# A New Approach for Fault-Tolerant Broadcasting in Two Dimensional Torus Networks

*N.W. Lo, Bradley S. Carlson* and *D.L. Tao*

The Department of Electrical Engineering
State University of New York at Stony Brook
Stony Brook, NY 11794-2350
Fax: (516)632-8494
Phone: (516)632-8474
Internet: bcarlson@jessica.ee.sunysb.edu

Technical Report: #764

**Abstract**

For a multicomputer system with a large number of processors the probability of having 10 faulty ones is very high in any two-day mission time. Existing fault-tolerant broadcasting algorithms for two dimensional tori/meshes tend to tolerate only a few faults. Unfortunately this level of fault tolerance cannot guarantee the success of executions for time-consuming computational tasks. The problem is that it is almost impossible to come up such a simple broadcasting algorithm to achieve 100% fault tolerance with tens of faults occurred. To achieve such a goal, in this paper, we propose simple broadcasting algorithms to tolerate 98% or more processor failures in a reasonable mission time in a $5^p \times 5^p$ torus network. A stochastic fault model for a multicomputer system is applied to further analyse the fault distribution in a 2-D torus network. According to the analysis a novel one-to-all fault-tolerant broadcasting algorithm for a 2-dimensional torus network to tolerate up to 24 faults is developed. It is shown in the experimental measurements of the new fault-tolerant broadcasting algorithm that in a $125 \times 125$ torus the failure possibility of a one-to-all broadcast is less than 2% in a two-day mission time and corroborated to the analysis of the fault distribution. From our knowledgement we are the first one to directly address the problem of fault tolerance for broadcasting in a interconnection network with a large number of processor failures occurred.

Keywords: fault tolerance, adaptive wormhole routing, circuit-switched broadcast, 2-D torus

# 1 Introduction

Recent *circuit-switched* multicomputer systems such as the Cray T3D [1], Intel Paragon [2], nCUBE-2 [3], and iWarp [4, 5] have become a very good candidate to achieve cost-effective high-performance computing tasks. In order to take advantage of the computation power provided by multiple processors in a multicomputer the message-passing mechanism and the communication overhead from the interconnection network between processors must be addressed. Network topology, routing mechanism and flow control are the key factors to design and implement a fast and reliable interconnection network. With their numerous attractive features *n-dimensional hypercube*, *multidimensional tori* (*wrap-around meshes*) and *multidimensional meshes* become the most popular interconnection networks. A lot of routing algorithms [6, 7, 8, 9] are developed under these network topologies.

For current multicomputer systems *wormhole routing* [10], one of the circuit-switched routing types, is more favorable than the other types (*direct connect*, *virtual cut-through*) because the network latency of wormhole-routed messages is nearly distance-insensitive and the design of router can be simplified such as reducing the buffer size or eliminating the buffer in every processor [11, 12, 13]. However, in [14] the analysis indicates the general transmission time for direct-connect routing [15, 16] and wormhole routing are similar to each other. Direct-connect routing also avoid link contention and deadlock problems associated with wormhole routing. For simplicity, in this paper direct-connect routing is adopted as the routing scheme.

As the number of components (processors, routers and memory blocks) in a multicomputer system increases, the system is more prone to have a component failure between the maintenance timetables. Consequently, for large scale multicomputers system reliability [17] becomes a very important issue. In [18] network fault tolerance has been defined as the maximum number of elements that can fail without inducing a possible disconnection in the network.

A one-to-all broadcasting problem is to deliver a message from a source node (processor) to all other nodes in the network. Most existing multicomputer systems do not support broadcast operation in hardware. Therefore, specific broadcasting algorithms for software implementation of broadcast operation in different network topologies are in demand. While many broadcasting algorithms have been proposed [14, 19, 20, 21], the fault tolerance of broadcast is rarely well addressed [22, 23, 24]. Most researchers devote to fault-tolerant routing mechanisms [25]. Many schemes are developed to estiablish fault tolerance of rout-

ing including but not excluding the design of hardware router [26, 27], the usage of safty vector [28], compressionless routing [29], virtual channel scheme [30, 31, 32]. Although it can be done by using only fault-tolerant routing mechanisms to estiablish the fault tolerance of broadcast operation, but in order to gain good performance of broadcast operation and good utilization of network resources broadcasting oriented fault-tolerant algorithms are the better choice.

For a multicomputer with a large number of processors, the probability of having 10 faulty ones is very high in any two-day mission time. However, existing fault-tolerant broadcasting algorithms for two dimensional tori/meshes tend to tolerate only a few faults. In real environment this is not enough. If we can have a broadcasting algorithm to tolerate a number of faults that will occur in a given mission time, say a few days, then that will be extremely important. The problem is that it is almost impossible to come up such a simple algorithm to achieve 100% fault tolerance with tens of faults occurred. Hence, in this paper we adopt a new approach to achieve such a goal. By first analyzing the fault distribution in a 2-D torus network we use the information to develop simple broadcasting algorithms to tolerate 98% or more faults in a reasonable mission time. To analyse the fault distribution a stochastic fault model [33, 34, 35] for a generic multicomputer system is applied to two dimensional tori. Only static processor failures are considered and global knowledge of fault is assumed. The Poisson process [36] is adopted to describe the stochastic behavior of the processor failures. For a $5^p \times 5^p$ torus network the whole network can be partitioned into multiple $5 \times 5$ *square grids*. We focus our attention on the probability of having at least $l$ faults in one partitioned $5 \times 5$ square grid out of total $j$ processor failures in a 2-D torus. From this advanced information a new fault-tolerant broadcasting algorithm is developed by using divide-and-conquer scheme. The proposed algorithm adopting the same concept shown in [22] improves the network fault tolerance from 3 to 24 node (processor) failures. Simulation experiments are taken to show the effectiveness of the proposed fault-tolerant broadcasting algorithm.

The remainder of this paper is presented as follows. In the next section we formally formulate the broadcasting problem. The analysis of fault distribution is presented in Section 3. Based on the fault analysis a multi-fault tolerant broadcasting algorithm for 2-dimensional torus is developed in Section 4. Section 5 presents the simulation experiments and measurement analysis. Finally, the conclusion is given in Section 6.

# 2 Problem Formulation

Under the assumption that with faulty nodes occurred the interconnection network of a mulitcomputer does not be divided into two or more isolated small networks a one-to-all fault-tolerant broadcasting problem is to deliver a message from a fault-free source node to all other fault-free nodes in the whole network. As the number of failed processors increases, the success of broadcast operation becomes more and more difficult. Totally depending on fault-tolerant routing algorithms to solve the multi-fault broadcasting problem is not an efficient solution. Especially when the scale of the network is very large and more faulty nodes are likely to occur, the effectiveness of using routing algorithms to perform the broadcast operation is a big question mark. However, it is also very difficult to develop a broadcasting algorithm to achieve 100% fault tolerance for a network with a large number of faults.

Therefore, our goal is to develop simple and efficient broadcasting algorithms such that during a reasonable mission time a 99% to 95% fault tolerance can be achieved with a large number of faults occurred in the network. One way to reduce the complexity of the problem is to apply the divide-and-conquer scheme to break down the problem size (the scale of network) into small pieces (subnetworks). For example, a $25 \times 25$ torus network can be partitioned into 25 small meshes with the size of $5 \times 5$ each. Now the broadcasting problem can be divided into two phases. The problem of the first phase is to find a fault-free node in each partitioned subnetwork and broadcast the message from the source node to these chosen nodes successfully. The problem of the second phase is to broadcast the message in each subnetwork from the received node (after the first phase) to all the other fault-free nodes.

The key factors to solve the first phase problem are the combination of proper fault-tolerant routing algorithm, router model,network topology and broadcast mechanism. Depending on the role and function of a network different combinations of these components are composed. Broadcasting algorithms for the first phase should be carefully designed to meet the actual requirement of a network. The complexity of the second phase is that if a developed broadcasting algorithm can tolerate a lot of faults in a subnetwork, then the algorithm can be too complicate to perform efficient broadcast. That is, too many broadcasting steps are required in a broadcast operation. On the other hand if a simple broadcasting algorithm is developed to tolerate only a few faults, the requirement of more than 95% fault tolerance for the whole network can be hard to achieve because there are more faults occurred in a subnetwork than the simple algorithm can handle. To conquer the second phase problem

an analysis of fault distribution is the key to success. A mathematical analysis of fault distribution provides the observation of the limitation of fault tolerant capability in a network and its subnetworks. From the analysis we can develop simple broadcasting algorithms to tolerate only a few faults in a subnetwork while at the same time we still achieve 99% to 95% fault tolerance for the whole network. The mathematical model can also reveal the proper maintenance timetables (the mission time) for different sizes of network.

Because two dimensional torus network is a very popular network topology and existing broadcasting algorithms for 2-D torus can only tolerate a few faults, therefore, in this paper we adopt two dimensional torus as the target network to investigate.

# 3 Fault Distribution in 2-D Tori

In this paper we mainly consider the problem of *static* (permanent) processor failure between the regular maintenance schedules. By labeling the connected processors of the both ends of a faulty link (physical channel) as faulty nodes the faulty link problem can be transformed to the processor-failed problem. To address link fault problem in details more dedicated models or theories should be considered [30].

