Path-Based Fault-Tolerant Multicasting in Mesh-Connected Multicomputers*

Xiao Chen and Jie Wu

Department of Computer Science and Engineering

Florida Atlantic University

Boca Raton, FL 33431

{xchen, jie}@cse.fau.edu

Abstract

We present a deadlock-free path-based fault-tolerant multicast algorithm in 2-D meshes. The fault model considered is the faulty block model with inter-block distance of at least three. The path is Hamiltonian that does not need to be reconstructed when a faulty block is encountered. Instead, the path is updated locally in the neighborhood of faulty blocks. Two virtual channels are used to prevent the deadlock. The approach can be easily extended to 2-D meshes with inter-block distance of at least two and to 3-D meshes. This is the first attempt to localize the effect of a faulty block in a path-based fault-tolerant multicast algorithm.

Keywords: *deadlock, fault tolerance, mesh, multicast, path-based routing, virtual channel*

*This work was supported in part by NSF grant CCR 9900646.

1 Introduction

The distributed memory multiprocessor paradigm provides a promising means of constructing scalable parallel computers. These systems comprise a collection of *nodes*, where each node consists of a processor with its own local memory and a router which supports message communication between nodes. The routers are connected by *channels* according to a particular interconnection topology. Among the most common topologies for multicomputers are low-dimensional meshes. These topologies are scalable and have a number of features that make them particularly amenable to high-performance computing [8], [13]. For example, a two-dimensional mesh topology is used in the Intel Touchstone DELTA [11] and the Symult 2010 [18] and a three-dimensional torus (mesh with wraparound connections) is used in Cray T3D [7] and Cray T3E [17].

In order to minimize network latency, the current generation of multicomputers employ the *wormhole routing* switching strategy [16]. Communication in the network can be either *unicast* or *multicast*. In unicast communication a message is sent from a source processor to a single destination processor, whereas in multicast communication a message is sent from a source processor to an arbitrary set of destination processors. Multicast communication has applications in a number of fundamental operations such as barrier synchronization [23], cache coherency in distributed share-memory architectures [14], and clock synchronization [1], among others.

In a path-based multicast scheme, a source node prepares a message for delivery to a set of destinations by first sorting the addresses of the destinations in the order in which they are to be delivered, and then placing this sorted list in the header flits of the message. When the header enters a router with address α, the router checks to see if α is the next address in the header. If so, the address α is removed from the message header and the data flits are forwarded both to the local processor at this node as well as to the next node on the path. Otherwise, the message is forwarded only to the next node on the path. In this way, the message is eventually delivered to every destination in the header.

In this paper we propose a deadlock-free path-based fault-tolerant multicast algorithm. First of all, a Hamiltonian path is constructed that does not need to be reconstructed when a faulty block is encountered. Instead, the path is updated in the neighborhood of faulty blocks. The routing algorithm is made deadlock-free by using two virtual channels. The main advantage of this approach is scalability, that is, the complexity of path reconfiguration does not increase rapidly when the number of faulty blocks increases.

The rest of the paper is organized as follows: Section 2 reviews some related work. Section 3 gives the notation and preliminary, where a path-based multicast algorithm without faulty block is proposed. Section 4 presents a path-based multicast algorithm of faulty blocks with an inter-block distance of at least three. Section 5 presents two extensions: one extension to 3-D meshes and the

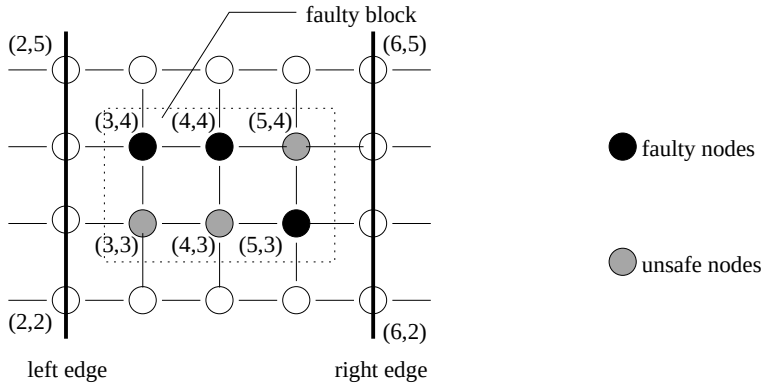

Figure 1: The propagation of the faulty block information

other to the faulty block model with a distance of at least two. Section 6 is the conclusion.

