AN ABSTRACT OF THE THESIS OF

Robert Richard Broeg, SDS for the degree of Doctor of Philosophy in Computer Science presented on October 19, 1995.

Title: Topics in Toroidal Interconnection Networks.

Abstract approved: Redacted for Privacy

Bella Bose

A multicomputer is a machine having multiple processing elements that communicate with each other by sending messages. The messages are transmitted through an interconnection network, which can be classified as a direct or an indirect network depending on the style of connection among the processing elements. The pattern of interconnection is called the topology of the network, and a popular topology for a direct interconnection network is the torus.

The torus can be characterized as a graph that is the cross product of cycles. The graphs of the k-ary n-cube (Q_n^k) and the hypercube (Q_n) are special cases of a torus graph ($T_{\mathbf{K}}$), and a network based on one of these topologies is referred to a toroidal interconnection network.

This thesis considers various topological characteristics of a toroidal interconnection network using Lee distance, a metric from the field of Error-Correcting codes. Using Lee distance, the torus is defined, and the number and length of edge disjoint paths between two nodes is given. In addition, five Lee distance Gray codes are given; and these Gray codes are applied to finding both a Hamiltonian cycle and a cycle of any even length in a torus of certain dimensions.

For the k-ary n-cube, formulae for the volume and surface area of a sphere of radius d are derived, and methods of decomposing a Q_n^k into n disjoint Hamiltonian cycles are considered.

In addition to topological characteristics, communication algorithms are considered. A one–to–all, an all–to–all, and a redundant fault tolerant (up to $2n-1$ faults) one–to–all broadcasting algorithm for the general torus are presented, and a non-redundant fault tolerant (up to $n-1$ faults) one–to–all broadcasting algorithm for the k-ary n-cube are given.

Topics in Toroidal Interconnection Networks

by

Robert Richard Broeg, SDS

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

Completed October 19, 1995
Commencement June 1996

Doctor of Philosophy thesis of Robert Richard Broeg, SDS presented on October 19, 1995

APPROVED:

Redacted for Privacy

Major Professor, representing Computer Science

Redacted for Privacy

Chair of the Department of Computer Science

Redacted for Privacy

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Redacted for Privacy

Robert Richard Broeg, SDS, Author

ACKNOWLEDGMENT

There are many people to whom I owe a great debt for their assistance and support throughout the process of this Ph. D. program. Although I cannot name them all, a few deserve special mention.

Firstly, I am grateful to Frs. Barry Griffin, SDS; Paul Portland, SDS; and Dennis Thiessen, SDS. Without their support as Provincials Superior, this Ph. D. program would not have been possible. There is a saying in the Roman church, "It is sometimes easier to ask for forgiveness than it is to get permission." If I did not exactly get permission to complete this program at Oregon State University, I was at least granted some measure of forgiveness.

Secondly, I owe an enormous debt to Dr. Bella Bose, my advisor and my friend. Without his continual support, flow of ideas, and belief in me, I would have long ago returned to high school teaching.

Thirdly, I wish to thank Dr. John Marsaglia, the chair of the Computer Science Department at Western Oregon State College. For much of my time at Oregon State University, Western has been my financial support, and John always arranged my schedule to my advantage. In particular, at certain crucial times, I noticed that my schedule was somewhat lighter than what my FTE called for, and this has been a great help.

Fourthly, I wish to thank the members of my committee: Dr. Phillip Sollins of the Forestry department, Drs. Paul Cull, Tim Budd, Walter Rudd, and Prasad Tadepalli of the Computer Science department. It has been a long and torturous road for me, but their patience and understanding has smoothed the way considerably.

Lastly, but certainly not the least, I wish to thank the most important people in the Computer Science department, the ones who make everything work: the office staff. Throughout my tenure at OSU, Bernie Feyerherm, Sheryl Parker, Clara Knutson, Joe Alvarez, and the rest of the office staff have reified the expression, "Once I was lost, but now I'm found."

TABLE OF CONTENTS

Page

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

TOPICS IN TOROIDAL INTERCONNECTION NETWORKS

1. INTRODUCTION

A computer designed for use as a parallel processing machine is often described as either a *multiprocessor* or a *multicomputer* [7, 85]. The term multiprocessor usually describes a machine having multiple processing units that communicate through a shared memory and having a global memory address space. The term multicomputer usually describes a machine having multiple nodes that communicate by passing messages through a network. At a minimum, each node of a multicomputer normally consists of one or more processing elements, a local memory, and a communication module. The terms *tightly coupled* and *loosely coupled* also have been used to describe a multiprocessor and a multicomputer respectively [84].

The network connecting the nodes of a multicomputer can be described as *direct* or *indirect* [77]. In a direct network, a node is connected to some other nodes, its neighbors, in a point-to-point manner. Sending a message to a node that is not a neighbor requires passing the message from neighbor to neighbor, starting from the source node, until the message reaches the destination node. This is in contrast to an indirect network where all nodes are connected to a switching network. In this case, when a node wishes to send a message, it injects the message into the switching network, which delivers the message to the appropriate destination.

The pattern of connection among the nodes in a direct network is called the *topology* of the network, and it can be modeled as a graph where a vertex of the graph represents a node of the network and an edge of the graph represents a communication link. Many topologies have been proposed for parallel machines. Among these are the torus [18], k-ary n-cube [86, 32], the binary hypercube [87] and the mesh [88, 36].

When modeled as a graph, the torus is the most general of the topologies listed in the previous paragraph. A k-ary n-cube is an instance of a torus, a hypercube is an instance of a k-ary n-cube, and a mesh is a subgraph of a torus. The torus, the k-ary n-cube, and the hypercube are referred to in this thesis as *toroidal* topologies.

Several researchers in interconnection topologies, for example Dally [31, 33] and Agarwal [2], have concluded that a low-dimensional toroidal network offers many attractive features. These include a lower message latency and a higher hot-spot throughput than a high-dimensional network such as a binary hypercube under the assumption of constant bisection width.

Based on these advantages, several parallel machines, both commercial and experimental have been designed with a toroidal (or mesh) interconnection network. Included among these machines are the following: the Cosmic Cube (hypercube) [87], the Ametek S/14 (hypercube) [7], the nCUBE (hypercube) [43, 20], the iPSC (hypercube) [42, 43], the CM-200 (hypercube) [21], the Mosaic (k-ary n-cube) [89], the Cray T3D (torus) [76], the iWarp (torus) [18], the Tera Parallel Computer (torus) [94], the J-Machine (mesh) [36, 38], the Goodyear Aerospace MPP (mesh) [10], the MasPar MP-1 (mesh) [20], the K2 Parallel Processor (mesh) [6], Ametek 2010 (mesh) [88], the Stanford DASH (mesh) [70] and the Touchstone Delta System (mesh) [100].

In addition to the machines above, there have been many topologies proposed that are variants to a toroidal interconnection network. Among them are: the Generalized Hypercube [15], the Extended Hypercube [64], the Banyan-Hypercube

[99, 65], the Cube-Connected Cycles [82], the Cube-Connected Möbius Ladder [83], the Möbius Cube [29], the Crossed Cube [45], the Fibonacci Cube [57], and the Twisted Torus [90].

1.1. Organization

This thesis states that Lee distance, a metric from the field of Error-Correcting codes, is a natural metric to use with a toroidal interconnection network. To validate this claim, topological characteristics for both the general torus and the more restricted k-ary n-cube are investigated. Results are derived that are both of theoretical interest and applicable to communication algorithms.

Specifically, this thesis is organized as follows:

- The remainder of this chapter presents the mathematical preliminaries. This includes the definitions of Lee weight, Lee distance, the torus, and the k-ary n-cube.

- Chapter 2 investigates topological properties of the general torus. The major results in this chapter include a theorem giving the number and lengths of edge-disjoint paths between two nodes and a Lee distance Gray code, which is used to embed a cycle of any even length, include a Hamiltonian cycle, in a torus meeting certain conditions.

- Chapter 3 contains results that apply to the k-ary n-cube in particular. In this chapter, a formula giving the number of nodes at a distance d from a given node is derived, along with three more Lee distance Gray codes. These Gray codes are used to embed a Hamiltonian cycle, a mesh, and a hypercube into a k-ary n-cube with certain restrictions.

- Chapter 4 contains partial results on decomposing a k-ary n-cube into disjoint Hamiltonian cycles. Using a Gray code from Chapter 2 and the fact that a k-ary n-cube is the cross product of cycles, the decomposition of a k-ary 2-cube into 2 disjoint cycles and a k-ary 3-cube into 3 disjoint Hamiltonian cycles is shown. Also shown is the decomposition of a k-ary $2n$-cube into $2n$ disjoint Hamiltonian cycles if the k-ary n-cube has been decomposed into n disjoint Hamiltonian cycles. In addition, if $n = n_1 + n_2$, where $n_1 < n_2$, and if both the k-ary n_1-cube and the k-ary n_2-cube have been decomposed into n_1 and n_2 disjoint Hamiltonian cycles, respectively, it is shown how to decompose the k-ary n-cube into $2n_1$ disjoint Hamiltonian cycles.

- Chapter 5 applies some of the earlier results to communication algorithms. In particular, a one–to–all and an all–to–all broadcast algorithm are given for the torus. Also, two fault-tolerant one–to–all broadcast algorithms, one for the torus and one for the k-ary n-cube are given.

This results in this thesis show that Lee distance and Lee distance Gray codes are both useful and productive tools for analyzing a toroidal interconnection network. This thesis demonstrates that theoretical understanding can be enhanced and that practical results, particularly in the area of communication algorithms, can be obtained by the application of these tools.

1.2. Mathematical Preliminaries

This section reviews the definitions and mathematical notation that are used in this thesis. These includes mixed radix numbers, Lee weight and distance, the cross product of graphs, and the definitions of a torus and a k-ary n-cube.

Mixed Radix Vectors

Let $\mathbf{X} = x_{n-1}x_{n-2}\cdots x_0$ be an n-dimensional vector over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$; that is, x_i has radix k_i for each i. $\mathbf{X}$ is said to be in *mixed radix notation*, and $I(\mathbf{X})$, the *Integer Value* of $\mathbf{X}$, is defined as

$$I(\mathbf{X}) = x_{n-1}k_0k_1\ldots k_{n-2} + \cdots + x_1k_0 + x_2k_0k_1 + x_0 = \sum_{i=1}^{n-1}\left(x_i\prod_{j=0}^{i-1}k_j\right) + x_0.$$

For example, if 432 is a vector over 543, then

$$I(432) = 4(12) + 3(3) + 2 = 59.$$

Hamming Weight

Let $\mathbf{X} = x_{n-1}x_{n-2}\cdots x_0$ be an n-dimensional vector over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$. The *Generalized Hamming Weight*, W_H, of $\mathbf{X}$ is defined as the number of non-zero components of $\mathbf{X}$. That is,

$$W_H(\mathbf{X}) = \sum_{i=0}^{n-1}|x_i|, \text{ where } |x_i| = 0 \text{ if } x_i = 0 \text{ and } |x_i| = 1 \text{ otherwise.}$$

For example, if 302 is a vector defined over 765, then

$$W_H(302) = 1 + 0 + 1 = 2.$$

Hamming Distance

Let $\mathbf{X} = x_{n-1}x_{n-2}\cdots x_0$ and $\mathbf{Y} = y_{n-1}y_{n-2}\cdots y_0$ be two n-dimensional vectors over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$. The *Generalized Hamming Distance*, D_H, between $\mathbf{X}$ and $\mathbf{Y}$ is defined as the number of positions in which they differ. That is,

$$D_H(\mathbf{X},\mathbf{Y}) = \sum_{i=0}^{n-1}|x_i - y_i|, \text{ where } |x_i - y_i| = 0 \text{ if } x_i = y_i \text{ and } |x_i - y_i| = 1 \text{ otherwise.}$$

For example,

$$D_H(531, 554) = 2.$$

Lee Weight

Let $\mathbf{X} = x_{n-1}x_{n-2}\cdots x_0$ be an n-dimensional vector over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$. The *Generalized Lee Weight*, W_L, of $\mathbf{X}$ is defined as

$$W_L(\mathbf{X}) = \sum_{i=0}^{n-1} |x_i|, \text{ where } |x_i| = \min(x_i, k_i - x_i).$$

For example, if 342 is a vector defined over 765, then

$$W_L(342) = \min(3,4) + \min(4,2) + \min(2,3) = 7.$$

Lee Distance

Let $\mathbf{X} = x_{n-1}x_{n-2}\cdots x_0$ and $\mathbf{Y} = y_{n-1}y_{n-2}\cdots y_0$ be two n-dimensional vectors over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$. The *Generalized Lee Distance*, D_L, between $\mathbf{X}$ and $\mathbf{Y}$ is defined as

$$D_L(\mathbf{X},\mathbf{Y}) = \sum_{i=0}^{n-1} \min(x_i - y_i, y_i - x_i), \text{ where the subtraction is modulo } k_i.$$

For example, let 131 and 554 be vectors over 765, then

$$D_L(131, 554) = 3 + 2 + 2 = 7.$$

The Lee distance between $\mathbf{X}$ and $\mathbf{Y}$ also can be stated as the Lee weight of their difference. That is, $D_L(\mathbf{X},\mathbf{Y}) = W_L(\mathbf{X} - \mathbf{Y})$, where the subtraction is componentwise and modulo k_i. In the previous example,

$$D_L(131, 554) = W_L(131 - 554) = W_L(342) = 7.$$

It should be noted also that $D_H(\mathbf{X},\mathbf{Y}) = W_H(\mathbf{X} - \mathbf{Y})$, and that $W_L(\mathbf{X}) = W_H(\mathbf{X})$ for radix 2 or 3 and $W_L(\mathbf{X}) \geq W_H(\mathbf{X})$ for a radix greater than three.

Lee Distance Gray Code

Let $\mathcal{S} = \langle S_0, S_1, \ldots, S_{N-1} \rangle$ be a sequence of distinct n digit vectors defined over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$, where $N = \sum_{i=0}^{n-1} k_i$. If $D_L(S_i, S_{i+1}) = 1$, for $0 \leq i \leq N-2$, and if $D_L(S_0, S_{N-1}) = 1$, then $\mathcal{S}$ forms a *Lee Distance Gray Code.*

Cross Product of Graphs

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two graphs. The *cross product* of G_1 and G_2, denoted by $G_1 \otimes G_2$, is defined as the graph $G = (V, E)$ where

$$V = \{(v_1, v_2) \mid v_1 \in V_1 \text{ and } v_2 \in V_2\} \text{ and}$$

$$E = \{((u_1, u_2), (v_1, v_2)) \mid (u_1, v_1) \in E_1 \text{ and } u_2 = v_2 \text{ or } (u_2, v_2) \in E_2 \text{ and } u_1 = v_1\}.$$

Next, both the torus and the k-ary n-cube are defined. Each is defined in two ways: first using Lee Distance and second as the cross product of cycles.

The Torus

Let $\mathcal{T}$ be an n-dimensional torus defined over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$. $\mathcal{T}$ is denoted by $T_{k_{n-1},k_{n-2},\cdots,k_0}$ or by $T_{\mathbf{K}}$ and is defined as follows. Let $N = \prod_{i=0}^{n-1} k_i$. $\mathcal{T}$ is a graph with N nodes, numbered $0 \ldots N-1$. Each node of $\mathcal{T}$ is labeled with a mixed radix, n-dimensional vector over $\mathbf{K}$ called its address. Given two nodes of $\mathcal{T}$, $\mathbf{X}$ and $\mathbf{Y}$, with addresses $x_{n-1}x_{n-2}\cdots x_0$ and $y_{n-1}y_{n-2}\cdots y_0$ respectively, the node numbers of $\mathbf{X}$ and $\mathbf{Y}$ are given by $I(\mathbf{X})$ and $I(\mathbf{Y})$ respectively; and there is an edge between $\mathbf{X}$ and $\mathbf{Y}$ if and only if $D_L(\mathbf{X}, \mathbf{Y}) = 1$. In addition, let D be the diameter of $\mathcal{T}$ and L be the number of links in $\mathcal{T}$, then

$$D = \sum_{i=0}^{n-1} \left\lfloor \frac{k_i}{2} \right\rfloor \qquad \text{and} \qquad L = n \prod_{i=0}^{n-1} k_i$$

A bipartite torus is defined over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$, where k_i is even for $0 \le i \le n-1$. It can be seen that the graph of such a torus is bipartite by grouping the nodes of the torus into two sets—those with even Lee weight and those with odd Lee weight—and noticing that each edge of the graph connects a node from one set with a node in the other. When a graph is bipartite, all cycles in the graph have even length.

Alternately, a torus can be described as the cross product of cycles. Let C_k be a cycle of length k. Then,

$$T_{k_{n-1}k_{n-2}\cdots k_0} \equiv C_{k_{n-1}} \otimes C_{k_{n-2}} \otimes \cdots \otimes C_{k_0}.$$

The k-ary n-cube

Let Q be a k-ary n-cube, denoted by Q_n^k. Q is a special case of an n-dimensional torus defined over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$ where each dimension has the same radix k. Therefore, Q has k^n nodes; and, when $k > 2$, the diameter, D, and the number of links, L, in Q are given by

$$D = n\left\lfloor \frac{k}{2} \right\rfloor \qquad \text{and} \qquad L = nk^n.$$

Since a Q_n^k is a torus, it can be described recursively as the cross product of n cycles of length k. That is,

$$Q_n^k \equiv \begin{cases} C_k & \text{if } n = 1 \\ C_k \otimes Q_{n-1}^k & \text{if } n > 1 \end{cases}$$

The binary hypercube of dimension n, Q_n, is the special case of a k-ary n-cube where $k = 2$. Therefore, in this thesis, a reference to a k-ary n-cube will assume that $k > 2$. Also note that if $\{\text{TORI}\}$ represents the set of all tori, $\{k\text{-ARY } n\text{-CUBES}\}$ the set of all k-ary n-cubes, and $\{\text{HYPERCUBES}\}$ the set of all hypercubes, then the following relation holds.

$$\{\text{HYPERCUBES}\} \subset \{k\text{-ARY } n\text{-CUBES}\} \subset \{\text{TORI}\}$$

2. TOPOLOGICAL PROPERTIES OF A TORUS

This chapter presents several topological properties of a torus. Since the graph of a $T_{\mathbf{K}}$ is more general than the graph of a Q_n^k, the results given in this chapter apply to a Q_n^k as well. In this chapter, Section 2.1 describes the number and the lengths of edge disjoint paths between two nodes in a torus; Section 2.2 describes a Hamiltonian cycle in a torus by means of a Gray code; and Section 2.3 shows how an even length cycle can be embedded in a torus using the same Gray code presented in Section 2.2.

2.1. Disjoint Paths

Given two nodes, $\mathbf{X}$ and $\mathbf{Y}$, in a torus, two paths, P_1 and P_2, between $\mathbf{X}$ and $\mathbf{Y}$ are disjoint if they have no edge in common and the only common nodes are $\mathbf{X}$ and $\mathbf{Y}$. When routing a message between $\mathbf{X}$ and $\mathbf{Y}$, it is helpful if there is more than one disjoint path between them. Among the reasons why are the following.

(1) Multiple paths allow for *adaptive* routing. This means that if one path between $\mathbf{X}$ and $\mathbf{Y}$ is congested, a message may take an alternate path.

(2) If a large amount of data must be transferred between $\mathbf{X}$ and $\mathbf{Y}$, it can be divided into small packets, each of which is sent using a different path. This reduces the time to transfer a large amount of data.

(3) If one path is unusable, either because of a failure in a link or a node, $\mathbf{X}$ and $\mathbf{Y}$ may still communicate with each other via a different path. This increases the *fault-tolerance* of the communication system.

This section presents a theorem stating the number and length of disjoint paths between two nodes in a torus. In the following, D_L denotes Lee distance and D_H denotes Hamming distance.

Theorem 2.1 *Let $\mathcal{T}$ be a $T_{\mathbf{K}}$ where $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$ and $k_i > 2$ for $0 \leq i \leq n-1$. Let $\mathbf{X} = x_{n-1}x_{n-2}\cdots x_0$ and $\mathbf{Y} = y_{n-1}y_{n-2}\cdots y_0$ be two distinct nodes of $\mathcal{T}$. Also, let $l = D_L(\mathbf{X}, \mathbf{Y})$, $h = D_H(\mathbf{X}, \mathbf{Y})$, and $w_i = D_L(x_i, y_i)$ for $0 \leq i \leq n-1$. Then, there are exactly $2n$ disjoint paths between $\mathbf{X}$ and $\mathbf{Y}$ in $\mathcal{T}$ of which*

(1) *h paths have length l,*

(2) *$2(n-h)$ paths have length $l+2$, and*

(3) *h paths are such that if $w_i > 0$, there is a path of length $l + k_i - 2w_i$.*

Proof: Without loss of generality, assume that the addresses of $\mathbf{X}$ and $\mathbf{Y}$ differ in dimensions 0 through $h-1$ and are the same in dimensions h through $n-1$. Then,

(1) The first h paths are constructed as follows. Beginning with dimension 0 and for each i, $0 \leq i < h$, the address of $\mathbf{X}$ is corrected using the shortest path in dimension i. This means successively incrementing (or decrementing) x_i modulo k_i, w_i times until it equals y_i. Repeat this procedure for the remaining digits of $\mathbf{X}$, proceeding sequentially through dimensions $i+1$, $i+2$, ..., $h-1$, 0, ..., $i-1$. Since $l = \sum_{i=0}^{h-1} w_i$, this path has length l. The remaining $h-1$ paths are constructed in a similar manner, but each path starts the correction procedure with a different dimension between 0 and $h-1$.

(2) The next $2(n-h)$ paths are constructed in the following manner. First, for each j, $h \leq j < n$, add one modulo k_j to x_j. Then, for $0 \leq i < h$, correct x_i as in (1). Complete the path by subtracting one modulo k_j from x_j. This procedure

produces $n-h$ paths of length $l+2$. To produce another $n-h$ paths of length $l+2$, repeat this procedure, but first subtract one modulo k_j from x_j and finish by adding one modulo k_j to x_j.

(3) The remaining h paths are constructed as follows. For each i, $0 \leq i < h$, first add or subtract one modulo k_i from x_i. The choice of addition or subtraction is the opposite of the choice made in (1). That is, the first step is along the *longest* path in dimension i. Next, the remaining digits of $\mathbf{X}$ are corrected using the shortest paths in dimensions $i+1, i+2, \ldots h-1, 0, \ldots i-1$. Finally, the path is completed by correcting dimension i along the longest path from x_i to y_i. The length of the path is calculated as follows. Correcting dimension i using the longest path from x_i to y_i takes $k_i - w_i$ steps. Correcting the remaining digits using the shortest paths take $l - w_i$ steps. Each path, therefore, has length $l - w_i + k_i - w_i$ or $l + k_i - 2w_i$.

It is evident that steps (1), (2), and (3) produce $2n - 2h + 2h = 2n$ paths and that the only nodes common to any path are $\mathbf{X}$ and $\mathbf{Y}$. ■

The following example demonstrates the construction of disjoints paths according to the procedure of Theorem 2.1.

Example 2.1 Let $\mathcal{T}$ be a $T_{6,4,5,3}$, and let $\mathbf{X} = 0000$ and $\mathbf{Y} = 0131$. Then $D_L(\mathbf{X},\mathbf{Y}) = l = 4$, $D_H(\mathbf{X},\mathbf{Y}) = h = 3$, $w_0 = 1$, $w_1 = 2$, $w_2 = 1$, and $w_3 = 0$. The 8 disjoint paths between $\mathbf{X}$ and $\mathbf{Y}$ are the following.

Step	Path 1	Path 2	Path 3	Path 4	Path 5	Path 6	Path 7	Path 8
0	0000	0000	0000	0000	0000	0000	0000	0000
1	0001	0040	0100	1000	5000	0002	0010	0300
2	0041	0030	0101	1001	5001	0042	0110	0301
3	0031	0130	0141	1041	5041	0032	0111	0341
4	0131	0131	0131	1031	5031	0132	0121	0331
5				1131	5131	0131	0131	0231
6				0131	0131			0131

■

2.2. Embedding a Hamiltonian Cycle in a Torus

This section describes how a Hamiltonian cycle can be embedded in a torus. First, a function, f_1, is described. This function maps a vector from a general mixed radix notation to a Gray code representation. Property 1, below, lists two assumptions. Function f_1 and the results of the next section are based on this property.