Assume the occurrences of processor failures are independent and the probability of two failures appearing at the same time is zero. The Poisson process [36] is adopted to simulate the stochastic behavior of the failures as considered in [33, 34, 35]. Therefore, the stochastic fault model is described by the mathematical formula of the probability ($P\{X(t) = j\}$) of having exactly $j$ faulty processors in a multicomputer with total $k$ processors in the interval [0, $t$].

$$P\{X(t) = j\} = \frac{(k \times \lambda \times t)^j}{j!} exp(-k \times \lambda \times t) \tag{1}$$

The random process $X(t)$ denotes the total number of processor failures in a multicomputer system during the time period [0, $t$]. The failure rate of a processor is denoted as $\lambda$. Consequently the probability of having faults between 1 and $j$ is defined as follows.

$$P\{1 \leq X(t) \leq j\} = \sum_{i=1}^{j} \frac{(k \times \lambda \times t)^i}{i!} exp(-k \times \lambda \times t) \tag{2}$$

In this paper 2-dimensional $M \times 5^p \times 5^p$ torus networks are considered where $1 \leq M \leq 24$ and $p \in Z^+$. In [22] a 2-dimensional $5^p \times 5^p$ torus network is partitioned into multiple identical $5 \times 5$ square grids. The number of faulty processors in a $5 \times 5$ square grid of a 2-D torus network is denoted as the random process $Z(t)$. We are interested in the probability

$(P\{Z(t) = l \mid X(t) = j\})$ of having at least $l$ faults in one partitioned $5 \times 5$ square grid out of total $j$ faulty processors.

Let $k$ represents the total number of processors in a multicomputer. The total number of partitioned $5 \times 5$ square grids, $B$, is computed as $\frac{k}{25}$. From the definition of $P\{Z(t) = l \mid X(t) = j\}$ two properties are derived. If $j < l$, then $P\{Z(t) = l \mid X(t) = j\} = 0$. Secondly, $\sum_{i=1}^{j} P\{Z(t) = i \mid X(t) = j\} = P\{X(t) = j\}$. Assume $P\{X(t) = j\}$ is computed from equation 1 in which $j \geq l > 0$ and $j, l \in Z^+$. Define ${}_nC_r = \frac{n!}{(n-r)! \times r!}$ and ${}_nP_r = \frac{n!}{(n-r)!}$. The process to compute the probability $P\{Z(t) = l \mid X(t) = j\}$ is shown as follows.

**Process for computing probability** $P\{Z(t) = l \mid X(t) = j\}()\{$

1. Let $S = \{(m_1, m_2, \ldots, m_l) \mid j = m_1 \times l + m_2 \times (l-1) + \cdots + m_l \times 1$, where $m_1 \neq 0$, $m_1, m_2, \ldots, m_l \in \{0\} \cup Z^+\}$.
2. Set variable $W = 0$.
3. For each possible fault pattern $(m_1, m_2, \ldots, m_l) \in S$ do the following probability computation and accumulate the result value to variable $W$.

$$ {}_BP_{(m_1+\cdots+m_l)} \times (\frac{1}{B})^j \times \prod_{i=1, m_i \neq 0}^{l} \frac{\prod_{n=0}^{m_i-1} (j - \sum_{r=1}^{i-1}(m_r \times (l+1-r)) - n \times (l+1-i))}{m_i!} C_{(l+1-i)} $$

4. Compute $P\{Z(t) = l \mid X(t) = j\} = W \times P\{X(t) = j\}$.

$\}$

In this computation process we first decide what fault patterns are possible by solving the equation $j = m_1 \times l + m_2 \times (l-1) + \cdots + m_l \times 1$ where $m_1 \neq 0$ and all variables are positive integers (including zero). A fault pattern $(m_1, m_2, \ldots, m_l)$ represents that there are $m_i$ square grids containing $(l + 1 - i)$ faults with $i \in \{1, 2, \ldots, l\}$. For each fault pattern the probability of occurrence is calculated and the result value is accumulated to the temporary variable $W$. Since all possible fault patterns are under the assumption that total $j$ faults are occurred in the network with the probability $P\{X(t) = j\}$, the probability $P\{Z(t) = l \mid X(t) = j\}$ is obtained from multiplying $W$ by $P\{X(t) = j\}$. The probability of occurrence for a fault pattern is derived as follows. ${}_BP_{(m_1+\cdots+m_l)}$ denotes the possible combinations of having $(m_1 + \cdots + m_l)$ faulty square grids out of $B$ partitioned square grids. The probability for each faulty processor to be located in a $5 \times 5$ square grid is $B^{-1}$. Because each node failure is independently occurred, therefore, for the occurrence of $j$ faulty nodes the probability that each fault is located in a square grid is computed as $(B^{-1})^j$. The product

Table 1: The Probability of $P_2^Z\{t\}$

| | *the time interval between maintenance dates* | | |
|---|---|---|---|
| *size* | t = 48 hours | t = 78 hours | t = 120 hours |
| 125 × 125 | 98.7% | 52.2% | 1.3% |
| 125 × 75 | 99.8% | 98.8% | 66.8% |
| 75 × 75 | 99.9% | 99.7% | 98.9% |

term computes the possible combinations of assigning $(l+1-i)$ faults into each of $m_i$ square grids from the rest of unlocated faults, $j - \sum_{r=1}^{i-1}(m_r \times (l+1-r))$, where $1 \leq i \leq l$.

**Example 1:** Assume the failure rate of a processor, $\lambda$, is equal to 0.00005. The time interval between two maintenance dates, $t$, is 100 hours. To compute the probability of having at least 3 faults in one partitioned $5 \times 5$ square grid out of total 5 faulty processors in a 1000 node multicomputer system we have $Z(t) = l = 3$, $X(t) = j = 5$ and $B = \frac{1000}{25} = 40$. By applying the computation process the set $S = \{(1,0,2), (1,1,0)\}$ and the probability $P\{Z(t) = 3 \mid X(t) = 5\} = ({}_{40}P_3 \times (\frac{1}{40})^5 \times \frac{{}_5C_3}{1!} \times \frac{{}_2C_1 \times {}_1C_1}{2!} + {}_{40}P_2 \times (\frac{1}{40})^5 \times \frac{{}_5C_3}{1!} \times \frac{{}_2C_2}{1!}) \times \frac{(1000 \times 0.00005 \times 100)^5}{5!} exp(-1000 \times 0.00005 \times 100) = 0.1\%$. □

In order to develop simple and efficient fault-tolerant broadcasting algorithms for 2-D torus the probability $(P\{Z(t) \leq l \mid X(t) = j\})$ of having at most $l$ faulty nodes in a partitioned $5 \times 5$ square grid out of total $j$ node failures in a torus network becomes the focus. From previous discussion we can derive $P\{Z(t) \leq l \mid X(t) = j\} = \sum_{i=1}^{l} P\{Z(t) = i \mid X(t) = j\} \times P\{X(t) = j\}$.

Consider $\lambda = 0.00002$, then the MTTF (*mean-time-to-failures*) of a processor element in a multicomputer is equal to 50,000 hours (5.7 years). Under different maintenance timetables and the scale sizes of multicomputer systems the probabilities of having at most 2 faults in a $5 \times 5$ square grid out of total node failures from 1 to 24 in a 2-D torus, $P_2^Z\{t\} = \sum_{i=1}^{24} P\{Z(t) \leq 2 \mid X(t) = i\}$, are computed and displayed in Table 1.

As shown in Table 1, depending on the scale size of the multicomputer the proper timetable for network maintenance can be decided such that simple fault-tolerant algorithms can be developed and adopted to tolerate up to tens of processor failures between maintenance dates.

# 4 Fault-Tolerant Broadcasting Algorithm for 2-D Tori

According to the discussion in Section 3 a broadcasting algorithm with the capability to tolerate up to 24 node failures is developed in this section. Notice that the proposed algorithm is based on the assumption that no more than 2 faulty processors are located in the same partitioned $5 \times 5$ square grid. Therefore, the fault-tolerant broadcasting algorithm can be simplified and easy to implement. For the sake of simplicity we mainly focus on $5^p \times 5^p$ ($p \in Z^+$) torus networks in this section. The related issues for $M \times 5^p \times 5^p$ torus networks are discussed in Section 4.3. The communication model applied in this paper and previous related work are discussed in Section 4.1 and Section 4.2, respectively. The proposed fault-tolerant broadcasting algorithm is described in Section 4.3.

## 4.1 The Communication Model

In this paper only permanent node failure is considered and all four links (physical channels) of a faulty node are assumed to be inoperable. Under the assumption that only one fault occurs at any given time we can broadcast the address of the faulty node from one of its alive neighbor nodes to all other fault-free nodes in the network by applying our fault-tolerant broadcasting algorithm. Therefore, we can assume the addresses of faulty nodes are known by every non-faulty node.

Let $V_T$ represents the nodes in a 2-D torus $T$ and $E_T$ contains all corresponding link edges between nodes. A $5^p \times 5^p$ 2-dimensional torus network $T = (V_T, E_T)$ is defined as $V_T = \{(i,j) \mid 0 \leq i,j \leq 5^p - 1, p \in Z^+\}$ and $E_T = \{((u_1,v_1),(u_2,v_2)) \mid u_1 = (u_2 \pm 1) \bmod 5^p$ and $v_1 = v_2$ OR $u_1 = u_2$ and $v_1 = (v_2 \pm 1) \bmod 5^p \; \forall (u_1,v_1),(u_2,v_2) \in V_T, p \in Z^+\}$. We assume the node $(0,0)$ is in the left-bottom corner of a torus graph and the other three corner nodes are labeled $(0, 5^p - 1), (5^p - 1, 5^p - 1)$, and $(5^p - 1, 0)$.

