Lee Distance and Topological Properties of k-ary n-cubes

Bella Bose, Fellow, IEEE Computer Society, Bob Broeg, Student Member, IEEE Computer Society, Younggeun Kwon, and Yaagoub Ashir

Abstract—In this paper, we consider various topological properties of a k-ary n-cube ($Q^n_k$) using Lee distance. We feel that Lee distance is a natural metric for defining and studying a $Q^n_k$.

After defining a $Q^n_k$ graph using Lee distance, we show how to find all disjoint paths between any two nodes. Given a sequence of radix k numbers, a function mapping the sequence to a Gray code sequence is presented, and this function is used to generate a Hamiltonian cycle.

Embedding the graph of a mesh and the graph of a binary hypercube into the graph of a $Q^n_k$ is considered. Using a k-ary Gray code, we show the embedding of a $k^n \times k^n \times \ldots \times k^n$-dimensional mesh into a $Q^n_k$, where $n = \sum_{i=1}^{n} n_i$. Then using a single digit, 4-ary reflective Gray code, we demonstrate embedding a $Q^n_k$ into a $Q^4_4$.

We look at how Lee distance may be applied to the problem of resource placement in a $Q^n_k$ by using a Lee distance error-correcting code. Although the results in this paper are only preliminary, Lee distance error-correcting codes have not been applied previously to this problem.

Finally, we consider how Lee distance can be applied to message routing and single-node broadcasting in a $Q^n_k$. In this section we present two single-node broadcasting algorithms that are optimal when single-port and multi-port I/O is used.

Index Terms—K-ary n-cubes, Lee distance, error-correcting codes, Gray codes, Hamiltonian cycles, routing, broadcasting, embedding.

1. INTRODUCTION

The k-ary n-cube ($Q^n_k$) graph has been used in the design of several concurrent computers, including the AMT I-Machine [6], [7], the Mosaic [16], the iWarp [3], and the Cray T3D [12]. This is partly because many linear algebra computations and partial differential equations can be performed efficiently on machines having a topology based on a $Q^n_k$. A k-ary n-cube parallel machine consists of $k^n$ identical processors. Each processor has its own memory and is connected to 2n other processors.

This paper presents several topological properties of a $Q^n_k$ using Lee distance. The rest of this paper is organized as follows. Section II discusses definitions and mathematical preliminaries. Section III discusses the number of node-disjoint paths in a $Q^n_k$. Section IV considers embedding a ring, a mesh, and a hypercube into a $Q^n_k$. Section V discusses a method for resource allocation; that is, identifying a set of nodes to be given a resource such as an I/O processor so that all nodes in the system have convenient access to the resource. Section VI considers two single-node broadcasting algorithms and a d-dimensional wormhole routing algorithm based on Lee distance, and Section VII discusses future research.

II. PRELIMINARIES

This section contains definitions and mathematical background that will be useful for subsequent sections.

A. Lee Distance and the k-ary n-cube

Let $A = a_0, a_1, a_2, \ldots, a_n$ be an n-digit radix k vector. The Lee weight of $A$ is defined as

$$W_L(A) = \sum_{i=0}^{n-1} |a_i|,$$

where

$$|a_i| = \min(a_{i-1}, k - a_{i-1}).$$

The Lee distance between two vectors $A$ and $B$ is denoted by $D_L(A, B)$ and is defined to be $W_L(A - B)$. That is, the Lee distance between two vectors is the Lee weight of their bitwise difference, mod k. For example, when $k = 4$, $W_L(321) = 14 = 3 + 2 + 1 + 4$, and $D_L(123 - 321) = W_L(202) = 4$.

Let $D_H(A, B)$ be the Hamming distance between two vectors $A$ and $B$, i.e., the number of positions in which $A$ and $B$ differ. Then $D_H(A, B) = D_H(A, B)$ when $k = 2$ or 3, and $D_H(A, B) \geq D_H(A, B)$ when $k > 3$.

Just as Hamming distance may be used to define the binary hypercube graph, $Q_2$, and the generalized hypercube graph [2], Lee distance may be used to define the k-ary n-cube graph, $Q^n_k$.

A $Q^n_k$ graph is a 2n-regular graph containing $k^n$ nodes. Each node is labeled with a distinct n-digit radix k vector $(a_0, a_1, a_2, \ldots, a_n)$. In this paper, node labels will be written as $(a_0, a_1, a_2, \ldots, a_n)$ or as $a_0, a_1, a_2, \ldots, a_n$ rather than n-tuples $(a_0, a_1, a_2, \ldots, a_n)$ when there is no confusion. In addition, the notation $U = (a_0, a_1, a_2, \ldots, a_n)$ will mean that node $U$ has address label $(a_0, a_1, a_2, \ldots, a_n)$. If $U$
and V are two nodes in the graph of a $Q_n^k$, then there is an edge between them if and only if $D_2(U, V) = 1$. From the definition of Lee distance, it can be seen that every node in $Q_n^k$ shares an edge with two nodes in every dimension, resulting in a graph of degree 2n. In addition, the shortest path between any two nodes, $U$ and $V$, has length $D_2(U, V)$. Since $Q_n^k = Q_n$ when $k = 2, k \geq 3$ will be assumed in the sequel.

Given $Q_n^k$, the subcube, $Q_n^k, p < n$, is denoted by strings over $\{0, 1, 2, \ldots, (k - 1), *\}^n$, where $p$ component of the string are $\ast$, the don't care symbol. For example, the two-dimensional subcube of $Q_3^3$ formed by the nodes 000, 001, 002, 010, 011, 012, 020, 021, and 022 is denoted by $0**$.

An edge joining two nodes whose addresses differ in the ith position is said to be an edge of dimension $i$, and the two nodes are said to be opposite in dimension $i$. For example, if $U = (000)$ and $V = (001)$ are nodes in $Q_3^3$, then the edge $(U, V) = (000, 001)$ is an edge of dimension 0, and $U$ and $V$ are nodes opposite in dimension 0.

In addition, other facts about $Q_n^k$ include the following.

