

Fault-Tolerant Broadcasting in Toroidal Networks

Bader Almohammad¹ and Bella Bose²

¹ Department of Mathematics and Computer Science,
Kuwait University, P.O. Box 5969 Safat 13060, Kuwait

² Department of Computer Science,
Oregon State University, Corvallis, OR 97331 USA

Abstract. A non-redundant fault-tolerant broadcasting algorithm in a faulty k -ary n -cube is designed. The algorithm can adapt up to $(2n - 2)$ node failures. Compared to the optimal algorithm in a fault-free k -ary n -cube, the proposed algorithm requires at most 3 extra communication steps in the case of cut through packet routing and $(n + 1)$ extra steps in the case of store-and-forward routing. Conditions under which this algorithm works correctly in a general toroidal network are given.

1 Introduction

The low-dimensional toroidal networks are more wire-efficient communication networks than high-dimensional ones under the assumptions of constant wire bisection and constant number of nodes [5]. For this reason, low-dimensional toroidal interconnection network has become significant from the practical point of view. Examples of systems that use toroidal interconnection networks are: Tera systems [2], Cray T3D [14], and Cray T3E [18].

Broadcasting (single-node one-to-all) in a multicomputer is one of the important communication primitives [9]. In the past many fault-tolerant broadcasting algorithms have been published for hypercube interconnection networks [1, 12, 15, 16, 17]. Not much work has been done in this area for toroidal networks. In [4], a fault-tolerant broadcasting algorithm for k -ary n -cube that can adapt up to $(n - 1)$ failures is given. This paper presents a fault-tolerant broadcasting algorithm for toroidal networks that can adapt up to $(2n - 2)$ failures. The algorithm is non-redundant, requires a global fault information, and is close to optimal in terms of communication time.

2 Preliminaries

This section reviews the notation and definitions used in the rest of this paper.

Mixed Radix Notation: In a mixed radix notation, any integer I , $0 \leq I < K = k_{(n-1)}k_{(n-2)} \dots k_0$, is represented as an n -dimensional vector $X = x_{(n-1)} \dots x_0$ over $K = k_{(n-1)}k_{(n-2)} \dots k_0$, where $x_i \in \{0, 1, 2, \dots, k_i - 1\}$. $I(X)$, the *Integer Value* of X , is defined as: $I(X) = x_0 + x_1k_0 + x_2k_0k_1 + \dots + x_{(n-1)}k_0k_1 \dots k_{(n-1)} =$

$(\sum_{i=1}^{n-1} (x_i \prod_{j=0}^{i-1} k_j)) + x_0$. For example, if 432 is a vector over 543; then $I(432) = 4(12) + 3(3) + 2 = 59$.

Lee Weight: Let X be a vector in the mixed radix notation over K . The *Lee Weight*, W_L , of X is defined as: $W_L = \sum_{i=0}^{n-1} \min(x_i, k_i - x_i)$. For example, if 342 is a vector defined over 765; then $W_L(342) = 3 + 2 + 2 = 7$.

Lee Distance: Let X and Y be vectors in the mixed radix notation over K . The *Lee Distance*, D_L , is defined as: $D_L(X, Y) = \sum_{i=0}^{n-1} \min(x_i - y_i \pmod{k_i}, y_i - x_i \pmod{k_i}) = W_L(X - Y)$. For example, let 131 and 554 be vectors over 765; then $D_L(131, 554) = 7$.

Torus: Let T be an n -dimensional torus denoted as $T_{k_{(n-1)}k_{(n-2)}\dots k_0}$, with each dimension of size $k_i, i = 0, 1, \dots, n-1$. T has $N = \prod_{i=0}^{n-1} k_i$ nodes. Each node of T is labeled with a mixed radix n -dimensional vector over $K = k_{(n-1)}k_{(n-2)}\dots k_0$ called its address. For any node $X \in T$, the node number of $X = I(X)$. For any two nodes X and $Y \in T$, there is an edge between X and Y iff $D_L(X, Y) = 1$. A k -ary n -cube graph, Q_k^n , is a torus in which all the dimensions have the same radix.

