1989

Hypercube-Based Topologies With Incremental Link Redundancy.

Shahram Latifi
Louisiana State University and Agricultural & Mechanical College

Order Number 9017272

Hypercube-based topologies with incremental link redundancy

Latifi, Shahram, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1989

U·M·I

300 N. Zeeb Rd.
Ann Arbor, MI 48106

HYERCUBE-BASED TOPOLOGIES WITH INCREMENTAL LINK REDUNDANCY

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Electrical and Computer Engineering

by
Shahram Latifi
M.S., Teheran university, February 1980
M.S., Louisiana State University, May 1986
August 1989

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to Dr. Ahmed El-Amawy, Associate Professor of Electrical Engineering, for his invaluable supervision and guidance of the work represented by this dissertation.

Appreciation is also extended to Dr. Owen T. Tan, Dr. Subhash C. Kak, Dr. Suresh Rai of the Electrical and Computer Engineering Department, Dr. J. Bush Jones of the Department of Computer Science, and Dr. Richard Parish of the Department of Agricultural Engineering for serving as members of the examining committee.

Above all the author is grateful to his parents, sisters, and brother, whose patience, understanding, and inspiration have made this study a reality.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Hypercube structures have received a great deal of attention due to the attractive properties inherent to their topology. Parallel algorithms targeted at this topology can be partitioned into many tasks, each of which running on one node processor. A high degree of performance is achievable by running every task individually and concurrently on each node processor available in the hypercube. Nevertheless, the performance can be greatly degraded if the node processors spend much time just communicating with one another. The goal in designing hypercubes is, therefore, to achieve a high ratio of computation time to communication time. The dissertation addresses primarily ways to enhance system performance by minimizing the communication time among processors.

The need for improving the performance of hypercube networks is clearly explained. Three novel topologies related to hypercubes with improved performance are proposed and analyzed. Firstly, the *Bridged Hypercube (BHC)* is introduced. It is shown that this design is remarkably more efficient and cost-effective than the standard hypercube due to its low diameter. Basic routing algorithms such as one to one and broadcasting are developed for the BHC and proven optimal. Shortcomings of the BHC such as its asymmetry and limited application are clearly discussed.

The *Folded Hypercube (FHC)*, a symmetric network with low diameter and low degree of the node, is introduced. This new topology is shown to support highly efficient communications among the processors. For the FHC, optimal routing algorithms are developed and proven to be remarkably more efficient than those of the conventional hypercube. For both BHC and FHC, network parameters such as average distance, message traffic density, and communication delay are derived and comparatively analyzed. Lastly, to enhance the fault tolerance of the hypercube, a new design called *Fault Tolerant Hypercube (FTH)* is proposed. The FTH is shown to exhibit a

graceful degradation in performance with the existence of faults. Probabilistic models based on Markov chain are employed to characterize the fault tolerance of the FTH. The results are verified by Monte Carlo simulation.

The most attractive feature of all new topologies is the asymptotically zero overhead associated with them. The designs are simple and implementable. These designs can lend themselves to many parallel processing applications requiring high degree of performance.

CHAPTER 1

Introduction

Interconnection networks came about as a natural result of advances in computer technology in response to increasing demands for improved system performance. As computer systems evolved from the batch-processing models of the 1960's to the time-sharing models of the 1970's, their evolution was basically confined within the von Neumann architectural model, with hardware costs being a significant limiting factor. However, the advent of VLSI technology in the early 1980's had a great impact on computer architecture. It is now economically feasible to construct a multiprocessor system by interconnecting a large number of processors and memory modules. In addition, because of today's increased performance requirements, the number of functional modules (homogeneous or heterogeneous) in parallel processing systems continues to rise as the domain of applications grows. Dealing with this trend is quite a challenge to computer architects.

One fact immediately presents itself. System performance in the future can only be significantly increased through additional concurrent processing as the speed of logic devices reaches a limit. Supercomputers built around few processing units - the Cray-2, the NEC SX-2, or the Fijitsu VP 200- may already be within an order of magnitude of the technological limit. This theoretical upper bound, i.e. some 3 gigaflops (billions of floating operations per second), is established by the length of time it takes electrical signals to propagate, traveling through the wires at about half the speed of the light. Future scientific problems, however in such fields as fluid dynamics, aerodynamics, computational chemistry, weather forecasting, imagery analysis, missile guidance, ballistic-missile defense, robot vision, and speech understanding, are expected to require processing rates far in excess of that 3 gigaflops limit.

But by dividing applications among many processors working in parallel, rates in the teraflops range- trillions of floating point operations per second- are in theory possible. Systems with hundreds and even thousands of processors have already been built, and intensive work is now under way to build practical machines that can be expanded to accommodate many more processors.

A major problem in designing such machines is the construction of an interconnection network to provide interprocessor communication and , in some cases, memory access for the processors. The task of interconnecting N processors and N memory modules, where N may be more than one thousand, is not trivial. The interconnection scheme must provide fast and flexible communications at a reasonable cost. But for such a large number of processing elements, conventional interconnection organizations, such as timeshared buses, become so awkward that the required performance level cannot be achieved satisfactorily. The single shared bus, as shown in Figure 1.1, is not sufficient, because its performance degrades drastically as the number of communicating processors increases. Ideally, each processor should be linked directly to every other processor so that the system is fully connected, as shown in Figure 1.2 for $N=6$. In this example, it could be assumed that each node is a processor with its own memory. Unfortunately, these configurations are highly impractical when N is large because $N-1$ unidirectional links are required for each processor. For example, for $N=2^{10}$, more than one million unidirectional links would be needed.

An alternative interconnection scheme is the crossbar network shown in Figure 1.3. In this example, the processors communicate through the memories. The network may be viewed as a set of intersecting lines, where interconnections between processors and memories are specified by the cross point switches at each line intersection. The difficulty with crossbar networks is that N^2 crosspoint switches are needed. As a result, the cost of the network grows with N^2, making it infeasible for large systems.

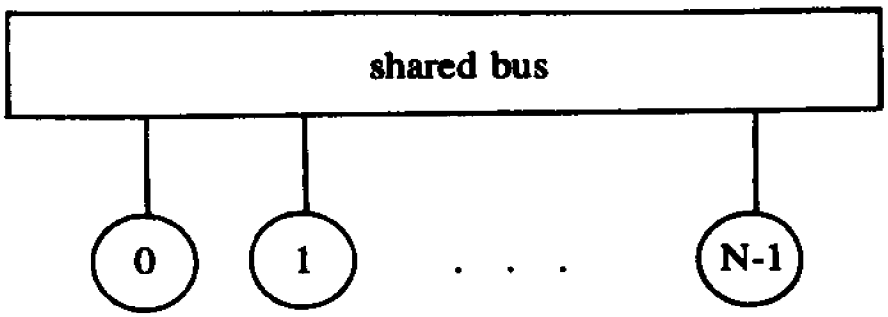


Figure 1.1. A Shared Bus providing Communications for N Processors.

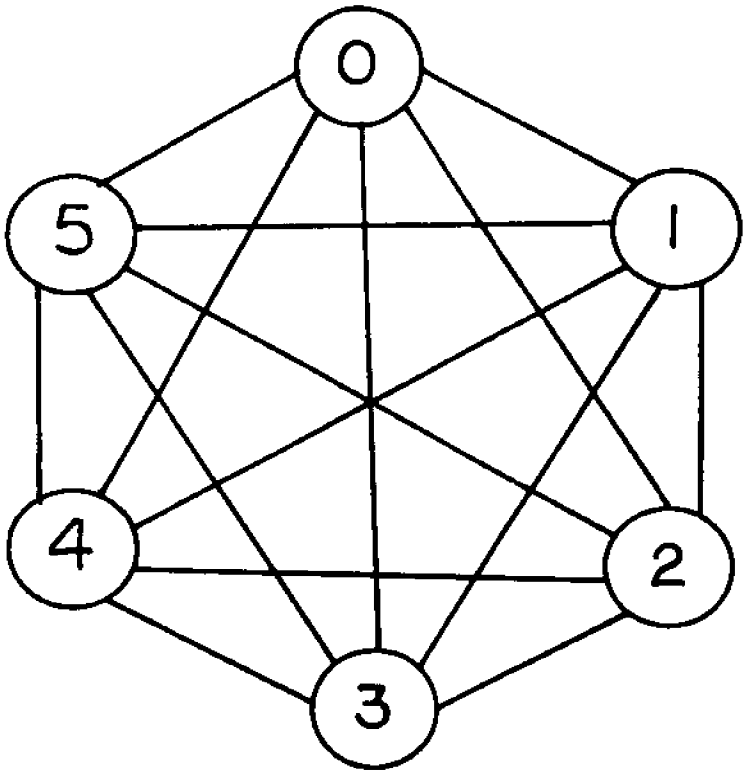


Figure 1.2. A Completely Connected System for N=6.