**DIAMETER.** Let $D$ be the diameter of a $Q_n^k$, then $D = n\lfloor \frac{k}{2} \rfloor$.

**LINKS.** Let $L$ be the total number of links in a $Q_n^k$, $k > 2$, then $L = nk^2$.

**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 $d < \frac{n}{2}$. That is, $A_n^k(d)$ is the surface area of a sphere of radius $d$. Then

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

**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 $d < \frac{n}{2}$. That is, $V_n^k(d)$ is the volume of a sphere of radius $d$. Then

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

**B. The Product of Graphs**

Given graphs $G_1 = (V_1, E_1), G_2 = (V_2, E_2), \ldots, G_p = (V_p, E_p)$, define the cross product of $G_1, G_2, \ldots, G_p$, denoted by $G_1 \otimes G_2 \otimes \cdots \otimes G_p$ as the graph $G = (V, E)$, where

$$V = \left\{ \left( v_1, v_2, \ldots, v_p \right) \mid v_i \in V_i, 1 \leq i \leq p \right\}$$

and

$$E = \left\{ \left( (u_1, u_2, \ldots, u_p), (v_1, v_2, \ldots, v_p) \right) \mid i, j \leq p, i \neq j \text{ such that } (u_i, v_i) \in E_i \text{ and } u_j = v_i \text{ for } j \neq i \right\}$$

Define a $k$-ary cycle of length $k$, denoted by $G_k$, as a graph consisting of $k$ vertices and $k$ edges. Each vertex is labeled with a radius $k$ number, $0 \ldots k - 1$. There is an edge between vertices $u$ and $v$ if and only if $D_2(u, v) = 1$.

A $Q_n^k$ can alternately be defined as a product of cycles. That is

$$Q_n^k = C_k \otimes C_k \otimes \cdots \otimes C_k$$

The above demonstrates a useful topological characteristic of a $Q_n^k$: A $Q_n^k$ can be recursively defined in terms of smaller cubes.

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

Define a $k$-ary ring in dimension $l$ to be a sequence of node addresses $(A_0, A_1, \ldots, A_{k-1})$ such that $D_2(A_i, A_j) = 1$ if and only if $i = j + 1 \mod k$ and all node addresses are identical except in position $l$. The following theorem follows directly from the fact that a $Q_n^k$ is the product of $n$ cycles of length $k$.

**THEOREM 1.** A $Q_n^k$ contains $k^{n-1}$ node-disjoint $k$-ary rings in each dimension.

**EXAMPLE 1.** The four node-disjoint rings in each dimension for the $Q_2^4$.

<table>
<thead>
<tr>
<th>Direction 1</th>
<th>Direction 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1 = {00, 01, 02, 03}$</td>
<td>$R_1 = {00, 10, 20, 30}$</td>
</tr>
<tr>
<td>$R_2 = {10, 11, 12, 13}$</td>
<td>$R_2 = {10, 11, 21, 31}$</td>
</tr>
<tr>
<td>$R_3 = {20, 21, 22, 23}$</td>
<td>$R_3 = {02, 12, 22, 32}$</td>
</tr>
<tr>
<td>$R_4 = {30, 31, 32, 33}$</td>
<td>$R_4 = {03, 13, 23, 33}$</td>
</tr>
</tbody>
</table>

**C. Graph Embeddings**

This section reviews some basic terminology of graph embedding. Let $G$ and $H$ be two undirected graphs where $G$ is called the guest graph and $H$ is called the host graph. Let $V_G, E_G, V_H, E_H$ 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, $(x_1, x_2, \ldots, x_n) \in P_H$ 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 $(f_G, f_E)$ where $f_V: V_G \rightarrow V_H$ and $f_E: E_G \rightarrow P_H$. Also,

$$(a, b) \in E_G \Rightarrow f_E(a, b) = (x_1, \ldots, x_n) \text{ such that }$$

$$(x_1, \ldots, x_n) \in P_H, x_1 = f_G(a), \text{ and } x_n = f_G(b)$$

Given graphs $G$ and $H$ with an embedding $(f_G, f_E)$ of $G$ into $H$, the following terms are used to describe the embedding. For more information see [9].

**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 $\frac{|V_G|}{|V_H|}$ 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 $f_G$.

If an embedding of a graph $G$ into a graph $H$ can be found having dilation, congestion and load equal to one, then $G$ is isomorphic to a subgraph of $H$. 
III. DISJOINT PATHS

The transfer of a large amount of data between two nodes in a multicomputer may be facilitated by dividing the data into small packets and sending the packets along different routes. In order to avoid contention, the packets should travel by routes having no common nodes except the sending and receiving nodes. Such paths between two nodes A and B, referred to as node-disjoint parallel paths, provide a means of selecting alternate routes between A and B and increase fault-tolerance. The following theorem states the number and lengths of disjoint-parallel paths between any two nodes belonging to $Q_n^k$.

**Theorem 2.** Given $A = (a_1, a_2, \ldots, a_0)$ and $B = (b_1, b_2, \ldots, b_0)$, let $l = D_0(A, B)$, $h = D_0(A, B)$, and $w_i = D_0(a_i, b_i)$ for $0 \leq i \leq n-1$. Then in a $Q_n^k$, $k > 2$, there are a total of $2n$ node-disjoint parallel paths between A and B of which

1) $h$ paths have length $l$,
2) $2(n-h)$ paths have length $l + 2$,
3) for each $i$ such that $w_i > 0$, there is a path of length $l + k - 2w_i$ (for a total of $h$ paths).

**Proof.** Without loss of generality, assume that the first $h$ digits of the addresses of A and B are different while the remaining $n-h$ digits are the same. Then

1) For each $i$, $0 \leq i < h$, construct the $i$th path as follows. Starting with the address of A, correct digit $i$ using the shortest path in the ring of dimension $i$ between $a_i$ and $b_i$. Repeat this procedure for the remaining digits of A, proceeding sequentially through dimensions $i+1$, $i+2$, $\ldots$, $h-1$, $0$, $\ldots$, $i-1$. This produces $h$ paths of length $l$.

2) Construct the next $2(n-h)$ paths of length $l + 2$ from A to B by the following. First, for each $i$, $h \leq i < n$, add 1 to digit $i$. Then follow the correction procedure of 1) for digits $i$, $1 \leq i \leq h$, and finish by subtracting 1 from digit $i$. This results in $n-h$ paths of length $l + 2$. For the remaining $n-h$ paths, repeat this procedure but subtract 1 from digit $i$ first and finish by adding 1 to digit $j$. This step produces $2(n-h)$ paths of length $l + 2$.

3) Construct the remaining $h$ paths as follows. For each $i$, $0 \leq i < h$ add or subtract 1 to move along the longest ring in dimension $i$ between $a_i$ and $b_i$. This correction of digit $i$ is the opposite of the correction in step 1). Now, correct each of the remaining digits following the shortest path in the rings of dimension $i+1$, $i+2$, $\ldots$, $h-1$, $0$, $\ldots$, $i-1$. Finally complete the path to B by continuing to correct digit $i$ following the longest path in dimension $i$. The length of each path may be calculated as follows. Correcting digit $i$ using the longest path in the ring in dimension $i$ uses $(k-w_i)$ steps. Correcting the remaining digits using the shortest path in each ring requires $(l-w_i)$. Altogether, the length of each path is $l + k - 2w_i$.