*All-port* communication capability and direct-connect routing are assumed to be implemented onto the interconnection network. A node can pass an incoming message from one of its input ports to one of its output ports. Up to four messages can be switched through a node at the same time when no two incoming messages are directed to the same output port. As the direct-connect type of circuit-switched routing is applied, message transmission can only proceed after the physical link path between two processors is established. By adopting direct-connect routing we avoid the contention and deadlock problems associated with wormhole routing.

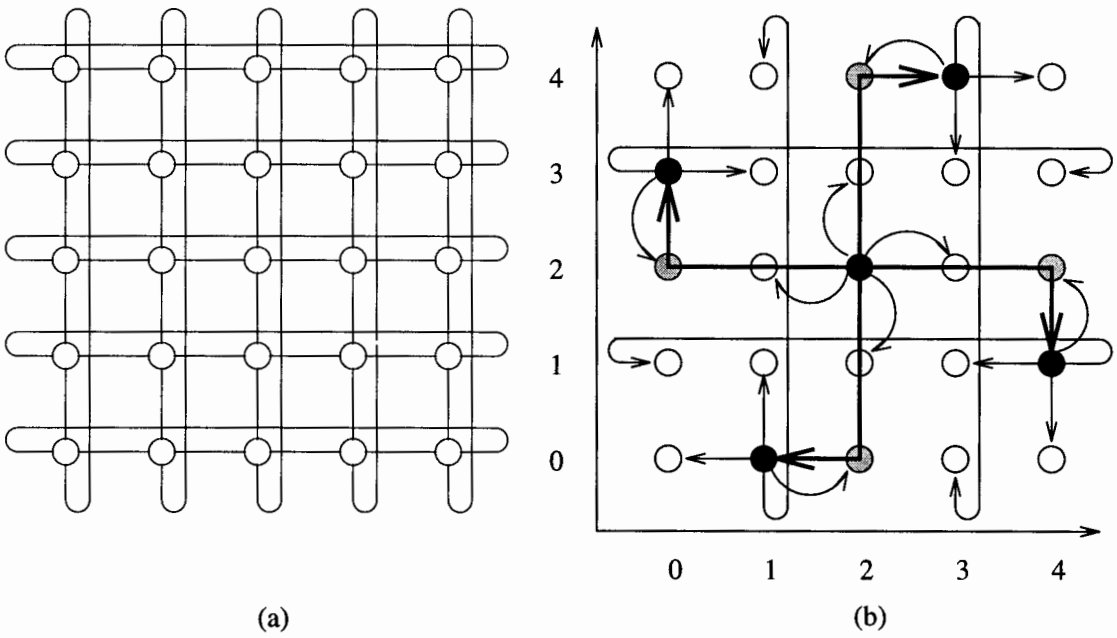


Figure 1: (a) The $5 \times 5$ torus graph, and (b) broadcasting in a $5 \times 5$ torus graph.

## 4.2 Previous Related Work

### 4.2.1 An Optimal Broadcasting Algorithm

An optimal circuit-switched broadcasting algorithm (denoted as the *PSB algorithm*) for 2-dimensional $5^p \times 5^p$ torus network is presented by Peters and Syska in [14]. By embedding the recursive tiling scheme [37, 38] the PSB algorithm can complete a broadcast in $2p$ broadcasting steps. Consider the example of a $5 \times 5$ 2-dimensional torus shown in Fig. 1(a) and assume the node located at the left-bottom corner is labeled $(0, 0)$. Two phases of the PSB algorithm in a $5 \times 5$ 2-dimensional torus are shown in Fig. 1(b). The black node in the center of the graph is assumed to be the source node with label $(2, 2)$. In the first phase the source node broadcasts the message to the four black nodes with labels $(4, 1), (1, 0), (0, 3)$, and $(3, 4)$ via four disjoint paths marked with large arrows. In the second phase each black node $(u, v)$, including the source node $(2, 2)$, sends the message to its four immediate neighbors $(u+1, v), (u-1, v), (u, v+1)$, and $(u, v-1)$. If we unwrap the $5 \times 5$ torus carefully, we can get a mesh-form diagram shown in Fig. 2.

We define a *level*-1 *cross unit* as one node and its four neighbor nodes connected together with its four physical links. A mesh-form diagram of $5 \times 5$ torus composed of 5 cross units (Fig. 2) is defined as a *level*-1 *group unit* (*basic group unit*). A *level-i cross unit* contains five *level*-$(i-1)$ *group units* such that one level-$(i-1)$ group unit is positioned at

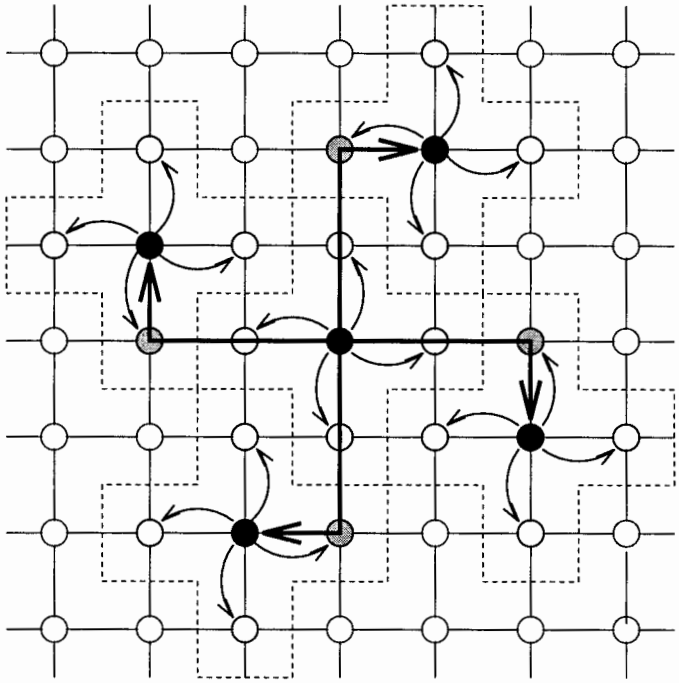

Figure 2: A broadcast process on a $5 \times 5$ torus which is composed of 5 cross units (one basic group unit) drawn as a mesh form.

the center with the four other level-$(i-1)$ group units connected along each edge in which $i \in Z^+$.

**Example 2:** To understand the recursive tiling scheme used in the PSB algorithm a $25 \times 25$ torus composed of five level-2 cross units (subgraphs) is drawn as a mesh form in Fig. 3. Some details of the PSB algorithm are omitted from the four outer level-2 cross units to simplify the diagram. During a broadcast process the center source node in the center subgraph first sends the message to each center node of the other four subgraphs. Each subgraph is composed of 5 basic group units. At the second step every center node of these 5 subgraphs simultaneously sends the message to the center of the other four basic group units in its own subgraph. Before the third broadcasting step each basic group unit in the $25 \times 25$ torus has the message in its center node. After two more steps are applied in each basic group unit (as shown in Fig. 2) the broadcast task is completed. □

### 4.2.2 3-Fault Tolerant Broadcasting Algorithm

In [22], based on the PSB algorithm Lo at el. proposed fault tolerant broadcasting algorithms for 2-dimensional $5^p \times 5^p$ torus network. The proposed algorithm can tolerate at most 3 faults with the cost of spending at most one additional broadcast step compared to the optimal

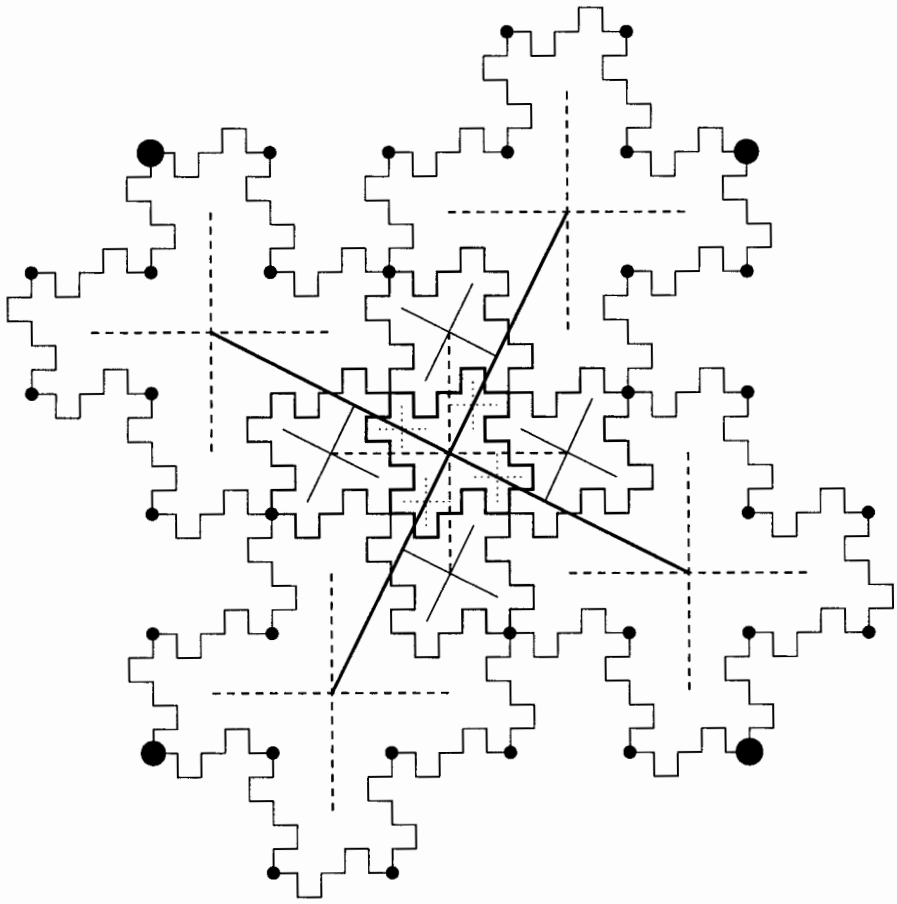

Figure 3: A broadcast process on a $25 \times 25$ torus drawn as a mesh form.