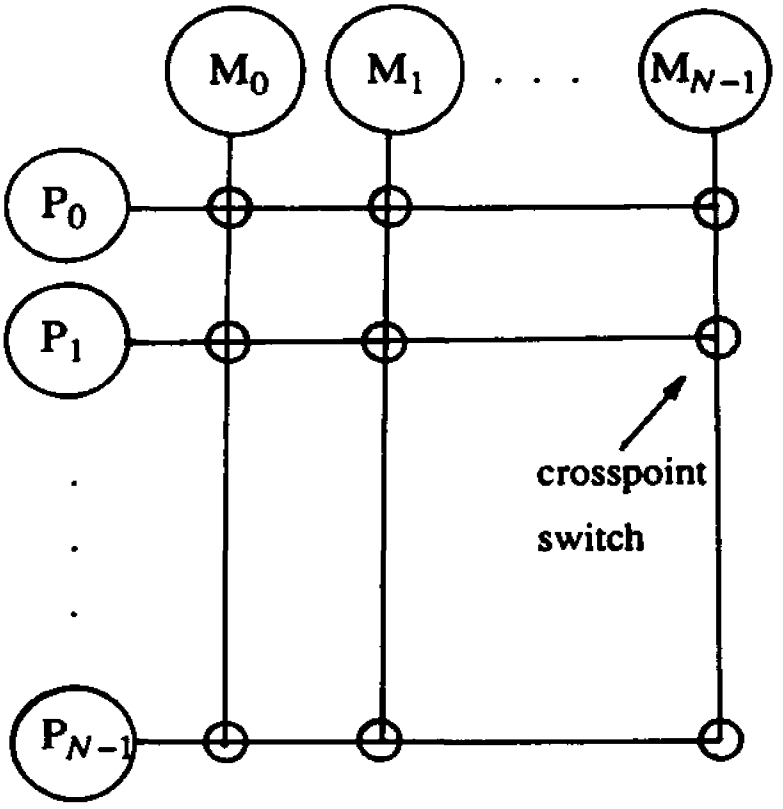


Figure 1.3. Crossbar Network.

To solve the problem of providing a fast, efficient communication means at a reasonable cost, many different networks between the extremes of the single bus and the completely connected scheme have been proposed in the literature [1]-[9]. There is no single network that is generally considered best. The cost effectiveness of a particular network design depends on such factors as the computational tasks for which it will be used, the desired speed of interprocessor data transfers, the actual hardware implementation of the network, the number of processors in the system, and any cost constraints on the construction. A variety of networks have been proposed and are overviewed in numerous survey articles and books [10]-[17]. In the next section, the design issues involved with interprocessor communication techniques as applied to parallel processing systems are explored. The emphasis is on local interconnection networks that could, conceptually, fit in a room complex.

1.1 Design Issues

The following issues are considered the most fundamental in the design of interconnection networks [3]-[7], [9].

1.1.1 Network Topologies

A network can be depicted by a graph in which nodes represent switching points and edges represent communication links. The overall graph representation is called network topology. The topology of the network is a key parameter in parallel architectures. Many topologies have been proposed or used. They tend to be regular and can be grouped into two categories- static and dynamic. In a static topology, each switching point is connected to a processor, while in the dynamic case only those switching points in the input/output side are connected to processors.

1.1.2 Control Strategies

Control strategies concern how to route data from a source to various destinations. Every network needs a control strategy. The control of data flow can be managed by a centralized controller (centralized control) , or by individual switching points (distributed control). Depending on individual network definitions, some networks have very simple and effective control strategies while others need relatively time-consuming algorithms to calculate the proper control settings for switching points. In a manner similar to today's mail system, each node need an address so that the routing path(s) can be uniquely specified. Therefore, address labeling has to be done before one can actually specify a proper routing path from a source to a destination.

Data routing can be implemented in two quite different switching methods: circuit switching and packet switching. Because of their simplicity, circuit switching networks were the first to appear. In circuit switching, a physical path is actually established between a source and a destination. This path should remain unaltered for the duration of data transfer. Techniques employed for implementing such networks include *Time Division Multiplexing (TDM)* and *Frequency Division Multiplexing (FDM)* [13]. In packet switching, data is put in a packet and routed through the interconnection network without establishing a physical connection path. Thus, it is possible to send many packets of a message simultaneously because they are independent. These packets, when in transmission, are controlled by an algorithm which determines their route of travel from the source to the destination, called the *routing algorithm*. Not all packets of a message are necessarily transmitted through the same set of intermediate nodes in traversing the "path" of travel between source and destination. Generally, each packet is routed individually. Thus, the basic function of the routing algorithm is to minimize delay and maximize throughput in the network while being simple to compute. Integrated switching, which incorporates both circuit switching and packet switching, can also be implemented for more flexibility in data communication.

1.1.3 Network Analysis

Various measurements must be performed to observe network characteristics, which in turn will be used to decide whether the network is adequate in regard to system requirements. Characteristics to be observed include those which are not related to communication request distribution, such as combinatorial power, overall cost, reliability, and the number of connections per node; and those which are related to communication request distribution, such as bandwidth, message delay, and message density per link. Combinatorial power in permutation is one of the most important characteristics since parallel permutations of data are needed in array processing. The quality of parallel algorithms is mostly reflected by the way in which the permutations of data are done. It would be desirable to have an interconnection network which can realize necessary permutations in a minimal number of routing steps. The analysis is then concerned with the ability of an interconnection network to realize desirable permutations and the way it realizes a particular permutation.

1.1.4 Reliability

In multiple-processor systems, reliable processing relies partly on the integrity of data communication paths. Since a large network is also prone to be faulty, fault-tolerance issues are crucial. A fault-tolerant interconnection network can tolerate faults to some degree and still provide reliable and gracefully degradable communication. There are several ways to achieve fault tolerance: multiple paths between an input/output pair, multiple-port connection, multiple-pass routing, and fault-tolerant switching elements. Fault tolerance is an inherent characteristic of the interconnection network. Once the interconnection network has been constructed, there is little one can add to enhance its deserved fault-tolerance level. This implies that fault tolerance must be considered as one of the prime factors in choosing a proper interconnection network for system connection.

1.1.5 Reconfiguration Techniques

Reconfiguration techniques are used to allocate hardware resources, such as processors and communication switches, to a specific task. The hardware resources allocated are usually interconnected to form a suitable network topology for the task. The reconfiguration is typically performed when a task requests resources through a system controller. It can also be performed to provide fault tolerance when the current configuration contained faulty components. An efficient reconfiguration technique should embed various network topologies with very short overhead, while utilization of resources remains very high, The reconfiguration technique is particularly important in scheduling large-scale systems for high performance computation, since there is a need for matching architecture and algorithm and for allocating both processor and communication resources.

1.2 Research Objectives

The main objective of this work is to develop new hypercube-based interconnection structures for highly parallel computing systems. The standard hypercube topology is selected as the framework for its many appealing features. The new networks will thus be relatives of the hypercube. New network topologies are investigated which can be obtained by selectively and carefully introducing a small amount of link redundancy to the basic structure aimed at substantially enhancing some critical performance measures. It is inconceivable that a single network topology such obtained will be able to enhance all performance measures relevant to all parallel processing environments. However, it is quite realistic to anticipate significant improvement in a subset of network parameters as a result of adopting a certain link redundancy pattern.

Since the hypercube is an n–dimensional topology, it is practically impossible to implement it and still preserve the equal length feature. The natural question is thus: Is the standard n–cube (hypercube) the best topology in its class if we disregard the

equal link length requirement? . The answer to this question cannot be a definite one since it must be qualified. This work investigates in detail the different responses to the above question. The intention is to research topologies based on the hypercube with a small amount of link redundancy.

It is quite evident that performance parameters of redundant cubes will be mainly dependent on the a) amount of redundancy and b) the exact location of the redundant links. For example, in a recent study at LSU [18] to improve simulation times of other network topologies on redundant cubes, it has been found that specific (limited) redundancy patterns could help to substantially improve the bounds. It is intuitive that certain redundant link positioning makes the cube-based network capable of efficiently performing certain inter-node communication patterns [18]. It is clear that if one is to consider improving other aspects of hypercube performance, such as diameter, then positioning of extra links would be approached differently. For another example, if the system is to be much more resilient to node failures, then link redundancy alone may not be adequate. Thus, generally speaking, positioning of the redundant links and the amount of redundancy must be approached in the context of the targeted performance measures. The rest of the dissertation is organized as follows.

Chapter 2 gives a comprehensive review of research relevant to hypercube networks. Concepts such as routing, fault tolerance, embedding, and mapping various algorithms for this topology are presented. Focusing on hypercube-based networks, a brief review of some of these networks is provided. The merits and drawbacks of each network are pointed out. Chapter 3 provides the necessary foundation for the material in subsequent chapters. This chapter includes the notation and terminology, network parameters, and basic routing algorithms for the hypercubes multiprocessors.

Chapter 4 is devoted to the development of a cube-based network with approximately halved diameter. The smallest amount of link redundancy to achieve a halved diameter cube as well as the positions of the additional links are determined. For the

resultant network, namely *Bridged Hypercube* or BHC, optimal one to one and broadcast algorithms are developed and proven correct. The figure of merits of BHC and its superiority over the conventional n-cube are addressed. Despite having a very small hardware overhead, BHC may not be attractive for applications requiring a symmetric network. This gives rise to motives for designing a symmetric network with a small diameter.