Property 1: The vectors are n-dimensional and are defined over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$ where

(1) at least one k_i is even, for $0 \leq i \leq n-1$;

(2) the dimensions are relabeled so they are ordered as follows.

$$\overbrace{k_{n-1}\cdots k_\ell}^{\textit{even radices}} \qquad \overbrace{k_{\ell-1}\cdots k_0}^{\textit{odd radices}}$$

Let $\mathbf{R} = r_{n-1}r_{n-2}\ldots r_0$ and $\mathbf{G} = g_{n-1}g_{n-2}\ldots g_0$ be two vectors in mixed radix form over $\mathbf{K}$. Define $\overline{r_i} = k_i - 1 - r_i$, and let f_1 be the mapping such that

$f_1(\mathbf{R}) = \mathbf{G}$, where f_1 is defined as follows.

$$g_{n-1} = r_{n-1}, \text{ and}$$

for even radices:

$$g_i = \begin{cases} r_i, \text{ if } r_{i+1} \equiv 0 \bmod 2 \\ \overline{r_i}, \text{ otherwise} \end{cases}$$

for odd radices:

$$g_i = \begin{cases} r_i, \text{ if } \sum_{j=i+1}^{\ell} r_j \equiv 0 \bmod 2 \\ \overline{r_i}, \text{ otherwise} \end{cases}$$

The inverse mapping, f_1^{-1}, is defined in the following manner.

$$r_{n-1} = g_{n-1}, \text{ and, for } 0 \le i < n-1,$$

$$r_i = \begin{cases} g_i, \text{ if } \sum_{j=i+1}^{n-1} g_i \text{ is even} \\ \overline{g_i}, \text{ otherwise} \end{cases}$$

The following theorem shows that f_1 produces a Gray code. In the proof of the theorem, the following definitions are used.

- Let $\mathbf{K}$ be the vector $k_{n-1}k_{n-2}\cdots k_0$.
- Let $\mathbf{0}$ the vector $0_{n-1}\cdots 0_0$.
- Let $\mathbf{Z}$ the vector $(k_{n-1}-1)(k_{n-2}-1)\cdots(k_0-1)$.
- Let $N = I(\mathbf{Z}) + 1$.
- Let S be the sequence $\langle \mathbf{0}, \ldots, \mathbf{Z} \rangle$ of N vectors in mixed radix form over $\mathbf{K}$. S is ordered such that the ith vector in the sequence has integer value $i - 1$.
- Let S' be the sequence $\langle f_1(\mathbf{0}), \ldots, f_1(\mathbf{Z}) \rangle$

Theorem 2.2 *Given the sequence S defined above, the sequence S' is a Gray code.*

Proof: In proving the theorem, two basic cases are considered. Case I shows that

$D_L(f_1(\mathbf{0}), f_1(\mathbf{Z})) = 1$. Case II shows that $D_L(f_1(\mathbf{X}), f_1(\mathbf{Y})) = 1$, where $\mathbf{X}$ and $\mathbf{Y}$ are adjacent in S.

Case I: Note that

$$\begin{aligned} f_1(\mathbf{0}) &= f_1(00\cdots 0) = 00\cdots 0 \text{ and that} \\ f_1(\mathbf{Z}) &= f_1\Big((k_{n-1}-1)(k_{n-2}-1)\cdots(k_0-1)\Big) \\ &= (k_{n-1}-1)(\overline{k_{n-2}-1})\cdots(\overline{k_\ell-1})(\overline{k_{\ell-1}-1})\cdots(\overline{k_0-1}). \end{aligned}$$

But, $\overline{k_i-1} = k_i - 1 - (k_i - 1) = 0$. Therefore, $f_1(\mathbf{Z}) = (k_{n-1}-1)0\cdots 0$ and $D_L(f_1(\mathbf{0}), f_1(\mathbf{Z})) = 1$.

Case II: Let $\mathbf{X}$ and $\mathbf{Y}$ be two vectors adjacent in the sequence S such that $I(\mathbf{Y}) = I(\mathbf{X}) + 1$. Also, let m be the index of the first position from the left in which $\mathbf{X}$ and $\mathbf{Y}$ differ, and express $\mathbf{X}$ and $\mathbf{Y}$ as

$$\begin{array}{rclllcll} \mathbf{X} &=& \{x_{n-1}x_{n-2}\cdots x_{m+1}\}^* & x_m & \{\ (k_{m-1}-1) & \cdots & (k_0-1) & \}^* \\ \mathbf{Y} &=& \{x_{n-1}x_{n-2}\cdots x_{m+1}\}^* & x_m+1 & \{\qquad 0 & \cdots & \quad 0 & \}^* \end{array}$$

where the segments marked by a "$*$" may or may not exist depending on the value of m.

Let $f_1(\mathbf{X}) = a_{n-1}a_{n-2}\cdots a_0$ and $f_1(\mathbf{Y}) = b_{n-1}b_{n-2}\cdots b_0$. $f_1(\mathbf{X})$ and $f_1(\mathbf{Y})$ are considered in three segments: (a) between dimensions $n-1$ and $m+1$, (b) dimension m, and (c) between dimensions $m-1$ and 0. It is shown below that $a_i = b_i$ for $i \neq m$ and that $D_L(a_m, b_m) = 1$. This shows that $D_L(f_1(\mathbf{X}), f_1(\mathbf{Y})) = 1$.

(a) $[n-1 \geq i \geq m+1]$ For this range, $x_i = y_i$; therefore, $a_i = b_i$.

(b) $[i = m]$ Also because of (a), either $a_m = x_m$ and $b_m = x_m + 1$ or $a_m = \overline{x_m}$ and $b_m = \overline{x_m + 1}$.

$$\begin{aligned} \text{Further,} \quad \overline{x_m} &= k_m - 1 - x_m \quad \text{and} \\ \overline{x_m + 1} &= k_m - 1 - (x_m + 1) \\ &= (k_m - 1 - x_m) - 1 \\ &= \overline{x_m} - 1. \end{aligned}$$

$$\begin{aligned} \text{Note that} \quad D_L(x_m, x_m + 1) &= x_m + 1 - x_m &&= 1 \quad \text{and that} \\ D_L(\overline{x_m}, \overline{x_m + 1}) &= \overline{x_m} - (\overline{x_m} - 1) &&= 1. \end{aligned}$$

Therefore, in either case, $D_L(a_m, b_m) = 1$.

(c) $[m - 1 \geq i \geq 0]$ Now two cases are considered: (1) $m \geq \ell$ and (2) $m < \ell$. Each case considers whether x_m is even or odd, and, for both cases, it is shown that $a_i = b_i$ for $m > i \geq 0$.

(1) This case is considered in three parts: (i) $i = m - 1$, (ii) $m - 1 > i \geq \ell$, and (iii) $\ell > i \geq 0$.

(i) $[i = m - 1$ and $m \geq \ell]$ Suppose that x_m is even; therefore, $x_m + 1$ is odd. This implies that

$$\begin{aligned} a_{m-1} &= k_{m-1} - 1 \quad \text{and that} \\ b_{m-1} &= \overline{0} \;=\; k_{m-1} - 1. \end{aligned}$$

On the other hand, suppose that x_m is odd; therefore, $x_m + 1$ is even. In this case,

$$\begin{aligned} a_{m-1} &= \overline{k_{m-1} - 1} \;=\; 0 \quad \text{and} \\ b_{m-1} &= 0. \end{aligned}$$

Thus, in either case, $a_{m-1} = b_{m-1}$

(ii) $[m-1 > i \geq \ell]$ For each i in this range, $x_i = k_i - 1$ is odd and $y_i = 0$ is even. Therefore, for each i in this range,

$$a_i = \overline{k_i - 1} = 0 = b_i.$$

(iii) $[\ell > i \geq 0$ and $m \geq \ell]$ Recall that k_i is even for $i \geq \ell$ and odd for $i < \ell$. This means that $x_\ell = k_\ell - 1$ is odd, while, for $\ell > i \geq 0$, $x_i = k_i - 1$ is even. Therefore, for each i, $\ell > i \geq 0$, $\sum_{j=i+1}^{\ell} x_j$ is odd and $a_i = \overline{x_i} = \overline{k_i - 1} = 0$. Also, for each i, $y_i = 0$; therefore, $\sum_{j=i+1}^{\ell} y_j = 0$ and $b_i = y_i = 0$.

Based on (i), (ii), and (iii) above, it can be seen that, when $m \geq \ell$, $a_i = b_i$ for $0 \leq i < m$.

(2) $[m < \ell]$ First, consider i, such that $m < i \leq \ell$. For this range, $x_i = y_i$; therefore, let

$$A = \sum_{i=m+1}^{\ell} x_i = \sum_{i=m+1}^{\ell} y_i$$

Now, consider i, such that $0 \leq i < m$. Since $m < \ell$, $x_i = k_i - 1$, while $y_i = 0$. This means that

$$B = \sum_{j=i+1}^{m-1} x_j \text{ is even and}$$

$$C = \sum_{j=i+1}^{m-1} y_j = 0.$$

Therefore, let

$$D = \sum_{j=i+1}^{\ell} x_j = A + x_m + B \text{ and}$$

$$E = \sum_{j=i+1}^{\ell} y_j = A + x_m + 1 + C$$

$$= A + x_m + 1.$$

Note that if D is even and E is odd, then

$$a_i = k_i - 1 = \overline{0} = b_i.$$

On the other hand, if D is odd and E is even, then

$$a_i = \overline{k_i - 1} = 0 = b_i.$$

Since B is even, there are four cases to be considered: (i) A is even and x_m is even; (ii) A is even and x_m is odd; (iii) A is odd and x_m is even; and (iv) A is odd and x_m is odd. For each case, it is clear that D and E have opposite parity. That is

case	A	x_m	D	E
i	even	even	even	odd
ii	even	odd	odd	even
iii	odd	even	odd	even
iv	odd	odd	even	odd

As the table above shows, when $m < \ell$, D and E have opposite parity; therefore, $a_i = b_i$ for $0 \leq i < m$.

Thus, the three cases above, (a), (b), and (c), combine to show that $a_i = b_i$, when $i \neq m$ and that $D_L(a_m, b_m) = 1$. This shows that $D_L(f_1(\mathbf{X}), f_1(\mathbf{Y})) = 1$ and the theorem is true. ■

As Example 2.2 illustrates, the sequence S', produced by Theorem 2.2, can be used to generate a Hamiltonian cycle in any torus, $\mathcal{T}$, meeting Property 1.

Example 2.2 Given the torus $T_{2,3,3}$, find a Hamiltonian cycle.

In this example, $\mathbf{R}$ is a vector in mixed radix form over $\mathbf{K} = 233$, $I(\mathbf{R})$ is the integer value of $\mathbf{R}$, and $f_1(\mathbf{R})$ is the Gray code vector mapped to $\mathbf{R}$ by Theorem 2.2.

$I(\mathbf{R})$	$\mathbf{R}$	$f_1(\mathbf{R})$	$I(\mathbf{R})$	$\mathbf{R}$	$f_1(\mathbf{R})$	$I(\mathbf{R})$	$\mathbf{R}$	$f_1(\mathbf{R})$
0	000	000	6	020	020	12	110	110
1	001	001	7	021	021	13	111	111
2	002	002	8	022	022	14	112	112
3	010	012	9	100	122	15	120	102
4	011	011	10	101	121	16	121	101
5	012	010	11	102	120	17	122	100

■

Even if a torus does not meet Property 1, but has at least one dimension with a even radix, Theorem 2.2 can still be used to find a Hamiltonian cycle. First, a dimensional permutation is found such that the permuted dimensions satisfy Property 1. The Hamiltonian cycle is found based on the permutation, and then the inverse permutation is applied to the cycle. For example, to find a Hamiltonian cycle in a $T_{5,6,4,7}$, the permutation which interchanges dimensions 1 and 3 may be used. Theorem 2.2 is used to find a Hamiltonian cycle in a $T_{4,6,5,7}$. Finally, dimensions 1 and 3 are interchanged again, and the result is a Hamiltonian cycle in a $T_{5,6,4,7}$.

2.3. Embedding an Even Length Cycle in a Torus

An additional benefit of Theorem 2.2 is its usefulness in finding non-Hamiltonian cycles of even length in a torus. This section describes a simple method for embedding an even length cycle in a torus that meets Property 1. Before describing this method, however, the reflective quality of the Gray code produced by Theorem 2.2 is discussed.

Let $N = 2m$ for some integer m, and let $S = \langle G_0, G_1, \cdots, G_{N-1}\rangle$ be a Gray code sequence of length N. S is called *reflective* if $D_L(G_i, G_{N-1-i}) = 1$ for

$i = 0, 1, \ldots m-1$. The familiar Binary Reflected Gray code is an example. However, when a code is based on a radix greater than 4, it is not, in general, reflective when measured with Lee distance. A Gray code sequence, S, is called *block reflective* for block length L [19] if it can be divided into blocks $B_0, B_1, \ldots, B_{\frac{N}{L}-1}$, each of length L, such that adjacent blocks are reflective. That is, block B_i reflects with block B_{i+1} if their concatenation forms a sequence of length N where $N = 2L$ and this sequence is reflective. If blocks B_0 and $B_{\frac{N}{L}-1}$ do not reflect with each other, but all other adjacent blocks are reflective, the sequence is called *partially block reflective.*

An examination of the Gray code produced by Theorem 2.2 shows that it is block reflective for block length $L = \prod_{i=0}^{n-2} k_i$. That is, the Gray code can be divided into k_{n-1} blocks, each of length L and each denoted by B_i, where i is leftmost digit of the vectors contained in the block. Then, block B_i reflects with block B_{i+1}, where the addition is modulo k_{n-1}.

Using the idea of block-reflectivity, the following method describes how to find an even length cycle in a torus. Let $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$, and $\mathcal{T}$ be a $T_{\mathbf{K}}$, and assume that $\mathcal{T}$ meets Property 1. In addition,

- let $N = \prod_{i=0}^{n-1} k_i$;
- let ℓ be an even number such that $2 \leq \ell \leq N$; and
- let S be the Gray code sequence calculated by Theorem 2.2.

To find a cycle of length ℓ in $\mathcal{T}$, do the following. First, divide ℓ by k_{n-1} and write $\ell = a_1(k_{n-1}) + a_0$, where

(1) $0 \leq a_1 \leq \prod_{i=0}^{n-2} k_i$;

(2) $0 \leq a_0 < k_{n-1}$; and

(3) $a_1 + \frac{a_0}{2} \leq \prod_{i=0}^{n-2} k_i$.

In a few cases, condition (3) may not be met, but it is explained below how such cases are handled.

Now, the cycle consists of $\frac{a_0}{2}$ vectors from the first and last blocks of S, together with a_1 additional vectors from each of the $B_{k_{n-1}-1}$ blocks of S. These vectors are chosen in the following manner, which is not the only way possible. Starting with $00\ldots0$, $\frac{a_0}{2} + a_1$ vectors are chosen. Then skipping to the reflection in block B_1 of the last vector chosen, another a_1 consecutive vectors are chosen. This process, choosing a_1 consecutive vectors in block B_i and then skipping to the reflection in block B_{i+1} of the last vector chosen, is continued until block $B_{k_{n-1}-1}$ is reached. The choices in the last block are the mirror images of those in the first: a_1 consecutive vectors are chosen plus an additional $\frac{a_0}{2}$ vectors. This process ends at vector $(k_{n-1} - 1)0\ldots0$, which is adjacent to vector $00\ldots0$. The next example illustrates this procedure.

Example 2.3 Find a cycle of length 14 in a $T_{6,5,3}$.

Note that $14 = 2(6) + 2$. In the following table, n is the position in the cycle and G is the node address.

n	G	n	G	n	G	n	G	n	G	n	G
1	000									14	500
2	001	5	101	6	201	9	301	10	401	13	501
3	002	4	102	7	202	8	302	11	402	12	502

■

In a few cases, $a_1 + \frac{a_0}{2} > \prod_{i=0}^{n-2} k_i$; see condition (3) above. This requires a slight manipulation of the expression. Since $a_0 < k_{n-1}$, let $k_{n-1} = a_0 + x$. Then the expression

$$\ell = a_1(k_{n-1}) + a_0 \text{ is rewritten as}$$

$$\ell = a_1(x + a_0) + a_0 = (a_1 + 1)(a_0) + a_1(x)$$

and is interpreted as the following. Choose $a_1 + 1$ vectors from the first a_0 blocks and a_1 vectors from the remaining x blocks. This is illustrated by the next example.

Example 2.4 Find a cycle of length 88 in a $T_{6,5,3}$. $88 = 14(6) + 4$; however, $14 + 2 > 15$. Rewriting the expression gives, $88 = 14(4 + 2) + 4 = 15(4) + 14(2)$. The cycle then consists of all 15 vectors from the first 4 blocks plus 14 vectors each from the remaining 2 blocks.

n	G	n	G	n	G	n	G	n	G	n	G
1	000	30	100	31	200	60	300	61	400	88	500
2	001	29	101	32	201	59	301	62	401	87	501
3	002	28	102	33	202	58	302	63	402	86	502
4	012	27	112	34	212	57	312	64	412	85	512
5	011	26	111	35	211	56	311	65	411	84	511
6	010	25	110	36	210	55	310	66	410	83	510
7	020	24	120	37	220	54	320	67	420	82	520
8	021	23	121	38	221	53	321	68	421	81	521
9	022	22	122	39	222	52	322	69	422	80	522
10	032	21	132	40	232	51	332	70	432	79	532
11	031	20	131	41	231	50	331	71	431	78	531
12	030	19	130	42	230	49	330	72	430	77	530
13	040	18	140	43	240	48	340	73	440	76	540
14	041	17	141	44	241	47	341	74	441	75	541
15	042	16	142	45	242	46	342				

■

Cycles of small lengths may result in $a_1 = 0$. In this case, the first $\frac{a_0}{2}$ vectors are chosen from block B_0 together with the last $\frac{a_0}{2}$ vectors in block $B_{k_{n-1}-1}$. For completeness, the next example illustrates this.

Example 2.5 Find a cycle of length 4 in a $T_{6,5,3}$.

Since $4 = 0(6) + 4$, the cycle consists of vectors from the first and last blocks: 000, 001, 501, and 500. ■

3. TOPOLOGICAL PROPERTIES OF A k-ARY n-CUBE

In this chapter, several topological properties of a k-ary n-cube (Q_n^k) are presented. Because a Q_n^k has the same radix, k, in each dimension, the results in this chapter apply to a Q_n^k rather than the more general $T_{\mathbf{K}}$.

Section 3.1 presents a formula for counting the number of nodes whose distance from a given node is d, where $d < \frac{k}{2}$. Section 3.2 introduces three functions that produce a Lee distance Gray code given a sequence of radix k numbers. These Gray codes are then used to find a Hamiltonian cycle in a Q_n^k. And, Section 3.3 discusses the embedding of a mesh and a hypercube into a Q_n^k given certain restrictions.

3.1. The Surface Area and Volume of a k-ary n-cube

Given a Q_n^k, the *surface area of a sphere of radius* d is the number of nodes whose Lee distance from a given node, the center, is exactly d. The *volume of a sphere of radius* d for the same Q_n^k is the number of nodes whose Lee distance from a given node is less than or equal to d. Since the distance used in both the surface area and the volume is Lee distance, it is assumed that $d < \frac{k}{2}$.

In this section, a formula is given for both the volume and surface area. Previously, Golomb and Welch [52] gave a closed form expression for the volume. In addition, Berlekamp [11] has given a generating function, and Reed and Fujimoto [84] (also in [85]) have given a recurrence relation for the surface area. The formula of this section, however, is a closed form for the surface area. The formula for surface area is given as a theorem, and the formula for volume follows as a simple corollary. This section begins, however, with a lemma concerning the number of ways an address may be obtained.

Lemma 3.1 *Let d and n be natural numbers such that $d > 2$ and $1 \leq n \leq d$. Let $A = \langle a_n a_{n-1} \ldots a_1 \rangle$ be a vector such that*

(1) *for each i, $1 \leq i \leq n$, $1 \leq a_i \leq d-(n-1)$*

(2) $d = \sum_{i=1}^{n} a_i$

Let $P(d,n)$ be the number of such vectors for a given d and n. Then

$$P(d,n) = \binom{d-1}{n-1}$$

Proof: The proof is by induction on n, and it uses the following standard formula for binomial coefficients [55, p. 359].

$$\sum_{i=m}^{n} \binom{i}{m} = \binom{n+1}{m+1}$$

Clearly, when $n = 1$, $P(d,1) = 1 = \binom{d-1}{0}$ and the vector A is $\langle d \rangle$.

Base Case: $n = 2$. The only possible vectors for A are:

$$\langle 1 \; d-1 \rangle$$
$$\langle 2 \; d-2 \rangle$$
$$\vdots$$
$$\langle d-1 \; 1 \rangle$$

Note that there are exactly $d - 1 = \binom{d-1}{1}$ such vectors.

Induction Hypothesis: Assume $P(d,n) = \binom{d-1}{n-1}$ for all $n < m$, where $m \leq d$.

Induction Step: Consider $P(d,m)$ and note that a_m, the leftmost component of A, can take on values from 1 to $d-(m-1)$. When $a_m = i$, $1 \leq i \leq d-(m-1)$, the remaining $m-1$ components of A must sum to $d-i$, and there are $P(d-i, m-1)$ such vectors. Therefore,

$$P(d,m) = \sum_{i=1}^{d-(m-1)} P(d-i, m-1).$$

Now, let $j = d-1-i$, then

$$1 \le i \le d-(m-1) \iff m-2 \le j \le d-2 \quad \text{and}$$

$$P(d,m) = \sum_{j=m-2}^{d-2} P(j+1, m-1).$$

By the induction hypothesis, however,

$$P(j+1, m-1) = \binom{j}{m-2}$$

Therefore,

$$P(d,m) = \sum_{j=m-2}^{d-2} \binom{j}{m-2} = \binom{d-2+1}{m-2+1} = \binom{d-1}{m-1} \qquad \blacksquare$$

Next, the formula for the surface area of a sphere of radius d is given as a theorem. The proof of the theorem is based on the following observation. Given two nodes, $\mathbf{X}$ and $\mathbf{Y}$ of a Q_n^k, $D_L(\mathbf{X}, \mathbf{Y}) = W_L(\mathbf{X} - \mathbf{Y})$, where the subtraction is componentwise and modulo k. Therefore, if $d < \frac{k}{2}$, calculating the number of nodes at a distance d from a given node is equivalent to calculating the number of distinct n-dimensional vectors of weight d. By weight, of course, it is meant Lee weight based on a radix k.

Theorem 3.1 [Surface Area] *Let $A_n^k(d)$ be the number of nodes in a Q_n^k whose distance from a given node is exactly d where $d < \frac{k}{2}$. That is, $A_n^k(d)$ is the surface area of a sphere of radius d. Then*

$$A_n^k(d) = \sum_{i=1}^{\min(d,n)} \binom{d-1}{i-1} \binom{n}{i} 2^i$$

Proof: Let $\mathbf{X}$ be the given node of the Q_n^k, the center of the sphere. The proof consists of counting the number of n-dimensional vectors of Lee weight d. This is because if $\mathbf{D}$ is a vector such that $W_L(\mathbf{D}) = d$, then $\mathbf{Y} = \mathbf{X} + \mathbf{D}$ is a node such that $D_L(\mathbf{X}, \mathbf{Y}) = d$.

To count the number of vectors, the following procedure is used.

(1) First, count the vectors of Lee weight d having only positive, radix k components. That is, each component is between 1 and $\left\lfloor \frac{k}{2} \right\rfloor$. Call these vectors Type 1 vectors. The length of a Type 1 vector can vary between 1, which is the vector $\langle d \rangle$, and the minimum of n and d. Note that if $d < n$, then the longest Type 1 vector has a one in each dimension and has length d.

By Lemma 3.1, the number of vectors of Type 1 is

$$\sum_{i=1}^{\min(d,n)} \binom{d-1}{i-1}.$$

(2) Each vector of Type 1 has only positive components. The definition of Lee weight, however, does not discriminate between positive and negative components. This means that if $\mathbf{V} = v_i v_{i-1} \ldots v_1$ is a Type 1 vector and $\mathbf{V}'$ is obtained from $\mathbf{V}$ by replacing one or more component, v_j, with its negative, $k - v_j$, then the Lee weight of $\mathbf{V}'$ is also d. Therefore, for each Type 1 vector $\mathbf{V}$, there are 2^i vectors $\mathbf{V}'$ for a total of

$$\sum_{i=1}^{\min(d,n)} \binom{d-1}{i-1} 2^i$$

vectors of Lee weight d. Call these Type 2 vectors.

(3) Each vector of Type 2 has non-zero components, a Lee weight of d, and a length, i, between 1 and the minimum of n and d. The theorem, however, is about counting the number of vectors of length n having Lee weight d. A

vector of length n can be obtained from a Type 2 vector by adding $n-i$ zeros. These additional zeros do not change the weight of the vector, and there are $\binom{n}{i}$ different ways in which the zeros may be added. Therefore each Type 2 vector produces $\binom{n}{i}$ vectors of length n and Lee weight d.

Taking the three steps together, it can be seen that there are

$$\sum_{i=1}^{\min(d,n)} \binom{d-1}{i-1}\binom{n}{i}2^i$$

different vectors of length n and Lee weight d. When added to $\mathbf{X}$, each vector yields a vector $\mathbf{Y}$ which is at a Lee distance of d. This proves the theorem. ■

Corollary 3.1 [Volume] *Let $V_n^k(d)$ be the number of nodes in a Q_n^k whose distance from a given node is less than or equal to d where $d < \frac{k}{2}$. That is, $V_n^k(d)$ is the volume of a sphere of radius d. Then*

$$V_n^k(d) = 1 + \sum_{i=1}^{d} A_n^k(i)$$

.