It can be seen that none of these paths share any nodes except A and B.

Note the theorem above is not valid when $k = 2$. This is because adding one to a bit is the same as subtracting one from a bit in the binary case. Example 2 shows the six disjoint parallel paths between two nodes labeled 013 and 034 in a $Q_n^2$.

**Example 2.** The six disjoint parallel paths between 013 and 034 in a $Q_2^3$.

Path 1: 013 $\rightarrow$ 014 $\rightarrow$ 024 $\rightarrow$ 034
Path 2: 013 $\rightarrow$ 023 $\rightarrow$ 034
Path 3: 013 $\rightarrow$ 113 $\rightarrow$ 114 $\rightarrow$ 124 $\rightarrow$ 134 $\rightarrow$ 034
Path 4: 013 $\rightarrow$ 013 $\rightarrow$ 114 $\rightarrow$ 124 $\rightarrow$ 134 $\rightarrow$ 034
Path 5: 013 $\rightarrow$ 012 $\rightarrow$ 022 $\rightarrow$ 032 $\rightarrow$ 031 $\rightarrow$ 030 $\rightarrow$ 034
Path 6: 013 $\rightarrow$ 003 $\rightarrow$ 004 $\rightarrow$ 044 $\rightarrow$ 034

IV. EMBEDDINGS

This section considers embedding various structures into a $Q_n^k$. The structures considered are a ring, a mesh, and a binary hypercube ($Q_n$).

A. Embedding a Ring

Let $A = (a_1, a_2, \ldots, a_r)$ be a sequence of node addresses in a $Q_n^k$. A is called a ring, or cycle, of length $r$ if

1) $A_i \neq A_j$ for $1 \leq i, j \leq r$,
2) $D_0(A_i, A_i+1) = 1$ for $1 \leq i \leq (r-1)$, and
3) $D_0(A_i, A_1) = 1$.

A ring of length $N$, where $N = k^r$ is called Hamiltonian or a Hamiltonian cycle.

Let $A$ be a sequence of node addresses forming a ring in a $Q_n^k$. Since the Lee distance between any two successive addresses must be 1, the sequence of node addresses, A, forms a Gray code.

The preceding suggests that one means of generating a Hamiltonian cycle is to generate a Gray code. The following theorem presents a method for constructing a $k$-ary Gray code.

**Theorem 3.** Given $f: \{0 \ldots k-1\}^n \rightarrow \{0 \ldots k-1\}^n$ where

$f(a_1, a_2, \ldots, a_0) = a_{n-1} \oplus (a_{n-2} \oplus (a_{n-2} \oplus \ldots \oplus (a_0 \oplus a_1))$ and a sequence of $n$ digit, radix $k$ numbers

$S = (00 \ldots 00), (00 \ldots 01), \ldots, (00 \ldots 0(k-1)),$

$(00 \ldots 10), \ldots, ((k-1)(k-1) \ldots (k-1)),$

then the sequence

$S' = (f(00 \ldots 00), f(00 \ldots 01), \ldots, f(00 \ldots 0(k-1)),$

$f(00 \ldots 10), \ldots, f((k-1)(k-1) \ldots (k-1))$ forms a Gray code.

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

$A = \{a_{m-1}, a_{m-2} \cdots a_{m+1}\}^* a_m (k-1 \ldots (k-1)) *$

$B = \{b_{m-1}, b_{m-2} \cdots b_{m+1}\}^* a_m + 1 \{00 \ldots 0\}^*$

where sequences marked with * may or may not exist depending on the value of $m$ and addition is mod $k$. Then,
$$f(A) = \{x_{n-1} x_{n-2} \cdots x_m\}^* \ x_m \ \{x_{m-1} 0 \cdots 0\}^*$$

where

$$x_{n-1} = a_{n-1}$$

$$x_i = (a_i - a_{i+1}) \mod k \ for \ i = n-2 \ldots m + 1$$

$$x_m = a_m - a_{m+1} \mod k$$

$$x_{m-1} = ((k-1) - a_m) \mod k$$

and

$$f(B) = \{x_{n-1} x_{n-2} \cdots x_m\}^* \ y_m \ \{y_{m-1} 0 \cdots 0\}^*$$

where

$$y_m = (a_m + 1) - a_{m+1} \mod k$$

$$= (x_m + 1) \mod k$$

$$y_{m-1} = 0 - (a_m + 1) = (k-1) - a_m \mod k$$

$$= x_{m-1}$$

Therefore, $f(A)$ and $f(B)$ differ in the $m$th digit only and $D_k(f(A), f(B)) = 1$.

Note that the function

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

also generates a Gray code. The proof is similar to Theorem 3.

By using Theorem 3, a Hamiltonian cycle can be generated for any $Q_k^1$. First, the sequence

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

is generated. Then the sequence

$$S' = \{f(0), f(1), \ldots, f(k^n - 1)\}$$

is obtained, giving a Hamiltonian cycle. Two examples are given below. In Example 3, a Hamiltonian cycle for a $Q_2^1$ is given, and in Example 4, a Hamiltonian cycle for a $Q_3^1$ is given.

**Example 3.** A Hamiltonian cycle in a $Q_2^1$. $R$ represents the radix representation of a node address and $f(R)$ the transformation under the mapping $f$ defined in Theorem 3.

<table>
<thead>
<tr>
<th>$R$</th>
<th>$f(R)$</th>
<th>$R$</th>
<th>$f(R)$</th>
<th>$R$</th>
<th>$f(R)$</th>
<th>$R$</th>
<th>$f(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>10</td>
<td>13</td>
<td>20</td>
<td>22</td>
<td>30</td>
<td>31</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>11</td>
<td>10</td>
<td>21</td>
<td>23</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>12</td>
<td>11</td>
<td>22</td>
<td>20</td>
<td>32</td>
<td>33</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
<td>13</td>
<td>12</td>
<td>23</td>
<td>21</td>
<td>33</td>
<td>30</td>
</tr>
</tbody>
</table>

**Example 4.** A Hamiltonian cycle in $Q_3^1$. $R$ represents the radix representation of a node address and $f(R)$ the transformation under the mapping $f$ defined in Theorem 3.