Chapter 5 introduces a symmetric cube-based network called the *Folded Hypercube* or FHC. Topological properties of the FHC are addressed and some interesting features of FHC are revealed. The routing algorithms for FHC are developed and shown to be 50% faster than those developed for the n–cube. Network parameters such as diameter, message traffic density, average distance, etc. are derived for the FHC and compared to parameters evaluated for the n–cube. The FHC can also be regarded as an standard hypercube augmented by some spare links. This network is referred to as the *Fault Tolerant Hypercube* (FTH). The fault tolerance capabilities of the FTH are compared to those of the standard cube. The chapter also includes the reliability evaluation for the n–cube and the FTH and employs an analytical model to derive the reliability measures for the above mentioned networks. In this analysis, it is shown how the extra links of FTH can be utilized as spares to improve the reconfigurability and fault tolerance of the standard hypercube. The thesis concludes in Chapter 6.

CHAPTER 2

Review of the Literature

The Hypercube has recently become a popular architecture for large-scale multiprocessing computers -intended for applications normally handled by supercomputers. More than 100 hypercubes are already in use, most of them in academic institutions and government laboratories. Researchers are learning how to use the technology, determine where it is applicable, and develop software tools to support programming.

Hypercubes run multiple programs that operate on multiple sets of data. Within the machine, the individual processing units, called nodes, are independent and communicate with each other while executing programs. In some cases, each node has its own memory, floating-point hardware, I/O processor, and copy of the operating system and application programs [19]-[22].

The topology is called hypercube because the architecture can be thought of as a cube of any dimension, with a node at each "corner". The higher the dimension, the more nodes there are. For example, a two-dimensional hypercube consists of four nodes connected by communication links to form a square. In a three-dimensional hypercube, eight nodes are connected to form a cube. Higher dimension (hyper) cubes are more difficult to visualize. In general, the number of nodes is always a power of 2, the exponent representing the hypercube dimension.

The hypercube topology offers a rich interconnection structure with large bandwidth, logarithmic diameter, and high degree of fault tolerance. Another appealing feature of the hypercube is its homogeneity and symmetry. In contrast with networks such as tree or shuffle-exchange, in a hypercube no node or edge plays a special role. This property facilitates algorithm design as well as programming. In essence, from a topological point of view, the hypercube provides a good balance between node

connectivity, network diameter, algorithm embeddability, and programming ease. This balance makes the hypercube suitable for a wide class of computationally extensive problems.

Based on various considerations of the foregoing kind, proposals to build large hypercube machines have been made for more than twenty years. In 1962, Squire and Palais at the University of Michigan, motivated by the hypercube's rich interconnection geometry and programming ease, carried out a detailed design of a hypercube computer [23],[24]. They estimated that a 12-dimensional (4096-node) version of their machine would require about 20 times as many components as the IBM Stretch, one of the largest and most complex computers of the time. Around 1975, IMS, an early manufacturer of personal computers, announced a 256-node commercial hypercube based on the Intel 8080 microprocessor, but its design details were not published and the machine was never produced. In 1977, Sullivan and Bashkow presented a thorough analysis of hypercube architectures, and a proposal to build a large hypercube called CHOPP (Columbia Homogeneous Parallel Processor) containing up to a million processors [25],[26]. In the same year, Pease published a study of the "indirect" binary n-cube architecture, in which a multistage interconnection network of the omega type is suggested for implementing the hypercube topology [27].

It is clear that the early hypercube designs were impractical because of the large number of components (logic and memory elements) they required using the available circuit technology at their respective times. The situation began to change rapidly in the early 1980's as advances in VLSI technology allowed powerful 16/32-bit microprocessors to be implemented on a single IC chip, and RAM densities moved into the 10^5 -10^6 bits/chip range. A working hypercube computer was not demonstrated until the completion of the first 64-node hypercube, Cosmic Cube, at Caltech in 1983 [19]. As the hypercube node processor, Cosmic Cube employed a single-board microcomputer containing the Intel 8086 16-bit microprocessor and the 8087 floating-point coproces-

sor. Since then, Caltech researchers have built several hypercube machines, and successfully applied them to numerous scientific applications often obtaining impressive performance improvement over SISD machines of comparable cost [28].

Influenced primarily by the Caltech work, a number of commercial hypercubes have been developed since 1983. In July 1985, Intel delivered its first hypercube, the 128-node iPSC (Intel Personal Supercomputer) which had a 16-bit 80286/287 CPU as its node processor. Assuming a performance of 0.07 MFLOPS per node, the 128-node iPSC has a potential throughput of about 8 MFLOPS, far below that of traditional supercomputers such as the Cray-1 (160 MFLOPS) [29]. Other commercial supercomputers were also introduced in 1985 by Ametek Inc. [30], and NCUBE Corp. [31]. The Ametek System/14 hypercube can have up to 256 nodes, which employ an 80286/287-based CPU similar to that of the iPSC, with the addition of an 80186 processor for communication management. Even though the number of node processors in System/14 is twice that of iPSC, the maximum memory per node in System/14 is 256 Kbytes which is only half that of iPSC. The NCUBE/10 can accommodate up to 1024 nodes, each based on a VAX-like 32-bit processor with a peak performance of 0.5 MFLOPS. Thus, a fully configured NCUBE system has a throughput potential of around 500 MFLOPS. This high performance level is supported by extremely fast communication rates making the NCUBE/10 a true supercomputer. NCUBE machines have been installed at several test sites, including the University of Michigan since early 1985, and have been in general production since December 1985.

Several other hypercube machines with supercomputing potential have been developed, mostly in 1986, including the iPSC-VX by Intel, T-series by Floating Point Systems, Connection Machine by Thinking Machines Corp. [32], and Mark III by Jet Propulsion Laboratory at Caltech [22]. All these machines, but the Connection Machine, employ a vector processor to boost node performance by an order of magnitude or more. The iPSC-VX consists of 64 nodes with a maximum memory per node

of 1.5 Mbytes. T-series is a modified hypercube comprising of 16384 node processors each containing one Mbyte memory. The Connection Machine contains 64K processors, distributed evenly among 4K chips interconnected as a 12-dimensional hypercube. The memory size at each node is 500 Kbytes yielding an overall memory size of around 32 Gbytes. Mark III was developed by a joint JPL-Caltech research team. Mark III has a 68020/68081 node processor and a vector processor of the company's own design. It has 1024 processors each containing 4 Mbytes memory.

The trends in hypercube technology have been toward faster node processors, bigger memories on each node, and higher dimension. Single-board vector processors, which boost the arithmetic performance of each node in a system, have also found wider use. Such coprocessors offer high performance in repetitive calculations on ordered sets of numbers, such as matrices and vectors. Expanding the memory capacity of each node greatly increases efficiency, particularly for medium - to large problems. With more memory, more data can be stored at each node, and more computations can be performed by the node without exchanging results with other nodes. As memory chips cost less and less, nodes with 10's of Mbytes will become available. The hypercube architecture can fairly take advantage of increased VLSI performance.

Hypercubes have been extensively analyzed by many theoreticians. The fundamental properties of the hypercube have been studied in [33]. In that, many appealing features of this topology including ease of routing, high connectivity, fault resilience, node and edge symmetry have been revealed. It was found that a wide class of parallel algorithms can either run directly on the hypercube, or can run on topologies embeddable in the hypercube. Since 1985, data communication in hypercubes has received significant attention [33]-[38]. Various algorithms for moving data from one node to another node, one node to every other node (broadcasting), and every node to every other node in the hypercube were developed. Ho and Johnsson [38] provided the most comprehensive study in this regard. This study is based on some new spanning trees

which govern the routings, namely broadcasting and personalized communication, in the structure. It is shown that using these spanning trees, communication links in the hypercube can be utilized efficiently resulting in some speedup over the algorithms developed in the past. The analysis takes into account the size of data sets, the communication bandwidth, and the overhead in communication. Sending data from one node to several nodes (i.e. multicast) is not as straightforward as the other routings. Finding an optimal algorithm for multicasting in hypercubes is conjectured to be NP-hard, and only some heuristic greedy algorithms have been proposed for this problem [39].

Embedding various geometries in the hypercube has received much attention. Tree embedding has been addressed by many researchers [33],[40]-[42]. Saad and Shultz [33] showed that it is impossible to embed an n-level complete binary tree ($n \geq 3$), consisting of $2^n - 1$ nodes, into a subgraph obtained by removing one of the nodes of the n-cube. Independently, Deshpande and Jenevein [40], and Bhatt and Ipsen [41] arrived at the same result. They showed, however, that by stretching the edge between the root and one of its sons, it is indeed possible to map an n-level tree onto a hypercube of degree n. In this mapping, the extra node of the cube is used solely for communication between the root and one of its sons. Wu [42] has discussed the embedding of k-ary trees.

The above studies showed that when embedding trees (or graphs in general) into a hypercube, there is a tradeoff between the utilization of the nodes of the hypercube and the maximum edge length of the embedded (guest) graph. Chan and Shin [43] introduced two parameters, namely *expansion* and *dilation*, to measure the efficiency of an embedding scheme. Expansion is the ratio of the size (number of nodes) of the embedding cube to the size of the smallest cube having enough nodes to embed the guest graph. Dilation is defined to be the maximum edge length of the guest graph on the host graph. Thus, embedding complete binary trees into the hypercube can be done

with unit expansion and dilation 2, or unit dilation and expansion 2. The mapping of incomplete binary trees is treated in [41], where it is found that the embedding of any incomplete binary tree of size N into the hypercube can be done with expansion 2 and dilation $\log \log N + O(1)$. Nevertheless, the question "can every binary tree be embedded in a hypercube with dilation 2 and unit expansion?" still remains open.