Proof: The number of nodes at a distance less than or equal to d from a given node, $\mathbf{X}$, is the number of nodes at a distance 0, which is 1—the node itself—plus the number of nodes at distances 1, 2, ..., d. But, this is exactly the expression given by the corollary. ■

3.2. Embedding a Hamiltonian Cycle in a k-ary n-cube

In Section 2.2, the function f_1, mapping a sequence in mixed radix notation to a Gray code sequence, was presented. This function was used to find a Hamiltonian cycle in a torus provided at least one dimension had an even radix. In addition, the Gray code produced by this function could be used to find a cycle of any even length less than that of the Hamiltonian cycle.

In this section, three more functions that map from a sequence of n digit, radix k numbers to a Gray code sequence are given. The functions presented in this section are simpler that the function presented in Section 2.2, but they are not designed for numbers in a mixed radix notation. The first function is presented in a theorem. Because the proof of the next two is similar, they are defined, but not stated as theorems.

Theorem 3.2 *Given* $f_2 : \{0 \ldots k-1\}^n \rightarrow \{0 \ldots k-1\}^n$ *where*

$$f_2(a_{n-1}a_{n-2} \cdots a_0) = a_{n-1}(a_{n-2} - a_{n-1}) \cdots (a_0 - a_1)$$

and a sequence of n digit, radix k numbers

$$\langle(0\ldots00),(0\ldots01)\cdots(0\ldots0(k-1)),(0\ldots10)\cdots((k-1)\ldots(k-1)(k-1))\rangle$$

then the sequence

$$\langle f_2(0\ldots00), f_2(0\ldots01)\cdots f_2(0\ldots0(k-1)), f_2(0\ldots10)\cdots f_2((k-1)\ldots(k-1)(k-1))\rangle$$

forms a Gray code.

Proof: Let $\mathbf{A} = a_{n-1}a_{n-2}\cdots a_0$ and $\mathbf{B} = b_{n-1}b_{n-2}\cdots b_0$ where $\mathbf{B} = \mathbf{A}+1 \bmod k^n$. Further, let m be the index of the first digit from the left for which $\mathbf{A}$ and $\mathbf{B}$ differ. Then,

$$\begin{aligned} \mathbf{A} &= \{a_{n-1}a_{n-2}\cdots a_{m+1}\}^* \quad a_m \quad \{(k-1)(k-1)\cdots(k-1)\}^* \text{ and} \\ \mathbf{B} &= \{a_{n-1}a_{n-2}\cdots a_{m+1}\}^* \quad (a_m+1) \bmod k \quad \{00\cdots 0)\}^* \end{aligned}$$

where sequences marked with $*$ may or may not exist depending on the value of m. Then,

$$\begin{aligned}
f_2(\mathbf{A}) &= \{x_{n-1}x_{n-2}\cdots x_{m+1}\}^*\ x_m\ \{x_{m-1}00\cdots 0\}^*, \text{ where} \\
x_{n-1} &= a_{n-1}, \\
x_i &= (a_i - a_{i+1}) \bmod k \text{ for } i = n-2 \ldots m+1, \\
x_m &= a_m - a_{m+1}, \text{ and} \\
x_{m-1} &= ((k-1) - a_m) \bmod k. \\
f_2(\mathbf{B}) &= \{x_{n-1}x_{n-2}\cdots x_{m+1}\}^*\ y_m\ \{y_{m-1}00\cdots 0\}^*, \text{ where} \\
y_m &= (a_m + 1) - a_{m+1} \\
&= (x_m + 1) \bmod k, \text{ and} \\
y_{m-1} &= 0 - (a_m + 1) \\
&= (k - 1 - a_m) \bmod k \\
&= x_{m-1}.
\end{aligned}$$

Therefore, $f_2(\mathbf{A})$ and $f_2(\mathbf{B})$ differ in the m^{th} digit only and $D_L(f_2\mathbf{A}), f_2(\mathbf{B})) = 1$. This implies that the mapping produced by f_2 results in a Gray code, and the theorem is proved. ■

Using Theorem 3.2, a Hamiltonian cycle can be generated for any Q_n^k. First the sequence

$$S = \langle 0, 1, \ldots, k^n - 1\rangle \text{ radix } k$$

is generated. Then the sequence

$$S' = \langle f_2(0), f_2(1), \ldots, f_2(k^n - 1)\rangle$$

is obtained, giving a Hamiltonian cycle. Two examples are given below. In Example 3.1, a Hamiltonian cycle is given for a Q_2^4, and in Example 3.2 a Hamiltonian cycle for a Q_2^5 is given.

Example 3.1 Find a Hamiltonian cycle in a Q_2^4.

Let **R** represent the radix representation of a node address and $f_2(\mathbf{R})$ the transformation under the mapping f_2 defined in Theorem 3.2. Then

R	$f_2(\mathbf{R})$	**R**	$f_2(\mathbf{R})$	**R**	$f_2(\mathbf{R})$	**R**	$f_2(\mathbf{R})$
00	00	10	13	20	22	30	31
01	01	11	10	21	23	31	32
02	02	12	11	22	20	32	33
03	03	13	12	23	21	33	30

■

Example 3.2 Find a Hamiltonian cycle in Q_2^5.

R	$f_2(\mathbf{R})$	**R**	$f_2(\mathbf{R})$	**R**	$f_2(\mathbf{R})$	**R**	$f_2(\mathbf{R})$	**R**	$f_2(\mathbf{R})$
00	00	10	14	20	23	30	32	40	41
01	01	11	10	21	24	31	33	41	42
02	02	12	11	22	20	32	34	42	43
03	03	13	12	23	21	33	30	43	44
04	04	14	13	24	22	34	31	44	40

■

Given the function f_2 defined in Theorem 3.2, the inverse function, f_2^{-1} can be obtained in the following way. Let $\mathbf{G} = f_2(\mathbf{R})$ where $\mathbf{R} = r_{n-1}r_{n-2}\cdots r_0$ is a radix k number and $\mathbf{G} = g_{n-1}g_{n-2}\cdots g_0$ is its corresponding Gray code representation. Then,

$$\begin{aligned} \mathbf{R} &= f_2^{-1}(\mathbf{G}) && \text{where} \\ r_{n-1} &= g_{n-1} && \text{and} \\ r_i &= (r_{i+1} + g_i) \bmod k \\ &= \left(\textstyle\sum_{j=i}^{n-1} g_i\right) \bmod k && \text{for } i = n-2, n-3, \ldots, 0 \end{aligned}$$

This is because $f_2(r_i) = g_i = r_i - r_{i+1}$, and, therefore,

$$\begin{aligned} r_i &= r_{i+1} + g_i \\ &= (r_{i+2} + g_{i+1}) + g_i \\ &\vdots \qquad\quad \vdots \\ &= r_{n-1} + g_{n-2} + \cdots + g_i \\ &= g_{n-1} + g_{n-2} + \cdots + g_i \end{aligned}$$

Similar to function f_2, a function f_3 can be defined that produces a Hamiltonian cycle in a Q_n^k. The proof is similar to Theorem 3.2.

$$f_3(a_{n-1}a_{n-2}\cdots a_0) = a_{n-1}(a_{n-1} - a_{n-2})(a_{n-2} - a_{n-3})\cdots(a_1 - a_0)$$

Two examples follow. Examples 3.3 and 3.4 illustrate finding a Hamiltonian cycle in a Q_2^4 and a Q_2^5 respectively. These two examples contrast with Examples 3.1 and 3.2.

Example 3.3 Find a Hamiltonian cycle in a Q_2^4.

Let **R** represent the radix representation of a node address and $f_3(\mathbf{R})$ the transformation under the mapping f_3 defined in Theorem 3.2. Then

R	$f_3(\mathbf{R})$	**R**	$f_3(\mathbf{R})$	**R**	$f_3(\mathbf{R})$	**R**	$f_3(\mathbf{R})$
00	00	10	11	20	22	30	33
01	03	11	10	21	21	31	32
02	02	12	13	22	20	32	31
03	01	13	12	23	23	33	30

■

Example 3.4 Find a Hamiltonian cycle in Q_2^5.

R	f_3(**R**)	**R**	f_3(**R**)	**R**	f_3(**R**)	**R**	f_3(**R**)	**R**	f_3(**R**)
00	00	10	11	20	22	30	33	40	44
01	04	11	10	21	21	31	32	41	43
02	03	12	14	22	20	32	31	42	42
03	02	13	13	23	24	33	30	43	41
04	01	14	12	24	23	34	34	44	40

■

The next function presented, f_4, also generates a Gray code sequence from a radix k sequence. When k is even, the resulting Gray code forms a Hamiltonian cycle in a Q_n^k, but if k is odd, the Gray code forms a Hamiltonian path. Like function f_1, however, the resulting Gray code also has reflective qualities. If $k = 2$ or 4, the Gray code is reflective. Otherwise, if k is even, the code is block reflective for block lengths k, k^2, ..., k^{n-1}, and, when k is odd, the code is partially block reflective (see page 18) for the same block sizes.

To define f_4, let $\mathbf{R} = r_{n-1}r_{n-2}\cdots r_0$ be a radix k number and $\mathbf{G} = g_{n-1}g_{n-2}\cdots g_0$ be the corresponding Gray code image of **R** under the mapping f_4, i.e. $\mathbf{G} = f_4(\mathbf{R})$. Now, define

$$\overline{r_i} = k - 1 - r_i \qquad \text{and} \qquad r' = \sum_{j=i+1}^{n-1} r_j.$$

Then, $g_{n-1} = r_{n-1}$ and, for $i = n-2, \ldots, 0$, if k is even,

$$g_i = \begin{cases} r_i, & \text{if } r_{i+1} \text{ is even} \\ \overline{r_i}, & \text{otherwise} \end{cases}$$

or, if k is odd,

$$g_i = \begin{cases} r_i, & \text{if } r' \text{ is even} \\ \overline{r_i}, & \text{otherwise} \end{cases}$$

Two examples are given below. In Example 3.5, $k = 5$ and $n = 2$. In Example 3.6, $k = 4$ and $n = 3$.

Example 3.5 A 2 digit, 5-ary Gray code sequence which gives a Hamiltonian path and is partially block reflective.

R	$f_4(\mathbf{R})$	**R**	$f_4(\mathbf{R})$	**R**	$f_4(\mathbf{R})$	**R**	$f_4(\mathbf{R})$	**R**	$f_4(\mathbf{R})$
00	00	10	14	20	20	30	34	40	40
01	01	11	13	21	21	31	33	41	41
02	02	12	12	22	22	32	32	42	42
03	03	13	11	23	23	33	31	43	43
04	04	14	10	24	24	34	30	44	44

■

Since f_4 is block reflective for even values of k, f_4 can be used to find even length cycles in a Q_n^k when k is even. Recall that when k is even, the graph of a Q_n^k is bipartite, implying that all cycles are of even length.

The procedure for finding an even length cycle in Q_n^k is similar to that used in Section 2.3. To find a cycle of length ℓ in a Q_n^k, first divide ℓ by k and write $\ell = a_1(k) + a_0$, where

(1) $0 \leq a_1 \leq k^{n-1}$;

(2) $0 \leq a_0 < k$; and

(3) $a_1 + \frac{a_0}{2} \leq k$.

Then, as the Gray code sequence produced by f_4 is block reflective for a block size of k^{n-1}, the cycle consists of a_1 vectors from each of the k blocks together

Example 3.6 A 3 digit, 4-ary Gray code sequence which gives a Hamiltonian cycle and is block reflective.

R	$f_4(\mathbf{R})$	**R**	$f_4(\mathbf{R})$	**R**	$f_4(\mathbf{R})$	**R**	$f_4(\mathbf{R})$
000	000	100	130	200	200	300	330
001	001	101	131	201	201	301	331
002	002	102	132	202	202	302	332
003	003	103	133	203	203	303	333
010	013	110	123	210	213	310	323
011	012	111	122	211	212	311	322
012	011	112	121	212	211	312	321
013	010	113	120	213	210	313	320
020	020	120	110	220	220	320	310
021	021	121	111	221	221	321	311
022	022	122	112	222	222	322	312
023	023	123	113	223	223	323	313
030	033	130	103	230	233	330	303
031	032	131	102	231	232	331	302
032	031	132	101	232	231	332	301
033	030	133	100	233	230	333	300

■

with $\frac{a_0}{2}$ vectors from the first and last blocks. The vectors are chosen in the same manner as the procedure in Section 2.3.

3.3. Embedding a Mesh and a Hypercube

This section presents methods for embedding two popular topologies, the hypercube and the mesh, into a general Q_n^k. An embedding is important as it allows one topology to *simulate* another. This means that an algorithm, which has been optimized to run on one topology, can be used effectively on another. The remainder of this section defines what is meant by an embedding and, then, presents an embedding for a mesh and a hypercube.

3.3.1. Graph Embeddings

Let G and H be two undirected graphs where G is called the *guest* graph and H is called the *host* graph. Let V_G, E_G, V_H, and E_H denote the vertex and edge sets of G and H respectively, and let P_H denote the set of all paths in H. That is, $(x_1, x_2, \cdots, x_n) \in P_H$ if $x_i \in V_H$ and $(x_i, x_{i+1}) \in E_H$ for $1 \leq i < n$. Then an embedding of G in H is a pair (f_V, f_E) where f_V: $V_G \rightarrow V_H$ and f_E: $E_G \rightarrow P_H$. Also, if $(a, b) \in E_G$ then $f_E(a, b) = (x_1, \ldots, x_n)$ such that $(x_1, \ldots, x_n) \in P_H$, where $x_1 = f_V(a)$, and $x_n = f_V(b)$.

Given graphs G and H with an embedding (f_V, f_E) of G into H, the following terms are used to describe the embedding. For more information see [69].

Dilation The dilation of an embedding is the length of the longest path in H that is associated with an edge in G by f_E.

Expansion The expansion of an embedding is the ratio $\frac{|V_H|}{|V_G|}$ where $|V_G|$ denotes the cardinality of V_G.

Congestion The congestion of an embedding is the maximum number of times a single edge of H belongs to paths in H associated with edges in G by f_E.

Load The load of an embedding is the maximum number of vertices of G associated with a single vertex of H by f_V.

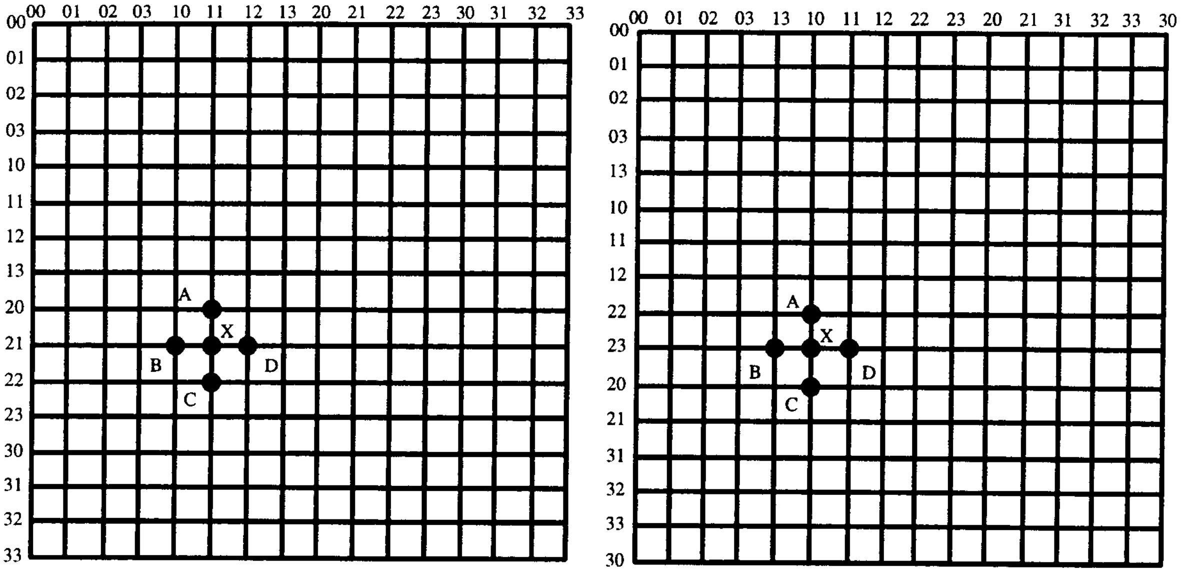
If an embedding of a graph G into a graph H can be found that has dilation, congestion and load equal to one, then G is isomorphic to a subgraph of H.

3.3.2. Embedding a Mesh into a Q_n^k

A $k_n \times k_{n-1} \times \cdots \times k_1$–mesh is an interconnection topology consisting of $\prod_{i=1}^{n} k_i$ nodes. Each node is labeled with an n-digit address $a_{n-1}a_{n-2}\cdots a_0$ where a_i is radix k_i. Each node has, at most, a connection to $2n$ other nodes. Given a node $\mathbf{A}$ with address $a_{n-1}\cdots a_i \cdots a_0$, for each $i, 0 \leq i < n$, $\mathbf{A}$ has a connection to the node with address $a_{n-1}\cdots a_i + 1 \cdots a_0$ if $a_i < k_i$ and to the node with address $a_{n-1}\cdots a_i - 1 \cdots a_0$ if $a_i > 0$.

The Gray code presented in Theorem 3.2 can be used to embed a mesh of certain dimensions into a Q_n^k. Let $\mathcal{M}$ be a $k^{n_1} \times k^{n_2} \times \cdots \times k^{n_m}$–dimensional mesh, and $\mathcal{Q}$ be a Q_n^k, where $n = \sum_{i=1}^{m} n_i$. The following construction shows how to embed $\mathcal{M}$ into $\mathcal{Q}$.

Assume that each dimension i of $\mathcal{M}$ is labeled with a radix k, n_i digit number $0 \ldots k^{n_i} - 1$. Using Theorem 3.2, relabel each dimension with the corresponding Gray code sequence. Now each node of $\mathcal{M}$ can be identified with an m-tuple whose i^{th} component is the node's location in the i^{th} dimension, $1 \leq i \leq m$. If $\mathbf{X}$ is a node of



(a) Sequential Labels

(b) Gray Code Labels

FIGURE 3.1. Sequential (a) and Gray code (b) labels in a $4^2 \times 4^2$ mesh.

$\mathcal{M}$ with label $(x_1, x_2, \cdots, x_m)$, then define $f_V(\mathbf{X}) = x_1 x_2 \cdots x_m$, the concatenation of $x_1, \ldots, x_m$.

It should be clear that if $\mathbf{X}$ and $\mathbf{Y}$ are any two adjacent nodes in $\mathcal{M}$, then $f_V(\mathbf{X})$ and $f_V(\mathbf{Y})$ are adjacent in $\mathcal{Q}$. For if $\mathbf{X}$ and $\mathbf{Y}$ are adjacent in $\mathcal{M}$, their addresses differ only in some dimension i. Since each dimension is labeled with a Gray code, the Lee distance between $\mathbf{X}$ and $\mathbf{Y}$ in dimension i is one. Therefore, $D_L(f_V(\mathbf{X}), f_V(\mathbf{Y})) = 1$, and $\mathbf{X}$ and $\mathbf{Y}$ are adjacent in $\mathcal{Q}$.

Example 3.7 As an example, Figure 3.1 shows $4^2 \times 4^2$-dimensional mesh. The mesh has both sequential and Gray code labels. The node $\mathbf{X}$ has address (21,11) which, when translated to the Gray code, becomes (23,10). In the mesh, $\mathbf{X}$ is adjacent to 4 other nodes: **A** (20,11), **B** (21,10), **C** (22,11), and **D** (21,12). When relabeled with the Gray code, these addresses become: **A** (22,10), **B** (23,13), **C** (20,10), and

D (23,11). After embedding the mesh into the Q_4^4, the addressed are: **X** (2310), **A** (2210), **B** (2313), **C** (2010), and **D** (2311). It is easily verified that the Lee distance between **X** and **A**, **B**, **C** or **D** is 1. Therefore, **X** is adjacent to the other 4 nodes in the Q_4^4. ■

3.3.3. Embedding a Hypercube into a Q_n^k

This section considers embedding a Q_n in a $Q^4_{\lceil \frac{n}{2} \rceil}$. To begin, consider the following lemma.

Lemma 3.2 *Let $f : \{0,1\}^2 \longrightarrow \{0,1,2,3\}$ where $f(00) = 0,\ f(01) = 1,\ f(10) = 3,$ and $f(11) = 2$. Then f maps the 2 digit binary reflective Gray code onto the single digit 4-ary reflective Gray code.*

Proof: The lemma can be verified by observation. First, the binary vectors $\{00, 01, 11, 10\}$ form the familiar 2-digit binary reflective Gray code. Second, if $\mathbf{U}, \mathbf{V} \in \{0,1\}^2$ and $D_L(\mathbf{U}, \mathbf{V}) = 1$, then it can be seen that $D_L(\mathrm{f}(\mathbf{U}), \mathrm{f}(\mathbf{V})) = 1$. ■

The procedure for embedding a hypercube into a k-ary n-cube is given in the next theorem. The theorem applies to the case where the dimension of the hypercube is even. A corollary follows that considers a hypercube having an odd dimension.

Theorem 3.3 *Let $\mathcal{H}$ be a Q_{2d} and $\mathcal{K}$ be a Q_d^4. Let $\mathbf{A} = (a_{2d}a_{2d-1} \cdots a_1)$ and $\mathbf{B} = (b_{2d}b_{2d-1} \cdots b_1)$ be vertices of $\mathcal{H}$ and, $\mathbf{A}' = (a'_d a'_{d-1} \cdots a'_1)$ and $\mathbf{B}' = (b'_d b'_{d-1} \cdots b'_1)$ be vertices of $\mathcal{K}$. Let $f_V(\mathbf{A}) = \mathbf{A}'$ where $a'_i = f(a_{2i}a_{2i-1})$ and f is the function defined in Lemma 3.2. Further, let $f_E(\mathbf{A}, \mathbf{B}) = (\mathbf{A}', \mathbf{B}') = (f_V(\mathbf{A}), f_V(\mathbf{B}))$. Then the pair (f_V, f_E) embeds $\mathcal{H}$ into $\mathcal{K}$.*

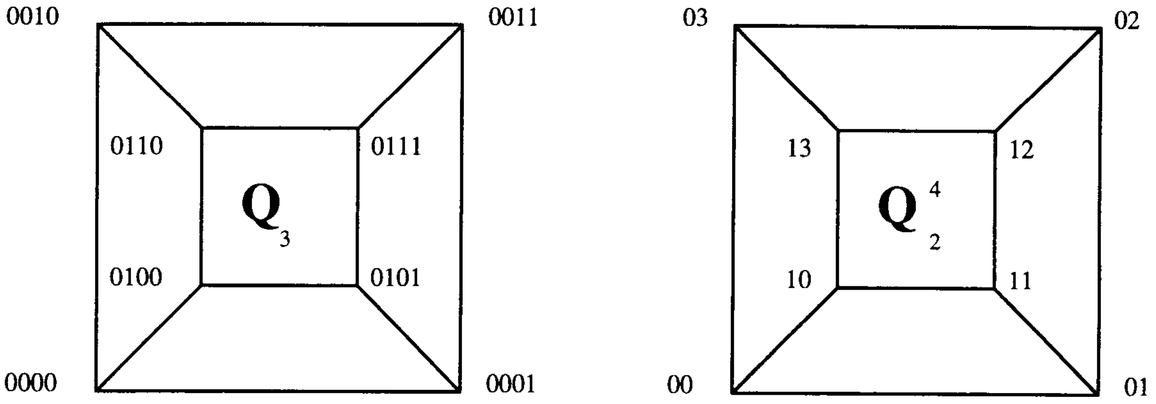


FIGURE 3.2. A Q_3 embedded into a Q_2^4

Proof: First note that $\mathcal{H}$ and $\mathcal{K}$ have the same number of vertices. The number of vertices of $\mathcal{K}$ is $4^d = 2^{2d}$ which is the number of vertices of $\mathcal{H}$. Also note that the function f_V, maps nodes from $\mathcal{H}$ onto nodes of $\mathcal{K}$ uniquely. Showing the pair (f_V, f_E) is an embedding requires showing that adjacent nodes in $\mathcal{H}$ are mapped to adjacent nodes in $\mathcal{K}$. Suppose **A** and **B** are adjacent in $\mathcal{H}$. Then the labels of **A** and **B** differ in some position, say j, $1 \leq j \leq 2d$. Without loss of generality, assume $j = 2p$. Then $D_L(a_j a_{j-1}, b_j b_{j-1}) = D_L(a_j a_{j-1}, \bar{a}_j a_{j-1}) = 1$. By Lemma 3.2, however, $D_L(f(a_j a_{j-1}), f(\bar{a}_j a_{j-1})) = D_L(a'_p, b'_p) = 1$. But, $\mathbf{A}'$ and $\mathbf{B}'$ differ only in position p; therefore, $D_L(\mathbf{A}', \mathbf{B}') = 1$, and $\mathbf{A}'$ and $\mathbf{B}'$ are adjacent in $\mathcal{K}$. ■

Corollary 3.2 *Suppose $\mathcal{H}$ is a Q_{2d-1} and $\mathcal{K}$ is a Q_d^4. Then using the notation of Theorem 3.3 with $f_V(\mathbf{A}) = \mathbf{A}'$ where*

$$a'_i = \begin{cases} f(0a_{2i-1}), & \text{if } i = d \\ f(a_{2i}a_{2i-1}), & \text{if } 1 \leq i < d \end{cases}$$

the pair (f_V, f_E) embeds $\mathcal{H}$ into $\mathcal{K}$.

