A Deterministic Fault-Tolerant and Deadlock-Free Routing Protocol in 2-D Meshes Based on Odd-Even Turn Model*

Jie Wu
Department of Computer Science and Engineering
Florida Atlantic University
Boca Raton, FL 33431
jie@cse.fau.edu

ABSTRACT

We propose a deterministic fault-tolerant and deadlock-free routing protocol in 2-dimensional (2-D) meshes based on dimension-order routing and the recently proposed odd-even turn model. The proposed protocol, called *extended X-Y routing*, does not use any virtual channels by prohibiting certain locations of faults and destinations. Faults are contained in a set of disjointed rectangular regions called faulty blocks. The number of faults to be tolerated is unbounded as long as nodes outside faulty blocks are connected in the mesh network. The extended X-Y routing can also be used under a special convex fault region called *orthogonal faulty block*, which can be derived from a given faulty block by activating some nonfaulty nodes in the block. Extensions to partial adaptive routing, traffic- and adaptivity-balanced using virtual networks, and routing without constraints using virtual channels and virtual networks are also discussed.

Categories and Subject Descriptors

B.8 [**Hardware**]: Performance and Reliability; C.2.2 [**Computer Systems Organization**]: Computer-Communication Networks—*network protocols*

General Terms

Algorithms, reliability, theory

Keywords

2-dimensional (2-D) meshes, deterministic routing, fault models, fault tolerance, freedom of deadlock, turn models, virtual channels, virtual networks.

*This work was supported in part by NSF grants CCR 9900646 and ANI 0073736.

ICS'02, June 22-26, 2002, New York, New York, USA.

1. INTRODUCTION

The direct network is a popular means to construct multicomputers, where a set of channels are used to connect each processor (node) to a limited neighbors. In a multicomputer system, routing algorithms provide mechanisms for communication between nodes. The performance of such a system depends heavily on the efficiency of routing algorithms. Routing algorithms are either deterministic or adaptive. Deterministic routing uses only one path to route packets from a source to a destination, while adaptive routing makes use of many different routes. Most commercial systems use deterministic routing because of its deadlock freedom and ease of implementation.

Dimension-order routing is a commonly used deterministic routing algorithm in mesh-connected multicomputers which include meshes, tori (meshes with wraparound connections), and hypercubes. In dimension-order routing, a routing packet is routed in one dimension at a time (the offset between the source and destination nodes is reduced to zero along that dimension). X-Y routing is an example of dimension-order routing used in 2-dimensional (2-D) meshes and tori. In X-Y routing, the packet is routed first in the x dimension and then in the y dimension. Unfortunately, X-Y routing is not fault-tolerant and it cannot tolerate even a single fault.

Designing a deterministic routing protocol that is both fault-tolerant and deadlock-free poses a major challenge. The wormhole switching technique used in the latest generation of multicomputers is subject to deadlock more than packet switching. In addition, wormhole switching tends to support routing with less fault tolerance. Wormhole routing divides a message into packets and packets into flits. It then routes flits through the network in a pipeline fashion. When the header flits reach a node that has no output channel available, all of the flits are blocked where they are (in place). A deadlock occurs when some packets from different messages cannot advance toward their destinations because the channels requested by them are not available. All the packets involved in a deadlocked configuration are blocked forever. Deadlock avoidance is a commonly used approach in which channels are granted to a packet in such a way that a request never leads to a deadlock. To achieve fault tolerance, faults are normally contained in a set of disjointed rectangular regions called *faulty blocks*. Each faulty block may include some non-faulty nodes as shown in Figure 1 (a). The convexity of faulty blocks facilitates a simple design of deadlock-free routing. To design a deadlock-free routing, virtual channels [6] and virtual networks [13] are usually used to provide a certain

degree of routing freedom to route around a faulty block.

In this paper, we propose a deterministic fault-tolerant and deadlock-free routing protocol in 2-D meshes based on X-Y routing and the recently proposed *odd-even turn model* [5], an extension to Glass and Ni's turn model [11] where certain turns are prohibited to avoid deadlock. The proposed protocol, called *extended X-Y routing*, does not use any virtual channels by prohibiting certain locations of faults and destinations. The main purpose of posing such restrictions is to better present our idea without going into messy details of boundary situations. The number of faults to be tolerated is unbounded as long as nodes outside faulty blocks are connected in the resultant mesh network. Each faulty block is surrounded by a *boundary ring* consisting of four boundary lines, one for each direction. However, the boundary line defined at the east (and west) side of each faulty block consists of two lines: one in an even column (column with an even label) and one in an odd column (see Figure 1). The faulty block is so defined that nodes on the boundary lines of a faulty block (simply called boundary nodes) do not intersect with any other faulty block. In the absence of faults, the extended X-Y routing works like a regular X-Y routing which routes packets along the x dimension first followed by the y dimension. When packets reach a boundary node of a faulty block, the boundary ring is used to route packets around the block. Two boundary lines at the east and west of a faulty block (except the one at the edge of the mesh) are used to avoid certain turns as specified in the odd-even turn model.

The extended X-Y routing can also be applied to a special convex fault region called an *orthogonal faulty block* [19], which can be derived from a given faulty block after activating some nonfaulty nodes in the block. The *localized algorithm* [8, 20], a special type of decentralized algorithms, is used to construct faulty blocks, orthogonal faulty blocks, and boundary lines. Extensions to partial adaptive routing, traffic- and adaptivity-balanced using virtual networks, and routing without constraints using virtual channels and virtual networks are also discussed.

The following are assumptions used in this paper: (1) Only node faults are considered and they are contained in a set of disjointed faulty blocks defined in the paper. (2) The fault model is static, that is, no new faults occur during a routing process. (3) Both source and destination nodes are outside any faulty block. In addition, the destination is not a boundary node of any faulty block. (4) Faults do not appear at four edges of a mesh. In addition, no fault appears at two columns that are adjacent to the west and east edge of the mesh. (5) A connected 2-D mesh is used with four directions: North (positive x), South (negative x), East (positive y), West (negative y). (Unlike the convention the north-south axe is used here as the x axe to match the notation used in the odd-even turn model.)

Note that almost all the above assumptions can be relaxed. Link faults can be treated as node faults by considering its end nodes faulty. Since the faulty block can be calculated quickly in a few rounds, the proposed approach can be extended to a dynamic fault model with a careful design. Condition (3) can be removed by using two virtual channels as will be shown in the extension of the approach. Condition (4) is used to avoid handling complex boundary situation. If condition (4) fails, either nodes of the corresponding edge(s) are disabled and removed from the mesh or virtual channels are introduced (as used in many existing approaches) to route around faulty blocks that are at the edge of the mesh.