The embedding of other geometries such as linear arrays, loops, 2-D and multidimensional meshes have also been the focus of many researchers. The common core in all embeddings is the concept of *Binary Reflected Gray Codes* or BRGC [44]. The Gray Codes are successive binary numbers differing in precisely one bit. This coding allows embedding of linear arrays of any size less than 2^n into a hypercube of order n [33]. Johnsson [45] showed that any loop of length l can be embedded into an n–cube so long as l is even and $4 \le l \le 2^n$ ($n \ge 2$). The constraints on l arise since in any hypercube of order 2 or more, there is no odd cycle or cycle of length less than 4.

Compared to embedding trees, the embedding of meshes has been considered to a lesser extent. As in embedding linear arrays and loops, attempts to embed the mesh structure on the hypercube have been based on BRGC [33],[45]. In these works, it is shown that any $m_1 \times m_2 \times .. m_d$ mesh in the d–dimensional space can be mapped into an n–cube where $n = p_1 + p_2 + .. + p_d$ and $p_i \equiv \log_2(m_i)$ for $1 \le i \le d$. Although such embeddings guarantee unit dilation, expansion in the worst case is 2^{k-1} for k–dimensional meshes. A recent result shows that this expansion is best if the unit dilation is insisted upon [46],[47]. Chan and Chin [43] relaxed the constraint of unit dilation in order to improve the expansion. They gave an embedding scheme for an infinite class of 2-D meshes with dilation two and minimum expansion. The expansion obtained for most of such 2-D meshes is one. However, just as in tree embedding, it is not known yet if there is a scheme to embed 2-D meshes into a hypercube with dilation two and unit expansion. Also, worth mentioning, is the work by Ho and Johnsson [48] who have

proposed a method for embedding 2-D meshes which yield dilation two and minimum expansion.

In addition to embedding some popular topologies in the hypercube, this architecture can be employed to simulate (i.e. perform) the functions of networks such as PM2I, Illiac, and Shuffle-Exchange with little or no hardware overhead [49]. In this case, the hypercube is modeled as an SIMD machine. It was found that an n–cube can simulate any of the above networks in at most n steps. The overhead for the simulation includes masks to activate particular groups of node processors for data transfer, and data registers to preserve the data in intermediary nodes.

As mentioned earlier, the hypercube can lend itself to the execution of parallel algorithms which realize the rich interconnections inherent to this structure. Many efforts, most notably at Yale University, have been launched to develop efficient algorithms which can run directly on the hypercube [35],[50]-[52],[53]-[61]. Johnsson [35] developed a few algorithms for matrix transposition, matrix multiplication, matrix factorization, and solving triangular linear systems using Gauss-Jordan elimination. These algorithms are mainly based on two storage schemes (i.e. data structures), namely *cyclic* and *consecutive* and BRGC encoding of rows and columns of the matrices. The multiplication of two arbitrary shaped matrices is treated in [50]. In that, it is shown how the choice of algorithm with respect to the total execution time (arithmetic and communication) depends on the shapes of the matrices, and the number of processors relative to the size of matrices. Several other algorithms, targeted at the hypercubes, can be found in the literature for matrix multiplication [36],[51],[52], matrix transposition [53], the solution of sparse systems of linear equations [54], and the Fast Fourier Transforms [55],[56].

Algorithms for stable dimension permutations have been proposed by Ho and Johnsson [57]. A *stable dimension permutation* is specified by a permutation function defined on the bits of the address field of node processors in the n–cube. The

k-shuffle permutation, the bit reversal permutation, and matrix transposition are special cases of stable dimension permutations. In [57], lower bounds for this problem were found both for one-port and n-port communications. Later, this problem was extended to the *generalized shuffle permutations* where the permutation is performed on the address field (rather than the bits of the address field), and few algorithms were developed for this problem [58].

In addition to linear algebraic algorithms, some graph algorithms have been suggested for execution on the hypercube. These algorithms include Maze routing [59], all pairs shortest paths [60], and dictionary machines [61], to name a few.

Most of the research effort on hypercube architecture has focused on the fault free situation. Some researchers have investigated the fault tolerance of the hypercube [62]-[63]. They have shown that this topology is extremely fault resilient, and its performance under highly faulty conditions will not degrade drastically, so long as the network diameter is selected as a criterion for fault tolerance. Becker and Simon [64] examined the problem of finding the minimum number $f(n,k)$ of faulty components, necessary for an adversary to make any $(n-k)$-cube faulty †. Since there exist many ways to choose a faulty set F with $|F| = f(n,k)$, in the analysis they allowed an "intelligent" adversary to distribute the faulty components in the worst case fashion. However, this work does not give a precise characterization of hypercube fault tolerance since the worst-case situation considered in the analysis is very unlikely to occur. Besides, for $k>2$ only bounds on $f(n,k)$ are given which are not very tight. An attempt to characterize the reliability of hypercube structures as network components begin to fail was launched by Abraham and Padmanabhan [65]. Their study showed that even though the n-cube is destroyed by the very first component failure, the cube is quite resilient in terms of its ability to support several smaller subcubes in the

† Reversely formulated: The existence of an $(n-k)$ dimensional healthy cube is guaranteed, unless there are at least $f(n,k)$ faults in the n-cube.

damaged structure.

Enhancement of the fault tolerance of the hypercube has also been the subject of a few studies [66],[67]. In [66], approaches for implementing concurrent fault-detection in the processing nodes are examined. It is shown that a hypercube structure can be decomposed hierarchically, and redundancy can be employed at several levels so that it can be utilized efficiently. Sparing is strictly local and similar to those suggested in [68] for 3-D VLSI arrays. In [67], some communication links are added to the structure to enhance the communication reliability of the network. This approach was shown to improve factors such as *connectivity* and *Two-Terminal* reliability in the network.

Fault tolerance routing/broadcasting has been addressed in the literature [69],[70]. In these works, routing is achieved by going around the faulty node. This approach can be used only if it is possible to identify the faulty nodes "on-line". In another approach, fault tolerance is achieved by sending multiple copies of the message through *disjoint* paths. The nodes that receive the message identify the original message from the multiple copies using majority voting [71]. This approach has the advantage of not having to identify the faulty nodes during the normal operation of the system. This advantage is especially important in critical real-time applications. A different method for reliable broadcasting in the hypercube has been recently proposed [72]. In that, the broadcast algorithm is performed by sending multiple copies of the message via disjoint paths to other nodes. The salient feature of this algorithm is that the delivery of multiple copies is transparent to the processors receiving the message and does not require the processors to know the identity of the faulty nodes. The algorithm can tolerate $O(n)$ faulty nodes/links and has the time complexity of $O(n)$.

2.1 Hypercube-based Networks

A number of other interesting architectures closely related to the hypercubes have been proposed in the literature [73]-[77]. As in hypercubes, these structures are

constructed in a multi-dimensional medium and allow message delays to increase as slowly as $\log N$, where N is the number of node processors. Each network has been characterized by determining several key factors in the design of interconnection networks. These factors include degree of the node (δ), diameter (d), number of communication links (L), average inter-node distance ($\bar{d}$), and message traffic density (ρ) [74],[75]. Additional factors in comparing topologies include ease of routing messages between nodes, communication reliability, and ease of laying out the design onto a 2-D plane for VLSI implementation. These factors together indicate the suitability of each topology for parallel processing applications. In the following, some of these topologies are addressed and comparatively analyzed.

2.1.1 Nearest Neighbor Mesh

The regular nearest neighbor mesh consists of W^D nodes placed in D dimensions with W nodes per dimension. Each node is connected to two neighbors in each dimension. This architecture was used for the slave processing element connections on the SIMD Illiac IV [77]. The familiar 2-D mesh array consisting of $N=W^2$ with end around connections is an example of this family of networks. Figure 2.1 illustrates a 2-D mesh with $W=4$. Wittie [74] has analyzed this architecture and determined such factors as diameter, average node distance, and message traffic density. He has discussed the scalability of the mesh connected networks in detail. Essentially, there are two ways in which network size ($N=W^D$) may increase:

1) dimension D may be fixed and width W may grow as $N^{1/D}$ or

2) width W may be fixed and D may increase as $\log_W N$.

Each extreme has its advantages and disadvantages. If W grows as $N^{1/D}$, the number of connections per node is constant. The same module can be used for each node regardless of network size. However, diameter, average distance, and traffic density increase relatively rapidly, as the D th root of N.