PSB algorithm. Therefore, the proposed 3-fault tolerant broadcasting algorithm is optimal[1] and the complexity of this algorithm is the same as the PSB algorithm, $O(\log_5 N)$, where $N$ denotes the total number of nodes in a torus network.

In a basic group unit the *root node* is the center node of the center cross unit. A center node of a surrounding cross unit is called a *branch node.*

**Definition 1:** An *embedded torus* $G \subset T$ is defined as $G = (V_G, E_G)$ such that

1. The broadcasting center node is labeled $(a, b) = (\frac{5^p-1}{2}, \frac{5^p-1}{2})$.
2. $V_G = \{(i,j) \mid \mid i - a \mid = 5m, \mid j - b \mid = 5n, 0 \leq m, n \leq \lfloor \frac{5^p-1}{10} \rfloor, p \geq 2\}$.
3. $E_G = \{((u_1, v_1), (u_2, v_2)) \mid u_1 = (u_2 \pm 5) \bmod 5^p$ and $v_1 = v_2$ OR $u_1 = u_2$ and $v_1 = (v_2 \pm 5) \bmod 5^p \; \forall (u_1, v_1), (u_2, v_2) \in V_G, p \geq 2\}$. □

**Definition 2:** A node set $V_L \subset V_T$ is defined as the nodes in $V_T$ located on any link edge $l \in E_G$. □

With the PSB algorithm implemented as the default broadcasting mechanism the basic idea of the proposed fault-tolerant algorithm is to identify a proper fault-free node as the pseudo-source node to bypass faulty nodes during a broadcast operation. The concept of embedded torus is introduced to find the proper pseudo-source node such that the nodes in the corresponding embedded torus $G(V_G, E_G)$ and $V_L$ are fault-free. In Fig. 4 an embedded $5 \times 5$ torus $G$ (all black and gray nodes) is constructed from a $5^2 \times 5^2$ torus network $T$ by applying the PSB algorithm on the pseudo-source node (the center node). The *find_pseudo_source_node algorithm* is developed in [22] to identify the pseudo-source node. The message is transmitted from the real source node to the pseudo-source node, and the PSB algorithm is applied to send the message from the pseudo-source node to all of the nodes in $V_G$. After every node in $V_G$ receives the message, depending on the fault positions, either the *Prebranch_Broadcast PSB algorithm* or the PSB algorithm is used to broadcast the message from each node in $V_G$ to the rest of fault-free nodes. The Prebranch_Broadcast PSB algorithm offers three disjoint pre-determined routing paths to bypass a fault on a branch node position of a basic group unit.

## 4.3 24-Fault Tolerant Broadcasting Algorithm

In this section we extend the concept of embedded torus in [22] with the observation of fault distribution derived in Section 3 to develop a 24-fault tolerant broadcasting algorithm.

[1]See Theorem 4 in [39] for proof.

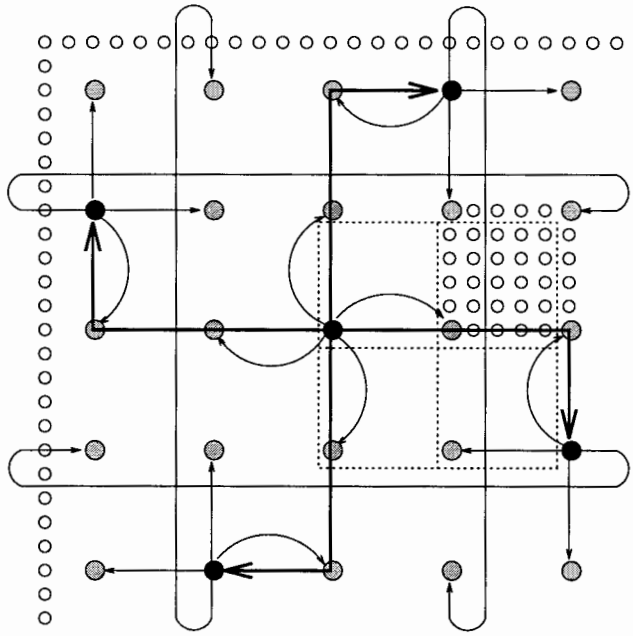

Figure 4: Broadcasting in a $25 \times 25$ torus graph; an embedded $5 \times 5$ torus is observed.

From observation there are exactly 25 unique embeddings $G$ that can be constructed in a 2-D torus network $T$. Therefore, under the assumption that every fault-free node can always communicate with each other when less than 25 faults are occurred in the network we can always find an embedded torus $G$ such that all nodes in $V_G$ are fault-free. The idea of the fault-tolerant algorithm is to identify an embedded torus $G$ in which all nodes in $V_G$ are fault-free and send the message from the original source node to one of the nodes in $V_G$ as the pseudo-source node. Then apply the PSB algorithm to broadcast the message from the pseudo-source node to the other nodes in $V_G$. Since multiple faults are occurred, for the PSB algorithm fault-tolerant routing is applied to replace the dimensional ordering routing used in [14]. Finally every node in $V_G$ as the center node of a $5 \times 5$ square grid broadcasts the message to the rest of fault-free nodes in its corresponding square grid. Notice that we assume no more than 2 faults are occurred in a partitioned $5 \times 5$ square grid. It is shown from the analysis in Table 1 under proper scheduled maintenance our assumption is reasonable. To simplify the broadcast problem synchronous broadcast is assumed in this paper. Consider Fig. 3 as an example, until every center node of the four surrounding level-2 cross units receives the message from the center source node of the center level-2 cross unit no further broadcasting action will start alone with any node already received the message.

Because a $M \times 5^p \times 5^p$ torus graph is symmetric, it can be partitioned into multiple identical $5 \times 5$ square grids as the one shown in Fig. 4. We define a $5 \times 5$ virtual square

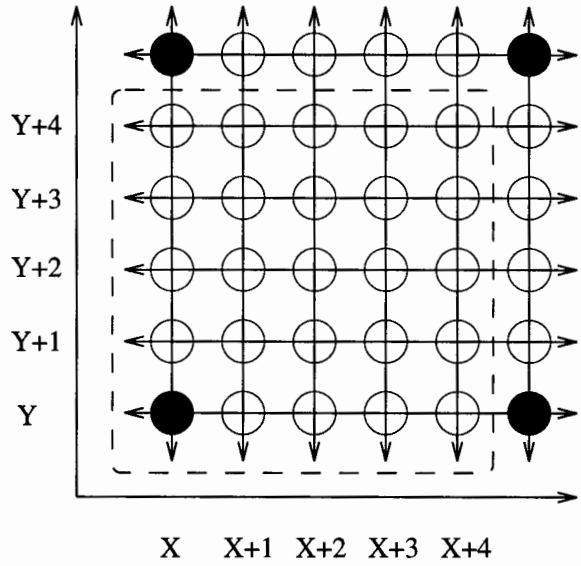


Figure 5: The 4 black nodes are belong to the embedded torus $G$; the mapping range of the $5 \times 5$ mapping square grid is indicated by dashed lines.

grid as *the mapping square grid* shown in Fig. 5. All square grids partitioned in a 2-D torus can be superimposed on the mapping square grid. We denote two link edges passed through the left and bottom sides of the mapping square grid as the left and bottom boundaries of a mapped square grid, respectively. In a partitioned square grid the left and bottom boundary offsets of a node $(u, v)$ are defined as follows.

**Definition 3:** A *left boundary offset* of a node $(u, v)$, $LB(u, v)$, is the distance between the node $(u, v)$ and the left boundary of the $5 \times 5$ square grid which contains the node $(u, v)$.

**Definition 4:** A *bottom boundary offset* of a node $(u, v)$, $BB(u, v)$, is the distance between the node $(u, v)$ and the bottom boundary of the $5 \times 5$ square grid which contains the node $(u, v)$.

Assume the original source node is labeled $(x, y)$. To compute boundary offsets of a node a simple algorithm is derived as follows.

- A left boundary offset of a node $(u, v)$ can be computed as $(u - x) \bmod 5$ if $u - x \geq 0$; otherwise, the value is equal to $5 - (x - u) \bmod 5$.
- A bottom boundary offset of a node $(u, v)$ can be computed as $(v - y) \bmod 5$ if $v - y \geq 0$; otherwise, the value is equal to $5 - (y - v) \bmod 5$.

Let the set $F = \{$ the positions of all faulty nodes $\}$. Define the final $X$ and $Y$-axis offsets between the originator and the selected pseudo-source node as $x_{offset}$ and $y_{offset}$ where $0 \leq x_{offset}, y_{offset} \leq 4$. Now we present the *simple_find algorithm* to identify the

pseudo-source node.

