1-1-2001

Near-optimal broadcast in all-port wormhole-routed hypercubes using error-correcting codes

Hyosun Ko
University of Nevada, Las Vegas

Follow this and additional works at: <https://digitalscholarship.unlv.edu/rtds>

Repository Citation

Ko, Hyosun, "Near-optimal broadcast in all-port wormhole-routed hypercubes using error-correcting codes" (2001). *UNLV Retrospective Theses & Dissertations*. 1326.
<http://dx.doi.org/10.25669/g70s-7ji6>

This Thesis is protected by copyright and/or related rights. It has been brought to you by Digital Scholarship@UNLV with permission from the rights-holder(s). You are free to use this Thesis in any way that is permitted by the copyright and related rights legislation that applies to your use. For other uses you need to obtain permission from the rights-holder(s) directly, unless additional rights are indicated by a Creative Commons license in the record and/or on the work itself.

This Thesis has been accepted for inclusion in UNLV Retrospective Theses & Dissertations by an authorized administrator of Digital Scholarship@UNLV. For more information, please contact digitalscholarship@unlv.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

**ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600**

UMI[®]

**NEAR -OPTIMAL BROADCAST IN ALL-PORT WORMHOLE-ROUTED
HYPERCUBES USING ERROR-CORRECTING CODES**

By

Hyosun Ko

**Bachelor of Science
Hanyang University, Seoul, Korea
1990**

**Master of Science
University of Nevada, Las Vegas
2001**

**A thesis submitted in partial fulfillment
of the requirements for the**

**Master of Science Degree
Department of Electrical Engineering
Howard R. Hughes College of Engineering**

**Graduate College
University of Nevada, Las Vegas
December 2001**

UMI Number: 1409036

Copyright 2002 by
Ko, Hyosun

All rights reserved.

UMI[®]

UMI Microform 1409036

Copyright 2002 by ProQuest Information and Learning Company.
All rights reserved. This microform edition is protected against
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Thesis Approval

The Graduate College
University of Nevada, Las Vegas

August 9, 2001

The Thesis prepared by

Hyosun Ko

Entitled

Near-Optimal Broadcast in All-Port Wormhole-Routed Hypercubes Using
Error-Correcting Codes.

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative

ABSTRACT

Near –Optimal Broadcast in All-Port Wormhole-Routed Hypercubes Using Error-Correcting Codes

by

Hyosun Ko

Dr. Shahram Latifi, Examination Committee Chair
Professor of Electrical Engineering
University of Nevada, Las Vegas

A new broadcasting method is presented for hypercubes with wormhole routing mechanism. The communication model assumed allows an n -dimensional hypercube to have at most n concurrent I/O communication along its ports. It assumes a distance insensitivity of $(n + 1)$ with no intermediate reception capability for the nodes. The approach is based on determination of the set of nodes called stations in the hypercube. Once stations are identified, node disjoint paths are formed from the source to all stations. The broadcasting is accomplished first by sending the message to all stations, which will inform the rest of the nodes. To establish node-disjoint paths between the source node and all stations, we introduce a new routing strategy. We prove that multicasting can be done in one routing step as long as the number of destination nodes are at most n in an n -dimensional hypercube. The number of broadcasting steps using our routing is equal to or smaller than that obtained in an earlier work; this number is optimal for all hypercube dimensions $n \leq 12$, except for $n = 10$.

TABLE OF CONTENTS

ABSTRACT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	v
LIST OF TABLES	vi
CHAPTER 1 INTRODUCTION	1
CHAPTER 2 NOTATION AND BACKGROUND	5
2. 1 Interconnection network architectures	5
2. 2 Characteristics of Interconnection Network	8
2. 2. 1 Topology	8
2. 2. 2 Routing	9
2. 2. 3 Flow control	11
2. 2. 4 Switching	14
2. 3 Hypercubes	20
2. 4 Error-Correcting Codes	21
2. 4. 1 Error Correcting Codes	21
2. 4. 2 Perfect Codes	23
CHAPTER 3 MULTICASTING IN Q_n	25
3.1 One to Many Disjoint Paths in Q_n	25
3. 2 Wormhole Multicasting in Hypercube	36
CHAPTER 4 BROADCASTING IN HYPERCUBE USING ERROR-CORRECTING CODES	39
4.1 Hypercube Q_n , $n = 2^m - 1$, $m \geq 2$	41
4.2 Hypercube Q_n , $2^m - 1 < n < 2^{m+1} - 1$	51
4.3 Unified Algorithm for Q_n with Arbitrary n	58
4. 4. Deadlock	59
CHAPTER 5 CONCLUSIONS	61
BIBLIOGRAPHY	63
VITA	65

LIST OF FIGURES

Figure 1. A generic multiprocessor based on an interconnection network.	6
Figure 2. A generic node architecture	7
Figure 3. 3D view of the 3-cube	9
Figure 4. Packet B is blocked behind packet A while all physical channel remain idle	12
Figure 5. Virtual channels provide additional buffers allowing packet B to pass blocked packet A.	13
Figure 6. Wormhole routing	16
Figure 7. Comparison of different switching techniques	17
Figure 8. Communication latency in milliseconds versus distance for transmitting a 1-Kbyte message.	19
Figure 9. Stepwise Execution of the Algorithm in Q_5	30
Figure 10. Stepwise Execution of the Algorithm in Q_7	31
Figure 11. Pseudo-code of the Algorithm CSR to Compute n disjoint Paths in Q_n	32
Figure 12. Repeated partitioning process for (n, k) Hamming code words.	44
Figure 13. A general broadcasting algorithm for Q_n	55
Figure 14. Procedure Hamming (α, n, m)	56
Figure 15. Procedure <i>Compute_sf</i> (n, m, sf)	56
Figure 16. Merit of each bound for Q_n	62

LIST OF TABLES

Table 1. (7, 4) Hamming Code Words Sorted According to Weight	41
Table 2. CSR Table for the second Routing Step in Q_7	48
Table 3. Example: $n = 15, k = 11, m = 4$	48
Table 4. Example: $n = 31, k = 26, m = 5$	49
Table 5. Possible Routing Steps for Hamming Codes	50
Table 6. CSR Table for the First Routing Steps in Q_5	53
Table 7. Partitioning for Q_7	53
Table 8. Partitioning for Q_9	53
Table 9. CSR Table for the First Routing Step in Q_9	54
Table 10. CSR Table for the first Routing Step in Q_5	54
Table 11. CSR Table for the frist Routing Step in Q_{10}	54
Table 12. Lower and Upper Bounds of Routing Steps for Broadcasting	62

CHAPTER 1

INTRODUCTION

Advances in semiconductor and integrated circuit technology have drive to achieve faster and more efficient supercomputer. A recent trend in supercomputing has been towards the use of parallel processing to solve computationally-intensive problems. Massively parallel computers(MPC) with thousands of processors play an important role in solving computationally intensive problems. Such large-scale multiprocessors are characterized by distribution of memory among an ensemble of computing nodes, which each node has its own processor, local memory, and other supporting devices. The nodes are connected by direct, or point-to-pint network.

The processing of a concurrent computer exchange data and synchronize with one or others by passing messages over an interconnection network. The interconnection network is often the critical component of a large parallel computer because performance is very sensitive to network latency and efficiency because the network describes a large fraction of the cost and power dissipation of the machine. An interconnection network is characterized by four factors: topology, routing, flow control, and switching[14]. The topology of a network, usually modeled as a graph, defines how the node is interconnected by channels [15]. Routing specifies how a packet chooses a path in this graph. Flow control deals with the allocation of channels and buffers to a packet as it

traverses along a path through the network [5]. Switching is the actual mechanism that removes data from an input channel and places it on an output channel while the input and output selection policies determine how a packet uses channels as it travels an intermediate router [4][14].

In robust networks, it is desirable at times to send (receive) data/control information to (from) all the processors in minimal time. This type of communication commonly referred to as broadcasting (gathering) is often employed for synchronization, initialization, diagnosis, or algorithm execution. Broadcasting and gathering, as treated in this paper, are conceptually similar: Any algorithm for broadcasting can be translated to a gathering algorithm (or vice versa) by simply reversing the data paths. Thus the following treatment of broadcasting holds for gathering as well.

To speed up the broadcasting in a given network without modifying its topology, certain processors called stations can be specified to act as relay agents. Stations, being connected to the host, will each have one sphere of influence. Each sphere is essentially a set of nodes whose distance from the station assigned to that sphere is no more than some positive integer t . If the station assignment is such that the intersection of any two spheres is empty, it is called a non-overlapping assignment; otherwise the assignment is overlapping. Assuming a complete coverage of direct network nodes by the spheres surrounding the stations, broadcasting routing steps can be reduced from d to $t + c$, where d is the network diameter, and c is the number of steps to send the message to stations.

If the network is a hypercube, the stated problem can be shown to be equivalent to that of designing a t -error correcting code with parameters (n, k) where n is the length of

the code word and k is the number of symbols in the message. The code words in such a code can be regarded as the labels of the stations, i.e. 2^k stations in an n -dimensional hypercube.

Wormhole routing, whereby the message packet is split in small words called flits, and flits will flood the network from source to destination in a pipeline fashion, is a dominant switching technique in massively parallel computers. Several researchers have developed efficient broadcasting algorithms using wormhole routing. McKinley and Trefftz [12] gave a broadcast algorithm of $\lceil n/2 \rceil$ routing steps in an n -cube with wormhole, e-cube routing and all-port communication. Ho and Kao have developed more efficient broadcasting algorithm in an n -cube with wormhole, e-cube routing and all-port communication[9]. Recently Latifi has proposed a new broadcasting method using error-correcting codes[10]. In this thesis, we give an algorithm that provides some improvement in broadcasting time over Ho and Kao's method. Our algorithm uses error-correcting codes and a routing method called CSR(Circular Shift Routing) method, assuming that the network is an n -cube equipped with wormhole routing and all-port communication.

The organization of the thesis is as follows. The next chapter describes notion and background explanation for interconnection network architecture and characteristics to understand the broadcasting algorithm for the hypercube and error-correcting codes. Chapter 3 introduces the construction of the CSR table and states multicasting in the hypercube. In Chapter 4, we give theorems and a general broadcasting algorithm for n -

dimensional hypercube using error-correcting codes. We show that our algorithm is optimal for $n \leq 12$ in all cases but when $n = 10$. This thesis concludes with Chapter 5.

CHAPTER 2

NOTATION AND BACKGROUND

The purpose of this chapter is to provide the necessary foundation for this thesis. The notation and terminology, adopted throughout the remainder of this thesis, are introduced.

