AN ABSTRACT OF THE THESIS OF

Seungjin Park for the degree of Doctor of Philosophy in
Computer Science presented on March 4, 1993.

Title: Fault-Tolerant Communications in Parallel Systems

Abstract approved: Redacted for privacy

Dr. Bella Bose.

In distributed memory systems communication between processors is mainly done via message passing. Since communication time is more costly than computation time, efficient communication algorithms are essential to achieve high performance in these systems. Furthermore, since the messages may not be transmitted successfully due to some reasons such as noise and/or faulty components, the system should contain fault-tolerant features to avoid the problems.

Among many topologies suggested for parallel systems, the hypercube has been very popular due to its numerous merits such as regularity, easy construction, high fault-tolerance, etc.

In this thesis we present the research that has led to the following results in an n-dimensional hypercube.

1. Optimal fault-tolerant single node broadcasting which tolerates up to $n-1$ faulty links/nodes.

2. Near optimal fault-tolerant single node broadcasting which tolerates up to $\frac{n^2-n}{2}$ faulty links.

3. Near optimal fault-tolerant all-to-all broadcasting which tolerates up to $\frac{n}{2}$ faulty links.

4. Fault-tolerant all-to-all broadcasting which tolerates up to $n-2$ faulty links in wormhole-routed hypercubes, which produces a factor of approximately n less traffic than previously known algorithms.

Fault-Tolerant Communications in Parallel Systems

by

Seungjin Park

A Thesis

submitted to

Oregon State University

in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

Completed March 4, 1993

Commencement June 1993

Approved:

Redacted for privacy

Professor of Computer Science in charge of major

Redacted for privacy

Head of Department of Computer Science

Redacted for privacy

Dean of Graduate School

Date thesis is presented March 4, 1993

Typed by Seungjin Park for Seungjin Park

dedicated to my parents

Hankeum Park

Sookja Kim

ACKNOWLEDGMENTS

I would like to express my appreciation to all the people without whom this thesis could not have been completed. First, I would like to thank my parents, Hankeum Park and Sookja Kim, for their encouragement and support throughout my education. Without them I would not have reached this stage of my education. My special thanks to Professor Bella Bose, my major advisor, for his guidance, advice, financial support, and patience during the last four years. My success in this endeavor can be largely attributed to his assistance.

Many thanks to other members of my Ph.D. Committee: Professor T. Minoura, Professor V. Saletore, Professor P. Tadepalli and Professor P. Watson for their time and help during my preparation of this thesis.

I thank Professor J. Chandler at Oklahoma State University and Professor P. Hsia at University of Texas at Arlington for their guidance and special friendship.

My special thanks to my wife, Hyeryun, and my children, Inhye and Seik, for their patience and love.

I would like to thank my colleagues: Bob Broeg, Mark Clement, Phyl Crandle, Jie Liu, Bob Rowley, and Brad Seavers for their consistent efforts to improve this thesis.

Finally, I would like to thank to my Korean friends: Mungmun Bae, Eunbae Kong, Younggeun Kwon, Suckjun Lee, and Dokyung Ok who alway share good times with me.

Table of Contents

List of Figures

List of Tables

Fault-Tolerance Communications in Parallel Systems

Chapter 1

Introduction

Massively parallel distributed-memory machines are receiving considerable attention to meet the demand for extraordinarily powerful computers. In order to achieve efficient parallel processing many topologies have been suggested and advocated [Sto71, HZ81, BK79, PV81, SP89, YN90, Dal90, CS90a]. Among them, the hypercube has been very popular due to its various merits such as regularity, embeddability of many other topologies, relatively small diameter, easy construction and high potential for the accommodation of various algorithms. Significant research efforts [CS87, JH89, Sei85, SS88, B+92, CS90b, BS86, GS89] have led to several research [Sei85] and commercial hypercubes by Intel, NCUBE, Floating Point System, Ametek, Thinking Machine.

Parallel architectures often consist of thousands of processors, and in distributed memory systems communication between processors is mainly done via message passing. Thus, efficient communication schemes are extremely important

to achieve high performance in the systems. Many researchers have proposed various communication algorithms for hypercube multicomputers [JH89, SB77, HJ86, B+91, L+90, JH91]. However, the messages may not be successfully transmitted to the receiving nodes due to various impairments such as noise or faulty components in the system. Error correcting codes have been found very useful tool for correcting partial incorrectness of the transmitted messages[PB90a, PB90b]. Furthermore, if the system contains some faulty components, the communication algorithms listed above may not work properly.

One way to accomplish the fault-tolerance in the system is by reconfiguration, i.e., adding spare nodes and links to the system so that under certain faulty conditions, the faulty nodes/links are replaced by the redundant components [Ban89, LH89, JBH91b, JBH91a]. The disadvantages of this approach are (1) huge number of extra nodes and links may be needed as the number of faults increases, and (2) most of the extra nodes and links may be idle until some faults actually occur.

Our approach to tolerate faults is to devise communication algorithms which avoid those faulty components. Numerous fault-tolerant communication algorithms in this category have been proposed [CS90c, CS90d, RS88, Fra92, LH88, PB90c, CS89, PB92]. The differences between our approaches and the previous approaches are explained in the appropriate chapters.

In this chapter, we define the hypercube and introduce some of the com-

munication primitives such as routing and broadcasting and explain how they can be done in perfect hypercubes, i.e., hypercubes which do not contain any faulty component.

The *n-dimensional hypercube*, also called n-cube, is denoted as Q_n, and it has $N = 2^n$ nodes and $n2^{n-1}$ links. Each node has a unique address $(a_{n-1}, a_{n-2}, \ldots, a_0)$, where $a_i \in \{0, 1\}$ for $i = 0, 1, \ldots, n-1$. Two nodes are connected by a link iff their addresses differ by exactly one bit.

Let the nodes a and b differ in the i-th bit. Then the link between a and b is uniquely represented by $(a_{n-1}a_{n-2}\ldots a_{i+1} - a_{i-1}\ldots a_0)$ where $a = (a_{n-1}a_{n-2}$ $\ldots$ a_{i+1} a_i a_{i-1} $\ldots$ $a_0)$. We also say that the link is in the i-th dimension. For example, the link connecting two nodes 10110 and 10100 is denoted as $101 - 0$ and is in dimension 1.

The definition of the product of graphs is as follows [CS87].

Definition 1.1: Let $G_p = (V_p, E_p)$ be the product of two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, denoted by $G_p = G_1 \times G_2$. Then $V_p = V_1 \times V_2$ and two nodes $u = (u_1, u_2)$ and $v = (v_1, v_2)$ are adjacent in G_p iff $[u_1 = v_1$ and u_2 adjacent to $v_2]$ or $[u_1$ adjacent to v_1 and $u_2 = v_2]$.

Then Q_n can be recursively defined as follows.

Definition 1.2: A Q_n can be expressed as

a) Q_0 is a trivial graph with one node, and

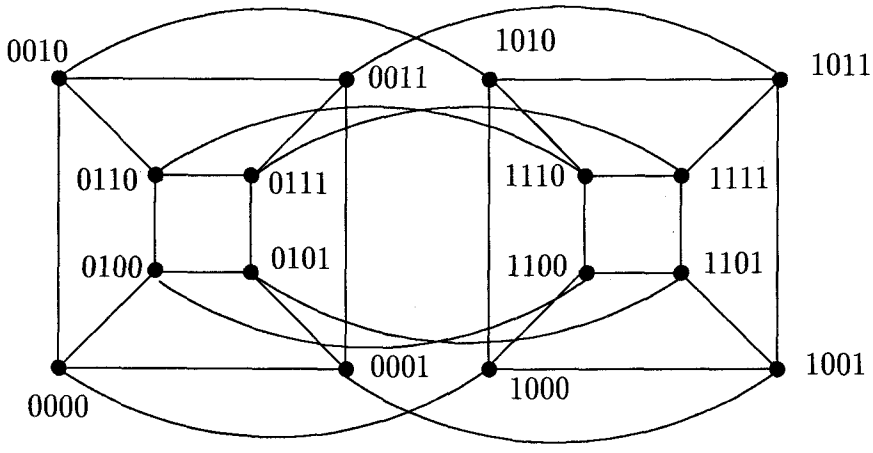

Figure 1.1. Construction of Q_4 from two Q_3's.

b) $Q_n = K_2 \times Q_{n-1}$, where K_2 is the complete graph with two nodes.

Figure 1.1 shows how Q_4 can be constructed from two Q_3's.

The routing algorithm in an interconnection network is the mechanism by which packets are guided from their sources to their destinations through the network. The main object of the routing algorithm is to select paths of small total delay for each packet. The problem of minimum delay routing from a source node to a destination node would then be reduced to the problem of finding a path connecting the two nodes with minimum sum of link delay [BT89].

The *Hamming distance* between two nodes, a and b, is denoted by a bitwise *Exclusive-Or* operation of the two nodes, $a \oplus b = c = (c_{n-1}, c_{n-2}, \ldots, c_0)$, where $c_i = a_i \oplus b_i$ for $i = 0, 1, \ldots, n-1$. The number of links on any path between two nodes can not be less than the Hamming distance of the two nodes. Furthermore, there exists at least one path with a number of links that is equal to the Hamming

distance. Such a path can be obtained by switching in sequence the bits in which the addresses of the two nodes differ. For example, in Q_5, let nodes 00010 and 10101 be the source and destination nodes, respectively. Then $00010 \oplus 10101 = 10111$. Then the routing path can be obtained by converting the bits in 10111 one by one from lowest to highest dimension as follows.

$$00010 \rightarrow 00011 \rightarrow 00001 \rightarrow 00101 \rightarrow 10101.$$

Johnsson and Ho [JH89] introduce four different communication primitives, 1) *one-to-all broadcasting* (or *single node broadcasting*) in which a single node distributes a common data to all other nodes, 2) *one-to-all personalized communication* (or *scattering*) in which a single node sends unique data to all other nodes, 3) *all-to-all broadcasting* (or *multinode broadcasting*) in which all nodes broadcast concurrently to all other nodes, and 4) *all-to-all personalized communication* (or *total exchange*) where each and every node sends a unique data to every other node. Figure 1.2 explains the primitives in detail.

Communication algorithms can be implemented in either *one*-port or *n*-port model. In an one-port model, a node can transmit a packet along at most one incident link and can simultaneously receive a packet along at most one incident link, whereas in an n-port model all incident links of a node can be used simultaneously for packet transmission and reception.

In the case of single node broadcasting, Sullivan et al.[SB77] have given what is now the standard algorithm, called *e*-cube algorithm, for broadcasting in

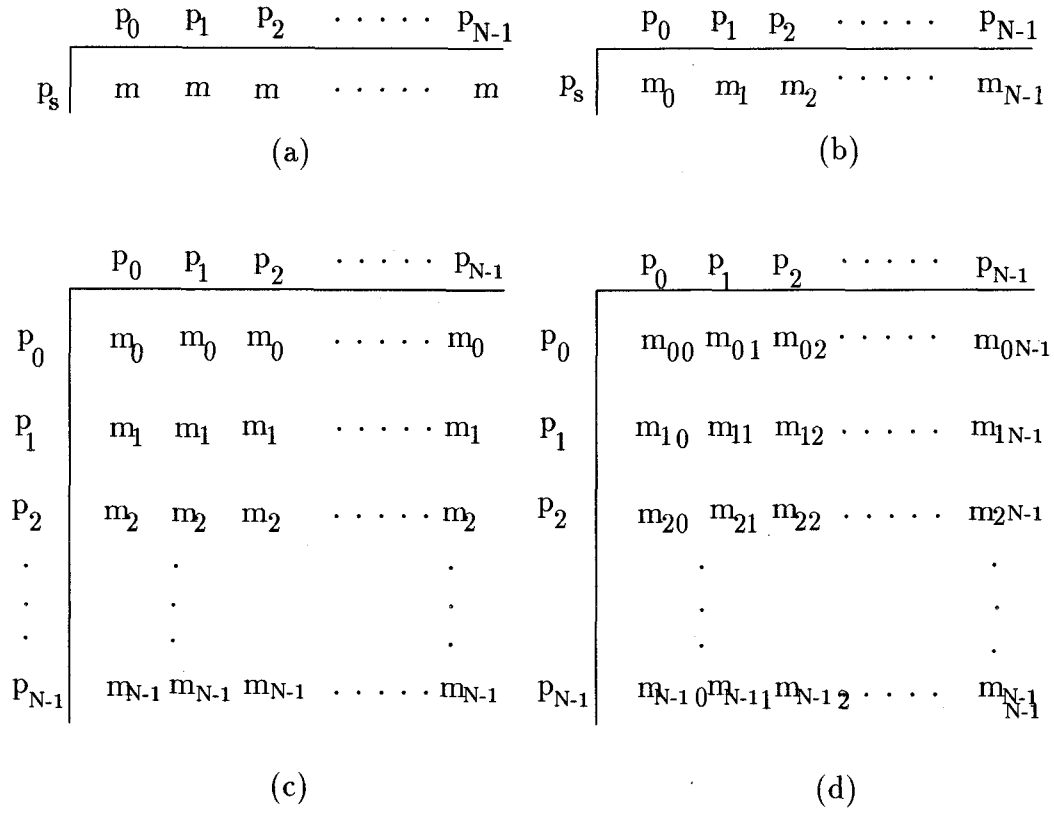

Figure 1.2. Communication primitives. (a) *single node broadcasting* in which source node p_s send same message to all other nodes, (b) *scattering* in which source node p_s send unique messages to all other nodes, (c)*all-to-all broadcasting*, where processor p_i broadcasts its message $m_i, i = 0, 1, \ldots, N-1$, (d) *total exchange*, where node p_i sends the message m_{ij} to processor p_j, for all $0 \leq i \leq N-1$ and $0 \leq j \leq N-1$.

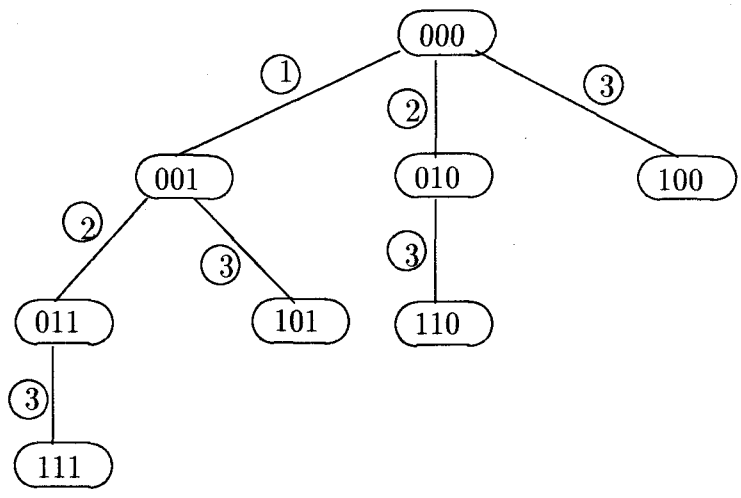

Figure 1.3. Broadcasting tree in Q_3. Here node 000 is a source node in Q_3.

the hypercube multicomputers. This algorithm works as follows. In the first time unit the source node sends the broadcast message along the 0-th dimension and thus at the end of the first step two nodes will have the message; at the second time unit both of these nodes send the message along the first dimension, so four nodes will have the message at the end of the second time unit; next all of these four nodes will send the message along the second dimension, and so on. At the end of n time units all 2^n nodes will have the message. Since the diameter (the longest path length between any two nodes) of Q_n is n, the e-cube algorithm is optimal. Figure 1.3 shows the *broadcasting tree* which is resulted by e-cube algorithm in Q_3 with source node as 000, which is also known as a binomial tree.

Other communication primitives are described in [JH89, B$^+$91, Fra92, L$^+$90, BT89, HJ86, JH91].

Chapter 2

Single Node Broadcasting in Faulty Hypercubes

2.1 Introduction

Parallel processing has been known as the only solution to overcome the von Neumann bottleneck which is caused by sequential request and reply between single CPU and memory [Tan90]. In order to achieve efficient parallel processing many topologies have been suggested and advocated [Sto71, HZ81, BK79, PV81, SP89, YN90]. Among them, the hypercube has been very popular due to its various merits such as regularity, embeddability of many other topologies, relatively small diameter, easiness of construction, etc.

Parallel architectures often consist of thousands of processors, and in distributed memory systems communication between processors are mainly done via message passing. Thus, efficient communication schemes are extremely important to achieve high performance in the systems. Many researchers have proposed various communication algorithms for hypercube multicomputers [JH89, SB77, HJ86, B+91,

L+90, JH91]. However, most of these communication schemes do not work properly in the presence of faulty components in the system. Numerous fault-tolerant communication algorithms have been proposed [CS90c, CS90d, RS88, Fra92, LH88, PB90c, CS89, PB92].

Lee and Hayes [LH88] have proposed fault-tolerant broadcasting algorithm based on the concept of *unsafeness* of a node which may cause communication difficulties in faulty hypercubes. They showed that by avoiding, if possible, these unsafe nodes, broadcasting can be easily achieved. However, if there are more than $\lfloor \frac{n}{2} \rfloor$ faulty nodes in an n-dimensional hypercube, all the nodes in the hypercube become unsafe, so their algorithm can tolerate up to $\lfloor \frac{n}{2} \rfloor$ faulty nodes/links. Their algorithm takes $n + 1$ time steps. Ramanathan and Shin[RS88] have described fault-tolerant broadcasting algorithm in which source node delivers multiple copies of the broadcasting message to all other nodes in the faulty hypercube through edge disjoint paths. They advocate that the algorithm is suitable for real-time applications since source node does not have to know the identities of the faulty components. However, this approach may cause much more traffic in the system than the one in which each node receives only one copy of the broadcast message. Their algorithm can tolerate up to $n - 1$ faulty components, and it takes $n + 1$ and $2n$ time steps for n-port and *one*-port communications, respectively.

In this chapter a simple and optimal fault-tolerant broadcasting algorithm in hypercube multicomputers in the presence of up to $n-1$ faulty links is given. Further results for up to $2n - 3$ faulty links are also described. In addition, fault-tolerant

broadcasting algorithm in the presence of $n-1$ faulty *nodes* is also presented. Our algorithm takes $n+1$ time steps even in the presence of $n-1$ faulty links or nodes; this can be achieved even with one-port communication. For up to $2n-3$ link or node faults, the proposed algorithm takes at most $n+3$ time steps even with one-port communication.

The outline of the chapter is as follows. Section 2.2 summarizes the notations and definitions which will be used throughout the chapter. Section 2.3 introduces a new broadcasting algorithm which can tolerate up to $n-1$ link faults. In Section 2.4, we extend our algorithm to tolerate up to $2n-3$ faulty links. Broadcasting algorithm with node failures is presented in Section 2.5. The conclusion follows in Section 2.6.

2.2 Preliminaries

An n-dimensional hypercube, Q_n, consists of 2^n nodes and $n2^{n-1}$ links. Each node has a unique address $(a_{n-1}, a_{n-2}, \ldots, a_0)$, where $a_i \in \{0,1\}$ for $i = 0,1,\ldots,n-1$. Two nodes are connected by a link iff their addresses differ by exactly one bit.

Let the nodes a and b differ in the i-th bit. Then the link connecting a and b is uniquely represented by $(a_{n-1}a_{n-2}\ldots a_{i+1} - a_{i-1}\ldots a_0)$ where $a = (a_{n-1}a_{n-2}\ldots a_0)$. We also say that the link is in the i-th dimension. For example, the link connecting two nodes 10110 and 10100 is denoted as $101-0$ and is in dimension 1.

In the case of broadcasting, sometimes called single node broadcasting, a

single node sends the same message to all other nodes. As explained in Chapter 1, Sullivan et al.[SB77] have given what is now the standard algorithm, called e-cube algorithm, for broadcasting in the perfect hypercube multicomputers.