Proof: The proof follows directly from Theorem 3.3. ■

Taken together, Lemma 3.2, Theorem 3.3, and Corollary 3.2 demonstrate how to embed a Q_n into a $Q^4_{\lceil \frac{n}{2} \rceil}$. Let the address label of node **A** be $(a_n a_{n-1} \cdots a_1)$. If n is odd, relabel **A** as $(0 a_n a_{n-1} \cdots a_1)$. Then let node $\mathbf{A}'$ of $Q^4_{\lceil \frac{n}{2} \rceil}$ have address label $(a'_j a'_{j-1} \cdots a'_1)$ where $j = \lceil \frac{n}{2} \rceil$, $a'_i = f(a_{2i} a_{2i-1})$ for $1 \leq i \leq j$, and f defined as in Lemma 3.2.

As an example, Figure 3.2 shows a Q_3 and its embedding into a Q^4_2. In the figure, since n is odd, the node labels of the Q_3 have a zero prepended.

4. DECOMPOSING A Q_n^k INTO DISJOINT HAMILTONIAN CYCLES

In Section 3.2, several methods of generating a Gray code sequence were presented. The resulting Gray code sequence was used to find a Hamiltonian cycle in a Q_n^k. As will be seen in Section 5.2.2, among the reasons for finding a Hamiltonian cycle in a Q_n^k is for use in a communication algorithm. When a Hamiltonian cycle is used in a communication algorithm, its effectiveness is increased if more than one edge-disjoint Hamiltonian cycle exists.

How many disjoint Hamiltonian cycles should exist in a Q_n^k? Corollary 4 of [5] states, "If $G_1, G_2, \ldots, G_p$ can all be decomposed into n Hamilton cycles, then $G_1 \times G_2 \times \ldots \times G_p$ can be decomposed into pn Hamilton cycles." Since a Q_n^k is the cross product of n cycles of length k and each cycle can be decomposed trivially into one Hamiltonian cycle, the corollary above implies that a Q_n^k can be decomposed into n disjoint Hamiltonian cycles.

Although the existence of disjoint Hamiltonian cycles in the cross product of various graphs has been discussed in the literature [28, 27, 96, 91, 92, 58, 13, 48], a straightforward way of generating such disjoint cycles in a Q_n^k is not evident. This section considers the problem of decomposing a Q_2^k and a Q_3^k into 2 and 3 disjoint Hamiltonian cycles respectively. The method described here relies on the definition of a Q_n^k as the cross product of n cycles of length k, and it is conjectured that this method can be extended to decompose any Q_n^k into n disjoint Hamiltonian cycles. First, however, a method that allows the decomposition of any Q_n^k into two disjoint Hamiltonian cycles is defined. For the remainder of this section, it is assumed that $k > 2$.

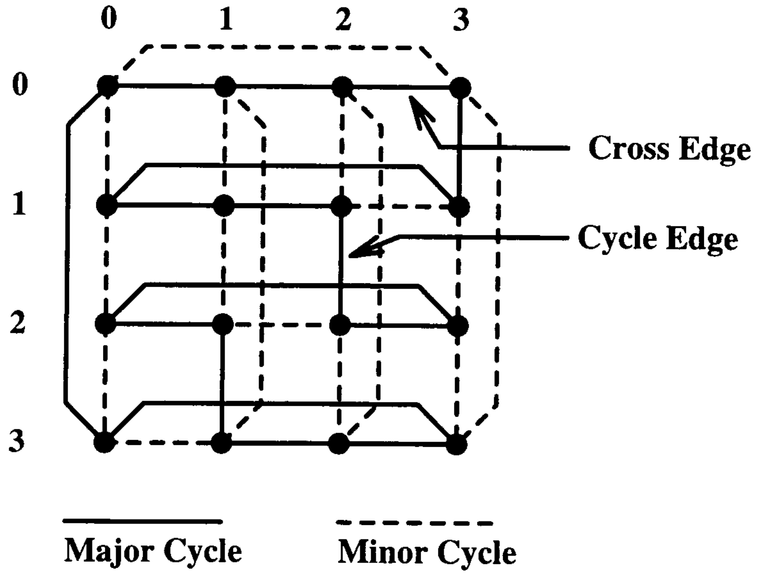


FIGURE 4.1. The standard decomposition of a Q_2^4

4.1. The Standard Decomposition

Let the node addresses of a Q_n^k be arranged in a $k^{n-1} \times k$ rectangular array. An edge in the horizontal direction is called a *cross* edge, and an edge in the vertical direction is called a *cycle* edge. Figure 4.1 shows a Q_2^4 decomposed into 2 Hamiltonian cycles. One cycle, represented by the solid line, has the pattern: $k-1$ cross edges followed by a cycle edge. This cycle is called the *major* cycle. The second cycle, represented by the dashed line, has the pattern: $k-1$ cycle edges followed by one cross edge, and it is called the *minor* cycle. The decomposition shown in Figure 4.1 is referred to as the *standard* decomposition.

The standard decomposition shows how any Q_2^k can be decomposed into two disjoint Hamiltonian cycles. Arranging the node addresses of a Q_2^k in a $k \times k$ square array and applying the standard decomposition, two disjoint Hamiltonian cycles are produced by the major and minor cycles. It is interesting to note that the

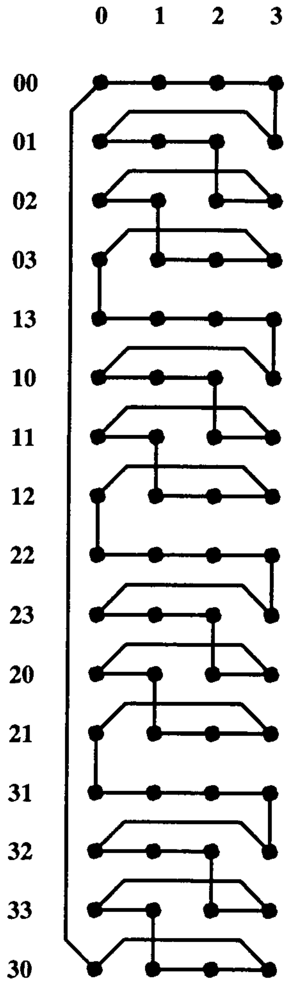


FIGURE 4.2. A Hamiltonian cycle (H_1) of a Q_3^4

major cycle of a Q_2^k is the Hamiltonian cycle produced by the function f_2 defined in Theorem 3.2. The minor cycle can be viewed as the major cycle with the dimensions interchanged.

The standard decomposition can also be used to decompose any Q_n^k into two disjoint Hamiltonian cycles. By arranging the node labels in a $k^{n-1} \times k$ rectangular array, the major and minor cycles of the standard decomposition produce two disjoint Hamiltonian cycles.

As an example, Figure 4.2 shows a Q_3^4 arranged as a 16×4 rectangle with the major cycle drawn. In this diagram, the numbers along the top edge are the first numbers of the node addresses, while the numbers down the left edge are the last two numbers of the node addresses. Note also that the sequence of numbers along the left edge is the Gray code sequence produced by the function f_2 for a two digit radix 4 sequence.

4.2. Decomposing a Q_3^k

The decomposition of a Q_3^k into 3 disjoint Hamiltonian cycles is accomplished by viewing the Q_3^k as the cross product of three cycles of length k. The resulting Hamiltonian cycles, however, are of length k^3. The standard decomposition of a Q_2^k gives two disjoint cycles of length k^2, and these two cycles can be obtained using the function f_2.

Let C_k denote a cycle of length k, and let C_M and C_m denote the major and minor cycles obtained by the standard decomposition of a Q_2^k. Forming the cross product $C_M \otimes C_k$ and taking the standard decomposition gives two disjoint Hamiltonian cycles of the Q_3^k. Let these two cycles be denoted by H_1 and H_2 where H_1 is the major cycle of the standard decomposition. See Figure 4.2.

A problem with finding the third Hamiltonian cycle, H_3 is now apparent: the cross product $C_m \otimes C_k$ does not form a disjoint Hamiltonian cycle. The cycle edges of $C_m \otimes C_k$ are disjoint from both H_1 and H_2 because they are determined by the sequence C_m while the cycle edges of H_1 and H_2 are determined by the sequence C_M, and C_M and C_m are disjoint. However, the cross edges of $C_m \otimes C_k$, H_1, and H_2 are all determined by the same sequence: C_k; and H_1 and H_2 use all the cross edges between themselves.

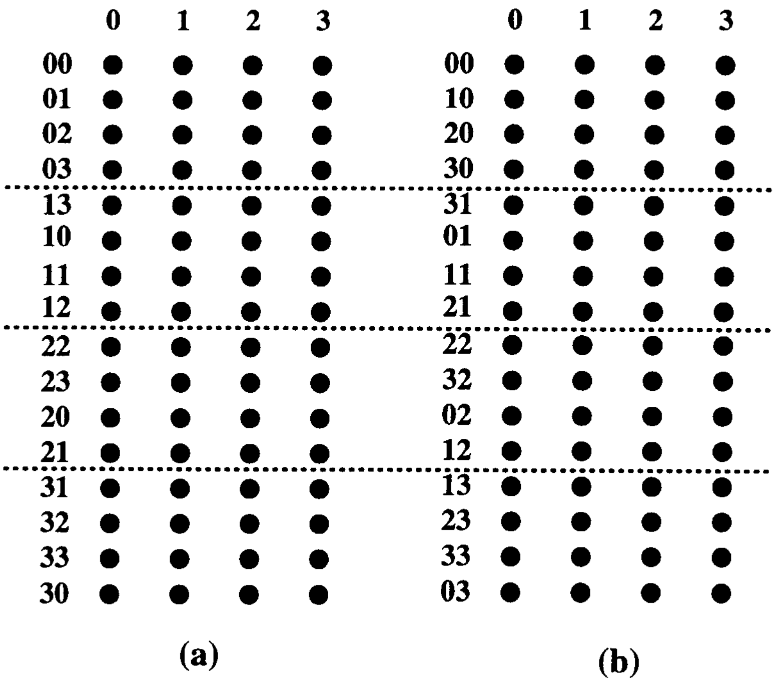


FIGURE 4.3. CP_1 (a) and CP_2 (b)

In order to form H_3, the third Hamiltonian cycle of Q_3^k, some cycle edges of $C_m \otimes C_k$ must be exchanged for cross edges of H_1, since it has the most cross edges. The edges to be exchanged must be chosen carefully, however, if H_1 is to remain a Hamiltonian cycle. The discussion below focuses on the conditions for choosing edges to exchange and, then, states a procedure for choosing proper edges.

To begin, consider Figure 4.3. This figure shows the two cross products $C_M \otimes C_k$, denoted by CP_1 (for *Cross Product*$_1$), and $C_m \otimes C_k$, denoted by CP_2 (for *Cross Product*$_2$), side by side, where C_M is the major cycle of $C_4 \otimes C_4$ and C_m is the minor cycle (refer to Figure 4.1). Also note that C_M, labeling the nodes in the vertical dimension of CP_1, is obtained alternately from the sequence generated by the function f_2. The problem requires exchanging cross edges from CP_1 with cycles edges from CP_2. As defined previously, a cross edge connects nodes in a horizontal direction, while a cycle edge connects nodes in a vertical direction. To provide more flexibility when referring to a cross edge, call a cross edge connecting a node in column 0 to a node in column 1 as a *type-1* cross edge, a cross edge connecting a node in column 1 to a node in column 2 as a *type-2* cross edge, etc. A wrap around edge, connecting a node in column $k-1$ with a node in column 0 is called a *type-0* cross edge.

Now, some of the conditions on the edges to be exchanged are apparent.

- First, the cross edges must come from the major cycle of CP_1 as it has the most cross edges to exchange. Figure 4.2 shows the major cycle (formed by the standard decomposition) of CP_1.

- Second, three pairs of cross edges must be chosen, with three out of the four types—*type-0, type-1, type-2, type-3*—being represented.

- A third condition is that the cycle edges exchanged must be adjacent in the vertical dimension of CP_2.

- Lastly, after the exchange of edges, the major cycle of CP_1 must remain a Hamiltonian cycle.

The implication of the last condition is not as easily seen as those the first three. While the cycle edges must be adjacent in the vertical dimension of CP_2, they must not overlap in the vertical dimension of CP_1. As seen in Figure 4.2, the major cycle can be viewed as beginning at the top and working its way down, level by level, to the bottom; call this the *forward* direction. Now consider Figure 4.4. In this figure, one pair of *type-3* cross edges, $(022, 023)$ and $(122, 123)$, in the major

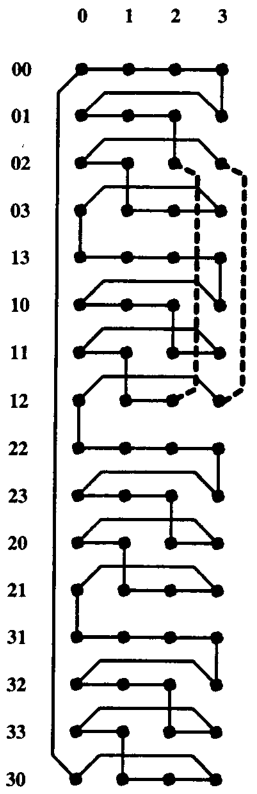


FIGURE 4.4. H_1 with one pair of cross edges exchanged for cycle edges.

cycle of CP_1 have been exchanged for edges (022, 122) and (023, 123). These two edges are cycle edges in CP_2 and are shown in Figure 4.4 with dotted lines.

As the figure shows, the modified cycle follows the original cycle until encountering one of the new cycle edges: (022, 122) in Figure 4.4. After taking the new cycle edge, the path again follows the original cycle, but in the opposite—that is, backwards—direction until it encounters the second new cycle edge. Taking the second new cycle edge, the path continues in the forward direction along the original path.

Now, consider what happens when two more cross edges are exchanged for cycle edges as in Figure 4.5. In this figure, the dotted edges are the two pairs of cycle

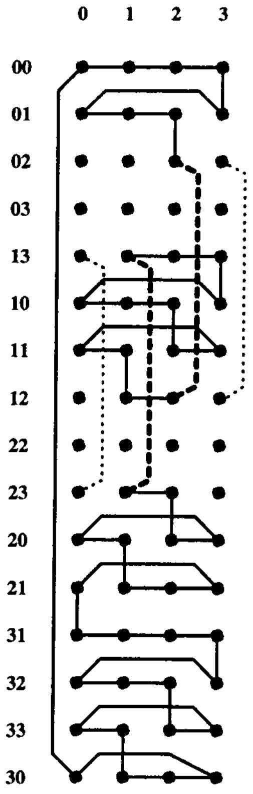


FIGURE 4.5. H_1 with two pairs of cross edges exchanged for cycle edges.

edges replacing the *type-3* and *type-1* cross edges. Note that the pairs overlap each other in the vertical dimension in CP_1 and that this breaks the Hamiltonian cycle, H_1. This is because the path, having taken the downward cycle edge (022, 122) encounters another downward cycle edge, (131, 231), before the matching cycle edge to (022, 122). The cycle edge (131, 231) starts the path forward again along the original route of H_1, leaving 16 nodes unvisited.

Therefore, given the conditions above for choosing edges to exchange and using the Q_3^4 cube as an example, the following procedure will decompose a Q_3^k into three disjoint Hamiltonian cycles. First, use function f_2 to form a two digit, radix k Gray code sequence, S. Then interchange the two digits, forming a second radix

k Gray code sequence, S'. Forming two $k^2 \times k$ rectangular arrays of nodes label the vertical dimension of one with the sequence S and the vertical dimension of the other with the sequence S'. The horizontal dimension of both arrays are labeled with the numbers $0 \ldots k-1$. Call the array labeled with sequence S as CP_1, and call the other array as CP_2. Figure 4.3 is an example of the procedure to this point using a Q_3^4.

The standard decomposition of CP_1 forms two disjoint Hamiltonian cycles of the Q_3^k; call the major cycle H_1 and the minor cycle H_2. Now, $k-1$ pairs of cross edges from H_1 will be exchanged for cycle edges from CP_2. The following method for choosing the edges leaves H_1 as a Hamiltonian cycle disjoint from H_2 and transforms CP_2 into a Hamiltonian cycle that is disjoint from both H_1 and H_2. This third cycle will be called H_3.

An examination of Figure 4.3a, shows that labels in the vertical dimension—derived from sequence S—can be grouped into k blocks. Within each block, the leading digit is the same. If each pair of exchanged cycle edges can be confined to a separate block in CP_1, then the problem of overlapping in the vertical dimension is solved. An examination of Figure 4.3b reveals that the only cycle edges having the same first digit are those on either side of the dotted lines; call these cycle edges *level-i border* edges, where i is the first digit.

Unfortunately, only using border edges does not suffice. The only cross edges associated with a pair of border edges are *type-2* ... *type-($k-1$)*; that is, only $k-2$ different types of cross edges are associated with a pair of border edges. However, $k-1$ different pairs of cross edges are needed to transform CP_2 into a Hamiltonian cycle.

The solution, then, is the following choices. Exchange the cycle edges between levels 00 and 10 for the corresponding *type-1* cross edges. Then, for $2 \leq i \leq k-1$, exchange the level-i border edges for the corresponding *type-i* cross edges. These choices exchange a sufficient number of non-overlapping cross edges to convert CP_2 into a Hamiltonian cycle, while allowing H_1 to remain a Hamiltonian cycle.

Example 4.1 As an example, in a Q_3^4, the following edges are exchanged between CP_1 and CP_2. The cross edges come from CP_1 and the cycle edges come from CP_2. *Type*-1 cross edges $(000, 001)$ and $(100, 101)$ are exchanged for cycle edges $(000, 100)$ and $(001, 101)$. *Type*-2 cross edges $(221, 222)$ and $(211, 212)$ are exchanged for level–2 border edges $(221, 211)$ and $(222, 212)$. Lastly, *type*–3 cross edges $(312, 313)$ and $(302, 303)$ are exchanged for level–3 border edges $(312, 302)$ and $(313, 303)$. Figure 4.6 shows the resulting decomposition of a Q_3^4 into three Hamiltonian cycles: H_1, H_2, and H_3. ■

4.3. Decomposing a Q_4^k

In this section, a simple and straightforward procedure for decomposing a Q_4^k into 4 disjoint Hamiltonian cycles is given. The procedure used in this section also can be used to decompose any Q_{2n}^k into $2n$ disjoint Hamiltonian cycles provided the Q_n^k has been decomposed into n disjoint Hamiltonian cycles.

The first step is to decompose a Q_2^k into two disjoint Hamiltonian cycles (see Section 4.1). Let S_1 and S_2 refer to the two sequences of length k^2 that comprise the two disjoint cycles of the Q_2^k. Note also that S_1 and S_2 consist of two digit, radix k numbers.

Since a Q_4^k has k^4 nodes and each node is labeled with a four digit, radix k number, the second step is to form two $k^2 \times k^2$ cross products, referred to as CP_1 and CP_2. Label CP_1 in both the vertical and the horizontal dimension with S_1, and label CP_2 in a similar manner using S_2. The labels in the vertical dimension

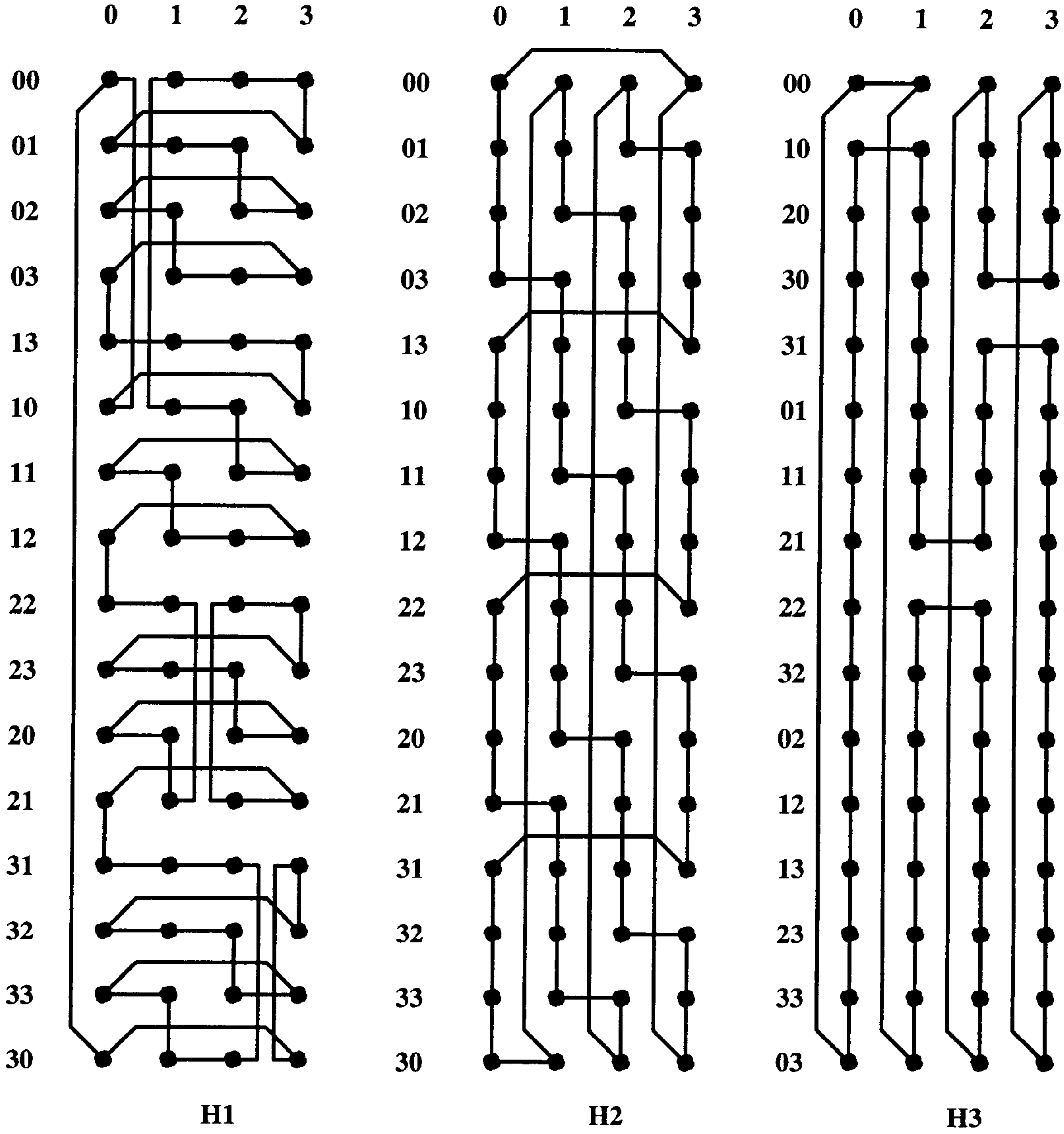


FIGURE 4.6. A Q_3^4 decomposed into 3 disjoint Hamiltonian cycles.