3 Fault-Free Broadcasting Algorithm

Broadcast Using Cut-Through (S, M)

S : Source node,

M : Broadcasted message,

An uncovered-node is a node that has not received M yet, and

A covered-node is a node that has received M .

(1) **Mark S as a covered-node.**

(2) **For $i = 1$ to n**

(2.1) **For $j = 1$ to $\lceil \lg k \rceil$**

(2.1.1) **Each covered-node sends M to an uncovered-node at a distance of $\lceil \frac{k}{2^j} \rceil$ along i^{th} dimension.**

(2.1.2) **The nodes which received M in Step (2.1.1) are marked as covered-nodes.**

Fig. 1. A fault-free broadcasting algorithm for Q_k^n using cut-through packet routing.

MECA is one of the first routing and broadcasting algorithms for Q_k^n 's [8]. It uses *wormhole packet routing* technique [6, 13], and to avoid deadlocks, it uses virtual channels [7]. Later, Bose et. al. introduced the Basic Broadcasting

Broadcast Using Store and Forward (S, M)

S : Source node,

M : Broadcasted message,

A covered-node is a node that has already received M ,

An uncovered-node is a node that has not received M yet, and

- (1) **Mark S as an old-covered-node.**
 - (2) **For $i = 1$ to n**
 - (2.1) **Mark all old-covered-nodes to be new-covered-nodes.**
 - (2.2) **For $j = 1$ to $\lceil \frac{k}{2} \rceil$**
 - (2.2.1) **Each new-covered-node sends M to an adjacent uncovered-node along i^{th} dimension.**
 - (2.2.2) **The nodes which received M in Step (2.2.1) are marked as new-covered-nodes.**
 - (2.2.3) **Each new-covered-node adjacent to two covered-nodes (old or new) along i^{th} dimension is marked as an old-covered-node.**
-

Fig. 2. A fault-free broadcasting algorithm for Q_k^n using store and forward packet routing.

Algorithm which is similar to the e -cube algorithm used in hypercube [3]. The algorithm needs n phases to broadcast in a Q_k^n . In the 0^{th} -phase, the source node performs a ring broadcasting along the 0^{th} -dimension. In the i^{th} -phase, each node that has received the broadcasted message from a previous phase performs a ring broadcasting along the i^{th} -dimension. In the case of store-and-forward using single port I/O model, this algorithm is proved to be optimal for k even; for k odd, the algorithm takes n more extra steps than the known lower bound [3].

Figures 1 and 2 describe fault-free broadcasting algorithms for a Q_k^n when *cut through* [10] or *store-and-forward* [11] routing strategy is used, respectively. The basic idea of both algorithms is the same as that of the e -cube algorithm described above. However, broadcasting in a ring differs depending on the packet routing strategy used. When cut through strategy is used, the message path length becomes less significant. In such a case, the communication complexity of sending a message of size M words along a path of length P becomes:

$$t_s + M t_w + P t_h,$$

where t_s is startup time, t_w is per word time, and t_h is per hop time. On the other hand, the same message sent over the same path using store-and-forward routing has a communication complexity of:

$$t_s + P(M t_w + t_h).$$

Since t_h is small compared to $(t_s + Mt_w)$, broadcasting in a ring R of k nodes using cut through routing can be performed in $\lceil \log k \rceil$ phases as follows. In the 0^{th} -phase, the source node, S , sends the message, M , to a node at a distance of $\lceil \frac{k}{2} \rceil$. In the i^{th} -phase, all nodes which have received M in a previous phase send M to distinct nodes at a distance of $\lceil \frac{k}{2^i} \rceil$. The communication complexity of this algorithm is:

$$\lceil \log k \rceil (t_s + Mt_w) + \sum_{i=1}^{\log k} \lceil \frac{k}{2^i} \rceil = \lceil \log k \rceil (t_s + Mt_w) + t_h (k - 1).$$