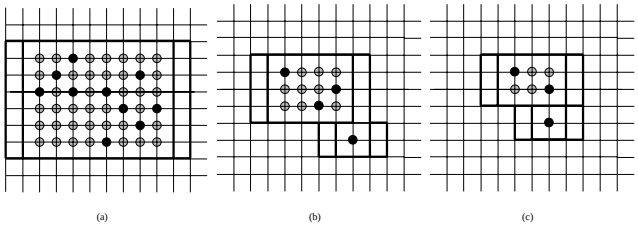

Figure 1: Three examples of faulty blocks where black nodes are faulty nodes and gray nodes are nonfaulty nodes.

The rest of the paper is organized as follows: Section 2 discusses related work. Section 3 provides preliminaries where the odd-even turn model is reviewed, the general methodology of localized algorithms is discussed, and an extended faulty block model is introduced. Section 4 proposes the extended X-Y routing, which is a fault-tolerant and deadlock-free routing protocol without using virtual channels. Section 5 extends the protocol to a 2-D mesh with orthogonal faulty blocks. A localized algorithm for the formation of orthogonal faulty blocks is also included. Section 6 lists ideas for other possible extensions. Section 7 concludes the paper and discusses possible future work.

2. RELATED WORK

Virtual channels [6] were first introduced to prevent deadlock and offer adaptivity in routing, not for fault tolerance. Duato [7] provided a general deadlock-free routing approach by introducing the notion of escape channels. In this general approach, virtual channels are divided into two groups: one for nonminimal adaptive routing and the other (called escape channels) for minimal, deterministic routing. Park and Agrawal [14] discussed a similar design methodology for deadlock-free routing but routing functions are based on the history of channels in addition to destination information. Fleury and Fraigniand [9] gave a comprehensive survey on different deadlock-free routing protocols.

Linder and Harden [13] were the first to use virtual channels and virtual networks to achieve fault tolerance. Their method requires $O(2^n)$ virtual channels for a fully adaptive fault-tolerant routing in an n-D mesh. Using virtual channels has some disadvantages, for example, routers based on virtual channels require more gates and time compared with those not based on virtual channels. To reduce the number of virtual channels, Chien and Kim [4] introduced the planar adaptive routing which provides partial adaptivity in an n-D mesh by first dividing the routing process into a sequence of phases and then forwarding packets in two dimensions within each phase.

In recent years many deterministic fault-tolerant routing algorithms in 2-D meshes and tori have been proposed. In Boppana and Chalasani's approach [1], fault regions are surrounded by either fault rings or fault chains (at boundaries of a mesh). When a packet encounters a fault region, a fault ring or chain is used to route the packet around the region. Deadlock is avoided by using four virtual channels per physical channel for dimension-order routing. Many extensions based on Boppana and Chalasani's approach have been proposed [2, 3, 16, 17, 21]. These extensions try to reduce either the number of nonfaulty nodes in a faulty block by considering dif-

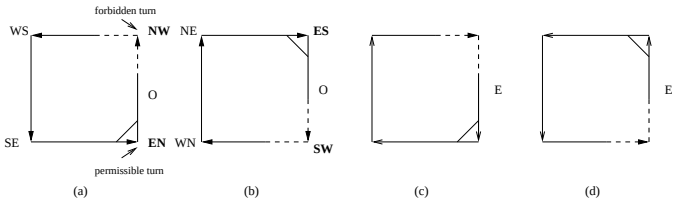

Figure 2: Permissible EN, NW, ES, and SW turns.

ferent types of fault regions or the number of virtual channels. So far the best results can reduce the number of virtual channels to two or three depending on the type and distance between faulty blocks used. To our knowledge, there is no deadlock-free dimension-order routing that can tolerate unlimited number of faults without using virtual channels.

Glass and Ni's fault-tolerant routing [10] in meshes without using virtual channels is based on the turn model [11]. However, its fault tolerance capability is limited to $n-1$ in an n-D mesh, i.e., one fault in a 2-D mesh. Fault-tolerant routing without using virtual channels exists for non-dimension-order routing. For example, fault-tolerant path-based routing [12] is based on finding a Hamiltonian path or pseudo Hamiltonian path in a faulty mesh or torus. However, it is nonminimal, and unlike the faulty block model, path information is difficult to maintain in a localized way. Note that routing that allows backtracking can potentially tolerate unlimited number of faults such as the one proposed by Suh et al [15]. In such an approach routing history is coded in the header to navigate the routing process.

3. PRELIMINARIES

In this section, we first review the mesh topology and Chiu's odd-even turn model which is an extension to Glass and Ni's turn model. We then discuss the general methodology of localized algorithms. Finally, we introduce an extended faulty block model.

3.1 2-D meshes

A 2-dimensional (2-D) mesh with n^2 nodes has an interior node degree of 4. Each node u has an address u: (u_x, u_y), where $u_x, u_y \in \{0, 1, 2, ..., n-1\}$. Two nodes u: (u_x, u_y) and v: (v_x, v_y) are connected if their addresses differ in one and only one dimension, say x, moreover, $|u_x - v_x| = 1$. Each node in a 2-D mesh has four neighbors, one in each of four directions: east, south, west, and north.

3.2 Odd-even turn model

Chiu [5] proposed an odd-even turn model, an extension to Glass and Ni's turn model [11]. In general, deadlock avoidance tries to avoid the formation of a cycle, which is a necessary condition for deadlock. A cycle in a mesh consists of several turns. For examples, SW (south-west), WN, NE, and ES turns are essential in a clockwise cycle. The X-Y routing is made deadlock-free by prohibiting a turn from the y dimension to the x dimension. Specifically, four types of turns are disallowed: two in a clockwise cycle and two in a counterclockwise cycle. The basic concept behind the turn model is to prohibit a minimum number of turns, and hence, increase the routing adaptivity. In general, only one turn is prohibited in each cycle. For example, in a positive-first turn model two types of turns are disallowed (one for each cycle), that is, the turns from the negative to positive directions. The odd-even turn model restricts the locations where some of the turns can occur so that an EN (east-north) turn and an NW turn are not taken at nodes in the same column, and neither are an ES turn and a SW turn. Specifically, the odd-even turn model tries to prevent the formation of the *rightmost column segment of a cycle*. Chiu gives two rules for turn [5]:

Rule 1: *Any packet is* not *allowed to take an EN turn at any node located in an even column, and it is* not *allowed to take an NW turn at any node located in an odd column.*

Rule 2: *Any packet is* not *allowed to take an ES turn at any node located in an even column, and it is* not *allowed to take a SW turn at any node located in an odd column.*

Figure 2 shows these two rules on the EN, NW, ES, and SW turns. These four turns are called *sensitive turns*. Turns without restriction are called *insensitive turns*. A small triangle is placed at each sensitive turn that is permissible (as shown in Figure 2). Forbidden turns are represented as ones with dashed lines. A turn in an even (odd) column is represented by E (O). Basically in odd-even turn model, *once east-bound starts, no more west-bound is allowed in the routing process*. Again, four directions are defined as: East ($+y$), South ($-x$), West ($-y$), and North ($+x$).

