres and Duato, 2000; Tseng *et al.*, 1996; Chin *et al.*, 2000). 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 delays, i.e., the elapsed time after the head of a message has entered the network at the source until the tail of the message emerges from the network at the destination, while blocking latency accounts for delays due to message contention over network resources, e.g., buffers and channels.

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, 1987; Moharam *et al.*, 2000; Nen-Chung *et al.*, 2002; Lin *et al.*, 1994; Fleury and Fraigniaud, 1998; Malumbres and Duato, 2000; Chin *et al.*, 2000; Kumar *et al.*, 2001; Fan, 2002; Darwish *et al.*, 2005; Al-Dubai and Ould-Khaoua, 2003; Amnah and Wan, 2007).

THE SYSTEM MODEL

This study, discussion is restricted to the 3-D mesh topology with Bi-directional channels. Figure 1a shows (4×4×4) 3-D mesh. The vertices represent the computing nodes and the arcs represent the communication links. The basic node architecture is shown in Fig. 1b. An m (rows) × n (columns) × r (layers) 3-D mesh comprises mnr nodes interconnected in a grid fashion. The 3-D mesh topology can be modeled as a 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. The mesh graph is formally defined below as follow:

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

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

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

The mesh topology is asymmetric due to the absence of the wrap-around connections along each dimension. As a result, nodes may not be connected to the same number of neighbors; those at the corners, edges and middle of the network have four and six neighbors, respectively. In this system, the node consists of a Processing Element (PE) and router. The processing element contains a processor and some local memory. There are local channels used by the processing element to inject/eject messages to/from the network, respectively. Messages generated by the processing element are injected into the network through the injection channel.

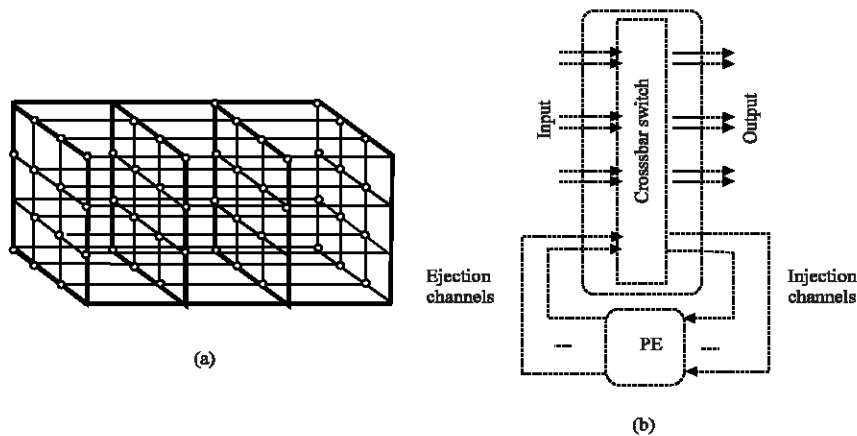


Fig. 1: (a) The 3D mesh and (b) the node structure in 3D meshes

This study considers the All-Port router model where routers are able to relay multiple messages simultaneously provided that each incoming message requires a unique outgoing channel leading to a neighboring node and that a node can simultaneously send and receive messages along all ejection and injection channels.

PRELIMINARIES

Hamiltonian schema: 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. A Hamiltonian schema can be modeled as a graph $G = (V, E)$ and $|V| = N$, we suppose that $\delta = (v_0, v_1, \dots, v_{N-1}, v_N)$ is a Hamiltonian path in the graph G . According to the order of the vertex in the Hamiltonian δ , we can assign each vertex in the graph a label. The label of the vertex $v_i \in V$ is denoted as $l(v_i)$, where $l(v_i)$ is a natural number. That is, the Hamiltonian paths starts at the node labeled 0 and go

following the nodes with labels 1, 2, ..., to the node with label $N-1$ consecutively.

The network partitioning strategy is fundamental to our multicast routing algorithms. After assigning each node a label in the network, we can divide the network channel network. The high-channel network contains all of the directional common channels with the nodes labeled from low to high numbers. The low-channel network contains all of the directional common channels with the nodes labeled from high to low numbers. After partitioning the network into two subnetworks, it is easy to see that every physical communication link lies in one and only one subnetwork, a high-channel network or low-channel network. Each of the two subnetworks has an independent set of physical links in the network. Figure 2a shows such a labeling in a $3 \times 3 \times 3$ mesh, in which each node is represented by its integer coordinate (x, y, z) . 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 as shown in Fig. 2b and