2. 1 Interconnection network architectures

Multicomputers are organized as ensembles of nodes, where each node is a programmable computer with its own processor, local memory, and other supporting devices[1]. The way the nodes are connected to one another varies among machines. In interconnection network architecture, each node has a point-to-point, or interconnection, connection to some number of other nodes, called neighboring nodes. Interconnection networks have become a popular architecture for constructing massively parallel computers because of their scalability; that is, as the number of nodes in the system increase, the total communication bandwidth, memory bandwidth, and processing capability of the system also increase. Figure 1 shows a generic multiprocessor with a set of nodes interconnected an interconnection network. Nodes must communicate by passing messages through the network because they do not physically share memory. Message size may vary, depending on the application. For efficient and fair use of network resources, a message is often divided into packets (or flits in wormhole routing

switching). A packet is the smallest unit of communication that contains routing and sequencing information; this information is carried in the packet header. Neighboring nodes may send packets to one another directly, while nodes that are not directly connected must rely on other nodes in the network to relay packets from source to destination. In many systems, each node contains a separate router to handle such communication-related tasks. Usually dedicated routers are used to allow overlapped computation and communication within each node.

Figure 1. A generic multiprocessor based on an interconnection network.

Figure 2 shows the architecture of a generic node. Each router supports some number of input and output channels. Normally, every input channel is paired with corresponding output channels. Internal channels connect the local processor/memory to the router. Although it is common to provide only one pair of internal channels, some systems use more internal channels to avoid a communication bottleneck between the local processor/memory and the router[3]. External channels are used for communication

between routers and, therefore, between nodes. By connecting the input channels of one node to the output channels of other nodes, the topology of the interconnection network is defined. Each node has a fixed number of input and output channels, and every input channel is paired with a corresponding output channel. A packet, which is smallest unit of communication that contains routing and sequencing information, sent between two nodes that are not neighboring must be forwarded by routers along multiple external channels. The sequential list of channels traversed by such a packet is called a path, and the number of channels in the path is called the path length.

Figure 2. A generic node architecture

2. 2 Characteristics of Interconnection Network

2. 2. 1 Topology

The topology of a network, usually modeled as a graph, defines how the nodes are interconnected by channels. If every node is connected directly to every other node, the network topology is fully connected, or complete. Although complete topologies obviate forwarding of packets by intermediate nodes, they are practical only for very small because the number of physical connections per node is limited by rigid constraints. These engineering and scaling difficulties preclude networks with large complete topologies. Therefore, many interconnection networks use a fixed, multiple-hop topology, such as hypercube or two-dimensional mesh. Most of the popular direct topologies fall in the general category of either n -dimensional meshes or k -ary n -cubes because their regular topologies simplify routing. For systems in which the network latency depends on the path length, the hypercube is a popular choice of topology because of its relatively small intermediate distance. The hypercube graph and its basic properties are intensively mentioned as follows[15] since hypercube topology is used for efficient broadcasting algorithm in this thesis and we will discuss more details on hypercube later in this section.

In what follows, the hypercube is regarded as a graph and we will often use the term vertices or nodes interchangeably for the processors they represent. A 3-cube can be represented as an ordinary cube in three dimensions where the vertices are the $8 = 2^3$

nodes of the 3-cube, see Figure 3. More Generally, one can construct an n -cube as follows. First, the 2^n nodes are labeled by the 2^n binary numbers from 0 to $2^n - 1$. Then a link between two nodes is drawn if and only if their binary numbers differ by one and only one bit.

Figure 3. 3D view of the 3-cube

An n -cube graph is an undirected graph consisting of $k = 2^n$ vertices labeled from 0 to $2^n - 1$ and such that there is an edge between any two vertices if and only if the binary representations of their labels differ by one and only one bit.

2. 2. 2 Routing

An interconnection network topology must allow every node to send packets to every other node. In the absence of a complete topology, routing determines the path selected by a packet to reach its destination. Efficient routing is critical to the performance of direct network.

Routing can be classified in several ways. In source routing, the source node selects the entire path before sending the packets. Each packet must carry this routing

information, increasing the packet size. Furthermore, the path cannot be changed after the packet has left the source. Most direct network systems use distributed routing. In this approach, each router, upon receiving the packet, decides whether it should be delivered to the local processor or forwarded to a neighboring router. In the latter case, the routing algorithm is invoked to determine which neighbor should be sent the packet. In a practical router design, the routing decision process must be as fast as possible to reduce the network latency. A good routing algorithm should also be easily implemented in hardware.

Routing can also be classified as deterministic or adaptive. In deterministic routing, the path is completely determined by the source and destination addresses. This method is also referred to as oblivious routing. One approach to designing a deadlock-free routing algorithm for a wormhole-routed network is to ensure that cycles are avoided in the channel dependence graph. This can be achieved by assigning each channel a unique number and allocating channels to packets in strictly ascending(or descending) order. If the behavior of the algorithm is independent of current network conditions, it is deterministic. A routing technique is adaptive if, for a given source and destination, the path taken by a particular packet depends on dynamic network conditions, such as the presence of faulty or congested channels. An adaptive routing algorithm for a wormhole-routed network, however, must address the deadlock-free issue. To do so often requires the use of additional channels; in particular, some adjacent nodes must be connected by multiple pairs of opposite unidirectional channels. These pairs of channels may share one or more physical channels. The concept of virtual channels will be discussed later.

A routing algorithm is said to be minimal if the path selected is one of the shortest paths between the source and destination pair. Using a minimal routing algorithm, every channel visited will bring the packet closer to the destination. One general adaptive routing technique works by partitioning the channels into disjoint subsets. Each subset constitutes a corresponding subnetwork. Packets are routed through different subnetworks, depending on the location of destination nodes. A nonminimal routing algorithm allows packets to follow a longer path, usually in response to current network conditions. If minimal routing is not required, deadlock-free adaptive routing can be provided using fewer additional channels. If r pairs of channels connect every pair of adjacent nodes, then the following nonminimal adaptive routing algorithm can be applied to k -ary n -cube and mesh topologies. Both of these algorithms allow the packets to take a longer path if there is no shortest path with all its channels available.

2. 2. 3 Flow control

A network consists of many channels and buffers. Flow control deals with the allocation of channels and buffers to a packet as it travels along a path through the network. A resource collision occurs when a packet cannot proceed because some resource that it requires is held by another packet. Whether the packet is dropped, blocked in place, buffered, or rerouted through another channel depends on the flow control policy. Typically, a single buffer is associated with each channel. Once a packet A is allocated a buffer b_i , no other packet B can use the associated channel c_i until A releases b_i . In networks that use flit-level flow control, packet A may be blocked due to contention elsewhere in the network while still holding b_i . In this case channel, c_i is idled even

though there may be other packets in the network, e.g., packet B, that can make productive use of the channel.

This situation is illustrated in Figure 4. In the Figure, a fragment of a network is depicted with a rounded box denoting a flit buffer. Shaded arrow denotes routes that are in progress. Packet A is blocked holding buffers 3E (eastside of node 3) and 4S. Packet B is unable to make progress even though all physical channels it requires, (1E to 2W) through (4E to 5W), are idle because packet A holds buffer 3E which is coupled to channel (3E to 4W).

This problem of idling channels due to resource coupling is unique to interconnection networks that perform flow control at the flit-level. Most modern multicomputer networks that use circuit switching or wormhole routing fall into this class. The problem does not occur in traditional packet-switched networks that perform flow control at the packet level since such networks never block a partially transmitted packet.

Figure 4. Packet B is blocked behind packet A while all physical channel remain idle

A virtual channel consists of a buffer that can hold one or more flits of a packet and associated state information. Several virtual channels may share the bandwidth of a single physical channel. Virtual channels decouple allocation of buffers from allocation of channels by providing multiple buffers for each channel in the network. If a blocked packet A holds a buffer b_{i0} associated with channel c_i , another buffer b_{i1} is available allowing other packets to pass A. Figure 5 illustrates the addition of virtual channels to the network of Figure 4. Packet A remains blocked holding buffers 3E.1 and 4S.1. In Figure 5, however, packet B is able to make progress because buffer 3E.2 is available allowing it access to channel (3E to 4 w).

Adding virtual channels to an interconnection network is analogous to adding lanes to a street network. A network without virtual channels is composed of one-lane streets. In such a network, a single blocked packet blocks all following packets. Adding virtual channels to the network adds lanes to the streets allowing blocked packet to be passed.

Figure 5. Virtual channels provide additional buffers allowing packet B to pass blocked packet A.

In addition to increasing throughput, virtual channels provide an additional degree of freedom in allocation resources to packets in the network. This flexibility permits the use of scheduling strategies, such as routing the oldest packet first, that reduce the variance of network latency.

The most costly resource in an interconnection network is physical channel (wire) bandwidth. The second most costly resource is buffer memory. Adding virtual channel flow control to a network makes more effective use of both of these resources by decoupling their allocation. The only expense is a small amount of additional control logic.

2. 2. 4 Switching

While the input and output selection policies determine how a packet uses channels as it traverses an intermediate router, switching is the actual mechanism that removes data from an input channel and places it on an output channel. Network latency highly depends on the switching technique used. Four switching techniques have been adopted in interconnection networks: store-and-forward, circuit switching, virtual cut-through, and wormhole routing. In store-and-forward switching, also called packet switching, when a packet reaches an intermediate node, then packet is then forwarded to a selected neighboring node when the channel is available and the neighboring node has an available packet buffer. In circuit switching, a physical circuit is constructed between the source and destination nodes. After the packet has been transmitted along the circuit to the destination, the circuit is torn down. In virtual cut-through, the packet header is examined upon arrival at an intermediate node. The packet is stored at the intermediate

node only if next required channel is busy; otherwise, it is forwarded immediately without buffering.

Although both virtual cut-through and circuit switching offer low network latencies that are relatively independent of path length, virtual cut-through required that blocked packets be buffered, and circuit switching makes it difficult to support sharing of channels among packets. Wormhole routing, proposed by Dally and Seitz, was designed to overcome these difficulties while offering similar network latency.

Wormhole routing also uses a cut-through approach to switching. A packet is divided into a number of flits (flow control digits) for transmission. A flit is the smallest unit of information that a queue or channel can accept or refuse. The size of a flit depends on system parameters, in particular the channel width. Normally, the bits constituting a flit are transmitted in parallel between two routers. The header flit (or flits) of a packet governs the route. As the header advances along the specified route, the remaining flits follow in a pipeline fashion, as shown in Figure 6. It is possible for the first flit of a message to arrive at the destination node before the last flit of the message has left the source. Because most flits contain no routing information, the flits in a message must remain in contiguous channels of the network and cannot be interleaved with the flits of other messages. If the header flit encounters a channel already in use, it is blocked until the channel becomes available.