2 Notation and Preliminaries

2.1 2-D meshes and block fault model

A k-ary n-dimensional (n-D) mesh with $N{=}k^n$ nodes has an interior node degree of $2n$ and the network diameter is $(k-1)n$. Each node has an address $(a_1, a_2, ..., a_n)$, where $0 \leq a_i \leq k-1$. Two nodes $(a_1, a_2, ..., a_n)$ and $(b_1, b_2, ..., b_n)$ are connected if their addresses differ in one and only one element (dimension), say dimension i; moreover, $|a_i - b_i| = 1$. Basically, nodes along each dimension are connected as a linear array. Each node in a 2-D mesh is labeled as (x, y).

The fault model we use is block fault model which is defined as follows:

Definition 1: *In a 2-D mesh, a healthy node is unsafe if there are two or more unsafe or faulty neighbors. A faulty block contains all the connected unsafe and faulty nodes.*

The block fault model has the following interesting property: In a 2-D mesh, each faulty block is a rectangle and the distance between any two faulty blocks is at least three [21].

Definition 2: *The left (right) edge of a faulty block is a one-unit away parallel line to the left (right) side of the faulty block.*

In Figure 1, the left solid line is the left edge of the faulty block and right solid line is the right edge of the faulty block. The positions of left and right edges of a faulty block define its type which determines the way path reconfiguration is done.

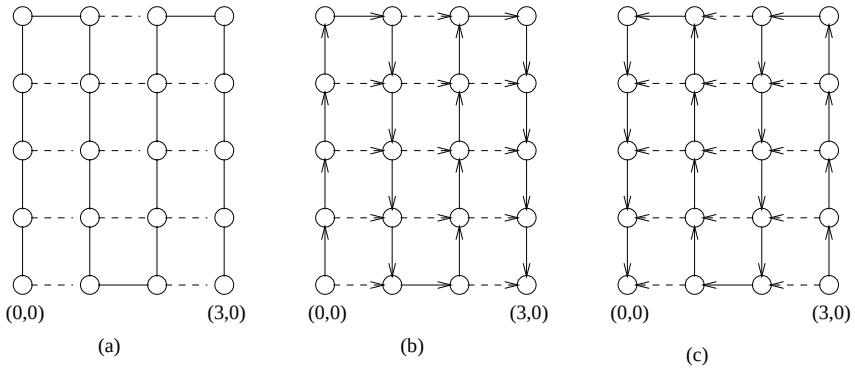

Figure 2: Example of (a) an undirected Hamiltonian path and the corresponding (b) P_f and (c) P_b directed networks of a mesh.

2.2 Path-based multicast algorithm in a fault-free 2-D mesh

First an undirected Hamiltonian path, which goes through each node exactly once, is constructed. An example of an undirected Hamiltonian path, with node $(0, 0)$ as an end node, is given in Figure 2 (a). The solid lines in the figure constitute the Hamiltonian path. From this two directed Hamiltonian paths can be constructed: one starts from $(0, 0)$, the P_f path (see the solid lines in Figure 2 (b)), and another ends at $(0, 0)$, the P_b path (see the solid lines in Figure 2 (c)). The links that are not part of the Hamiltonian path may be used to reduce path length and these links are called *shortcut*. Shortcuts are represented by dashed arrow lines in Figures 2 (b) and (c). For example, a shortcut from $(2, 0)$ to $(3, 0)$ saves eight steps. All the nodes in the system can be ordered based on the traversal order in the P_f path. $(x_1, y_1) \prec (x_2, y_2)$ means that the second node is after the first one in the P_f path starting from $(0, 0)$. The P_f network includes the P_f path and all the relevant shortcuts (see Figure 2(b)). Similarly, the P_b network includes the P_b path and all the relevant shortcuts (see Figure 2(c)).