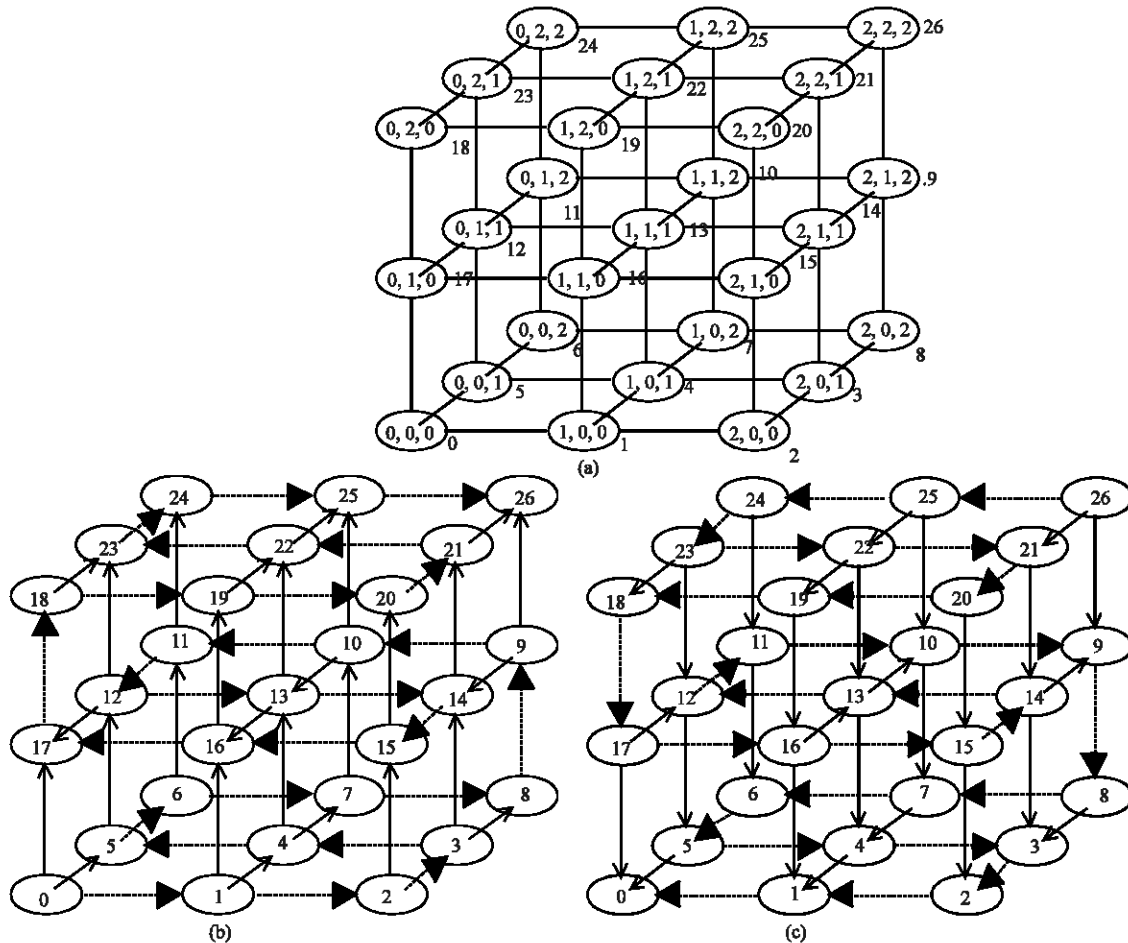


Fig. 2: The labeling of a $3 \times 3 \times 3$ mesh; (a) physical network, (b) high-channel network and (c) low-channel network

the low-channel network contains all of the channels whose direction is from higher-labeled nodes to lower-labeled nodes (Fig. 2c).

Mapping hamiltonian model to 3-D mesh networks: A 3-D mesh network contains many Hamiltonian paths. In the following, we give the node labeling function $l(\mu)$ for a 3-D mesh. The label assignment function l for an $m \times n \times r$ mesh can be expressed in terms of the x -, y - and z -coordinates of nodes as follows:

If y is even

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

If y is odd

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

The node labeling function will assign each node a unique number. For a 3-D $m \times n \times r$ the Hamiltonian model starts at the node numbered 0, following the nodes with labels 1, 2, ..., to the node with label $mnr-1$ consecutively.

Routing function: When implementing a multicast routing algorithm, the routing function must be considered. Once a node sends a message to a set of destination nodes, it uses the multicast message preparation algorithm to prepare the destination list in the message header, which will be later sent to the router. The router determines the path for the message according to the routing function. In the following we describe the routing function for the network with Hamiltonian paths model.