The pipelined nature of wormhole routing produces two positive effects. First, the absence of network contention makes the network latency relatively insensitive to path length. Second, large packet buffers at each intermediate node are obviated; only a small

FIFO (first in, first out) flit buffer is required. In some wormhole-routed systems such as the Ncube-2 and Symult 2010, the flit buffer can hold only one flit. Other systems, such as the J-machine, a fine-grained system built at the Massachusetts Institute of Technology, have demonstrated improved network performance by using larger flit buffers. In the extreme, when the flit buffers are as large as the packet themselves, the behavior of wormhole routing resembles that of virtual cut-through. However, virtual cut-through differs from wormhole routing in that it buffers messages when they block, removing them from the network. The deadlock properties of cut-through routing are accordingly identical to those of store-and-forward routing. With wormhole routing, blocked messages remain in the network.

Figure 6. Wormhole routing

Figure 7 compares the communication latency of wormhole routing with that of store-and-forward switching and circuit switching in a contention-free network. In this

case, the behavior of virtual cut-through is the same as that of wormhole routing, so virtual cut-through is not shown explicitly. The channel propagation delay is typically small relative to L/B (where L is the length of the message and B is the channel bandwidth) and is ignored here. For example, in the Intel Touchstone Delta, the time to transmit one flit on a channel is 75 nanoseconds for packets traveling in the same direction and 150 nanoseconds for packets that change direction. Even for a channel a foot long, the propagation delay is only 1.5 nanoseconds. The delay in transmitting a ready signal from the destination to the source in circuit switching is also ignored.

Figure 7. Comparison of different switching techniques: (1) store-and-forward switching; (2) circuit switching; (3) wormhole routing

The Figure 7 shows the activities of each node over time when a packet is transmitted from a source node S to the destination node D through three intermediate nodes, I_1 , I_2 , and I_3 . The time required transferring the packet between the source processor and its router, and between the last router and the destination processor, is ignored. Unlike store-and-forward switching, both circuit switching and wormhole routing have communication latencies that are nearly independent of the distance between the source and destination nodes. This characteristic is confirmed by measurements on actual machines.

Figure 8 plots the communication latency versus path length for a 1-Kbyte packet transmitted using three switching technique: store-and-forward switching (on a 64-node Ncube-1 at Michigan State University), circuit switching (on a 32-node iPSC/2 at the University of Missouri-Rolla), and wormhole routing (on a 64-node Ncube-2 at Purdue University). The latencies of both circuit switching and wormhole routing demonstrate virtually no sensitive to distance. In these measurement, the traffic is generated such that there is no channel contention. Thus, the communication latency does not include the blocking time.

Under the wormhole routing model, the communication complexity for sending an m -byte message to a node which is d hops away can be modeled as $\tau + \delta(d - 1) + mt_c$ when there is no congestion. In this complexity estimate, τ is the initial start-up time (overhead), δ is the start-up time for each additional hop, and t_c is the data transmission time per byte. Typically, τ is much larger than δ , and the second term $\delta(d - 1)$ can generally be ignored, especially when m is large. For example, on an Intel iPSC/2

hypercube, $\tau = 0.7 \text{ msec}$, $t_c = 0.36 \text{ } \mu\text{sec/byte}$, $\delta = 60 \text{ } \mu\text{sec}$ as measured in [12]. As mentioned about the property of a circuit-switched routing in Section 2.2.4, although the iPSC/2 uses circuit-switched routing, the communication system modeling is the same as that of the wormhole routing when there is no congestion. Experimental results demonstrating the distance-insensitivity of wormhole routing can be found on a 64-node Ncube. The complexity estimate remains the same when intermediate nodes also intercept a copy of a by-passing message.

Figure 8. Communication latency in milliseconds versus distance for transmitting a 1-Kbyte message.

2.3 Hypercubes

Hypercubes are introduced under different names (n -cube, binary n -cube, Boolean n -cube, etc.). An n -cube parallel processor consists of 2^n identical processors, each provided with its own local memory, and interconnected with n neighbors[15]. A hypercube Q_n , of order n , is defined to be a symmetric graph $G = (V, E)$ where V is the set of 2^n vertices, each representing a distinct n -bit binary number and E is the set of symmetric edges such that two nodes are connected by an edge iff the Hamming distance between the two nodes is 1, i.e., the number of positions where the bits differ in the binary labels of the two nodes is 1. Links (edges) are also labeled such that link i , $0 \leq i \leq n-1$, connects two nodes which differ in the i th bit position, with the rightmost bit being in the 0^{th} position. For example, in Q_3 , the node 010 is connected to node 110 by link 2, to node 000 by link 1 and to node 011 by link 0. The Q_3 is symmetric with node degree of 3. Any subcube, Q_{n-m} , of a Q_n can be described by $(n-m)$ arbitrary bits and m fixed bits each denoted by "x". For example, a subcube Q_2 in a Q_5 consisting of four nodes {00100, 00110, 01100, 01110} is denoted as $0x1x0$. Similarly, $x^{n-1}0$ denotes the $(n-1)$ dimensional subcube of Q_n with all nodes with the last bit 0. The weight of a node v in Q_n is defined to be the number of 1s in the binary label of v and the Hamming distance of two nodes u and v is denoted by $H(u, v)$.

A path $\mathbf{P} = (S: d_0, d_1, \dots, d_{i-1})$ starts from the node S and traverses links labeled $d_0, d_1, \dots, d_{i-1}$ in that order. Where S is known from the context, the path is simply denoted as $(d_0, d_1, \dots, d_{i-1})$. The length of a path $\mathbf{P}$ is the number of links it contains and is denoted by $|\mathbf{P}|$. The link labels of a path may be arranged in any order to yield a different path with

the same end node. A cyclic shift of a path is an operator that rotates the link labels in a path to left (or right) by certain positions. As an example, $\mathbf{P} = (0000000: 0, 1, 4, 5)$ is a path in Q_7 with intermediate nodes - 0000001, 0000011, 0010011 and destination node 0110011. $|\mathbf{P}| = 4$ and $\mathbf{P}' = (4, 5, 0, 1)$ is a path obtained from $\mathbf{P}$ after shifting its link labels cyclically to left by two positions. Given $|\mathbf{P}| = m$, where $m \leq n$, $m-1$ different paths can be obtained by shifting $\mathbf{P}$ cyclically. It is well known that all these m paths are node-disjoint. If $\mathbf{P}$ contains $\mathbf{P}_1$'s all dimensions, then $\mathbf{P}_1$ is said to be a subset of $\mathbf{P}$.

2. 4 Error-Correcting Codes

Error-correcting codes can be used to speed up broadcasting in a communication network[10]. In this section, we highlight the characteristics of the error-correcting codes that are relevant for our purpose.

2. 4. 1 Error Correcting Codes

An error control code C consists of a set of M code words. Each code word is of the form $c = (c_0, c_1, c_2, \dots, c_{n-1})$: if the individual coordinates take on values from the Galois field $GF(q)$, then the code C is said to be q -ary $M \leq q^n$. The encoding process consists of breaking up the data stream into blocks and mapping these blocks onto code words in C . This mapping is usually one to one, ensuring that the encoding process can be reversed at the receiver and the original data block can be recovered. If the symbols in the data stream can take on any value in $GF(q)$, then the collection of all possible k -tuples

$$m = (m_0, m_1, m_2, \dots, m_{M-1})$$

forms a vector space over $GF(q)$. Clearly, there are q^k possible k -symbol data vectors.

The weight of a code word is the number of nonzero coordinates in the code word. The weight of a code word c is commonly written as $w(c)$. The Hamming Distance between two codes v and w is the number of coordinates in which two codes differ. The Minimum Distance of a code C , $d_{min}(C)$, is the minimum Hamming distance between all distinct pairs of codes in C .

It is known that a code with minimum distance d_{min} can correct all error patterns of weight less than or equal to $\lfloor (d_{min} - 1)/2 \rfloor$. The geometric interpretation of error correction offers some interesting insights in using the error-correcting codes for broadcasting in hypercubes.

The weight distribution of a Hamming code of length $n = 2^m - 1$ is known[11]. The number of code words of weight i , A_i in a Hamming code with length n , is simply the coefficient of x^i in the expansion of the following polynomial (referred to as the weight enumerator):

$$A(x) = \frac{1}{n+1} \{ (1+x)^n + n(1-x)(1-x^2)^{(n-1)/2} \}$$

Definition 1. A **Hamming Sphere** of radius t , $HS_t(c)$, of a code word c is defined to be the set of all vectors that are at a Hamming distance $\leq t$ from the code word c .

Example 1. Let $c = 1101$ be a code word, $n = 3$ and $q = 2$.

Then $HS_1(c) = \{(1101), (0101), (1001), (1111), (1100)\}$.

$HS_2(c) = HS_1(c) \cup \{(0001), (0111), (0100), (1011), (1000), (1110)\}$.

2. 4. 2 Perfect Codes

A binary code C of length n is perfect if every node $x \in$ subset of C is within distance t from exactly one code word of C [6]. In general a perfect t -error correcting code with q^k code words must satisfy the following relation[11]

$$\sum_{j=0}^t \binom{n}{j} (q-1)^j = q^{n-k}$$

Linear binary perfect codes have only four classes: the trivial case, which has no redundancy ($t=0$); Repetition Codes ($t = (n-1)/2$); Hamming codes($t = 1$); and Golay code($t = 3$) [14]. For the Hamming code, the algorithm is very simple because $t = 1$.

Remark 1. Note that $|HS_t(c)| = \sum_{j=0}^t \binom{n}{j} (q-1)^j$ for all code words $c \in C$

and for any two arbitrary $c, c' \in C$, we have $HS_t(c) \cap HS_t(c') = \emptyset$ for any code C (with M code words in $GF(q)$, which is t -error correcting. Note that $q^n \geq M \times |HS_t(c)|$. The code C is called **perfect** if $q^n = M \times |HS_t(c)|$.

Remark 2. The number of code words in a perfect q -ary code must be of the form $M = q^k$ for some positive integer k [6]; also a perfect t -error correcting code with q^k code words must have parameters $\{q, n, k, t\}$ that satisfy the relation

$$\sum_{j=0}^t \binom{n}{j} (q-1)^j = q^{n-k}.$$

For our purpose, to use the error-correcting codes to facilitate broadcasting in hypercubes using wormhole routing, we assume that $q = 2$ and $t = 1$. So, we specify a

code C by a 2-tuple $\{n, k\}$. Note that the n -dimensional vector space over $GF(2)$ is the hypercube Q_n of dimension n where each node of the hypercube is a vector in the space

CHAPTER 3

MULTICASTING IN Q_n

Multicasting involves sending a message from a source to several destinations. In this section, we show that in the Q_n equipped with wormhole routing and all-port communication, multicasting can be performed in one routing step as long as the number of destination nodes is at most n .

3.1 One to Many Disjoint Paths in Q_n

Given a hypercube Q_n , a source node $s = (00\dots 0)$ and a set of l distinct destination nodes $D = \{d_1, d_2, \dots, d_l\}$, $1 \leq l \leq n$, the objective in this section is to design an algorithm to compute node disjoint paths from the source node to the destination nodes.