**simple_find algorithm**(){
  if ($\exists LB(u,v) == BB(u,v) == 0$, where $(u,v) \in F$) { /* if $F \cap V_G \neq \emptyset$ */
    For ($x_{offset} = 0; x_{offset} \leq 4; x_{offset} = x_{offset} + 1$) {
      For ($y_{offset} = 0; y_{offset} \leq 4; y_{offset} = y_{offset} + 1$) {
        if ( $\nexists$ ($LB(u,v) == x_{offset}$ AND $BB(u,v) == y_{offset}$) $\forall (u,v) \in F$)
          goto FOUND;
      }
    }
  FOUND:
    set the pseudo-source node $(x', y') = (x + x_{offset}, y + y_{offset})$;
  } /* the end of if-then-else statement */
}

Because multiple faults are considered, the nodes in $V_T$ located on any link edge $l \in E_G$ can be faulty. In order to bypass faults blocked on the routing path when the pseudo-source node broadcasts the message to the other nodes in $V_G$ fault-tolerant routing is required. Depending on the implementation cost, the size of network, network performance requirement and network reliability issues, different fault-tolerant routing algorithms can be adopted [28, 29, 30, 31, 32]. The fault-tolerant routing algorithm used in this paper is discussed in Section 5.

In order to accomplish the broadcast task we apply the divide-and-conquer concept to virtually partition a 2-D torus $T$ into multiple $5 \times 5$ square grids with each node in $V_G$ as the center node of a square grid. Every center node in a square grid is responsible for broadcasting the message to the rest of fault-free nodes in its square grid. As no more than 2 faults are considered in a square grid, if we use the simplest store-and-forward routing to do the local broadcast, in the worst case 7 broadcasting steps are required to accomplish the task. Fig. 6 illustrates the worst situation. To reduce the broadcasting steps in a square grid the charasteristics of circuit-switched routing is applied to develop the $5 \times 5$ *square grid broadcasting algorithm.* Under fault-free environment the proposed algorithm only requires three broadcasting steps to complete the broadcast task in a $5 \times 5$ square grid compared to four broadcasting steps required by the store-and-forward scheme. We define the 16 nodes on the borders of a $5 \times 5$ square grid as the *external square.* The *internal square* of a square grid

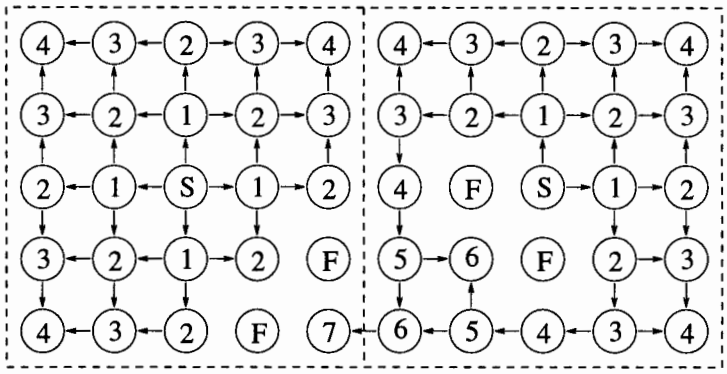


Figure 6: Broadcasting in faulty square grids with store-and-forward routing; the number in a node denotes in which broadcasting step the node receives the message and symbols S and F indicate the source nodes and the faulty nodes, respectively.

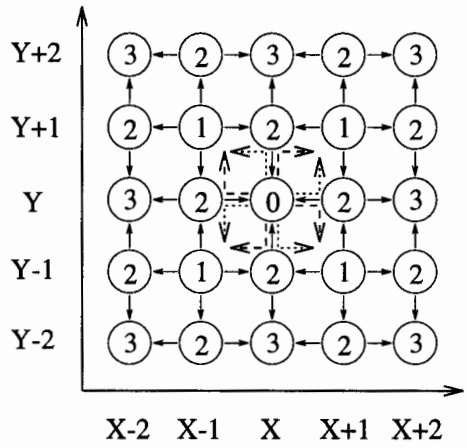


Figure 7: Broadcasting in a fault-free square grid with the center node as the source node; two sets of broadcast paths for the first broadcasting step are indicated by dashed and dotted arrow lines and the rest broadcasting steps are denoted by solid arrow lines.

is composed of the 8 nodes surrounding the center node. The $5 \times 5$ square grid broadcasting algorithm is presented in the following.

**$5 \times 5$ Square Grid Broadcasting Algorithm**(){

1. Send the message from the center source node $(x, y)$ to the four corner nodes of the internal square through routing paths shown in Fig. 7.
2. Each corner node of the internal square broadcasts the message to its four neighbors.
3. After receiving the message each node broadcasts it to the neighbors except the directions from which the message is received.
4. Repeat Step 3 until all nodes in the $5 \times 5$ square grid receive the message.

}

Notice that in Fig. 7 the number in a node represents in which broadcasting step the node receives the message. If a fault-free node receives duplicate messages, it just discards them.

By observing the broadcasting sequence of the $5\times5$ square grid broadcasting algorithm in Fig. 7 it is shown that if two possible faults both are positioned in the external square, no modification for the square grid broadcasting algorithm is necessary to tolerate these faults. Because there are at least two different links for a node to receive a message from other nodes in the same square grid. The only exception is the two adjacent nodes of a corner node in the external square of a square grid become faulty as shown in Fig. 6. Therefore, the fault-free corner node has to receive the message from nodes in adjacent square grids. The last step of the *2-fault tolerant grid broadcasting algorithm* proposed in the following handles the problem by broadcasting the received message from fault-free corner nodes of the external square in every square grid to the adjacent nodes of the two neighbor square grids. Since we assume all fault-free nodes are connected together through fault-free links, a fault-free corner node of the external square in a square grid always has at least one fault-free link to reach a fault-free adjacent node in the same square grid or a neighbor one. However, particular routing paths to bypass faults and accomplish the broadcast in three broadcasting steps have to be established provided that there are some faulty nodes in the internal square.

To simplify the following discussion all fault-tolerant routing paths in Figs. 8 – 13 are drawn with dashed and dotted arrow lines to indicate the first and second fault-tolerant broadcasting steps. All regular routings follow the proposed $5 \times 5$ square grid broadcasting algorithm. General fault patterns are proposed to illustrate possible fault distributions and the corresponding routing solutions for broadcasting. Any possible fault distribution can be mapped into one of the general fault patterns by rotating the $5 \times 5$ virtual square grid.

In the case of only one fault occurred in the internal square two general fault patterns are presented in Fig. 8. In Fig. 8(a) the southwestern corner node of the external square receives the message from the center source node $S$ during the first broadcasting step instead of the faulty default node $F$ in the internal square by passing the message through one of the two dashed arrow paths. The other fault pattern is described in Fig. 8(b) with the faulty node $F$ adjacent to the south of the source node $S$. In this case the source node has to send the message to the specific fault-free nodes shown in Fig. 8(b) during the first and second broadcasting steps. Under the assumption that only one fault is in the internal square we can learn that the same fault patterns can be applied while another node in the external

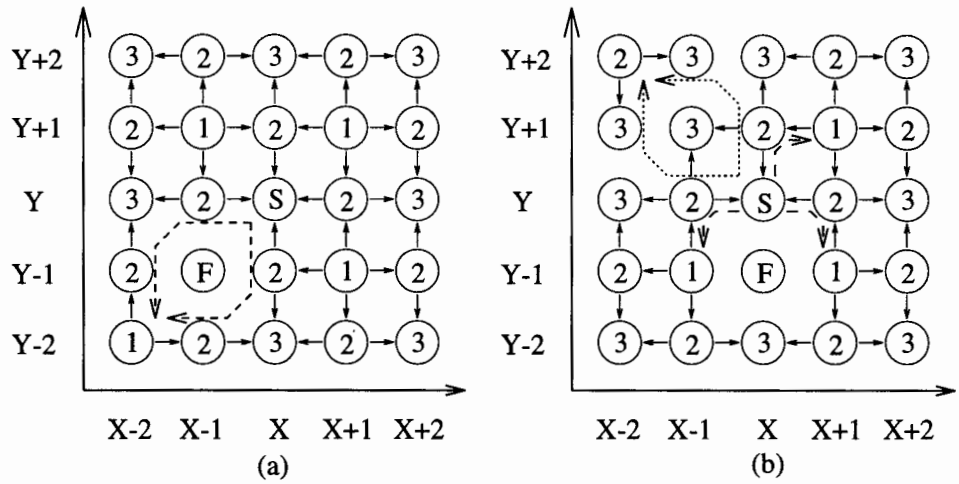


Figure 8: Broadcasting with one fault occurred in the internal square of a square grid; two general fault patterns are displayed in (a) and (b) with the first and second broadcasting steps indicated by the dashed and dotted arrow lines, respectively.

square becomes faulty. Fig. 9(a) and (b) provide new fault-tolerant routing paths to handle the exception cases of the two general fault patterns in Fig. 8(a) and (b), respectively. For two faults located in the internal square eight general fault patterns are derived. According to the distance between two faults in the internal square the eight general fault patterns are partitioned into four categories displayed in Figs. 10 – 13. By sending the message from the source node $S$ to proper fault-free node positions such as the corners of the internal or external squares and the center of each edge of the external square only three broadcasting steps are required to complete the broadcast task in a square grid with two faults in the internal square. The proposed 2-fault tolerant grid broadcasting algorithm is addressed as follows.

**2-Fault Tolerant Grid Broadcasting Algorithm**(){

1. If no faults are located in the internal square, then apply the $5 \times 5$ *square grid broadcasting algorithm* and goto Step 4.
2. If one fault is located in the internal square, apply the broadcasting sequence of the matched fault pattern in Fig. 8 or 9, then goto Step 4.
3. If two faults are located in the internal square, apply the broadcasting sequence of the matched fault pattern in Fig. 10, 11, 12 or 13, then goto Step 4.
4. Each fault-free corner node of external square broadcasts the received message to the adjacent nodes of the two neighbor square grids .