2.3 Path-based fault-tolerant multicasting in 2-D meshes

The path-based multicast algorithm for a fault-free 2-D mesh can be extended to handle faulty blocks. If there is no fault, use the column-path-based algorithm. If there are faulty blocks in the system, construct detour paths around faulty blocks according to four cases in the P_f and P_b networks, respectively. We first look at a single faulty block. Based on the path directions of the left and right edges, there are four cases in constructing a detour path around a faulty block for the P_f network. That is, $(down, up)$, $(down, down)$, $(up, down)$, or (up, up). The corresponding detour paths are shown in Figures 3 (a), (b), (c), and (d), respectively. Similarly, there are also four cases

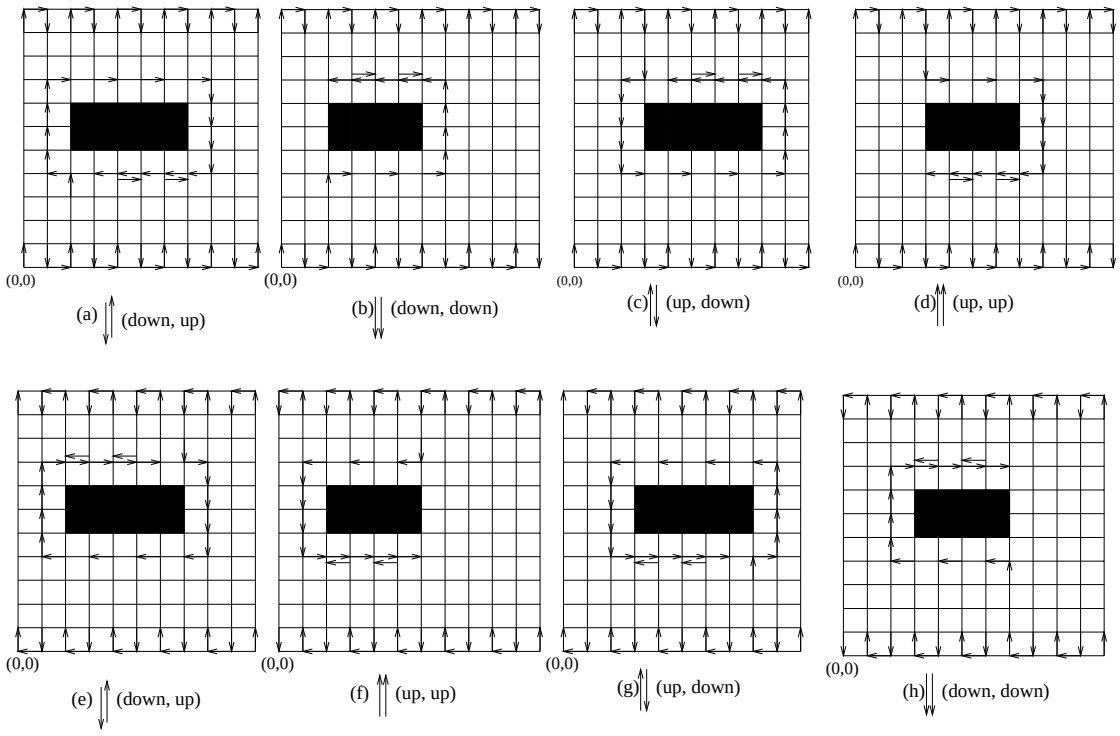

Figure 3: The four cases of routing around a faulty block in the P_f network.

in constructing a detour path in the P_b network when a message meets a faulty block (See Figure 3 (e), (f), (g), and (h), respectively). In our algorithm, we do not consider dynamic faults, i.e., it is assumed that a new faulty block comes during idle time.

If the network involves more than one faulty block, because the distance between two faulty blocks is at least three, reconfiguration can be done independently around each faulty block. We can still use the proposed method to establish the detour path around each individual faulty block. Figure 4 (a) shows an example of detour path involving three faulty blocks.

To facilitate the discussion on deadlock-free routing, we number the physical channels around the faulty block as shown in Figures 5(e) and (f), the clockwise physical channels are numbered from 1 to 4 and the counter-clockwise physical channels are numbered from 5 to 8. Figures 5(a),(b),(c),(d) are the representations of the four cases of detour path in the P_f network shown in Figures 3(a),(b),(c),(d), respectively. Channel $2a$ represents the left section of physical channel 2 and $2b$ represents the right section of physical channel 2. Channels $4a$ and $4b$ are defined in the same way. Figures 5(a)',(b)',(c)',(d)' are the representations of the four cases of detour path in the P_b network shown in Figures 3(e)(f)(g)(h). In all the eight cases in Figure 5, we simplify the