<table>
<thead>
<tr>
<th>$R$</th>
<th>$f(R)$</th>
<th>$R$</th>
<th>$f(R)$</th>
<th>$R$</th>
<th>$f(R)$</th>
<th>$R$</th>
<th>$f(R)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>10</td>
<td>14</td>
<td>20</td>
<td>22</td>
<td>30</td>
<td>32</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>11</td>
<td>10</td>
<td>21</td>
<td>24</td>
<td>31</td>
<td>33</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>12</td>
<td>11</td>
<td>22</td>
<td>20</td>
<td>32</td>
<td>42</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
<td>13</td>
<td>12</td>
<td>23</td>
<td>21</td>
<td>33</td>
<td>43</td>
</tr>
<tr>
<td>04</td>
<td>04</td>
<td>14</td>
<td>13</td>
<td>24</td>
<td>22</td>
<td>34</td>
<td>41</td>
</tr>
</tbody>
</table>

Now suppose $R = r_{n-1} r_{n-2} \cdots r_0$ is a radix $k$ number and $G = g_{n-1} g_{n-2} \cdots g_0$ is the Gray code representation given by the function $f$ defined in Theorem 3. That is $G = f(R)$. Then the inverse of $f$, $f^{-1}$, can be obtained in the following manner.

$$R = f^{-1}(G)$$

where

$$r_{n-1} = g_{n-1}$$

$$r_i = (r_{i+1} + g_i) \mod k$$

$$= \left(\sum_{j=1}^{i} g_i\right) \mod k \ i = n-2, n-3, \ldots, 0$$

This is because $f(r_i) = g_i = r_i - r_{i+1}$.

Therefore

$$r_i = r_{i+1} + g_i$$

$$= (r_{i+2} + g_{i+1}) + g_i$$

$$\vdots$$

$$= r_{n-1} + g_{n-2} + \cdots + g_1$$

$$= g_{n-1} + g_{n-2} + \cdots + g_0$$

Another function, $f'$, which also generates a Gray code is presented below. An interesting aspect of $f'$ is that the resulting code has reflective properties. Let $n = 2m$ for some integer $m$ and $S = (G_0, G_1, \ldots, G_{n-1})$ be a Gray code sequence of length $n$. $S$ is called reflective if $D_k(G_i, G_{i+1}) = 1$ for $i = 0, 1, \ldots, m - 1$. $S$ is called block reflective for block length $k'$ if it can be divided into blocks $B_0, B_1, \ldots, B_{k'-1}$, each of length $k'$, such that adjacent blocks are reflective. That is, block $B_j = \{G_{jk'}, G_{jk'+1}, \ldots, G_{jk'+k'-1}\}$, for $j = 0, \ldots, k'-1$ and the sequence $\{B_0, B_{k'-1}\}$ is not reflective, then $S$ is called partially block reflective for block length $k'$.

To define $f'$, let $R = r_{n-1} r_{n-2} \cdots r_0$ be a radix $k$ number and $G = g_{n-1} g_{n-2} \cdots g_0$ the Gray code representation given by $f'$, i.e., $G = f'(R)$, where

$$g_{n-1} = r_{n-1}$$

and

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

or, if $k$ is odd,

$$g_i = \begin{cases} r_i, & \text{if } r' \text{ is even} \\ k-1-r_i, & \text{if } r' \text{ is odd} \end{cases}$$

Let $S = \{R_0, R_1, \ldots, R_{k'-1}\}$ be a sequence of $n$ digit, radix $k$ numbers and let $S' = \{G_0, G_1, \ldots, G_{k'-1}\}$ where $G_i = f'(R_i)$ for $0 \leq i \leq k'-1$. For even values of $k$, $S'$ forms a Hamiltonian cycle. If, however, $k$ is odd, $S'$ forms a Hamiltonian path. In addition, if $k = 2$ or $4$, $S'$ is reflective. Otherwise, $S'$ is block reflective for block lengths $k, k', \ldots, k^{-1}$ when $k$ is even and partially block reflective when $k$ is odd.
Two examples are given later. In Example 5, \( k = 4 \) and \( n = 3 \).
In Example 6, \( k = 5 \) and \( n = 2 \).

**Example 5.** A three digit, 4-ary Gray code sequence which gives a Hamiltonian cycle and is reflective.

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

**Example 6.** A two digit, 5-ary Gray code sequence which gives a Hamiltonian path and is partially block reflective.

<table>
<thead>
<tr>
<th>R</th>
<th>( \gamma^*(R) )</th>
<th>R</th>
<th>( \gamma^*(R) )</th>
<th>R</th>
<th>( \gamma^*(R) )</th>
<th>R</th>
<th>( \gamma^*(R) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>10</td>
<td>14</td>
<td>20</td>
<td>20</td>
<td>30</td>
<td>34</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>11</td>
<td>13</td>
<td>21</td>
<td>21</td>
<td>31</td>
<td>33</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>12</td>
<td>13</td>
<td>22</td>
<td>22</td>
<td>32</td>
<td>32</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
<td>13</td>
<td>12</td>
<td>23</td>
<td>23</td>
<td>33</td>
<td>31</td>
</tr>
<tr>
<td>04</td>
<td>04</td>
<td>14</td>
<td>10</td>
<td>24</td>
<td>24</td>
<td>34</td>
<td>30</td>
</tr>
</tbody>
</table>

B. Embedding a Mesh

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 \( a_n \cdots a_2 \cdots a_0 \) where \( a_i \) is radix \( K_i \). Each node has, at most, a connection to \( 2n \) other nodes. Given a node \( A \) with address \( a_{n-1} \cdots a_i \cdots a_0 \) for each \( i, 0 \leq i < n \), \( A \) 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 can be used to embed meshes of certain dimensions into a \( Q^4 \). Let \( M \) be a \( k^n \times k^n \times \cdots \times k^n \)-dimensional mesh, and \( Q \) be a \( Q^4 \), where \( n = \sum_{i=1}^m n_i \). The following construction shows how to embed \( M \) into \( Q \).

Assume that each dimension \( i \) of \( M \) is labeled with a radix \( k_i, n_i \) digit number \( 0 \cdots k_i^{n_i} - 1 \). Using Theorem 3 relabel each dimension with the corresponding Gray code sequence. Now each node of \( M \) 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 \( x \) is a node of \( M \) with label \( (x_1, x_2, \cdots, x_m) \), then define \( f(x) = x_1 x_2 \cdots x_m \), the concatenation of \( x_1, \cdots, x_m \).

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

As an example, Fig. 1 shows \( 4^1 \times 4^3 \)-dimensional mesh. The mesh has both sequential and Gray code labels. The node X has address \((21, 11)\) which, when translated to the Gray code, becomes \((23, 10)\). In the mesh, X is adjacent to four 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 \( Q^4 \), 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 four nodes in the \( Q^4 \).

C. Embedding a Hypercube

This section considers embedding a \( Q_n \) in a \( Q^4 \). To begin, consider the following lemma.