In a fault-free Q_k^n , broadcasting using cut through routing can be achieved by performing n ring broadcastings. Figure 1 describes such an algorithm which has a communication complexity of:

$$n \lceil \log k \rceil (t_s + Mt_w) + nt_h (k - 1).$$

On the other hand, broadcasting in a ring using store-and-forward routing strategy has a communication complexity of:

$$\lceil \frac{k}{2} \rceil (t_s + t_w M + t_h).$$

Hence, the broadcasting algorithm for Q_k^n described in Figure 2 has a communication complexity of:

$$n \lceil \frac{k}{2} \rceil (t_s + t_w M + t_h).$$

4 Fault-Tolerant Broadcasting Algorithm

In [4], the authors have proposed a non-redundant fault-tolerant broadcasting algorithm that can adapt up to $(n - 1)$ node failures and requires global fault information [4]. In this paper, this algorithm is extended to adapt up to $(2n - 2)$ node failures. The algorithm is given in Figure 3. Its basic idea is as follows. Let the number of faulty nodes be $t \leq 2n - 2$ and the address of these nodes be:

$$\begin{aligned} f_1 &= (a_{n-1,1} \ a_{n-2,1} \ a_{n-3,1} \ \dots \ a_{0,1}) \\ f_2 &= (a_{n-1,2} \ a_{n-2,2} \ a_{n-3,2} \ \dots \ a_{0,2}) \\ &\vdots \\ f_t &= (a_{n-1,t} \ a_{n-2,t} \ a_{n-3,t} \ \dots \ a_{0,t}) \end{aligned}$$

If $k > (2n - 2)$, then at least one of the digits, say b_i , in $Z_k = \{0, 1, \dots, (k - 1)\}$ will not appear in the i^{th} -digits, $i = 0, 1, \dots, (n - 1)$, of the faulty node addresses. Then, $(** \dots b_i \dots **)$ is a fault-free $(k - 1)$ -ary n -cube.

Broadcasting in a Faulty Q_k^n (S, M)

Assumptions:

- 1) Total number of faults $< 2n - 1$.
- 2) $k > 2n - 2$ and $k > 3$.

Definitions:

S : is source node,

M : is message to be broadcasted,

Q : is Q_k^n to broadcast M in,

C_i 's : are sub-cubes of Q along a certain dimension X each of which is a Q_k^{n-1} ,

C : is one of the C_i 's that is a fault-free, and

R_N : is a ring along the X dimension to which the node N belongs and $N \in C$.

- (1) S sends M to a node in C , say S' .
 - (2) S' broadcasts M in C using one of the algorithms given in Figures 1 or 2.
 - (3) Each node $A \in C$ starts a ring broadcasting along the X dimension iff R_A is fault-free.
 - (4) Each faulty ring along the X dimension gets M from a fault-free adjacent ring (one communication step).
-

Fig. 3. A broadcasting algorithm for a faulty Q_k^n .

Example 1. In Q_5^3 , let the faulty nodes be:

$$\begin{aligned} f_1 &= (3 \ 2 \ 1) \\ f_2 &= (1 \ 3 \ 2) \\ f_3 &= (0 \ 4 \ 3) \\ f_4 &= (2 \ 0 \ 4) \end{aligned}$$

Then, the sub-cube $(4 \ * \ *)$ is fault-free because the digit "4" does not appear in the second digit addresses of the faulty nodes. Similarly, the sub-cubes $(* \ 1 \ *)$ and $(* \ * \ 0)$ are also fault-free.

Let $S = (s_{n-1}s_{n-2}s_{n-3} \dots s_0)$ be the source node. For the time being, assume S is in a fault-free Q_k^{n-1} , say $(b_{n-1} \ * \ * \ \dots \ *)$. This means that the faulty nodes are along the $(n-1)^{th}$ -dimensional rings. The number of faulty rings will be $\leq t$, where t is the number of faulty nodes. This is because there may be more than one faulty node in the same ring. In the first phase, S broadcasts the message in the fault-free Q_k^{n-1} using the basic broadcasting algorithm. In the second phase, each node A in the fault-free sub-cube Q_k^{n-1} will broadcast the message along the $(n-1)^{th}$ dimensional ring, provided this ring is fault-free. In the last phase, the fault-free nodes in the faulty rings receive the message from a unique adjacent fault-free ring nodes. It will be shown shortly that there exists a unique fault-free ring adjacent to a given faulty ring.

If the source node S is not in any of the fault-free k -ary $(n-1)$ -cubes, then S can send the message to a closest node in a fault-free k -ary $(n-1)$ -cube. After this, the above steps can be repeated.