The routing function R for the Hamiltonian model can be viewed as $R: V \times V \rightarrow V$. It is defined as a function of the node currently holding a message and the destination node of this message. It returns the neighboring node to which the message must be forwarded. More precisely, if u is the current node and v is the destination node, then $R(u, v) = w$, such that w is a neighboring node of u and if $l(u) < l(v)$, then we have the following equation:

$$l(w) = \max \left\{ L(z) \mid \begin{array}{l} L(z) \leq L(v) \text{ and } z \text{ is a} \\ \text{neighboring node of } u \end{array} \right\}$$

or, if $L(u) > L(v)$, then we have the following equation: (3)

$$l(w) = \min \left\{ L(z) \mid \begin{array}{l} L(z) \geq L(v) \text{ and } z \text{ is a} \\ \text{neighboring node of } u \end{array} \right\}$$

As proved in (Lin *et al.*, 1994) this routing function is deadlock-free even using the path based facility.

THE PROPOSED ALGORITHMS (GTDTPM AND GTDSPM)

Here we, introduces the General Three-Dimension Two-Phase Multicast (GTDTPM for short) and General Tree-Dimension Six-Phase Multicast (GTDSPM for short) algorithms for All-Port 3-D mesh based on the Hamiltonian model. The proposed algorithms exploit the features of Hamiltonian paths to implement multicast in two and six message-passing steps, thus considerably reducing the effects of both network size and start-up latency. GTDTPM is based on splitting the destination set into two disjoint subsets (D_U and D_L), while GTDSPM is based on splitting the destination set into six disjoint subsets (D_{U1} , D_{U2} , D_{U3} , D_{L1} , D_{L2} and D_{L3}) and multicasting the message to these different sets in a pipeline fashion.

General Three-Dimension Two-Phase Multicast (GTDTPM): The GTDTPM algorithm concept for the multicast routing involves restricting the maximal path length that two messages are routed. At the source node, GTDTPM 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 simple idea of this algorithm is as follow:

Step 1: In GTDTPM 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 N_U and to the destination nodes in D_L using the low-channel network N_L .

Step 2: 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.

Step 3: 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 .

Step 4: The GTDTPM 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.

Step 5: If the sets of the destination nodes are not empty, the algorithm continues according to the previous method.

Theorem 1: GTDTPM is deadlock-free.

Proof: At the source node, GTDTPM algorithm divides the network into two disjoint subnetworks. This is obvious since, $N_U \cap N_L = \phi$. Then GTDTPM algorithm is deadlock-free at the two subnetworks. Now, we will prove that there are no dependencies within each subnetwork. Since each copy of the message is routed entirely within a single subnetwork and monotonic order (ascending order in N_U as shown in Fig. 3a and descending order in N_L as shown in Fig. 3b) of requested channels is guaranteed, there cannot exist a cycle within any subnetwork; hence, no cyclic dependency can be created among the channels. So GTDTPM is deadlock-free.

General Three-Dimension Six-Phase Multicast (GTDSMP): In a 3-D mesh, most nodes have outgoing degree 6 so up to six paths can be used to deliver a message, depending on the locations of the destinations relative to the source node. The only difference between GTDSMP algorithm and GTDTPM algorithm concerns message preparation at the source node, in which the destination sets D_U and D_L of the GTDTPM algorithm are further partitioned. The set D_U is divided into three

subsets, D_{U1} containing the nodes whose x coordinates are greater than to that of source, D_{U2} containing the nodes whose x coordinates are smaller than to that of source and the D_{U3} containing the remaining nodes in D_U . The set D_L is partitioned in a similar manner. The message is sent to the six sets simultaneously through the six output ports of source. 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 message preparation of the GTDSMP algorithm is as follow:-

Step 1: Divide D into two sets D_U and D_L such that D_U contains all the destination nodes with higher l value than $l(\mu_0)$ and D_L the nodes with lower l value than $l(\mu_0)$.

Step 2: 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.

Step 3: Divide D_U into three sets, D_{U1} , D_{U2} and D_{U3} as follows:

$$D_{U1} = \{ (x, y, z) \mid (x, y, z) \in D_U, x > x_0, (0 \leq y < m), (0 \leq z < r) \},$$

$$D_{U2} = \{ (x, y, z) \mid (x, y, z) \in D_U, x < x_0, (0 \leq y < m), (0 \leq z < r) \},$$

$$D_{U3} = \{ (x, y, z) \mid (x, y, z) \in D_U, x = x_0, (0 \leq y < m), (0 \leq z < r) \}$$

Construct three messages, one containing D_{U1} as part of the header and second containing D_{U2} as part of the header and the other containing D_{U3} , as part of the header. The source sends three messages to neighboring nodes through N_U using routing function R.