Remarks 3. Q_n is node symmetric [15], and hence we can consider the identity node (the all zero node $(00\dots 0)$) to be the source node for our purpose without any loss of generality. We also note that Q_n (denoted also by x^n) consists of two subcubes $x^{n-1}1$ and $x^{n-1}0$. A node u in Q_n is denoted as $u = (u_{n-1}u_{n-2}\dots u_0)$, where u_i denotes the i th bit of the node label, $0 \leq i \leq n-1$ (the dimensions are numbered 0 through $n-1$, from least significant bit (LSB) position to most significant bit (MSB) position).

Definition 2. A path $P(u, v)$ in Q_n between two nodes u and v is denoted by a sequence of link labels. For example, in Q_4 is denoted by $P(0001, 1010) = (0, 1, 3)$. Note that $P(1010, 0001) = (0, 1, 3)$ denotes a different path e.g., $1010 \rightarrow 1011 \rightarrow 1001 \rightarrow 0001$, between the same two nodes. Since the sequence of link labels is ordered, we apply the sequence $P(u, v)$ starting at node u and reaching node v . The length of a path is defined to be the number of links in the path.

Definition 3. Given two nodes u and v in Q_n , the first Hamming distance path $FHP(u, v)$ is the minimal (shortest) path obtained by flipping the nonmatching bits in ascending order of dimensions. For example in Q_4 ,

$$FHP(0001, 1010) = (0, 1, 3) = 0001 \rightarrow 0000 \rightarrow 0010 \rightarrow 1010.$$

Definition 4. Consider an arbitrary set of $(n - 1)$ distinct nodes $V = \{v^1, v^2, \dots, v^{n-1}\}$ in Q_n . $N_v(v^i)$, $1 \leq i \leq n - 1$, is defined to be an adjacent node $v' = v'_{n-1}v'_{n-2} \dots v'_1v'_0$ of v^i such that $v'_{n-1} \dots v'_0 \notin V$ and v' is obtained as follows:

- 1) if possible, v' is obtained from v^i by inverting the first “1” (in v^i) in descending dimension order, or
- 2) if v' cannot be obtained by 1), then v' is obtained from v^i by inverting the first “0” (in v^i) in descending dimension order. $\eta_v(v^i)$ is defined to be the bit position of v^i that is reversed to get v' .

Example 2. Consider $V = \{0101, 0111, 0110\}$ in Q_4 ; we have $N_v(0111) = 0011$ and $\eta_v(0111) = 2$. As another example, consider $V = \{0101, 0001,$

0100} in Q_4 ; we get $N_v(0101) = 1101$ and $\eta_v(0101) = 3$. Note that in the second example, $N_v(0101)$ cannot be obtained by reversing any of the “1” bits of the node 0101.

Lemma 1. Given any set V of $(n - 1)$ distinct nodes in Q_n , $N_v(v)$, $v \in V$, always exists.

Proof: Consider an arbitrary node $v \in V$. Since $v \in Q_n$, node v has n distinct neighbors in Q_n and $|V| = n - 1$, $N_v(v)$ always exists. $\square$

Our strategy in designing the algorithm is to use a recursive approach – we trace a path to one of the subcubes and then map the rest of the destination nodes onto the other subcube and then repeat the process.

- Consider two subcubes $x^{n-1}0$ and $x^{n-1}1$ of Q_n . Of n destination nodes, consider the set S_1 of those nodes that are in $x^{n-1}1$ and choose one, say $v \in S_1$ (resolve ties arbitrarily), closest to the source node s and trace a shortest (*FHP*) path from node s to node v . Note that since the *FHP* is computed in ascending order of the dimensions, all points in this path are in $x^{n-1}1$.
- We delete node v from the set D (also from S_1) and map the rest of the nodes in S_1 to the subcube $x^{n-1}0$ (note that the source node s is in $x^{n-1}0$).
 - If for any node $y = y_{n-1} \dots y_1 y_0$ in $S_1 - \{v\}$, there does not exist any node $w = w_{n-1} \dots w_1 w_0$ in $D - \{v\}$ such that

$$y_{n-1} \dots y_1 0 = w = w_{n-1} \dots w_1 0, \quad \text{then use } y_{n-1} \dots y_1 0 \text{ as the new destination node in } x^{n-1}0 \text{ (corresponding to the node } y);$$

- otherwise, use the node $w_{n-1} \dots w_1 0$ as the new destination node in $x^{n-1} 0$ (corresponding to the node y), where $w = N_{D-\{v\}}(y)$.
- We need to map the nodes in $D - S_1$. If $S_1 = \emptyset$, then we choose one, say $v \in D$, farthest to the source node, and go from node v to v' by traversing link 0 and then trace a shortest (FHP) path from node s to the node v' (all points in the path are in $x^{n-1} 1$). For each remaining node $y = y_{n-1} \dots y_1 y_0$ in $D - S_1$, use $y_{n-1} \dots y_1 0$ as the new destination node in $x^{n-1} 0$ (corresponding to the node y).
- We now have a set of $(n - 1)$ destination nodes and a source node in $x^{n-1} 0$ (which is a $n-1$ dimensional hypercube); invoke the algorithm recursively.

Example 3. Consider the source node $s = (00000)$ and the set of destination nodes

$$D = \{01100, 11100, 01010, 00010, 01110\}$$

in Q_5 . Note that at this stage of the algorithm the set S_1 is empty. The nodes 01110 and 11100 both have the maximum weight of 3. We choose 01110 arbitrarily, and compute the path $P(00000, 01110) = \{0, 1, 2, 3, 0\}$. At the second stage of recursion we have a source node (0000) and a set of destination node $\{0110, 1110, 0101, 0001\}$. At this stage, S_1 consists of two nodes $\{0101, 0001\}$ of which the node 0001 is closer to the source and we compute the path $P(0000, 0001) = \{1\}$. For the other node 0101, we go to its image 01000 in $x^3 00$ and take the images of the other nodes, as

well. So, at the third stage of recursion, we have a source node 000 and a set of destination nodes $\{011, 111, 010\}$. $S_1 = \{011, 111\}$ and the closest node in S_1 is 011 and we compute the path $P(000, 011) = \{2, 3\}$. For the other node 111 in the set S_1 , we simply take its image in $x^2 000$, which is 11000 (by traversing the link "2" from the node 111(00)) and then compute the FHP from the node 00 to the node 11 in $x^2 000$ by traversing the links 4 and 3; see Figure 3.1 for details; note that the paths in the table are given as link sequences from the source node to the destination nodes.

Example 4. Consider the source node $s = (0000000)$ and the set of destination nodes $D = \{0001100, 0101001, 0111011, 1010111, 1100010, 1110000, 1110010\}$. At this stage, S_1 consists of three nodes $\{0101001, 0111011, 1010111\}$ of which 0101001 is closest to the source node, and hence we trace the path $P(s, 0101001) = (0, 3, 5)$. At the second stage of recursion, we have six destination nodes $\{000110, 011101, 101011, 110001, 111000, 111001\}$ and a source node $s = (000000)$ in Q_6 . S_1 consists of four nodes $\{011101, 101011, 110001, 111001\}$ of which 110001 is closest to the source node, and hence we trace the path $P(000000, 110001) = (1, 5, 6)$ – note that this also the path $P(0000000, 1100010)$ in the original problem. The rest of the computation is similar. See Figure 10 for the details: note that the paths in the table are given as link sequences from the source node to the destination nodes.

Destination Nodes	Path from Source to Destination
01100	23
11100	432
01010	31
00010	1
01110	03210

Figure 9. Stepwise Execution of the Algorithm in Q_5


```

Function Node_Disjoint_paths ( $k, D^k, P[1...k], Q[1,...k-1]$ )
  if  $k = 1$  then exit;
   $j = k; D^{k-1} = \phi$ ;
  Compute  $S_1 = \{y \mid y \in D^k \wedge y \in x^{k-1} 1\}$ ;
  if  $S_1 \neq \phi$  then
    begin
      Select  $v$  such that  $w(v) = \min_{u \in S_1} \{w(u)\}$ ;
      Compute the path  $FHP(s, v); P[j] = FHP(s, v) \sqcup Q[j]$ ;
      For each  $y \in \{S_1 - v\}$  do
         $j = j - 1$ ;
        if  $y_{n-1}y_{n-2}...y_1 0 \notin \{D^k - v\}$ 
          then
            begin
               $D^{k-1} = D^{k-1} \cup y_{n-1}y_{n-2}...y_{n-k+1}$ ;
               $P[j] = \{n - k\} \sqcup Q[j]$ ;
            end
          else
            begin
              Compute the node  $u = N_{(D^k - v)}(y); x = \eta_{(D^k - v)}(y)$ ;
               $D^{k-1} = D^{k-1} \cup u_{n-1}u_{n-2}...u_{n-k+1}$ ;
               $P[j] = \{n - k, x\} \sqcup Q[j]$ ;
            end
          end
        end
      if  $|D^k - S_1| \neq 0$  then
        begin
          if  $j = k$  then /*  $S_1 \neq \phi$  */
            begin
              Select  $v$  such that  $w(v) = \max_{u \in D^k} \{w(u)\}; D^k = D^k \cap \{v\}$ ;
              Compute the path  $\rho = FHP(s, v_{n-1}v_{n-2}...v_{n-k+1}1); P[j] = \{k\} \sqcup \rho \sqcup Q[j]$ 
            end;
          For each  $y = y_{n-1}y_{n-2}...y_{n-k} \in \{D^k - S_1\}$  do
             $D^{k-1} = D^{k-1} \cup y_{n-1}y_{n-2}...y_{n-k+1}; j = j - 1; P[j] = \{k\} \sqcup Q[j]$ 
          end
        end
      Node_Disjoint_paths ( $k - 1, D^{k-1}, P[1...k-1], P[1...k-2]$ );

```

Figure 11. Pseudo-code of the Algorithm CSR to Compute n disjoint Paths in Q_n

We now can present the pseudocode of the algorithm that generates the disjoint paths in Q_n given a source node $s = 00\dots 0$ and n destination nodes. $P[]$ and $Q[]$ are two array variables of ordered sets (sequences) to store the generated link label sequences for the node disjoint paths. D^k variables are set variables to hold the destination nodes at each stage of recursion. We use one primitive function $\text{II } x \text{II } y$, where both x and y are ordered set variables, returns a new sequence obtained by appending the sequence y to the sequence x . See Figure 3.3 for the complete pseudocode.

Lemma 2. The n paths, generated by the algorithm are mutually node-disjoint.