The rest of this section is divided into four subsections. In Subsection 4.1, an algorithm to find a fault-free Q_k^{n-1} (and so a node in this Q_k^{n-1}) closest to the source node S is given. Subsection 4.2 describes an algorithm to find a unique fault-free ring for each of the faulty rings. The communication complexity of the algorithm is derived in Subsection 4.3. Subsection 4.4 discusses conditions under which the algorithm can be used in a general toroidal network.

4.1 Closest Fault-Free Q_k^{n-1} From the Source

The *Lee* distance between two rings along the same dimension is defined in the following.

Definition 1. Let R_A and R_B be two rings along the same dimension X in a toroidal space. R_A and R_B can be addressed as $(a_{n-1}a_{n-2}\dots a_{x+1}*a_{x-1}\dots a_0)$ and $(b_{n-1}b_{n-2}\dots b_{x+1}*b_{x-1}\dots b_0)$, respectively where $*$ is a don't care character. The Lee Distance between R_A and R_B , $D_L(R_A, R_B)$, is defined as:

$$D_L(R_A, R_B) = \sum_{i=0 \text{ and } i \neq x}^{n-2} \min(|a_i - b_i|, |k_i - a_i + b_i|),$$

where k_i is the size of the i^{th} dimension.

For example if $R_A = (*223)$, $R_B = (*364)$, and $K = (4578)$, $D_L(R_A, R_B) = 1 + 3 + 1 = 5$.

Suppose $(*\dots*b_i*\dots*)$ is one of the closest fault-free Q_k^{n-1} from the source node $S = (s_{n-1}, s_{n-2}, \dots, s_0)$. Then the node in this sub-cube closest to S is $S' = (s_{n-1}, s_{n-2}, \dots, s_{i+1}, b_i, s_{i-1}, \dots, s_0)$ and $D_L(S, S') = D_L(s_i, b_i)$. If the shortest path from S to S' is fault-free, then this path can be used in Step(1) of the algorithm given in Figure 4 to send the message. However, if some node in this path is faulty, then S can send the message to a node, T , in an adjacent fault-free ring, and then T can send the message along its ring to a node in the fault-free Q_k^{n-1} (Note that the faulty nodes are along the i^{th} -dimensional rings; further, there are $(2n-2)$ adjacent rings to a faulty ring and so at least one of the adjacent rings is fault-free). The following lemma is useful to find the shortest distance fault-free sub-cube, Q_k^{n-1} , from S .

Lemma 2. Suppose $S = (s_{n-1}, s_{n-2}, \dots, s_0)$ is the source node and the faulty nodes are $f_i = (a_{n-1,i}, a_{n-2,i}, \dots, a_{0,i})$, $i = 1, 2, \dots, t$, where $t \leq (2n-2)$. Then, one of the sub-cubes $(*\dots*s_p + j*\dots*)$, where $j = 0, \pm 1, \pm 2, \dots, \pm(n-1)$, is fault-free.

Proof. The t faulty nodes can have a maximum of t distinct digits in the p^{th} address digits. However, the set $\{s_p + j : j = 0, \pm 1, \pm 2, \dots, \pm(n-1)\}$ contains $(2n-1)$ digits and so at least one of the digits, say $(s_p + j')$, is distinct from

Finding Closest Fault-Free Sub-Cube to S

S : is source node in a Q_k^n ,

Output is (i, j) indicating that the j^{th} sub-cube along the i^{th} dimension is one of the closest sub-cubes to S .