Any subcube Q_x in Q_n, $x \leq n$, can be uniquely represented by a sequence of n ternary symbols $(t_{n-1}, t_{n-2}, \ldots, t_0)$, $t_i \in \{0, 1, *\}$, $0 \leq i \leq n-1$, where $*$ is a *don't care* symbol. For example, the subcube $011**$ consists of the nodes {01100, 01101, 01110, 01111}. If we divide (*or partition*) Q_n into two subcubes along dimension d, the addresses of the two Q_{n-1} subcubes are $***\ldots 1_d**\ldots**$ and $***\ldots 0_d**\ldots**$. For example, if Q_4 is divided along the 1st dimension, the resulting two subcubes are $**0*$ and $**1*$. In the following Q_x^p indicates an x-dimensional hypercube which contains at most p faulty links.

The originator or source node is the node which initiates the broadcasting in Q_n, and originating cube is the subcube which contains the originator node. A cube (or subcube) is called *faulty* if it contains some faulty links or nodes and *perfect* if it doesn't.

The following assumptions are made in this chapter.

(1) Each node knows all the identities of faulty links and nodes in the networks.

(2) It takes one time unit to send a message to an adjacent node.

2.3 Broadcasting strategy in the presence of up to $n-1$ faulty links

In the e-cube algorithm only $2^n - 1$ out of $n2^{n-1}$ links are used. Also note that all the 2^{n-1} links in one dimension (i.e., the dimension used in the last step of the e-cube algorithm) participate in the broadcasting. For example, in Q_3, if 000 is the originator node, then all the links in dimension 2 are used in step 3. Thus, if there is no faulty link in dimension 2 and broadcasting is done in one of the Q_2's, $1**$ or $0**$, then all the nodes in the other Q_2 can receive the message along the dimension 2. Using these observations, we propose a new fault-tolerant broadcasting algorithm for Q_n^{n-1}.

Let F be the set of faulty links in Q_n. If F is empty, broadcasting can be done using e-cube algorithm. When there are faulty links in Q_n, at least one of the dimensions, say dimension p, does not contain any faulty link since $|F| \leq n-1$. Let us divide Q_n into two subcubes, Q_{n-1} and Q'_{n-1}, along dimension p. Then broadcasting in Q_n can be done if we could broadcast in one of the subcubes, say Q_{n-1}; this is because all the nodes in Q'_{n-1} can receive the message from the corresponding nodes in Q_{n-1}, since no link in dimension p is faulty. When we divide Q_n along dimension p, two cases can occur.

CASE 1) All the faulty links belong to one of the two subcubes, say Q'_{n-1}. Suppose the source node is in Q_{n-1}. Then using the e-cube algorithm, broadcasting can first be done in Q_{n-1}; then all the nodes in Q_{n-1} can send the message to the

corresponding nodes in Q'_{n-1} along dimension p.

On the other hand if the source node is in Q'_{n-1}, first the source node can send the message to its neighboring node in Q_{n-1}; then the steps of CASE 1 described above can be repeated except that the receiving node should not send the message back to the sender. In any case broadcasting in Q_n can be done in at most $n+1$ time units.

CASE 2) Each subcube contains some faulty links. The subcube, say Q_{n-1}, which contains the source node can have at most $n-2$ faulty links because the other subcube contains at least one faulty link. If we could broadcast the message among the nodes in Q_{n-1}, then the complete broadcasting can be done in one more step by sending the message from the nodes in Q_{n-1} to the nodes in Q'_{n-1} along p-th dimension. Now the original problem of broadcasting in Q_n with up to $n-1$ faulty links is reduced to the problem of broadcasting in Q_{n-1} with up to $n-2$ faulty links. We keep dividing the subcube Q_i which contains the originator node into two subcubes, Q_{i-1} and Q'_{i-1}, along fault-free dimensions for $i = n-1, n-2, \ldots, 2$. Eventually for some $k \geq 1$, we will get a subcube Q_k which does not contain any faulty link. If the adjacent subcube Q'_k contains the source node, first that node can send the message to the corresponding node in Q_k. On the other hand, if Q_k itself contains the originator node this extra step is unnecessary. Now using the e-cube algorithm broadcasting can be done first in Q_k in k steps. Then the message can be successively sent along the fault-free dimensions to the new nodes. In any case the complete broadcasting can be done in $n+1$ steps.

Example 2.1. Let $F = \{000-, 11-1, 00-0\}$ in Q_4. Let node 0000 be the source node. Here, no link in dimension 2 or 3 is faulty. Choose any one from these fault-free dimensions, say dimension 2, and divide Q_4 along this dimension. The originating cube $*0**$ contains the faulty links 000− and $00-0$, and no link in dimension 3 in this cube is faulty. Since neither of the resulting Q_3's is perfect, the originating cube $*0**$ is again divided into two subcubes, $Q_2 = 00**$ and $Q_2' = 10**$. Now Q_2' is perfect, but Q_2 contains the source node. The source node 0000 sends the message to node 1000 in Q_2'. Then broadcasting in Q_2' can be done in two steps. Nodes except 1000 send the message to the nodes in Q_2 in the next time step. Finally nodes in Q_3, which is the union of Q_2 and Q_2', send the message to the corresponding nodes in Q_3'. □

In the following we prove the correctness and optimality of the above algorithm.

Theorem 2.1. Every node in the hypercube receives the broadcast message exactly once.

Proof : The algorithm recursively divides originating cube Q_x into two Q_{x-1}'s along the dimension p such that no link which is in dimension p and in cube Q_x is faulty. The division precess continues until a perfect subcube is found. After completing broadcasting in the perfect subcube using e-cube algorithm, it starts to broadcast along the fault-free dimensions. In this way every node receives the message exactly once. Note that for the case when both all the faulty links and the source node are

in the same subcube, we have modified the algorithm so that the source node does not receive the message it sent. □

Theorem 2.2. At least $n+1$ time steps are needed to broadcast in Q_n^{n-1}.

Proof : Suppose the source node is $000\ldots000$ and the $n-1$ faulty links are $0111\ldots111-, 0111\ldots11-1, 0111\ldots1-11, \ldots, 0-11\ldots1111$. Then node $0111\ldots1111$ has to receive the message from node $111\ldots111$. But the distance between $00\ldots00$ and $111\ldots111$ is n. Thus node $0111\ldots1111$ receives the message only after n time units. □

Since the lower bound is $n+1$ and the proposed algorithm takes at most $n+1$ time steps, the given algorithm is optimal.

2.4 Broadcasting strategy in the presence of up to $2n-3$ faulty links

Even in the presence of up to $2n-3$ link failures, we assume that the hypercube is connected. In this case, there must exist a dimension where at most one link can be faulty. This is because if all the dimensions have two or more faulty links then we will have more than $2n-1$ link failures which contradicts our assumption.

Let p be the dimension with least number of link faults. Divide the hypercube along p-th dimension to get the two subcubes Q_{n-1} and Q'_{n-1}. Then two cases can occur.

CASE 1) Suppose there is no link fault along this p-th dimension. Note that one of the subcubes, say Q_{n-1}, must contain at most $n-2$ faulty links. If the source node is in Q_{n-1}, then using the algorithm developed in the previous section, first broadcasting can be done in Q_{n-1} and then all nodes in Q_{n-1} can send the message along dimension p to nodes in Q'_{n-1}. However, if the source node is in Q'_{n-1}, first the source node can send the message to the corresponding node in Q_{n-1}; then the above steps can be repeated except that the receiving node should not send the message back to the sender. In any case broadcasting can be done in at most $n+2$ steps.

CASE 2) Suppose the number of faulty links in dimension p is 1. Again one of the subcubes, say Q_{n-1}, will have at most $n-2$ faulty links. If the source node is in Q_{n-1}, broadcasting in Q_{n-1} can first be done in n steps using the algorithm developed in the previous section. Then all nodes in Q_{n-1} send the message along dimension p to nodes in Q'_{n-1}. In this case exactly one node, say b, in Q'_{n-1} will not receive the message because there is a faulty link in dimension p. Note that not all links connecting b to the nodes in Q'_{n-1} can be faulty. Otherwise node b would have been disconnected from the rest of the nodes and this is contradictory to our assumption that the faulty hypercube is connected. Thus b can receive the message from one of the adjacent nodes which is in Q'_{n-1}.

On the other hand if the source node is in Q'_{n-1}, first it can try to send the message to a node in Q_{n-1}. If the link connecting the source node which is in Q'_{n-1} and its corresponding node in Q_{n-1} is not faulty then this can be done in one step;

otherwise this can be done in two steps. After this all the steps described in the previous paragraph can be executed. In any case the complete broadcasting can be done in $n + 3$ steps.

2.5 Broadcasting strategy in the presence of up to $n - 1$ faulty nodes

Our new broadcasting algorithm in hypercube multicomputers with node failures is again based on the fact that the hypercube has a recursive structure; i.e., Q_n is composed of two Q_{n-1}'s. We recursively divide Q_n into smaller subcubes such that one fault-free node in each subcube contains the message, thus the original problem of broadcasting in Q_n is divided into two subproblems of broadcasting in Q_{n-1} with at most $n - 2$ faulty nodes in each subcube. Once we have found the fault-free subcube, nodes in the corresponding faulty subcube will receive the message from the nodes in the fault-free subcube. The detail of our algorithm is given below.

In the presence of a single faulty node $b = (b_{n-1}b_{n-2}\ldots b_0)$ the source node $a = (a_{n-1}a_{n-2}\ldots a_0)$ can broadcast to all other nodes in n time steps as follows. Let a and b differ in some bit, say i-th bit (i.e., $a_i = \bar{b_i}$). Then the subcube $Q_{n-1} = **\ldots*a_i*\ldots**$ contains the source node a but not the faulty node b. Now node a can broadcast to all the nodes in Q_{n-1} in $n-1$ steps using the e-cube algorithm. In the n-th step all the nodes in Q_{n-1} can send the message to all the nodes except b in the other subcube $Q'_{n-1} = **\ldots*b_i*\ldots**$.

Now consider the case for $t \leq n-1$ faulty nodes in Q_n. Let $a_0 = (a_{0n-1}, a_{0n-2}, \ldots, a_{00})$ be the source node and $a_k = (a_{kn-1}, a_{kn-2}, \ldots, a_{k0})$ for $k = 1, 2, \ldots, t$ be the t faulty nodes.

CASE 1) Suppose values of some bit, say bit i, of all the faulty nodes are the same. Then we can partition the original cube Q_n into two Q_{n-1}'s, $Q_{n-1} = * * * \ldots * *a_{ki} * * \ldots * *$ and $Q'_{n-1} = * * * \ldots * *\bar{a_{ki}} * * \ldots * *$ such that all the faulty nodes are in Q_{n-1} and none of the nodes in Q'_{n-1} is faulty. Note that Q_{n-1} and Q'_{n-1} are adjacent each other. If the source node a_0 is in Q'_{n-1} then a_0 can broadcast in Q'_{n-1} using e-cube algorithm in $n-1$ steps; in the n-th step all the nodes in Q'_{n-1} can send the message to all the non-faulty nodes in Q_{n-1}. However, if the source node a_0 is in Q_{n-1} which contains all the faulty nodes, first a_0 can send the message to the adjacent node a'_0 which is in Q'_{n-1}. Then a'_0 can broadcast the message to all the nodes in Q'_{n-1} in $n-1$ time steps using the e-cube algorithm and finally all the nodes in Q'_{n-1} can send the message to all the nodes other than the faulty nodes and a_0 in Q_{n-1} in one step. Thus the total number of time steps taken by the algorithm will be $n+1$.

CASE 2) Now let us consider the other case; i.e., no position of the $t \leq n-1$ faulty nodes has the same bit value. Let a'_0 be a non-faulty adjacent node of the source node a_0. Let a_0 and a'_0 differ in the i-th bit position, i.e., $a'_0 = (a_{0n-1}, a_{0n-2}, \ldots, a_{0i+1}, \bar{a_{0i}}, a_{0i-1}, \ldots, a_{00})$. Node a_0 can send the message to a'_0 in one step. Now consider the subcubes $Q_{n-1} = * * * \ldots * *a_{0i} * * \ldots * **$ and $Q'_{n-1} = * * * \ldots * *\bar{a_{0i}} * * \ldots * *$. Not all faulty nodes can be in Q_{n-1} or in Q'_{n-1}. If

they were, the i-th bit of the faulty nodes would have the same bit value; this would contradict the original assumption. Let there be t_1 and t_2 faults in Q_{n-1} and in Q'_{n-1} respectively, where $1 \leq t_1 < n-1, 1 \leq t_2 < n-1$ and $t_1 + t_2 = t \leq n-1$. After one step, the original problem of broadcasting in Q_n in the presence of $t \leq n-1$ faulty nodes with a_0 as the source node is reduced to two subproblems of broadcasting in Q_{n-1} and in Q'_{n-1} with a_0 and a'_0 respectively as the source nodes. Each subcube can have at most $n-2$ faulty nodes.

Now depending on the bit values of the faulty nodes in Q_{n-1} and Q'_{n-1} either CASE 1 or CASE 2 can be applied to each subcube. In any case the maximum number of steps taken by the algorithm will be at most $n+1$.

By using similar ideas shown in Section 2.4, broadcasting in hypercube multicomputers with up to $2n-3$ faulty nodes can be achieved.

2.6 Conclusion

We have given broadcasting algorithms, one that tolerates up to $n-1$ and the other that tolerates up to $2n-3$ link faults in an n-dimensional hypercube. An optimal fault-tolerant broadcasting algorithm for up to $n-1$ faulty *nodes* is also described. Even though the implementation of the algorithm is non-adaptive, it can be made adaptive with some minor modifications.

Chapter 3

Highly Fault-Tolerant Single Node Broadcasting in Hypercubes

3.1 Introduction

In this chapter we concentrate on single node broadcasting in hypercubes in which a single source node sends a broadcast message to all other nodes. Lee and Hayes [LH88] have proposed an algorithm that achieves fault-tolerant broadcasting in the presence of up to $\lfloor \frac{n}{2} \rfloor$ faulty nodes. They introduce the concept of *unsafeness*; if the status of a node is unsafe, then the messages to be routed through the node may experience some difficulty. They show that by avoiding, if possible, these unsafe nodes, broadcasting can be achieved. However, if the hypercube contains more than $\lfloor \frac{n}{2} \rfloor$ faults, all the nodes in the cube become unsafe. Their algorithm takes $n+1$ time steps. Ramanathan and Shin [RS88] have proposed a broadcasting algorithm which delivers multiple copies of the broadcast message to all nodes through multiple edge-disjoint paths. In their algorithm the broadcasting node does not have to know the identities of the faulty components in the network, so it may be suitable for the real-

time applications. However, this approach may cause a large amount of unnecessary traffic in the system. Their algorithm can tolerate up to $n-1$ faulty components, and it takes $n+1$ and $2n$ time steps for n-port and *one*-port communications, respectively.

In Chapter 2, we have proposed a broadcasting algorithm which can tolerate up to $n-1$ link/node faults. The algorithm takes $n+1$ time steps, which is optimal [PB92]. Further, it is shown in Section 2.4 that the proposed algorithm with some minor modifications can easily tolerate up to $2n-3$ faults. The algorithm which tolerates up to $n-1$ link faults will be briefly explained in Section 3.3. This chapter presents a new broadcasting algorithm which can tolerate up to $\frac{n^2-n}{2}$ faulty links. The assumptions made in this chapter are (1) even though there are up to $\frac{n^2-n}{2}$ faulty links, the network is connected, and (2) each node knows the identities of the faulty links.

The rest of the chapter is organized as follows. Section 3.2 introduces the notations, definitions and background which will be used throughout the chapter. Section 3.3 briefly explains the algorithm which completes broadcasting in n-dimensional hypercube with up to $n-1$ faulty links. Section 3.4 proposes a new fault-tolerant broadcasting algorithm in hypercubes with up to $\frac{n^2-n}{2}$ faulty links. The conclusion follows in Section 3.5.

3.2 Notations and background

An n-dimensional hypercube, Q_n, consists of 2^n nodes and $n2^{n-1}$ links. Each node has a unique address $(a_{n-1}, a_{n-2}, \ldots, a_0)$, where $a_i \in \{0,1\}$ for $i = 0, 1, \ldots, n-1$. Two nodes are connected by a link iff their addresses differ by exactly one bit. Bitwise *Exclusive-Or* operation of the two nodes, a and b, is denoted by $a \oplus b = c = (c_{n-1}, c_{n-2}, \ldots, c_0)$, where $c_i = a_i \oplus b_i$ for $i = 0, 1, \ldots, n-1$. e^i denotes a unit vector such that all the bits have value 0 except the i-th bit which has value 1. Thus if nodes a and b are adjacent to each other, then $a \oplus b = e^i$ for some i. It is said that link connecting the two nodes is in dimension i, and this link is represented uniquely by $(a_{n-1}, a_{n-2}, \ldots, a_{i+1}, -, a_{i-1}, \ldots, a_0)$, where the address of the node a is $(a_{n-1}, a_{n-2}, \ldots, a_0)$. For example, the link connecting nodes 01010 and 01011 is denoted as 0101- and is in dimension 0. There are 2^{n-1} links in each dimension.

A dimension is *fault-free* if no faulty link is in that dimension. Let S be a set of links. Then $\|S\|$ is a n-dimensional vector whose i-th component, denoted $\|S\|(i)$, is equal to the number of links in dimension i in S. For example, if $S = \{01-00, 1011-, 0111-, -1101, 0011-\}$, then $\|S\| = 10103$, and $\|S\|(2) = 1$.

Any subcube Q_x in Q_n, $x \leq n$, can be uniquely represented by a sequence of n ternary symbols $(t_{n-1}, t_{n-2}, \ldots, t_0)$, $t_i \in \{0, 1, *\}$, $0 \leq i \leq n-1$, where $*$ is a *don't care* symbol. For example, the subcube $011**$ consists of the nodes {01100, 01101, 01110, 01111}. If we divide (*or partition*) Q_n into two subcubes along dimension d, the addresses of the two Q_{n-1} subcubes are $***\ldots 1_d **\ldots **$ and $***\ldots 0_d **\ldots **$.

For example, if Q_4 is divided along the 1st dimension, the resulting two subcubes are $**0*$ and $**1*$. $FAULT(Q_x)$ denotes the number of faulty links in Q_x.

A cube is *connected* if there is a path between every pair of nodes. Likewise, a subcube is connected if for every pair of nodes in the subcube, there exists a path between them such that all the links in the path are in the subcube. A cube is called *faulty* if it contains some faulty links and *perfect* otherwise.

In the case of broadcasting, a single node, called a *source node*, sends the same message to all other nodes. Sullivan and Bashkow [SB77] have given what is now the standard algorithm, called the e-cube algorithm, for broadcasting in perfect hypercubes. The detail of the e-cube algorithm is explained in Chapter 1.

3.3 Broadcasting algorithm which tolerates up to $n-1$ faulty links

In this section we will briefly explain the algorithm proposed in [PB92] which tolerates up to $n-1$ faulty link in Q_n. Even though it may be a repetition of Section 2.3, we present the algorithm in slightly different way with different example. The algorithm will be referred to as $BRST^{n-1}$ in the rest of the chapter.

Let s and F denote the source node and the set of faulty links in Q_n, respectively. If F is empty, broadcasting can be done using e-cube algorithm. When there are faulty links in Q_n, there exists at least one fault-free dimension, say dimension p, since $|F| \leq n-1$. Let us divide Q_n into two subcubes, Q_{n-1} and Q'_{n-1}, along the

fault-free dimension p. Then the broadcasting in Q_n can be done if broadcasting in one of the subcubes, say Q_{n-1}, can be done. This is because all the nodes in Q'_{n-1} can receive the broadcast message from the corresponding nodes in Q_{n-1}, since no link in dimension p is faulty. When Q_n is divided along dimension p, two cases can occur.

CASE 1 : All the faulty links belong to one of the subcubes, say Q'_{n-1}. Suppose the source node, s is in Q_{n-1}. Then using the e-cube algorithm, broadcasting can first be done in Q_{n-1}, and then all the nodes in Q_{n-1} send the message to the corresponding nodes in Q'_{n-1} along the dimension p. On the other hand if s is in Q'_{n-1}, first s can send the message to its corresponding node, $s' = s \oplus e^p$, in Q_{n-1}, and then the steps of CASE 1 described above can be repeated except that the receiving node should not send the message back to the sender. In any case, broadcasting in Q_n can be done in at most $n+1$ time units.