3.3 Localized algorithms

In a *localized algorithm* [8, 20], which is a special type of decentralized algorithms, each processor (process) interacts with others in a restricted vicinity, but nevertheless collectively achieves a desired global objective. This type of algorithms are usefully in a system with a set of independent, autonomous processors (processes). In general, each processor (process) performs exceedingly simple tasks, such as maintaining and propagating information "markers". In this paper, we study several localized algorithms in which only neighboring nodes exchange and update their markers.

3.4 Faulty blocks

We first introduce a special faulty block model. Faulty nodes in a 2-D mesh are contained in a set of disjointed rectangular faulty blocks. The *regular faulty block* model is defined as follows: *All nonfaulty nodes are safe initially. A nonfaulty node is changed to unsafe if it has two unsafe or faulty neighbors in different dimensions.* In the extended faulty block model proposed in this paper, each faulty block is surrounded by a boundary ring consisting of four boundary lines, one for each direction. The boundary line at the east (and west) side of the block consists of two lines. Two faulty blocks are disjointed if the boundary ring of one faulty block does not intersect with the other faulty block. The following gives a formation of any faulty block by first classifying nonfaulty nodes into unsafe and safe.

Definition 1: *All nonfaulty nodes are* safe *initially. A nonfaulty node is changed to* unsafe *if*

1. *it has two unsafe or faulty neighbors that are not all in the x dimension; or*

Safe/unsafe status:

1. all nonfaulty nodes are initialized to *safe*;
2. **repeat**
3. **doall**
4. (1) nonfaulty node *u* exchanges its status with its neighbors. In addition, the status of its east (west) neighbor is passed to its west (east) neighbor.
5. (2) change *u*'s status to *unsafe* if
6. (a) it has two unsafe or faulty neighbors that are not all in the *x* dimension, or
7. (b) it has an unsafe or faulty neighbor along the *x* dimension and an unsafe or faulty 2-hop neighbor along the *y* dimension.
8. **odall**
9. **until** there is no status change

Figure 3: A localized algorithm for determining safe/unsafe status.

2. it has an unsafe or faulty neighbor in the x dimension and an unsafe or faulty 2-hop neighbor (neighbor's neighbor) in the y dimension.

An extended faulty block consists of connected unsafe and faulty nodes. Note that although both unsafe and faulty nodes are included in faulty blocks, they are treated differently as will be seem later in the orthogonal faulty block model where certain unsafe nodes can be activated by removing from the blocks. The difference between the extended faulty block definition (Definition 1) and the conventional one lies in the different treatments of adjacent nodes in different dimensions. An extended faulty block and its boundary nodes are so defined to facilitate fault-tolerant routing based on the odd-even turn model to be discussed in the next section. It can be easily shown that faulty blocks in 2-D meshes are disjointed rectangles. Let *u*: (u_x, u_y) and *v*: (v_x, v_y) be two nodes in a 2-D mesh, $d(u,v) = |u_x - v_x| + |u_y - u_y|$ denotes the distance between *u* and *v*. The *distance* between two faulty blocks *A* and *B* is defined as $d(A,B) = \min_{u \in A, v \in B}\{d(u,v)\}$. It can be easily shown that the distance between any two faulty blocks, *A* and *B*, is at least 3 along the *y* dimension or is at least 2 along the *x* dimension. Figure 1 shows three faulty blocks with boundary rings shown in boldface. In the subsequent discussion, a faulty block refers to an extended faulty block unless otherwise specified.

In the localized algorithm for safe/unsafe status (see Figure 3), each nonfaulty node is marked either safe or unsafe. Neighbors exchange and update their markers. Eventually, connected unsafe and faulty nodes form a faulty block (which is a global objective). To facilitate the decision process of node status, each node sends its status to its 2-hop neighbors along the *y* dimension. It is assumed that each node knows the status of its neighbors.

Theorem 1: *Boundary nodes of a faulty block do not intersect with any other faulty block.*

Proof: Assume that node *u* is a boundary node of a faulty block *A*, i.e., it is either 1-hop neighbor along dimension *x* or 1-hop or 2-hop neighbor along dimension *y*. Assume node *u* also belongs to faulty block *B*. Based on the faulty block definition, faulty blocks *A* and *B* should be combined to form a single block. This brings a contradiction. ■

Extended X-Y routing:

1. /* `the packet is sent to an even column` */
   (a) If the source is in an odd column and Δ_x is non-zero, then the packet is sent to its west neighbor in an even column.
2. /* `phase 1: reduce` Δ_x */
   (a) (Normal mode) reduce Δ_x to zero by sending the packet north (or south) (with no 180° turn).
   (b) (Abnormal mode) when a north-bound (south-bound) packet reaches a boundary node of a faulty block, it is routed around the block clockwise (counter-clockwise) by following the boundary ring of the faulty block as shown in Figure 4 (a) (Figure 4 (b)). The packet takes the first even column turn whenever possible and step (a) is followed.
3. /* `phase 2: reduce` Δ_y */
   (a) Once Δ_x is reduced to zero, an NW or NE turn is performed for the north-bound packet (see Figure 4 (a)) and a SW or SE turn is performed for a south-bound packet (see Figure 4 (b)). The selection of a turn depends on the relative location of the destination to the current node.
   (b) (Normal mode) reduce Δ_y to zero by sending the packet east (west) (with no 180° turn).
   (c) (Abnormal mode) when a east-bound (west-bound) packet reaches a boundary node of a faulty block, it is routed around the block, clockwise or counterwise, along odd columns of the boundary ring as shown in Figure 4 (c) (even columns of the boundary ring as shown in Figure 4 (d)). Routing around the block is completed when Δ_x is again reduced to zero and step (b) is followed.

Figure 5: Extended X-Y routing.

Note that although boundary nodes of a faulty block do not intersect with any other faulty block, boundary nodes of different faulty blocks may overlap, i.e., a node can be a boundary node of more than one faulty block. The complexity of the safe/unsafe status procedure, in terms of the number of rounds needed, is the maximum diameter of faulty blocks in the mesh: $\max\{diam(A)\}$.

4. EXTENDED X-Y ROUTING

We propose a deterministic routing process, called *extended X-Y routing*, which consists of two phases, similar to a regular X-Y routing. (That is why it is still belong to dimension-order routing although the other dimension is used within each phase to bypass faulty blocks encountered.) In phase 1, the offset along the *x* dimension is reduced to zero, and in phase 2, the offset along the *y* dimension is reduced to zero. Assume source and destination nodes are both safe. Let $s: (s_x, s_y)$ and $d: (d_x, d_y)$ be the source and destination nodes, respectively. $\Delta_x = |d_x - s_x|$ and $\Delta_y = |d_y - s_y|$ are offsets along dimension *x* and dimension *y*, respectively. The extended X-Y routing provides a special implementation of the requirement posed in the even-odd model and, at the same time, supports fault-tolerant routing.