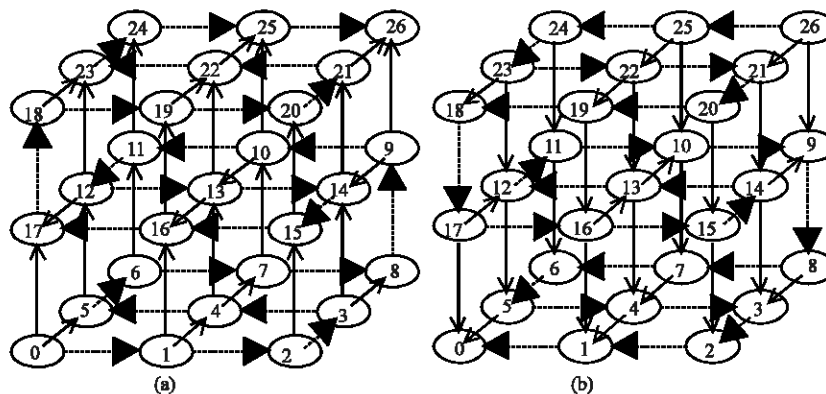


Fig. 3: The monotonic order of a 3x3x3 mesh (a) ascending order in N_U network and (b) descending order in N_L network

Step 5: Similarly, partition D_L into D_{L1} , D_{L2} and D_{L3} and construct three messages.

Theorem 2: The GTDSPM is deadlock-free.

The same argument used in the proof of theorem 1 can be used to prove this theorem.

COMPARATIVE STUDY

As an example, to demonstrate the difference between GTDTPM and the GTDSPM algorithms, consider the example shown in Fig. 4 for a $4 \times 4 \times 4$ mesh topology labeling using a Hamiltonian path. The source node labeled 25 with integer coordinate (1, 1, 1) initiates a multicast to the destination set $D = \{0, 15, 31, 23, 40, 56, 9, 17, 38, 54, 5, 21, 42, 61, 50, 3, 11, 28, 19, 35, 59\}$.

The GTDTPM algorithm splits and sorts, D into two subsets $D_U = \{28, 31, 35, 38, 40, 42, 50, 54, 56, 59, 61\}$ and $D_L = \{23, 21, 19, 17, 15, 11, 9, 5, 3, 0\}$, the routing pattern is shown with bold lines in Fig. 5(a). The GTDSPM algorithm splits destination set first into two subsets, $D_U = \{(3,1,0), (0,1,0), (3,2,0), (1,2,1), (0,2,2), (2,2,2), (2,3,3), (1,3,2), (0,3,1), (3,3,1), (2,3,0)\}$ with Hamiltonian labels $\{28, 31, 35, 38, 40, 42, 50, 54, 56, 59, 61\}$, respectively and $D_L = \{(0,1,2), (2,1,2), (3,1,3), (1,1,3), (0,0,3), (3,0,2), (1,0,2), (2,0,1), (3,0,0), (0,0,0)\}$ with Hamiltonian labels $\{23, 21, 19, 17, 15, 11, 9, 5, 3, 0\}$, respectively. D_U is further divided into three subsets D_{U1} , D_{U2} and D_{U3} , with $D_{U1} = \{(3,1,0), (3,2,0), (2,2,2), (2,3,3), (3,3,1), (2,3,0)\}$ with Hamiltonian labels $\{28, 35, 42, 50, 59, 61\}$, respectively, $D_{U2} = \{(0,1,0), (0,2,2), (0,3,1)\}$ with Hamiltonian labels $\{31, 40, 56\}$, respectively and $D_{U3} = \{(1,2,1), (1,3,2)\}$ with Hamiltonian labels $\{38, 54\}$,