Proof: The node-disjoint property follows from the fact that at each stage of recursion, we are computing the path from one destination node to the source node (at any stage the hypercube has two subcubes: the 0-subcube and the 1-subcube, for instance; the path is traced only in the 1-subcube) and mapping the other destination nodes to distinct nodes in the 0-subcube (intermediate nodes on the path are always chosen from the 1-subcube).

Thus, the paths computed at any stage of recursion cannot intersect with each other. Also, note that the nearest neighbor node always exists, since there is always one fewer node to have conflict with than the degree of a node. Note that in Step 2, only the closest node to the source is considered, so that the path will not go through any other destination node at that stage. Similarly, for the path computed for a node in $D - S_1$, the path

consists of nodes in the 1-subcube while all destination nodes are in the 0-subcube.

Remarks 4. The length of a path, from a destination node to the source node, generated by the CSR algorithm, can be greater than its distance to the source node iff we pay a penalty by flipping a "0" to a "1" in generating the *FHP* at some stage(s) of the algorithm; each time we pay a penalty, the path length increases by 2.

Assume that for a given set D of n destination nodes, our CSR algorithm generates a path of length $\geq n + 2$ in Q_n . Let the specific destination node be v (in D) such that $w(v) = m$.

Lemma 3. $2 \leq m \leq n - 2$

Proof: m cannot be 1, since for the path to node v to be of length $\geq n + 2$, the node must enter the set S_1 at some stage of recursion to pay the penalty.

$m = n - 1$ is not possible, since in that case there is only one zero in the bit string of node v , there is only one penalty possible, and hence the length of the path can be at most $n - 1 + 2 = n + 1$.

Assume that for the path we pay x number of penalties (i.e., we flip an original "0" to "1" x times) – each penalty adds 2 to the path length – thus, we must have

$$m + 2x > n + 1$$

Now, let us see what happens to the node v when we execute the algorithm; assume $v = 000\dots 11\dots 1$ without any loss of generality with all the m ones at the end. For us to pay the penalty, there must be at least " m " destinations which will coincide with the label of the troublesome node after changing one of its " m " 1's to 0, as well, as there is one

node that is already mapped to the source node; so, for the first penalty for node v , there are at least $m + 2$ nodes (including v). Consider the scenario of next penalty – the node v (in its reduced version in a subcube of the original cube) has still m 1's (penalty means one 0 was flipped to 1) – for us to penalty, there again must be m other offending nodes – only one of them can be from the offending nodes of the previous stage due to the special 1 in the node v because of previous penalty. Repeat the argument to get

$$m + 2 + (x - 1)(m - 1) \leq n$$

or

$$(m - 1)x \leq n - 3$$

Combining the two inequalities, we see that in order for a path to node v to have a length greater than $n + 1$, we must have $m = 2$. If $m = 2$, then the minimum value of n is 7 in order to satisfy the inequalities, and the corresponding x is 4. This is impossible in the execution of the algorithm CSR, since m must be bigger than x (for node v to pay penalty at any stage, v must be in S_1 for that stage, i.e., the rightmost bit of v must be "1"; also note that a "0" is flipped to "1" (as a penalty) in a left to right fashion and the algorithm proceeds on a right to left fashion); similarly, larger values of n and x with $m = 2$ are also ruled out, since m must be bigger than x . Thus, we can state the following theorem.

Theorem 1. The maximum length of any path traced by the algorithm CSR in a hypercube Q_n is $n + 1$, $n > 2$.

.

3. 2 Wormhole Multicasting in Hypercube

The CSR method, which constructs node-disjoint paths between the source node and at most n destination nodes, achieves multicasting in one routing step. In order to describe the wormhole multicast in Q_n and then, subsequently, to design a broadcast algorithm and to evaluate its performance, we need to specify the communication model.

Consider that a source node in the network wants to send an m -flit message to another node y which is Δ hops away in the network (Δ is the number of links to be traversed by the message as determined by the routing algorithm chosen). If wormhole routing is chosen, the total routing time or the so-called communication complexity will be given by $s + s'(\Delta - 1) + m\tau$, where s denotes the startup latency at the source node, s' denotes the startup latency at each of the intermediate nodes before the message reach its destination and τ denotes the transmission time per flit[14]. This is under the assumption that there is no link congestion, i.e., no two messages contend for the same communication channel at the same time. (Note that when there is link congestion, modeling communication complexity for wormhole routing becomes the same as in circuit switched routing). The efficiency of the wormhole routing derives from the fact that $s \gg s'$, and that when m is large and Δ is small, the second term $s'(\Delta - 1)$ becomes negligibly small – in other words, if the message traverses a relatively small distance, the communication delay in wormhole routing becomes insensitive to the distance [mac94]. Experimental results are available [8] to verify this in case of such wormhole routing in hypercube networks; for an Intel iPSC/2 hypercube, it was measured that $s = 0.7ms$, $s' = 60\mu s$, and $\tau = 0.36\mu s$. There are several factors that need to be carefully considered to

decide on the assumptions in the communication model, and thereafter to evaluate the performance of the communication primitives.

The communication latency in wormhole broadcast depends on the capability of the nodes to send/receive messages. In the single-port model, each node can send or receive a message along only one port while in all-port model a node can send or receive message along one port while receiving messages along other ports.

The distance insensitivity of point to point wormhole routing in a direct network has an immediate consequence: any source node can send a message to another destination node (say, Δ hops away) in one routing step (in the process, it can also inform all the nodes on the path if intermediate reception capability is assumed at the nodes). This statement need be qualified. The distance insensitivity depends on two factors: the message size being relatively large and the distance between source and destination being relatively small. So, there must be a limit on the length of the path between the source and destination beyond which the communication latency would not be distance insensitive. Without such a limit, one can trivially grow a Hamiltonian path from the source and inform all nodes in the network in one routing step.

In their study on wormhole broadcast in hypercubes, Ho et. al [9] have chosen n to be the limit for a hypercube with $N = 2^n$ nodes. This seems very logical, especially when a single message need be broadcast to all the nodes in the network, we will follow the same guidelines and use $(n + 1)$ to be the limit on the distance between the source and destination in a hypercube Q_n with 2^n nodes.

Theorem 2. Given a source node and n distinct nodes in the Q_n , it is always possible to establish n node-disjoint paths between the source and n destinations using CSR algorithm of the previous section.

Corollary 1. Multicasting can be done in Q_n in one routing step if the number of destination nodes is at most n .

CHAPTER 4

BROADCASTING IN HYPERCUBE USING ERROR-CORRECTIN CODES

In order to broadcasting a message from a given source node to all the nodes in a given hypercube Q_n of dimension n , we use two stages: 1) the source node does a multicasting of the message to a set of special nodes, called station nodes; the M code words of a t -error correcting code C can be the labels of the station nodes in the Q_n ; 2) each station node then concurrently sends the message to all the nodes in its own Hamming Sphere using a simple algorithm. Note that in this scheme the broadcasting takes $(t + c)$ steps, where c is the number of steps required to get the message to the station nodes. In order for each node in the hypercube Q_n to receive the message, it is required that for every n -tuple in the q^n space, there exists at least one code word with a Hamming distance of at most t . This is possible only when $2^n = M \times |HS_t(c)|$, i.e., when the t -error correcting code is perfect. Thus, we have the following theorem.

Theorem 3. A t error-correcting q -ary code of length n can be used as the labels of stations in a Q_n to broadcast a message in $t + c$ steps if and only if it is perfect.

Among the perfect binary codes [11], [2], the Hamming code is the best candidate because it is easy to implement. The only other example of a perfect binary code is binary Golay code $G_{23}[11]$.

Definition 1. The binary Hamming codes (n, k) are a family of single error-correcting codes and have the following parameters: $n = 2^m - 1$, $k = 2^m - m - 1$, and $t = 1$, where n is the code length and m is the number of information symbols (bits) in the code vectors (and m is the number of parity symbols in each code vector).

The weight distribution of a Hamming code of length $n = 2^m - 1$ is known [11]. The number of code words of weight i , A_i , is simply the coefficient of x^i in the expansion of the following polynomial (referred to as the weight enumerator):

$$A(x) = \frac{1}{n+1} \{ (1+x)^n + n(1-x)(1-x^2)^{(n-1)/2} \}.$$

Consider an example when $m = 3$. Then $n = 2^3 - 1 = 7$ and the weight enumerator for the $(7, 4)$ Hamming code is: $A(x) = 1 + 7x^3 + 7x^4 + x^7$. Thus, there are $2^k = 2^4 = 16$ code vectors (that can potentially be used as station nodes in a hypercube Q_7) one of these code vectors has a weight of 3, and seven have weight of 4.

Remarks 5. The binary Hamming codes (n, k) are single-error correcting ($t = 1$); hence, if the code vector are used as station nodes in the hypercube, then one the station nodes are informed of the broadcast message, it takes only one more time step to inform all other nodes in the network.

Remarks 6. Note that binary Hamming codes can be defined only when $n = 2^m - 1$ for some integer m ; thus, binary Hamming codes cannot be defined for all possible hypercube, Q_n , $n \geq 1$.

Table 1. (7, 4) Hamming Code Words Sorted According to Weight

0000000	0011101
0001110	0101011
0010011	0110110
0100101	1000111
0111000	1011010
1001001	1101100
1010100	1110001
1100010	1111111

We consider the special case of hypercubes Q_n for which binary Hamming codes can be defined first separately, and then we generalize the scheme to consider all possible hypercubes.

4.1 Hypercube Q_n , $n = 2^m - 1$, $m \geq 2$

When an (n, k) Hamming code is used to broadcast a message (2^k code vectors are used as station nodes in Q_n), the number of routing steps would be $(1 + c)$, where c is the number of routing steps necessary to inform all the 2^k station nodes from a given source node. The parameter c determines the performance of the broadcast algorithm and it depends on the number 2^k (the number of the station nodes) with respect to the dimension of the hypercube n .

Remarks 7.

1. If $n \geq 2$, the source node can send the message to all the station nodes in one step ($c = 1$); the node-disjoint paths for all-port wormhole multicasting can be set up by the CSR algorithm, as discussed in the previous section.

2. If $n < 2^k$, the code words (station nodes) must be partitioned into several subgroups with a representative within each subgroup. First, all the representatives will be informed and they, in turn, inform all the nodes within their own subgroup. Clearly, in this case, informing all the code words will require more than one step. If $2^k \gg n$, the code words must be partitioned several times, increasing the number of steps required to inform all code words. We use N_{sg} = number of subgroups and N_{sc} = number of code words in each subgroup during a given partition.

In order to design an algorithm to partition the station nodes, we need to make sure that the successive grouping of the station nodes be done over mutually disjoint subcubes (no node should get the message more than once). We first note an interesting property of the Hamming codes.

Remarks 8. If we arrange the code words of an (n, k) code in ascending order, we will note that the first half of the code words have “0” as their MSB, while the second half have “1” as their MSB. Similarly, in the column for any other position, there are the same number of “1”s as “0”s (Table 12 shows the (7, 4) Hamming code words. Observe that each consecutive 2^j code words in the list form a subgroup, $1 \leq j \leq 4$).