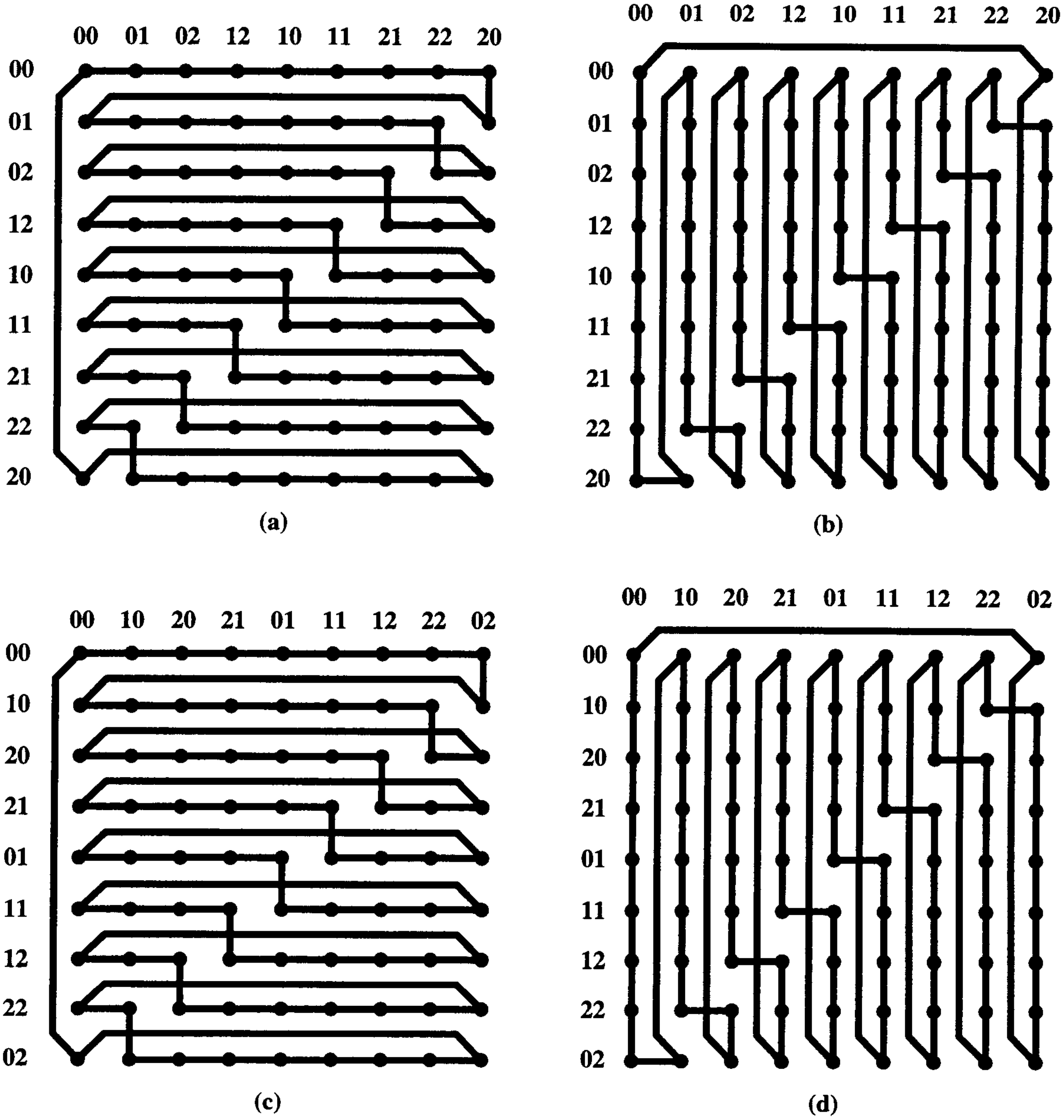


FIGURE 4.7. A Q_4^3 decomposed into 4 disjoint Hamiltonian cycles. Panel (a) shows the major cycle of the cross product formed using the major cycle of a Q_2^3, and panel (b) shows the minor cycle of the same cross product. Panel (c) shows the major cycle of the cross product formed using the minor cycle of a Q_2^3, and panel (d) shows the minor cycle for the same cross product.

represent the left two digits of the node labels, and the horizontal labels represent the right two digits. For example, Figure 4.7a illustrates one cross product denoting a Q_4^3. The address of the node in the upper left corner of the figure is 0000, the address of the node in the upper right corner is 0020, and the address of the node in the lower left corner is 2000. Figure 4.7c shows a second representation of a Q_4^3 as a cross product but labeled with the second sequence.

Once the two cross products are formed, a decomposition of the each cross product using the standard decomposition results in four disjoint Hamiltonian cycles of the Q_4^k. Let H_1 be the major cycle of CP_1, and let H_2 be the minor cycle: Figures 4.7a and 4.7b respectively. Further, let H_3 and H_4 represent the major and minor cycles of the standard decomposition of CP_2, Figures 4.7c and 4.7d respectively.

Clearly, H_1 is disjoint from H_2 and H_3 is disjoint from H_4 by virtue of the standard decomposition. In addition, H_1 must be disjoint from either H_3 or H_4, for suppose that H_1 had an edge in common with either H_3 or H_4, and let $(a_3a_2a_1a_0, b_3b_2b_1b_0)$ be that edge in H_1. If it is a cross edge, then $a_3 = b_3$, $a_2 = b_2$, and $\langle a_1a_0, b_1b_0\rangle$ belongs to S_1. But, if the edge is found also in either H_3 or H_4, it implies that $\langle a_1a_0, b_1b_0\rangle$ belongs to S_2, and this contradicts the fact that S_1 and S_2 represent disjoint sequences. In a similar manner, if the common edge is a cycle edge, then $a_1 = b_1$, $a_0 = b_0$, and $\langle a_3a_2, b_3b_2\rangle$ belongs to both S_1 and S_2, which, again, is a contradiction. The same argument can be applied to show that H_2 cannot share an edge with either H_3 or H_4.

As an example, Figure 4.7 illustrates the 4 disjoint Hamiltonian cycles obtained from a Q_4^3 using this method.

4.4. Decomposing Cubes of Higher Dimension

The results in this chapter are preliminary. This short section conjectures how a cube of higher dimension might be decomposed.

The procedure for decomposing the Q_4^k given in the previous section can be extended to decompose a Q_{2n}^k into $2n$ disjoint Hamiltonian cycles if the Q_n^k has been decomposed into n disjoint cycles.

For example, since the procedure for decomposing a Q_3^k into three disjoint Hamiltonian cycles, H_1, H_2, and H_3, is given in Section 4.2, a Q_6^k can be decomposed by writing three $k^3 \times k^3$ cross products, CP_1, CP_2, and CP_3. CP_1 is labeled in both the horizontal and vertical dimensions using H_1. Similarly, CP_2 and CP_3 are labeled using H_2 and H_3 respectively. Now taking the standard decomposition of each cross product gives two disjoint Hamiltonian cycles per cross product. Taken altogether, this procedure yields 6 disjoint Hamiltonian cycles for a Q_6^k.

Decomposing a Q_n^k into n disjoint cycles when n is odd and $n > 3$ is still an unsolved problem. In the following, some possible solutions are sketched. For example, two approaches to decomposing the Q_5^k seem to suggest themselves.

The first approach is similar to that of decomposing the Q_3^k. It considers four cross products, $CP_1, \ldots, CP_4$, where CP_i is labeled in the horizontal dimension with the single digit, radix k values $0, \ldots, k-1$ and in the horizontal dimension with the four digit, radix k sequence H_i. In this case, $H_1, \ldots, H_4$ are the four disjoint Hamiltonian cycles of a Q_4^k. The standard decomposition of CP_1 produces two disjoint Hamiltonian cycles for the Q_5^k, and, for each of the remaining cross products, the cycle edges form k disjoint cycles of length k^4. As in the case of the Q_3^k, an exchange of edges between CP_1 and the three other cross products must occur. This exchange must be such that the two disjoint Hamiltonian cycles in

CP_1 remain disjoint Hamiltonian cycles, and each of the remaining cross products is converted to a disjoint Hamiltonian cycle of a Q_5^k.

At this point, even though a Q_5^3 has been successfully decomposed into five disjoint Hamiltonian cycles using this approach, the general question of how to chose edges to exchange remains open. Because the sequences $H_1, \ldots, H_4$ are themselves the result of edge exchanges, a general pattern, such as the one seen in Section 4.2 that allows edges to be chosen for exchange, is difficult to discern.

The second approach to decomposing a Q_5^k is similar to that of decomposing the Q_4^k. Let the three disjoint Hamiltonian cycles of the Q_3^k be denoted by H_1^3, H_2^3, and H_3^3; and let the two disjoint Hamiltonian cycles of the Q_2^k be denoted by H_1^2 and H_2^2. Then, create three $k^3 \times k^2$ cross products: CP_1, CP_2, and CP_3. Next, label the vertical dimension of CP_i with H_i^3; label the horizontal dimensions of CP_1 and CP_3 with H_1^2; and label the horizontal dimension of CP_2 with H_2^2.

The labels are interpreted in the following manner. If a node in the Q_5^k has address $a_4a_3a_2a_1a_0$, then the labels in the vertical dimension provides the values for a_4, a_3, and a_2; while the labels in the horizontal dimension provides the values for a_1 and a_0.

Taking the standard decomposition of CP_1 and CP_2 gives four disjoint Hamiltonian cycles, and the cycle edges of CP_3 form k disjoint cycles of length k^3. To create the fifth Hamiltonian cycle, $k-1$ pairs of cross edges from CP_1 must be exchanged with $k-1$ pairs of cycles edges from CP_3. The exchanged edges must meet the four conditions given on page 46; however, once again, a general pattern for the choice of edges to exchange is not clear. In part, this difficulty also is because the sequences H_1^3, H_2^3, and H_3^3 are the result of edge exchanges, and, therefore, a regular pattern is disrupted.

Even if this second approach does not produce five disjoint Hamiltonian cycles for a Q_5^k, it does produce easily four disjoint cycles. In fact, using this approach, if $n = n_1 + n_2$ and if a $Q_{n_1}^k$ and a $Q_{n_2}^k$ have been decomposed into n_1 and n_2 disjoint Hamiltonian cycles respectively, then a Q_n^k can be decomposed into $2 \times \min(n_1, n_2)$ disjoint cycles.

To see the above, assume that $n_1 < n_2$. Then create n_1 cross products, each $k^{n_2} \times k^{n_1}$, and denote these as $CP_1, \ldots, CP_{n_1}$. Let $H_1^{n_1}, \ldots, H_{n_1}^{n_1}$ and $H_1^{n_2}, \ldots, H_{n_2}^{n_2}$ denote the n_1 and n_2 disjoint Hamiltonian cycles of the $Q_{n_1}^k$ and the $Q_{n_2}^k$ respectively. Now for each cross product, CP_i, label the vertical dimension with $H_i^{n_2}$ and the horizontal dimension with $H_i^{n_1}$, where the vertical labels provide the left-most n_2 values and the horizontal labels provide the right-most n_1 values of the address for the Q_n^k. Taking the standard decomposition of each cross product gives $2n_1$ disjoint Hamiltonian cycles for the Q_n^k.

5. COMMUNICATION ALGORITHMS

This chapter considers several communication algorithms for a toroidal interconnection network. The most basic communication problem in an interconnection network is the *routing* problem. In this problem, a node has a message to send to another node. Although referring to a hypercube, Johnsson and Ho [60] identified four other common communication problems. These problems are applicable to any toroidal interconnection network also. They are the following.

- *One–to–all broadcasting*: One node has a message to send to all other nodes in the system.
- *All–to–all broadcasting*: Each node has a message to send to all other nodes in the system.
- *One–to–all personalized communication*: One node has a different message to send to each other node in the system.
- *All–to–all personalized communication*: Each node has a different message to send to each other node in the system.

In addition, one other common communication problem has been identified by several researchers, *multicasting*, in which a node has a message to send to a subset of the other nodes in the system [67, 79, 78, 98].

In this chapter, an example of how Lee Distance can be used in a simple routing algorithm, and then three one-to-all broadcasting algorithms and one all–to–all broadcasting algorithm are discussed.

5.1. Dimensional Routing

Routing is one of the most active areas of research in interconnection networks. Before presenting a simple dimensional routing algorithm, a brief overview of the area will be given.

5.1.1. A Brief Overview of Routing in Toroidal Interconnection Networks

When a message is routed from one node to another in a direct multicomputer, it must be switched through the interconnection network, and the switching technology falls into two broad categories, *circuit* and *packet* switching [75]. In a multicomputer, packet switching is implemented as *store-and-forward* (SAF) and circuit switching has evolved into *worm-hole routing* (WR) [46, 95, 59, 1, 8].

When a message is sent using SAF, it is broken into fixed-sized packets. A distinguishing characteristic of SAF is that a packet must be completely received by a node before it can be forwarded to another. This means that the time to send a packet is proportional to the product of the packet size and the path length. Additionally, SAF encourages messages to be fixed length [66]. This is because a message spanning more than one packet requires additional complexity in the router to handle sequencing.

While first generation multicomputers used SAF routing, most later generation multicomputers use WR. This approach to flow control divides a message into small units called *flits*, or flow control digits. Only the initial flits, which comprise the header, carry routing information. As the message travels through the interconnection network, the flits follow each other in a pipeline fashion with each intermediate node on the path buffering a single flit of the message at time. Since the routing information is carried in the header only, once the header passes through an intermediate node, the path is allocated to the message. The path remains allocated to the message until the last flit passes, and for this reason, flits from different messages cannot be intermingled along the same path.

Typically, flits are only a few bytes in size. An advantage of wormhole routing is its small buffering requirement for a node. In addition, because of its small size, a flit passes through a node quickly once a path has been established. In general, the time to send a message using wormhole routing is proportional to the sum of the message size and the path length. This fact of wormhole routing is captured in a comment by Li *et alia* [71] who state that wormhole routing gives, "the illusion of a completely connected graph." Further, messages may vary in size: there is no requirement that the tail flits of a message leave the source node before the header flits of the message arrive at the destination node.

SAF routing has been augmented by a switching technology called *virtual cut-through* [62]. Virtual cut-through relaxes the requirement that a packet be received completely by a node before it can be retransmitted. In a lightly loaded network, SAF with virtual cut-through resembles wormhole routing; but, as congestion increases, it acts like traditional SAF.

A disadvantage of WR is its use of a *blocking* buffering scheme. That is, as long as the header can advance, so too do the following flits. If the header cannot advance because, for example, another worm holds the path, all flits in the first message hold their position. This blocks another message from using the path. For this reason, wormhole routing is susceptible to deadlock, particularly in toroidal interconnection networks.

An example of deadlock can be seen in Figure 5.1. This figure shows four messages (worms), each traveling in a different direction. Worm 1 is traveling east

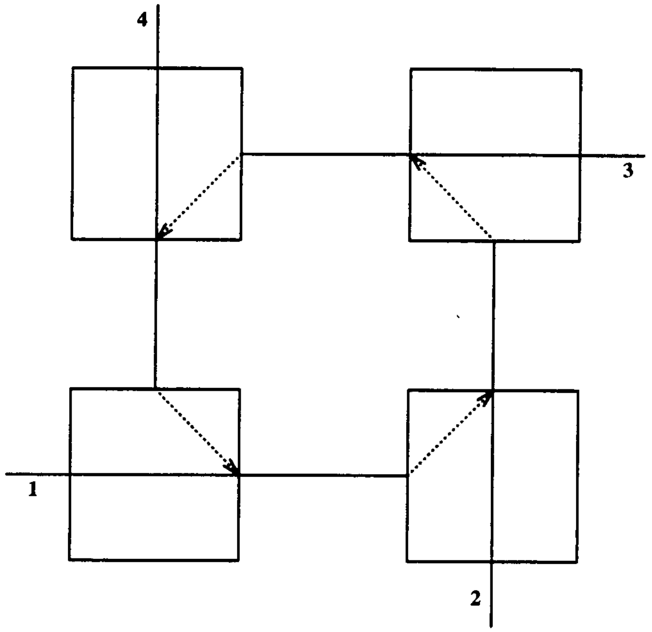


FIGURE 5.1. An example of deadlock when using wormhole routing

(left to right); worm 2 is traveling north; worm 3 is traveling west; and worm 4 is traveling south. Each worm is more than 2 flits long, and each want to turn to its left. No worm can advance, however, as each is blocked by another worm. This gives rise to a situation of circular wait and, hence, deadlock.

There are several solutions to the problem of deadlock proposed. One proposal, the *Turn Model* [51], restricts the directions in which a worm can turn. This model has the advantage of not requiring extra hardware support, but it can only be used to address the problem of deadlock. A second, more popular, approach is the use of *virtual channels* [35]. In this approach, multiple virtual channels are time multiplexed over a physical channel. Each physical channel has the same number of virtual channels assigned to it, and each virtual channel has its own input and output flit buffer.

A simple solution to the deadlock problem in a two dimensional torus using virtual channels is the following. Each physical channel is divided into two virtual channels, a high channel and a low channel. If a message is at a node numbered *less* than its destination, it is routed through the *high* channel. Otherwise, it is routed through the *low* channel. The node numbering is obtained in the manner presented in Section 1.2. Dally and Seitz have shown that this removes the possibility of deadlock by removing the circular wait [35].

A virtual channel increases the hardware complexity of a router because of the need for extra buffers and for multiplexor and demultiplexor hardware. However, in addition to preventing deadlock, a virtual channel can be used to improve throughput of the network [32], decrease latency [33], or provide adaptivity in the routing algorithm [34].

A routing algorithm can be classified as ***oblivious*** or ***adaptive***. An oblivious routing algorithm, such as the *e*-cube algorithm [93], does not consider network conditions, such as congestion or the presence of a faulty node or communication link, in its choice of a path. It might appear initially that an adaptive routing algorithm, which does consider the condition of the network in its choice of a path, would always give better performance than an oblivious algorithm. However, some simulations have shown that the complexity required for an adaptive router diminishes its performance when compared to an oblivious router [25].

Regardless, many researchers have proposed adaptive routing algorithms for a toroidal interconnection network [1, 12, 16, 17, 23, 26, 30, 34, 40, 41, 47, 49, 56, 72, 74]. An adaptive routing algorithm can be classified in one of two general categories. It is called a *minimal* algorithm if the message always makes progress towards its destination. That is, given a metric for distance, a minimal routing algorithm always chooses a route of least distance. Otherwise, it is called a *non-*

minimal or a *misrouting* algorithm. The non-minimal routers include a *deflection* router and a *queuing* router.

A deflection router, also called a *desperation* or a *hot potato* (except in Indiana where it is called a hot potatoe) router, uses fixed-sized packets for messages and is a *synchronous* router. When an interconnection network uses synchronous routing, a routing cycle is defined by a global clock, and all nodes may only send and receive a message in a time frame determined by the routing cycle. During each routing cycle, a deflection router sends each message at a node to a neighbor. Because all messages at a node must be sent out, a node can always accept incoming messages, and, hence, deadlock is avoided. A deflection router attempts to send a packet along a minimal path to its destination; however, if two packets at a node are destined for the same output channel, one must be misrouted.

Differing slightly from a deflection router is a queuing router. This class of router contains a central queue to which a packet is sent if it cannot be forwarded to the appropriate output channel. In the event that the queue is full, some packet must be misrouted. One version of a queuing router is the *Chaos* router [63]. When this router chooses a packet to misroute—because the central queue is full—it chooses one randomly.

Misrouting introduces a new problem, however. If a message is continually moving—that is, it is not deadlocked—but never reaches its destination, it is said to be *livelocked.* Two approaches to livelock prevention are *time-stamps* and *battle scars* [16]. In the first approach, when a message enters the network, it is given a time-stamp. When misrouting is necessary, the routing algorithm chooses the message with the latest time-stamp. That is, the longer a message has been in the network, the less likely it is to be misrouted. The second approach records each time a message is misrouted, its battle scar. Then, the routing algorithm chooses the message with the fewest battle scars to misroute. Both these livelock solutions suffer from the need for extra hardware and complexity in the router. In addition, the size of the header must be increased to accommodate the extra information.

5.1.2. A Simple Dimensional Routing Algorithm

This section presents a simple dimensional routing algorithm for a torus. This algorithm is oblivious and minimal. Its primary purpose is to illustrate how Lee distance is a natural metric to use with a toroidal interconnection network.

Let $\mathcal{T}$ be a $T_{\mathbf{K}}$, where $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$. Let **S** be a node with address label $s_{n-1}s_{n-2}\cdots s_0$; let **D** be a node with address label $d_{n-1}d_{n-2}\cdots d_0$; and suppose **S** wants to send a message to **D**. The routing information can be contained in n flits where each flit consists of a sign ($+$ or $-$) and a magnitude $|x_i| \in \{0, \ldots, \left\lfloor \frac{k_i}{2} \right\rfloor\}$. Let $\mathbf{X} = x_{n-1}x_{n-2}\cdots x_0$ be the displacement vector, where x_i encodes the routing information for dimension i. That is, x_i encodes both the sign and the magnitude.

The displacement vector **X** is calculated in the following manner. For each i, $0 \leq i < n$, let $x'_i = (d_i - s_i) \bmod k_i$. If $x'_i \leq \left\lfloor \frac{k_i}{2} \right\rfloor$, then $|x_i| = x'_i$ and the sign is positive. Otherwise, $|x_i| = k_i - x'_i$ and the sign is negative.

Once **X** has been calculated, the message is routed as follows. In dimension i, with i initially 0, if x_i is positive, the message proceeds from the node labeled $a_{n-1}a_{n-2}\cdots a_i \cdots a_0$ to the node labeled $a_{n-1}a_{n-2}\cdots(a_i + 1) \bmod k_i \cdots a_0$ and the magnitude is decremented. When $|x_i| = 0$, the flit is dropped from the header, and the message turns, moving in dimension $i + 1$. If, on the other hand, x_i is negative, the procedure is the same except that the message moves from the node labeled $a_{n-1}a_{n-2}\cdots a_i \cdots a_0$ to the node labeled $a_{n-1}a_{n-2}\cdots(a_i - 1) \bmod k_i \cdots a_0$.

Example 5.1 Let **S** be labeled 634 and **D** be labeled 452 in a $T_{8,6,5}$. Suppose **S** wants to send a message to **D**. Then

$$\begin{array}{rclrclrcll}
x_2' & = & (4-6) \bmod 8, & x_1' & = & (5-3) \bmod 6, & x_0' & = & (2-4) \bmod 5 & \\
 & = & 6, & & = & 2, & & = & 3, & \\
|x_2| & = & 2, & |x_1| & = & 2, & |x_0| & = & 2 & \text{and} \\
x_2 & = & -2, & x_1 & = & +2, & x_0 & = & -2 &
\end{array}$$

Using the displacement vector $\mathbf{X} = -2{+}2{-}2$, the message follows the path

$$\mathbf{S} = 634 \longrightarrow 633 \longrightarrow 632 \longrightarrow 642 \longrightarrow 652 \longrightarrow 552 \longrightarrow 452 = \mathbf{D} \qquad \blacksquare$$

5.2. Basic Broadcast Algorithms

This section presents an algorithm for one–to–all, or single-node, broadcasting in a torus; and an algorithm for all–to–all broadcasting. As stated in the beginning of this chapter, one–to–all broadcasting is a communication pattern in which one node has a message to share with all other nodes in the system, while all–to–all broadcasting means that each node in the system wants to do one–to–all broadcasting.

Generally, a broadcasting algorithm falls into one of two categories: *redundant* or *non-redundant.* A redundant broadcasting algorithm results in a node possibly receiving more than one copy of the message. On the other hand, when a non-redundant broadcasting algorithm is used, each node receives one copy of the broadcast message only. Broadcasting algorithms for a variety of topologies have been proposed. Examples include the hypercube [14], the star graph [73, 53], the hexagonal mesh [24], and the k-ary n-cube [19, 40].

A communication algorithm also can be described as *fault-tolerant* or *non-fault-tolerant.* A fault-tolerant broadcasting algorithm is one which broadcasts correctly even when one or more links or nodes are faulty. Previous work on fault-tolerant broadcasting algorithms include many topologies, e.g. the hexagonal mesh [61] and the star graph [9], but most research on fault-tolerant broadcasting has been done for the hypercube topology [4, 80, 81, 68, 97].

A fault-tolerant broadcasting algorithm relies on either *global* or *local* fault knowledge. Global fault knowledge means that each node knows the location of all faulty components in the system. Global fault knowledge also requires that the system have a method of determining and disseminating the fault information to all non-faulty components and is practical only when changes in the fault distribution occur occasionally. By contrast, local fault knowledge means that a node has information about its immediate neighbors only.

5.2.1. The Basic Broadcast Algorithm

The one–to–all broadcast algorithm presented in this section is non-redundant and non-fault-tolerant. It is based on the description of a torus as a cross-product of cycles. Also integral to this algorithm is the fact that a torus is vertex-symmetric.

Describing a graph as vertex-symmetric means that given any two vertices, $\mathbf{X}$ and $\mathbf{Y}$, an automorphism can be defined mapping $\mathbf{X}$ to $\mathbf{Y}$. More descriptively, a graph is vertex-symmetric if the *view* from a node is the same as from any other node [3]. This is useful when describing a broadcast algorithm because the source node of the broadcast can be assumed to be $\mathbf{0} = 00\cdots0$. Since an automorphism exists mapping $\mathbf{0}$ to any other node, an algorithm that broadcasts correctly from $\mathbf{0}$ also broadcasts correctly from any other node.

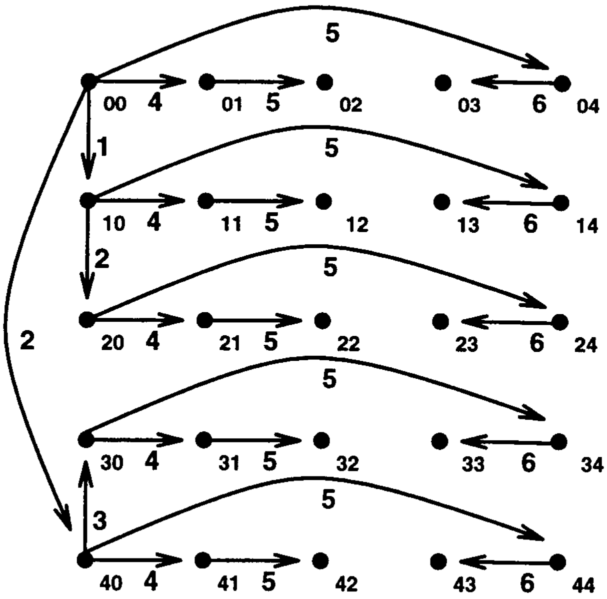


FIGURE 5.2. An example of BBA in a Q_2^5. The source node is 00, dimension 0 is the vertical dimension, and dimension 1 is the horizontal. The I/O model is single-port, and the large, single-digit numbers are the time step when the message is sent.