**Lemma 1.** Let \( f : \{0, 1\}^2 \rightarrow \{0, 1, 2, 3\} \) where \( f(00) = 0 \), \( f(01) = 1 \), \( f(10) = 2 \), and \( f(11) = 3 \). Then \( f(a) \) maps the two digit binary reflective Gray code onto the single digit 4-ary reflexive 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 \( u, v \in \{0, 1\}^2 \) and \( D_2(u, v) = 1 \), then it can be seen that \( D_2(f(u), f(v)) = 1 \).

**Lemma 2.** Let \( H \) be a hypercube of dimension \( 2d \) and \( K \) be a \( Q_d^4 \).

Let \( u = (a_{2d} a_{2d-1} \cdots a_1) \) and \( v = (b_{2d} b_{2d-1} \cdots b_1) \) be vertices of \( H \) and \( u' = (a'_{2d} a'_{2d-1} \cdots a'_1) \) and \( v' = (b'_{2d} b'_{2d-1} \cdots b'_1) \) be vertices of \( K \). Let \( f_2(u) = u' \) where \( a'_i = f(a_{2i-1}, a_{2i}) \) and \( f \) is the function defined in Lemma 1. Further, let \( f_2(u, v) = (u', v') = (f_2(u), f_2(v)) \). Then the pair \( (f_2, f_2) \) embeds \( H \) into \( K \).

**Proof.** First note that \( H \) and \( K \) have the same number of vertices. The number of vertices of \( K \) is \( 4^d = 2^{2d} \) which is the number of vertices of \( H \). Also note that the function \( f_2 \) maps nodes from \( H \) onto nodes from \( K \) uniquely. Showing the pair \( (f_2, f_2) \) is an embedding requires showing that adjacent nodes in \( H \) are mapped to adjacent nodes in \( K \). Suppose \( u \) and \( v \) are adjacent in \( H \). Then the labels of \( u \) and \( v \) differ in some position, say \( j \), \( 1 \leq j \leq 2d \). Without loss of generality, assume \( j = 2p \). Then \( D_2(a_{2j-1}, b_{2j-1}) = D_2(a_{2j-1}, b_{2j-1}) = 1 \). By Lemma 1, however, \( D_2(\{f(a_{2j-1}), f(a_{2j-1})\}) = D_2(a'_{2j-1}, b'_{2j-1}) = 1 \). But, \( u' \) and \( v' \) differ only in position \( p \); therefore, \( D_2(u', v') = 1 \), and \( u' \) and \( v' \) are adjacent in \( K \).

**Corollary 1.** Suppose \( H \) is a hypercube of dimension \( 2d - 1 \) and \( K \) is a \( Q_d^4 \). Then using the notation of Lemma 2 with \( f_2(u) = u' \) 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 \( (f_2, f_2) \) embeds \( H \) into \( K \).

**Proof.** The proof follows directly from Lemma 2.

Taken together, Lemmas 1, 2, and Corollary 1 demonstrate how to embed a \( Q_3 \) into a \( Q_4^4 \). Let the address label of node \( x \) be \( (a_k a_{k-1} \cdots a_1) \). If \( n \) is odd, relabel \( x \) as \( (0 a_k a_{k-1} \cdots a_1) \). Then let node \( x' \) of \( Q_4^4 \) have address label \( (b_i b_{i-1} \cdots b_1) \) where \( j = \left\lfloor \frac{x}{2} \right\rfloor \), \( b_j = f(a_{2j}, a_{2j-1}) \), \( 1 \leq i \leq j \), and \( f \) defined as in Lemma 1. As an example, Fig. 2 shows a \( Q_3 \) and its embedding into a \( Q_4^4 \). In the figure, the node labels of the \( Q_3 \) have a zero prepended.

**V. Resource Placement**

This section discusses an application of a Lee distance-error-correcting code to resource placement in a \( Q_d^A \). The terms node and processor will be used interchangeably throughout this section.

In a multicomputer, there may be a resource, such as an I/O processor or a software package, that each processor needs to access. However, because of expense or frequency of use, it may not be desirable to place a copy of the resource at each node in the system. In general, then, the problem of resource placement is how should a limited number of copies of a resource be disseminated throughout a system giving comparable access to all processors. Problems in resource placement fall into two general classes.

The first class of problem is the \( j \)-adjacency problem. This problem considers placing resources such that each processor in the system either has a copy of the resource or is adjacent to \( j \) processors having a copy of the resource. In this case, one processor is adjacent to another if they share a link, and processors having the resource are known as resource processors or resource nodes. The resource distance of a node is the shortest distance from the node to a resource node, and the resource diameter is the maximum resource distance taken over all nodes.

The second class of problem is referred to here as the \( t \)-embedding problem. This problem considers placing resources such that each nonresource node has a resource distance of \( t \) or less and no two resource nodes are within distance \( 2t + 1 \) of each other. If each nonresource node has resource distance \( t \) or less from exactly one resource node, the \( t \)-embedding is perfect.

Note that when \( j = t = 1 \), the \( j \)-adjacency problem and the \( t \)-embedding problem are the same.

Recently, there have been several papers on resource placement in hypercube computers [4], [5], [10], [14]. Several papers approached the problem by using error-correcting codes. The approach used in [14] is summarized below.

A perfect embedding is defined as a configuration where each nonresource node in a \( Q_n \) has a resource distance one and no two resource nodes are adjacent. A \( Q_n \) has a perfect embedding if and only if \( n = 2^r - 1 \) for some integer \( r \). To find this embedding, let \( k = n - r \) and construct the \((n, k)\) Hamming code. By placing a copy of the resource at each node whose address is a code word, an embedding is found. This configuration satisfies the definition of a perfect embedding because a Hamming code is both a single-error correcting and a perfect code.

In [13], Ramanathan and Chalasani consider the \( j \)-adjacency problem in a \( Q_n^4 \). A perfect \( j \)-adjacency placement is defined as a placement of resources where each nonresource node is adjacent to \( j \) resource nodes and no two resource nodes are adjacent. They use an approach based on a node address (considered as an \( n \)-tuple) being orthogonal to a specified \( n \)-tuple called the characteristic \( n \)-tuple of a perfect \( j \)-adjacency placement.

Finding a placement based on error-correcting codes is more difficult for a \( Q_n^A \) than for a \( Q_n \). First, the address space...
of a $Q^k_d$ is not necessarily a vector space, and, second, there is much more known about Hamming distance codes than about Lee distance codes.

There is application, however, for a Lee distance single error-correcting code to the 1-adjacency problem. Presented below is a code based on a proof by Berlekamp [1, Theorem 13.21, p. 304].

Let $k$ be an odd integer and let $n = k^r - 1$ for some integer $r$. Then a perfect Lee distance single error-correcting code may be found as follows.