The extended X-Y routing (shown in Figure 5) follows the reg-

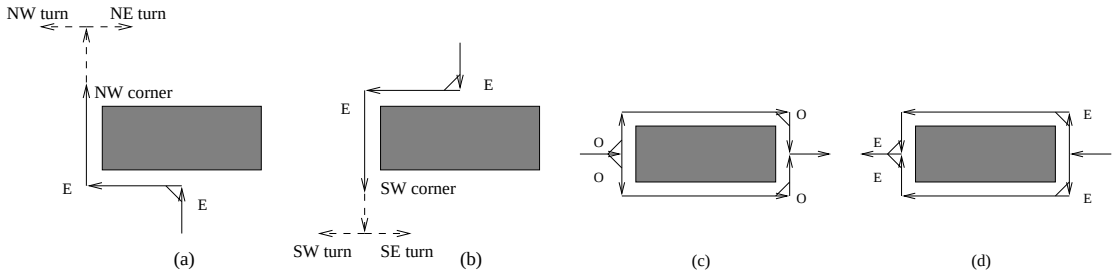

Figure 4: Two cases of routing along the x dimension (column) (a) north-bound and (b) south-bound and two cases along the y dimension (row) (c) east-bound and (d) west-bound.

ular X-Y routing (and the packet is in a "normal" mode) until the packet reaches a boundary node of a faulty block. At which point, the packet is routed around the block (and the packet is in an "abnormal" mode) clockwise or counterclockwise based on the following rules: When a routing packet routes around a faulty block following the boundary ring, the corresponding block is called the *routing block*. During phase 1 the packet is routed around the routing block through the west side of the block. Even columns are used to route the packet along the x dimension (column). In phase 2, to route around the routing block, odd columns (even columns) are used to perform routing along the y dimension when the packet is east-bound (west-bound) (see Figure 4 (c) and (d)). The packet is routed around the routing block either clockwise or counterclockwise in phase 2 (see Figures 4 (c) and (d)). Note that during the normal mode of routing the packet along the x or y dimension, no 180° turn is allowed. For example, the positive x direction cannot be changed to the negative x direction.

A special case occurs when the destination is at the east side of the routing block. In this case, when phase 1 completes, the routing packet is still at the west side of the routing block as shown in Figure 6. The even (marked as E) boundary column of the routing block is switched to the odd (marked as O) boundary column (two subcases are shown in Figures 6 (a) and (b)), and then, the packet is routed around the block either clockwise or counterclockwise. Figure 7 shows three routing examples (s_i, d_i), with $i \in \{1, 2, 3\}$, in a 10×10 mesh (wraparound connections are not shown) with four faulty blocks F_1, F_2, F_3, and F_4. Note that when the routing packet reaches a northwest (NW) or southwest (SW) corner of a routing block in phase 1, the packet goes straight, north-bound or south-bound, without further routing around the block (see Figures 4 (a) and (b)).

A fault-tolerant routing process is livelock-free if it can deliver packets from the source to destination, regardless of the number and location of faults. The following result shows that the extended X-Y routing is both deadlock-free and livelock-free in a 2-D mesh where faults are contained in a set of disjointed faulty blocks.

Theorem 2: *The extended X-Y routing is deadlock-free and livelock-free.*

Proof: In the routing process, all sensitive turns are permissible, based on the results from the odd-even turn model [5], the extended X-Y routing is deadlock-free. To show the livelock-free property, we only need to show that Δ_x (Δ_y) is eventually reduced to zero in

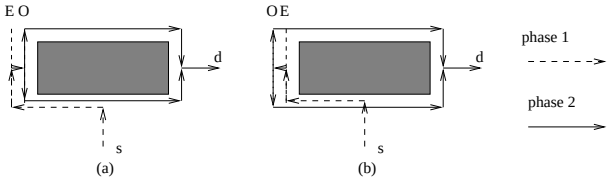

Figure 6: Two subcases of special routing.

phase 1 (phase 2). In phase 1, Δ_x is always reduced by one at each step (with no 180° turn), except when the packet is routed around a faulty block by going west. There are two cases of west-bound hops, one case is in the first hop for the packet to reach an even column and the other case occurs when the packet is routed around a faulty block. Since the size of each faulty block is limited, the number of west-bound hops is a finite number. In addition, although the packet may have to route around several faulty blocks (or several abnormal modes), Δ_x is reduced by at least one in a normal mode between every two adjacent abnormal modes. Therefore, Δ_x is reduced to zero in a finite number of steps in phase 1. Similarly, Δ_y is also reduced by one at each step in phase 2, except when the packet is routed around a faulty block. Since the size of each faulty block is limited, using the same argument used in phase 1, Δ_y is eventually reduced to zero in a finite number of steps in phase 2. Note that when a packet is routed around a faulty block in phase 2, Δ_x may temporarily become non-zero, based on the routing process, Δ_x is reduced to zero again when the process of routing around the faulty block is completed. ■

Note that the destination is not a boundary node of any faulty block. This is to prevent the following case: If the destination is at the east side of a faulty block and it is on an even boundary line which is closer to the block than the odd boundary line, then the rightmost column segment of a cycle may be constructed when a east-bound message routes around the block as shown in Figure 4 (c). However, this restriction can be removed by introducing two virtual channels as will be shown in Section 6.

5. ORTHOGONAL FAULTY BLOCKS

A faulty block may include many nonfaulty nodes labeled as unsafe as shown in Figure 1. Many unsafe nodes can be activated and removed from a faulty block while still keeping its convexity. The following definition provides such a special convex fault region.

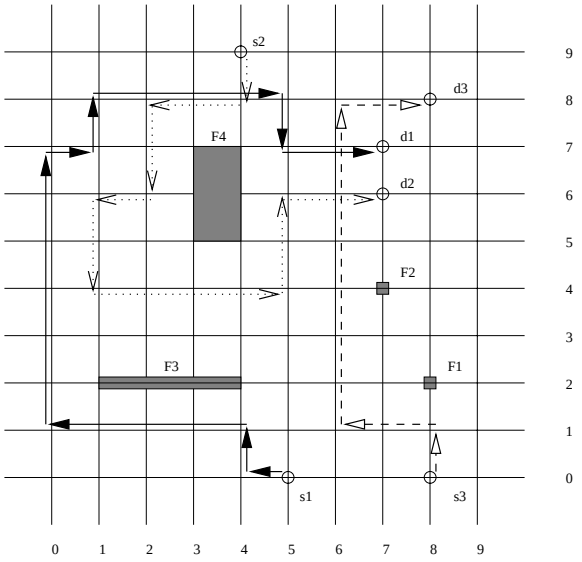

Figure 7: Three routing examples in a 10 $\times$ 10 mesh with four faulty blocks.