This algorithm is called the Basic Broadcast Algorithm (BBA) and consists of a sequence of n rounds, R_0, R_1, ..., R_{n-1}. Assuming that $\mathbf{0}$ is the source node of the broadcast and that $\mathcal{T}$ is a $T_{\mathbf{K}}$ defined over $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$, a description of BBA follows.

At the beginning of round R_0, $\mathbf{0}$ sends the message to its neighbors in dimension 0. Each of these nodes passes on the message to its neighbor in dimension 0 until, by the completion of round R_0, all nodes with addresses $00\cdots 0*$, "$*$" being a wildcard character, have received the message. Then, at the beginning of round R_1, each node having the message sends it to its neighbors in dimension 1. In general, by the completion of round R_i, all nodes having an address of the form $00\cdots 0*_i*_{i-1}\cdots *_0$ have received the message. As an example, Figure 5.2 illustrates BBA in a Q_2^5.

In particular, assume that during round R_i, node $\mathbf{X} = 0 \cdots 0 x_i x_{i-1} \cdots x_0$, where $x_i \neq 0$, receives message M from neighbor $\mathbf{X}' = 0 \cdots 0 x_i' x_{i-1} \cdots x_0$. If $x_i = x_i' + 1$ and $x_i + 1 \leq \left\lfloor \frac{k_i}{2} \right\rfloor$, $\mathbf{X}$ sends M to $0 \cdots 0(x_i + 1)x_{i-1} \cdots x_0$. If $x_i = x_i' - 1$ and $x_i - 1 > \left\lfloor \frac{k_i}{2} \right\rfloor$, $\mathbf{X}$ sends M to $0 \cdots 0(x_i - 1)x_{i-1} \cdots x_0$. Otherwise, R_i has completed. To begin round R_{i+1}, each node $\mathbf{Y}$, with an address of the form $0 \cdots 0 y_i y_{i-1} \cdots y_0$, now has a copy of M and sends it to both $0 \cdots 01 y_i \cdots y_0$ and $0 \cdots 0(k_{i+1} - 1)y_i \cdots y_0$.

Generally, one of two assumptions about a system are made when considering a broadcast algorithm [60]. The first assumption is *single-port I/O*, and it states that a node can transmit only in one dimension at a time. The second assumption, *multi-port I/O*, states that a node can transmit in multiple dimensions simultaneously.

In analyzing the performance of BBA, it should be clear that if single-port I/O is used, it takes $\left\lceil \frac{k_i}{2} \right\rceil$ time steps to complete round R_i. Since there are $n - 1$ rounds, the number of time steps BBA requires to complete a broadcast using single-port I/O is $\sum_{i=0}^{n-1} \left\lceil \frac{k_i}{2} \right\rceil$. The diameter of a $T_{\mathbf{K}}$ is $D = \sum_{i=0}^{n-1} \left\lfloor \frac{k_i}{2} \right\rfloor$; therefore, it can be seen that BBA is nearly optimal. In fact, if k_i is even for $0 \leq i \leq n - 1$, then BBA is optimal when single-port I/O is used. Otherwise, BBA requires only one extra time step for each odd radix.

If multi-port (2 or more) I/O is used, then each round R_i takes $\left\lfloor \frac{k_i}{2} \right\rfloor$ time steps regardless of whether k_i is even or odd. This implies that the total time for BBA to broadcast a message throughout a $T_{\mathbf{K}}$ is the same as the diameter; therefore, it is optimal.

5.2.2. All–To–All Broadcasting

All–to–all broadcasting becomes quite simple once a Hamiltonian cycle is embedded in a torus. Previously, four functions mapping a vector in a mixed radix sequence to a vector in a Lee distance Gray code sequence were given. Chapter 2, Section 2.2 presented function f_1. This function generates a Hamiltonian cycle in a torus meeting Property 1 (page 12). In Chapter 3, Theorem 3.2 presented function f_2, which finds a Hamiltonian cycle in a Q_n^k; and a similar function, f_3, is presented on page 31. Finally, on page 32, function f_4 is presented. This function also generates a Lee distance Gray code sequence from a radix k sequence. When k is even, the resulting Gray code sequence forms a Hamiltonian cycle in a Q_n^k, but if k is odd, the sequence forms a Hamiltonian path.

Now, a fifth function, a companion to f_1 is presented. This function finds a Hamiltonian cycle in a torus even when all the radices are odd. Recall that function f_1 requires at least one dimension to have an even radix. This new function is denoted by f_5.

Let $\mathcal{T}$ be a $\mathcal{T}_{\mathbf{K}}$, where $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$, and assume that k_i is odd for $0 \le i \le n-1$ and that the dimensions are ordered such that $k_{n-1} \ge k_{n-2} \ge \ldots \ge k_0$. Also, let $\mathbf{R} = r_{n-1}r_{n-2}\cdots r_0$ be a number in mixed radix form defined over $\mathbf{K}$, and let $\mathbf{G} = g_{n-1}g_{n-2}\cdots g_0$ be the Gray code representation of $\mathbf{R}$ given by f_5. That is, $\mathbf{G} = f_5(\mathbf{R})$. Further, define

$$\overline{r_i} = \begin{cases} r_i, & \text{if } r_{i+1} \text{ is odd} \\ k_i - 1 - r_i, & \text{otherwise} \end{cases}$$

Now, f_5 is defined as follows.

$$g_{n-1} = r_{n-1}, \text{ and for } 0 \le i \le n-2$$
$$g_i = \begin{cases} (r_i - r_{i+1}) \bmod k_i, & \text{if } r_{i+1} < k_i \\ \overline{r_i}, & \text{otherwise} \end{cases}$$

Assuming single-port I/O, the broadcast proceeds as follows. One of the five Gray code function is chosen—depending on the particular topology—and a

Hamiltonian cycle is generated. In the first time step, each node sends its message to its neighbor in the cycle. Then, during each time step t, each node sends the message received from its predecessor in the cycle at time step $t-1$. At the end of $\prod_{i=0}^{n-1} k_i - 1$ time steps (or $k^n - 1$ time steps in the case of a Q_n^k) the broadcast is complete. This is optimal for, assuming single-port I/O, each node must receive $\prod_{i=0}^{n-1} k_i - 1$ messages (or $k^n - 1$ messages) during an all–to–all broadcast.

5.3. The Fault-Tolerant Basic Broadcast Algorithm

This section describes a fault-tolerant variation of BBA called FTBBA (Fault-Tolerant Basic Broadcast Algorithm). As described here, FTBBA is a redundant broadcast algorithm which relies on local fault information only. It will broadcast correctly even in the presence of $2n-1$ faults.

FTBBA works in two phases. In Phase 1, the source node, S, executes BBA. In Phase 2, each node receiving the message during Phase 1 now acts as a source node and executes BBA. Since BBA takes $T(B) = \sum_{i=0}^{n-1} \lfloor \frac{k_i}{2} \rfloor$ times steps (or $T(B) = \sum_{i=0}^{n-1} \lceil \frac{k_i}{2} \rceil$ time steps when some k_i are odd and single-port I/O is used) for each phase, it is clear the NFTBA takes $2T(B)$ time steps to complete.

Theorem 5.1 shows that FTBBA broadcasts correctly in the presence of $2n-1$ faults. The basic idea of the proof is the following.

If S is the source node for the broadcast and D is an arbitrary non-faulty destination node in a $T_{\mathbf{K}}$, then Theorem 2.1 states that there are $2n$ disjoint paths between them. Since there are only $2n-1$ faults, at least one of these paths must be fault-free. Two items need to be shown, however. The first is how these disjoint paths are constructed, and, second, that all $2n$ paths are traversed during the execution of FTBBA.

Before showing how the paths are constructed, two preliminary concepts must be introduced. The first is that of a *half ring*, and the second is that of a *partial path.*

For each node $\mathbf{X}$ with address $x_{n-1}x_{n-2}\cdots x_0$, let ${}_X\mathrm{R}_i^+$ be the half ring of nodes in the positive direction of dimension i. That is,

$$ {}_X\mathrm{R}_i^+ = \{x^{(j)} \mid x^{(j)} = x_{n-1}x_{n-2}\cdots x_i + j \bmod k_i \cdots x_0\}, \text{ where } 1 \leq j \leq \left\lfloor \tfrac{k_i}{2} \right\rfloor $$

Similarly, Let ${}_X\mathrm{R}_i^-$ be the half ring of nodes in the negative direction of dimension i. That is,

$$ {}_X\mathrm{R}_i^- = \{x^{(j)} \mid x^{(j)} = x_{n-1}x_{n-2}\cdots x_i - j \bmod k_i \cdots x_0\} \begin{cases} 1 \leq j \leq \left\lfloor \frac{k_i}{2} \right\rfloor, & \text{if } k_i \text{ is odd} \\ 1 \leq j < \left\lfloor \frac{k_i}{2} \right\rfloor, & \text{if } k_i \text{ is even} \end{cases} $$

When the node $\mathbf{X}$ associated with a half ring is clear from the context, ${}_X\mathrm{R}_i^+$ is written as R_i^+.

Now, let $\mathbf{X}$ be a node with address $x_{n-1}x_{n-2}\cdots x_0$, and $\mathbf{Y}$ be a node with address $y_{n-1}y_{n-2}\cdots y_0$. In describing a path from $\mathbf{X}$ to $\mathbf{Y}$, the partial path Δ_i denotes a path from $\mathbf{X}$ to $\mathbf{X}' = x_{n-1}\ x_{n-2}\cdots x_{i+1}y_i\ x_{i-1}\cdots x_0$ that lies in the half ring of $\mathbf{X}$ in dimension i and containing $\mathbf{X}'$. Note that the definition of a half ring implies that $\mathbf{X}'$ lies in only one half ring of $\mathbf{X}$. The path from $\mathbf{X}$ towards $\mathbf{X}'$ contained in the other half ring in dimension i of $\mathbf{X}$ is denoted by $\bar{\Delta}_i$. Since this partial path does not reach $\mathbf{X}'$, the completion of this path to $\mathbf{X}'$ is denoted by $\bar{\Delta}_i^*$. When $x_i = y_i$, Δ_i is defined arbitrarily to be one step in the positive direction and $\bar{\Delta}_i$ as one step in the negative direction of dimension i.

Construction 5.1 Let $\mathcal{T}$ be a $T_{\mathbf{K}}$. Let $\mathbf{S} = 00\cdots 0$ be the source node of a broadcast and $\mathbf{D}$ be an arbitrary destination node. Assume that ℓ components of the address of $\mathbf{D}$ are 0; without loss of generality, assume the first ℓ dimensions are 0.

That is, **D** has address $d_{n-1} \cdots d_\ell 0 \cdots 0$. The following shows how to construct $2n$ disjoint paths between **S** and **D**.

Let $\mathcal{P}$ be the path Δ_ℓ, $\Delta_{\ell+1}$, $\cdots$, Δ_{n-2}, Δ_{n-1}. Then the first ℓ paths are called *Type I* paths and are formed as follows.

$$\begin{array}{lrl}
\text{Path} & 0: & \Delta_0, \quad \mathcal{P}, \ \bar{\Delta}_0 \\
\text{Path} & 1: & \Delta_1, \quad \mathcal{P}, \ \bar{\Delta}_1 \\
\vdots & & \quad\ \ \vdots \\
\text{Path} & \ell-1: & \Delta_{\ell-1} \ \mathcal{P}, \ \bar{\Delta}_{\ell-1}
\end{array}$$

The next ℓ paths are called *Type II* paths and have the form:

$$\begin{array}{lrl}
\text{Path} & \ell: & \bar{\Delta}_0, \quad \mathcal{P}, \ \Delta_0 \\
\text{Path} & \ell+1: & \bar{\Delta}_1, \quad \mathcal{P}, \ \Delta_1 \\
\vdots & & \quad\ \ \vdots \\
\text{Path} & 2\ell-1: & \bar{\Delta}_{\ell-1} \ \mathcal{P}, \ \Delta_{\ell-1}
\end{array}$$

The next $n-\ell$ paths are called *Type III* paths and are formed as cyclic shifts of $\mathcal{P}$.

$$\begin{array}{lrllll}
\text{Path} & 2\ell: & \Delta_\ell, & \Delta_{\ell+1} & \cdots & \Delta_{n-2}, \ \Delta_{n-1} \\
\text{Path} & 2\ell+1: & \Delta_{\ell+1}, & \Delta_{\ell+2} & \cdots & \Delta_{n-1}, \ \Delta_\ell \\
\vdots & & & & \vdots & \\
\text{Path} & \ell+n-1: & \Delta_{n-1}, & \Delta_\ell & \cdots & \Delta_{n-3}, \ \Delta_{n-2}
\end{array}$$

The last $n-\ell$ paths are formed also by cyclic shifts of $\mathcal{P}$, but by traveling in the half-ring in the longer direction in one dimension first. The path completes by traveling in the same direction in this dimension. These paths are called *Type IV* paths.

$$\begin{array}{lrllll}
\text{Path} & \ell+n: & \bar{\Delta}_\ell, & \Delta_{\ell+1} & \cdots & \Delta_{n-1}, \ \bar{\Delta}^*_\ell \\
\text{Path} & \ell+n+1: & \bar{\Delta}_{\ell+1}, & \Delta_{\ell+2} & \cdots & \Delta_\ell, \quad \bar{\Delta}^*_{\ell+1} \\
\vdots & & & & \vdots & \\
\text{Path} & 2n-1: & \bar{\Delta}_{n-1}, & \Delta_\ell & \cdots & \Delta_{n-2}, \ \bar{\Delta}^*_{n-1}
\end{array}$$

It can be seen that the total number of paths given by this construction is $\ell + \ell + (n-\ell) + (n-\ell) = 2n$. ■

The following two lemmas show that the $2n$ paths given by Construction 5.1 are disjoint and that FTBBA sends a message along each path.

Lemma 5.1 *The 2n paths given by Construction 5.1 are disjoint except for the source and destination nodes.*

Proof: Let $\mathcal{T}$, **S**, and **D** be defined as in Construction 5.1, and suppose there are two paths which are not disjoint.

The intersecting paths cannot be of type I, type II, or type IV. Each path of these types leaves **S** by a half ring unique to that path. A path of type I leaves **S** by the half ring determined by Δ_i, where $0 \leq i \leq \ell$; a path of type II leaves **S** by the half ring determined by $\bar{\Delta}_i$, where $0 \leq i \leq \ell$; and a path of type IV leaves **S** by the half ring determined by $\bar{\Delta}_i$, where $\ell < i \leq n-1$. After leaving **S** along a unique half ring, the nodes remaining on each path of type I, II, or IV retain a component determined by this half ring until entering **D**. Since no other path uses this half ring, each path of type I, II, or IV must be disjoint from all other paths.

The two intersecting paths, therefore, must be of type III. Without loss of generality, assume the two paths are Path 2ℓ and Path $2\ell+j$ where $0 < j \leq n-\ell-1$. Note that these two paths are the following.

$$\text{Path } 2\ell = \mathcal{P} = \Delta_\ell, \Delta_{\ell+1}, \cdots, \Delta_{n-1}$$

$$\text{Path } 2\ell + j = \Delta_{\ell+j}, \cdots, \Delta_{n-1}, \Delta_\ell, \cdots, \Delta_{\ell+j-1}$$

Now, suppose **X** is a node that is common to both paths. The first dimension which Path 2ℓ corrects is ℓ, where correcting dimension ℓ means that the remaining nodes along the path contain the same value in this dimension as the destination node: d_ℓ. This means that $\mathbf{X}$ cannot occur along Path $2\ell + j$ until after it has corrected dimension ℓ. However, Path $2\ell + j$ is a cyclic shift of Path ℓ; therefore, by the time Path $2\ell + j$ corrects dimension ℓ, it has also corrected dimension $n-1$. But, dimension $n-1$ is the last dimension corrected by Path 2ℓ. Therefore, $\mathbf{X}$ must occur on Path $2\ell+j$ before it corrects dimension ℓ. This contradicts the assumption that $\mathbf{X}$ lies on Path $2\ell + j$, and the two intersecting paths cannot be of type III. Since the intersecting paths cannot be of type I, II, III, or IV, the $2n$ paths given by Construction 5.1 are disjoint. ■

Lemma 5.2 *During its execution, FTBBA sends a message along each of the 2n paths described by Construction 5.1.*

Proof: FTBBA operates in two phases. During each phase, the dimensions through which a message travels monotonically increase from 0 to $n-1$.

Consider a path of type I or type II. During phase I of FTBBA, a message travels along that part of the path determined by Δ_i (or $\bar{\Delta}_i$), where $0 \leq i < \ell$, and by $\mathcal{P} = \Delta_\ell, \Delta_{\ell+1}, \cdots, \Delta_{n-1}$. The last component of the path, $\bar{\Delta}_i$ (or Δ_i) is traversed during phase II.

Consider a path of type III. Since a path of this type is a cyclic shift of $\mathcal{P}$, during phase I of FTBBA a message travels through that part of the path determined by $\Delta_{\ell+i}$ to Δ_{n-1} where $0 \leq i \leq n-\ell-1$. The remainder of the path, determined by Δ_ℓ to $\Delta_{\ell+i-1}$, is traversed during phase II.

The same argument used for a type III path can be applied to a type IV path. However, it must be noted that a type IV path begins with $\bar{\Delta}_i$ and ends with $\bar{\Delta}_i^*$, $\ell \leq i \leq n-1$; therefore, Path $\ell + n$ must be considered separately from the other type IV paths.

Path $\ell+n$ begins with $\bar{\Delta}_\ell$ and continues through Δ_{n-1}. This part of the path is traversed during phase I of FTBBA. The last component of the path, determined by $\bar{\Delta}_\ell^*$, is traversed during phase II. For the other type IV paths, if the next to last component is determined by Δ_i, the last component is determined by $\bar{\Delta}_{i+1}^*$. Therefore the last component of a type IV path is always traversed during phase II of FTBBA.

Taken together, it can be seen that each path described by Construction 5.1 is exercised during the execution of FTBBA. ■

Theorem 5.1 *Let $\mathcal{T}$ be a $T_{\mathbf{K}}$, where $\mathbf{K} = k_{n-1}k_{n-2}\cdots k_0$. The algorithm FTBBA correctly broadcasts a message throughout $\mathcal{T}$ even in the presence of $2n-1$ faults.*

Proof: Without loss of generality, assume that $\mathbf{S} = 00\cdots 0$ is the source node for the broadcast, and let $\mathbf{D} = d_{n-1}\cdots d_\ell 0\cdots 0$ be an arbitrary fault-free destination node having ℓ components of its address equal to 0. Construction 5.1 shows how to construct $2n$ paths between $\mathbf{S}$ and $\mathbf{D}$. Lemma 5.1 shows that these paths are disjoint except for $\mathbf{S}$ and $\mathbf{D}$, and Lemma 5.2 shows that FTBBA sends a message along each of these paths. There are at most $2n-1$ faults; therefore, at least one path must be fault free. Since FTBBA sends a message along each path, at least one copy of the message must get to $\mathbf{D}$. But, $\mathbf{D}$ is arbitrary; therefore, all fault-free nodes of $\mathcal{T}$ receive a copy of the message, and the theorem follows. ■

5.4. The Partner Fault Tolerant Broadcast Algorithm

The Fault Tolerant Basic Broadcast Algorithm discussed in Section 5.3 has several attractive features. Among these are a simple implementation, the need for only local fault knowledge, and an ability to broadcast in the presence of a large number of faults. The major disadvantages of FTBBA, however, include the high number of redundant broadcast messages generated, resulting in high message traffic, and, since it requires $2n\left\lfloor\frac{k}{2}\right\rfloor$ (or in some cases $2n\left\lceil\frac{k}{2}\right\rceil$) steps to complete, it is far from optimal.

This section discusses another broadcast algorithm, the Partner Fault Tolerant Broadcast Algorithm (PFTBA). While PFTBA handles fewer faults than FTBBA, it produces less redundant message traffic and takes less time than FTBBA. PFTBA makes the assumption that node faults occur infrequently. Therefore, the system can periodically test for faulty nodes, and all non-faulty nodes can maintain a list of those that have failed. That is, PFTBA assumes global fault knowledge. In addition, PFTBA is designed for a Q_n^k, and it will be assumed that if $\mathcal{Q}$ is a Q_n^k, then $k \geq n$. Two cases of fault occurrence are considered. The first is that of a single fault, and the second is $n-1$ faults.

5.4.1. Single Fault

This section illustrates the basic observation underlying PFTBA. Suppose S is the source node of a broadcast, without loss of generality, $0\cdots 0$. Also, suppose F is the faulty node, having address $f_{n-1}f_{n-2}\dots f_0$. Consider the result of a broadcast using BBA when $f_{n-1}\neq 0$.

During the first $n-1$ rounds, the message travels throughout the subcubes $00\cdots 0*$, $00\cdots 0**$, ..., $0*\cdots*$. Let $\mathcal{S}$ be the $n-1$–dimensional subcube $0*\cdots*$. Note that F does not belong to $\mathcal{S}$; therefore, all nodes in $\mathcal{S}$ receive the broadcast message.

During the last round of BBA, however, some nodes do not receive the broadcast message because of F. In particular, if $1 \leq f_{n-1} \leq \left\lfloor\frac{k}{2}\right\rfloor$, then nodes belonging to

$$\mathcal{A} = \{af_{n-2}\dots f_0 \mid f_{n-1} < a \leq \left\lfloor\tfrac{k}{2}\right\rfloor\}$$

will not receive the message. If $\left\lfloor \frac{k}{2} \right\rfloor < f_{n-1} < k$, then the nodes belonging to

$$\mathcal{B} = \{bf_{n-2}\ldots f_0 \mid \left\lfloor \tfrac{k}{2} \right\rfloor < b < f_{n-1}\}$$

do not receive the message.

For each node not receiving the message, however, there is a corresponding node which does. In particular, let

$$\mathcal{A}' = \{a(f_{n-2}-1)\ldots f_0 \mid f_{n-1} < a \leq \left\lfloor \tfrac{k}{2} \right\rfloor\}$$

and

$$\mathcal{B}' = \{b(f_{n-2}-1)\ldots f_0 \mid \left\lfloor \tfrac{k}{2} \right\rfloor < b < f_{n-1}\},$$

where the subtraction is modulo k. Then, depending on the value of f_{n-1}, if each node in $\mathcal{A}'$ or $\mathcal{B}'$ sends the message to its corresponding node in $\mathcal{A}$ or $\mathcal{B}$, the broadcast is complete. Note that in this case the time for PFTBA is the time for BBA plus one step. In fact, if k is odd and F belongs to ${}_S\mathrm{R}^{+}_{n-1}$, it can be seen that the broadcast can be completed in the same time as BBA. In addition, the amount of redundant message traffic is reduced to a minimum.

Example 5.2 Figure 5.3 is an example of PFTBA in a Q_3^k, $k > 3$, with one faulty node. In the figure, the source node is 000, and the faulty node is 113. Figure 5.3 (A) shows a section of the cube after the completion of BBA. In this panel, nodes 213 and 313 are non-faulty but are blocked from receiving the message by the faulty node. In Figure 5.3 (B), the two blocked nodes receive the message from their partners across dimension 1. ■

Now suppose that $f_{n-1} = 0$. When $00\cdots0$ broadcasts a message using BBA, more than a half ring of nodes are blocked from receiving the message by F. Two solutions present themselves, however.