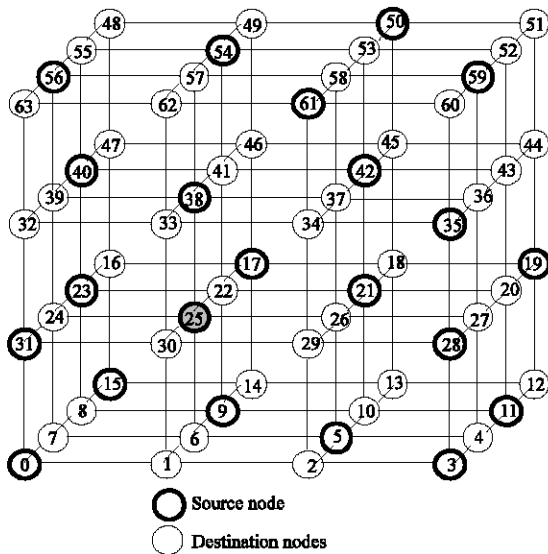


Fig. 4: An example of $4 \times 4 \times 4$ mesh

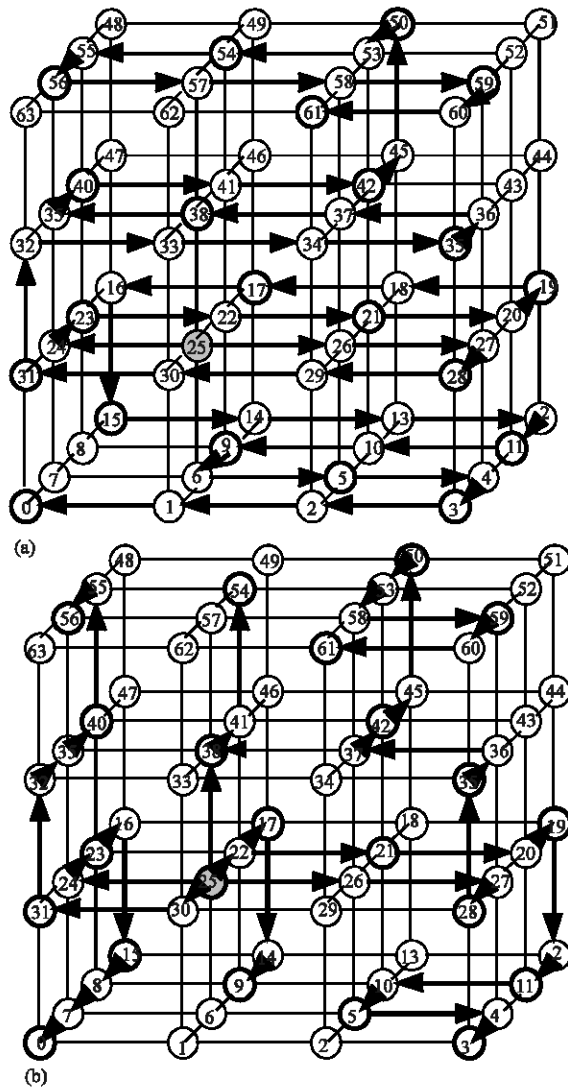


Fig. 5: The routing patterns of (a) GTDTPM algorithm and (b) GTDSPM algorithm

respectively. D_L is also $D_{L1} = \{(2,1,2), (3,1,3), (3,0,2), (2,0,1), (3,0,0)\}$ with Hamiltonian labels $\{21, 19, 11, 5, 3\}$, respectively, $D_{L2} = \{(0,1,2), (0,0,3), (0,0,0)\}$ with Hamiltonian labels $\{23, 15, 0\}$, respectively and $D_{L3} = \{(1,1,3), (1,0,2)\}$ with Hamiltonian labels $\{17, 9\}$, respectively, the routing pattern is shown with bold lines in Fig. 5b.

Using GTDTPM algorithm, Fig. 5a, number of channels used to deliver the message is 51 (28 in the high-channel network and 23 in the low-channel network). The maximum distance from the source to a destination is 28 hops. Using GTDSPM algorithm, Fig. 5b, number of channels used to deliver the message is 45 channels (24 in the high-channel network and 21 in the low-channel

network). The maximum distance from the source to destination is 24 hops. Hence, this example shows that GTDSPM algorithm can offer significant advantage over GTDTPM algorithm in terms of generated traffic and the maximum distance between the source and destination nodes.

SIMULATIONS

To evaluate the performance of the multicast schemes in an interconnection network, there are some parameters that must be considered: The injection rate, the multicast size, the message length and the startup latency. The injection rate is the average interarrival time, the multicast size is the number of destination nodes and the message length f is the number of flits in a message. The message startup latency β includes the software overhead for buffers allocating, messages coping, router initializing, etc.

We first give our assumptions to the parameters of system architecture in the simulations. All simulations were performed for a $5 \times 5 \times 5$ 3-D mesh. We examined the routing performance of our proposed schemes under various injection rate, multicast sizes and message lengths. The source node and the destination nodes for each multicasting were randomly generated. The small message startup latency β is 10 m sec. For all of the multicasting, the message sizes of 1, 100 and 1000 flits were simulated.

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 VC++ 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 figures. To compare the performance of GTDTPM and GTDSPM to the well known GTDMPM algorithm (The destinations in a multicast message are placed into submulticast messages according to the column, rows and diagonals) (Amnah and Wan, 2007), 3-D mesh network that contained single channels is used.

Effects of the injection rate and message length: The aim of this first set of experiments is to study the effects of average injection rate 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 three message lengths 1 flit, 100 flits and 1000 flits and startup latency $\beta = 10$.

Figure 6-8 show the performance of average network latency for various network loads on a $5 \times 5 \times 5$ network with small message latency. Results are shown for message lengths of 1, 100 and 1000 flits, respectively. It is observed that, the performance of GTDTPM and GTDSPM algorithms is superior to that of the GTDMPM algorithm. The higher latencies for the GTDMPM algorithm are mainly due to source congestion.