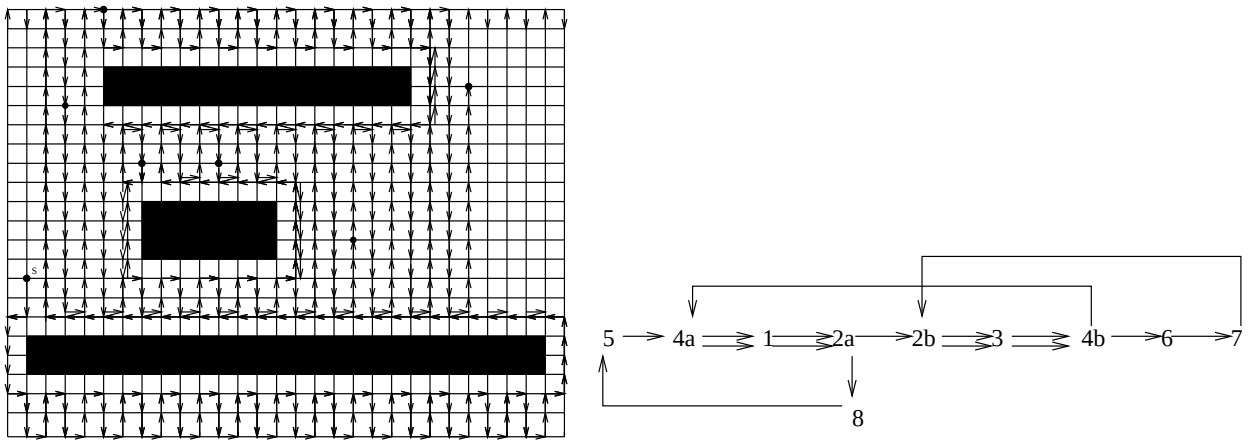

Figure 4: (a) Detour paths around multiple fault blocks (b) The dependency graph of physical channels in the $(down, up)$-type of the faulty block

presentation of all the sections of the detour path at each side of the faulty block that follow a particular direction and make them straight lines.

If we just use these physical channels for all the eight cases, deadlock may occur during the routing process; that is, there can be a cycle in the corresponding *channel dependency graph* [9]. For example, the channel dependency graph of cases Figures 5(a) and (a)' is shown in Figure 4(b) and we can see there are three cycles in the dependency graph. Bold lines represent channel dependencies in the P_b network and thin lines represent channel dependencies in the P_f network. *Virtual channels* can be used to avoid deadlock. Virtual channels are derived from a physical channel by multiplexing the physical channel into many logical (virtual) channels. We use two virtual channels for each physical channel. Thus the whole physical network can be divided into two virtual channel networks 0 and 1. That is, virtual channel network 0(1) consists of virtual channels 0(1) only. In Figure 5, the physical channels around the faulty block are assigned with either virtual channels in virtual channel network 0 or virtual channel network 1. Solid lines represent virtual channels in virtual channel network 0 while dashed lines represent virtual channels in virtual channel network 1. For example, physical channel 5 has two virtual channels: 5 in virtual channel network 0 and 5' in virtual channel network 1. Note that channels in Figure 5 include ones that are adjacent to faulty blocks and ones that are not adjacent to faulty blocks. Only the ones that are adjacent to faulty blocks are potentially used more than once in detour paths in the P_f and P_b networks. Therefore, virtual channels are used only for the channels that are adjacent to faulty blocks.

By assigning the virtual channels in this way, we proved the following theorem:

Theorem 1: *The fault-tolerant multicast algorithm is deadlock-free.*

Note that deadlock might still occur when there are dynamic faults, i.e., fault occur during a routing process. The handling of dynamic faults is much more complex and it will be considered in our future work.

The approach used in 2-D meshes with inter-block distance of at least three can be easily extended to 2-D meshes with inter-block distance of two and to 3-D meshes.

3 Conclusion

In this paper, we have presented a path-based fault-tolerant multicast algorithm in 2-D meshes with inter-block distance of at least three. The path we used is a Hamiltonian path that does not need to be reconstructed when a fault occurs. Instead, only the section around the faulty block is reconstructed and leave the rest of the path unchanged. We use two virtual channels to solve the deadlock problem. This is the first attempt to localize the affect of a faulty block to its neighboring nodes. In our future work, we plan to extend the design of path-based fault-tolerant multicast algorithm to cover dynamic faults, i.e., faults occur during a multicast process, and to consider multiple multicast [12].