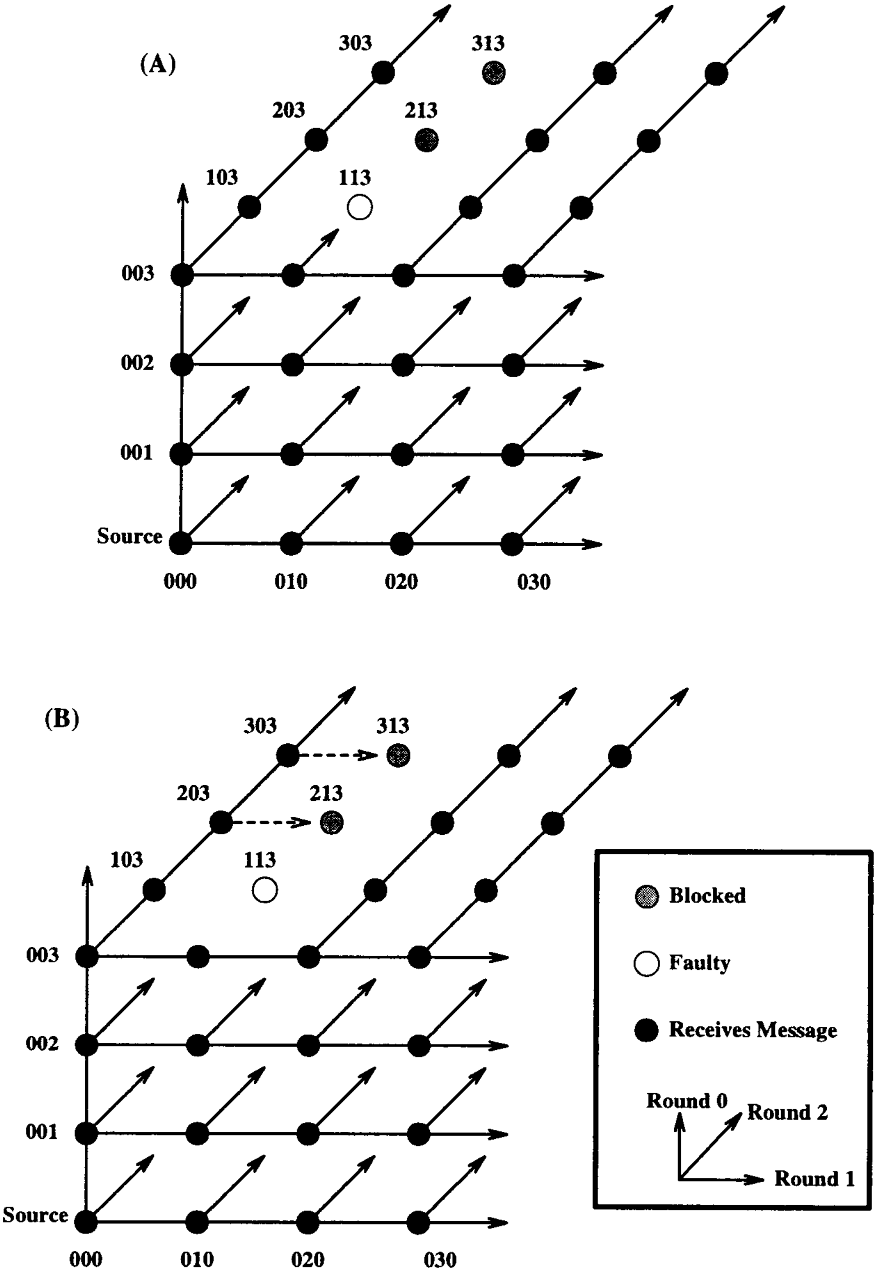


FIGURE 5.3. An example of PFTBA with one faulty node.

The first solution is for $00\cdots0$ to send the message to a neighbor such as $10\cdots0$, which now acts as the source of the broadcast. The second solution is to change the order of the dimensions used by BBA. If S and F are distinct, then there must be at least one dimension for which $f_i \neq 0$. Let j be the largest value of i for which $f_i \neq 0$. When S executes BBA but with the dimensions ordered as $j+1, j+2, \ldots, n-1, 0, 1, \ldots, j-1, j$, the broadcast proceeds as above.

5.4.2. Distributing Fault Information

In this section, the question of how fault information can be distributed through the Q_n^k is touched upon briefly.

An implicit assumption underlying global fault knowledge is that faults occur infrequently. One approach to the problem of disseminating fault information is to have one node responsible for detecting and distributing the fault information. Call this distinguished node M. Periodically, M broadcasts a *"You OK?"* message using an algorithm such as FTBBA. On receiving this message, each node sends an *"I'm OK"* response back to M. A node that M does not receive a response from is assumed to be faulty. M can then distribute the list of faulty nodes using FTBBA. Should M fail to broadcast a *"You OK?"* message within a certain time frame, then it is assumed to be faulty and another node takes over M's responsibilities.

In the above, an assumption is made that the communications library contains broadcast routines based on BBA, FTBBA, and PFTBA and that the message header identifies which algorithm is being used.

5.4.3. n - 1 Faults

Section 5.4.1 considers the case of a single node failure and illustrates the basic idea underlying PFTBA. This section extends the idea of Section 5.4.1 and considers $n-1$ faults.

Suppose $\mathcal{Q}$ is a Q_n^k where $k \geq n$. Also suppose that S $= 00\cdots 0$ is the source node of a broadcast, that $\mathcal{S}$ is the subcube $0*\cdots*$, and that $\mathcal{F} = \{F_1, F_2, \ldots, F_{n-1}\}$ is the set of faulty nodes such that

$$\begin{aligned} F_1 &= (f_{1,n-1})(f_{1,n-2})\cdots(f_{1,0}) \\ F_2 &= (f_{2,n-1})(f_{2,n-2})\cdots(f_{2,0}) \\ \vdots\ &\ \vdots \qquad\qquad\qquad \vdots \\ F_{n-1} &= (f_{n-1,n-1})(f_{n-1,n-2})\cdots(f_{n-1,0}) \end{aligned}$$

Consider what happens when $f_{i,n-1} \neq 0$ for $1 \leq i \leq n-1$. Since no faulty node occurs in $\mathcal{S}$, BBA proceeds without problems through the first $n-1$ rounds, and all nodes belonging to $\mathcal{S}$ receive the broadcast message. In the course of the last round, however, some non-faulty nodes are blocked from receiving the message by the members of $\mathcal{F}$. Note, however, that each faulty node prevents the message from reaching at most one half ring of nodes.

From the value of $f_{i,n-1}$, the actual nodes blocked by F_i can be calculated as in Section 5.4.1. To simplify notation, however, for each i, let BL_i be the set of nodes blocked from receiving the message by F_i. BL_i can be calculated in the following manner.

For a faulty node F_i, let f be the value in dimension $n-1$. That is, $f = f_{i,n-1}$. Then,

$$\begin{aligned} &\text{if} \quad 1 \leq f \leq \left\lfloor \tfrac{k}{2} \right\rfloor, \ \text{let } \mathrm{X}_i = \{x \mid f < x \leq \left\lfloor \tfrac{k}{2} \right\rfloor\}; \\ &\text{if} \quad \left\lfloor \tfrac{k}{2} \right\rfloor < f < k, \quad \text{let } \mathrm{X}_i = \{x \mid \left\lfloor \tfrac{k}{2} \right\rfloor < x < f\} \end{aligned}$$

Now,

$$BL_1 = \{x_1(f_{1,n-2})(f_{1,n-3}) \cdots (f_{1,0}) \mid x_1 \in X_1\}$$

$$BL_2 = \{x_2(f_{2,n-2})(f_{2,n-3}) \cdots (f_{2,0}) \mid x_2 \in X_2\}$$

$$\vdots \qquad\qquad \vdots$$

$$BL_{n-1} = \{x_{n-1}(f_{n-1,n-2})(f_{n-1,n-3}) \cdots (f_{n-1,0}) \mid x_{n-1} \in X_{n-1}\}$$

After the execution of BBA, for each fault F_i, the nodes in BL_i have not received the message. Following the approach in Section 5.4.1, for each set BL_i, it is tempting to define a set of nodes PS_i such as

$$PS_i = \{x_i(f_{1,n-2} - 1)(f_{1,n-3}) \cdots (f_{1,0}) \mid x_i \in X_i\}$$

where each set PS_i is called a *partner set* of BL_i. Then, when BBA has finished, each node in PS_i sends the message to its neighbor in BL_i, $0 \leq i \leq n-1$.

The problem, however, is that this may not always work. In particular if there are two faults F_i and F_j such that $f_{i,n-2} = f_{j,n-2} + 1$ and $f_{i,k} = f_{j,k}$, $0 \leq k < n-2$, then $BL_j \cap PS_i \neq \emptyset$. This means that some members of BL_i will not receive the message. If two faults such as F_i and F_j occur, call dimension $n-2$ *unusable*, and, in general, call a dimension d unusable if there are two faults, F_i and F_j, such that $D_L(f_{i,d}, f_{j,d}) = 1$, $f_{i,k} = f_{j,k}$, $0 \leq k \leq n-2$ and $k \neq d$. Otherwise, a dimension is called *usable*.

Example 5.3 Suppose S $= 0 \cdots 0$ is the source node of a broadcast in a Q_3^7 having two faults, $F_1 = 110$ and $F_2 = 100$. Then $BL_1 = \{210, 310\}$, $PS_1 = \{200, 300\}$, and

$BL_2 = \{200, 300\} = PS_1$. Note, however, that the set $PS_1' = \{216, 316\}$ could be used as a partner set for BL_1. PS_1' is formed by subtracting 1 modulo k from the value in dimension 0 of the nodes in BL_1. Therefore, dimension 1 is unusable, but dimension 0 is usable. ■

The following theorem shows that in a Q_n^k with $n-1$ faults, a usable dimension can always be found.

Theorem 5.2 *Let Q be a Q_n^k, $k \geq n$, with $n-1$ faults, $F_1, \ldots, F_{n-1}$, where $f_{i,n-1} \neq 0$, $1 \leq i \leq n-1$. There is at least one dimension which is usable.*

Proof: Suppose the theorem is false. That is, suppose no dimension d, $0 \leq d \leq n-2$, is usable and consider dimension $n-2$. Relabel the set of faults so that F_1 and F_2 are the ones making this dimension unusable.

Now consider dimension $n-3$. Let F_i and F_j be two faults making dimension $n-3$ unusable, and suppose F_i and F_j are different from F_1 and F_2. Relabel the set of faults again so that F_i and F_j become F_3 and F_4. It should be clear that this pattern cannot continue for long. That is, it takes at least two faults to make a dimension unusable. If, for each dimension d, $0 \leq d \leq n-3$, the two faults making it unusable are distinct from the faults making dimension i, $d < i \leq n-2$, unusable, then at least $2(n-1)$ faults are needed. However, Q is assumed to have at most $n-1$ faults.

It should also be clear from the definition that the same two faults cannot make two different dimensions unusable. Therefore, assume that either F_i or F_j above is F_1 or F_2. Without loss of generality, let that the faults making dimension $n-3$ unusable be F_2 and F_j. Now relabel the set of faults again so that F_j becomes F_3.

Suppose this process continues. For each dimension $n-d$, $3 \leq d \leq n$, one fault F_i, $i \geq d$, and one fault F_j, $j < d$ are found making dimension $n-d$ unusable, and then the set of faults is relabeled so that F_i becomes F_d. However, this means that showing dimension 0 unusable requires fault F_n. This cannot happen as the largest labeled fault is F_{n-1}.

Thus, the assumption that all dimensions are unusable must be false, and there must be at least one usable dimension. ■

Now extend the definition of PS_i to the following where X_i is defined as on page 80.

$$\begin{aligned}
PS_{1,j} &= \{x_1(f_{1,n-2})(f_{1,n-3})\cdots(f_{1,j}-1)\cdots(f_{1,0}) \mid x_1 \in X_1\} \\
PS_{2,j} &= \{x_2(f_{2,n-2})(f_{2,n-3})\cdots(f_{2,j}-1)\cdots(f_{2,0}) \mid x_2 \in X_2\} \\
\vdots \quad & \qquad\qquad \vdots \qquad\qquad \vdots \\
PS_{n-1,j} &= \{x_{n-1}(f_{n-1,n-2})(f_{n-1,n-3})\cdots(f_{n-1,j}-1)\cdots(f_{n-1,0}) \mid x_{n-1} \in X_{n-1}\}
\end{aligned}$$

It can be seen that if S $= 0\cdots0$ is the source node of a broadcast and if for each fault F_i, $f_{i,n-1} \neq 0$, then PFTBA proceeds in the following manner.

(1) S executes BBA

(2) A usable dimension d, $0 \leq d \leq n-2$ is chosen.

(3) For each partner set $PS_{i,d}$, $1 \leq i \leq n-1$, each node belonging to $PS_{i,d}$ sends the message to its corresponding neighbor in BL_i.

The key step in the above is step (2): choosing a usable dimension. The next section considers how a usable dimension may be chosen.

5.4.4. Finding A Usable Dimension

Let S $= 0 \cdots 0$ be the source node for a broadcast in $\mathcal{Q}$, a Q_n^k having $n-1$ faults. Also, let $\mathcal{F} = \{F_1, F_2, \ldots F_{n-1}\}$ be the set of faults where

$$F_i = (f_{i,n-1})(f_{i,n-2}) \cdots (f_{i,0}) \text{ and } f_{i,n-1} \neq 0, \quad 0 \leq i \leq n-1.$$

A usable dimension may be found by the following procedure.

First note that when two faults F_i and F_j differ in dimension $n-1$ only, any dimension that is usable for F_i will be usable for F_j also and vice-versa. This means that when looking for a usable dimension, all faults differing in dimension $n-1$ only form an equivalence class, and, for each equivalence class, only one member of the class needs to be considered. Refer to faults belonging to the same equivalence class as *equivalent.* Therefore, the first step in finding a usable dimension, is to form the reduced set $\mathcal{F}' \subseteq \mathcal{F}$ where $\mathcal{F}'$ has no equivalent faults.

Now if $|\mathcal{F}'| = 1$, all dimensions 0 through $n-2$ are usable. Otherwise, for each dimension d, $0 \leq d \leq n-2$ and for each $F_i \in \mathcal{F}'$, let

$$F_i^d = (f_{i,n-2})(f_{i,n-3}) \ldots (f_{i,d+1})(f_{i,d-1}) \ldots (f_{i,0}).$$

That is, F_i^d is the $n-2$-tuple which is F_i but with the values in dimensions $n-1$ and d removed. If, for any two faults F_i and F_j, $F_i^d = F_j^d$, then dimension d is not usable. Otherwise, dimension d is usable. Consider the following example.

Example 5.4 In a Q_5^6, let the following be three sets of faults.

Set 1	Set 2	Set 3
10000	22222	11234
20000	21222	11334
30000	22122	22231
40000	22212	22211

Note that for Set 1, all four faults form an equivalence class; therefore, $|\mathcal{F}'| = 1$ and the set of usable dimensions is $\{3,2,1,0\}$. For Sets 2 and 3, $\mathcal{F}' = \mathcal{F}$. Ignoring the value in dimension 4 and successively blocking out the values in dimensions 3 through 0, it can be seen that the usable dimensions are: Set 2 $= \{0\}$; and Set 3 $= \{3,0\}$. ■

Theorem 5.3 shows that this procedure always produces a usable dimension. Also, this theorem is an alternate proof that a usable dimension always exists, and it relies on two observations which are stated as lemmas.

Lemma 5.3 *Let F_i and F_j be two faults in a Q_n^k. If, for two dimensions d_1 and d_2, $0 \leq d_1, d_2 \leq n-2$, $F_i^{d_1} = F_j^{d_1}$ and $F_i^{d_2} = F_j^{d_2}$, then F_i and F_j are equivalent.*

Proof: If $F_i^{d_1} = F_j^{d_1}$, then F_i and F_j are the same in all dimensions between 0 and $n-2$ except for d_1. But, $F_i^{d_2} = F_j^{d_2}$ implies that F_i and F_j are the same in all dimensions except d_2, in particular F_i and F_j are the same in dimension d_1. Therefore, F_i and F_j have the same values in all dimensions between 0 and $n-2$ and are equivalent. ■

The next lemma extends Lemma 5.3 to multiple faults.

Lemma 5.4 *Let $F_{i_1}, \ldots, F_{i_m}$ be m faults in a Q_n^k. If, for dimensions $d_1, \ldots, d_m$, where $0 \leq d_1, \ldots, d_m \leq n-2$, and*

$$F_{i_1}^{d_1} = F_{i_2}^{d_1},\ F_{i_2}^{d_2} = F_{i_3}^{d_2},\ \ldots,\ F_{i_{m-1}}^{d_{m-1}} = F_{i_m}^{d_{m-1}},\ \textit{and}\ F_{i_m}^{d_m} = F_{i_1}^{d_m};$$

then faults $F_{i_1}, \ldots, F_{i_m}$ are equivalent.

Proof: Without loss of generality, assume that the m faults are $F_1, \ldots, F_m$, that the m dimensions are $1, \ldots, m$, and that

$$\begin{array}{rll} (1) & F_1^1 & = F_2^1 \\ (2) & F_2^2 & = F_3^2 \\ \vdots & & \vdots \\ (m-1) & F_{m-1}^{m-1} & = F_m^{m-1} \\ (m) & F_m^m & = F_1^m \end{array}$$

Ignoring the values in dimension $n-1$, statement (1) says that F_1 is the same as F_2 in all dimensions except 1. However, from statements (2) ... (m), it can be seen that

$$f_{2,1} = f_{3,1} = f_{4,1} = \ldots = f_{m,1} = f_{1,1}$$

Therefore, F_1 is the same as F_2 in all dimensions except $n-1$, and F_1 is equivalent to F_2.

Similarly, statement (2) says that F_2 is the same as F_3 in all dimensions except 2. But, from statements (3) ... (m) and (1), comes the equality

$$f_{3,2} = f_{4,2} = f_{5,2} = \ldots = f_{m,2} = f_{1,2} = f_{2,2}$$

and, thus, F_2 is equivalent to F_3.

Repeating this process shows that faults $F_1, \ldots, F_m$ are equivalent. ■

Theorem 5.3 *Let Q be a Q_n^k with $n-1$ faults. The procedure given in this section always finds a usable dimension.*

Proof: Let the faults be denoted by the set $\mathcal{F} = \{F_1, \ldots, F_{n-1}\}$, and assume each F_i differs from the source node in dimension $n-1$. Let $\mathcal{F}' \subseteq \mathcal{F}$ be the reduced set containing no equivalent faults. Recall that removing equivalent faults has no effect on finding a usable dimension. Construct a graph G whose vertex set is $\mathcal{F}'$. There is an edge in G between F_i and F_j if $F_i^d = F_j^d$ for some dimension d, $0 \leq d \leq n-2$. Then, form the connected subgraph G' by removing any unconnected vertices from G.

First note there can be no multiple edges in G' as Lemma 5.3 implies that multiple edges between two nodes F_i and F_j means they are equivalent. But, G' is constructed from $\mathcal{F}'$, which has no equivalent faults.

Second, note that Lemma 5.4 implies there can be no cycles in G' either as this also means equivalent faults. G' has $n-1$ vertices or less and, if no usable dimension is found, $n-1$ edges. However, a basic result from graph theory states that a connected graph having m vertices and m edges has at least one cycle [54]. Therefore, at least one usable dimension must be found. ■

5.4.5. The Source Node Problem

An assumption made in the preceding is that if S $= 0\cdots0$, then $f_{i,n-1} \neq 0$ for $1 \leq i \leq n-1$. This section considers the problem when $f_{i,n-1} = 0$ for at least one value of i. This is referred to as *the source node problem.*

It is important that the source node and the faulty nodes not share a value in dimension $n-1$. If they do, then more than a half-ring of non-faulty nodes are blocked from receiving the broadcast message by a faulty node. If the source node and one or more faulty nodes share a value in dimension $n-1$, then two approaches suggest themselves: (1) relabel the dimensions, or (2) let another node act as the source node.

Suppose S $= 0\cdots0$ is the source node and $\mathcal{F} = \{F_1, \ldots, F_{n-1}\}$ is the set of faults. Also suppose $f_{i,n-1} = 0$ for some i, $1 \leq i \leq n-1$. Then, if there is a dimension j such that $f_{i,j} \neq 0$ for all i, S can execute BBA using the following ordering of dimensions: $j+1$, $j+2$, …, $n-1$, 0, 1, …, $j-1$, j.

While reordering the dimensions is one solution to the source node problem, it creates new difficulties. The first is that whenever a different ordering of dimensions is used for BBA, the partner sets $PS_{i,d}$ must be recalculated, because finding a usable dimension depends upon the dimension last used by BBA. A second difficulty is that BBA must be modified so that the dimension order can be inferred from the broadcast message. That is, BBA cannot be assumed to begin with dimension 0 and proceed through dimension $n-1$.

Having another node act as the source node requires the original source node S send the broadcast message to the new source node, S′. This solution assumes the communication library has a fault-tolerant routing algorithm available. It also depends on the assumption that $k \geq n$.

Since there are at most $n-1$ faults and k values in each dimension, there must be at least one value in each dimension not shared by a faulty node. In particular, there is a value ℓ in dimension $n-1$ not shared by a fault $F_1, \ldots, F_{n-1}$. Therefore, a second solution to the source node problem is for S to send the message to S′ first, where S′ has the value ℓ in dimension $n-1$ but is the same as S otherwise. S′ then performs the broadcast.

A third solution to the source node problem is a variation of the above. The assumption $k \geq n$ means there is at least one value in each dimension not shared by a faulty node. Let s_i be a value in dimension i not shared by a faulty node. Then, even in the presence of $n-1$ faults, S′ $= s_{n-1}s_{n-2}\cdots s_0$ is fault-free. Suppose S′ is appointed as the "broadcast master"; that is, a broadcast message is routed to S′ first which, in turn, initiates the broadcast. In this case the partner sets $PS_{i,d}$ need to be recalculated only when a new fault is detected.

Since it is assumed that faults do not occur frequently, an advantage of this approach is that the partner sets $PS_{i,d}$ can be calculated and the member nodes notified of their responsibilities. Then, until a new fault is detected, a broadcast can proceed with little extra overhead.

However, this approach is not useful if many nodes need to broadcast information at the same time, for example a situation requiring an all-to-all broadcast. In this situation the "broadcast master", S′, becomes a bottleneck bringing down the system throughput.

5.4.6. Time Considerations

This section considers the time requirement for PFTBA. The time requirement cannot be calculated exactly as it depends upon the fault distribution and upon the time for a fault-tolerant routing algorithm.

First note that when k is odd and single-port I/O is used, it is possible for PFTBA to execute in the same time a BBA. This happens when, for example, the source node is $\mathrm{S} = 0 \cdots 0$ and each fault F_i has a value in dimension $n-1$ of f_i where $1 \leq f_i \leq \left\lfloor \frac{k}{2} \right\rfloor$ for $1 \leq i \leq n-1$. In this case, each node in the partner set $PS_{i,d}$ can send the message to its corresponding node in the set BL_i as BBA completes it final step. Otherwise, if k is even, PFTBA executes in the same time as BBA plus one step.

The analysis of the time needed by PFTBA becomes more complicated when it is necessary for the source node to send the message to another node first (the source node problem). In the following discussion, assume that $\mathrm{S} = 0 \cdots 0$ is the source node of a broadcast. The assumptions $k \geq n$ and $n-1$ faults imply that there is at least one value in each dimension not shared by a fault. Let s_i, $0 \leq i \leq n-1$, represent a value not shared by a fault in dimension i.

One solution to the source node problem is for S to send a broadcast message to node $\mathrm{S}' = s_{n-1}0 \cdots 0$ first, which then acts as the source node of the broadcast. In this case, the time for the broadcast is $\mathrm{TR}(\mathrm{S}, \mathrm{S}') + n \left\lfloor \frac{k}{2} \right\rfloor + 1$ where TR(S, S′) represents the time to route the message from S to S′. Depending on the distribution of the faults, TR(S, S′) could be close to $\left\lfloor \frac{k}{2} \right\rfloor + 2$. In this case the time for BFTBA could be as much as $(n+1) \left\lfloor \frac{k}{2} \right\rfloor + 3$.

Another solution to the source node problems is for S to send the broadcast message to the node $\mathrm{S}'' = s_{n-1}s_{n-2} \ldots s_0$. The advantage of this solution is that the partner sets $PS_{i,d}$ can be calculated *a priori*, reducing the overhead of the broadcast. The disadvantage of this solution is that S″ may become a bottleneck if many nodes need to broadcast. In addition, TR(S, S″) may be as much as the fault diameter of the Q_n^k. This means the time for PFTBA could exceed $2n \left\lfloor \frac{k}{2} \right\rfloor$, which is the time for FTBBA. Even in this case, however, the message traffic generated by PFTBA is still much less than that of FTBBA.

6. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

6.1. Conclusions

Computers are continually becoming faster and more powerful. Regardless of the measurement used—processor clock speed, word size, instructions executed per cycle, FLOPS, storage capacity, memory latency, I/O bandwidth, or a wide range of processor benchmarks—the capability of a computer continues to advance at a dizzying rate.

As the speed and power of a computer increases so too does the size and complexity of the problems to which it is applied. However, there are physical limits, such as the speed of light, that ultimately limit what an individual computer is capable of. As a response to the limitations of a single computer, computer scientists have proposed various models of parallel processing as a means of solving a given problem faster or of solving a larger version of a given problem.

In part, the approach of parallel processing requires a parallel machine, such as a multicomputer. A multicomputer is a machine with multiple processing elements that communicate with each other by sending messages through an interconnection network. While many topologies have been proposed for an interconnection network, a multidimensional torus is a topology popular with both academic researchers and commercial manufacturers.

This thesis has claimed that Lee distance is a natural metric to use in the study of toroidal interconnection networks. Gray codes based on Lee distance have been presented in Chapters 2, 3, and 5. These Gray codes were used to embed a Hamiltonian cycle in a general torus and in a k-ary n-cube. In addition, a Lee distance Gray code, one that was block-reflective, resulted in a simple method for embedding any even length cycle in a torus.