CASE 2 : Both subcubes contain some faulty links. In this case the subcube, say Q_{n-1}, which contains the source node can have at most $n-2$ faulty links because the other subcube contains at least one faulty link. If broadcasting in Q_{n-1} is done, then the complete broadcasting can be done in one more time step by sending the message from the nodes in Q_{n-1} to nodes in Q'_{n-1} along the p-th dimension. Now the original problem of broadcasting in Q_n with up to $n-1$ faulty links is reduced to the problem of broadcasting in Q_{n-1} with up to $n-2$ faulty links. We keep dividing the subcube Q_i which contains the source node into two subcubes, Q_{i-1} and Q'_{i-1}, along fault-free dimensions for $i = n-1, n-2, \ldots, 2$, at the same time the fault-free

dimensions are pushed into a stack, *STACK*. Eventually for some $k \geq 1$, we will get a perfect subcube Q_k. If the adjacent subcube Q'_k contains the source node s, first s sends the message to the corresponding node, $s \oplus e^r$, in Q_k, where r is the dimension dividing Q_k and Q'_k. On the other hand if Q_k itself contains the source node, this extra step is unnecessary. Now using the e-cube algorithm, broadcasting can be done first in Q_k in k steps. Then the message can be successively sent along the fault-free dimensions which are obtained by popping the *STACK*. In any case, the complete broadcasting can be done in $n+1$ time steps. The following example illustrates this.

Example 3.1. Let the source node be 00000 and $F = \{0011-, 001-0, 00-00, 1100-\}$ be the set of faulty links in Q_5. Then, since dimensions 3 and 4 are fault-free, we arbitrarily choose dimension 3 and divide Q_5 along it. Since both subcubes $*0***$ and $*1***$ contain some faulty links, CASE 2 will be applied; Dimension 3 is pushed into the *STACK*, and we have a smaller problem - broadcasting in $Q_4 = *0***$ with $F = 0X11-, 0X1-0, 0X-00$, where X is a *don't care* symbol. Since dimension 4 is fault-free, $*0***$ is divided along it, which produces two subcubes, $00***$ and $10***$. At the same time, dimension 4 is pushed into the *STACK*. Refer to Figure 3.1. Since subcube $10***$ does not contain any faulty links, source node 00000 sends the message to node 10000. The e-cube algorithm completes broadcasting in $10***$ in three steps. At step 4, all nodes in $10***$ except node 10000 send the message along dimension 4 which is obtained by popping the *STACK*. Now, all nodes in $*0***$ have the message, and at step 5 these nodes send the message

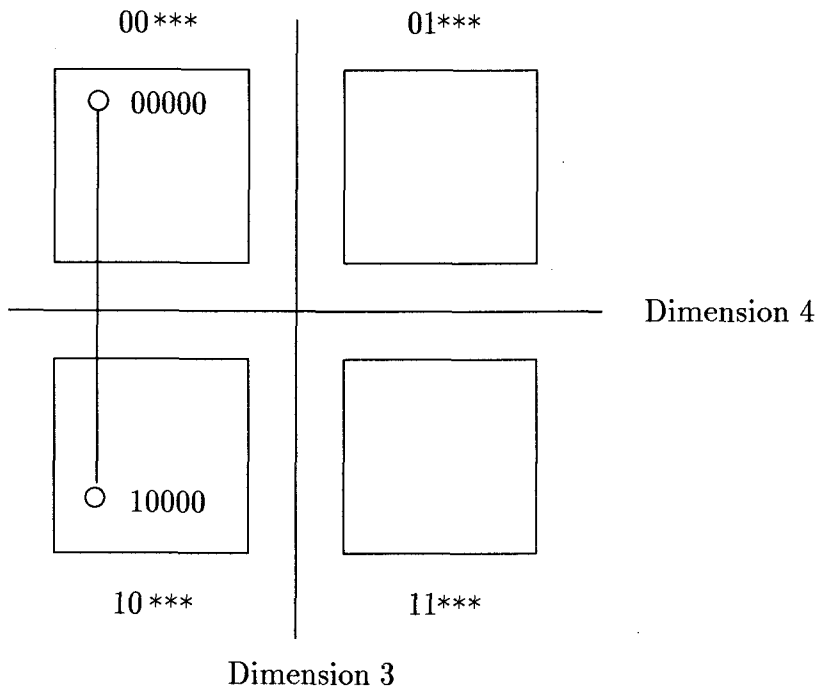

Figure 3.1. Division of Q_5 along two dimensions. Q_5 is divided first along dimension 3 and then along dimension 4. Node 00000 is the source node, and 0011-, 001-0, 00-00, 1100- are the faulty links.

along dimension 3 which, again, is obtained by popping the *STACK*. At this point broadcasting in Q_5 is completed. □

3.4 New single node broadcasting algorithm

In the previous section, a broadcasting algorithm for Q_n with $n-1$ faulty links was described. In that algorithm, Q_n is continuously divided into smaller subcubes until a perfect subcube is found and at the same time the sequence of the dimensions along which Q_n is divided is saved. After the perfect subcube is found, broadcasting is completed in the subcube first, and then the broadcasting for the rest of the Q_n

is done along the sequence of the dimensions stored. The algorithm works since 1) whenever a subcube, say Q_m, is partitioned into smaller subcubes, Q_{m-1}'s, it is guaranteed that one of the subcubes contains at least one fewer fault than Q_m, and 2) the subcube is connected since it contains at most $m-2$ faulty links.

Note that the algorithm tolerates *only* up to $n-1$ faults since *only one* faulty link is guaranteed to be eliminated when a cube is partitioned into smaller subcubes. Thus, for example, if there is an algorithm which guarantees that two faulty links are eliminated by each partitioning, the algorithm will tolerate up to $2n-2$ faulty links. The above observation leads us to the following question : Can we develop an algorithm that can guarantee that a large number of faulty links are eliminated by each partitioning? This chapter presents an algorithm which guarantees that at least $m-1$ faulty links are eliminated when $Q_m, m \leq n$, is partitioned. Thus, our algorithm tolerates up to $1+2+...+(n-1) = \frac{n^2-n}{2}$ faulty links.

The basic idea of the proposed algorithm, $BRST^{\frac{n^2-n}{2}}$, is similar to that of $BRST^{n-1}$ which is described in the previous section; Q_n is continuously divided until a subcube, Q_x which contains $x-1$ faulty links, is found. Let Q_x be the *starting subcube.* Once the starting subcube is found, then broadcasting in the subcube can be done using $BRST^{n-1}$. Note that since the total number of faulty links in Q_n can be at most $\frac{n^2-n}{2}$, it is guaranteed that there exists at least one Q_2 starting subcube which contains at most one faulty link (this is because Q_3 contains at most $\frac{3^2-3}{2} = 3$ faulty links, one of the Q_2's contains at most one faulty link). Broadcasting in the rest of the Q_n can be continued along the stored dimensions in

the $STACK$ as was done in $BRST^{n-1}$. However, there are some problems left to be solved in this case.

PROBLEM 1) Does there exist a dimension such that if a cube, say Q_m, is divided along it, one of the subcubes contains at least $m-1$ fewer faulty links than Q_m and at the same time the subcube is connected?

PROBLEM 2) The source node may not be in the starting subcube.

PROBLEM 3) In $BRST^{n-1}$, cubes are divided along fault-free dimensions. Therefore, the nodes not belonging to starting subcube will receive the message without any difficulty. However, in $BRST^{\frac{n^2-n}{2}}$ some of the dimensions along which cubes are divided may contain some faulty links. As a result, some of the nodes may not receive the message directly from the corresponding nodes in adjacent subcube. For example, at step 4 in Example 3.1, if some of the links connecting $10***$ and $00***$ are faulty, then some of the nodes in $00**$ may not receive the message directly from the nodes in $10**$.

We first prove the existence of the dimension which satisfies PROBLEM 1 above, i.e., we will prove the existence of a dimension such that if Q_m is partitioned along the dimension, at least one of following conditions is satisfied.

1) One subcube contains at most $m-2$ faulty links.

2) One subcube is connected, and it contains at most $\frac{m^2-m}{2} - (m-1) = \frac{(m-1)(m-2)}{2}$ faulty links.

Note that if 1) is satisfied, then broadcasting can be done in the subcube by using $BRST^{n-1}$, and broadcasting will be continued to the rest of Q_n along the stored dimensions. If 2) is satisfied, then the problem is reduced to a smaller problem. We keep dividing Q_n into smaller subcubes until a subcube, say Q_x which contains less than x faults, is found. Since, as mentioned before, there exists at least one Q_2 which contains at most one faulty link, condition 1 will be satisfied eventually. The following theorem shows the existence of the dimension which satisfies condition 2 above.

Theorem 3.1. Suppose Q_m, $m \leq n$, is connected and has at most $\frac{m^2-m}{2}$ faulty links. Then Q_m contains an $(m-1)$-dimensional subcube which is connected.

Proof : Since there are m dimensions in Q_m, Q_m can be divided into two subcubes along m different dimensions, which produces $2m$ different Q_{m-1}'s. For each dimension, the number of faulty links needed to disconnect the subcubes will be calculated. If the total number of faulty links needed to disconnect all the Q_{m-1}'s is greater than $\frac{m^2-m}{2}$, we get a contradiction and so the theorem proved. Without loss of generality, let us first divide Q_m along dimension 0. Then, in order to disconnect both subcubes, there should be some disconnected groups of nodes such as g_1, g_2 and g_3 as shown in Figure 3.2.a. We will consider only one group, say g_1, which will give the least number of faulty links. In the following, from the two subcubes resulting from each division only one subcube which needs less number of faulty links to be disconnected is considered. Thus, the total number of faulty links needed to disconnect only m subcubes will be obtained. Note that in order

to form g_1, there should be at least $m-1$ faulty links in Q_{m-1}. Since no node is disconnected in the network, at least one node, say a_1, in g_1 should be connected to a node, say a_2, in Q'_{m-1}. Now let us divide Q_m along dimension 1 as shown in Figure 3.2.b. Note that a_1 and a_2 are in the same subcube. Also note that it does not matter whether g_1 is in Q'_{m-1} or it is spread out in both subcubes. Now we will disconnect Q'_{m-1} instead of Q_{m-1} since the former takes less faulty links than the latter. That is, in order to disconnect Q'_{m-1}, there should be at least $m-2$ faulty links which are incident to node a_2, whereas there should be at least $m-1$ faulty links to disconnect Q_{m-1}. Note that node a_2 should be connected to a node, say a_4, in Q_{m-1}; if not, the nodes in g_1 and a_2 are disconnected in the network. Now let us divide the Q_m along dimension 2 as shown in Figure 3.2.c. Again, the nodes a_1 , a_2 and a_4 will be in the same subcube. By the same reason explained above in order to disconnect Q'_{m-1}, at least $m-2$ faulty links incident to node a_4 are needed. In general, for each dimension, at least $m-2$ faulty links are needed to disconnect a subcube, except dimension 0 which needs $m-1$ faulty links. Thus, a total of $(m-2)*(m-1)+(m-1)$ faulty links are needed. Since this number is greater than $\frac{m^2-m}{2}$, the theorem follows. □

We now discuss PROBLEM 2 - Source node may not be in the starting subcube. Recall that in the proof of Theorem 3.1, of the two subcubes resulting from a division only one subcube which needs fewer faulty links to be disconnected is considered. This implies that there exists a dimension such that if Q_m is divided along the dimension, at least one of the subcubes 1) contains the source node, 2)

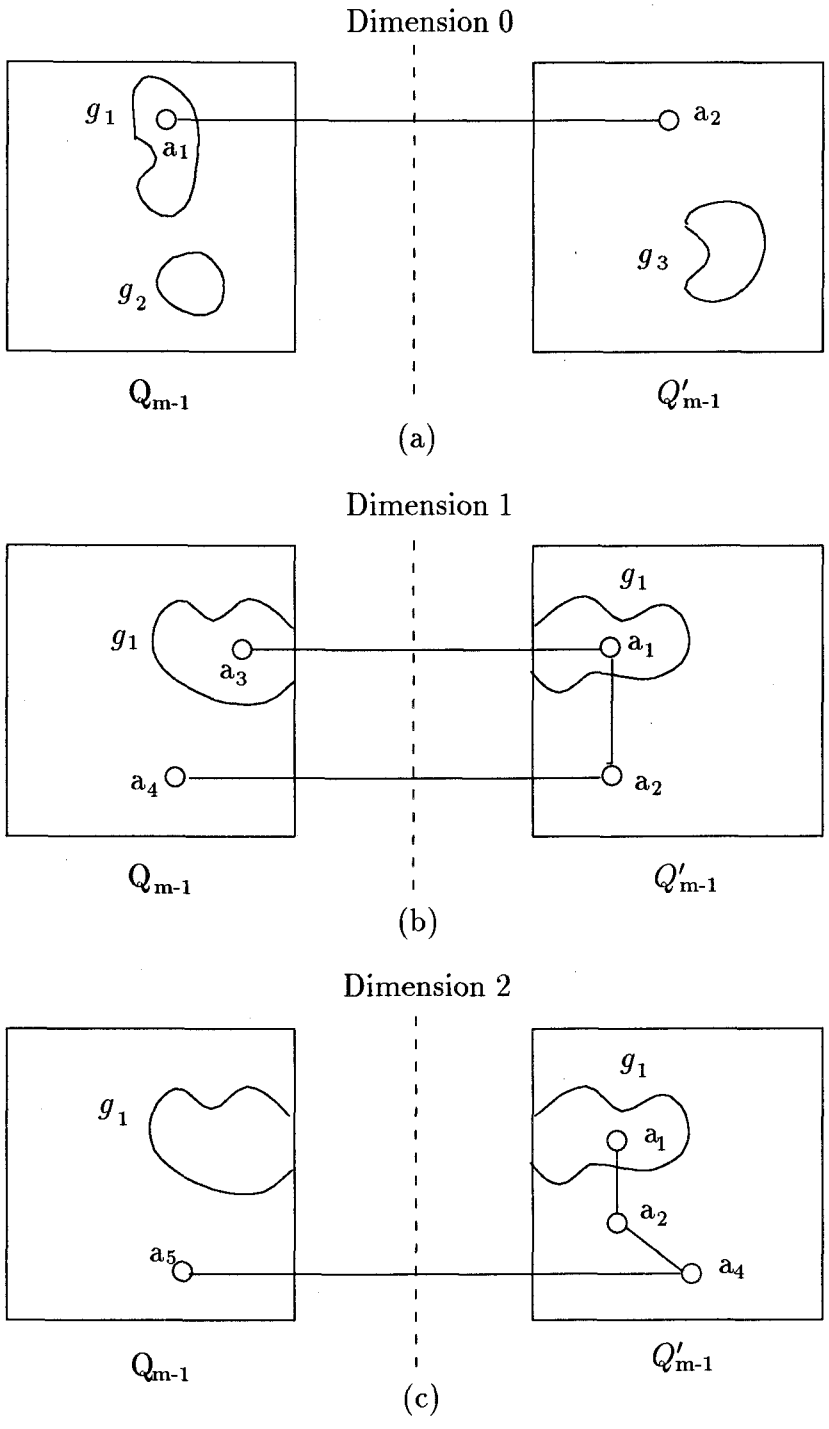

Figure 3.2. Division of Q_m along several dimensions. g_1 through g_3 denote the sets of nodes disconnected in the subcubes they belong.

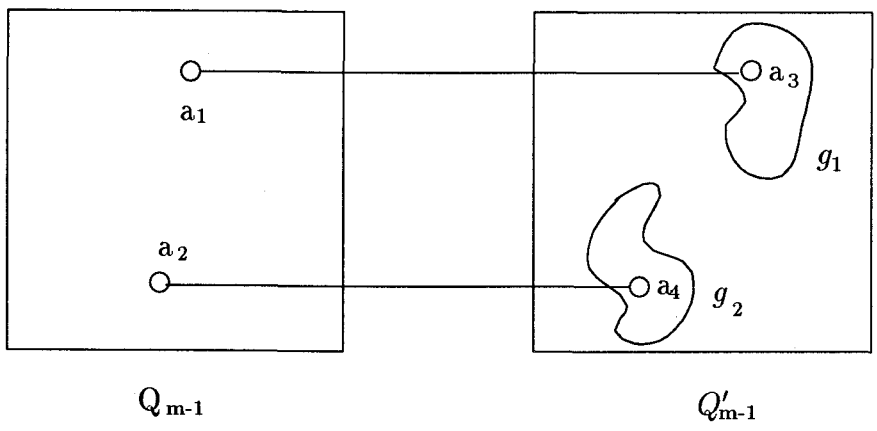

Figure 3.3. Division of Q_m into two subcubes. Every node in Q_{m-1} has broadcast message, and they are sending the message to corresponding nodes in Q'_{m-1}. Here some of the links connecting two subcubes may be faulty. g_1 and g_2 denote the sets of nodes disconnected in Q'_{m-1}.

contains at least $m-1$ fewer faults than Q_m, and 3) is connected.

PROBLEM 3 concerns the possibility that some of the dimensions along which cubes are divided may contain faulty links. As a result, some of the nodes may not receive the message directly from the corresponding nodes in the adjacent subcube. Refer to Figure 3.3. Suppose Q_m is divided along dimension d, and suppose all the nodes in Q_{m-1} already have the broadcast message and that they have sent the message to the corresponding nodes in Q'_{m-1}. However, since some of the links in dimension d may be faulty, some of the nodes in Q'_{m-1} may not receive the message. Let U be the set of nodes which did not receive the message. In this case, nodes which are in the same subcube as U try to send the message to nodes in U. This is possible since even if some groups of nodes, e.g., g_1 and g_2 in Figure 3.3, are disconnected in Q'_{m-1}, at least one node in each disconnected group should

Algorithm *COMPLETE*

P = set of nodes in Q'_{m-1} which do not receive the messages from corresponding nodes in Q_{m-1}. F = set of all the faulty links in Q'_{m-1}

while $P \neq \{\}$ **do begin**

calculate $\|F\|$

let k be the dimension such that $\|F\|(k) \leq \|F\|(i)$, for all $0 \leq i \leq m-1$, $i \neq k$

for all $p \in P$ **do**

if $(p \oplus e^k) \notin P$ **then begin**

$(p \oplus e^k)$ send the message to p

$P = P - p$

end

end

Figure 3.4. Algorithm *COMPLETE* which completes broadcasting in Q'_{m-1}.

be able to receive the message from the corresponding node in Q_{m-1}. We present a simple algorithm *COMPLETE* in Figure 3.4 which completes the broadcasting in Q'_{m-1}.

Now we give the complete algorithm $BRST^{\frac{n^2-n}{2}}$ in Figure 3.5 which broadcasts in an n-dimensional hypercube with up to $\frac{n^2-n}{2}$ faulty links.

Algorithm $BRST^{\frac{n^2-n}{2}}$

if number of faulty links in Q_n is less than n, **then** call $BRST^{n-1}$
else begin
  $STACK = \{\}$, $Q = ***...**$, $qsize = n$, $Dim = \{0, 1, \ldots, n-1\}$
  do begin
    $Found = false$
    for all $i \in Dim$ **do begin**
      Divide the Q along dimension i
      if one of the subcubes contains $qsize - 2$ or less faulty links

      then begin
        $Found = true$
        $FoundD = i$
        Push i into $STACK$
      end
      else if one of the subcubes contains $\leq \frac{(m-1)(m-2)}{2}$ faulty links
        and contains source node, and if it is connected **then**
        $FoundD = i$
    end
    if $Found = true$ **then begin**
      Complete broadcast in Q using $BRST^{n-1}$
      Nodes start to broadcast along the sequence of dimensions
        which will be obtained by popping the $STACK$.
        At the same time call $COMPLETE$ for the nodes which
        did not receive the message.
    end
    else begin
      $Q = Q_{qsize-1}$ which contains the source node.
      Push $FoundD$ into $STACK$
      $Dim = Dim - FoundD$
    end
  until $(Found = true)$
end

Figure 3.5. Algorithm $BRST^{\frac{n^2-n}{2}}$. It completes broadcasting in hypercubes with up to $\frac{n^2-n}{2}$ faulty links.