We use this property to partition the code words into subgroups. More specifically, codes can be divided into two subgroups, each matching $0x^{n-1}$ and $1x^{n-1}$, respectively (superscripts are repetition factors). In general, by fixing the first j MSB’s of the code words to one of possible j -tuples, one can obtain subgroups, each containing 2^{k-j} code

words where $1 \leq j < k$. The 2^j subgroups obtained as such are clearly disjoint. It follows that $N_{sg} = 2^j$ and $N_{sc} = 2^{k-j}$. Once the first j MSB's are fixed, they cannot be used (or fixed) in the next partitioning process if further partitioning is necessary. This is because we want to obtain disjoint subcubes at every step so that paths established within sub subcubes do not step over each other. Therefore, for the next round of partitioning, there are only $(n - j)$ bits available. If, in the next round, each subgroup is further divided into $2^{j'}$ subgroups by fixing j' bits, the number of usable dimensions after this division will reduce to $(n - j - j')$. We use N_{ud} to denote the number of remaining dimensions (bits) which have not been used (fixed) at the end of a specific partitioning process.

Broadcast scheme of the Hamming code lies in Theorem 4.1, i.e., we can send a message to n nodes in one routing step in Q_n . To send a message to all code words, we partition the code words into subgroups repeatedly until the number of code words in a subgroup is smaller than the number of code words in a subgroup is smaller than the number of usable dimensions N_{du} . Initially, $N_{ud} = n$. If j bits are used during the first stage of partitioning, $N_{ud} = n - j$ and $N_{sg} = 2^j$ after the first stage. This process is referred to as j -reduction, since the number of usable dimensions has been effectively reduced by j . We do not use these j bits in the second stage of partitioning to guarantee node-disjoint paths in our concurrent wormhole routing algorithm.

In the first time partitioning, we can inform as many as $(n + 1)$ representatives (including the source), which is equal to 2^m if we use the Hamming code (n, k) . Each of these representatives will become the source node within a subcube whose m MSBs are fixed and distinct. Therefore, after the first division, each representative has to inform 2^{k-m}

code words within its own subcube and the number of unfixed bits in the subcube label is reduced to $(n - m)$. In the second division, only $(m - 1)$ bits are fixed since $\lfloor \log_2(n - m) + 1 \rfloor = m - 1$. The process is repeated every time by fixing $(m - 1)$ bits of the subcube label from left to right until the argument for the log function is not large enough to accommodate $(m - 1)$ bit reduction, at which point $(m - 2)$ bits will be fixed. The division process will continue until $N_{ud} \geq N_{sc} - 1$.

Figure 12 shows the successive partitioning stages for Hamming codes. C is the number of routing steps necessary to inform all the code words and $(c - 1)$ is the number of partitions. Forwarding the message takes place after each partition. Note that before the first partition, we have only one group of code words of size 2^k , i.e., $N_{ud} = n$, $\log_2 N_{sg} = 0$, and $\log_2 N_{sc} = k$. In the first partition, the code words are divided into 2^m subgroups each of size $2^{(k-m)}$. This will reduce N_{ud} to $(n - m)$. In the next partition, only $(m - 1)$ bit reduction is possible, and so on. Note that in the last row in Figure 4.1, no partitioning is needed, and therefore there are dashes in entries for $\log_2 N_{sg}$ and $\log_2 N_{sc}$.

Step	N_{ud}	$\log_2 N_{sg}$	$\log_2 N_{sc}$
0	n	0	K
1	$(n - m)$	m	$(k - m)$
2	$(n - m) - (m - 1)$	$(m - 1)$	$(k - m) - (m - 1)$
...	...	...	...
$(c - 2)$	$(n - p + x)$	$(x + 1)$	$(x + t)$, where $n - p + x \geq 2^x - 1$
$(c - 1)$	$(n - p)$	x	t , where $n - p \geq 2^t - 1$
c	-	-	-

Figure 12. Repeated partitioning process for (n, k) Hamming code words.

During each stage of partitioning, we have, in general, several choices to choose the number of bits to fix. Often, several choices for N_{sg} yield the same overall number of routing steps for this case (i.e., $n = 2^m - 1$). We refer to these choices as acceptable values for N_{sg} . Consider the case of (31, 26) Hamming code for Q_{31} . In the first partitioning stage, possible values of N_{sg} are 2, 4, 8, 16 and 32 (depending on whether we fix 1, 2, 3, 4, or 5 bits) and the corresponding values of N_{sc} are 2^{25} , 2^{24} , 2^{23} , 2^{22} , or 2^{21} . Of all possible values of N_{sg} , 2, 4, or 8 are unacceptable because they do not utilize the full power of broadcasting to $n = 31$ nodes in one routing step; the values 16 and 32 are acceptable since they eventually yield the smallest number of routing steps for broadcasting. In all successive stages of partitioning after the first one, we choose the largest possible N_{sg} for that stage, and the partitioning process continues until $N_{ud} \geq N_{sc} - 1$; at this point, each node can inform all the nodes assigned to it within the subcube specified by unused dimensions (i.e., $Q_{N_{ud}}$) in one step (the reason for subtracting one is that one node within the subcube already has the message). Note that one routing step is defined as traversing a path (or paths) between a source and destination(s) according to the communication model assumed here.

Remarks 9. Even for the case when $n \neq 2^m - 1$, it will be shown that by always choosing the smallest of acceptable values for N_{sg} in the first time partitioning, one can minimize the number of routing steps. Let sf be the smallest of all acceptable values of $\log_2 N_{sg}$ in the first stage of partitioning. Note that for a given n , sf can be computed through an iterative process: compute $m = \lfloor \log_2(n + 1) \rfloor$ and set $sf = m$ and compute the number of

divisions needed to be δ (Figure 12); Then sf is decremented by 1 and δ is recomputed; this process is repeated as long as δ remains the same; sf is the smallest number for which δ is the same as that for m .

Example 5. Consider Q_3 and the Hamming code (3, 1), i.e., $n = 3$, $m = 2$, and $k = 1$. There are two (2^k) station nodes: 000 and 111. Since $n \geq 2^k$, the source node can send the message to all the station nodes in one routing step (the source node 000 sends the message to the station node using the path (012), according to CSR algorithm) and each station node concurrently sends the message to all its neighbors in the next routing step. Thus, the number of routing steps to broadcast a message is only 2.

Example 6. Consider Q_7 and Hamming code (7, 4), i.e., $n = 7$, $m = 3$, and $k = 4$. Since the number of code words or the station nodes is $2^k = 16$ (i.e., $> n = 7$), we need to partition the station nodes into groups and send the message from the source in steps. In Q_7 , the (7, 4) Hamming code is used for the station nodes. The number of routing steps between a source node and station nodes is 2 because the number of station nodes is larger than n ($2^4 - 1 \geq 7$). The station nodes should be divided into smaller groups.

According to the partitioning procedure described in Figure 4.1, the smallest number of subgroups in the first division is 2, i.e., there are eight code words in each tow subgroups. Different choices for N_{sg} are given in Table 7.

In either case, we are going to need two steps for informing all the station nodes.

While for Q_7 , the choice of N_{sg} is irrelevant, for Q_8 and Q_9 this choice matters since we are basing the broadcast algorithm for a larger size cube on partitioning such cubes into disjoint Q_7 s and then running the algorithm developed for Q_7 concurrently for all partitioned Q_7 s refer to Remarks 9 and Theorem 6. It can be easily verified that for any of the choices of $N_{sg} = 2, 4, 8$, the algorithm obtained gives rise to at least four broadcasting steps for Q_8 and Q_9 . To reduce this number, we use a special broadcast algorithm for Q_7 , which divides the code words based on their weight and not their being in disjoint cubes. The (7, 4) Hamming codes' weight distribution is

$$A(x) = 1 + 7x^3 + 7x^4 + x^7$$

Table 1 shows all (7, 4) Hamming code words sorted by weight. So we divide code words into two groups: one group is weight 0 and 3, the other is weight 4 and 7. The first routing step is to send the message from a source node to two groups, i.e., the source node to the weight 0 node and the weight 7 node. The second routing step is to send the message from the weight 0 node to the weight 3 nodes, and from the weight 7 node to the weight 4 nodes, and from the weight 7 node to the weight 4 nodes. Table 3 shows the paths between the weight 0 node and the weight 3 nodes. The weight 7 node to the weight 4 nodes path are the same as Table 4.1, because the Hamming code is symmetric. Thus, the number of routing steps to broadcast a message is three.

Table 2. CSR Table for the second Routing Step in Q_7

Destination Nodes	Path from Source (00...0) to Destination
0001110	123
0010011	014
0100101	250
0111000	453
1001001	360
1010100	642
1100010	561

Table 3. Example: $n = 15, k = 11, m = 4$

No. of divisions	N_{sg}	N_{sc}	N_{ud}
1	16	Q_7s	11
2	8	Q_4s	8
3	8	Q_1s	5
4	-	-	-

Example 7. Consider Q_{15} where $n = 15, k = 11, m = 4$. We divide the 2^{11} code words into 16 subgroups with each subgroup containing 2^7 code words arranged as the nodes of a Q_7 . After the first round, $N_{ud} = 15 - \log_2(16) = 11$. In the second round, each Q_7 is partitioned into eight Q_4s and $N_{ud} = 11 - \log_2(8) = 8$ (see Table 3.) After the third round of division, $N_{ud} = 5 \geq$ the number of node in Q_1 . So the process of informing code words will be completed in one more step by informing all the nodes in every Q_1 . It should be noted here that the number of steps would not have

changed if we had split the code words into eight Q_8 s or even four Q_9 s. Choosing the smallest $\log_2 N_{sg}$ (in this case, four), however, has another advantage which will be clear later.

Table 4. Example: $n = 31, k = 26, m = 5$

No. of divisions	N_{sg}	N_{sc}	N_{ud}
1	32(16)	$Q_{21}s(Q_{22}s)$	26(27)
2	16	$Q_{17}s(Q_{18}s)$	22(23)
3	16	$Q_{13}s(Q_{14}s)$	18(19)
4	16	$Q_9s(Q_{10}s)$	14(15)
5	8(16)	Q_6s	11
6	8	Q_3s	8
7	-	-	-

Example 8. Consider Q_{31} where $n = 31, k = 26, m = 5$. The code words are divided into 32 Q_{21} s and $N_{ud} = 31 - 5 = 26$. In the next round, since

$$\lfloor \log_2 26 \rfloor = 4,$$

it is possible to inform only 24 nodes, which means each Q_{21} is further divided into 16 Q_{17} s (see Table 4.4). The process is repeated until the code words are divided into 16 Q_9 s and $N_{ud} = 14$. Again, since $\lfloor \log_2 14 \rfloor = 3$, each Q_9 is divided into 8 Q_6 s, and later each Q_6 is divided into eight Q_3 s with $N_{ud} = 8$. Since $N_{ud} \geq$ the number of nodes in Q_3 , the process of informing the code words is completed in one more step. Table 4.4 depicts this scenario with the numbers in parentheses corresponding to the

alternative case where we split the code words into 16 Q_{22} s. The numbers inside the parentheses correspond to the minimum N_{sg} or $sf = 4$.