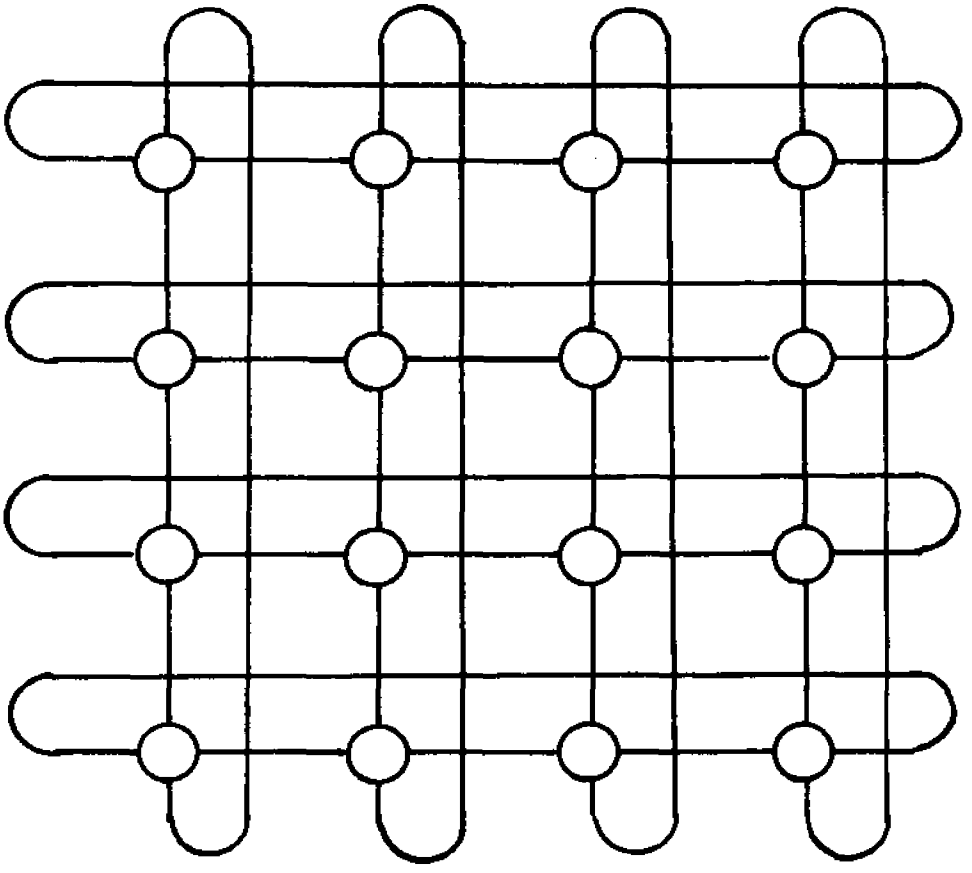


Figure 2.1. Nearest Neighbor Mesh ($N=4^2$).

If W is fixed, as N increases, (i.e. the W–ary cube where $W>2$), both diameter and average distance increase only as slowly as $\log_W N$, whereas the average traffic density remains constant. However, connections cost grows as $N\log_W N$ and, in particular, the number of connections per node ($=2D$) grows as $\log_W N$. Slow growth ($\log N$) in average distance is very desirable. With $\log N$ growth in average distance in the network, allocation of tasks to nodes is easier since the nearness of communicating nodes is guaranteed, even in very large networks.

However, a $\log N$ increase in connections per node means that fixed width, expanding dimension hypercube meshes cannot be arbitrarily extended if standard modules are used for nodes. A standard node would have to include the $2D$ local ports for the largest possible dimension D. After D is reached, N can be increased only by changing W and causing an increase in average distance and message density.

2.1.2 Cube-Connected Cycles (CCC)

The n–cube is not very suitable for VLSI implementation, since each of the 2^n nodes in the system is connected to n other processors. In 1981, Preparata and Vuillemin [73] proposed a cube-based network with an efficient VLSI layout. The CCC is a standard cube where each node in the hypercube is replaced by a set of PE's interconnected as a loop. This network is proven to be a feasible substitute for the cube networks in many applications since it can lend itself to efficiently solving a large class of problems including FFT, sorting, and permutations [73]. A CCC of dimension D consists of $N=D\times 2^D$ nodes for some integer D. A CCC with $D=3$ is shown in Figure 2.2. In a CCC, each node is connected to exactly three links: two links to neighbors on the same cycle (cycle-links), and one link crossing the hypercube in one of the n dimensions to the corresponding node in another cycle (cross-link).

Although the low degree of the node makes CCC implementable on a VLSI chip, it reduces the fault tolerance capabilities of the network since there exist only three

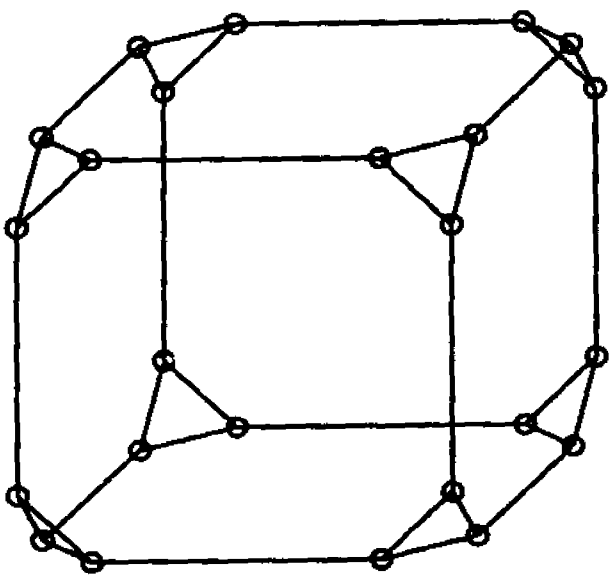


Figure 2.2. Cube-Connected Cycles ($N=3\times2^3$).

disjoint paths between any pair of nodes in the network.

2.1.3 Arbitrary Hypercubes

A standard D–cube can be visualized as a structure of dimension D and base 2 consisting of 2^D nodes. Arbitrary hypercubes are extensions to the standard hypercube where there exist W nodes at each dimension and thus W^D nodes in the structure ($W>2$). Bhat has shown that this class of networks possess attractive graph properties [78]. The main difference between the mesh and arbitrary cube lies in the number of neighbors of each node. While in the mesh each node is connected to only two neighbors at each dimension, in an arbitrary cube each node is connected to all $W-1$ nodes residing at the same dimension.

As in the mesh, there are two ways to increase the size of the network. If W is fixed and D grows, the diameter, the degree of the node, and the average distance grow proportional to $\log_W N$. The number of links increases as $N \log_W N$. The message density stays the same regardless of the network size. Keeping the dimension constant and increasing the base W will result in an increase in degree of the node proportional to the Dth root of N. Nevertheless, as N grows, the diameter and the average distance remain the same as before. When W grows, the number of links in the network increases very rapidly at the rate of $N \times N^{1/D}$ resulting in less message traffic density in the system.

2.1.4 Generalized Hypercubes

This class of cube-based networks was proposed by Bhuyan and Agrawal [75] in 1984. The interconnections are based on a mixed radix number system and the technique results in a variety of cube structures for a given number of processors N, depending on the desired network diameter. Let N be represented as a product of r_i's, $r_i>1$ for $1 \le i \le n$ or $N=\prod_{i=1}^{n} r_i$, i.e. there are r_i nodes at dimension i. Figure 2.3 shows a

generalized hypercube where $N = 4\times3\times2$. The class of arbitrary hypercubes is a subset of this class where $r_i=r$ for all i's. The generalized hypercubes support a wide range of N, the number of processors in the network. This is in sharp contrast with the arbitrary cube where $N=r^n$ for some integers r and n. Since N may be factored in several ways, for constant number of processors more than one network can be obtained.

In generalized hypercubes, there is a tradeoff between the average distance and the diameter on one hand, and the the degree of the node and connections cost on the other hand. Therefore, the designer has some freedom in choosing a network with desired network parameters.

The generalized hypercube has a large degree of the node and hence too many links. The large number of links available in this structure adds to the complexity of the structure, but improves parameters such as connectivity, average distance, and message traffic density. On the other hand, for an arbitrary N, it may not be possible to determine r_i's and n such that the resultant network has the desired performance parameters.

Note that unless there is a high message traffic in the network, the connections cost may not be justified. A cost optimal (minimal number of links) GHC with fixed diameter "r" is obtained if $r_i=N^{1/n}$ for all $1\le i\le n$ [75]. Therefore, arbitrary hypercubes are in fact cost optimal GHC's where the cost is directly related to the number of communication links in the network. If the cost factor ξ is defined as the product of the diameter and the links per node, a discrete optimization of $n\sum_{i=1}^{n}(r_i-1)$ with respect to r and subject to the constraint that $\prod_{i=1}^{n} r_i = N$ and integer values of r_i's, yields an optimal structure. It is conjectured [55] that for N being a power of 2, an absolute cost (ξ) optimal GHC is obtained when $n=\lfloor \log_4 N \rfloor$. This optimal GHC is called OGHC.