A linear array is a network topology which looks like a ring with one faulty link. The *length* of a linear array is the same as the number of links it contains. For example, in Figure 3.6 nodes 111...111 through 00011..111 with links only shown in the figure form a linear array of length three. Let us present the following proposition before we discuss the communication complexity of the $BRST^{\frac{n^2-n}{2}}$. In the following, we derive an approximate bound on the broadcasting time.

Proposition 3.2. Suppose Q_m contains $\frac{m^2-m}{2}$ faulty links. Then the number of time steps taken to complete the broadcasting in Q_m is at least $\frac{3m}{2}$.

Proof : We will give the fault diameter, i.e., the diameter of the network with the faulty links, of the Q_m, since the time steps taken by broadcasting cannot be smaller than the diameter of the network. Without loss of generality, let the source node be 000...000. Then node 111...111 is the farthest from the source node. In order to increase the diameter, we try to form the longest possible linear array starting at node 111...111 as follows. Refer to Figure 3.6. We make all incident links to node 111...111 faulty except the two links 111...111- and -111...111. Likewise, let all links incident to node 0111...111 be faulty except the two links -111...111 and 0-111...111, and so on. Then the maximum diameter of Q_m is $m + x$, where the value of $x \approx$ the length of the linear array and can be calculated from $(m-2)x \leq \frac{m^2-m}{2}$. Since the value of x is close to $\frac{m}{2}$, the diameter of the Q_m is roughly $\frac{3m}{2}$. □

Proposition 3.2 implies that with a given number of faulty links, the smaller the subcube the longer the linear array that can be formed. For example, if the

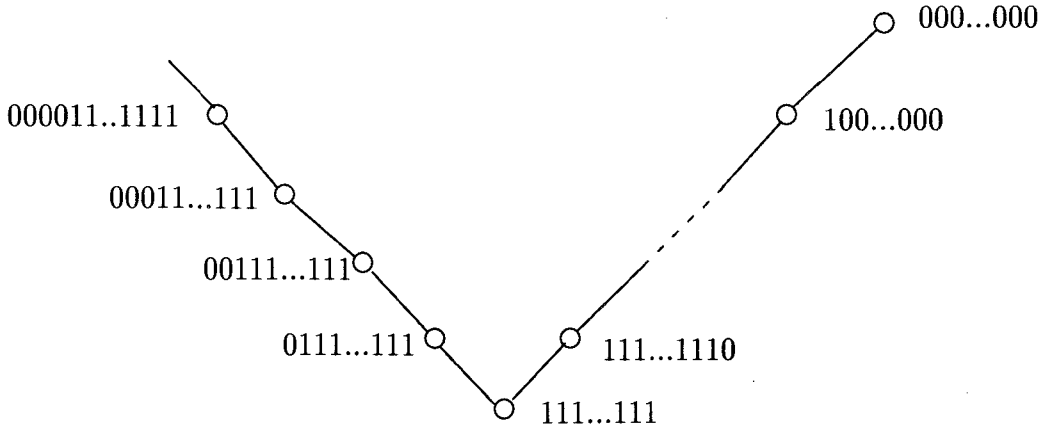

Figure 3.6. Generation of the longest possible fault diameter. Here m is the dimension of the cube and the number of faulty links allowed in Q_m is $\frac{m^2}{2} - m - 1$. Among all the links incident to nodes 111...111, 0111...111, 00111...11, 000111...111, ..., those links which are not shown in the figure are faulty.

number of tolerable faulty links is 20, then the length of the longest possible linear array in Q_{10} is two, whereas it is four in Q_6. Since Q_2 is the smallest subcube which can contain linear array, starting from Q_2 we assign as many faults as possible to subcubes $Q_i, i = 2, 3, \ldots$, until we run out of given faults, $\frac{n^2-n}{2}$. This will give the longest possible linear arrays in all subcubes. Refer to Figure 7. Since Q_m contains at most $\frac{m^2-m}{2}$ faulty links, Q_2 contains at most $\frac{2^2-2}{2}$ faulty links. However, Q_3 which consists of Q_2 and Q_2', contains up to $\frac{3^3-3}{2} - \frac{2^2-2}{2}$ faulty links in Q_2', which will be used to calculate the possible longest linear array in Q_3 since faults in Q_2 is already considered. In general, $Q_i, i \geq 3$, contains $f_i = \frac{i^2-i}{2} - \frac{(i-1)^2-(i-1)}{2}$ faulty links. Starting from Q_3, we will assign f_i faulty links for each Q_i until we run out of faulty links. Thus, we arrive at the following formula

$$\frac{2^2-2}{2} + (\frac{3^2-3}{2} - \frac{2^2-2}{2}) + (\frac{4^2-4}{2} - \frac{3^2-3}{2}) + \ldots + (\frac{r^2-r}{2} - \frac{(r-1)^2-(r-1)}{2}) \leq \frac{n^2-n}{2}$$

Above formula gives the value $r = n$. Since each term $\frac{i^2-i}{2} - \frac{(i-1)^2-(i-1)}{2}, 3 \leq i \leq n$, gives a linear array with length 1, the total length of the linear arrays, L_{total}, is

$$L_{total} = n - 1$$

Thus, the total time steps taken by the proposed algorithm will be $n + L_{total} \approx 2n$.

3.5 Conclusion

We have given a single node broadcasting algorithm which tolerates up to $\frac{n^2-n}{2}$ faulty links in an n-dimensional hypercube. The total time steps taken by the

algorithm does not exceed twice the dimension of the hypercube. Since the fault diameter of the network is approximately $\frac{3n}{2}$, the total time steps taken by the proposed algorithm is close to optimal.

Chapter 4

All-to-All Broadcasting in Faulty Hypercubes

4.1 Introduction

The hypercube has been studied extensively as an interconnection network topology for multicomputer systems[Sei85, SS88, BS86], and has led to numerous experimental and commercial machines [JH89] including the recent development of the system with more than 6000 nodes by NCUBE [DB92]. The hypercube contains many salient features such as regularity, symmetry, high fault-tolerance and structural recursiveness, and some have been explored [Sei85, SS88, BS86].

In distributed memory systems communication between the processors is mainly done via message passing. Since the communication time may be quite costly compared to the computation time, efficient communication schemes are extremely important to achieve the high performance in the system. Johnsson and Ho [JH89] introduce four different communication primitives, 1) *one-to-all broadcasting* (or *single node broadcasting*) in which a single node distributes a common data to all other nodes, 2) *one-to-all personalized communication* (or *scattering*)

Primitives	Time steps taken	No. of Transmissions
Routing	n	$n-1$
SNB	n	2^n-1
ATAB	$\lfloor \frac{2^n-1}{n} \rfloor$	$2^n(2^n-1)$
TX	2^{n-1}	$n2^{2n-1}$

Table 4.1. Optimal time steps and traffic for some communication primitives. Here the number of transmissions is used for the measurement of the traffic in the network. n-port communication is assumed. n is the dimension of the hypercube, and SNB, ATAB and TX stand for *single node broadcasting, all-to-all broadcasting* and *total exchange*, respectively.

in which a single node sends unique data to all other nodes, 3) *all-to-all broadcasting* (or *multinode broadcasting*) in which all nodes broadcast concurrently to all other nodes, and 4) *all-to-all personalized communication* (or *total exchange*) where each and every node sends a unique data to every other node. Many researchers have proposed various communication algorithms for hypercube multicomputers [JH89, SB77, HJ86, B+91, JH91], most of them concentrating on routing or one-to-all broadcasting. However, most of these communication schemes may not work properly in the presence of faulty components in the system. Numerous fault-tolerant communication algorithms have been proposed in [CS90c, CS90d, GS88, PB92, LH88, PB90c, CS89, CS88, RS88], again most of them concentrating on routing or one-to-all broadcasting.

Table 4.1 shows optimal times and number of packet transmissions for some basic communication primitives with *n-port assumption*, i.e., all the incident links to a node can be used simultaneously for packet transmission and reception. In general, the number of packet transmissions is used as a synonym to the *traffic* in the network which is quantified by the number of packet transmissions taken by an algorithm that solves the corresponding communication problem. Note that a factor of 2^n difference between single node broadcasting (SNB) and all-to-all broadcasting (ATAB) or total exchange (TX) in terms of both time and traffic. Thus, it is not difficult to imagine that inefficient algorithms for ATAB and TX may result very poor performance in the system.

In this chapter we introduce a simple and near optimal fault-tolerant all-to-all broadcasting algorithm in hypercube multicomputers in the presence of up to $\lfloor \frac{n}{2} \rfloor$ faulty links, where n is the dimension of the hypercube. Lee and Shin [LS90] have given some of the important applications of fault-tolerant all-to-all broadcasting – distributed agreement [LLP82], clock synchronization [LMS85], distributed diagnosis of intermittently faulty processors [YM88], etc. In these algorithms, each non-faulty node must be able to deliver its message to all other non-faulty nodes in the system. Both Lee and Shin (LS) [LS90] and Fraigniaud (FR) [Fra92] have proposed algorithms which achieve this under the assumption of non-availability of global fault information, i.e., each non-faulty node does not know the identities of faulty components. In both LS and FR algorithms each node delivers multiple copies of the broadcasting message through disjoint paths to all other nodes in the

system. On receiving the multiple copies of each message, each non-faulty node identifies the original message using some schemes such as majority voting. These algorithms have the advantage of not having to know the addresses of the faulty components, and therefore they may be suitable for the real-time applications.

However, since multiple copies of the same message cause much more traffic in the network, it may severely degrade the performance in the system, especially ones using *wormhole* [S+85, Dal87, Dal90] or *virtual cut-through* [KK79] routing as shown in [LS90]. Further, since the occurrence of the component faults is infrequent, it may be more efficient to broadcast the fault information by using some some fault-tolerant single node broadcasting algorithm such as in [RS88], so that each node contains the identities of the faulty components (note that the fault-tolerant single node broadcasting requires at most $2n$ time steps even when the algorithm does not require the global fault information) This allows that each node sends only one copy of the message, as proposed here, to complete all-to-all broadcasting. Therefore, if merging messages is not allowed, our algorithm produces approximately a factor of n less traffic than LS and FR algorithms.

Many fault-tolerant algorithms do not have the capability of utilizing algorithms developed for the non-faulty system; this forces the parallel systems to have two totally different algorithms, one for the faulty and the other for the non-faulty system (these will be referred to as *faulty* and *non-faulty algorithms*, respectively.) One of the advantages of the proposed algorithm is that it fully utilizes the non-faulty algorithm, and this non-faulty algorithm can be any existing one or any new

one yet to be developed.

The rest of the chapter is organized as follows. Section 4.2 summarizes the notation and definitions which will be used throughout the chapter. Section 4.3 introduces a new all-to-all broadcasting algorithm which can tolerate up to $\lfloor \frac{n}{2} \rfloor$ link faults. The conclusion follows in Section 4.4.

4.2 Preliminaries

As mentioned in Chapter 1, in hypercube topology two nodes are connected by a link iff their addresses differ by exactly one bit, and they are called the *end nodes* of the link. The relative address of two nodes, a and b, is bitwise *Exclusive-Or* operation of the two nodes $a \oplus b = c = (c_{n-1}, c_{n-2}, \ldots, c_0)$, where $c_i = a_i \oplus b_i$ for $i = 0, 1, \ldots, n-1$. e^i denotes a unit vector such that all the bits have value 0 except the i-th bit which has value 1. Thus if nodes a and b are adjacent to each other, then $a \oplus b = e^i$ for some i. It is said that link connecting the two nodes a and b is in dimension i, and this link is uniquely represented by $(a_{n-1}, a_{n-2}, \ldots, a_{i+1}, -, a_{i-1}, \ldots, a_0)$, where the address of the node a is $(a_{n-1}, a_{n-2}, \ldots, a_0)$. For example, the link connecting nodes 01010 and 01011 is denoted as 0101- and is in dimension 0. The relative address of the two links l and m is also bitwise Exclusive-Or of their addresses, $l \oplus m$, where $l_i \oplus m_i = 1$ iff $l_i = 0$ (respectively, 1) and $m_i = 1$ (respectively, 0); $l_i \oplus m_i = 0$, otherwise.

The *weight* of a node or link r is the number of 1's in r. The *distance* between two nodes a and b (or two links l and m) is given by $W(a \oplus b)$ (or $W(l \oplus m)$). Let

S be the set of nodes or links. Then $|S|$ denotes the cardinality of S.

Any subcube Q_x in Q_n, $x \leq n$, can be uniquely represented by a sequence of n ternary symbols $(t_{n-1}, t_{n-2}, \ldots, t_0)$, $t_i \in \{0, 1, *\}$, $0 \leq i \leq n-1$, where $*$ is a *don't care* symbol. If Q_n is divided (or *partitioned*) into two subcubes along dimension d, the addresses of the two resulting Q_{n-1} subcubes are $***\ldots 1_d **\ldots **$ and $***\ldots 0_d **\ldots **$. The dimensions which contain $*$ are called *don't care dimensions* and *non-don't care*, otherwise. For example, if Q_4 is divided along 1st dimension, the resulting two subcubes are $**0*$ and $**1*$. Dimensions 0, 2 and 3 are don't care dimensions and dimension 1 is a non-don't care dimension. *Faulty* (*Perfect*) subcube is the one which contains some (no) faulty components.

If Q_n is divided along k dimensions, $d_1, d_2, \ldots, d_k$, then there will be 2^k $(n-k)$-dimensional subcubes. A *partner set (PS)* denotes a set of nodes obtained by giving the same value for the $*$'s in each subcube. There are 2^{n-k} PS's each of which contains 2^k nodes, and each PS forms a k-dimensional cube. *Corresponding nodes* are a pair of nodes adjacent to each other in the same *PS*. Corresponding nodes *along dimension d* are the corresponding nodes which differ in d-th bit. Likewise, *corresponding links* are a pair of links such that they are in the same dimension and their addresses differ in only one of the non-don't care dimensions, and if they differ in d-th dimension, then they are called the corresponding links *along dimension d.* The dimension d is also referred to as a *corresponding dimension.* Links belonging to PS's are called *intersubcubal* and *intrasubcubal* otherwise.

Example 4.1. If Q_5 is divided along three dimensions, 0, 1, 2, then there are eight 2-dimensional subcubes, $**000$, $**001$, $**010$, $**011$, $**100$, $**101$, $**110$, $**111$. Nodes 01000, 01001, 01010, 01011, 01100, 01101, 01110, 01111 form one of the 2^2 PS's since all the $*$'s in each subcube have the same value, 01. Each PS forms a 3-dimensional cube. Nodes 01000 and 01001 are the corresponding nodes since they are in the same PS and differ in only one bit. Also they are corresponding nodes along dimension 0. Two links $0-000$ and $0-001$ form corresponding links along dimension 0. □

In all-to-all broadcasting, each node has a message to broadcast to all other nodes. Let M_x denote the set of messages initially belonging to the nodes in subcube Q_x.

Communication algorithms can be implemented in either *one*-port or n-port model. Algorithms designed for a one-port model are simpler than the ones designed for an n-port model [B+91] and are not considered here.

As mentioned earlier, the proposed algorithm fully utilizes any non-faulty algorithm employed by the system. Thus, we will choose any one of the optimal non-faulty algorithms, for example shown in [B+91], and refer to it as $ATAB^p$ throughout the chapter. Since it is optimal, it would take $\frac{2^n-1}{n}$ time steps in a perfect Q_n.

4.3 New all-to-all broadcasting algorithm in faulty hypercubes

We now present a fault-tolerant all-to-all broadcasting algorithm in hypercubes with up to $\lfloor \frac{n}{2} \rfloor$ faulty links. This section starts with a simple case and goes through different cases of link failures, which will help in understanding the general algorithm presented in Section 4.3.4.

4.3.1 Case of a single link failure

First, let us start with a case where there is only one faulty link in Q_n. Even though the case is very simple, it presents the basic and the most important idea presented in this chapter.

Let f be the dimension of the faulty link. If we divide Q_n into two subcubes, Q_{n-1} and Q'_{n-1}, along dimension d, $d \neq f$, one of the subcubes contains no faulty link. Without loss of generality, let us assume that Q'_{n-1} contains the faulty link connecting nodes a' and b' as shown in Figure 4.1. Let M_{n-1} and M'_{n-1} be the sets of messages initially belonging to the nodes in Q_{n-1} and Q'_{n-1}, respectively.

At the first step of the proposed algorithm, every node in Q_n sends its message along dimension d. After this, both subcubes contain M_{n-1} as well as M'_{n-1}, which allows each subcube to perform all-to-all broadcasting independently using $ATAB^p$. We will describe how all-to-all broadcasting can be done in Q_n first with M_{n-1}. In the new algorithm, step i of $ATAB^p$ is done in two steps, $(2i-1)$ and

$(2i)$, for $i = 1, 2, 3, \ldots$. At the $(2i-1)$-th step both subcubes execute i-th step of the $ATAB^p$, except nodes a' and b' can not exchange the messages between them. In the $(2i)$-th step nodes a and b will send to nodes a' and b', respectively, the messages they received from each other in the previous step (this action will be referred to as *intersubcubal transmission*). Note that the intersubcubal transmissions have the same effect as the messages being exchanged between nodes a' and b'. Thus all-to-all broadcasting with M_{n-1} can be done in both Q_{n-1} and Q'_{n-1} in $2\frac{2^{n-1}-1}{n-1} + 1 = \frac{2^n-2}{n-1} + 1$ steps. However, note that when a and b send the messages to a' and b' during $(2i)$-th steps along d-th dimension, the intrasubcubal links, i.e., the links in Q_{n-1} and Q'_{n-1} are idle; further at $(2i-1)$-th steps, $i = 2, 3, \ldots$, the intersubcubal links are idle. In order to achieve more efficient link utilization, in step $2i$ both Q_{n-1} and Q'_{n-1} execute the i-th step of $ATAB^p$ with M'_{n-1}, and at time step $(2i+1)$, a and b will send to a' and b', respectively, the messages they received from each other at time step $2i$, $i = 1, 2, \ldots$. Thus using this '*interleaving scheme*', the links in the network are fully utilized. The total time needed to complete the all-to-all broadcasting in this case is $\frac{2^n-2}{n-1} + 2$. The complete algorithm is given in Figure 4.2 as *Algorithm1*.

It is straightforward to verify that Algorithm1 is correct. We now consider the optimality of the algorithm.

Lemma 4.1. It takes at least $\frac{2^n-1}{n-f}$ time steps to complete all-to-all broadcasting if

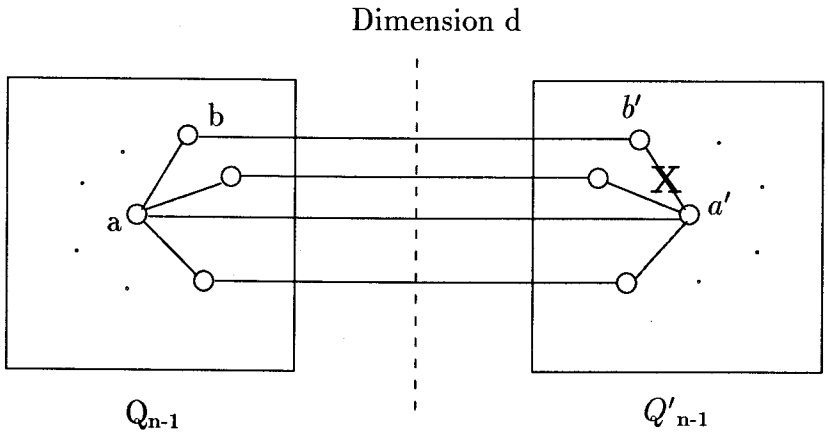

Figure 4.1. Q_n with one faulty link. Link connecting nodes a' and b' is faulty.

the maximum number of faulty links incident to a node is f.

Proof : Each node has to receive $2^n - 1$ different messages. Since one node has $n - f$ non-faulty links incident to it, the node can receive at most $n - f$ messages at one time unit. Thus the lemma follows. □

Algorithm1 takes at most $\frac{2^n-1}{n-1} + 2$, which is close to the lower bound shown in Lemma 4.1.