Let $H$ be an $r \times n$ matrix. $H$ is called a parity check matrix. The columns of $H$ consist of nonzero $r \times 1$ vectors,

$$V_i = \begin{bmatrix} v_{i,1} \\ v_{i,2} \\ \vdots \\ v_{i,r} \end{bmatrix}$$

such that $0 \leq v_{ij} \leq \left\lfloor \frac{k}{2} \right\rfloor$ where $j$ is the smallest integer such that $v_{ij} \neq 0$. Then an $n$ digit, radix $k$ vector, $A = a_{n-1} a_{n-2} \cdots a_0$ is a code word if and only if $A \cdot H^T = 0$.

**Example 7.** Let $k = 5$ and $r = 2$. Then $n = 5^2 - 1 = 12$ and

$$H = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 0 & 1 & 2 & 3 & 4 & 0 & 1 & 2 & 3 & 4 \end{bmatrix}$$

Since $n = 12$ and $r = 2$, there are $5^{10}$ vectors orthogonal to $H$. These vectors form a perfect single error correcting Lee distance code, and placing resources at nodes in a $Q^5_{12}$ corresponding to code words results in a solution to the 1-adjacency (or 1-embedding) problem.

When $k$ is a prime, the resulting code is equivalent to a negacyclic code [1, ch. 9]. Negacyclic codes correcting $t$ errors, where $2t - 1 < k$, can be constructed. It would be useful if, for example, the 2-error correcting negacyclic code for the $Q^5_{12}$ would result in a perfect 2-embedding. Unfortunately, this is not the case.

It was conjectured in [8] that there are no perfect Lee distance code blocks length $n > 2$, with radix $k > 3$, and correcting $t > 1$ errors, and the only known perfect codes based on Lee distance correcting $t$-errors ($t > 1$), are given in a theorem by Berlekamp [1, p. 305]. This theorem states that, for any given $t$, there exists a perfect $t$-error correcting code based on Lee distance where $n = 2$, $k = 2^t + 2t + 1$, and the generator polynomial $g(x) = x + (2t + 1)$. Thus, for example, a perfect solution to the 2-embedding problem can be found for the $Q^5_{12}$, a perfect solution to the 3-embedding problem can be found for the $Q^5_{25}$, etc.

It might be more fruitful to investigate whether or not negacyclic codes can be used to find quasi-perfect $t$-embeddings. A placement is quasi-perfect if each nonresource node has resource distance $t + 1$ or less and no two resource nodes are within distance $2t + 1$ of each other.

## VI. BROADCASTING AND ROUTING

This section considers two single node broadcasting algorithms and a message routing algorithm based on Lee distance. In single node broadcasting, one processor sends the same message to all other processors in the system. In message routing, one node sends a single message to one other node.

Broadcasting algorithms generally make one of two assumptions about the system. The first assumption is single-port I/O. Under this assumption 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 several dimensions simultaneously. In a $Q^k_d$, the multi-port I/O assumption usually implies that a node can transmit to all $2n$ neighboring nodes at the same time and that the transmission is full-duplex.

A useful property of the graph of a $Q^k_d$ is vertex symmetry. A graph is vertex symmetric if, for any two vertices of the graph $a$ and $b$, there is an automorphism mapping $a$ to $b$. This property is useful in broadcasting and routing algorithms because the source node can be assumed to have address label $(00 \cdots 0)$.

### A. Single Node Broadcasting

Two single node broadcasting algorithms will be presented. The first will construct a spanning tree and, assuming multi-port I/O, will be shown to be optimal. The second will use the decomposition of a $Q^k_d$ into a product of cycles. When $k$ is even, the second algorithm will be shown to be optimal for single-port I/O and, when $k$ is odd, optimal for 2-port I/O.

Starting with node $(00 \cdots 0)$, the first single node broadcasting algorithm constructs a spanning tree. Given a node $u$ with address $(a_{m-1} a_{m-2} \cdots a_0)$, the addresses of its children are obtained by the following rule.

Let $m = \max(i \mid a_i \neq 0, 0 \leq i < n)$. Then $m$ is the index of the leftmost nonzero digit. Let a child node be denoted by $(b_{m-1} b_{m-2} \cdots b_0)$. If $m < n - 1$, then for each $j$, $m < j < n$, $u$ will have two children where $b_j = a_j$ except for $i = j$. For one child $b_j = 1$ and for the other child $b_j = k - 1$.

In addition, if $a_m < \left\lfloor \frac{k}{2} \right\rfloor$, then $u$ has a child where $b_i = a_i$ except for $i = m$ and $b_m = a_m + 1$. If $a_m > \left\lfloor \frac{k}{2} \right\rfloor + 1$, then $u$ has a child where $b_i = a_i$ except for $i = m$ and $b_m = a_m - 1$.

![Fig. 3. A spanning tree for single node broadcasting in a 5-ary 2-cube.](image-url)

Using this rule, note that nonleaf nodes may have up to $2(n - l) + 1$ children, and leaf nodes have either $\left\lfloor \frac{k}{2} \right\rfloor$ or $\left\lfloor \frac{k}{2} \right\rfloor + 1$ as their leftmost digit. As an example, Fig. 3 show the span-
ning tree constructed for $Q^2_n$. Assuming multiport communication, the broadcast takes four time steps.

Once a spanning tree has been constructed, it may be used to broadcast from any node in the network. To broadcast from the node $x$ with address label $(a_{n,1}, a_{n,2}, \ldots, a_{n,n})$, simply add the vector $(k - a_{n,1}, k - a_{n,2}, \ldots, k - a_{n,n})$ to the address label of each node in the network. This will result in $x$ having the label $(00 \ldots 0)$, and the remaining nodes relabeled accordingly.

To analyze the time complexity of this algorithm, first note that, in each dimension, the address either monotonically increases from 0 to $\lceil k/2 \rceil$ or monotonically decreases from 0 to $\lfloor k/2 \rfloor$. Second, consider the following simple lemma.

**Lemma 3.** Given the sequence $(0, 1, \ldots, k - 1)$, then $D_2(0, \lfloor k/2 \rfloor) \leq D_2(0, \lceil k/2 \rceil + 1)$.

**Proof.** Suppose $k$ is even. Then $\lfloor k/2 \rfloor = k/2$, and

\[
D_2(0, \lfloor k/2 \rfloor) = D_2(0, k/2) = k/2
\]

Also, $D_2(0, \lceil k/2 \rceil + 1) = D_2(0, k/2 + 1) = k/2 + 1 = k - (k/2) = k/2 - 1$.