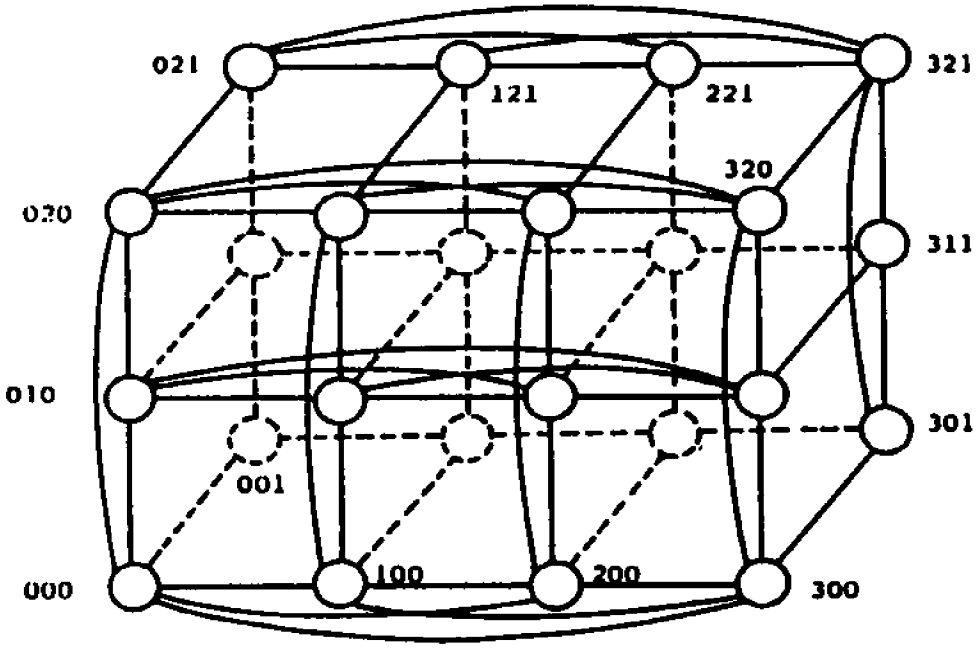


Figure 2.3. Generalized Hypercube (N=4×3×2).
(reproduced from [75])

There are some other variations of the standard hypercube in the literature where the improvement in network performance is either not significant or offset by some undesirable byproducts due to the change. For instance, recently Esfahanian et al [79] introduces the *Twisted Cube* which can be obtained from an n–cube by exchanging two parallel links in any cycle of four nodes by two crossed links (Figure 2.4). As the result of this link swapping, the diameter of the n–cube is reduced to $n-1$. Also, an n–level binary tree can be mapped in the structure with unit dilation. The Twisted cube is clearly not symmetric and may not be suitable for some parallel processing applications.

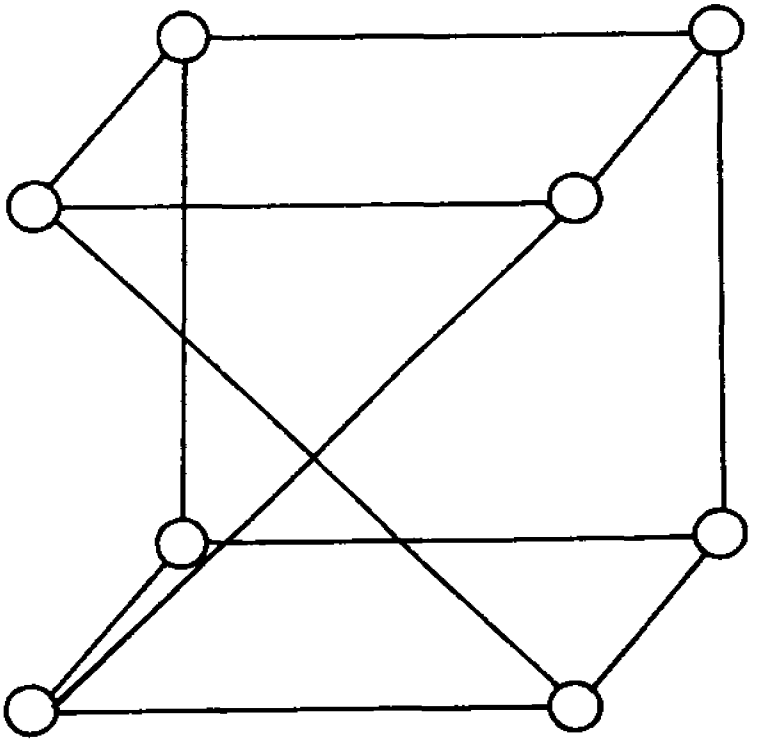


Figure 2.4. Twisted 3-cube ($N=2^3$).

In all networks considered above, there exist dedicated, point to point, passive links between every pair of nodes. There is a rich literature, however, pertaining to cube-based networks which use shared busses for processor interconnections such as: Spanning Bus Hypercube and Dual-Bus Hypercube [74], and Spanning Multiaccess channel hypercubes [76]. The attractive features of such structures are, mainly, low degree of the node and low diameter. The applications for such networks are restricted, however, by the limited bandwidth of the single bus spanning several nodes.

2.2 Performance Comparison

The performance measures for the five structures discussed above are shown in Table 2.1. Arbitrary hypercubes offer the lowest diameter for $W>4$. However, this attractive feature is offset by high connections cost and degree of the node. Note that the OGHC is a special case of the arbitrary cube where $W=4$. The CCC has a large diameter and average distance, due to the low degree of the node and low number of links in the network. Observe that the average message density is constant for the n-cube and the OGHC, since the average distance and the network size grow at the same rate as the number of links. In a mesh, the average message density is proportional to W, whereas in the arbitrary cube this factor is proportional to $1/W$. This factor is linearly proportional to D in CCC. Table 2.2 gives the parameters for various networks of size 64. Since OGHC is the same as the arbitrary cube in this example, another version of a four dimensional GHC is considered. As can be seen, the arbitrary cube offers the lowest diameter and largest number of links. While CCC enjoys possessing the lowest degree of the node and number of links, it has a large diameter and average traffic density making it not so desirable under heavy load conditions.

The idea of the work presented in this thesis is to investigate new cube-based networks which can offer better overall performance than those offered by the aforementioned networks. A network with a diameter of an arbitrary cube and a constant degree of the node of a CCC may represent a desirable target structure.

Table 2.1. Performance Measures of some Hypercube-Based Networks.

Structure	N	d	δ	L	$\bar{d}$	ρ
n−cube	2^n	$\log_2 N$	$\log_2 N$	$0.5\ N\ \log_2 N$	$0.5\ \log_2 N$	1.0
Mesh	W^D	$0.5\ W\ D$	$2\ D$	$D\ N$	$0.25\ W\ D$	$0.25\ W$
Arb. cube	W^D	D	$D\ (W-1)$	$0.5\ D\ (W-1)\ N$	$D\ (W-1)\ /\ W$	$2/W$
OGHC	4^m	$0.5\ \log_2 N$	$1.5\ \log_2 N$	$0.75\ N\ \log_2 N$	$0.375\ \log_2 N$	0.5
CCC	$D \times 2^D$	$2.5\ D - 1$	3	$1.5\ D$	$7\ D\ /\ 4$	$5\ D\ /\ 4$

Table 2.2. Performance Measures of 64-Node Hypercube-Based Networks.

Structure	N	d	δ	L	$\bar{d}$	ρ
n−cube	2^6	6	6	192	3	1.0
Mesh	4^3	6	8	256	3	1.0
Arb.cube	4^3	3	9	288	2.25	0.5
OGHC	4^3	3	9	288	2.25	0.5
GHC	$2^2 \times 4^2$	4	8	256	2.54	0.5
CCC	4×2^4	9	3	96	7	5

CHAPTER 3

Notation and Background

The purpose of this chapter is to provide the necessary foundation for the material in subsequent chapters. The notation and terminology, adopted throughout the remainder of this dissertation, are introduced. A graph theoretical model of the n-cube is reviewed, and significant properties of this topology are pointed out. The chapter also includes one to one and broadcast algorithms for the hypercube multiprocessors. In addition, network parameters including those discussed in Chapter 2 are examined more closely for hypercube networks. Aspects of fault tolerance in hypercubes such as *connectivity, two-terminal reliability, f-fault diameter, and container* are also included. Such performance measures provide appropriate reference for judging the merits (or demerits) of new hypercube-based architectures.

3.1 Graph Theoretical Model of the n-cube

Let $G(V,E)$ be a finite graph without loops or parallel edges, with the *node set* $V(G)=V$ and the *edge set* $E(G)=E$. When G is known from the context, the sets $V(G)$ and $E(G)$ will be referred to by V and E, respectively. If an edge $e=(u,v)\in E$, then the nodes u and v are said to be *adjacent* and the edge e is said to be *incident* on these nodes. The *degree*, $deg_G(v)$ of a node $v\in V$ is equal to the number of edges in G which are incident on v. A *tree* is a connected graph which contains no *cycles*. A graph $G(V,E)$ is a *subgraph* of another graph $F(V,E)$, if $V(G)\subseteq V(F)$ and $E(G)\subseteq E(F)$. When $V(G)=V(F)$, then G is called a *spanning* subgraph F. A subgraph of a tree is referred to as a *subtree*. The *distance* $d_G(u,v)$, between a pair of nodes u and v in G, is equal to the length (in number of edges) of the *shortest* path joining u and v. The *diameter* of G is the maximum distance between two nodes in

G over all pairs of nodes in V.

An n–dimensional hypercube (i.e. n–cube) can be modeled as a graph $G_n(V,E)$, with $|V| = N = 2^n$, $|E| = n2^{n-1}$. Each node represents a processor and each edge represents a link between a pair of processors. Nodes are assigned binary numbers from 0 to 2^n-1 such that labels of any two neighbors differ only in one bit position. Links are also labeled from 0 to $n-1$ such that link i connects two nodes whose labels differ in the ith bit. An n–cube denoted by Q_n for $n \geq 2$ can be defined recursively in terms of the graph product operation $\times$ as follows[80], where $K_2=Q_1$ is the complete 2-node graph:

$$Q_n = K_2 \times Q_{n-1}$$

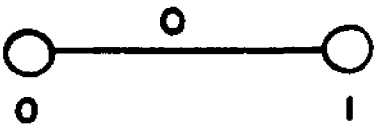
Figure 3.1 illustrates the hypercube topology for n <4. As illustrated in the figure, Q_n is composed of two copies of Q_{n-1}. Also, observe that there are n ways to partition a Q_n into two Q_{n-1}'s.