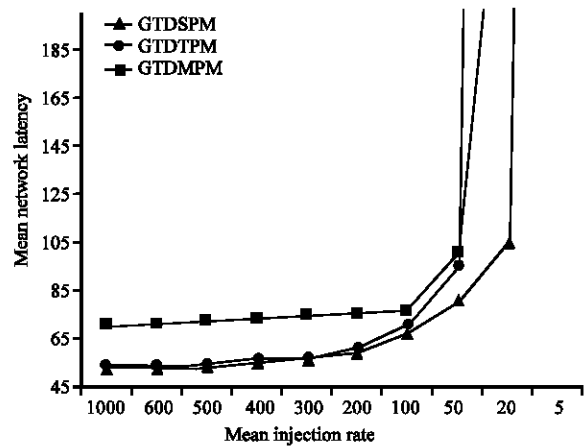


Fig. 6: Performance under different loads. $\beta = 10$, Message length = 1 flit and No. of destinations = 12

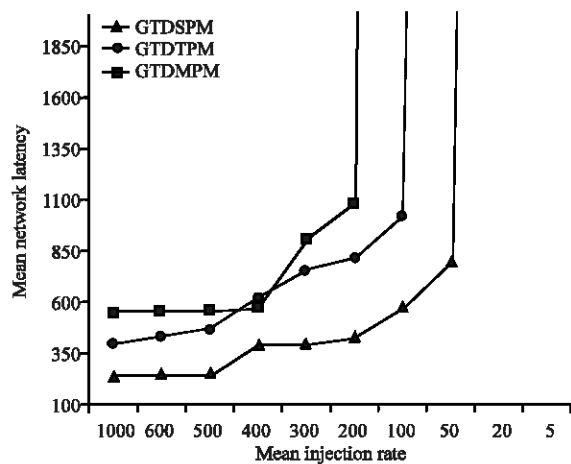


Fig. 7: Performance under different loads. $\beta = 10$, message length = 100 Flits and No. of destinations = 12

In Fig. 6, with small message startup latency the performance of our proposed GTDTPM and GTDSPM algorithms is superior to that of the GTDMPM algorithm. This difference shows the different strategies to improve the performance of the multicast communication. This implies that the message preparation is the critical part of the multicast routing algorithm. Improvement of the message preparation is more effective to the performance of the multicast communication. GTDTPM and GTDSPM exhibit good performance at low load. Because message length and message startup latency is very small ($f = 1$ 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. GTDSPM algorithm exhibits slight improvement than GTDTPM algorithm.

Figure 7, 8 compare three algorithms, again. The message length is 100 and 1000 flits, respectively. A gain both GTDTPM and GTDSPM algorithms obtain better performance over GTDMPM algorithm. This is because the new algorithms implement multicasting with a high degree of parallelism. The performance of the GTDSPM algorithm is better than GTDTPM algorithm, because, as shown earlier, paths tend to be shorter, generating less traffic. Hence, the network will not saturate as quickly. For larger injection rate, the network becomes saturated. The GTDSPM algorithm saturates lately in any case and GTDMPM algorithm saturates first in any case. The dependencies among message preparation of the GTDMPM become more critical to performance and cause the delay to increase rapidly. The disadvantage of GTDTPM algorithm increases with the message lengths as shown in Fig. 8. Because the destinations are divided into six sets in GTDSPM rather than two in GTDTPM, they are

reached more efficiently from the source, which is approximately centrally located among the sets, which allows decreasing the lengths of the paths used to reach the destinations.

All the experimental results show that the best performances are obtained by the GTDSPM algorithm. However, the disadvantage of GTDSPM algorithm is not appeared until both the load and number of destinations are relatively high.

Effects of different multicast sizes and message length:

The aim of this second set of study is to study the effects of multicast size 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 startup latency $\beta = 10$.

Figure 9 and 10 show the performance of the various multicast schemes on a $5 \times 5 \times 5$ network with small message latency. The results reveal that as the multicast size increases, the performance of the GTDMPM degrades significantly. The GTDMPM do not match the good scalability of the mesh. In contrast, the GTDTPM achieves the highest parallelism during multicast operation in all multicast sizes. Furthermore, it manages to maintain a good level of performance irrespective of the multicast size. This is because in the GTDTPM, when the multicast size increases, there is no increase in the number of message-passing steps required to complete the multicast operation. However, the GTDMPM implement the multicast operation in a highly sequential manner, i.e., it require more message-passing steps to implement multicast operations as the multicast size increases.