- (1) Let faulty nodes be denoted by f_i , $0 \leq i \leq (2n - 3)$.
 - (2) Define a 2-dimensional matrix $MAT[n, k]$, and initialize it to be 0's.
 - (3) For $i = 0$ to $(n - 1)$
 - (3.1) For $j = 0$ to $(k - 1)$
 - (3.1.1) $MAT[i, i^{th} - \text{digit of } f_j] = 1$.
 - (4) Found = False, $i = 0$.
 - (5) While not Found
 - (5.1) If $MAT[i, i^{th} - \text{digit of } S] = 0$
 - (5.1.1) Found = True, Return($i, i^{th} - \text{digit of } S$).
 - (5.2) Else $i = i + 1$.
 - (6) Found = False, $i = -1$.
 - (7) While not Found
 - (7.1) $i = i + 1, j = 0$.
 - (7.2) While (not Found) and $(j < n)$
 - (7.2.1) If $MAT[j, j^{th} - \text{digit of } S + i] = 0$
 - (7.2.1.1) Found = True, Return($j, j^{th} - \text{digit of } S + j$).
 - (7.2.2) Else If $MAT[j, j^{th} - \text{digit of } S - i] = 0$;
 - (7.2.2.1) Found = True, Return($j, j^{th} - \text{digit of } S - j$).
 - (7.2.2.2) Else $j = j + 1$;
-

Fig. 4. Finding one of the closest sub-cubes to S .

the p^{th} digits of the faulty node address digits. Thus, $(* \dots * s_p + j' * \dots *)$ is a fault-free Q_k^{n-1} . \square

To find a closest fault-free Q_k^{n-1} from S , we proceed in the following order. First, check whether any of the sub-cubes $(* \dots * s_p * \dots *)$, $p = 0, 1, \dots, (n-1)$, is fault-free. If so, the source node is in a fault-free sub-cube and stop this process. Otherwise, check any one of the sub-cubes $(* \dots * s_p + j * \dots *)$, where $j = \pm 1$, and $p = 0, 1, 2, \dots, (n-1)$ is fault-free. Continue this process for $j = \pm 2, j = \pm 3, \dots, j = \pm(n-1)$. At any time a fault-free sub-cube is found the search process is stopped. Lemma 1 guarantees to find a fault-free sub-cube, say $(* \dots * s_{p'} + j' * \dots *)$, where $0 \leq p' \leq (n-1)$ and $j = 0, \pm 1, \pm 2, \dots, \pm(n-1)$. This fault-free sub-cube must be one of the closests to S and the distance from S to a node in $(* \dots * s_{p'} + j' * \dots *)$ closest to S is j' . Suppose there is a fault-free sub-cube $(* \dots * s_{p'} + j'' * \dots *)$ which is closer to S than $(* \dots * s_{p'} + j' * \dots *)$. In that case $|j''| < |j'|$. Then in the above search process, the sub-cube $(* \dots * s_{p'} + j'' * \dots *)$ would have been declared a fault-free before searching for values $j = j'$ (i.e. the search process for $j = +j'$ and $j = -j'$ would not have occurred). Thus, this search process finds one of the fault-free k -ary $(n-1)$ -cubes closest to S .

The distance between the source node S and S' which is a node in a fault-free sub-cube can be at most $(n - 1)$ if S sends the message along the same ring; or this distance can be at most n if S first sends the message to its adjacent fault-free ring and then that node sends the message along its ring to the node in the fault-free Q_k^{n-1} .

The algorithm given in Figure 4 formally describes how to find a closest fault-free sub-cube from S . This algorithm has a local computation complexity of $O(n^2)$. First, the algorithm constructs a 2D bit-map matrix $M[(n-1), (k-1)]$ in which all the faulty sub-cubes are marked (i.e. if $M[i, j] = 1$ then the j^{th} -sub-cube along the i^{th} -dimension is faulty; otherwise it is fault-free). This is given in Step 3 of the algorithm. Then, it tries to find a fault-free sub-cube along some dimension to which S belongs (Step 5). If there is none, then it tries to find a fault-free sub-cube along some dimension at a minimum distance, $D_M = 1, 2, \dots, (n - 1)$, from S (Step 7). From the arguments given in the previous paragraphs, it is easy to verify that $D_M \leq (n - 1)$.

If there are two or more fault-free sub-cubes at a distance of D_M from S , the algorithm can be enhanced by choosing the one to which S can send the message, M , directly. Furthermore, it can be modified to try to find a fault-free sub-cube that is connected to S through a fault-free path. (This fault-free sub-cube may not be closest to S). In this case only one routing operation needs to be done through a longer path rather than two routing operations. In cut through routing, the message start up time is more significant than the path length and hence, this approach might be favored. The given algorithm, even with the enhancements, has a local computation complexity of $O(n^2)$. In fact, it is possible to find a fault-free sub-cube, not necessarily the closest, in $O(n)$ local steps. Instead of searching all the dimensions, only one dimension is searched. In this case also, the fault-free sub-cube will be at a distance of $(n - 1)$ or less from S .