References

[1] M. Azevedo and D. Blough, "Fault-tolerant clock synchronization of large multicomputers via multi-dimensional interactive convergence", *Technical Report ECE 94-12-03*, University of California, Irvine, December 1994.

[2] R. Boppana and S. Chalasani, "Fault-tolerant multicast communication in multiprocessors", *Proc. of the 1995 Int'l Conf. on Parallel Processing*, Vol. 1, August 1995, 118-125.

[3] R. V. Boppana and S. Chalasani, "Fault tolerant wormhole routing algorithms for mesh networks", *IEEE Transactions on Computers*, Vol. 44, No. 7, July 1995, 848-864.

[4] Y. M. Boura and C. R. Das, "Fault-tolerant routing in mesh networks", *Proc. of the 1995 Int'l. Conf. on Parallel Processing*, Vol. 1, 1995, 106-109.

[5] X. Chen and J. Wu, "Minimal routing in 3-D meshes using extended safety levels", *Proc. of the 10th IASTED Int'l Conf. on Parallel and Distributed Computing Systems*, October 1998, 106-109.

[6] A. A. Chien and J. H. Kim, "Planar-adaptive routing: low cost adaptive networks for multiprocessors", *Proc. of the 19th Int'l. Symp. on Computer Architecture*, May 1992, 268-277.