Definition 2 [19]: *A region is* orthogonal convex *if and only if the following condition holds: For any horizontal or vertical line, if two nodes on the line are inside the region, then all the nodes on the line that are between these two nodes are also inside the region.*

A fault region that is orthogonal convex is called an *orthogonal faulty block*. The proposed extended X-Y routing can be easily extended to 2-D meshes with orthogonal faulty blocks. When a routing block is orthogonal, care should be taken in phase 2 when a routing packet goes around the block. The selection of either the clockwise or counterclockwise direction to route around a fault region becomes important (and is no longer arbitrary). The formation of the rightmost column of a cycle may occur as shown in Figure 11 (a) when a east-bound packet goes around the block clockwise. To avoid this situation, the boundary lines around east and west sides of an orthogonal faulty block are associated with *directional information* as shown in Figure 8 (a). Basically, boundary nodes at the west (east) side of an orthogonal faulty block should "point toward" east (west). The formation of directional information together with boundary nodes of an orthogonal faulty block using a localized algorithm is shown in [18]. It is assumed that directional information exists at boundary nodes of orthogonal convex regions. Let $\min_x$ and $\max_x$ ($\min_y$ and $\max_y$) be the minimum and maximum coordinates of the original faulty block of an orthogonal faulty block A along dimension x (dimension y), respectively. The rectangle spanning from nodes $(\min_x, \min_y)$ and $(\max_x, \max_y)$ is called a *container* for block A. One additional assumption is added: The source is outside any orthogonal faulty block and the destination is outside the container of any orthogonal faulty block.

Once directed boundary lines are defined, the extended X-Y routing can be directly applied. The only change occurs in phase 2, the routing packet has to follow the directed boundary lines when route around a routing block. In phase 1, the packet still goes west-bound around a routing block with one minor, but subtle, change: When the packet reaches a northeast (NE) or southeast (SE) corner (see Figure 8 (a)) of a routing block, the packet should be sent west immediately. Figures 8 (b) and (c) show two routing examples, where routing blocks are orthogonal faulty blocks and s and d represent source and destination, respectively. In phase 1, routing around a routing block completes when either the packet reaches the leftmost boundary line or Δ_x is reduced to zero. In the former case (as shown in Figure 8 (c)), the packet is then routed along dimension x until Δ_x is reduced to zero. In phase 2, routing around a routing block completes when either the packet reaches the rightmost boundary line or Δ_x is reduced to zero. In the former case, the packet is routed along dimension x until Δ_x is reduced to zero, and then Δ_y is reduced by sending the packet along dimension y. In the latter case, the packet is routed directly along dimension y to reduce Δ_y (as shown in the example (s_1, d_1) of Figure 8 (b)).

Theorem 3: *Using the orthogonal faulty block model, the modified extended X-Y routing is still deadlock-free and live-lock-free.*

Proof (Sketch): In phase 1, assume that the packet is north-bound (the south-bound case can be treated in a similar way), routing around a routing block involves a sequence of the following turns: NW, (WS, SW)*, (WN, NW)*, WN, where (WS, SW)* represents a zero or more repetitions of WS, SW turns (as shown in the example (s_2, d_2) of Figure 8 (b)). All sensitive turns, NW and SW, occur in even columns and they are permissible. In the transition between phase 1 and phase 2, either an NW or NE is performed in an even column. In phase 2, assume that the packet is east-bound (the west-bound case can be treated in a similar way), routing around a routing block (if any) involves a sequence of the following turns if the packet is routed in the counter-clockwise direction: ES, (SE, ES)*, SE, (EN, NE)*, EN. If the packet is routed in the clockwise direction, the following sequence of turns is used: EN, (NE, EN)*, NE, (ES, SE)*, ES. All sensitive turns are performed in odd columns and they are permissible. Once Δ_x is reduced to zero, either a SE or NE turn is performed in an odd column to reduce Δ_y. The fact that the destination being outside a container ensures that the packet is still east-bound after phase 2. Hence, the modified extended X-Y routing is still deadlock-free and livelock-free. ■

In the following, we propose a simple decentralized formation of orthogonal faulty blocks from a given set of faulty blocks. Given a faulty block, the corresponding orthogonal faulty block(s) can be derived by assigning *enabled/disabled* status to unsafe nodes in the faulty block. Basically, nonfaulty nodes are associated with a set of status: safe/unsafe and enabled/disabled. Node status is used not only for the formation of orthogonal faulty blocks but also for the eligibility check for the source and destination nodes.

Definition 3 [19]: *All safe nodes are marked* enabled. *An unsafe node is initially marked* disabled. *It is changed to the* enabled *status if it has two or more enabled neighbors.*

An orthogonal faulty block consists of connected disabled and faulty nodes. Wu [19] showed that a fault region derived from the enabled/disabled process is an orthogonal convex polygon. In addition, each region is the *smallest orthogonal convex polygon that covers all the faulty nodes within the region.*

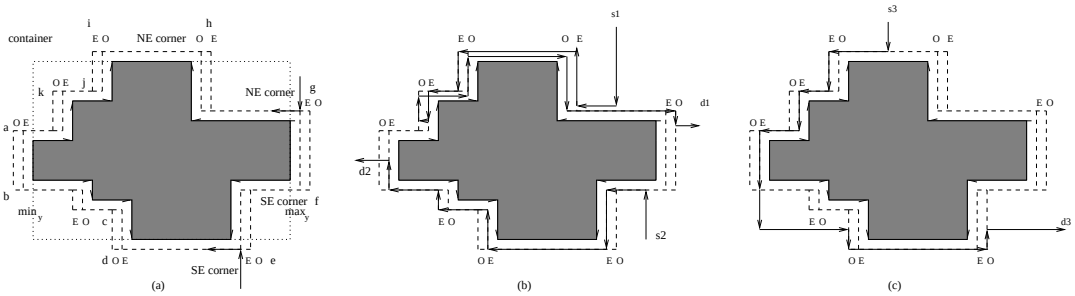

Figure 8: Routing along orthogonal faulty blocks.

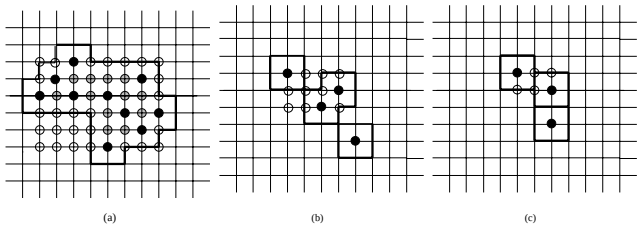

Figure 9: The orthogonal faulty blocks of three faulty blocks in Figure 1.

Figure 9 shows the corresponding orthogonal faulty blocks for three faulty blocks in Figure 1. The boundary nodes of an orthogonal faulty block are defined in the same way as in a faulty block. Two boundary lines are at the east and west side of the block (only one line is shown in Figure 9). White nodes are unsafe but enabled nodes. Gray nodes are unsafe and disabled nodes. For the faulty block in Figure 1 (a), the corresponding orthogonal faulty block is substantially reduced. The two faulty blocks in Figure 1 (b) are partitioned into three orthogonal faulty blocks as shown in Figure 9 (b).