4.3.2 Case when no node has more than one faulty link incident to it

Note that *Algorithm1* can also be applied to the case when 1) each node has no more than one faulty link incident to it, and 2) all the faulty links reside in one subcube. In this section it is shown that even after dropping the second condition,

Algorithm1

Let the link connecting nodes a' and b' be faulty as shown in Figure 4.1.

1) Divide Q_n into two subcubes, Q_{n-1} and Q'_{n-1}, along dimension d, $d \neq f$, where f is the dimension of the faulty link.

2) At time step 0, every node sends its message along dimension d.

3) Both Q_{n-1} and Q'_{n-1} execute the i-th step of $ATAB^p$ with M_{n-1} at time step $(2i-1)$ and with M'_{n-1} at time step $2i$, $i = 1, 2, \ldots$, except nodes a' and b' can not exchange any message between them.

4) At time steps i, $i = 2, 3, 4, \ldots$, nodes a and b, where $a = a' \oplus e^d$ and $b = b' \oplus e^d$, send to nodes a' and b', respectively, the messages they received from each other at time step $i-1$.

Figure 4.2. Algorithm1 which completes all-to-all broadcasting in hypercubes with one faulty links.

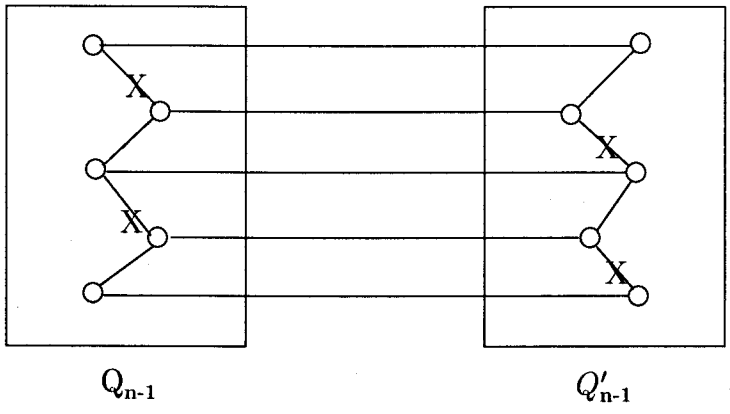

Figure 4.3. Faulty Q_n in which no node has more than one faulty link incident to it. Lines with X indicate faulty.

all-to-all broadcasting can be done in near optimal time. Refer to Figure 4.3.

Before giving the algorithm, some terminology is explained.

A dimension is *faulty-free* if there is no faulty link in that dimension. A dimension is *unsafe* if, when Q_n is partitioned into Q_{n-1} and Q'_{n-1} along dimension d, there exist faulty links l in Q_{n-1} and l' in Q'_{n-1} that differ only in d-th bit. If no such faulty links exist, then dimension d is called *safe*. For example, if the set of faulty links $F = \{0{-}00, 1{-}00, 00{-}1\}$, then dimension 3 is unsafe because of the links $\{0{-}00, 1{-}00\}$. All the other dimensions are safe. Note that if a dimension is unsafe, then there exists two faulty links l and l' such that $W(l \oplus l') = 1$. The converse is not necessarily true. For example, if the set of faulty links $F = \{0-00, 1-00, 00-1\}$ as before, even though $W(0-00 \oplus 00-1) = 1$, dimension 0 is safe by our definition.

A dimension is *fault-free safe* if it is fault-free and also safe. Dimension 0 in

the above example is fault-free safe. Note that if there is a fault-free safe dimension in Q_n, *Algorithm1* can be directly applied for all-to-all broadcasting. We show below that even in the presence of $n-1$ faulty links, there exists at least one fault-free safe dimension in Q_n.

Let $F = \{f_1, f_2, \ldots, f_k\}$ be the set of faulty links and let all these faulty links be in the same dimension. In this case, note that a dimension d is unsafe if there exist two faulty links f_i and f_j such that they differ only in the d-th bit. Now we can get another set $S = \{f_i \oplus f_j \mid f_i, f_j \in F\}$ and find out the number of distinct weight 1 vectors in S. This gives the list of the unsafe dimensions. For example, let $F = \{00011-, 00010-, 00111-, 00110-\}$. Then $S = \{00001-, 00100-, 00101-\}$. Therefore, there are two unsafe dimensions, which are 1 and 3; the other dimensions are safe. Some concepts from vector space are needed to prove the main results and are explained below.

The set of all binary n-tuples can be considered as a vector space V over $GF(2) = \{0, 1\}$. A set of t binary vectors $\{A_1, A_2, \ldots, A_t\}$ are linearly independent if no vector can be expressed as a linear combination of the other vectors. For example, the four vectors {0001, 0010, 0100, 1000} are linearly independent. The number of linearly independent vectors in the set S is referred to as the dimension of S, or $dim(S)$.

Lemma 4.2. If there are k faulty links in Q_n, and if all of them are in the same dimension, then there exist at least $n-k$ fault-free safe dimensions in Q_n.

Proof: Let the set of faulty links $F = \{f_1, f_2, \ldots, f_k\}$ be the set of non-zero vectors over $GF(2) = \{0, 1\}$. Consider the set $S = \{f_i \oplus f_j \mid i, j = 1, 2, \ldots, k\}$. We need to prove that the number of distinct weight 1 vectors in $S \leq k - 1$. However, we will prove the stronger result that the $dim(S) \leq k - 1$. $dim(S) \leq dim(F)$ since any vector v which is a linear combination of the vectors in S is also a linear combination of the vectors in F. Thus if $dim(F) \leq k - 1$, then the theorem is true. Suppose that $dim(F) = k$. This means all the vectors in F are linearly independent. In this case it is easy to show that none of the f_i's in F is a linear combination of the vectors in S. Thus $dim(S) < dim(F) = k$, i.e., $dim(S) \leq k - 1$. Since there is only one faulty dimension, the total number of fault-free safe dimensions is greater than or equal to $n - k$. □

Theorem 4.3. If k_i is the number of faulty links in dimension $i, 0 \leq i \leq n - 1$, in Q_n such that $\sum_{i=0}^{n-1} k_i = k \leq n - 1$, then there exist at least $n - k$ fault-free safe dimensions in Q_n.

Proof: This follows both from Lemma 4.2 and from the fact that no two links in different dimensions can make any dimension unsafe. □

The algorithm *Find_Safe_D* in Figure 4.4 determines the set of all safe dimensions. As it has been mentioned earlier, even in the presence of up to $n - 1$ faulty links, at least one fault-free safe dimension can be found. If Q_n is divided along the fault-free safe dimension, say s, then there is no pair of corresponding links in which both links are faulty. This allows intersubcubal transmissions be possible between

Find_Safe_D (F, Safe_D)

Input - $F = \{f_1, f_2, \ldots, f_k\}$: list of the faulty links in Q_n.

Output - *Safe_D* : list of safe dimensions.

$Unsafe = \{\}$

for $i = 1$ **to** k **do**

  for $j = i + 1$ **to** k **do**

    if (f_i and f_j are in the same dimension) **and** ($f_i \oplus f_j = e^d$ for some d)

    then

      $Unsafe = Unsafe \cup \{d\}$

$Safe\_D = \{0, 1, 2, \ldots, n-1\} - Unsafe$

Figure 4.4. Algorithm *Find_Safe_D* which finds all the safe dimensions.

any pair of corresponding links, since the messages can be exchanged along at least one of the two corresponding links. Thus, once we find a fault-free safe dimension, *Algorithm1* with minor modifications can be used for all-to-all broadcasting in this case. The complete algorithm is given in Figure 4.5 as *Algorithm2*. The time taken for this algorithm is $\frac{2^n-1}{n-1} + 2$, which is close to optimal.

Algorithm2

$F = \{f_1, f_2, \ldots, f_k\}$: list of faulty links

$D = \{d_1, d_2, \ldots, d_i\}$: list of fault-free dimensions, where $i \geq k$

Let a'_j and b'_j be the end nodes of faulty link f_j

1) Call *Find_Safe_D* to find the safe dimensions, *Safe_D*.

2) Divide the Q_n into two subcubes, Q_{n-1} and Q'_{n-1}, along dimension $d \in (Safe\_D \cap D)$.

3) At time step 0, every node sends its message along dimension d.

4) Both Q_{n-1} and Q'_{n-1} execute the i-th step of $ATAB^p$ with M_{n-1} at time steps $(2i-1)$ and with M'_{n-1} at time steps $2i$, $i = 1, 2, \ldots$, except the end nodes of all faulty links can not exchange any message.

5) At time steps i, $i = 2, 3, \ldots$, for all faulty links $f_j \in F$, nodes a_j and b_j, where $a_j = a'_j \oplus e^d$ and $b_j = b'_j \oplus e^d$, send to a'_j and b'_j, respectively, along the dimension d the messages they received from each other at time step $i-1$.

Figure 4.5. *Algorithm2* completes all-to-all broadcasting in Q_n in which no node has more than one faulty link incident to it.

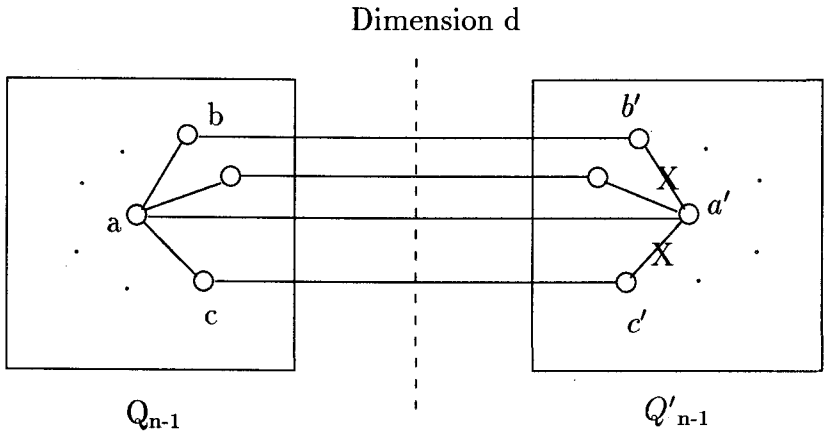

Figure 4.6. Case when all faulty links are incident to a single node. Here, node a' has two faulty links, (a', b') and (a', c'), incident to it.

4.3.3 Case when all the faulty links are incident to a single node

Let $F = \{f_1, f_2, \ldots, f_k\}$, $k \leq \lfloor \frac{n}{2} \rfloor$, be the set of faulty links incident to a single node. Figure 4.6 gives an example where two faulty links are incident to node a'. In order for an algorithm similar to *Algorithm2* to work in this case, a' must be able to receive two messages from a during each intersubcubal transmission step, one from node b and one from node c, because node a' has two faulty links, (a', b') and (a', c'), incident to it. In this case the time complexity of this algorithm can be twice that of *Algorithm2*. In general, when there are k faulty links incident to a single node, an algorithm similar to *Algorithm2* can be a factor of k slower than *Algorithm2*.

In order to overcome this situation, first, Q_n is partitioned into $(n-k)$-dimensional subcubes along some k fault-free safe dimensions $D = \{d_1, d_2, \ldots, d_k\}$. Note that, since the number of faulty links is $k \leq \lfloor \frac{n}{2} \rfloor$, there exist at least $(n-k) \geq k$ fault-free safe dimensions by Theorem 4.3. In addition, note that all nodes within a subcube will have the same values for the address bits $d_i, i = 1, 2, \ldots, k$, and we refer this subcube by $Q_{d_1 d_2 \ldots d_k}$. For example, Figure 4.7 shows the four subcubes Q_{00}, Q_{01}, Q_{10} and Q_{11} resulting from the division of Q_n along any two fault-free safe dimensions. Q_n is divided along two dimensions since node a' contains two faulty links incident to it.

This set of subcubes $\{Q_{d_1 d_2 \ldots d_k} \mid d_i \in \{0,1\}\}$ has the following properties.

(1) The set of 2^k nodes, one from each subcube and whose values in don't care dimensions are same, form a k-dimensional subcube. Recall that these nodes are called *partner set (PS)*. For example in Figure 4.7, nodes a, a', a'', and a''' form a PS which is Q_2. There are 2^{n-k} k-dimensional PS's; further, all these subcubes (PS's) are perfect since the partitions are done over the fault-free safe dimensions.

(2) Only one subcube contains all the faulty links, since Q_n is divided along fault-free dimensions. Further, the node which has all the faulty links incident to it has $n-k$ adjacent nodes, and of which $n-2k$ are in the same subcube and k are in k different perfect subcubes. For example in Figure 4.7, adjacent to node a', there are two nodes a and a'' which are in different perfect subcubes. Thus, for each faulty link, f_i, we may assign a unique dimension, d_i, such that the corresponding link of f_i along dimension d_i, i.e., $f_i \oplus e^{d_i}$, is fault-free and also is in a *distinct* perfect

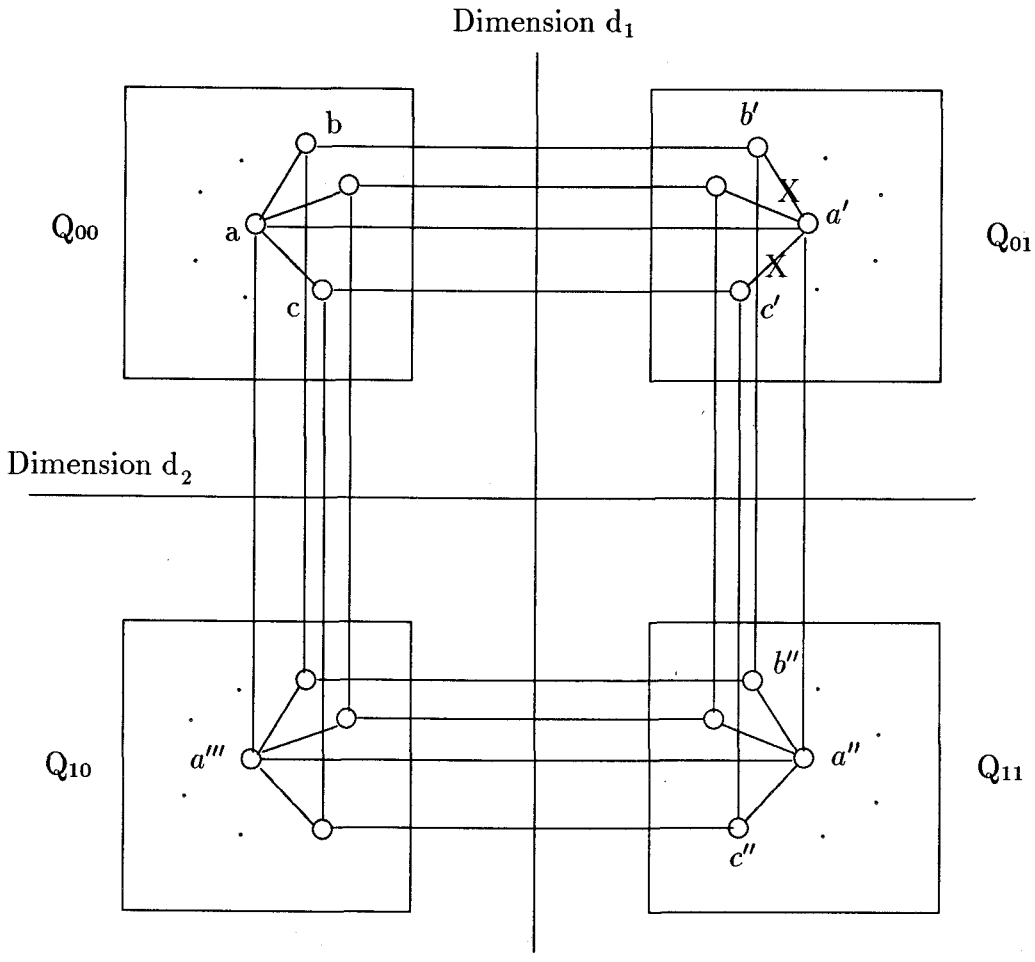

Figure 4.7. Case when all faulty links are incident to a single node. Link with X indicates faulty. Here node a' has two faulty links incident to it.

subcube.

By Property (2) above, node which has all k faulty links incident to it can receive the k messages simultaneously from its corresponding nodes one from each of its adjacent perfect subcube. Thus intersubcubal transmission in this case can be achieved. Let $Q_{d_1 d_2 \ldots d_k}$ initially contain a message set $M_{d_1 d_2 \ldots d_k}$. In the proposed algorithm, first, every node executes $ATAB^p$ within the PS it belongs to. Since every PS forms a perfect k-dimensional subcube and all PS's are edge-disjoint, i.e., no link appears more than once in all PS's, all-to-all broadcasting in all PS's can be done in $\frac{2^k-1}{k}$ steps using $ATAB^p$. After this, every subcube contains all the message sets, $M_0, M_1, \ldots, M_{2^k-1}$, with every node in Q_n containing 2^k messages. Next, $ATAB^p$ in each subcube and intersubcubal transmissions, if necessary, will be done. Let the link $f_i \oplus e^{d_i}$ be in charge of intersubcubal transmission for the faulty link f_i, for all $f_i \in F, d_i \in D$. Then the faulty subcube and each of its adjacent subcubes form a $(n-k+1)$-dimensional subcube which logically contains only one faulty link. For example in Figure 4.7, subcubes Q_{00} and Q_{01} form a $(n-k+1)$-dimensional subcube which contains only one faulty link (a', b') and subcubes Q_{01} and Q_{11} contains (a', c'). This enables us to use *Algorithm2* within the logical Q_{n-k+1}'s.

In *Algorithm2* there are only two sets of messages, M_{n-1} and M'_{n-1}. However, there are 2^k sets of messages in this case since there are 2^k subcubes after the division of Q_n along k dimensions. Thus, the following *exhaustive ordering* scheme is adopted for handling the 2^k message sets. Let $m = 2^k$ and assume that $ATAB^p$ in each PS

is completed at time step t. At time step $t+i, i = 1, 2, \ldots$, every subcube executes $((i \ div \ (m+1)) + 1)$-th step of $ATAB^p$ within itself with message set M_{i-1}, while every logical Q_{n-k+1}'s does intersubcubal transmissions, if necessary. *Algorithm3* in Figure 4.8 illustrates the process in detail.

Now let us consider the time complexity of the algorithm in this case. $ATAB^p$ in PS's takes $\frac{2^k-1}{k}$ steps. Execution of $ATAB^p$ in subcubes and intersubcubal transmissions would take $2^k\left(\frac{2^{n-k}-1}{n-k}\right) + 1$ time steps. Thus the total time steps taken would be $T = \frac{2^k-1}{k} + 2^k\left(\frac{2^{n-k}-1}{n-k}\right) + 1$. By Lemma 4.1 the lower bound would be $T_o = \frac{2^n-1}{n-k}$. Thus, $T - T_o \approx 2^k\left(\frac{1}{k} - \frac{1}{n-k}\right)$ denotes the difference in time steps taken by *Algorithm3* and the lower bound. Table 4.2 shows $T - T_o$ for some values of n. In the table, the values for the number of faulty links, k, is chosen such that the value $T - T_o$ is maximum. From these values of $T - T_o$, it can be seen that the proposed algorithm is close to optimal.

The total time steps taken by the algorithms presented so far can be expressed as

$$T_{total} = T_{ps} + T_{sc} + 1$$

where T_{ps} is the number of time steps taken to complete all-to-all broadcasting within PS's, T_{sc} is the number of time steps taken by all-to-all broadcasting in subcubes. Note that T_{sc} is the dominant factor in the above formula.

Algorithm3

$F = \{f_1, f_2, \ldots, f_k\}$: list of faulty links in Q_n

$D = \{d_1, d_2, \ldots, d_k\}$: list of k fault-free dimensions

1) Divide Q_n along every dimension in D.

2) At time step 0, all PS's start $ATAB^p$ which would take $\left(\frac{2^k-1}{k}\right)$ time steps to complete all-to-all broadcasting in all PS's.

Let $t_0 = \left(\frac{2^k-1}{k}\right)$, $m = 2^k$ and $j = 0$

3) At time step t_0, all subcubes start executing $ATAB^p$ with M_0.

while (all-to-all broadcasting in Q_n is not completed) **do begin**

At time step $t_0 + j$, all subcubes execute $((j \ div \ (m+1)) + 1)$-th step of $ATAB^p$ with $M_{j \ mod \ m}$.