}

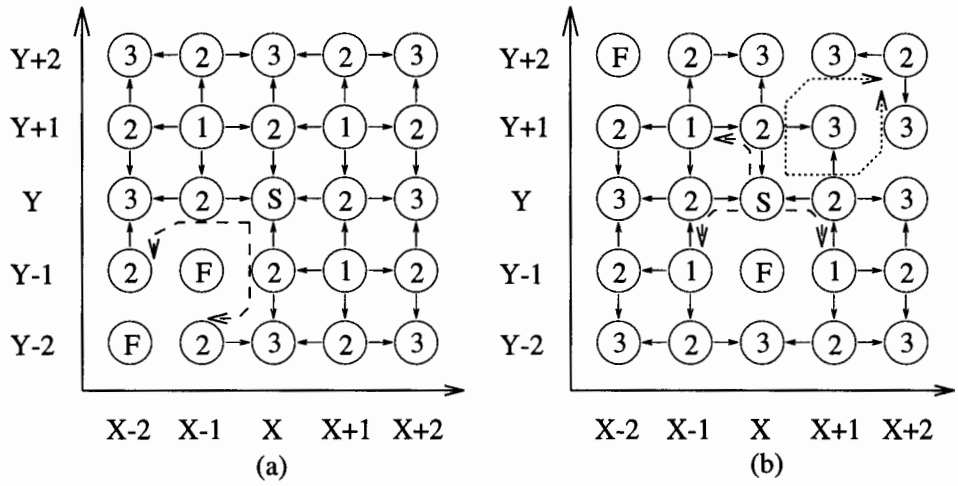


Figure 9: Two special fault patterns in broadcasting with one fault occurred in the internal square and the other occurred in the external square; (a) the special case of Fig. 8(a), and (b) the special case of Fig. 8(b).

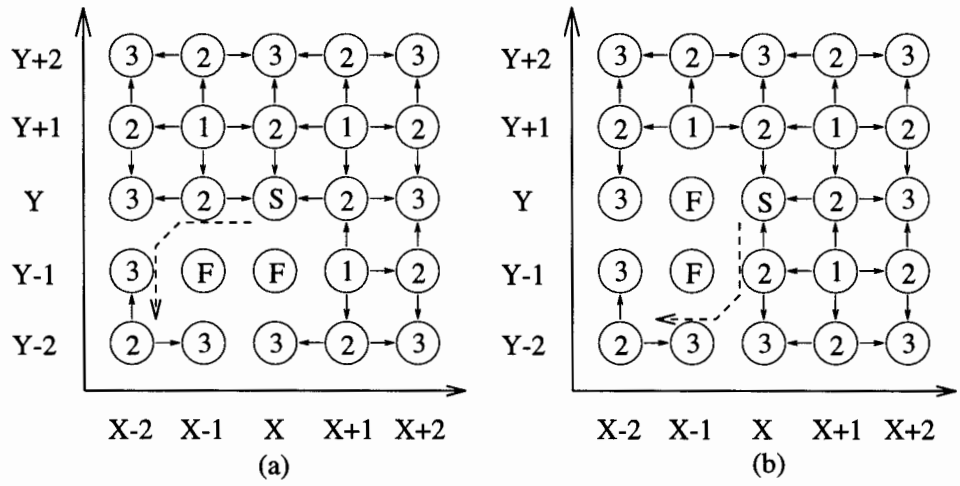


Figure 10: Broadcasting with two faults occurred in the internal square of a square grid; two general fault patterns are displayed in (a) and (b) with two faults adjacent to each other.

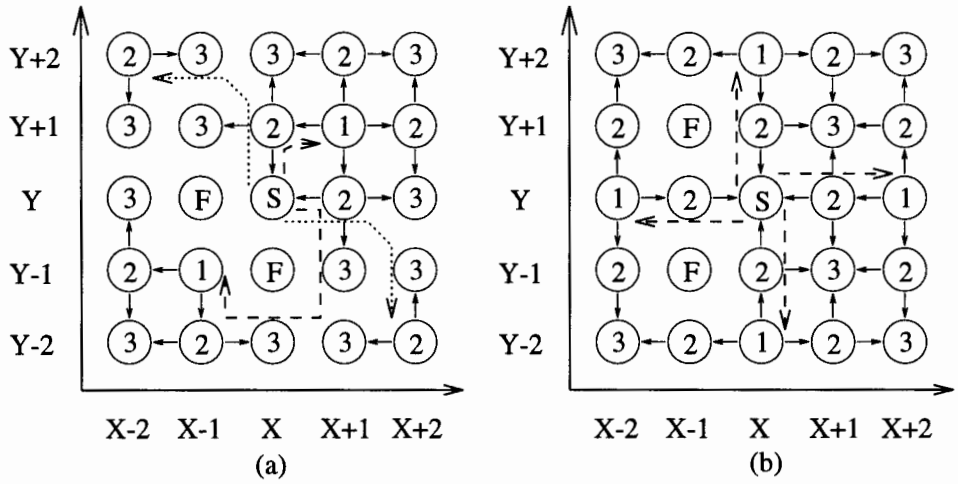


Figure 11: Broadcasting with two faults occurred in the internal square of a square grid; two general fault patterns are displayed in (a) and (b) with one fault-free node located between two faulty nodes in the internal square.

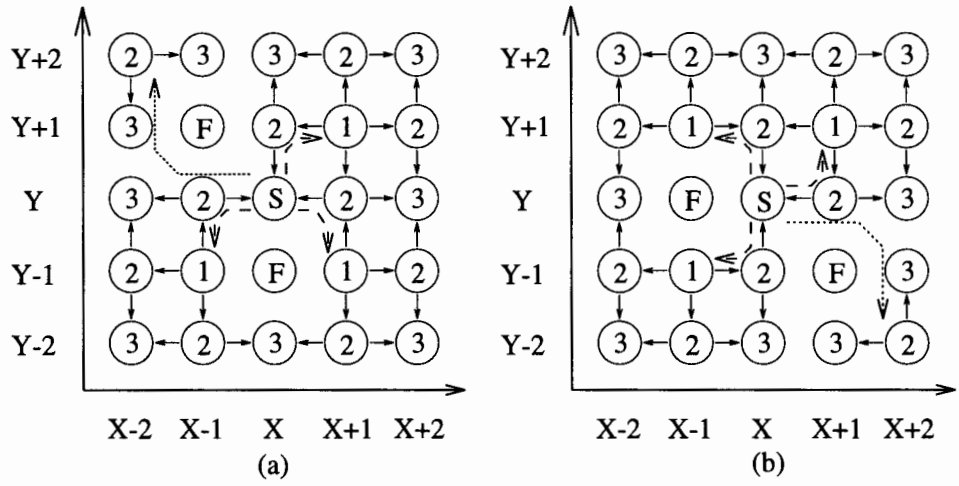


Figure 12: Broadcasting with two faults occurred in the internal square of a square grid; two general fault patterns are displayed in (a) and (b) with two fault-free nodes located between two faulty nodes in the internal square.

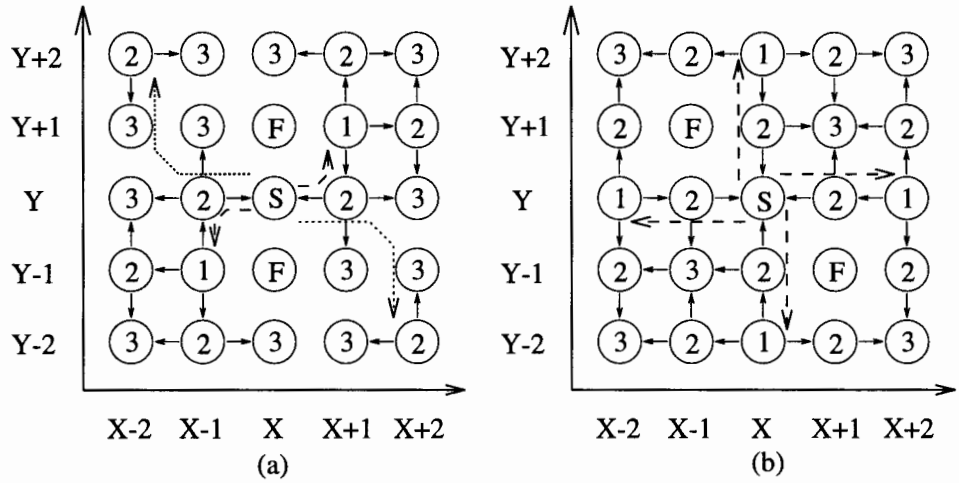


Figure 13: Broadcasting with two faults occurred in the internal square of a square grid; two general fault patterns are displayed in (a) and (b) with three fault-free nodes located between two faulty nodes in the internal square.

**Theorem 1** *In the presence of* $r$ $(0 \leq r \leq 2)$ *faulty nodes four steps are required by the* 2-Fault Tolerant Grid Broadcasting Algorithm *to perform a one-to-all broadcast in a* $5 \times 5$ *virtual square grid of a circuit-switched* $M \times 5^p \times 5^p$ *torus network.*

*Proof*: From previous discussion it is shown that by applying the 2-fault tolerant grid broadcasting algorithm only three broadcasting steps are required for a broadcast operation to tolerate up to two faulty nodes in a square grid. However, to overcome a similar fault distribution shown in Fig. 6 one more broadcasting step is necessary to insure the success of broadcasting in every square grid. Therefore, four broadcasting steps are required for the proposed algorithm to accomplish a broadcast task. □

Finally the 24-fault tolerant broadcasting algorithm for a 2-D torus $T$ is described as follows.