4.2 Finding A Unique Fault-Free Adjacent Ring to A Faulty Ring

If the source node S is part of the fault-free $(n - 1)$ dimensional sub-cube, C , then Step(1) of the algorithm given in Figure 4 is not needed. However, if $S \notin C$, then S is in a ring along the X dimension, say R_A . If R_A is fault-free, Step (1) can be done in one communication step. But if R_A is faulty, then R_A must have an adjacent fault-free ring along the dimension X , say R_B . Hence, S sends M to its correspondent node in R_B and from there to a node in C . The existence of a fault-free ring R_B adjacent to R_A is proved in the following lemma.

Lemma 3. *Suppose the source S is in a faulty ring R along the X dimension. If the total number of faults in Q_k^n is $< (2n - 2)$, and $k \geq 3$, then R_A has an adjacent fault-free ring R_B along the X dimension.*

Proof. The number of faults is $\leq (2n - 2)$. Since R_A is faulty, at most $(2n - 3)$ other faulty rings can exist. But R_A has $(2n - 2)$ distinct adjacent rings because $k \geq 3$. Thus, at least one of them is fault-free. \square

Finding An Adjacent Fault-Free Ring (R, X)

X : A dimension,

R : A ring along X to which a fault-free adjacent ring is needed to be found.

- (1) Let FR be the set of faulty rings along a given dimension X .
 - (2) Let the rings adjacent to R be denoted by $R_i, 0 \leq i \leq (2n - 2)$.
 - (3) Found = False, $i = 0$.
 - (4) While (not Found)
 - (4.1) If $R_i \notin FR$
 - (4.1.1) Found = True, Return(i).
 - (4.2) Else $i = i + 1$.
-

Fig. 5. Finding an adjacent fault-free ring along a given dimension.

Assigning Each Faulty Ring to A Unique Adjacent Fault-Free Ring

- (1) Let FR be the set of faulty rings along the X dimension.
 - (2) For each faulty ring f_i do
 - (2.1) Find the set of adjacent rings to f_i , let it be A_i .
 - (2.2) $A_i = A_i - FR$.
 - (2.3) Choose any element in A_i , let it be R_a . Assign R_a to f_i .
 - (2.4) $FR = FR \cup \{R_a\}$.
-

Fig. 6. Assigning each faulty ring along the X dimension an adjacent fault-free ring

In order to perform Step(4) of the algorithm we need to show that each faulty ring along the X dimension $R_N, N \in C$, can be uniquely assigned a fault-free adjacent ring along the same dimension. By uniquely, we mean that every faulty ring will be assigned to a distinct fault-free ring so that all the faulty rings are covered in a single communication step.

Lemma 4. *Let the faulty rings along the X dimension be denoted as $f_i, 0 \leq i < 2n - 2$. Furthermore, let A_i be the set of rings along the X dimension that are adjacent to f_i . If $f_j \in A_i$ and $k > 3$, then $A_i \cap A_j = \phi$.*

Proof. If $f_j \in A_i$ then $D_L(f_i, f_j) = 1$. Assume that $f_j \in A_i$ and $A_i \cap A_j \neq \phi$. This implies, there exists a ring R along the X dimension such that $R \in A_i$ and $R \in A_j$. Then, $D_L(R, f_i) = 1$ and $D_L(R, f_j) = 1$. This means either $[D_L(f_i, f_j) = 1$ and $k = 3]$ or $[D_L(f_i, f_j) = 2$ and $k > 3]$. The first case gives a contradiction because it is assumed that $k > 3$. The second case gives a contradiction because $D_L(f_i, f_j) = 1$. \square

Theorem 5. *In $Q_k^n, k > 3$, if the number of faulty rings is $\leq (2n - 2)$ in some dimension, then for each faulty ring there exists a unique adjacent fault-free ring.*