At the same time for all faulty links $f_i \in F$, end nodes of the link $f_i \oplus e^{d_i}$ send along the dimension d_i the messages which were exchanged , if any, at time $t_0 + j - 1$ along the link $f_i \oplus e^{d_i}$.

$j = j + 1$

end

Figure 4.8. Algorithm3 completes all-to-all broadcasting when all faulty links are incident to a single node.

n	k	T_o	$T - T_o$
5	1	8	2
10	1	114	2
15	1	3641	3
20	3	87382	4

Table 4.2. Comparison of the time steps taken by Algorithm3 and the lower bound. n is the dimension of the hypercube and k is the number of the faulty links which gives maximum value of $T - T_o$, where T_o is the lower bound and T is the time steps taken by Algorithm3.

4.3.4 General case of link failures

We are now ready for the general case in which up to $\lfloor \frac{n}{2} \rfloor$ faulty links can occur in Q_n in any pattern. First, choose a node, say p, which has the maximum number of faulty links incident to it among all nodes, and let k be the number of faulty links incident to p. From Lemma 4.1, it can be seen that node p is the bottleneck , i.e., all-to-all broadcasting can be slowed down most by node p since it has the minimum number of non-faulty links incident to it among all nodes. As explained in the previous section, first, k fault-free safe dimensions are chosen and these dimensions are used to divide Q_n. The existence of such k fault-free safe dimensions follows from Theorem 4.3. Since these k dimensions are fault-free, $ATAB^p$ in the PS's can be done without any difficulty.

After the completion of $ATAB^p$ in each PS, $ATAB^p$ within each subcube

and intersubcubal transmissions, if necessary, will be done next in the proposed algorithm. However, for intersubcubal transmission, it has to be decided which non-faulty links is in charge for the given faulty link. In Section 4.3.3, all faulty links are incident to a single node; thus they are confined in a single subcube and *are in different dimensions.* That allows a link $f_i \oplus e^{d_i}$ is in charge of the faulty link f_i, for all $f_i \in F, d_i \in D$, where F is the set of faulty links and $D = \{d_1, d_2, \ldots, d_k\}$ is a subset of the fault-free safe dimension set; each d_i serves as a *corresponding dimension* for a distinct faulty link f_i, $i = 1, 2, \ldots, k$. However, in general case, one dimension might be used as a corresponding dimension for more than one faulty link which may be in different dimensions. For example, let $F = \{0000000-, 000000-0, 00110-00, 0011-000\}$. Then, node 00000000 has the most faulty links incident to it, and here $k = 2$. Without loss of generality, let dimensions 6 and 7 be chosen to divide this Q_8. Since all links in F are in the same subcube (which is just coincidence), dimensions 6 and 7 should be used as the corresponding dimensions for more than one faulty links. In this example, dimension 6 might be a corresponding dimension for faulty links 0000000− and $00110-00$, and dimension 7 for faulty links $000000-0$ and $0011-000$. Algorithm *Correspondence* in Figure 4.9 shows how to couple faulty links with one of the fault-free safe dimensions. The basis for this algorithm is explained below.

Two links are said to be *adjacent* if they are incident to the same node. The distance between any two links l and m is denoted as

$$W(l \oplus m) = \sum_{i=0}^{n-1} l_i \oplus m_i, \text{ where } l_i \oplus m_i = 1 \text{ iff } l_i \neq -, m_i \neq -, \text{ and } l_i \neq m_i.$$

Note that two links are adjacent iff the distance between them is zero. Further, every pair of adjacent faulty links are in the same subcube and are in different dimensions. Any two non-adjacent faulty links which are in any dimensions, can have their corresponding links along the same fault-free safe dimension. Since any faulty link has at most $k-1$ adjacent faulty links, k fault-free safe dimensions should be enough to couple all faulty links with corresponding non-faulty links, even though the number of faulty links might be greater than k. Incorporating the above observations, *Correspondence* picks any faulty link, say f, and finds a set of faulty links, F', in which all links are at a distance one or more apart from link f, and assigns a fault-free safe dimension to the links in $F' \cup \{f\}$. With the set of faulty links $F - (F' \cup \{f\})$, *Correspondence* repeats the process described above until $F = \{\}$.

Once the corresponding links of all faulty links are found, the intersubcubal transmissions can be done according to it, and the rest of all-to-all broadcasting steps will be the same as in *Algorithm3*. *Algorithm4* in Figure 4.10 completes all-to-all broadcasting in Q_n with up to $\lfloor \frac{n}{2} \rfloor$ faulty links.

It is straightforward to see that the time complexity of *Algorithm4* is the same as that of *Algorithm3* described in previous section. Example 4.2 illustrates *Algorithm4*.

Example 4.2. Let the dimension of the cube n be 6 and the list of faulty links $F = \{00000-, 0000-0, 0010-0\}$. Since node 000000 has the maximum number of faulty links incident to it, $k = 2$. In order to get the list of safe dimensions, *Safe_D*,

Correspondence

$F = \{f_1, f_2, \ldots, f_t\}$: list of faulty links in Q_n.
$D = \{d_1, d_2, \ldots, d_r\}$: list of fault-free safe dimensions in Q_n.
k = maximum number of faulty links incident to a node.
Call $Find\_Safe\_D$ to get the list of safe dimensions $Safe\_D$.
FFS_D = set of arbitrary k fault-free safe dimensions from $D \cap Safe\_D$.

while $F \neq \{\}$ **do begin**

$F' = \{\}$

Choose first element, say f', from F

$F' = F' \cup \{f'\}$

$i = 2$

while $i \leq \mid F \mid$ **do begin**

Choose i-th element, say f^*, from F

if $(W(f^* \oplus f_j) \geq 1$, for all $f_j \in F')$ **then**

$F' = F' \cup \{f^*\}$

$i = i + 1$

end

$F = F - F'$

Generate tuples (f_j, d), for all $f_j \in F', d =$ first element in $Safe\_D$

$Safe\_D = Safe\_D - \{d\}$

end

Figure 4.9. *Correspondence* assigns fault-free safe dimensions to faulty links. This will be used for intersubcubal transmissions.

Algorithm4

$F = \{f_1, f_2, \ldots, f_t\}$: list of faulty links in Q_n

FF_D = list of all fault-free dimensions

k = maximum number of faulty links incident to a node

Call *Find_Safe_D* to get the list of all safe dimensions, *Safe_D*.

S = set of arbitrary k fault-free safe dimensions from $FF\_D \cap Safe\_D$

Call *Correspondence* to calculate tuples (f_i, d_h), for all $f_i \in F, d_h \in S$

At time step 0, each PS starts $ATAB^p$ which would take $\frac{2^k-1}{k}$ time steps.

Let $t_0 = \frac{2^k-1}{k}, m = 2^k$, and $j = 0$

while (all-to-all broadcasting in Q_n is not completed) **do begin**

At time step $t_0 + j$, all subcubes execute $((j \ div \ (m+1)) + 1)$-th step of $ATAB^p$ with $M_{j \ mod \ m}$.

At the same time for all faulty links $f_i \in F$, end nodes of the link $f_i \oplus e^{d_h}$ send along dimension d_h the messages which were exchanged, if any, at time $t_0 + j - 1$ along the link $f_i \oplus e^{d_h}$.

$j = j + 1$

end

Figure 4.10. *Algorithm4* completes all-to-all broadcasting in Q_n with up to $\lfloor \frac{n}{2} \rfloor$ faulty links.

call *Find_Safe_D*, which returns $Safe\_D = \{0, 1, 2, 4, 5\}$. The set of fault-free dimensions, $D = \{2, 3, 4, 5\}$, and thus the set of fault-free safe dimensions, $S = \{2, 4, 5\}$. Next, choose any S' from S, where $|S'| = k$, and call *Correspondence* to couple them with faulty links in F such that for every $(f_i, s_j), s_j \in S'$, not both f_i and corresponding link along s_j, $f_i \oplus e^{s_j}$, are faulty for all $f_i \in F$. Without loss of generality, let S' be the least two dimensions from S, i.e., $S' = \{2, 4\}$. Then output from *Correspondence* will be $C = \{(f_i, s_j)\} = \{(00000-, 2), (0010 - 0, 2), (0000 - 0, 4)\}$. Thus, for example, nodes 000000 and 000001, which are the end nodes of link 00000−, will receive the messages from 000100 and 000101, respectively. the complete set of pairs of corresponding nodes are $P = \{(p_i, p_j)\} = \{(000000, 000100), (000001, 000101), (001000, 001100), (001010, 001110), (000000, 010000), (000010, 010010)\}$. Now divide Q_6 along dimensions 2 and 4 into four 4-dimensional subcubes, $*0*0**, *0*1**, *1*0**, *1*1**$, which have message sets M_0, M_1, M_2, M_3, respectively. There are 2^4 *PS*'s each of which forms Q_2. At time step 0, $ATAB^p$ will be executed within each *PS* which would take $\frac{2^2-1}{2} = 2$ time steps. Now, each node has four messages, and each subcube of size Q_4 contains all the necessary information for all-to-all broadcasting.

At time step 2: All four subcubes start executing 1st step of $ATAB^p$ within themselves with M_0.

At time step 3: All four subcubes execute 1st step of $ATAB^p$ within themselves with M_1. At the same time, for all pairs in C, end nodes, say a_i and b_i, of the link $(f_i \oplus e^{s_i})$ send to the end nodes of link f_i the messages they (a_i and b_i) exchanged

at time step 2.(Recall that this is referred to as *intersubcubal transmission*).

At time step 4: All subcubes execute 1st step of $ATAB^p$ within themselves with M_2. At the same time, intersubcubal transmissions with the messages exchanged at time step 3 is done.

At time step 5: All subcubes execute 1st step of $ATAB^p$ within themselves with M_3. At the same time, intersubcubal transmissions with the messages exchanged at time step 4 is done.

At time step 6: All subcubes execute 2nd step of $ATAB^p$ within themselves with M_0. At the same time, intersubcubal transmissions with the messages exchanged at time step 5 is done.

At time step 7: All subcubes execute 2nd step of $ATAB^p$ within themselves with M_1. At the same time, intersubcubal transmissions with the messages exchanged at time step 6 is done.

And so on. □

4.4 Conclusion

We have presented a new fault-tolerant all-to-all broadcasting algorithm which can tolerate up to $\lfloor \frac{n}{2} \rfloor$ link faults. The proposed algorithm has several desirable features such as (1) each node sends only one copy of the broadcast message, which reduces traffic in the network by a factor of n over the schemes used in [LS90, Fra92], (2) it utilizes an algorithm developed for the non-faulty system (non-faulty algorithm), (3) further, it can use any of those efficient non-faulty algorithms, which have been

developed or yet to be developed, and (4) it achieves near optimal performance.

All-to-all broadcasting with *node failures* can be done by using the same idea presented in this chapter. In addition, the idea presented in this chapter can be extended to a set of problems in which each subcube performs the same algorithm.

Chapter 5

All-to-All Broadcasting in Wormhole-Routed Hypercube Multicomputers with Link Faults

5.1 Introduction

In this chapter, we consider networks that implement a *cut-through* routing technique rather than *store-and-forward.* In the store-and-forward method, all the intermediate nodes between source and destination nodes must completely store the incoming message before they forward the message to the next node. However, in the cut-through method, the head of the packet is advanced directly from incoming to outgoing channels. Only a few flow control digits are buffered at each node. Both *wormhole* [S+85, Dal87, Dal90] and *virtual cut-through* [KK79] routing methods belong to this category. The only difference between them is that virtual cut-through routing buffers messages when they are blocked, removing them from the network, whereas the blocked messages remain in the network in wormhole routing. The operation of advancing a message directly from incoming to outgoing channels is referred to as *cut-through.*

In this chapter we introduce a simple fault-tolerant ATAB algorithm in wormhole-routed hypercube multicomputers in the presence of up to $n-1$ faulty links, where n is the dimension of the hypercube. In ATAB, each non-faulty node must be able to deliver its message to all the other non-faulty nodes in the system. Both Lee and Shin (LS) [LS90] and Fraigniaud (FR) [Fra92] achieve this under the assumption of non-availability of global fault information, i.e., each non-faulty node does not know the identities of the faulty components. In their algorithms each node delivers multiple copies of the broadcast message through disjoint paths to all the other nodes in the system. On receiving the multiple copies of the same message, each non-faulty node identifies the original message using some schemes such as majority voting. These algorithms have the advantage of not having to know the addresses of the faulty components, and therefore they may be suitable for real-time applications.

The difference between the LS and FR algorithms is that the LS algorithm is based on Hamiltonian Cycles, whereas the FR algorithm is based on tree structure. Since tree structure is not suitable to the networks implementing cut-through routing [LN91], only the LS algorithm will be considered in the remainder of the chapter.

Since each node receives n copies of the same message in the LS algorithm, and since the optimal traffic is $T_0 = N(N-1)$, the traffic caused by the LS algorithm is at least $T_{LS} = nN(N-1)$. $T_0 - T_{LS}$ is huge and will severely degrade the performance in the system, especially ones using wormhole or virtual cut-through

routing [LS90].

The occurrence of the component faults is infrequent, therefore, it may be more efficient to broadcast the fault information by using some fault-tolerant single node broadcasting algorithm such as the Ramanathan and Shin's (RS) algorithm [RS88], so that each node contains the identities of the faulty components. (Note that the RS algorithm does not require the information of the identities of the faulty components in each node, and it takes only n time steps and $n2^n$ traffic to complete the single node broadcasting under n-port assumption.) This allows that each node to send only one copy of the message, as proposed here, to complete all-to-all broadcasting. Therefore, the traffic required by our algorithm is only $N(N-1)$, which is approximately a factor of n less than the LS algorithm.

The rest of the chapter is organized as follows. Section 5.2 summarizes the notations and definitions which will be used throughout the chapter. Section 5.3 introduces a new ATAB algorithm which can tolerate up to $n-1$ link failures. The conclusion follows in Section 5.4.

5.2 Preliminaries

The relative address of the two adjacent nodes in n-dimensional space is a unit vector $000\ldots001_p00\ldots00$, and p is the dimension of the link connecting the two nodes. We denote the above unit vector by e^p. The relative address of the two links l and m is also bitwise Exclusive-Or of their addresses, $l \oplus m$, where $l_i \oplus m_i = 1$ iff $l_i = 0$ (resp., 1) and $m_i = 1$ (resp., 0); $l_i \oplus m_i = 0$, otherwise. The *weight* of a node

or link r is the number of 1's in r. The *distance* between two nodes a and b (or two links l and m) is given by $W(a \oplus b)(or W(l \oplus m))$.

In the following we explain some of the notations and concepts similar to those used in [LS90]. It was noted in [LS90] that an n-dimensional hypercube contains $\lfloor \frac{n}{2} \rfloor$ edge-disjoint Hamiltonian Cycles (HC's).

Let $HC_1, HC_2, \ldots, HC_m, m = \lfloor \frac{n}{2} \rfloor$, be the HC's in Q_n. Further, each HC, say HC_i, is composed of two directed HC's, HC_i^+ and HC_i^-, which share a common undirected HC, HC_i, but their directions are opposite. HC_i^s denotes one of the two directed HC's in HC_i. Also $HC_i^{\bar{s}} = HC_i^+$ if $s = -$, and $HC_i^{\bar{s}} = HC_i^-$ if $s = +$. Note that these $2m$ directed HC's are edge-disjoint, i.e., no directed edge (link) appears more than once in all directed HC's. Since the difference between n and $2m$ becomes insignificant as n increases, n will be used for $2m$ for the remainder of the chapter. A *fault-free* HC does not contain any faulty link, whereas a *faulty* HC does. In the following, HC and HC^d denote a undirected and directed HC's, respectively. However, we will use them interchangeably when the context is clear. HC^F denotes a fault-free HC and HC^{cj} denotes an HC which contains some faulty links according to c and j, where $c \in \{>, <, \leq, \geq, =\}$ and j is an integer. For example, $HC^{>2}$ denotes an HC which contains more than two faulty links.

In our algorithm, different sets of messages are assigned to be routed along different HC's. Thus, if an HC is fault-free, then the set of messages assigned to the HC can be broadcast without any difficulty. We will refer to that as *partial*

completion of the ATAB in that HC. Since there are n edge-disjoint HC^d's, n sets of messages can be routed concurrently in n different HC^d's.

Let us assume that the broadcasting messages are delivered in packets of length $\mu \times B_{FIFO}$, where B_{FIFO} is the size of the FIFO buffers at the receivers of the nodes and μ is an integer. τ_L and τ_S denote *propagation time* between adjacent nodes and *startup time*, respectively. If the packet cuts through a node d, then the delay at the node is denoted by α, which is proportional to B_{FIFO}. If the packet is stored into the intermediate storage buffer before being transmitted, then the delay at the node is $\tau_S + L\tau_L = \tau_S + \mu\alpha$, where L is the length of the packet.

The LS algorithm accomplishes all-to-all broadcasting in η stages as follows. Let p_0 be any designated node. In stage i, every η-th node in direction HC_j starting from i-th neighbor of p_0 in direction HC_j is permitted to initiate a packet along HC_j for all $j, 1 \leq j \leq n$. Once packets have been started along directed HC's, they keep flowing for $N-1$ hops along the cycles in which they started. In a *dedicated network*, i.e., the entire network is devoted to the all-to-all broadcasting for the duration of the broadcast operation, the time required by the LS algorithm is $\eta(\tau_S + \mu\alpha + (N-2)\alpha)$. It is proved in [LS90] that the LS algorithm is optimal when $\eta = \mu$. Since τ_S is much smaller than τ_L, it is better to minimize μ. Since $\mu \geq 2$, when $\mu = 2$ the time taken by the LS algorithm is $2(\tau_S + N\alpha)$. In this case the size of the message can be no longer than $2 \times B_{FIFO}$. If it is necessary to send larger message, the LS algorithm requires more startup time. The traffic generated by the LS algorithm is $nN(N-1)$ since there are N nodes and each node must

receive up to n copies of the same packet from all other nodes in Q_n.

In the following we propose an all-to-all broadcasting algorithm which tolerates up to $n-1$ faulty links in Q_n. The time taken by our algorithm is at most $3(\tau_S + \mu\alpha + (N-2)\alpha)$ if the number of faulty links incident to a single node is not greater than $\lfloor \frac{3n}{4} \rfloor$. This would be most likely case since the probability of having more than $\lfloor \frac{3n}{4} \rfloor$ faulty links incident to a single node is quite low. Further, in the best case, i.e., when there are no faults, the proposed algorithm accommodates $\frac{n}{2}$ times longer message than that of the LS within the same time bound, $2(\tau_S + N\alpha)$. Even in the worst case, i.e., all $n-1$ faulty links are incident to a single node, our algorithm accommodates the same message size as that of LS's. Since the buffer size in the wormhole-routed network is usually quite small, it may not feasible to restrict the message size to $2 \times B_{FIFO}$. When the message size is a multiple of $n \times B_{FIFO}$, our algorithm outperforms LS algorithm by at least a factor of $\frac{n}{3}$ when the maximum number of faulty links incident to a node is less than or equal to $\frac{3n}{4}$. As for the traffic, our algorithm always produces $O(N(N-1))$ which is a factor of n less than that of the LS algorithm when there is no fault in the network.

In the following, it is assumed that the message size is $n \times B_{FIFO}$, unless otherwise specified.

5.3 New all-to-all broadcasting strategy using wormhole routing

In Section 5.3.1, we explain the simple ATAB algorithm which tolerates up to $\lfloor \frac{n}{2} \rfloor$ link faults. Section 5.3.2 presents a somewhat more complex algorithm which tolerates up to $n-1$ faults.

5.3.1 Case of up to $\lfloor \frac{n}{2} \rfloor$ link faults

We now present our new ATAB algorithm in faulty hypercubes with up to $\lfloor \frac{n}{2} \rfloor$ link faults. The following lemmas help in understanding and proving correctness of the proposed algorithm.

Lemma 5.1. Let HC_i be any undirected HC, and let M_i be the set of messages which are assigned to be broadcasted along, say HC_i^+. Suppose HC_i contains a single faulty link. Then if M_i is broadcasted along both HC_i^+ and HC_i^-, then the partial ATAB in HC_i^+ will be completed.