For each node $u \in V(G_n)$, let $a(u)$ denote the binary label of this node. The *Hamming weight* of a node u is defined to be the number of 1's in $a(u)$ and is denoted by $\|a(u)\|$. The class of nodes of Hamming weight i is denoted by W_i. Also, let $\oplus$ denote the *bitwise exclusive or (XOR)* operation on binary numbers. Then, $e=(u,v) \in E(G_n)$, iff $\|a(u) \oplus a(v)\|=1$. This implies that $deg_{G_n}(u)=n$ for every node $u \in V(G_n)$. Also, $d_{G_n}(u,v)=\|a(u) \oplus a(v)\|$, for every pair of nodes $u,v \in V(G_n)$.

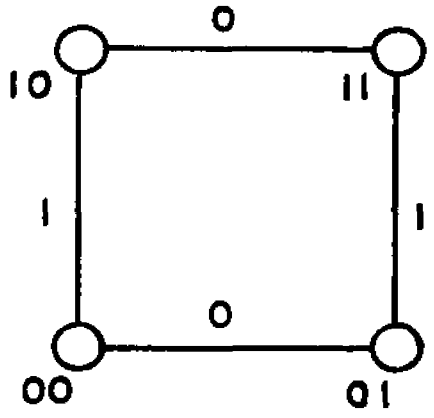
3.2 Topological properties of the n-cube [33]

A binary n–cube has $N = 2^n$ nodes, diameter n, $\binom{n}{i}$ nodes at distance i from a given node, and n node disjoint paths between any pair of nodes. The paths are either of the same length as the Hamming distance between the end points of the paths, or the Hamming distance plus two. The degree of every node is n and the total number

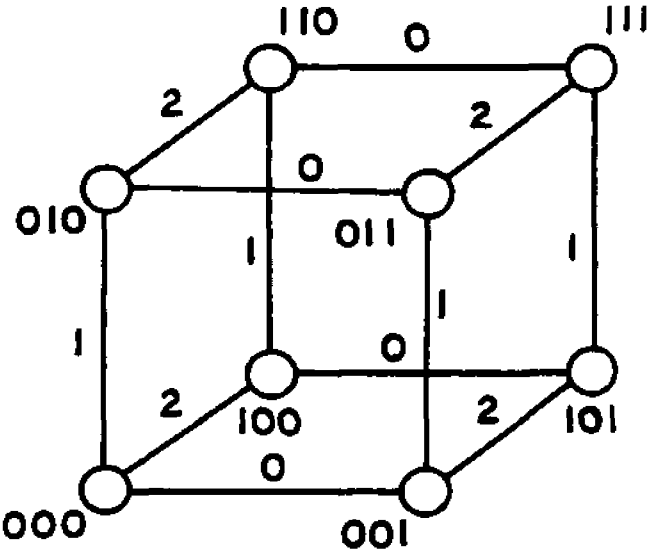
0-cube



1-cube



2-cube



3-cube

Figure 3.1. The n –dimensional Hyperucbe Topology ($n \leq 3$).

of communication links at each dimension is $N/2$ yielding a total of $nN/2$ bidirectional links in the structure. The n–cube is also vertex and edge symmetric. More formally, given any pair of nodes u and v in the cube, there can be found an automorphism which maps u to v.

3.3 Routing algorithms in the n-cube

3.3.1 One to One Routing

If a message is to be sent from node u to a neighboring node v such that $a(u)$ and $a(v)$ differ only in the ith bit, link i must be traversed. Alternatively, traversing link i from a node u is referred to as *correcting* the ith bit in $a(u)$. If two nodes have a Hamming distance of i, i bits in the source label must be corrected. These i bits can be corrected in any order. This implies that there exist $i!$ distinct paths of length i between two nodes of Hamming distance i †. The following algorithm performs the routing between any pair of nodes, namely u and v, in the n–cube [34].

One to One Algorithm:

input: $a(u)$, $a(v)$;

Begin

$a(w) = a(u) \oplus a(v)$

Step 1: Route the message sent from u via a path composed of links corresponding to non zero bit positions in $a(w)$, in any order.

End.

† These paths are not always node/link disjoint, however.

3.3.2 Broadcasting (One to All Communication)

Broadcasting data from a single source to all other nodes in a multiprocessor system is an important operation. It is used in many parallel algorithms such as matrix multiplication, the solution of irreducible linear systems, and forming transitive closure [38].

In the n-cube, any spanning tree can be used to broadcast data from a single source to all other nodes (such as a Hamiltonian path). A node replicates the data as many times as the out-degree of the node in the spanning tree. In broadcasting one element (or packet), the minimum number of routing steps † is n. Any spanning tree with height n can achieve this lower bound, if each node can send out data through all the links connected to it during one step. In broadcasting M elements using a packet size of B elements, and by pipelining the communication from the root toward the leaves along any spanning tree of height n, the number of routing steps becomes $\lceil M/B \rceil + n - 1$, which is not optimal [38]. Since each node has a fanout of n, the lower bound for the number of routing steps is $\lceil M/Bn \rceil + n - 1$. In order to achieve this lower bound, the data set has to be split into n subsets, each of which is communicated over a distinct communications link from the source node. It follows that the nodes adjacent to the source node must be roots of subtrees spanning all but one node of the cube (the source node). The length of the subtrees is n, and the lower bound for the number of routing steps, assuming concurrent bi-directional communication, is $\lceil M/Bn \rceil + n$ [38]. Next, routing algorithms for broadcasting based on a Spanning Binomial Tree (SBT) and Multiple Spanning Binomial Tree (MSBT) are reviewed. First, the topologies of SBT and MSBT are defined as in [38]. Then, the communication complexity of each topology is reviewed.

---

† Time-wise

3.3.2.1 Spanning Binomial Tree (SBT)

The problem of broadcasting in an n-cube from a given node s is one of finding a spanning tree in G rooted at s. The spanning tree rooted at node 0 of an n-cube contains the edges that connect a node u with the subset of its neighbors having addresses obtained by complementing any bit of leading zeroes of the binary encoding of u [33]-[38]. For clarity, the algorithm in [38] is briefly reviewed, adopting the same notations with minor modification. For an arbitrary source node s, the spanning tree is simply translated by a *bitwise XOR* operation on all addresses with the address of the source node, i.e. $a(v)=a(u) \oplus a(s)$ is formed. Complementation of those bits of $a(u)$ that correspond to the leading zeroes of $a(v)$ defines the edges of the translated spanning tree. More precisely, let $a(s)=(s_{n-1}s_{n-2}..s_0)$, $a(u)=(u_{n-1}u_{n-2}..u_0)$, and $a(v)=(v_{n-1}v_{n-2}..v_0)$, where $v_m=s_m \oplus u_m$. Let $v_k=1$ and $v_m=0$, $\forall\ m>k$ with $k=-1$ for $a(v)=0^n$ †. More specifically, k is the highest order bit position of $a(v)$ that is 1. Let *children(u,s)* be the set of child nodes of node u in the spanning binary tree SBT rooted at node s and define the set $M_{SBT}(u,s) \equiv \{n-1,..,k+1\}$. Then,

$$children_{SBT}(u, s) = \{(u_{n-1}u_{n-2}..\overline{u}_m..u_0)\},$$
$$\forall\ m \in M_{SBT}(u,s)$$

Let *parent (u,s)* be the parent of node u in the spanning tree rooted at node s. Then it follows that,

$$parent_{SBT}(u,s) = \begin{cases} \phi, & u=s; \\ (u_{n-1}u_{n-2}\ ..\ \overline{u}_k..u_0), & u\neq s. \end{cases}$$

It can be seen that children and parent functions are consistent. Figure 3.2 shows a spanning tree generated by the children (or parent) function in a 4-cube.

---

† Hereafter, the superscript in a binary notation is a repetition factor.

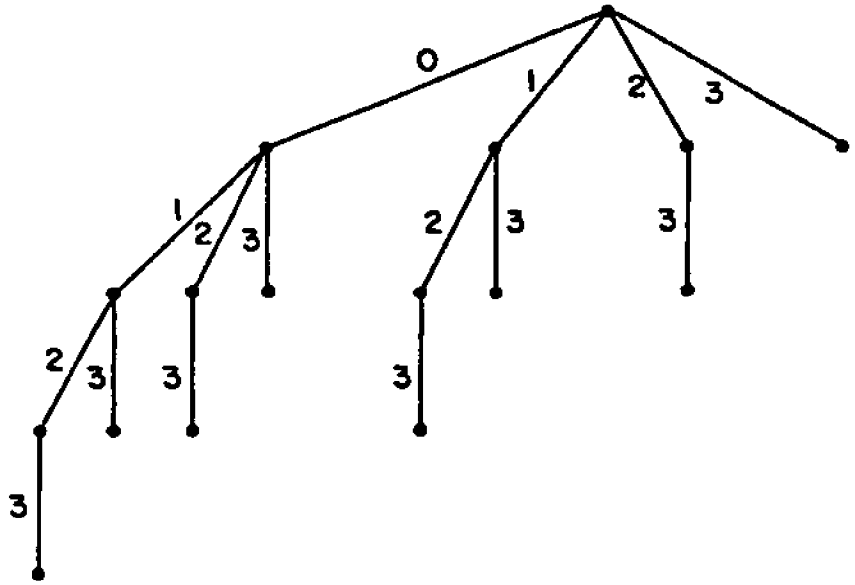


Figure 3.2. Spanning Binary Tree (SBT) for a 4-cube.