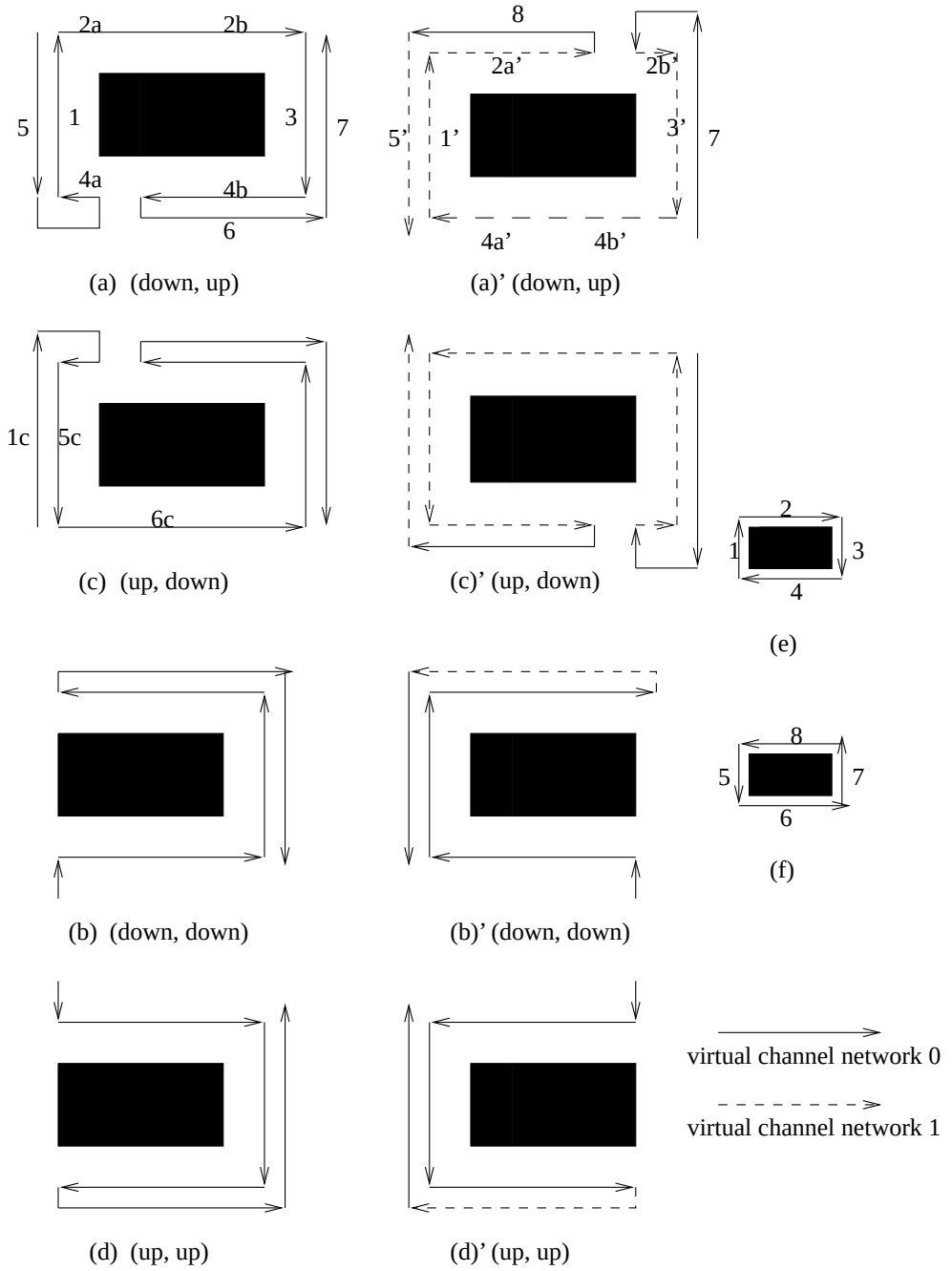

Figure 5: (a),(b),(c), and (d): The virtual channels around the faulty block in the P_f network. (a)',(b)',(c)', and (d)': The virtual channels around the faulty block in the P_b network. (e) and (f): channel labels of physical channels around the faulty block.

[7] C. M. Cunningham and D. R. Avresky, "Fault-tolerant adaptive routing for two-dimensional meshes", *Proc. of the 1st IEEE Symp. on High Performance Computer Arch.*, January 1995, 122-131.

[8] W. J. Dally, "Virtual channel flow control", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3. No. 2, March 1992, 194-205.

[9] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks", *IEEE Transactions on Computers*, Vol. C-36, No. 5, May 1987, 547-553.

[10] R. L. Hadas and E. Brandt, "Origin-based fault-tolerant routing in the mesh", *Future Generation Computer Systems*, Vol. 11, No. 6, October 1995, 603-615.

[11] S. L. Lillevik, "The Touchstone 30 gigaflop DELTA prototype", *Proc. of the 6th Distributed Memory Computing Conference*, 1991, 671-677.

[12] R. Kesavan and D. K. Panda, "Multiple Multicast with Minimized Node Contention on Wormhole k-ary n-cube Networks", *IEEE Transactions on Parallel and Distributed Systems*, Vol. 10, No. 4, April 1999, 371-393.

[13] F. T. Leighton, *Parallel Algorithms and Architectures: Arrays, Trees, and Hypercubes*, Morgan Kaufmann, 1992.

[14] K. Li and R. Schaefer, "A hypercube shared virtual memory", *Proc. of the 1989 Int'l. Conf. Parallel Processing*, volume 3, 1989, 125-132.

[15] X. Li, P. McKinley, and L. Ni, "Deadlock-free multicast wormhole routing in 2-D mesh multicomputers", *IEEE Transactions on Parallel and Distributed Systems*, Vol 5, No 8, Aug. 1994, 793-804.

[16] L. Ni and P. McKinley, "A survey of wormhole routing techniques in direct networks", *IEEE Computer*, Vol 26, No 2, Feb. 1993, 62-76.

[17] S. Scott and G. Thorson, "The Cray T3E network: adaptive routing in high performance 3D torus", *Proc. of Hot Interconnects Symp. IV, August 1996.*

[18] C. L. Seitz, "The architecture and programming of the Ameteck Series 2010 multicomputer", *Proc. of the 3rd Conf. on Hypercube Concurrent Computers and Applications*, Vol. 1, Jan. 1988, 33-36.

[19] C. C. Su and K. G. Shin, "Adaptive fault-tolerant deadlock-free routing in meshes and hypercubes", *IEEE Transactions on Computers*, Vol. 45, No. 6, June 1996, 672-683.

[20] Y. C. Tseng, M. H. Yang, and T. Y. Juang, "An Euler-Path-Based Multicasting Model for Wormhole-Routed Networks with Multi-Destination Capability", *Proc. of 1998 Int'l Conf. on Parallel Processing*, August 1998, 366-373.

[21] J. Wu, "Fault-tolerant adaptive and minimal routing in mesh-connected multicomputers using extended safety levels", *Proc. of the 18th Int'l. Conf. on Distributed Computing Systems*, May 1998, 428-435.

[22] J. Wu, *Distributed System Design*, The CRC Press, Boca Raton, FL, 1999.

[23] H. Xu, P. McKinley, and L. Ni, "Efficient implementation of barrier synchronization in wormhole-routed hypercube multicomputers", *Journal of Parallel and Distributed Computing*, Vol. 16, 1992, 172-184.