Proof : Let nodes a and b be the end nodes of the faulty link in HC_i. Then, HC_i is nothing but a linear array with nodes a and b as end nodes as shown in Figure 5.1. If the darkened nodes have messages to be broadcast along HC_i^+, then it is straightforward to see that if all the darkened nodes send their messages along both HC_i^+ and HC_i^-, then the partial ATAB in HC_i^+ will be completed with the set of messages belonging to the darkened nodes. □

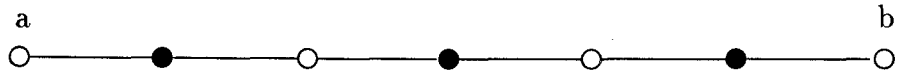

Figure 5.1. Linear array which is formed by a HC with a single link failure. Darkened nodes have the messages to be broadcast.

Lemma 5.2. Let F be the set of HC's each of which contains more than one faulty links in it. If there are at most $\lfloor \frac{n}{2} \rfloor$ faults in Q_n, then there are at least $|F|$ fault-free HC's.

Proof : Let $|F| = k$. Without loss of generality, let HC_i contain f_i faulty links, where $f_i \geq 2$, for all $1 \leq i \leq k$. The number of $HC^{=1}$'s is at most $\lfloor \frac{n}{2} \rfloor - \sum_{i=1}^{k} f_i$. Thus, the total number of fault-free HC's is $\lfloor \frac{n}{2} \rfloor - k - (\lfloor \frac{n}{2} \rfloor - \sum_{i=1}^{k} f_i) = \sum_{i=1}^{k} f_i - k \geq k$.

□

The *weight* of a node is *even* (resp., *odd*) if the number of 1's in its address is even (resp., odd). Let *even messages* (resp., *odd messages*) denote the broadcast messages which initially belong to the nodes with even (resp., odd) weight. In our new algorithm, packets are assigned to HC^d's so that each packet is routed along specific HC('s). The more evenly the assignments are distributed, the better link utilization is achieved. In the proposed algorithm, the even messages are assigned to the HC_i^+'s and the odd messages to HC_i^-'s, for all $1 \leq i \leq \lfloor \frac{n}{2} \rfloor$. Since each message is assigned to $\lfloor \frac{n}{2} \rfloor$ HC^d's and since all directed HC's are edge-disjoint, the message

at each node can be divided into $\lfloor \frac{n}{2} \rfloor$ packets which can be routed along different $\lfloor \frac{n}{2} \rfloor$ HC^d's simultaneously (assume a message size $\leq \lfloor \frac{n}{2} \rfloor \times$ *packet size* $= n \times B_{FIFO}$). Thus, every node with even weight sends its i-th packet along HC_i^+ and every node with odd weight sends its i-th packet along HC_i^-, for all $1 \leq i \leq \lfloor \frac{n}{2} \rfloor$. Once packets have been started along directed HC's, they keep flowing for $N-1$ hops along the cycles in which they started.

Algorithm $ATAB^F$ shown in Figure 5.2 completes all-to-all broadcasting in fault-free hypercubes. Let HC_j be the j-th directed HC, $1 \leq j \leq 2\lfloor \frac{n}{2} \rfloor$. Then, for any given node x and given HC_j, $NEXT_j(x)$ and $PREV_j(x)$ denote the next and previous nodes, respectively, of x in the HC_j. Note that in each HC, $\mu = 2$ since the distance between even (odd) weighted nodes is at least two. Also note that since even and odd weighted nodes use different HC's, they can start all-to-all broadcasting simultaneously. Thus if there is no faulty link and if the message size $\leq \lfloor \frac{n}{2} \rfloor \times$ *packet size*, then ATAB will be completed in $(\tau_S + \mu\alpha + (N-2)\alpha)$ since $\eta = 1$. This result is better than that of LS algorithm by τ_S when the message size $\leq 2 \times B_{FIFO}$ and by $\frac{n}{2}\tau_S$ when the message size is the multiple of $n \times B_{FIFO}$. The gain is huge since τ_S is usually much larger than τ_L. Further, the proposed algorithm completes ATAB with $\frac{n}{2}$ times bigger message size within the same time bound.

Let $FAULT(HC_i^s)$ denote the number of faulty links in HC_i^s. Note that $FAULT(HC_i^+) = FAULT(HC_i^-)$. Now we consider the case when there are up to $\lfloor \frac{n}{2} \rfloor$ faulty links in Q_n. In this case, our algorithm is composed of two stages. At

Algorithm $ATAB^F$

for all nodes **doparallel**

each node divides its broadcast message into $\lfloor \frac{n}{2} \rfloor$ packets

for $i = 1$ **to** $\lfloor \frac{n}{2} \rfloor$ **doparallel**

for every node x **doparallel**

begin

nodes with even weight send their i-th packets along HC_i^+.

nodes with odd weight send their i-th packets along HC_i^-.

end

for $k = 1$ **to** $N - 1$ **do**

for $j = 1$ **to** $2\lfloor \frac{n}{2} \rfloor$ **doparallel**

for every node x **doparallel**

begin

receive packet from $PREV_j(x)$

if $(k < N - 1)$ **then**

relay the message to $NEXT_j(x)$

end

Figure 5.2. Algorithm $ATAB^F$ which completes all-to-all broadcasting in perfect hypercubes.

the first stage, algorithm $ATAB^F$ will be executed. Therefore, after the first stage, partial ATAB will be completed in HC^F's, i.e., fault-free HC's. Partial ATAB in faulty HC's will be completed in the second stage as follows. Every packet initially assigned to the directed $HC^{=1}$'s (HC's which contain only one fault) are reassigned to the same undirected HC's with reversed direction. For example, if HC_i^+ contains a single faulty link, then all the nodes assigned to HC_i^+ send their messages along HC_i^- at the second stage. Since all the nodes assigned to $HC^{=1}$ send their messages along one direction in the first stage and the other at the second stage, the partial ATAB in the $HC^{=1}$ is completed by Lemma 5.1. Partial ATAB for the packets initially assigned to $HC^{>1}$'s will be completed at the second stage as follows. Let R be the set of $HC^{>1}$'s. Then since there are at least $|R|$ number of HC^F's by Lemma 5.2, the packets initially assigned to $HC^{>1}$'s will be reassigned to and broadcast along HC^F's. Algorithm $ATAB^{\leq\lfloor\frac{n}{2}\rfloor}$ shown in Figure 5.3 accomplishes all-to-all broadcasting in hypercubes with up to $\lfloor\frac{n}{2}\rfloor$ faulty links.

It is straightforward to verify that $ATAB^{\leq\lfloor\frac{n}{2}\rfloor}$ is correct. Note that since some nodes receive more than one copy of the same message, the nodes improve the possibility of receiving a correct message by comparing multiple copies of the same message.

5.3.2 Case of up to $n-1$ link faults

If the number of faulty links is more than $\lfloor\frac{n}{2}\rfloor$, then there may not be enough HC^F's to handle the packets initially assigned to $HC^{>1}$'s. For example, let $n = 10$

Algorithm $ATAB^{\leq \lfloor \frac{n}{2} \rfloor}$

Execute algorithm $ATAB^F$ {at stage 1}

Let R be the set of all the fault-free HC's (HC^F's) {at stage 2}

for every node in $HC^{\geq 1}$'s **doparallel**

begin

for all packets assigned to $HC^{=1}$ **do**

Send the packets along $HC_i^{\bar{s}}$ which were initially assigned to HC_i^s

for all packets assigned to $HC^{>1}$ **do**

Send the packets which were initially assigned to HC_i^s along $HC_j^s \in R$,

where HC_j^s is not taken by any other $HC^{>1}$

for $k = 1$ **to** $N - 1$ **do**

for $j = 1$ **to** $2\lfloor \frac{n}{2} \rfloor$ **doparallel**

for every node x **doparallel**

begin

receive packet from $PREV_j(x)$ in reassigned HC, HC_j

if $(k < N - 1)$ **then**

relay the message to $NEXT_j(x)$ in reassigned HC, HC_j

end

end

Figure 5.3. Algorithm $ATAB^{\leq \lfloor \frac{n}{2} \rfloor}$ completes all-to-all broadcasting in hypercubes with up to $\lfloor \frac{n}{2} \rfloor$ faulty links.

and suppose HC_1 through HC_3 contain two faulty links each and HC_4 and HC_5 contain one faulty link each. Then there are not enough HC^F's to take up the packets initially assigned to HC_1 through HC_3. We, of course, can reassign the packets in $HC^{>1}$'s to $HC^{=1}$'s, e.g., reassign the packets in HC_1 through HC_3 to HC_4 and HC_5. Then all-to-all broadcasting will take six stages in this case, since at the first and second stages partial broadcasting in HC_4 and HC_5 will be completed with the packets initially assigned to HC_4 and HC_5, at the third and fourth stages packets initially assigned to HC_1 and HC_2 will be reassigned to HC_4 and HC_5 and the partial broadcasting will be completed, and the fifth and sixth stages packets initially assigned to HC_3 will be reassigned to HC_4 and the partial broadcasting will be completed. In general this strategy takes $\lfloor \frac{n+1}{2} \rfloor$ stages in the worst case. This implies that, as far as startup time is concerned, it will take $\lfloor \frac{n+1}{2} \rfloor \tau_S$. Since, as mentioned before, τ_S takes too much time, we will not use this approach to handle the case.

The above observations have led us to develop a scheme for partial all-to-all broadcasting for $HC^{=2}$'s. Refer to Figure 5.4.a. Suppose nodes a through k form an HC in which links (f, k) and (a, g) are faulty (in the figure, link (c, i) is not part of the HC). Since the HC contains two faulty links, it forms two linear arrays, one with nodes a through f and the other with nodes g through k. Note that since the network is connected and the number of faulty links is less than or equal to $n - 1$, there must exist at least one link connecting the two linear arrays (see Lemma 5.3). We refer to that link as a *bridging link*. For example, in Figure 5.4.a, links (c, i) is

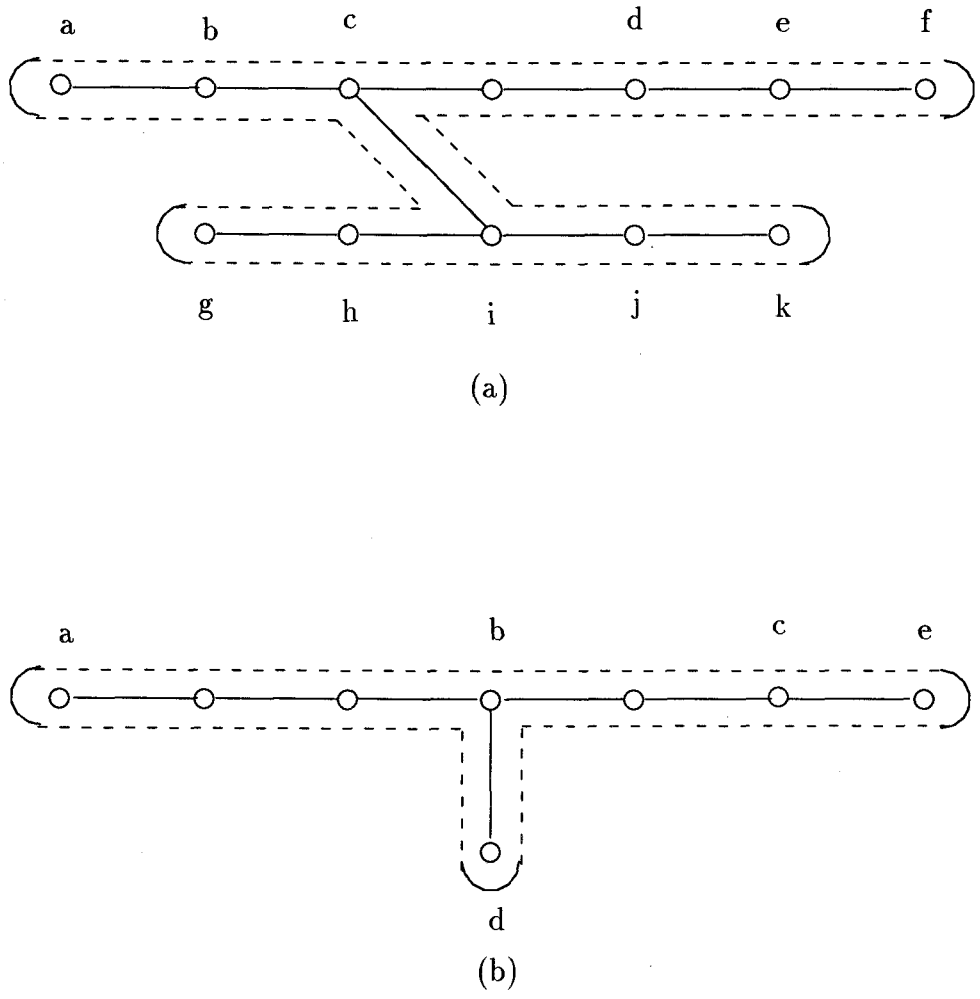

Figure 5.4. Two linear arrays formed by a HC with two faulty links. (a)Two linear arrays which result from the HC with two faulty links (a, g) and (f, k). Link (c, i) is not part of the HC and is called a bridging link. (b) Two linear arrays which result from the HC with two faulty links (a, d) and (e, d). In this case one linear array contains only one node. Link (b, d) is a bridging link.

the bridging link. Figure 5.4.b shows the case when one of the linear arrays contains only one node. Further, Lemma 5.3 shows that the bridging link can be chosen from the links in $HC^{\leq 1}$'s.

Lemma 5.3. For every $HC^{=2}$, there exist at least one bridging link chosen from $HC^{\leq 1}$'s.

Proof : Since there can be at most $n-1$ faulty links, there should exist at least one $HC^{=1}$. If we embed a HC^F into the two linear arrays, it is straightforward to see that there should be at least two bridging links, since if the HC^F starts from one of the nodes in one of the linear arrays, then the cycle should go to the other linear array and come back to to the starting node. Even if we do the same process described above with $HC^{=1}$ instead of HC^F, there should be at least one bridging link. Thus the lemma follows. □

The bridging links are used to form a HC with the two linear arrays as follows. Refer to Figure 5.4. The length of a HC is the number of (directed) links in the cycle. Dotted lines in (a) and (b) in Figure 5.4 indicate the HC formed by the two linear arrays and bridging links. Note that the length of the $HC^{=2}$ is less than $2N$. Thus, partial ATAB in $HC^{=2}$ can be completed within $T^2 = (\tau_S + \mu\alpha + (2N-2)\alpha)$. Let us assign $T^S = (\tau_S + \mu\alpha + (N-2)\alpha)$ time steps for each stage in ATAB. Then $T^2 \leq 2T^S$, i.e., partial ATAB in $HC^{=2}$'s can be completed within two stages.

We now present the outline of the strategy to complete the partial all-to-all broadcasting in $HC^{=2}$'s. In the previous section, different packets are assigned to

different directed HC^d's. However, note that since $HC^{=2}$'s contain links which are part of the $HC^{\leq 1}$'s, i.e., bridging links, partial ATAB in $HC^{=2}$'s can start only after the partial completion of ATAB in all the $HC^{\leq 1}$'s. In order to accomplish the early completion in $HC^{\leq 1}$'s, the same packet will be assigned to and routed along both directions in the $HC^{=1}$. Thus, by Lemma 5.1 partial completion in these HC's will be completed in one stage. $HC^{=2}$'s can start ATAB from second stage. Thus, partial completion of ATAB in $HC^{\leq 2}$'s can be done in three stages.

Even though Lemma 5.3 shows the existence of the bridging links from the $HC^{\leq 1}$'s, the bridging links may not be disjoint in $HC^{=2}$'s, i.e., some of the bridging links may appear in more than one $HC^{=2}$. We try to avoid this multiple appearances since it will cause the link contention problem. However, it may not be possible to avoid it for some cases. For example, suppose node d has $k = n - 1$ faulty links incident to it, where $n \geq 6$. Then the fault-free link incident to d may appear as the bridging links in all the $HC^{=2}$'s. This may cause link contention problem in $HC^{=2}$'s after the first stage of ATAB. Lemma 5.4 shows a tight upper bound on the number of faulty links incident to a single node so that each bridging link appears in only one $HC^{=2}$.

Lemma 5.4. Let k be the maximum number of faulty links incident to a single node. Then, if $k \leq \lfloor \frac{3n}{4} \rfloor$, it is possible to arrange the bridging links so that none of them appears in more than one $HC^{=2}$.

Proof : We will prove the lemma by showing that (1) if $k > \lfloor \frac{3n}{4} \rfloor$, there is an

example in which some bridging links must appear in more than one $HC^{=2}$, and (2) if $k \leq \lfloor \frac{3n}{4} \rfloor$, there is a way to assign bridging links to $HC^{=2}$'s so that no bridging link appears in more than one $HC^{=2}$.

(1) Since the worst case occurs when there is only one node, say d, in one of the two linear arrays as shown in Figure 5.4.b, let the node d have $k > \lfloor \frac{3n}{4} \rfloor$ faulty links incident to it. Note that in each HC, every node has at most two incident links. Thus, among all $\lfloor \frac{n}{2} \rfloor$ HC's, both incident links to node d in $t > \lfloor \frac{n}{4} \rfloor$ HC's are faulty, i.e., there are $t > \lfloor \frac{n}{4} \rfloor$ $HC^{=2}$'s. However, since there are less than $\lfloor \frac{n}{4} \rfloor$ non-faulty links incident to d, it is obvious that some bridging links should appear in more than one $HC^{=2}$.

(2) Let node d have $2 \leq k \leq \lfloor \frac{3n}{4} \rfloor$ incident faulty links (if node d has less than two faulty links incident to it, then there is no bridging link incident to d). Also let l be any of the bridging links incident to d. Then the lemma follows if we prove that the link l does not have to be the bridging link in any other $HC^{=2}$. Let d' be the node connected to d by link l. Without loss of generality, assume that the number of faulty links incident to node d is greater than or equal to that of node d'. Then, if $k < \lfloor \frac{3n}{4} \rfloor$, link l does not have to be the bridging link incident to node d. If $k = \lfloor \frac{3n}{4} \rfloor$, then since node d' has at most $(n-1) - \lfloor \frac{3n}{4} \rfloor - 1 = \lfloor \frac{n}{4} \rfloor$ faulty links incident to it, it does not have to use link l as a bridging link. Thus the lemma follows. □

Even when each node has $\lfloor \frac{3n}{4} \rfloor$ or less faulty links incident to it, $HC^{=2}$'s may be used only in the following situation. Let $S^{=2}$, $S^{=1}$ and S^F be the set of

undirected $HC^{=2}$'s, $HC^{=1}$'s and HC^F's, respectively. Recall that (1) each $HC^{=2}$ completes partial ATAB with two packets, one for each direction, within two stages starting from the second stage, (2) each $HC^{=1}$ completes partial ATAB with one packet within one stage by sending the packet in both direction (Lemma 1), and (3) each HC^F completes partial ATAB with two packets in one stage. Note that it would take at least 3 stages if $HC^{=2}$'s involve in ATAB. Thus, for example, since each HC^F and $HC^{=1}$ complete partial ATAB with 6 and 3 packets, respectively, in three stages, if $3 \times |S^{=1}| + 6 \times |S^F| \geq n$, it is not necessary to use $HC^{=2}$'s at all. Thus, $HC^{=2}$'s will be used only when $|S^{=1}| + 2|S^F| + 2\lceil \frac{n-3}{2|S^{=2}|} \rceil < 2\lceil \frac{n}{2|S^{=1}|+4|S^F|} \rceil$.

Even though the probability of having more than $\lfloor \frac{3n}{4} \rfloor$ faulty links incident to a single node is quite low, in the following we give a strategy to handle the case. Note that sometimes, depending on the size of the message, it may not be necessary to choose distinct bridging links for the $HC^{=2}$'s since we may not use any $HC^{=2}$ at all. Example 5.1 illustrates this.