However, the two faulty blocks in Figure 1 (c) are partitioned into three orthogonal faulty blocks but two of them do not meet the boundary node condition (Theorem 1). The problem in the example of Figure 9 (c) is that unsafe nodes in between two faulty nodes (at two adjacent rows) along the *y* dimension should not be enabled to ensure that boundary nodes of an orthogonal faulty block do not intersect with another block. We will explicitly mark these nodes disabled in the following extended enabled/disabled status definition.

Definition 4: *All safe nodes are marked* enabled. *Unsafe nodes can be remarked in the following sequence:*

1. *An unsafe node is remarked* semi-faulty *if it has a faulty south or north neighbor.*
2. *An unsafe or semi-faulty node is remarked* disabled *if the status of its east and west neighbors belong to one of the following three cases: faulty and semi-faulty, semi-faulty and faulty, or faulty and faulty, respectively.*
3. *An unsafe or semi-faulty is remarked* enabled *if it has two or more enabled neighbors.*

Extended enabled/disabled status:

1. all safe nodes are *enabled*;
2. unsafe nodes with faulty south or north neighbors are remarked *semi-faulty*;
3. unsafe or semi-faulty nodes are remarked *disabled* if both east and west neighbors are faulty or semi-faulty (with at least one being faulty);
4. **repeat**
5. **doall**
6. (1) unsafe or semi-faulty node *u* exchanges its status with that of its neighbors.
7. (2) remark *u*'s status to enabled if it has two or more enabled neighbors.
8. **odall**
9. **until** there is no status change

Figure 10: A localized algorithm for determining extended enabled/disabled status.

Unlike Definition 3, an unsafe node in a faulty block may not be assigned an enabled/disabled status in Definition 4. A semi-faulty node resembles an unsafe node and it is so labeled just to determine disabled nodes. To apply the extended X-Y routing in meshes with orthogonal faulty blocks, the source should be an enabled node and the destination should be a safe node that is not a boundary node of an orthogonal faulty block.

The connected nodes with status other than safe or enabled form an *extended orthogonal faulty block*. Since an extended orthogonal faulty block is generated from a given faulty block, the complexity of extended enabled/disabled status procedure is still the maximum diameter of faulty blocks. In the subsequence discussion, an extended orthogonal faulty block is simply called an orthogonal faulty block. Figure 11 shows the result of applying extended enabled/dis-abled process to the examples in Figure 1. The nodes with a cross are disabled nodes. Semi-faulty nodes are not explicitly marked since they can be easily identified. In the localized algorithm for extended enabled/disabled status (see Figure 10), three sets of markers are used: faulty and semi-faulty, safe and unsafe, and enabled and disabled.

Proposition 1: *A fault region derived from the extended enabled-/disabled process is an orthogonal convex polygon.*

Proposition 2: *Any boundary node of an orthogonal faulty block derived from the extended enabled/disabled process does not be-*

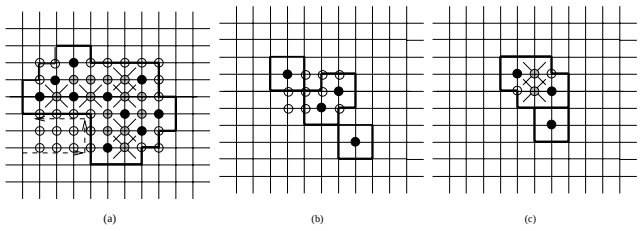

Figure 11: The extended orthogonal faulty blocks of three faulty blocks in Figure 1.

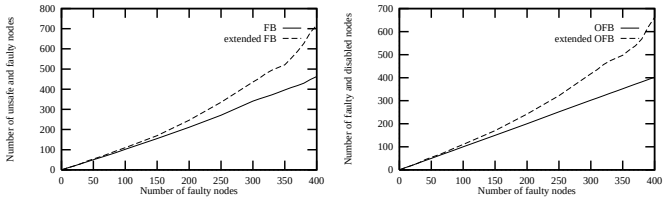

Figure 12: The numbers of nodes covered in (a) FBs and extended FBs and (b) OFBs and extended OFBs.

long to any other orthogonal convex polygon.

The proofs of Propositions 1 and 2 follow directly the ones used in [19] for properties of regular orthogonal faulty blocks. Note that the extended orthogonal faulty block is no longer the smallest convex region covering all faults in the region. However, the marking process can be enhanced to reduce its size. For example, step 2 in Definition 4 can be changed to the following: *an unsafe or semi-faulty node is marked disabled if its east neighbor is faulty and its west neighbor is either faulty or semi-faulty*. With this change, the status of two disabled nodes in Figure 11 (a) will be changed: one to unsafe status and the other (at the bottom row) to the enabled status. In Figure 11 (c), the disabled node at the bottom row is enabled.

To see the difference between regular faulty blocks (FB) (orthogonal faulty blocks (OFB) (Definition 3)) and extended faulty blocks (Definition 1) (extended OFB (Definition 4)) in terms of the amount of nonfaulty nodes included in these blocks, we conducted a simulation study on a 100×100 mesh where a number of faults are randomly generated. Figure 12 (a) shows the numbers of nodes covered in FBs and extended FBs for given number of faults. Figure 12 (b) shows the numbers of nodes covered in OFBs and extended OFBs for given number of faults. It is clear from the results that both extended faulty blocks and extended orthogonal faulty blocks include more nonfaulty nodes (that are enabled) than faulty blocks and orthogonal faulty blocks, respectively. However, the differences are not significant especially when the number of faults are small.

6. EXTENSIONS

In this section, we provide some ideas for extensions, which include partial adaptive routing, traffic- and adaptivity-bal-anced routing using virtual networks, and removing constraints using virtual channels and networks.

6.1 Partial adaptive routing

The extended X-Y routing is deterministic, that is, there is only one routing path (except when a packet routes around a routing block). In the following, we consider a *restricted zig-zag routing* based on the odd-even turn model.

Routing can be divided into *EW-routing* (from east to west) and *WE-routing* (from west to east). The case for the offset along the y dimension (Δy) being zero can be assigned to either group. WE-routing follows the extended X-Y routing which consists of phase 1 and phase 2 as discussed in the previous section (see Figures 14 (a) and (b)). EW-routing follows the restricted zig-zag routing which consists of a sequence of alternating phase 1 and phase 2 (see Figures 14 (c) and (d)). WE-routing is still deterministic while EW-routing is partially adaptive.

Let Δ_x^1 and Δ_y^1 (Δ_x^2 and Δ_y^2) be the offsets along dimensions x and y respectively in phase 1 (phase 2). The requirement at each phase is the following: In phase 1, Δ_x^1 is monotonically decreasing, and in phase 2, Δ_y^2 is monotonically decreasing and Δ_x^2 remains unchanged. It can be easily shown that as long as the above requirement is met in each phase, the restricted zig-zag routing is livelock-free. The deadlock freedom of the restricted zig-zag routing is obvious, since all turns in both phase 1 and phase 2 are permissible.