As shown in Section 5.2.2, finding a Hamiltonian cycle in a torus is useful for all–to–all broadcasting. The efficiency and fault-tolerance of all–to–all broadcasting in a Q_n^k can be increased if multiple, edge-disjoint Hamiltonian cycles are found in the Q_n^k. Chapter 4 considered ways of decomposing a Q_n^k into disjoint Hamiltonian cycles. Although the results are not complete, methods for decomposing a low dimensional k-ary n-cube are given. This is significant because the majority of parallel machines that are being built today and that have a toroidal interconnection topology are of dimension two or three.

Lee distance was also used in Section 3.1 to find a closed form expression for the surface area of a sphere of radius d. While other writers have given a non-closed form expression of this formula, this appears to be the first closed form version. And, in Section 3.3, a Lee distance Gray code was used to embed a mesh and a hypercube, with some restrictions, into a Q_n^k.

Finally, in Sections 5.2, 5.3, and 5.4, one–to–all broadcast algorithms are given. The algorithm in Section 5.2 is suitable for any $T_{\mathbf{K}}$; the algorithm in Section 5.3 is redundant, tolerates $2n-1$ faults, and also is suitable for any $T_{\mathbf{K}}$; and the algorithm in Section 5.4 is non-redundant, tolerates $n-1$ faults, but is designed for a Q_n^k. Although these broadcast algorithms do not directly use Lee distance, they are designed around some of the topological characteristics of a toroidal interconnection network developed in earlier chapters of this thesis.

6.2. Future Research

There is much work possible on the topic of Chapter 4. As mentioned previously, the results of this chapter are preliminary and a general framework needs to be found. When decomposing a cube of higher dimension there appears to be two approaches that can be used. To decompose a Q_n^k, is it better to use $n-1$ cross products, using the $n-1$ Hamiltonian cycles of the Q_{n-1}^k for the labels in one dimension and $0 \cdots k-1$ as the labels in the other dimension; or, if $n = n_1 + n_2$ where $n_1 \leq n_2$, is it better to use n_2 cross products and label the horizontal and vertical dimensions with the Hamiltonian cycles of the $Q_{n_1}^k$ and the $Q_{n_2}^k$, respectively? The first approach requires the exchange of more edges, but was successfully used to decompose a Q_5^3. On the other hand, the second approach immediately gives $2n_1$ disjoint Hamiltonian cycles and requires the exchange of fewer edges, but it is not clear how to choose the edges to exchange.

A second area of future research lies in the broadcast algorithms of Chapter 5. The Partner Fault Tolerant Broadcast Algorithm (PFTBA) of Section 5.4 is designed for a Q_n^k. A question remains if it can be extended to a general torus $T_{\mathbf{K}}$. This requires showing that Theorem 5.3 remains valid when the dimensions have different radices.

A second question about PFTBA is whether or not it can be extended to $2n-2$ faults. The following considers how this might be done.

Let $\mathcal{Q}$ be a Q_n^k with $k \geq 2n$. Let $\mathrm{S} = 0 \cdots 0$ be the source node of a broadcast and suppose there are $2n-2$ or less faults such that no fault has the value 0 in dimension $n-1$. Then using the assumptions above, the $n-1$-dimensional subcube $\mathcal{S} = 0 * * \cdots *$ is fault-free.

There are a total of k^{n-1} rings, each of size k, along dimension $n-1$. Each ring in dimension $n-1$ has $2n-2$ rings adjacent to it. Call a ring faulty if it contains one or more faults and non-faulty otherwise. If there are $2n-2$ or less faulty nodes, then, for each faulty ring, FR_i, at least one unique non-faulty adjacent ring, NFR_i, can be found.

Therefore, the broadcast proceeds as follows. First, S broadcasts the message throughout $\mathcal{S}$ using BBA. In the final round of BBA, the message is sent along each non-faulty ring. Finally, for each faulty ring FR_i, the message is sent from each node in its partner ring NFR_i to each corresponding non-faulty node in FR_e. This takes one additional time step.

The problem of how to choose the partner rings remains, however. With $n-1$ faults, the partners all lay in the same dimension, and finding a usable dimension was not a complicated procedure. With the extension to $2n-2$ faults, the pairs of partner rings may not all be adjacent across the same dimension. While the procedure that matches a non-faulty ring with a faulty ring may not be complex, it remains to be developed.

BIBLIOGRAPHY

[1] J. M. Adamo and N. Alhafez, "Methods for minimal, adaptive and deadlock-free routing in multiprocessors: A review", in *Proceedings of the Third Workshop on Parallel and Distributed Processing*, (Sofia, Bulgaria), pp. 159–178, April 1991.

[2] Anant Agarwal, "Limits on Interconnection Network Performance", *IEEE Transactions on Parallel and Distributed Systems*, vol. 2, no. 4, pp. 398–412, October 1991.

[3] Sheldon B. Akers and Balakrishnan Krishnamurthy, "A Group-Theoretic Model for Symmetric Interconnection Networks", Technical Report CR-86-29, Tektronix Laboratories, Beaverton, OR 97077, June 1987.

[4] Abdullah Al-Dhelaan and Bella Bose, "An Efficient Fault-Tolerant Broadcasting Algorithm for the Hypercube", in *Proceedings of the 4th Conference on Hypercube Concurrent Computers and Applications*, pp. 123–128, 1989.

[5] B. Alspach, J.-C. Bermond, and D. Sotteau, "Decompostion Into Cycles I: Hamilton Decompositions", in *Cycles and Rays* (Geňa Hahn *et al.*, eds.), pp. 9–18, Kluwer Academic Publishers, 1990.

[6] Marco Annaratone *et al.*, "The K2 Parallel Processor: Architecture and Hardware Implementation", in *Proceedings of the 17th Annual International Symposium on Computer Architecture*, (Seattle, WA), pp. 92–101, May 1990.

[7] William C. Athas and Charles L. Seitz, "Multicomputers: Message-Passing Concurrent Computers", *Computer*, vol. 21, no. 8, pp. 9–24, August 1988.

[8] Didier Badouel, Charles A. Wüthrich, and Eugene L. Fiume, "Routing Strategies and Message Contention on Low-dimensional Interconnection Networks", Technical Report CSRI-258, Computer System Research Institute, Universtiy of Toronto, December 1991.

[9] Nader Bagherzadeh, Nayla Nassif, and Shahram Latifi, "A Routing and Broadcasting Scheme on Faulty Star Graphs", *IEEE Transactions on Computers*, vol. 42, no. 11, pp. 1398–1403, November 1993.

[10] Kenneth E. Batcher, "Bit-Serial Parallel Processing Systems", *IEEE Transactions on Computers*, vol. C-31, pp. 377–384, May 1982.

[11] Elwyn R. Berlekamp, *Algebraic Coding Theory*. New York: McGraw–Hill, 1968.

[12] Pablo E. Berman, Luis Gravano, Gustavo D. Pifarré, and Jorge L. C. Sanz, "Adaptive Deadlock-and Livelock-Free Routing With All Minimal Paths in Torus Networks", in *ACM Symposium on Parallel Algorithms and Architectures*, (San Diego, CA), pp. 3–12, June 1992.

[13] J.-C. Bermond, O. Favaron, and M. Maheo, "Hamiltonian Decomposition of Cayley Graphs of Degree 4", *Journal of Combinational Theory, Series B*, vol. 46, pp. 142–153, 1989.

[14] D. P. Bertsekas *et al.*, "Optimal Communication Algorithms for Hypercubes", *Journal of Parallel and Distributed Computing*, vol. 11, pp. 263–275, 1991.

[15] Laxmi N. Bhuyan and Dharma P. Agrawal, "Generalized Hypercube and Hyperbus Structures for a Computer Network", *IEEE Transactions on Computers*, vol. C-33, no. 4, pp. 323–333, April 1984.

[16] Keven Bolding, Melanie L. Fulgham, and Lawrence Snyder, "The Case for Chaotic Adaptive Routing", Technical Report CSE-94-02-04, Universtiy of Washington, Seattle, Washington, 1994.

[17] Rajendra V. Boppana and Suresh Chalasani, "A Comparison of Adaptive Wormhole Routing Algorithms", in *Proceedings of the 20th Annual International Symposium on Computer Architecture*, (San Diego, CA), pp. 351–360, May 1993.

[18] Shekhar Borkar *et al.*, "iWarp: An Integrated Solution to High-Speed Parallel Computing", in *Proceedings of Supercomputing '88*, (Orlando, FL), pp. 330–339, November 1988.

[19] Bella Bose, Bob Broeg, Younggeun Kwon, and Yaagoub Ashir, "Lee Distance and Topological Properties of k–ary n–Cubes", *IEEE Transactions on Computers*, To appear.

[20] Jehoshua Bruck, Robert Cypher, and Ching-Tien Ho, "Efficient Fault-Tolerant Mesh and Hypercube Architectures", in *Proceedings of the Twenty-Second International Symposium on Fault-Tolerant Computing*, (Boston, MA), pp. 162–169, IEEE, July 1992.

[21] Jean-Philippe Brunet and S. Lennart Johnsson, "All-To-All Broadcast and Applications on the Connection Machine", *The International Journal of Supercomputer Applications*, vol. 6, no. 3, no. 3, pp. 241–256, 1992.

[22] J. Chung-Yaw Chaing and Jack K. Wolf, "On Channels and Codes for the Lee Metric", *Information and Control*, vol. 19, pp. 159–173, 1971.

[23] Ming-Syan Chen and Kang G. Shin, "Adaptive Fault-Tolerant Routing in Hypercube Multicomputers", *IEEE Transactions on Computers*, vol. 39, no. 12, pp. 1406–1416, December 1990.

[24] Ming-Syan Chen, Kang G. Shin, and Dilip D. Kandlur, "Addressing, Routing, and Broadcasting in Hexagonal Mesh Multiprocessors", *IEEE Transactions on Computers*, vol. 39, no. 1, pp. 10–18, January 1990.

[25] Andrew A. Chien, "A Cost and Speed Model for k-ary n-cube Wormhole Routers", in *Proceedings of the Hot Interconnects '93 Symposium*, (Palo Alto, CA), IEEE, August 1993.

[26] Andrew A. Chien and Jae H. Kim, "Planar–Adaptive Routing: Low–cost Adaptive Networks for Multiprocessors", in *Proceedings of the 19th International Symposium on Computer Architecture*, pp. 268–277, Association for Computing Machinery, 1992.

[27] Paul Cull, "Hamiltonian Circuits In Additive Machines", Technical Report 82-20-2, Oregon State University, Corvallis, OR, 1982.

[28] Paul Cull, "Tours of Graphs, Digraphs, and Sequential Machines", *IEEE Transactions on Computers*, vol. C-29, no. 1, pp. 50–54, January 1980.

[29] Paul Cull and Shawn M. Larson, "The Möbius Cubes", *IEEE Transactions on Computers*, vol. 44, no. 5, pp. 647–659, May 1995.

[30] Robert Cypher and Luis Gravano, "Storage-Efficient, Deadlock-Free Packet Routing Algorithms for Torus Networks", *IEEE Transactions on Computers*, vol. 43, no. 12, pp. 1376–1385, December 1994.

[31] William J. Dally, "Performance Analysis of k-ary n-cube Interconnection Networks", *IEEE Transactions on Computers*, vol. 39, no. 6, pp. 775–785, June 1990a.

[32] William J. Dally, "Virtual-Channel Flow Control", in *Proceedings of the* 17^{th} *International Symposium on Computer Architecture*, pp. 60–68, IEEEE Society, 1990b.

[33] William J. Dally, "Express Cubes: Improving the Performance of k–ary n–cube Interconnection Networks", *IEEE Transactions on Computers*, vol. 40, no. 9, pp. 1016–1023, September 1991.

[34] William J. Dally and Hiromichi Aoki, "Deadlock-Free Adaptive Routing in Multicomputer Networks Using Virtual Channels", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 4, pp. 466–475, April 1993.

[35] William J. Dally and Charles L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", *IEEE Transactions on Computers*, vol. C-36, no. 5, pp. 547–553, May 1987.

[36] William J. Dally *et al.*, "The J–Machine: A Fine–Grain Concurrent Computer", in *Information Processing 89*, pp. 1147–1153, Elservier Science Publishers B. V., 1989.

[37] Willaim J. Dally *et al.*, "The J-Machine: System Support For Actors", in *Actors: Knowledge-Based Concurrent Computing* (Hewitt and Agha, eds.), Cambridge, MA: MIT Press, 1991.

[38] William J. Dally *et al.*, "The Message-Driven Processor: A Multicomputer Processing Node with Efficient Mechanisms", *IEEE Micro*, vol. 12, no. 2, pp. 23–39, April 1992.

[39] Danny Dolev, Joseph Y. Halpern, Barbara Simons, and H. Raymond Strong, "A New Look at Fault-Tolerant Routing", *Information and Computation*, vol. 3, no. 72, pp. 180–196, March 1987.

[40] Jeffrey T. Draper and Joydeep Ghosh, "Multipath E-Cube Algorithms (MECA) for Adaptive Wormhole Routing and Broadcasting in k-ary n-cubes", in *Proceedings of the Sixth International Parallel Processing Symposium*, (Beverly Hills, CA), pp. 407–410, IEEE, March 1992.

[41] Josė Duato, "A New Theory of Deadlock-Free Adaptive Routing in Wormhole Networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 12, pp. 1320–1331, December 1993.

[42] Ralph Duncan, "A Survey of Parallel Computer Architectures", *Computer*, vol. 23, no. 2, pp. 5–16, February 1990.

[43] T. H. Duncan, "Performance of the Intel iPSC/860 and Ncube 6400 Hypercubes", *Parallel Computing*, vol. 17, pp. 1285–1302, 1991.

[44] Shantanu Dutt and John P. Hayes, "Some Practical Issues in the Design of Fault-Tolerant Multiprocessors", *IEEE Transactions on Computers*, vol. 41, no. 5, pp. 588–598, May 1992.

[45] Kemal Efe, "The Crossed Cube Architecture for Parallel Computation", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 5, pp. 513–524, September 1992.

[46] Sergio A. Felperin, Luis Gravano, Gustavo D. Pifarrė, and Jorge L. C. Sanz, "Routing Techniques for Massively Parallel Communication", *Proceedings of the IEEE*, vol. 79, no. 4, pp. 488–502, April 1991a.

[47] Sergio A. Felperin, Luis Gravano, Gustavo D. Pifarré, and Jorge L. C. Sanz, "Fully-Adaptive Routing: Packet Switching Performance and Wormhole Algorithms", in *Proceedings of Supercomputing '91*, (Albuquerque, NM), pp. 654–663, November 1991b.

[48] Marsha F. Foregger, "Hamiltonian Decompositions of Products of Cycles", *Discrete Mathematics*, vol. 24, pp. 251–260, 1978.

[49] Patrick T. Gaughan and Sudhakar Yalamanchili, "Adaptive Routing Protocols for Hypercube Interconnection Networks", *Computer*, pp. 12–23, May 1993.

[50] Cécile Germain, Jean-Luc Béchennec, Daniel Etiemble, and Jean-Paul Sansonnet, "A Communication Architecture for a Massively Parallel Message-Passing Multicomputer", *Journal of Parallel and Distributed Computing*, vol. 19, pp. 338–348, 1993.

[51] Christopher J. Glass and Lionel M. Ni, "The Turn Model for Adaptive Routing", in *Proceedings of the 19th Annual International Symposium on Computer Architecture*, (Gold Coast, Australia), pp. 278–287, ACM, May 1992.

[52] Solomon W. Golomb and Lloyd R. Welch, "Algebraic Coding and the Lee Metric", in *Error Correcting Codes* (Henry B. Mann, ed.), (University of Wisconsin, Madison), pp. 175–194, Mathematics Research Center, US Army, May 1968.

[53] Sidney W. Graham and Steven R. Seidel, "The Cost of Broadcasting on Star Graph and k-ary Hypercubes", *IEEE Transactions on Computers*, vol. 42, no. 6, pp. 756–759, June 1993.

[54] Nora Hartsfield and Gerhard Ringle, *"Pearls in Graph Theory: A Comprehensive Introduction"*. San Diego, CA: Academic Press, 1990.

[55] Charles D. Hodgman, ed., *"C. R. C. Standard Mathematical Tables"*. 2310 Superior Avenue, N.E., Cleveland, Ohio: Chemical Rubber Publishing Company, 11 ed., 1957.

[56] Ting-Wei Hou, S. R. Tsai, and L. M. Tseng, "Adaptive and Fault-Tolerant Routing Algorithms for High Performance 2D Torus Interconnection Networks", *Computers & Mathematics With Applications*, vol. 23, no. 1, pp. 3–15, January 1992.

[57] Wen-Jing Hsu, "Fibonacci Cubes–A New Interconnection Topology", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 3–12, January 1993.

[58] Yixiu Huang, "On Hamiltonian Decompositions of Cayley Graphs on Cyclic Groups", *Annals of the New York Acadamy of Sciences*, vol. 576, pp. 250–258, 1989.

[59] C. R. Jesshope, P. R. Miller, and J. T. Yantchev, "High Performance Communications in Processor Networks", in *Proceedings of the 16th Annual International Symposium on Computer Architecture*, pp. 150–157, IEEE, 1989.

[60] S. Lennart Johnson and Ching-Tien Ho, "Optimum Broadcasting and Personalized Communication in Hypercubes", *IEEE Transactions on Computers*, vol. 38, no. 9, pp. 1249–1268, September 1989.

[61] Dilip D. Kandlur and Kang G. Shin, "Reliable Broadcast Algorithms for HARTS", *ACM Transactions on Computer Systems*, vol. 9, no. 4, pp. 374–398, November 1991.

[62] Parviz Kermani and Leonard Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique", *Computer Networks*, vol. 3, pp. 267–286, 1979.

[63] Smaragda Konstantinidou and Lawrence Snyder, "The Chaos Router", *IEEE Transactions on Computers*, vol. 43, no. 12, pp. 1386–1397, December 1994.

[64] J. Mohan Kumar and L. M. Patnaik, "Extended Hypercube: A Hierarchical Interconnection Network of Hypercubes", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 1, pp. 45–57, January 1992.

[65] Younggeun Kwon, *"Embeddings in Parallel Systems"*. PhD thesis, Oregon State University, Corvallis, Oregon, May 1993.

[66] Simon S. Lam, "Store-and-Forward Buffer Requirements in a Packet Switching Network", *IEEE Transactions on Communications*, vol. COM-24, no. 4, pp. 394–403, April 1976.

[67] Youran Lan, Abdol-Hossein Esfahanian, and Lionel M. Ni, "Multicast in Hypercube Multiprocessors", *Journal of Parallel and Distributed Computing*, vol. 8, pp. 30–41, 1990.

[68] Tze Chiang Lee and John P. Hayes, "A Fault-Tolerant Communication Scheme for Hypercube Computers", *IEEE Transactions on Computers*, vol. 41, no. 10, pp. 1242–1256, October 1992.

[69] F. Thomson Leighton, *"Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes"*. San Mateo, CA: Morgan Kaufmann Publishers, 1992.

[70] Daniel Lenoski *et al.*, "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor", in *The 17th Annual International Symposium on Computer Architecture*, (Seattle, WA), pp. 148–159, IEEE, May 1990.

[71] Kai Li, Jeffrey F. Naughton, and James S. Plank, "An Efficient Checkpointing Method for Multicomputers with Wormhole Routing", *International Journal of Parallel Programming*, vol. 20, no. 3, no. 3, pp. 159–180, 1991.

[72] Daniel Linder and Jim C. Harden, "An Adaptive and Fault Tolerant Wormhole Routing Strategy for k–ary n–cubes", *IEEE Transactions on Computers*, vol. 40, no. 1, pp. 2–12, January 1991.

[73] Victor E. Mendia and Dilip Sarkar, "Optimal Broadcasting on the Star Graph", *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 4, pp. 389–396, July 1992.

[74] John Y. Ngai and Charles L. Seitz, "Adaptive Routing in Multicomputer Networks", in *Opportunities and Constraints of Parallel Computing* (Jorge L. C. Sanz, ed.), pp. 89–92, Springer-Verlag, 1989.

[75] Lionel M. Ni and Philip K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks", *Computer*, vol. 26, no. 2, pp. 62–76, February 1993.

[76] Wilfried Oed, "The Cray Research Massively Parallel Processor System: CRAY T3D", technical report, Cray Research Inc., November 1993.

[77] Krishnan Padmanabhan, "Cube Structures for Multiprocessors", *Communications of the ACM*, vol. 33, no. 1, pp. 43–52, January 1990.

[78] Dhabaleswar K. Panda and Pradeep Prabhakaran, "Multicasting Using Multidestination-Worms Conforming to Base Routing Schemes", in *Proceedings of the Eighth International Parallel Processing Symposium*, 1994.

[79] Dhabaleswar K. Panda and Sanjay Singal, "Broadcasting in k-ary n-cube Wormhole Routed Networks Using Path-based Routing", in *Proceedings of the Eighth International Parallel Processing Symposium*, 1994.

[80] Seungjin Park and Bella Bose, "Broadcasting in Hypercubes with Link/Node Failures", in *Proceedings of the 4th Symposium on the Frontiers of Massively Parallel Computation*, pp. 286–290, 1992.

[81] Seungjin Park, Bella Bose, and Bob Broeg, "Algorithms for Broadcasting in Faulty Hypercubes", in *Sixth International Conference on Parallel and Distributed Computing Systems*, (Louisville, Kentucky), October 1993.

[82] Franco P. Preparata and Jean Vuillemin, "The Cube-Connected Cycles: A Versatile Network for Parallel Computation", *Communications of the ACM*, vol. 24, no. 5, pp. 300–309, May 1981.

[83] David J. Pritchard and Denis A. Nicole, "Cube Connected Möbius Ladders: An Inherently Deadlock-Free Fixed Degree Network", *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, no. 1, pp. 111–117, January 1993.

[84] Daniel A. Reed and Richard M. Fujimoto, *"Multicomputer Netwoks: Message-Based Parallel Processing"*. Cambridge, MA: The MIT Press, 1987.

[85] Daniel A. Reed and Dirk C. Grunwald, "The Peformance of Multicomputer Interconnection Networks", *Computer*, vol. 20, no. 6, pp. 63–73, June 1987.

[86] Charles L. Seitz, "Concurrent VLSI Architectures", *IEEE Transactions on Computers*, vol. C-33, no. 12, pp. 1247–1265, December 1984.

[87] Charles L. Seitz, "The Cosmic Cube", *Communications of the ACM*, vol. 28, no. 1, pp. 22–33, January 1985.

[88] Charles L. Seitz *et al.*, "The Architecture and Programming of the Ametek Series 2010", in *Proceedings of the Third Conference on Hypercube Concurrent Computers and Applications*, vol. 1, (Pasadena, CA), pp. 33–37, January 1988a.

[89] Charles L. Seitz *et al.*, "Submicron Systems Architecture Project Semiannual Technical Report", Technical Report Caltec-CS-TR-88-18, California Institute of Technology, November 1988b.

[90] I. Stephenson and R. W. Taylor, "Creatures, Buckets and Spirals: Adventures on a Twisted Torus", Technical Report CSEG.93.01, University of York, York, UK, April 1993.

[91] Richard Stong, "On Hamiltonian Cycles In Cayley Graphs of Wreath Products", *Discrete Mathematics*, vol. 65, pp. 75–80, 1987.

[92] Richard Stong, "Hamilton decomposition of Cartesian Products of Graphs", *Discrete Mathematics*, vol. 90, pp. 169–190, 1991.

[93] Herbert Sullivan and T. R. Bashkow, "A Large scale, Homogeneous, Fully Distributed Parallel Machine", in *Proceedings of the 4th Annual Symposium on Computer Architecture*, pp. 105–117, IEEE, March 1977.

[94] Tera Computer Systems, "Overview of the Tera Parallel Computer", 1993.

[95] Martin Tompa, "Lecture Notes on Message Routing in Parallel Machines", Technical Report 94-06-05, University of Washington, Seattle, WA, June 1994.

[96] David Witte and Joseph A. Gallian, "A Survey: Hamiltonian Cycles in Cayley Graphs", *Discrete Mathematics*, vol. 51, pp. 293–304, 1984.

[97] Jie Wu and Eduardo B. Fernandez, "Reliable Broadcasting in Faulty Hypercube Computers", in *Proceedings of the 11th Symposium on Reliable Distributed Systems*, (Houston, Texas), October 1992.

[98] Jie Wu and Kaojun Yao, "Multicasting in Injured Hypercubes Using Limited Global Information", in *Proceedings of the 5th IEEE Syposium on Parallel and Distributed Processing*, (Dallas, Texas), pp. 548–555, December 1993.

[99] Abdou S. Youssef and Bhagirath Narahari, "The Banyan–Hypercube Networks", *IEEE Transactions on Parallel and Distributed Systems*, vol. 1, no. 2, pp. 160–169, April 1990.

[100] Glenn Zorpette, "Technology 1991: Minis and Mainframes", *IEEE Spectrum*, vol. 28, pp. 40–43, January 1991.