Figure 9 plot the network latency obtained by the three algorithms versus various values of number of

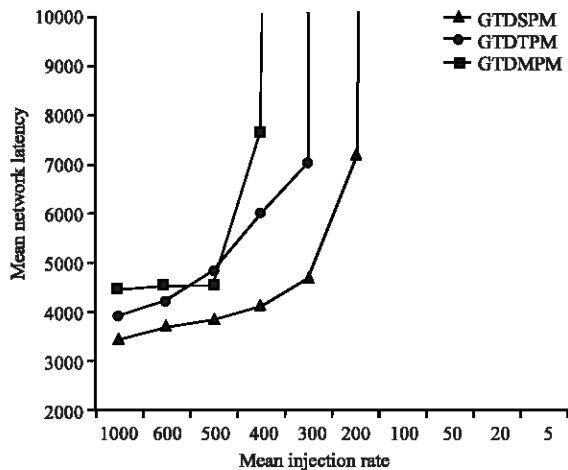


Fig. 8: Performance under different loads. $\beta = 10$, message length = 1000 flits and No. of destinations = 12

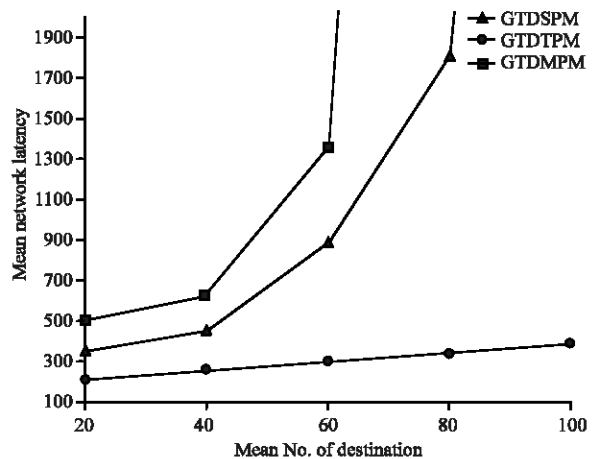


Fig. 9: Performance of different number of destinations. $B = 10$, message length = 100 flits and mean interarrival time = 300 μ sec

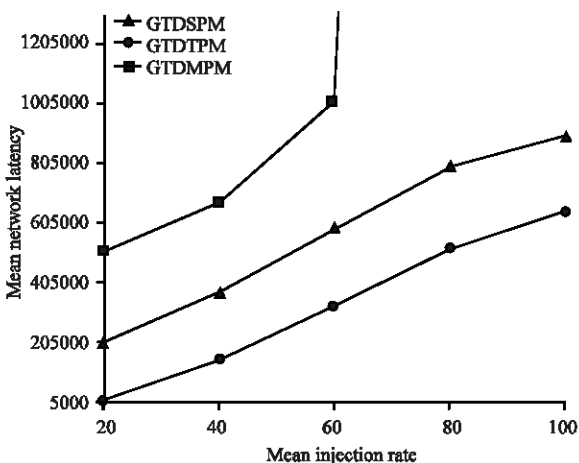


Fig. 10: Performance of different number of destinations. $\beta = 10$, Message length = 1000 flits and mean interarrival time = 300

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 startup latency $\beta = 10$. Notice that, the performance of GTDMPM algorithm worse than that of GTDTPM and GTDSPM algorithms. This is because the GTDMPM algorithm is a multiple-phase multicasting that needs more startup latency for processing. GTDMPM algorithm saturates faster when the number of destinations is greater than 60. The GTDTPM algorithm results in lower latency than the GTDSPM algorithm for large destination sets. However, in the GTDSPM and GTDMPM algorithms, the source node will send on all of its outgoing channels to reach a large set of destinations. Until this multicast transmission is complete, any flit from another multicast message that routes through that source node will be blocked at that point. The source node becomes a hot point. In fact, every node currently sending a multicast message is likely to be a hot point. If the load is very high, these hot points may decrease system throughput and increase message latency. In GTDTPM algorithm the source node will send on only two of its outgoing channels, hot points are less likely to occur, the behavior of GTDTPM algorithm is stable under high loads with large destination sets. However, the disadvantage of GTDSPM algorithm increases with the message lengths under large destination sets as shown in Fig. 10.

CONCLUSIONS

In this study, a new two 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 proposed algorithms GTDTPM and GTDSPM have the main advantage of exhibiting a high degree of parallelism and requiring only two and six startups latency message-passing steps, respectively irrespective of the destination nodes involved. Furthermore, a performance analysis has revealed that the proposed algorithms have superior latency characteristics over the well known the GTDMPM algorithm.

All the experimental results on the average network latency as a function of injection rate show that the best performances are obtained by the GTDSPM algorithm; this fact is somewhat independent of the startup time. In fact, experimental results on the average network latency as a function of the number of destinations, the hot point is the real cause of the performance degradation for the GTDSPM algorithm. The GTDTPM algorithm outperforms the GTDSPM algorithm for large destination sets independent of the startup time.