6.2 Traffic- and adaptivity-balanced routing using virtual networks

The proposed routing protocol does not make use of resources (channels) evenly. It is obvious that even columns are heavily used in routing in the x dimension. To balance the channel usage, we can use two versions of the routing protocol, one is based on Rules 1 and 2 which heavily uses even columns and the other one is discussed below which heavily uses odd columns. In the second version of the extended odd-even turn model, still the rightmost column segment of a cycle is prevented. However, *the rule of even and odd is exchanged*.

Rule $1'$: *Any packet is* not *allowed to take an EN turn at any node located in an odd column, and it is* not *allowed to take an NW turn at any node located in an even column.*

Rule $2'$: *Any packet is* not *allowed to take an ES turn at any node located in an odd column, and it is* not *allowed to take a SW turn at any node located in an even column.*

Figure 13 shows the permissible EN, NW, ES, and SW turns under Rules $1'$ and $2'$ in phases 1 and 2. In the second version of the extended odd-even turn model, odd columns are used to route the packet in the x dimension (in phase 1). The role of even and odd is also exchanged when route around faulty blocks in phase 2. To support two versions of the extended odd-even turn model, two virtual networks, VN_1 and VN_2, are used. VN_1 is used to enforce Rules 1 and 2 while VN_2 is applied to implement Rules $1'$ and $2'$. Each virtual network consists of a set of virtual channels, assuming that each physical channel may support several virtual channels multiplexed across the physical channel. Figure 15 shows the notions of virtual channels and virtual networks using a 2×2 mesh (Figure 15 (a)). Figure 15 (b) shows a 2×2 mesh with two virtual

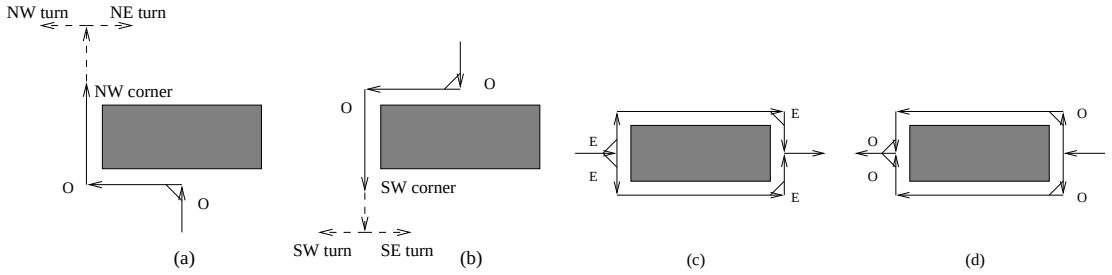

Figure 13: Two cases of phase 1 routing (a) and (b) and two cases of phase 2 routing (c) and (d) using Rule 1′.

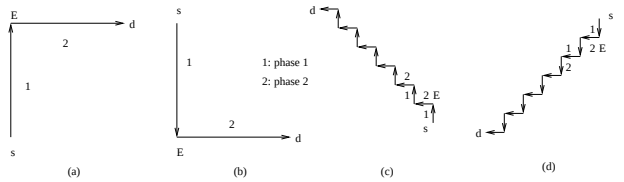

Figure 14: Partial adaptive routing: (a) and (b) extended X-Y routing for WE-routing and (c) and (d) restricted zig-zag routing for EW-routing.

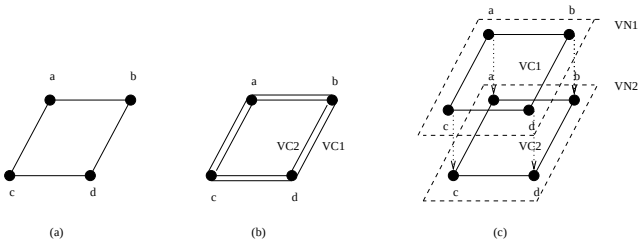

Figure 15: (a) A 2×2 mesh, (b) a 2×2 mesh with two virtual channels VC_1 and VC_2, and (c) a 2×2 mesh with two virtual networks VN_1 and VN_2.

channels, VC_1 and VC_2, but still one network. Figure 15 (c) shows a 2×2 mesh with two virtual networks VN_1 and VN_2. VN_1 (and VN_2) consists of virtual channels VC_1's (VC_2's) only.

A virtual network is selected whenever a packet is injected into the network. Each packet stays in the virtual network until it reaches the destination. It is possible to allow switching from VN_1 to VN_2 (see Figure 15 (c)) during the routing process to increase adaptivity without causing deadlock. We adopt the following rule for a source to select a virtual network: *VN_1 is used if the destination is in an odd column; otherwise, VN_2 is chosen.*

Note that other versions of the extended odd-even turn model can be derived either by *preventing the leftmost column segment of a cycle* or by *exchanging the role of column and row*. Rules by preventing the leftmost column segment of a cycle not only provides a traffic-balanced routing to complement Rules 1 and 2 but also provides an adaptivity-balanced routing when it is used together with Rules 1 and 2. Under these new rules, WE-routing should adopt the extended X-Y routing while EW-routing should follow the restricted zig-zag routing. When two versions are used together, they provide an adaptivity-balanced routing between EW-routing and WE-routing.

6.3 Removing constraints using virtual channels and virtual networks

So far we focus on presenting the basic idea without going into the messy details of boundary situations. These situations include faulty nodes at edges (or adjacent to edges) of the 2-D mesh and destinations are adjacent to a faulty block. Many existing approaches can be applied to handle the former case where virtual channels are used to route around faulty blocks at the edges of the mesh. Here we focus on handling the latter case using two virtual channels or virtual networks.

Note that condition (2) of the assumptions requires that the destination not to be a boundary node of any faulty block. In fact, the proposed algorithm works for all destinations that are boundary nodes of faulty block, except ones at the east side of a faulty block. Refer to Figure 4 (c) where an ES turn is made at an odd boundary line. If the corresponding even boundary line is at the west of the odd boundary line (i.e., closer to the faulty block) and the destination is at this even boundary line, it will force a SW turn at the odd boundary line (which is not permissible) as the last hop. To handle this situation, two virtual channels VC_1 and VC_2 are used at those even boundary lines that are adjacent to faulty blocks. All hops use VC_1's except the last hop which is a SW turn at the odd boundary line. Two virtual networks provide even a simpler solution. This approach not only balances traffic but also increases the scope of applicability, that is, the constraint that the destination is not a boundary node a faulty block can be removed. Rules 1 and 2 are implemented using VN_1 and it will take care of all destinations in odd columns. In this case, no SW or NW turn is needed at the east side of a faulty block. Rules 1′ and 2′ are implemented using VN_2 and it handles all destinations in even columns.