Proof. The proof is by construction, meaning that it shows how to assign a unique fault-free ring to a given faulty ring. Let the faulty rings be f_1, f_2, \dots, f_t where $t \leq (2n - 2)$. Let A_i be the adjacent rings of f_i , $i = 1, 2, \dots, t$. Note that $|A_i| = (2n - 2)$, for $i = 1, 2, \dots, t$. For each faulty ring f_i , we assign a unique fault-free ring as follows. For $i = 1$, assign a fault-free ring R_1 in A_1 to f_1 ; this is possible because f_1 has $(2n - 2)$ adjacent rings (i.e. $|A_1| = 2n - 2$) and at most $(2n - 3)$ of them can be faulty. Then remove R_1 from other A_i 's, $i = 2, 3, \dots, t$, provided R_1 is in A_i . At the j^{th} -step, assign a non-assigned fault-free ring in A_j to f_j . This is always possible because of the reasons given below. Let R_p be the fault-free ring assigned to the faulty ring f_p , $p = 1, 2, \dots, (j-1)$. If $D_L(f_p, f_j) = 1$, then $f_p \in A_j$ and $R_p \notin A_j$ since $D_L(R_p, f_j) = 2$. On the other hand, if $D_L(f_p, f_j) = 2$ then $f_p \notin A_j$ and R_p might be in A_j . In all other cases, both f_p and R_p are not in A_j . Thus, while trying to assign a fault-free ring to f_j , there should be at least $(2n - 2) - (j - 1)$ rings adjacent to f_j available; of these the number of fault-free rings should be $(2n - 2) - (j - 1) - (t - j) \geq (2n - 2) - (j - 1) - (2n - 2 - j) \geq 1$. Thus, this fault-free ring can be assigned to f_j . \square

4.3 Communication Complexity

Steps (2) and (3) are exactly equivalent to broadcasting in a Q_k^{n-1} and a Q_k^1 , respectively. Hence, the communication complexity of the proposed algorithm is the complexity of broadcasting in a Q_k^n plus the complexities of Steps (2) and (3). The complexities of Steps (2) and (3) depend on the packet routing strategy used. In the following the complexities are derived when cut through or store-and-forward is used.

Cut Through: In the worst case, two communication steps are needed in Step(1). First, the source node S sends the message M to an adjacent node S' in a fault-free adjacent ring. Next, S' sends M to any node in the fault-free sub-cube which is at most $(n - 1)$ hops away. Thus, Step(1) needs at most:

$$2(t_s + t_w M) + nt_h,$$

where t_s is the startup time, t_w is time per word, M is the size of broadcasted message in words, and t_h is time per hop. Furthermore, Step(4) needs exactly one communication step in which each faulty ring gets M from an adjacent fault-free ring and so the time for this step is:

$$t_s + t_w M + t_h,$$

Hence, the total communication complexity is:

$$n[\lg k](t_s + t_w M) + nt_h(k - 1) + 3(t_s + t_w M) + t_h(n + 1).$$

Store-and-Forward: In the worst case, here also, Step(1) needs to send M to an adjacent node S' in a fault-free ring. Next, M is sent from S' to a node in the fault-free sub-cube which is at a distance of $(n - 1)$ or less. The time required for this is:

$$2t_s + n(t_w M + t_h),$$

In addition, Step(4) needs exactly one communication step in which each faulty ring gets the message from a fault-free adjacent ring and so the time for this step is:

$$t_s + t_w M + t_h,$$

Hence, the resultant communication complexity in this case is:

$$(n \lceil \frac{k}{2} \rceil + n + 1)(t_s + t_w M + t_h) - (n - 2)t_s.$$

4.4 The Algorithm in Toroidal Networks

The above algorithm also works for any general toroidal network provided that at least one of the dimensions, say X , is greater than $(2n - 2)$ and all the other dimensions are greater than 3. The dimension X being greater than $(2n - 2)$ assures that there exists a fault-free sub-cube along this dimension. Furthermore, the other dimensions being greater than 3 guarantees that each faulty ring along the X dimension can be assigned a unique fault-free adjacent ring using a greedy-like strategy. These two conditions imply that Lemma 1 and Theorem 1 also hold in those cases and, hence, the algorithm works correctly. Furthermore, the communication complexities are exactly the same as those of Q_k^n case.