REFERENCES

- Al-Dubai, A.Y. and M. Ould-Khaoua, 2003. A new scalable broadcast algorithm for multiport meshes with minimum communication steps. *Microprocess. Microsyst.*, 27: 101-113.
- Al-Dubai, A.Y., M. Ould-Khaoua and L. Mackenzie, 2006. On balancing network traffic in path-based multicast communication. *Future Gene. Comput. Syst.*, 22: 805-811.
- Almasi, G.S. and A. Gottlieb, 1994. *Highly Parallel Computing Benjamin/Cummings.*
- Amnah El-Obaid and Wan Li-Zuo, 2007. Deadlock-free multicast wormhole algorithm in 3-D mesh multicomputers. *Inf. Technol. J.*, 6: 623-632.
- Anonymous, 1990. A touchstone DELTA system description. Intel Corporation. Intel Supercomputing Systems Division.
- Anonymous, 1995. CRAY T3E Scalable Parallel Processing System, Cray Research Inc., htGTDTPM: <http://www.cray.com/products/systems/crayt3e/>.
- Athas, W.C. and C.L. Seitz, 1988. Multicomputers: Message passing concurrent computers. *IEEE Comp.*, 21: 9-24.
- Chin, T.S., C.Y. Chang and J.P. Sheu, 2000. Efficient path-based multicast in wormhole-routed mesh networks. *J. Syst. Arch.*, 46: 919-930.
- Dally, W.J. and C.L. Seitz, 1987. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Trans. Co.*, C-36 (5).

- Darwish, M.G., A.A. Radwan, M. Abd El-Baky and K. Hamed, 2005. Gtgdtpmm-an efficient deadlock-free multicast wormhole algorithm for communication in 2d torus multicomputers. *IJICIS*, Vol. 5, No. 1.
- Duato, J., Y. Alamanchili and C.L. Ni, 2003. *Interconnection Networks: An Engineering Approach*, Elsevier Science.
- Fan, J., 2002. Hamilton-connectivity and cycle-embedding of the Mobius cubes. *Inform. Process. Lett.*, 2: 113-117, 30.
- Fleury, E. and P. Fraigniaud, 1998. Strategies for path-based multicasting in wormhole-routed meshes. *J. Parallel Distrib. Comput.*, 60: 26-62.
- Foschia, R., T. Rauber 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.
- Kumar, D.R., Walid A. Najjar and Pradip K. Srimani, 2001. A New adaptive hardware tree-based multicast routing in K-Ary N-Cubes. *IEEE Trans. Comput.*, 7: 647-659.
- Lessler, R.E. and J.L. Schwazmeier, 1993. CRAY T3D: A New Dimension for Cray Research. In: *COMPCON*. IEEE Comp. Soc. Press, pp: 176-182.
- 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.
- Malumbres, M.P. and J. Duato, 2000. An efficient implementation of tree-based multicast routing for distributed shared-memory multiprocessors. *J. Syst. Arch.*, 46: 1019-1032.
- McKinley, P.K. and C. Trefftz, 1993. MultiSim: A tool for the study of large-scale multiprocessors. In: *Proceeding of International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunication Networks (MASCOTS 93)*, pp: 57-62.
- McKinley, P.K., H. Gu, A. Esfahanian and L.M. Ni, 1994. Unicast-based multicast communication in wormhole-routed direct networks. *IEEE GTDTPMDS*, 5: 1254-1265.
- McKinley, P., Y. Tsai and J.D. Robinson, 1995. Collective communication in wormhole-routed massively parallel computers. *IEEE Comput.*, 28: 39-50.
- Moharam, H., M.A. Abd El-Baky and S.M.M., 2000. Yomna-An efficient deadlock-free multicast wormhole algorithm in 2-D mesh multicomputers. *J. Syst. Arch.*, 46: 1073-1091.
- Nen-Chung Wang, Chih-Ping Chu and Tzung-Shi Chen, 2002. A dual hamiltonian-path-based multicasting strategy for wormhole routed star graph interconnection networks. *J. Parallel Distrib. Comput.*, 62: 1747-1762.
- Nuth, P.R. and W.J. Dally, 1992. The J-Machine Network. In: *Proceeding IEEE International Conference on Computer Design: VLSI in Computer and Processors*. IEEE Comput. Soc. Press, pp: 420-423.
- Panda, D.K., S.L. Singa and R. Kesavan, 1999. Multidestination message-passing in wormhole k-ary n-cube networks with base routing conformed paths. *IEEE GTDTPMDS*, 10: 76-96.
- Schwetman, H.D., 1985. CSIM: A C-based, process-oriented simulation language. Technical Report Microelectron. Comput. Technol. Corp, pp: 80-85.
- Tseng, Y., D.K. Panda and T. Lai, 1996. A trip-based multicasting model in wormhole-routed networks with virtual channels. *IEEE Trans. Parallel and Distrib. Syst.*, 7: 138-150.