**24-Fault Tolerant Broadcasting Algorithm**(){

1. Construct the embedded torus graph $G$ from the original source node $(x, y)$.
2. Apply the *simple_find algorithm.*
3. Apply the *PSB algorithm* to broadcast the message to each node in $V_G$.
4. For every node received the message apply the *2-fault tolerant grid broadcasting algorithm* and stop.

}

We can easily modify the proposed algorithm to enlarge its adoptivity range from $5^p \times 5^p$ tori to $M \times 5^p \times 5^p$ tori where $1 \leq M \leq 24$. By partitioning the $M \times 5^p \times 5^p$ network

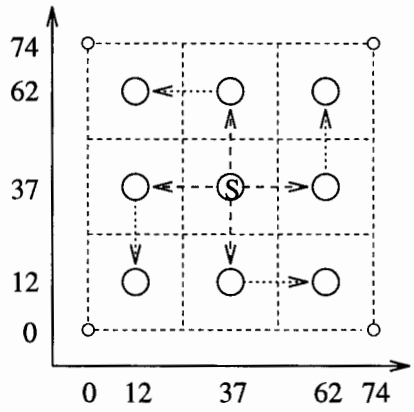


Figure 14: The first (dashed arrow lines) and second (dotted arrow lines) broadcasting steps in a $75 \times 75$ torus network with the pseudo-source node $S$ in the center position.

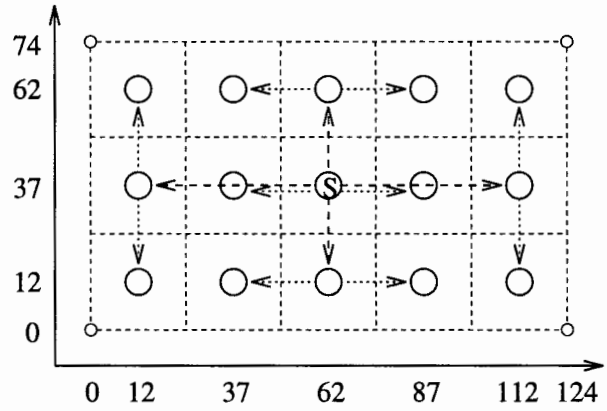


Figure 15: The first (dashed arrow lines) and second (dotted arrow lines) broadcasting steps in a $125 \times 75$ torus network with the pseudo-source node $S$ in the center position.

into $M$ $5^p \times 5^p$ subnetworks the selected pseudo-source node first broadcasts the message to the center nodes of other $M-1$ subnetworks. After every subnetwork received the message, apply the PSB algorithm for each $5^p \times 5^p$ subnetwork to broadcast the message to each node in $V_G$. Then the Step 4 of the proposed fault-tolerant algorithm is executed as the last broadcasting phase. Many broadcasting algorithms can be developed and adopted for the broadcast task between subnetworks. In Figs. 14 and 15 a $75 \times 75$ ($9 \times 5^2 \times 5^2$) torus and a $125 \times 75$ ($15 \times 5^2 \times 5^2$) torus are partitioned into 9 and 15 $5^2 \times 5^2$ subtori, respectively. By applying simple broadcasting algorithms in both cases two broadcasting steps are required for both networks to accomplish the broadcast from the center source node to every $25 \times 25$ subtorus.

Table 2: The Successful Broadcast Ratio

| | *the time interval between maintenance dates* | | |
|---|---|---|---|
| *size* | t = 48 hours | t = 78 hours | t = 120 hours |
| 125 × 125 | 98.0% | 44.2% | 0.7% |
| 125 × 75 | 99.8% | 97.9% | 59.2% |
| 75 × 75 | 99.9% | 99.7% | 98.3% |

# 5 Experiment Analysis

To examine our fault analysis in Section 3 and the proposed fault-tolerant broadcasting algorithm in Section 4.3 simulation experiments are taken for validation and evaluation. The number of faulty nodes and the position of each faulty node are generated indirectly from two random number generators [40][2] with different seed numbers. An intuitive routing algorithm is applied to bypass faults. The fault-tolerant routing algorithm implemented for simulation experiments is dimensional oriented. When there is no faulty node located on the routing path, the proposed algorithm functions as the typical dimensional routing algorithm. Once a fault is encountered during the routing phase, the algorithm tries to route the message around the fault and goes back to the same dimension and direction it intends to route before meeting the fault. Because backtracking routing is not implemented and the algorithm is made as simple as possible, routing failures caused by particular fault patterns or the contention of physical channels are possible.

In this paper three scale sizes of 2-D tori, 125 × 125, 125 × 75 and 75 × 75 are simulated under three different maintenance timetables – 48, 78 and 120 hours. 50,000 hours are assumed as the MTTF value of a processor element in a network. Each experiment is executed for 50,000 iteration times. The experimental measurements of the successful broadcast ratio, the ratio of more than 24 faults occurred, the ratio of routing failure, the ratio of more than 2 faults occurred in a 5 × 5 grid, the average broadcasting steps and the average faults occurred are displayed in Tables 2 – 7, respectively.

By comparing Table 1 with Table 2 it is shown that the experimental measurements match the analysis of fault distribution in Section 3. The results shown in Table 2 indicate the failure ratio of broadcasting in a faulty 2-D torus is able to be held down under 2% with proper scheduled maintenance timetable and simple fault-tolerant routing/broadcasting

[2]We apply the ran1() procedure in page 280.

Table 3: The Ratio of More Than 24 Faults Occurred

| | *the time interval between maintenance dates* | | |
|---|---|---|---|
| *size* | t = 48 hours | t = 78 hours | t = 120 hours |
| 125 × 125 | 1.79% | 55.56% | 99.29% |
| 125 × 75 | 0.00% | 1.47% | 40.11% |
| 75 × 75 | 0.00% | 0.00% | 0.66% |

Table 4: The Ratio of Routing Failure

| | *the time interval between maintenance dates* | | |
|---|---|---|---|
| *size* | t = 48 hours | t = 78 hours | t = 120 hours |
| 125 × 125 | 0.07% | 0.09% | 0.00% |
| 125 × 75 | 0.08% | 0.18% | 0.17% |
| 75 × 75 | 0.02% | 0.04% | 0.10% |

Table 5: The Ratio of More Than 2 Faults Occurred in a 5 × 5 Grid

| | *the time interval between maintenance dates* | | |
|---|---|---|---|
| *size* | t = 48 hours | t = 78 hours | t = 120 hours |
| 125 × 125 | 0.14% | 0.15% | 0.01% |
| 125 × 75 | 0.12% | 0.45% | 0.52% |
| 75 × 75 | 0.08% | 0.26% | 0.94% |

Table 6: The Average Broadcasting Steps

| | *the time interval between maintenance dates* | | |
|---|---|---|---|
| *size* | t = 48 hours | t = 78 hours | t = 120 hours |
| 125 × 125 | 9.36 | 9.68 | 9.74 |
| 125 × 75 | 8.90 | 9.27 | 9.57 |
| 75 × 75 | 8.59 | 8.84 | 9.15 |

Table 7: The Average Faulty Nodes Occurred

| | *the time interval between maintenance dates* | | |
|---|---|---|---|
| *size* | t = 48 hours | t = 78 hours | t = 120 hours |
| 125 × 125 | 15.98 | 25.38 | 38.47 |
| 125 × 75 | 9.97 | 15.61 | 23.49 |
| 75 × 75 | 6.39 | 9.78 | 14.48 |

algorithms implemented. Therefore, the analysis of the fault distribution for 2-D tori is valid and the proposed 24-fault tolerant broadcasting algorithm is simple and efficient enough to improve the network fault tolerance.

The consequential broadcast failures during simulation experiments are categorized into three different types listed in Tables 3, 4 and 5. To estimate the proposed algorithm conservatively any simulating iteration generates more than 24 faults is viewed as a broadcast failure. Table 3 displays the failure ratios of more than 24 faults occurred in all experiments. The percentage of routing failures in all experiments are illustrated in Table 4. Because the proposed 2-fault tolerant grid broadcasting algorithm can only handle at most 2 faults ocurred in a $5 \times 5$ square grid, after virtually partitioning the whole experiment network into multiple square grids in each simulating iteration the broadcast operation is failed if one of the square grids contains more than 2 faults. The corresponding failure ratios in all experiments are displayed in Table 5. To those who interested in 100% fault tolerant broadcasting, the information from these three tables shows that it is possible to have 100% fault tolerance in a normal mission time by introducing more complicate routing/broadcasting algorithms and/or spending more time steps to perform a broadcast operation.

Under fault-free environment only 6 broadcasting steps are required for each of the three experimented tori. Table 6 shows that under faulty condition 9 or 10 broadcasting steps are required to complete a one-to-all broadcast task in these 2-D torus networks. The first reason is because at the most of times one additional step is required to send the message from the originator to the selected pseudo-source node. The proposed 2-fault tolerant grid broadcasting algorithm[3] also requires two more steps to broadcast the message in a $5 \times 5$ square grid compared to the optimal PSB algorithm. The third, since synchronization technique is adopted in the implementation of broadcasting algorithm, it is possible for a broadcast task to require some extra broadcasting steps if some faults cause not enough outgoing links to use for broadcast from a source node during one broadcasting step. From Table 7 we confirm that the larger size of the network and the longer period between two maintenance time-points the more faults occur in an interconnection network. Hence, more complicate and powerful fault-tolerant routing/broadcasting algorithms are required to maintain the networks running under more serious faulty situation.