Table 5 shows the total number of routing steps needed (c) and the possible value of $\log_2 N_{sg}$ in the first routing step for different (n, k) Hamming codes. For example, for $(n, k) = (63, 57)$, the choice of $\log_2 N_{sg} = 5, 6$ both yield $c = 13$, whereas any choice of $\log_2 N_{sg} = 1, 2, 3, 4$ will produce $c = 14$; thus, the acceptable values of $\log_2 N_{sg}$ in this example are 5 and 6, while the value of $sf = 5$. Note that for each (n, k) code in Table 5, the optimal cases have been shown in boldface boxes.

Table 5. Possible Routing Steps for Hamming Codes

m	2	3	4	5	6	7	8	9	10
(n, k)	(3,1)	(7,4)	(15,11)	(31,26)	(63,57)	(127,120)	(255,247)	(511,502)	(1023,1013)
$c, \log_2 N_{sg}$	1 1	2 3	4 4	7 5	13 6	24 7	41 8	73 9	130 10
		2 2	4 3	7 4	13 5	24 6	41 7	73 8	130 9
		2 1*	4 2	8 3	14 4	24 5	41 6	73 7	130 8
		3 1	5 1	8 2	14 3	24 4	42 5	73 6	130 7
				8 1	14 2	24 3	42 4	73 5	130 6
					14 1	24 2	42 3	74 4	130 5
						25 1	42 2	74 3	130 4
							42 1	74 2	130 3
								74 1	131 2
									131 1

Next, we derive an upper and a lower bound for the number of routing steps needed for a given (n, k) code. The routing steps necessary to deliver the message to all the code words is the same as the number of partitioning stages. In the first stage, reduction of m

will take place so that after the first division $N_{ud} = n - m = 2^m - m - 1$. Successive partitioning will reduce the N_{ud} by $(m - 1)$ until $N_{ud} < 2^{m-1} - 1$. The number of partitioning stages for reduction by $(m - 1)$ will therefore be

$$\frac{(2^m - 1) - (2^{m-1} - 1)}{m - 1} = \frac{2^{m-1}}{m - 1}, \quad \text{where } n = 2^m - 1.$$

The division process will continue until $N_{ud} \geq N_{sc} - 1$. The number of stages can be calculated by adding the number of partitions for each reduction number. The division process terminates at the reduction number x where x depends on m . For instance, $x = 3$ for $m < 11$. Refer to Figure 12. Thus, we can now state the following theorem.

Theorem 4. The number of routing steps $T(n)$ between a source node and stations in Q_n is given below, where $n = 2^m - 1$.

$$\sum_{j=x}^{m-1} \frac{\lfloor 2^j \rfloor}{j} + 1 \leq T(n) \leq \sum_{j=x}^{m-1} \frac{\lceil 2^j \rceil}{j} + 1.$$

4.2 Hypercube Q_n , $2^m - 1 < n < 2^{m+1} - 1$

When $n \neq 2^m - 1$ for some integer m , we do not have any (n, k) Hamming codes; thus, technique developed in previous section cannot be directly applied for such hypercubes, but we can extend the scheme in the following way. Q_n is partitioned into 2^{n-n_0} many disjoint Q_{n_0} 's where $n_0 = 2^m - 1$. Through successive m -reductions, all Q_{n_0} 's (disjoint subcubes of dimension n_0) will receive the message; after that, the algorithm for previous special case is invoked to complete broadcasting. Thus, we state the following theorem.

Theorem 5. For any given Q_n , where $2^m - 1 < n < 2^{m+1} - 1$ for some integer m (and assume $n_0 = 2^m - 1$), the total number of routing steps for broadcast is given by $T(n) = T(n_0) + r_1$ where $n_0 = 2^m - 1$, and

$$r_1 = \left\lceil \frac{n - n_0}{m} \right\rceil.$$

Corollary 2. Broadcasting in Q_{n_0+i} can be done in $T(n_0) + 1$ routing steps, where $0 < i \leq m$.

The result in the above corollary can be improved if we pick the smallest N_{sg} of all acceptable values; see Remark 9.

Theorem 6. Let $2^m - 1 = n_0 \leq n \leq n_0 + x$ be such that $T(n) = T(n_0)$. Then $x = m - sf$ where $sf = \log_2$ minimum of all acceptable values of N_{sg} in the first partitioning step of n .

Proof: The proof is straightforward. It follows that $\lfloor \log_2(n+1) \rfloor = x + sf$.

So the first reduction in Q_n is by $(x + sf)$. But the first reduction of Q_{n_0} is by sf . The same number of steps will be required after the first division for both Q_n and Q_{n_0} .

Table 6. CSR Table for the First Routing Steps in Q_5

0000000	0011101
0001110	0101011
0010011	0110110
0100101	1000111
0111000	1011010
1001001	1101100
1010100	1110001
1100010	1111111

Table 7. Partitioning for Q_7

No. of divisions	N_{sg}	N_{sc}	N_{ud}
1	8	Q_1 's	4
	4	Q_2 's	5
	2	Q_3 's	6
	2	0^7 and 1^7	-
2	-	-	-

Table 8. Partitioning for Q_9

No. of divisions	N_{sg}	N_{sc}	N_{ud}
1	2	Q_3 's	6
2	-	-	-

Example 9. Let $n = 15$, $k = 11$. Then $m = 4$ and $sf = 2$. Therefore, $x = m - sf = 2$. This means that all the following cubes require the same number of steps for broadcasting: Q_{15} , Q_{16} , and Q_{17} .

Table 9. CSR Table for the First Routing Step in Q_9

Destination Nodes	Path from Source (00...0) to Destination
001111111	0123456
010000000	7
011111111	12345670
100000000	8
101111111	23456810
110000000	6786
111111111	345678210

Table 10. CSR Table for the first Routing Step in Q_5

Destination Nodes	Path from Source (00...0) to Destination
01000	3
10000	4
11000	2342

Table 11. CSR Table for the first Routing Step in Q_{10}

Destination Nodes	Path from Source (00...0) to Destination
0010000000	7
0100000000	8
0110000000	6786
1000000000	9
1010000000	5795
1100000000	4894
1110000000	37893

Example 10. Consider $n = 4$; we divide Q_4 into two Q_3 's (two disjoint subcubes each of dimension 3): $0xxx$ and $1xxx$. The Q_4 can accommodate

four nodes at once, which means one routing step. We choose Q_3 's nodes for the first routing step as {0000, 0111, 1000, 1111}:

1. **Routing step 1.** Source node (0000) to four Q_3 station nodes; (see Table 6 for the paths generated by the CSR algorithm.)
2. **Routing step2.** Each station node sends a message to its neighbor nodes.

```

main()
  if  $n = 2^m - 1$  then
     $\alpha = 0$ ; Hamming ( $\alpha, n, m$ ) : exit
  else
    begin
       $m = \lfloor \log_2(n + 1) \rfloor$ ; compute_sf( $n, m, sf$ )
       $x = m - sf$ ;  $n_0 = 2^m - 1$ ;
      if  $n \leq n_0 + x$  then
        begin
           $x' = n - n_0$ ;  $fix = 0$ ;
          multiple_bcast ( $msg, fix, x'$ );
          Hamming ( $x', n_0, k$ ); exit
        end
      else
        begin
          Compute  $q$  and  $r$  ( $r < m$ ) such that  $n - n_0 = qm + r$ ;
           $fix = 0$ ;  $\alpha = 0$ ;
          for  $i = 1$  to  $q$  do
            {multiple_bcast( $msg, fix + \alpha, m$ );  $fix = fix + m$ ;  $\alpha = \alpha + m$ ;}
          multiple_bcast ( $msg, fix + \alpha, r$ );
          Hamming ( $qm + r, n_0, k$ ); exit
        end
      end
    end
  
```

Figure 13. A general broadcasting algorithm for Q_n

Procedure Hamming (α, n, m)

C = set of code words for (n, k) Hamming code.

$k = n - m - 1$;

compute_sf(n, m, sf)

multiple_bcast (msg, α, sf);

$N_{sc} = 2^{k-sf}$; $N_{ud} = n - sf$; $fix = sf$; $j = m - 1$;

while ($N_{ud} \geq N_{sc} - 1$) **do**

while $N_{ud} \geq (2^j - 1)$ **do**

multiple_bcast ($msg, fix - \alpha, j$);

$N_{sc} = N_{sc} / 2^j$; $N_{ud} = N_{ud} - j$; $fix = fix + j$;

end while

$j = j - 1$;

end while

multiple_bcast ($msg, fix, 0$)

Broadcast message from each code word to its neighbor

Figure 14. Procedure Hamming (α, n, m)

Procedure Compute_sf (n, m, sf);

$d = \lfloor 2^m / m \rfloor$; $i = m$;

while $\lfloor 2^i / i \rfloor = d$ **do** $i = i - 1$;

$sf = i + 1$

Fig 15. Procedure *Compute_sf* (n, m, sf)

Example 11. Consider $n = 9$. We know that by Theorem 6, the number of routing steps in Q_9 is 3. Table 9 shows the routing table for the first routing step in Q_9 . The second and third routing steps for each subcube are the same as in Q_7 . Note that in Q_7 , in the first step the message is sent only to two groups, i.e., two representatives in two disjoint groups. These

disjoint groups are not two Q_6 s, but they are respectively the set of weight 3 and weight 4 nodes. Nevertheless, in Q_9 and in the first step, the message is sent to eight representatives in for disjoint Q_7 s. The labels of these cubes are $00x^7$, $01x^7$, $10x^7$, and $11x^7$, respectively. The rest of the labels of the pair of representatives in each Q_7 are identical and equal to 0^7 and 1^7 . The remaining two steps of the algorithm are the same as those in Q_7 , i.e., within each Q_7 , first weight 3 and weight 4 nodes are informed, and those nodes in turn will inform the remaining nodes. The main reason that Q_7 and Q_9 require the same number of steps is that in the first step of the algorithm, for Q_7 only two nodes are informed, whereas in Q_9 , eight nodes get informed. As seen before, informing more than two nodes in the first step for Q_7 does not reduce the total number of broadcast steps.

It should be noted that had we divided the station nodes into eight Q_3 s as depicted in Table 8, we would have required four steps to do the broadcasting. This is true since in the first step, we would inform eight representatives in eight disjoint Q_3 s. It takes two additional steps to inform all the station nodes with each Q_3 . Finally, in the fourth step, all the remaining nodes will get the message.