Suppose $k$ is odd. Then $\lfloor k/2 \rfloor = k/2 - 1/2$, and

\[
D_2(0, \lfloor k/2 \rfloor) = D_2(0, k/2 - 1/2) = (k - 1)/2
\]

Also, $D_2(0, \lceil k/2 \rceil + 1) = D_2(0, k/2 + 1) = (k + 1)/2 = k - (k/2 + 1/2) = (k - 1)/2$.

The time complexity of the first broadcast algorithm will be calculated with the following two assumptions. First, it takes one time step to traverse one link in the tree, and, second, a node may communicate with all $2n$ neighbors simultaneously (multi-port I/O).

Combining the monotonicity of addressing with Lemma 3, it is seen that the longest path in the tree will be from $(00 \ldots 0)$ to $(\lfloor k/2 \rfloor, \lfloor k/2 \rfloor, \ldots, \lfloor k/2 \rfloor, \lfloor k/2 \rfloor)$. Under the multi-port I/O assumption, the time to broadcast the message to reach the most remote node, and this is:

\[
\begin{align*}
\left(\frac{k}{2}\right)^n + \left(\frac{k}{2}\right)^n + \ldots + \left(\frac{k}{2}\right)^n &= n \left(\frac{k}{2}\right)^n
\end{align*}
\]

Since the diameter of $Q^2_n$ is also $n \left(\frac{k}{2}\right)^n$, the above broadcasting algorithm is optimal.

To illustrate the second algorithm, assume single-port I/O, $k$ is even, and that node $O = (00 \ldots 0)$ wants to broadcast a message. The procedure will be as follows. $O$ will first send the message along the ring in dimension 0. When the message has traversed the ring, each node in the subcube $00 \ldots 0^*$ will have a copy of the message. Each of these nodes then sends the message along the ring in dimension 1. In general, when each node in the $m$-dimensional subcube, $00 \ldots 0^{m+1}$ has the message, they each send the message along their ring in dimension $m+1$.

To show that this procedure is optimal when $k$ is even, suppose node $O = (00 \ldots 0)$ starts broadcasting at time $t = 0$ along the ring in dimension 0. That is, at time $t = 1$, $(00 \ldots 01)$ receives the message and sends it to $(00 \ldots 02)$ at time $t = 2$. Also, at time $t = 2$, $O$ sends the message to node $(00 \ldots 0(k - 1))$. Note that at time $t = k/2$, the message has reached node $(00 \ldots 0(k/2))$ while traveling in the positive direction, and it has reached node $(00 \ldots 0(k/2 - 1))$ while traveling in the negative direction. Since $k$ is even, however, $k/2$ and $k/2 - 1$ are adjacent, and the message has reached all nodes along the ring in dimension 0 by time $t = k/2$. There are $n$ dimensions through which the message must be passed, and it takes $n k/2$ time steps to pass through each dimension; therefore, the total time to broadcast the message is $n k/2$. When $k$ is even, $n k/2$ is also the diameter of a $Q^2_n$. Thus, this broadcast algorithm is optimal.

When $k$ is odd, the diameter of a $Q^2_n$ is $n \lfloor k/2 \rfloor$. In this case, however, $\lfloor k/2 \rfloor = k/2 - 1/2$. Using the algorithm above with single-port I/O, means that if $O = (00 \ldots 0)$ starts broadcasting at time $t = 0$ along the ring in dimension 0, by time step $\lfloor k/2 \rfloor$, the message has reached node $(00 \ldots 0(k/2 - 1))$ in the positive direction and node $(00 \ldots 0(k/2 + 1))$ in the negative direction, and the Lee distance between these two nodes is 2. Thus, it takes $\lfloor k/2 \rfloor$ time steps to broadcast a message in each dimension, and $n \lfloor k/2 \rfloor$ time steps are needed to broadcast throughout a $Q^2_n$ when using single-port I/O. While this can be shown to be non-optimal, it is within $n$ steps of the optimal and has the virtue of being implemented simply.

Suppose the broadcasting is done using 2-port I/O. Then at time step $t = 0$, $O = (00 \ldots 0)$ begins broadcasting to both node $(00 \ldots 01)$ and node $(00 \ldots 0(k - 1))$. By time step $\lfloor k/2 \rfloor$, the message will reach node $(00 \ldots 0(k/2 - 1))$ in the positive direction and node $(00 \ldots 0(k/2 + 1))$ in the negative direction. These two nodes are adjacent, and the message reaches all nodes in the ring by time step $\lfloor k/2 \rfloor$. Thus, the time to broadcast through-
out the $Q^k_n$ is $n\left\lceil \frac{k}{4} \right\rceil$; therefore, the broadcast algorithm is optimal when $k$ is odd if 2-port I/O is used.

### B. Dimensional Routing

In a direct network such as a $Q^k_n$, when node $x$ sends a message to node $y$, a path through the network along which the message will travel must be chosen. This process is called routing. For an excellent survey of routing technologies and algorithms for direct networks, see [11]. This section will briefly discuss how Lee distance is naturally applicable to dimensional wormhole routing in a $Q^k_n$.

In wormhole routing, a message is divided into small units called flits (flow control digits) which travel between nodes via routing chips. Each routing chip has a flit-sized buffer. If the channel to the next router is free, i.e., the buffer in the next router is unoccupied, the flit is sent through the communication channel. If the channel to the next router in the path is blocked, the flit is buffered at its current location.

When the header flit is sent along a communication channel, the remaining flits follow in a pipeline fashion. Should the leading flit be blocked because the communication channel ahead is occupied, the remaining flits in the message are also blocked. An advantage of wormhole routing is that message latency due to transit time (fly time) is less dependent on path length. A disadvantage of wormhole routing is that it is more susceptible to deadlock because of the way messages are blocked in place.

A routing algorithm is termed deterministic if the path selected does not depend on the current network condition. It is termed minimal if the path chosen by the algorithm is as short as any path between the source and destination nodes. A routing algorithm is dimensional if the path chosen takes the message through one dimension at a time.

A simple dimensional routing algorithm which is also minimal for a $Q^k_n$ can be constructed quite naturally using Lee distance. Let $S$ be a node with label $(s_{n-1}, s_{n-2}, \ldots, s_0)$ and $D$ be a node with label $(d_{n-1}, d_{n-2}, \ldots, d_0)$. 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 (+/-) and a magnitude $|x| \in \{0, \ldots, \left\lceil \frac{k}{4} \right\rceil\}$. Let $x = (s_{n-1}, s_{n-2}, \ldots, s_0)$. $x$ is the displacement vector where $x_i$ encodes the routing information for dimension $i$. That is, $x_i$ encodes both the sign and the magnitude.