The virtual channel approach can also be applied to the orthogonal faulty block model. Recall the additional constraint on the orthogonal faulty block model: The destination must be outside the container of any orthogonal faulty block. That is, unsafe but enabled nodes (i.e., nodes inside the container but outside the orthogonal faulty block) are used only as sources or intermediate nodes to bypass traffic, but not destinations. Again, we use two virtual channels to remove this restriction. In fact, unsafe but enabled nodes in a container $[min_x : max_x, min_y : max_y]$ form up to four *connected components*. The one contains node (min_x, min_y) is

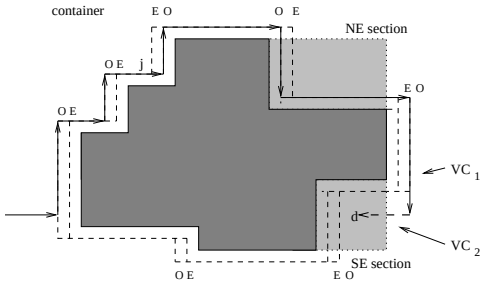

Figure 16: A SW turn at a odd column that switches from VC_1 to VC_2.

called SW section (see Figure 16), the one contains (min_x, max_y) SE section, the one contains (max_x, max_y) NE section, and the one contains (max_x, min_y) NW section. In fact, destinations at SW and NW sections are allowed without causing any problem. The problem occurs when route around the east side of a faulty block through a sequence of ES and SE turns in odd columns. If the destination is inside the SE section, a SW turn in an odd column cannot be avoided (as shown in Figure 16 for a phase 2 routing). Similar situation occurs when route around the east side of the block through a sequence of NS and SN turns in odd columns and the destination is inside the NE section. The solution is again using two virtual channels: VC_1's are used throughout until a SW turn (NW turn) in an odd column is made. In this case, the packet enters the SE section (NE section) of the faulty block. In the remaining steps VC_2's are used until reaching the destination. It is easy to show that the SE or NE section is fault-free. Therefore, the remaining steps can be completed without switching virtual channels.

7. CONCLUSIONS

In this paper, we have proposed a simple and efficient deterministic fault-tolerant and deadlock-free routing in 2-D meshes without virtual channels. This approach is based on the popular X-Y routing and the recently proposed odd-even turn model. The novelty of the approach is the use of two boundary lines at the east and west of a faulty block. This gives just enough freedom to route around a faulty block, and at the same time, to avoid certain turns that may cause deadlock. The proposed approach can be applied to 2-D meshes with any convex type of faulty blocks with simple modification. We have shown the use of localized algorithms to construct rectangular faulty blocks, a special type of convex faulty blocks, and boundary lines. Our future work includes applying the extended even-odd routing to high dimensional meshes. Another possible extension is to apply the method to 2-D torus networks with wraparound connections. In this case, there will be no boundary fault constraint. However, nodes around each dimension form a ring. Virtual channels (or virtual networks) need to be introduced to remove potential cyclic dependency.

8. REFERENCES

[1] R. V. Boppana and S. Chalasani. Fault-tolerant wormhole routing algorithms for mesh networks. *IEEE Transactions on Computers*. 44, (7), July 1995, 848-864.

[2] Y. M. Boura and C. R. Das. Fault-tolerant routing in mesh networks. *Proc. of 1995 International Conference on Parallel Processing*. August 1995, I 106- I 109.

[3] S. Chalasani and R. V. Boppana. Communication in multicomputers with nonconvex faults. *IEEE Transactions on Computers*. 46, (5), May 1997, 616-622.

[4] A. A. Chien and J. H. Kim. Planar-adaptive routing: Low-cost adaptive networks for multiprocessors. *Journal of ACM*. 42, (1), January 1995, 91-123.

[5] G. M. Chiu. The odd-even turn model for adaptive routing. *IEEE Transactions on Parallel and Distributed Systems*. 11, (7), July 2000, 729-737.

[6] W. J. Dally and C. L. Seitz. Deadlock-free message routing in multiprocessor interconnection networks. *IEEE Transactions on Computers*. 36, (5), May 1987, 547-553.

[7] J. Duato. A necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks. *IEEE Transactions on Parallel and Distributed Systems.* 6, (10), 1995, 1,055-1,067.

[8] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. *IEEE/ACM Mobicom99*. 1999, 263-270.

[9] E. Fleury and P. Fraigniaud. A general theory for deadlock avoidance in wormhole-routed networks. *IEEE Transactions on Parallel and Distributed Systems*. 9, (7), July 1998, 626-638.

[10] G. J. Glass and L. M. Ni. Fault-tolerant wormhole routing in meshes without virtual channels. *IEEE Transactions on Parallel and Distributed Systems*. 7, (6), June 1996, 620-636.

[11] G. J. Glass and L. M. Ni. The turn model for adaptive routing. *Journal of ACM*. 40, (5), Sept. 1994, 874-902.

[12] R. Libeskind-Hadas, K. Watkins, and T. Hehre. Fault-tolerant multicast routing in the mesh with no virtual channels. *Proc. of the 2rd International Symposium on High Performance Computer Architecture*. 1995, 180-190.

[13] D. H. Linder and J. C. Harden. An adaptive and fault tolerant wormhole routing strategy for k-ary n-cubes. *IEEE Transactions on Computers*. 40, (1), Jan. 1991, 2-12.

[14] H. Park and D. P. Agrawal. Generic methodologies for deadlock-free routing. *Proc. of the 10th International Parallel Processing Symposium*. April, 1996, 638-643.

[15] Y. J. Suh, B. V. Dao, J. Duato, and S. Yalamanchili. Software based fault-tolerant oblivious routing in pipelined networks. *Proc. of the 1995 International Conference on Parallel Processing*. August 1995, I 101 - I 105.

[16] P. H. Sui and S. D. Wang. An improved algorithm for fault-tolerant wormhole routing in meshes. *IEEE Transactions on Computers*. 46, (9), Sept. 1997, 1040-1042.

[17] D. Wang. Minimal-connected-component (MCC) - a refined fault block model for fault-tolerant minimal routing in mesh. *Proc. of IASTED Int'l Conf. on Parallel and Distributed Computing and Systems*. Nov. 1999, 95-100.

[18] J. Wu. A deterministic fault-tolerant and deadlock-free routing protocol in 2-d meshes without virtual channels. Technical Report, Florida Atlantic University, TR-CSE-00-26, Nov. 2000.

[19] J. Wu. A distributed formation of orthogonal convex polygons in mesh-connected multicomputers. *Proc. of International Parallel and Distributed Processing Symposium (IPDPS)*. 2001, (CD-ROM).

[20] J. Wu. Reliable unicasting in faulty hypercubes using safety levels. *IEEE Transactions on Computers*. 46, (2), Feb. 1997, 241-247.

[21] J. Zhou and F. Lau. Adaptive fault-tolerant wormhole routing in 2d meshes. *Proc. of the 15th International Parallel & Distributed Processing Symposium (IPDPS 2001)*. 2001, 56.