According to Corollary 2, Q_5 needs just one more than the routing steps needed by Q_3 for broadcast; broadcast in Q_{10} can be done in four routing steps (one more than the number of steps needed by Q_7). We give two examples for $n = 5, 10$.

Example 12. Consider Q_5 ($n = 5$); we can divide a Q_5 into four Q_3 subcubes – $00xxx$, $01xxx$, $10xxx$ and $11xxx$; we use station node 00000 ,

01000, 10000, and 11000. The Q_5 can only accommodate five nodes in one routing step.

Routing Step 1: Source node to Q_3 subcubes (see Table 10 for paths to the station nodes).

Routing Step 2 & 3: Follow the same step as in Q_3 for each subcube.

Example 13. Consider Q_{10} ($n = 10$); by Corollary 2, the number of routing steps is four. We divide a Q_{10} into eight Q_7 subcubes. The second through the fourth routing steps for each subcube are the same as the first to third routing steps in Q_7 . The Q_{10} can accommodate 10 nodes in one routing step.

Routing Step 1. Source node to Q_7 subcubes (see Table 11)

Routing Step 2, 3, and 4. Follow the same steps as for Q_7 for each subcube.

4.3 Unified Algorithm for Q_n with Arbitrary n

In this section, we collect the ideas presented in the previous two subsections and present a general broadcasting algorithm for any hypercube Q_n for any given n . Basically, we combine the two cases presented in the earlier sections: if n is of the form $2^m - 1$, then we use the Hamming codes directly; otherwise, when $n = n_0 + R$, R -reductions are done (through several m -reductions) until the number of dimensions is reduced to n_0 , and then the code for $n_0 = 2^m - 1$ is invoked.

We use two primitives in the description of the algorithm: 1) **multiple_bcast**(*msg*, α, β): This procedure executes concurrent broadcast of the message “*msg*” in a number of disjoint subcubes, each of dimension β and each having its own source node – each source node has an address of the type $Y^\alpha 0^{n-\alpha}$, where $Y \in \{0, 1\}$, for each source node the corresponding β -cube is $Y^\alpha 0^{n-\alpha-\beta} X^\beta$, where $X \in \{0, 1\}$, and total number of source nodes participating in the multiple concurrent broadcast is 2^α . Note that the broadcast is performed by computing the node disjoint paths given the source node and the β -cube and then using wormhole broadcasting along those paths. 2) **compute_sf**(*n*, *m*, *sf*): This procedure computes the value of *sf* from given values of *n* and *m*; the pseudocode is shown in Figure 15.

The pseudocode of the main procedure is shown in Figure 13, while the pseudocode of the Hamming procedure is shown in Figure 14. Note that the Hamming procedure does the broadcast when $n = 2^m - 1$ (when $n = 2^m - 1$, $\alpha = 0$; this parameter α is needed when $n \neq 2^m - 1$); first, “*sf*” is computed and an “*sf*” reduction is done which is followed by “*m*-1” reductions, “*m*-2” reductions, etc., until $N_{ud} \geq N_{sc} - 1$.

4. 4. Deadlock

One of the major problems inherent to wormhole routing is deadlock whereby channels are occupied by some flits indefinitely. A typical solution to deadlock is to ensure no cycle is generated during routing. While the e-cube routing (i.e. routing in ascending/descending dimension order) is an effective and simple solution for routing in hypercube networks, this approach is not applicable here. This is true, since dimensions

may need to be traversed in a random order to reach the station nodes. Therefore, if during broadcasting, some nodes decide to communicate, our approach does not guarantee a deadlock-free scheme. However, assuming that during broadcasting, there is no other communication taking place, the proposed method avoids any deadlock in the system. In our method, flits propagate through the network always in one direction (albeit not necessarily ascending or descending) by utilizing the hierarchy of the hypercube. Paths are always established in disjoint subcubes, and when they are not, they are predetermined, unique, and the shortest for each destination node. Recall that since perfect codes are used as station nodes, each such node is assigned its destination (i.e., its neighbors) unambiguously, which it will inform in the last step of the algorithm by simply activating all its communication ports. This is in contrast to the case for Quasi-perfect codes whereby a node may be of equal distance from two stations, thus causing ambiguity as to which station node must be the informer.

CHAPTER 5

CONCLUSION

We have given a broadcasting algorithm for an n -dimensional hypercube with wormhole routing and all-port communication. Although the previous algorithm by Ho and Kao [9] gave a good upper bound, our algorithm improves the upper bound for $n = 8, 9, 11, 12, 14$, and 15 . To quantify the merit of an upper bound, we may define a measure as follows:

$$\rho = \frac{\text{\# of nodes in the network}}{\text{\# of nodes ideally covered in } T(n) \text{ steps}} = \frac{2^n}{(n+1)^{T(n)}}$$

The ρ of ideal $T(n)$, where ideal reflects the case in which we can achieve broadcasting optimally, i.e., $T(n) = \text{lower bound}$, in Q_5 and Q_{10} are 0.90 and 0.77, respectively. If we fully utilize parallel port capability in each routing step, ρ will be 1. Clearly, it is very difficult to fully utilize parallel ports. Ho and Kao proved a lower bound of Q_5 should be 3, not 2. The higher the ρ , the more difficult will be to achieve the lower bound of routing steps in Q_n . Figure 16 shows the merit of each bound for n . A summary of routing steps for each n -dimensional hypercube is given in Table 12.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Lower bound	1	2	2	2	3	3	3	3	3	3	4	4	4	4	4
Upper bound[9]	1	2	2	2	3	3	3	4	4	4	5	5	5	6	6
Upper bound	1	2	2	2	3	3	3	3	3	4	4	4	5	5	5

Table 12. Lower and Upper Bounds of Routing Steps for Broadcasting

Fig 16. Merit of each bound for Q_n

BIBLIOGRAPHY

- [1] Athas, W. C., and Seitz, C. L., "Multicomputers: Message-passing concurrent computers", IEEE Computers, pp. 9-25, Aug. 1988
- [2] Berlekamp, E. E., "Algebraic Coding Theory", Aegina Park Press, 1984
- [3] Borkar, S. B., et. al, "iWarp: An integrated solution to high-speed parallel computing", Proceedings of Supercomputing '88, pp. 330 – 339, Nov. 1988
- [4] Dally, W. J., and Seitz, C. L., "The Torus Routing Chip", J. Distributed Computing, Vol. 1, No. 3, 1986, pp. 187-196.
- [5] Dally, W. J., "Virtual Channel Flow Control", IEEE Trans. Computers, vol. 3, No. 3, Mar. 1992, pp. 194-205
- [6] Etzion, T. and Vardy, A., "Perfect Binary Codes: Constructions, Properties, and Enumeration", IEEE Trans. on Information Theory, Vol. 40, No. 3, pp. 734-763, May. 1994
- [7] Graham, R. L. and Sloane, N. J. A. "On the Covering Radius of Codes", IEEE Trans. Inform. Theory, vol. 31, no. 3, pp. 386, May 1985
- [8] Ho, C. T. and Raghunath, M. T., "Efficient Communication Primitives on Hypercubes", J. Concurrency: Practice and Experience, , Vol. 4, No. 6, pp. 427-457. Sep. 1992.
- [9] Ho, C. T. and Kao, M. Y., "Optimal Broadcast in All-Port Wormhole-Routed Hypercubes", IEEE Transactions on Parallel and Distributed Systems, Vol 6, No. 2, pp. 200-204, Feb. 1995.
- [10] Latifi, S., "A Distributed Approach for Fast Gathering and Broadcasting in q-ary Cubes", IPCCC '96, pp. 406-412, Mar. 1996
- [11] MacWilliams. F. J. and Sloane, N. J., The Theory of Error Correcting Codes, Amsterdam, North-Holland, 1993.
- [12] McKinley, P. K. and Trefftz, C., "Efficient Broadcast in All-port Wormhole-routed Hypercubes", in 1993 Int. Conf. Parallel Process.. vol. II, pp. 288-291, Aug. 1993

- [13] McKinley, P. K., Xu, H., Esfahanian, A. H. and Ni, L. M., "Unicast Based Multicast Communication in Wormhole Routed Networks", IEEE Trans. Parallel and Distributed Systems, vol. 5, no. 12, pp. 1,252-1,264, Dec. 1994
- [14] Ni, L. M. and McKinley, P. K., "A Survey of Wormhole Routing Techniques in Direct Networks", Computer, pp. 62-76, Feb. 1993
- [15] Saad, Y. and Schultz, M. H., "Topological Properties of Hypercubes", IEEE Trans. On Computers. Vol. 37, No 7, pp. 867 -872, July 1988
- [16] Wicker, S. B., Error Control Systems for Digital Communication and storage, Prentice Hall, 1995.
- [17] M. Annaratone, "The K2 Parallel Processor: Architecture and Hardware implementation", Proceedings of International Symposium on Computer Architecture, May 1990, pp. 92-101.
- [18] Chang & Wang & Kankanhalli, "Multidimensional On-line Bin-packing: An Algorithm and its Average-case Analysis", IPL, 1993, vol. 48.
- [19] Berkey & Wang, "A Systolic-Based Parallel Bin Packing Algorithm", IEEE TPDS, 1994.

VITA

Graduate College
University of Nevada, Las Vegas

Hyosun Ko

Home Address:

955 Wensley Ave #H
El Centro, CA 92243

Degree

Bachelor of Science, Electrical Engineering, 1990
Hanyang University, Seoul, Korea

Publications:

H. Ko, S. Latifi, and P.K. Srimani, "Near-optimal Broadcast in All-port Wormhole-routed Hypercubes using Error Correcting Codes", IEEE International Performance, Computing, and Communications Conference, pp. 8-14, Feb. 1998.

S. Latifi, Hyosun Ko, and P.K. Srimani, "Node-to-Set Vertex Disjoint Paths in Hypercube Networks", International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'98)}, vol. II, pp. 661-665, July 1998, Las Vegas.

H. Ko, S. Latifi, and P.K. Srimani, "Near-optimal Broadcast Algorithms for Wormhole Hypercubes", IEEE Transactions on Parallel and Distributed Systems. Vol. 11, No. 3, March 2000, 247 - 260

Thesis Title : Near -Optimal Broadcast in All-Port Wormhole-Routed
Hypercubes Using Error-Correcting Codes

Thesis Examination Committee:

Chairperson, Dr. Shahram Latifi, Professor, Ph. D
Committee Member, Dr. Yahia Baghzouz, Professor, Ph. D.
Committee Member, Dr. Henry Selvaraj, Assistant Professor, Ph. D.
Graduate Faculty Representative, Dr. Woosoon Yim, Associate Professor, Ph. D.