Example 5.1. Suppose $n = 20$, the number of faulty links in Q_n is 16, and node d has 16 faulty links incident to it. Then there are at least 6 $HC^{=2}$'s, and only 4 links incident to d are available for the bridging links. Since the 6 $HC^{=2}$'s should share the 4 bridging links, there may be a contention problem. However, if the message size $\leq 2 \times B_{FIFO}$, then ATAB can be done by using only one of $HC^{\leq 1}$'s. In this case no $HC^{=2}$ is needed at all. □

However, the contention problem may occur in the bridging links as the

message size increases. Recall that the contention problem occurs only when a node has more than $\lfloor \frac{3n}{4} \rfloor$ faulty links incident to it. There can be at most one such node since the total number of faulty links in Q_n is $n-1$ or less.

Our solution to the contention problem is to reassign to $HC^{\leq 1}$'s the packets initially assigned to $HC^{\geq 2}$, i.e., no $HC^{\geq 2}$ is used in this case. Thus, with this strategy, the total time steps to complete ATAB is $\lceil \frac{n}{|S^{=1}|+2|S^F|} \rceil$. Example 5.2 illustrates this.

Example 5.2. Suppose $n = 20$, the number of faulty links in Q_n is 19, and node d has 19 faulty links incident to it. Also suppose that there are one $HC^{=3}$, seven $HC^{=2}$'s, two $HC^{=1}$'s and one HC^F. Since the seven $HC^{=2}$'s should share the 4 bridging links, there is contention problem. If the broadcast message size at each node is $\lfloor \frac{n}{2} \rfloor \times$ *packet size*, then the 10 packets are assigned to HC's as follows. Since one HC^F and one $HC^{=1}$ can complete two and one packets, respectively, in one stage, all the $HC^{\leq 1}$ in Q_n can complete $2 \times 1 + 2 = 4$ packets in one stage. Thus, total number of stages taken in this case is $\frac{10}{4} = 3$ stages. Note that with LS algorithm, it would take 20 stages since their algorithm takes two stages with one packet. □

Theorem 5.5 shows the optimality of the algorithm in this case.

Theorem 5.5. When the maximum number of faulty links incident to a single node is greater than $\lfloor \frac{3n}{4} \rfloor$, the strategy which uses only $HC^{\leq 1}$'s for ATAB requires opti-

mal number of stages.

Proof: The performance in this case is restricted by the links in $HC^{\leq 1}$ since if $HC^{=2}$'s are used, then the bridging links are chosen from $HC^{\leq 1}$. However, the strategy is optimal since it fully utilizes the links in $HC^{\leq 1}$'s at every stage. □

The general idea in this chapter is that whenever an HC has difficulty completing the partial ATAB with the packets initially assigned to it, the packets will be reassigned to another HC which can complete partial ATAB relatively easily. At the same time, we try to reduce the total startup time and to maximize the message size. In order to achieve those it is critical to finish ATAB with a minimum number of stages. Thus an efficient algorithm is needed to reassign the packets initially assigned to the $HC^{\geq 2}$'s to $HC^{\leq 2}$'s. Note that the reassigning problem is quite similar to that of scheduling; $HC^{>2}$'s are tasks and HC^{F}'s, $HC^{=1}$'s and $HC^{=2}$'s are the processors which complete two, one, and $\frac{1}{2}$ tasks, respectively, in one stage. Algorithm *Feed* in Figure 5.5 shows this reassigning procedure when the number of faults is greater than $\lfloor \frac{n}{2} \rfloor$.

Now we present the algorithm $ATAB^{n-1}$ in Figure 5.6 which completes ATAB in hypercube with up to $n-1$ faulty links.

Algorithm *Feed*

F = set of faulty links in Q_n

k = the maximum number of faulty links incident to a single node, d.

$< HC >$= set of all undirected HC's, thus $| < HC > | = \lfloor \frac{n}{2} \rfloor$

S^F = set of HC^F's, $S^{=1}$ = set of $HC^{=1}$'s, $S^{>1}$ = set of $HC^{>1}$'s

if $k \leq \lfloor \frac{3n}{4} \rfloor$ **then begin**

  if $|S^F| \geq \lfloor \frac{n}{4} \rfloor$ **then**

    reassign the packets which are initially assigned to $HC^{\geq 1}$'s to HC^F's

  else if $|S^{=1}| + 2|S^F| + 2\lceil \frac{n-3}{2|S^{=2}|} \rceil < 2\lceil \frac{n}{2|S^{=1}|+4|S^F|} \rceil$ **then**

    reassign the packets which are initially assigned to $HC^{\neq 2}$'s to $HC^{=2}$'s

  else

    reassign the packets which are initially assigned to $HC^{\geq 2}$'s to $HC^{\leq 1}$'s

end

else begin{case when $k > \lfloor \frac{3n}{4} \rfloor$}

  Let R be the set of $HC^{>1}$'s

  Let $\bar{HC} =< HC > -R$

  Let $q = (\frac{|R|}{2|S^F|+|S^{=1}|})$

    reassign $2q$ HC's in R to each of HC^F

    reassign q HC's in R to each of $HC^{=1}$

end

Figure 5.5. Algorithm *Feed* reassigns the packets. The packets were initially assigned to HC's which have difficulty to complete partial ATAB.

Algorithm $ATAB^{n-1}$
F = set of faulty links in Q_n
k = the maximum number of faulty links incident to a single node, d.
$< HC >$= set of all the undirected HC's, thus $| < HC > | = \lfloor \frac{n}{2} \rfloor$
S^F = set of HC^F's, $S^{=1}$ = set of $HC^{=1}$'s, $S^{>1}$ = set of $HC^{>1}$'s
if no faulty link in Q_n **then** execute $ATAB^F$
else if $|F| \leq \lfloor \frac{n}{2} \rfloor$ **then** execute $ATAB^{\leq \lfloor \frac{n}{2} \rfloor}$
else if $k \leq \lfloor \frac{3n}{4} \rfloor$ **then begin**
call algorithm *Feed*
for every node **doparallel** in each stage
if the number of HC^F's $\geq \lfloor \frac{n}{4} \rfloor$ **then begin**
At the 1st stage : HC^F's complete partial ATAB with the packets initially assigned to them
At the 2nd stage : HC^F's complete partial ATAB with the packets which are reassigned to them
end
else if $|S^{=1}| + 2|S^F| + 2\lceil \frac{n-3}{2|S^{=2}|} \rceil < 2\lceil \frac{n}{2|S^{=1}|+4|S^F|} \rceil$ **then begin**
calculate the bridging links for $HC^{=2}$'s
At the 1st stage : $HC^{\leq 1}$'s complete partial ATAB with the packets initially assigned to them
At each stage $i = 2, 3, \ldots, |S^{=1}| + 2|S^F| + 2\lceil \frac{n-3}{2|S^{=2}|} \rceil$: $HC^{=2}$'s complete partial ATAB with packets which are reassigned to them.
end
else begin
At the 1st stage : $HC^{\leq 1}$'s complete partial ATAB with the packets initially assigned to them
At the 2nd stage : $HC^{\leq 1}$'s complete partial ATAB with the packets which are reassigned to them.
end
end
else begin {case when $k > \lfloor \frac{3n}{4} \rfloor$}
Let R be the set of $HC^{=2}$'s in which both incident links to node d are faulty and the set of $HC^{>2}$'s.
Let $\bar{HC} =< HC > -R$
for every node **doparallel**
At the 1st stage : all the HC's in $\bar{HC}$ complete partial ATAB with the packets which are assigned to them.
Let $q = (\frac{|R|}{2|SH^F| + |SH^{=1}|})$
each of HC^F executes $2q$ packets reassigned to it
each of $HC^{=1}$ executes q packets reassigned to it
end
end

Figure 5.6. Algorithm $ATAB^{n-1}$ completes all-to-all broadcasting in hypercubes with up to $n-1$ faulty links.

5.4 Conclusion

We have proposed a new all-to-all broadcasting algorithm in faulty wormhole-routed hypercubes with up to $n-1$ faulty links. The algorithm often produces a factor of n less traffic and accommodates a larger message size than the previously known algorithms. Further, it tries to minimize the startup time which is much slower than the propagation time. Even though the proposed algorithm may work for networks implementing store-and-forward routing technique, it better suits wormhole– or virtual cut-through–routed networks since in those networks startup time is the dominant factor in the communication performance.

The packet size is $2 \times B_{FIFO}$ in both the proposed algorithm and Lee and Shin's (LS). The traffic generated by the proposed algorithm is close to the lower bound, $N(N-1)$, which is a factor of n less than that of the LS algorithm. Further, it can accommodate n times longer message than that of the LS algorithm within the same time bound. As for the time, the LS algorithm is close to the proposed one only when the message size $\leq 2 \times B_{FIFO}$. However, since the buffer size in the wormhole-routed network is usually small, it may not be feasible to restrict the message size to $2 \times B_{FIFO}$. When the message size is $n \times B_{FIFO}$ and when the number of faulty links incident to each node is less than or equal to $\lfloor \frac{3n}{4} \rfloor$, the time taken by our algorithm is often at most $3(\tau_S + \mu\alpha + (N-2)\alpha)$, whereas that of LS's is $n(\tau_S + \mu\alpha + (N-2)\alpha)$. Thus, when the message size is a multiple of $n \times B_{FIFO}$, our algorithm outperforms the LS algorithm by a factor of up to $\frac{n}{3}$.

One of the open questions we are currently working on is how to find all the disjoint Hamiltonian Cycles dynamically such that all the faulty links are confined to a minimum number of cycles. If such an algorithm is found, then many more faults may be tolerated.

Chapter 6

Conclusion

We have proposed (1) fault-tolerant single node broadcasting in hypercubes with up to $n-1$ link/node faults, (2) fault-tolerant single node broadcasting in hypercubes with up to $\frac{n^2-n}{2}$ faulty links, (3) fault-tolerant all-to-all broadcasting in hypercubes with up to $\lfloor \frac{n}{2} \rfloor$ faulty links, and (4) fault-tolerant broadcasting in wormhole-routed hypercubes with up to $n-1$ faulty links.

The proposed single node broadcasting algorithm which tolerates up to $n-1$ link/node faults is optimal in terms of both time and traffic. It utilizes the characteristic of the recursive construction of the hypercube, which is the basis of most of the work done here.

We improved the algorithm described above to tolerate up to $\frac{n^2-n}{2}$ faulty links. Traffic caused and time steps taken by the algorithm are optimal and close to optimal, respectively. Similar ideas can also be applied to the case of node failures.

Two fault-tolerant all-to-all broadcasting algorithms have been presented: one for networks which implement *store-and-forward* and the other for *wormhole*

routing techniques. All the previously known algorithms assume that each node does not know the identities of the faulty components, which forces the algorithms to send multiple copies of the same message to disjoint paths. This causes unnecessary traffic in the network. Whereas, in our algorithms each node knows the addresses of the faulty components, thus there will be no redundant traffic in the network.

Since our assumption is that each node knows the global information in the network, each node should always be alert to the conditions of the neighboring components. When fault occurs, a node which is in charge of the fault may broadcast the identity of the faulty component to all nodes in the system. This causes some overheads in the system. One of the future tasks may be to minimize the overheads. For example, the concept of *unsafeness* [LH88] can be applied to the proposed algorithms so that the node in charge of the fault broadcasts the faulty information only to a subset of the nodes in the system.

As mentioned in the previous chapter, another future task is to find a method to confine as many faulty links to the minimum number of Hamiltonian Cycles. If it is possible, then we can utilize more fault-free Hamiltonian Cycles for reassigning the packets in faulty Hamiltonian Cycles.

BIBLIOGRAPHY

[B+91] D. P. Bertsekas et al. "Optimal communication algorithms for hypercubes". *Journal of Parallel and Distributed Computing*, 11:263–275, 1991.

[B+92] J. Bruck et al. "Tolerating faults in hypercubes using subcube partitioning". *IEEE Trans. Comput.*, 41(5):599–604, 1992.

[Ban89] P. Banerjee. "Reconfiguring a hypercube multiprocessor in the presence of faults". In *Proc. 4th Conf. on Hypercubes, Concurrent Computers and Applications*, pages 95–102, Mar. 1989.

[BK79] J. Bently and H. T. Kung. "A tree machine for searching problem". In *Proc. 1979 Int'l Conf. Parallel Processing*, pages 257–266, Aug. 1979.

[BS86] B. Becker and H. U. Simon. "How robust is the n-cube?". In *Proc. 27-th Annual Sympo. on Foundation of Computer Science*, pages 283–291, 1986.

[BT89] D. Bertsekas and J. Tsitsiklis. *"Parallel and distributed computation"*. Prentice-Hall, 1989.

[CS87] M. Chen and K. Shin. "Processor allocation in an n-cube multiprocessor using Gray codes". *IEEE Trans. Comput.*, c-36(12):1396–1407, Dec. 1987.

[CS88] M. S. Chen and K. Shin. "Message routing in an injured hypercube". In *Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications*, pages 312–317, 1988.

[CS89] M. S. Chen and K. Shin. "On hypercube fault-tolerant routing using global information". In *Proc. 4th Conf. on Hypercube Concurrent Computers and Applications*, pages 83–86, 1989.

[CS90a] M. Chen and K. Shin. "Addressing, Routing, and broadcasting in hexagonal mesh multiprocessors". *IEEE Trans. Comput.*, 39(1):10–18, Jan. 1990.

[CS90b] M. Chen and K. Shin. "Subcube allocation and task migration in hypercube multiprocessors". *IEEE Trans. Comput.*, 39(9):1146–1155, Sept. 1990.

[CS90c] M. S. Chen and K. Shin. "Depth-first search approach for fault-tolerant routing in hypercube multicomputers". *IEEE Trans. Parallel and Distributed Syst.*, 1(2):152–159, Apr. 1990.

[CS90d] M. S. Chen and K. Shin. "Adaptive fault-tolerant routing in hypercube multicomputers". *IEEE Trans. Comput.*, 39(12):1406–1416, Dec. 1990.

[Dal87] W. Dally. "Deadlock-free message routing in multiprocessor interconnection networks". *IEEE Trans. Comput.*, c-36(5):547–553, May 1987.

[Dal90] W. Dally. "Performance analysis of k-ary n-cube interconnection networks". *IEEE Trans. Comput.*, 39(6):775–785, June 1990.

[DB92] B. Duzett and R. Buck. "An overview of the nCUBE 3 supercomputer". In *Proc. 4th Symp. on the Frontiers of Massively Parallel Computation*, pages 458–464, 1992.

[Fra92] P. Fraigniaud. "Asymptotically optimal broadcasting and gossiping in faulty hypercube multicomputers". *IEEE Trans. Comput.*, 41(11):1410–1419, Nov. 1992.

[GS88] J. M. Gordon and Q. F. Stout. "Hypercube message routing in the presence of faults". In *Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications*, pages 318–327, 1988.

[GS89] A. Ghafoor and P. Sole. "Performance of fault-tolerant diagnostics in the hypercube systems". *IEEE Trans. Comput.*, 38(8):1164–1171, Aug. 1989.

[HJ86] C. T. Ho and S. Johnsson. "Distributed routing algorithms for broadcasting and personalized communication in hypercubes". In *Proc. Int'l Conf. on Parallel Processing*, pages 640–648, 1986.

[HZ81] E. Horowitz and A. Zorat. "The binary tree as an interconnection network: Applications to multiprocessor systems and VLSI". *IEEE Trans. Comput.*, c-30:247–253, 1981.

[JBH91a] R. Cypher J. Bruck and C. Ho. "On the construction of fault-tolerant cube-connected cycles networks". In *Int'l Conf. on Parallel Processing*, pages 692–693, 1991.

[JBH91b] R. Cypher J. Bruck and C. Ho. "Fault-tolerant parallel architectures with minimum number of spares". In *Tech. Report, IBM RJ 8029,* Mar. 1991.

[JH89] S. Johnsson and C. T. Ho. "Optimum broadcasting and personalized communication in hypercubes". *IEEE Trans. Comput.*, 38(9):1249–1268, Sept. 1989.

[JH91] S. Johnsson and C. T. Ho. "Optimal all-to-all personalized communication with minimum span on boolean cubes". In *Proc. 6-th Distributed Memory Computing Conference*, pages 299–304, 1991.

[KK79] P. Kermani and L. Kleinrock. "Virtual cut-through: A new computer communication switching technique". *Comput. Networks*, 3:267–286, 979.

[L+90] Y. Lan et al. "Multicast in hypercube multiprocessors". *J. of Parallel and Distributed Computing*, 8:30–41, 1990.

[LH88] T. C. Lee and J. P. Hayes. "Routing and broadcasting in faulty hypercube computers". In *Proc. 3rd Conf. on Hypercube Concurrent Computers and Applications*, pages 625–630, 1988.

[LH89] T. Lee and J. Hayes. "One-step-degradable fault-tolerant hypercubes". In *Proc. 4th Conf. on Hypercubes, Concurrent Computers and Applications*, pages 87–93, Mar. 1989.

[LLP82] R. Shostak L. Lamport and M. Pease. "The Byzantine general problem". *ACM Trans. Prog. Languages and Systems*, 4(3):382–401, July 1982.

[LMS85] L. Lamport and P. M. Melliar-Smith. "Synchronizing clocks in the presence of faults". *J. of ACM*, 32(1):52–78, Jan. 1985.

[LN91] X. Lin and L. M. Ni. "Deadlock-free multicast wormhole routing in multicomputer networks". In *Proc. 18th Int'l Symp. Computer Architecture*, pages 116–125, 1991.

[LS90] S. Lee and K. Shin. "Interleaved all-to-all reliable broadcast on meshes and hypercubes". In *Proc. Int'l Conf. on Parallel Processing*, pages III–110–III–113, 1990.

[PB90a] S. Park and B. Bose. "Burst unidirectional/asymmetric error correcting codes". In *Int'l Symposium on Information Theory*, Jan. 1990.

[PB90b] S. Park and B. Bose. "Burst unidirectional/asymmetric error correcting and detecting codes". In *Proc. Symposium on Fault Tolerant Computing*, pages 273–280, June 1990.

[PB90c] M. Peercy and P. Banerjee. "Distributed algorithms for shortest-path, deadlock-free routing and broadcasting in arbitrarily faulty hypercubes". In *Proc. 20th FTCS*, pages 218–225, 1990.

[PB92] S. Park and B. Bose. "Broadcasting in hypercubes with link/node failures". In *Proc. 4th Symp. on the Frontiers of Massively Parallel Computation*, pages 286–290, 1992.

[PV81] F. P. Preparata and J. E. Vuillemin. "The cube-connected-cycles: A versatile network for parallel computation". *CACM*, pages 300–309, May 1981.

[RS88] P. Ramanathan and K. Shin. "Reliable broadcast in hypercube multicomputers". *IEEE Trans. Comput.*, 37(12):1654–1657, Dec. 1988.

[S+85] C. L. Seitz et al. "Wormhole chip project report", Winter 1985.

[SB77] H. Sullivan and T. R. Bashkow. "A large scale homogeneous fully distributed parallel machine". In *Proc. 4-th Symp. Computer Architecture*, pages 105–117, 1977.

[Sei85] C. L. Seitz. "The cosmic cube". *Commun. of the ACM*, 28(1):22–33, Jan. 1985.

[SP89] M. Samatham and D. Pradhan. "The de Bruijn multiprocessor network: a versatile parallel processing and sorting network for VLSI". *IEEE Trans. Comput.*, 38(4):567–581, Apr. 1989.

[SS88] Y. Saad and M. Schultz. "Topological properties of hypercubes". *IEEE Trans. Comput.*, 37(7):867–872, July 1988.

[Sto71] H. S. Stone. "Parallel processing with the perfect shuffle". *IEEE Trans. Comput.*, c-20:153–161, 1971.

[Tan90] A. N. Tanenbaum. *"Structured Computer Organization"*. Prentice-Hall, 1990.

[YM88] C. L. Yang and G. M. Masson. "A distributed algorithm for fault diagnosis in systems with soft failures". *IEEE Trans. Comput.*, 37(11):1476–1479, Nov. 1988.

[YN90] A. Youssef and B. Narahari. "The Banyan-hypercube networks". *IEEE Trans. Comput.*, 1(2):160–169, Apr. 1990.