[3]Refer to Theorem 1

# 6 Conclusion

In this paper a new approach to tolerate multiple faults for the broadcast operation in two dimensional torus networks is introduced. From our knowledgement we are the first one to directly address the heavy multiple-fault problem for fault-tolerant broadcasting in interconnection networks. In order to tolerate a large number of faults in a 2-D torus network a stochastic fault model is adopted to analyse the fault distribution. From the analysis of fault distribution a new fault-tolerant broadcasting algorithm is proposed to improve the network fault tolerance up to 24 node failures. Simulation experiments are proceeded to evaluate the correctness of the analysis and the effectiveness of the proposed algorithm. It is shown in the experimental results that the broadcast failure ratio is less than 2% for all three different sizes of experimented networks in a two-day mission time. The match between the analysis of the fault distribution and the experimental measurements corroborate the validation of the proposed process for computing the probability $P\{Z(t) = l \mid X(t) = j\}$.

Because we only consider $M \times 5^p \times 5^p$ tori with static node failures, further investigation for other types of networks and different shapes of $n$-dimensional tori will be interesting topics. Other research directions include the model analysis of channel (link) failure, the corresponding algorithms developed to tolerate multiple link faults and the network fault tolerance of dynamic faults.

# References


[1] W. Oed, "The CRAY Research Massively Parallel Processor System, CRAY T3D," Cray Research, Munich, 1993.

[2] *Paragon XP/S Product Overview,* Intel Corporation, 1991.

[3] *nCUBE 6400 Processor Manual,* nCUBE Company, 1990.

[4] S. Borkar, et al., "iWarp: An Integrated Solution to High-speed Parallel Computing," *Proc. Supercomputing '88,* IEEE, pp. 330-339, 1988.

[5] S. Borkar, et al., "Supporting Systolic and Memory Communication in iWarp," Tech. Report TR CMU-CS-90-197, School of Computer Science, Carnegie-Mellon Univ., 1990.

[6] K. Wendy Tang and S. A. Padubidri, "Diagonal and Toroidal Mesh Networks," *IEEE Tran. on Computers,* vol. 43, no. 7, pp. 815-826, 1994.

[7] A.G. Greenberg and J. Goodman, "Sharp Approximate Models of Deflection Routing in Mesh Networks," *IEEE Tran. on Communications,* vol. 41, no. 1, pp. 210-223, 1993.

[8] E. Chow, H. Madan and J. Peterson, "A Real-Time Adaptive Message Routing Network for the Hypercube Computer," *Proc. 8th Real Time Syst. Symp.,* IEEE, pp. 88-96, 1987.

[9] M.-S. Chen and K.G. Shin, "Adaptive Fault-Tolerant Routing in Hypercube Multicomputers," *IEEE Tran. on Computers,* vol. 39, no. 12, pp. 1406-1416, 1990.

[10] W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Tran. on Computers,* vol. 36, no. 5, pp. 547-553, 1987.

[11] W.J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Tran. on Par. and Dis. Sys.,* vol. 4, no. 4, pp. 466-475, 1993.

[12] R.I. Greenberg and H.-C. Oh, "Universal Wormhole Routing," *IEEE Tran. on Par. and Dis. Sys.,* vol. 8, no. 3, pp. 254-262, 1997.

[13] J. Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks," *IEEE Tran. on Par. and Dis. Sys.,* vol. 4, no. 12, pp. 1320-1331, 1993.

[14] Joseph G. Peters and Michel Syska, "Circuit-Switched Broadcasting in Torus Networks," *IEEE Tran. on Par. and Dis. Sys.,* vol. 7, no. 3, pp. 246-255, 1996.

[15] S. H. Bokhari, "A Network Flow Model for Load Balancing in Circuit-Switched Multicomputers," *ICASE Report* 90-38, May 1990.

[16] S. H. Bokhari, "Communication Overheads on the Intel iPSC-2 Hypercube," *ICASE Interim Report* 10, May 1990.

[17] D.K. Pradhan, *Fault Tolerant Computing,* Prentice Hall International, Englewood Cliffs, New Jersey, 1986.

[18] D. Pradhan, "Dynamically Restructurable Fault-Tolerant Processor Network Architectures," *IEEE Tran. on Computers,* vol. 34, no. 5, pp. 434-447, May 1985.

[19] Ju-Young L. Park and Hyeong-Ah Choi, "Circuit-Switched Broadcasting in Torus and Mesh Networks," *IEEE Tran. on Par. and Dis. Sys.,* vol. 7, no. 2, pp. 184-190, 1996.

[20] Yih-jia Tsai and Philip K. McKinley, "A Broadcast Algorithm for All-Port Wormhole-Routed Torus Networks," *IEEE Tran. on Par. and Dis. Sys.,* vol. 7, no. 8, pp. 876-885, 1996.

[21] Y.-C. Tseng, "A Dilated-Diagonal-Based Scheme for Broadcast in a Wormhole-Routed 2D Torus," *IEEE Tran. on Computers,* vol. 46, no. 8, pp. 947-952, 1997.

[22] N.W. Lo, B.S. Carlson and D.L. Tao, "Fault-Tolerant Broadcasting Algorithms in Two Dimensional Circuit-Switched Torus Networks," *Proc. Intl. Conf. on Par. and Dis. Comp. Sys.,* pp. 40-45, Oct. 1997.

[23] N.W. Lo, B.S. Carlson and D.L. Tao, "Fault Tolerant Algorithms for Broadcasting on the Star Graph Network," *IEEE Tran. on Computers,* vol. 46, no. 12, pp. 1357-1362, 1997.

[24] Ju-Young L. Park, Sang-Kyu Lee and Hyeong-Ah Choi, "Fault-Tolerant Broadcasting in Circuit-Switched Mesh," *Proc. SIAM Conf. on Parallel Processing for Scientific Computing,* pp. 22-24, Mar. 1993.

[25] J. Duato, S. Yalmanchili and L. Ni, *Interconnection Networks - An Engineering Approach,* IEEE Computer Society Press, 1997.

[26] K. Bolding, M. Fulgham and L. Snyder, "The Case for Chaotic Adaptive Routing," *IEEE Tran. on Computers,* vol. 46, no. 12, pp. 1281-1292, 1997.

[27] W.J. Dally, L. Dennison, D. Harris, K. Kan and T. Xanthopoulus, "The Reliable Router: A Reliable and High-Performance Communication Substrate for Parallel Computers," *Proc. Parallel Computer Routing and Comm. Workshop,* pp. 241-255, May, 1994.

[28] J. Wu, "Adaptive Fault-Tolerant Routing in Cube-Based Multicomputers Using Safty Vectors," *IEEE Tran. on Par. and Dis. Sys.,* vol. 9, no. 4, pp. 321-334, 1998.

[29] J.H. Kim, Z. Liu and A.A. Chien, "Compressionless Routing: A Framework for Adaptive and Fault-Tolerant Routing," *IEEE Tran. on Par. and Dis. Sys.,* vol. 8, no. 3, pp. 229-244, 1997.

[30] J. Duato, "A Theory of Fault-Tolerant Routing in Wormhole Networks," *IEEE Tran. on Par. and Dis. Sys.,* vol. 8, no. 8, pp. 790-802, 1997.

[31] W.J. Dally and H. Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels," *IEEE Tran. on Par. and Dis. Sys.,* vol. 4, no. 4, pp. 466-475, 1993.

[32] Y.M. Boura and C.R. Das, "Fault-tolerant Routing in Mesh Networks," *Proc. Intl. Conf. on Parallel Processing,* vol. I, pp. 106-109, 1995.

[33] D.L. Tao and K. Kantawala, "Evaluating Reliability Improvements of Fault Tolerant Array Processors Using Algorithm-Based Fault Tolerance," *IEEE Tran. on Computers,* vol. 46, no. 6, pp. 725-730, 1997.

[34] I. Koren and D.K. Pradhan, "Modeling the Effect of Redundancy on Yield and Performance of VLSI Systems," *IEEE Tran. on Computers,* vol. C-36, no. 3, pp. 344-355, 1987.

[35] Y.X. Wang and Jose A.B. Fortes, "Estimates of MTTF and Optimal Number of Spares of Fault-Tolerant Processor Arrays," *Proc. of FTCS-20,* pp. 184-191, June, 1990.

[36] Athanasios Papoulis, *Probability, Random Variables, and Stochastic Processes, Third Edition,* McGraw-Hill Inc., 1991.

[37] B. Grünbaum and G.C. Shephard, *Tilings and Patterns,* W.H. Freeman, 1987.

[38] M. Senechal, "Tiling the torus," *Discr. and Comp. Geometry,* vol. 3, pp. 55-72, 1988.

[39] N.W. Lo, B.S. Carlson and D.L. Tao, "Fault-Tolerant Broadcasting Algorithms in Two Dimensional Circuit-Switched Torus Networks," Technical Report #745, CEAS, State Univ. of New York at Stony Brook.

[40] W.H. Press, S.A. Teukolsky, W.T. Vetterling and B.P. Flannery, *Numerical Recipes in C, Second Edition,* Cambridge University Press, 1992.