5 Conclusion

This paper presents a fault-tolerant broadcasting algorithm that can adapt up to $(2n - 2)$ failures for toroidal networks of dimension n . Since the connectivity of toroidal networks is $(2n)$, the algorithm is very close to optimal with respect to the maximum possible number of failures a broadcasting algorithm can tolerate.

The algorithm in the worst case needs three more communication steps than the optimal broadcasting algorithm used in a fault-free toroidal network, when cut through routing is used. When the systems uses store-and-forward method, the proposed algorithm requires at most $(n + 1)$ more steps than the optimal fault-free algorithm.

Acknowledgment

This work is supported by the NSF grant MIP-9705738.

References

1. A. Al-Dhelaan and B. Bose. "An Efficient Fault-Tolerant Broadcasting Algorithm for the Hypercube". In *4th Conference on Hypercube Concurrent Computers and Applications*, pages 123–128, 1989.
2. R. Alverson, D. Callahan, D. Cummings, B. Koblenz, A. Porterfield, B. Smith. "The Tera Computer System". Technical report, Tera Computer Company, 1991.
3. B. Bose, R. Broeg, Y. Kwon, and Y. Ashir. "Lee Distance and Topological Properties of k -ary n -cubes". *IEEE Transactions on Computers*, 44(8):1021–1030, August 1995.
4. R. Broeg and B. Bose. "Fault-Tolerant Broadcasting in A K -ary N -cube". In *Annual IEEE Workshop on Fault-Tolerant Parallel and Distributed Systems*, 1996. Chapter 15 of Fault-Tolerant Parallel and Distributed Systems, to be published by Kluwer Academic Publishers, Boston.
5. W. Dally. "Performance Analysis of k -ary n -cube Interconnection Networks". *IEEE Transaction on Computers*, 39(6):775–785, June 1990.
6. W. Dally and C. Seitz. "The Torus Routing Chip". *Journal of Distributed Computing*, 1(3):187–196, 1986.
7. W. Dally and C. Seitz. "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks". *IEEE Transaction on Computers*, 36(5):547–553, May 1987.
8. J. Draper and J. Ghosh. "Multipath E-cube Algorithms (MECA) for Adaptive Wormhole Routing and Broadcasting in k -ary n -cube". In *6th International Parallel Processing Symposium*, pages 407–410, March 1992.
9. S. Johnson and C. Ho. "Optimum Broadcasting and Personalized Communication in Hypercubes". *IEEE Transactions on Computers*, 38(9):1249–1268, September 1989.
10. P. Kermani and L. Kleinrock. "Virtual Cut-Through: A New Communication Switching Technique". *Computer Networks*, 3(4):267–286, 1979.
11. S. Lam. "Store-and-Forward Buffer Requirements in a Packet Switching Network". *IEEE Transactions on Communications*, COM-24(4):394–403, April 1976.
12. T. Chiang Lee and J. Hayes. "A Fault-Tolerant Communication Scheme for Hypercube Computers". *IEEE Transactions on Computers*, 41(10):1242–1256, October 1992.
13. L. Ni and McKinley. "A Survey of Wormhole Routing Techniques in Direct Connect Networks". *IEEE Computer*, 26(2):62–76, February 1993.
14. W. Oed. "Massively Parallel Processor System CRAY T3D". Technical report, Cray Research GmbH, November 1993.
15. S. Park and B. Bose. "Broadcasting in Hypercubes with Link/Node Failures". In *4th Symposium on the Frontiers of Massively Parallel Computation*, pages 286–290, 1992.
16. S. Park and B. Bose. "All-to-All Broadcasting in Faulty Hypercubes". *IEEE Transactions on Computers*, 46(7):749–755, July 1997.
17. S. Park, B. Bose, and R. Broeg. "Algorithms for Broadcasting in Faulty Hypercubes". In *6th International Conference on Parallel and Distributed Computing Systems*, pages 361–366, October 1993.
18. S. Scott and G. Thorson. "The Cray T3E Network: Adaptive Routing in High Performance 3D Torus". In *HOT Interconnects IV, Stanford University*, August 15-16 1991.

This article was processed using the L^AT_EX macro package with LLNCS style