Once $x$ has been calculated, the routing proceeds as follows. In dimension $i$, if $x_i$ is positive, the message will proceed from the node labeled $(a_{n-1}, a_{n-2}, \ldots, a_0)$ to the node labeled $(a_{n-1}, a_{n-2}, \ldots, a_i + 1, a_0)$. At each node, the magnitude $|x|$ is decremented. When $|x| = 0$, the flit is dropped from the header, and the message turns, moving in dimension $i + 1$. This procedure is the same if $x_i$ is negative except the message proceeds from the node labeled $(a_{n-1}, a_{n-2}, \ldots, a_i)$ to the node labeled $(a_{n-1}, a_{n-2}, \ldots, a_i - 1, a_0)$. Note that the addition and subtraction are mod $k$. Finally, the message is routed in dimensional order, starting from dimension 0 and ending with dimension $n - 1$.

The displacement vector $x$ can be calculated in the following manner. For each $i$, $0 \leq i < n$, let $\hat{x} = (d_i - x_i) \mod k$. If $\hat{x} \leq \left\lceil \frac{k}{4} \right\rceil$, then $|x| = \hat{x}$ and $x_i$ is positive. Otherwise, $|x| = k - \hat{x}$ and $x_i$ is negative.

**Example 8.** Let $S$ be labeled (634) and $D$ be labeled (026) in a $Q^4_8$. Suppose $S$ wants to send a message to $D$. Then, using subtraction mod 7,

$$\begin{align*}
\hat{x}_4 &= (0 - 6, \hat{x}_1 = (2 - 3, \hat{x}_0 = (6 - 4) = 1, x_1 = 6, x_0 = 2, \\
|\hat{x}_4| &= 1, |\hat{x}_1| = 1, |\hat{x}_0| = 2 \\
x_2 &= +1, x_1 = -1, x_0 = +2
\end{align*}$$

Using the displacement vector $x = (+1 -1 +2)$, the message follows the path

$$S = 634 \rightarrow 635 \rightarrow 636 \rightarrow 626 \rightarrow 026 = D$$

### VII. Conclusions

The thesis of this paper is that Lee distance is a natural metric to use with a $Q^k_n$. To support this conjecture, we have shown how various topological properties can be viewed using Lee distance. It is our feeling that there are other aspects of a $Q^k_n$ that might be explored fruitfully using Lee distance. In the future, we would like to consider these additional problems.

It is clear that when $k$ is even, a $Q^k_n$ is a bipartite graph; therefore, it has cycles of even lengths from 4 to $k^2$. We intend to use Lee distance and Gray codes to devise a simple algorithm generating these cycles.

It is easy to embed a binary tree of height $h$ (where the root is at height 0) into a $Q^h_2$. However, the number of nodes in a tree of height $h$ is $O(2^h)$ and the number of nodes in a $Q^h_2$ is $O(3^h)$. We plan to explore ways for a more efficient embedding of a tree into a $Q^k_n$.

There has been much less work done in the area of Lee distance error-correcting codes, yet these codes seem well-suited for applications in a $Q^k_n$ or other $k$-ary structures. In particular, our preliminary investigation of the $t$-embedding resource placement problem in a generalized hypercube using a cyclic code has shown some interesting results. Recently, Roth and Siegel [15] have introduced BCH codes based on Lee distance which promise a higher information rate than cyclic codes. We will continue to investigate these and other types of Lee distance error-correcting codes and their application to the $t$-embedding resource placement problem.

The dimensional wormhole routing algorithm given in Section VI is minimal; however, it is subject to deadlock. We would like like to investigate other adaptive wormhole routing algorithms that are based on Lee distance. Future work will also include applying Lee distance to fault-tolerant routing algorithms.

Finally, many of the results obtained are valid also for a multi-dimensional torus. That is, a $Q^k_n$ having radix $k_i$ in dimension $i$ and $k_i$ may differ from $k_j$, $0 \leq i, j \leq n - 1$. Our future research will include extending results to multidimensional tori.
ACKNOWLEDGMENTS
This research was supported in part by the National Science Foundation and the Oregon Advanced Computing Institute under grant MIP-910852.

REFERENCES

Bella Bose (S'78-M'79-SM'94-F'95) received the BE degree in electrical engineering from Madras University, Madras, India, in 1973, the ME degree in electrical engineering from the Indian Institute of Science, Bangalore, in 1975, and the MS and PhD degrees in computer science from Southern Methodist University, Dallas, Texas, in 1979 and 1980, respectively.

Since 1980, he has been with the Department of Computer Science, Oregon State University, Corvallis, where he is a professor. He was a visiting associate professor at Stanford University from January to September 1987.

His current research interests include error control codes, fault-tolerant computing, parallel processing, and VLSI.

Dr. Bose is a member of the Association for Computing Machinery and the IEEE Computer Society.

Bob Broeg (S'83) is a professor of computer science and an associate professor of computer science at Oregon State University. He received the BA degree in mathematics, philosophy, and education from Dominican College, Racine, Wisconsin, in 1973, the MS degree in mathematics from Marquette University, Milwaukee, Wisconsin, in 1975, the MTS degree in theology from the Franciscan School of Theology, Berkeley, California, in 1979, and the MS degree in computer science from the California State University at Sacramento in 1989.

He is presently a PhD student in the Department of Computer Science at Oregon State University. He taught high school mathematics for 12 years and is currently a lecturer in computer science at Western Oregon State College in Monmouth, Oregon.

Brother Bob is a member of the Association for Computing Machinery and a student member of the IEEE Computer Society.

Younggeun Kwon received the BS degree in electrical engineering and the MS degree in electrical engineering from the ROK Air Force Academy, Seoul, Korea, in 1978, and the MS degree in computer engineering from the ROK Air Force Academy, Seoul, Korea, in 1986, and the PhD degree in computer science from the Oregon State University, Corvallis, Oregon, in 1993.

From 1978 to 1980 he was a computer software programmer in ROK Air Force. From 1988 to 1988, he was an assistant professor in the Department of Computer Science, ROK Air Force Academy. Col. Kwon is currently an assistant professor in the Department of Computer Science at the ROK Air Force Academy. His research interests include distributed and fault-tolerant computing, graph theory, and parallel algorithms.

Yaagoub Ashir received the BS degree in computer science from the University of Petroleum and Minerals, Saudi Arabia, in 1989, and the MS degree in computer science from Oregon State University, Corvallis, Oregon, in 1993, with an award of scholarship from the University of Bahrain, Bahrain. Currently, he is an assistant professor in the Department of Computer Science, University of Bahrain.

Ashir's research interests include parallel processing, interconnection networks, and graph theory with applications in computer science.