3.3.2.2 Multiple Spanning Binomial Tree [38]

The Multiple Spanning Binomial Tree (MSBT) graph can be viewed as being composed of n SBT's with one tree rooted at each of the nodes adjacent to the source node. The SBT's are *rotated* such that the source node of the MSBT graph is in the smallest subtree of each SBT. The MSBT graph is then obtained by reversing the edges from the roots of the SBT's to the source node . After the edge reversal each SBT becomes an ERSBT (*Edge Reversed Binomial Tree*). The diameter of MSBT graph is $n+1$, since the source node is adjacent to all the roots of the SBT's used in the definition of the MSBT graph, and each SBT is of height n. The total number of edges in the n SBT's is $(N-1)n$, which is n less than the total number of directed edges from the roots of the SBT's toward the source node. The SBT's used for the construction of the MSBT graph can be obtained by translation and rotation of the SBT defined before. The SBT rooted at node $0^{n-j-1}10^{j}$ is referred to as the j^{th} SBT of the MSBT graph. The j^{th} ERSBT is obtained from the j^{th} SBT by reversing the edge directed to node 0^n (i.e. the source). Let $a(u)=(u_{n-1}..u_0)$, and k be such that $a_k=1$ and $a_m=0$, $\forall\ m\in M_{MSBT}(i,j)$ where $M_{MSBT}(i,j)=\{(k+1)mod\ n,(k+2)mod\ n,$ $..,(j-1)mod\ n\}$. For $a(u)=0^n$, define $k=-1$. For the jth ERSBT of the *MSBT* graph with source node 0, the set of child nodes of node u is defined as follows.

$$children_{MSBT}\ (u,j,0) = \begin{cases} (u_{n-1}..\overline{u}_j..u_0), & \text{if } k=-1; \\ \{(u_{n-1}..\overline{u}_m..u_0)\}, & \\ \forall\ m\in M_{MSBT}(i,j), & \text{if } u_j=1,\ k=j; \\ \{(u_{n-1}..\overline{u}_m..u_0)\}, & \\ \forall\ m\in M_{MSBT}(i,j)\bigcup j, & \text{if } u_j=1,\ k\neq j; \\ \phi, & \text{if } u_j=0,\ k\neq -1. \end{cases}$$

$$parent_{MSBT}(u,j,0) = \begin{cases} \phi, & \text{if } k=-1; \\ (u_{n-1}..\overline{u}_j..u_0), & \text{if } u_j=0,\ k\neq -1; \\ (u_{n-1}..\overline{u}_k..u_0), & \text{if } u_j=1. \end{cases}$$

Figure 3.3 shows an MSBT graph with source node 0 in a 3-cube.

It has been shown [38] that the n directed ERSBT's are edge-disjoint and the height of the MSBT graph is minimal among all possible configurations of n edge-disjoint spanning trees.

As with SBT, for an arbitrary source node s, an MSBT graph can be defined by translating the MSBT graph rooted at node 0. This can be done by taking the XOR of all labels in the MSBT graph rooted at node 0 with the label of s.

3.3.3 Communication Complexity of SBT- and MSBT-based Broadcasting

3.3.3.1 Spanning Binomial Tree

Assuming all ports can send or receive a message concurrently, pipelining can be employed to speed up the broadcasting. The number of steps for the message to arrive at the node farthest away from the source is at least n. When this node has received all packets, the broadcasting is terminated. Hence, $\lceil M/B \rceil + n - 1$ steps are needed to perform the broadcasting [38].

3.3.3.2 Multiple Spanning Binomial Tree

If communication is allowed to take place on all ports concurrently, the packets can be split to n and sent to different subtrees in the MSBT graph. Since the height of MSBT is $n+1$, by employing pipelining it follows that the number of steps to perform broadcasting is $\lceil M/Bn \rceil + n$ [38].

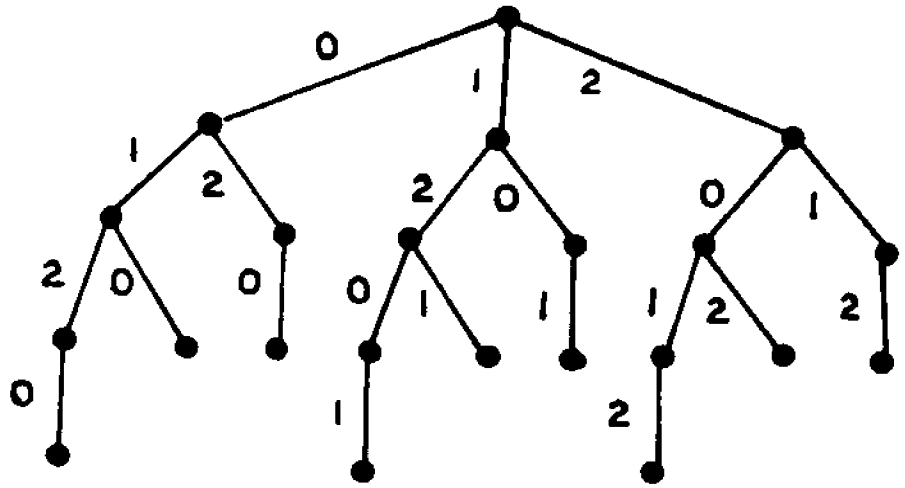


Figure 3.3. Multiple Spanning Binary Tree (MSBT) for a 3-cube.
(reproduced from [38])

3.4 Network Parameters for the n-cube

For a multiprocessor network, parameters such as diameter, degree of the node, message traffic density, and inter-node message delay are crucial and may be used to evaluate the performance of the network [75]. These parameters have been already discussed in Chapter 2 for several hypercube-based networks including the standard n–cube. Here, the focus will be on the *average internode message delay* and fault tolerance measures.

3.4.1 Average Distance ($\bar{d}$)

The summation of distances of all nodes from a given node (source) over the total number of nodes determines the average distance of the network. Average distance plays a key role in determining the queueing delay in a computer network. For the n–cube, there exist $\binom{n}{i}$ nodes at distance i from a given node. Thus,

$$\bar{d} = \frac{\sum_{i=1}^{n} i \binom{n}{i}}{N - 1}$$

$$= \frac{n\ 2^{n-1}}{N-1} \approx \frac{n}{2} \quad ; \quad N \gg$$

3.4.2 Diameter (d)

From the earlier discussion, the diameter for an n–cube is

$$d = n$$

3.4.3 Cost (ξ)

A network with large diameter typically has a small degree of the node but suffers from long delay in interprocessor communication. On the other hand, a small-diameter network usually has a large degree of the node. The loop and fully connected networks are two such examples. It is desirable to have a computer network

with small diameter and degree of the node. Thus, for a symmetric network, a cost factor ξ can be defined as the product of the diameter and the degree of the node (i.e. number of links per node) [75]. For the n–cube it follows:

$$\xi = d\ n = n^2$$

3.4.4 Message Traffic Density (ρ)

This factor is defined as

$$\rho \equiv \frac{Average\ message\ distance \times number\ of\ nodes}{total\ number\ of\ links}$$

Assuming each node sending one message to a node at distance $\bar{d}$ on the average and considering the availability of $n \times N/2$ links to accommodate such a traffic, for the n–cube it follows:

$$\rho = \frac{\bar{d}\,N}{n \times N/2} = 2\,\frac{\bar{d}}{n} \approx 1;\ N \gg$$

3.4.5 Average message delay (T)

One of the crucial factors in performance analysis of a computer network is the average message delay (T) [81]. T can be obtained by taking the average delay of all inter-node message delays through the network. To determine T, a hypercube-based network is modeled as a communication net with the ith channel represented as an $M/M/1$ system with Poisson arrivals at the rate λ_i and exponential service time of mean $1/\mu c_i$, μ being average service rate and c_i the capacity of the ith channel. In addition, the following assumptions are made:

i) Each node is equally likely to send a message to every other node in a fixed time period.

ii) The routing is prefixed. This implies that a unique path exists through the network for a given origin-destination pair. Conventionally, a node may transmit

the message along the direction of the right most differing bit.

iii) The load is evenly distributed, i.e. λ_i is the same for all i.

iv) The capacity of all channels are the same, i.e. c_i is the same for all i.

Under the above conditions, the delay of the network is given by [81]:

$$T = \frac{\bar{d}\,(\sum_{i=1}^{M}(\frac{\lambda_i}{\lambda})^{1/2})^2}{\mu C(1-\bar{d}\Gamma)}$$

where,

M = total number of directed links,

$\lambda = \sum_{i=1}^{M}\lambda_i = M\ \lambda_i$,

Γ = utilization factor [†] , and

$C = \sum_{i=1}^{M} c_i = M$, c_i total capacity of the architecture.

Due to the symmetry and homogeneity of the n-cube, the above formula can be substantially simplified. Assuming all links are bidirectional,

$$M = (\frac{N}{2}).2(n)$$

Hence,

$$T = \frac{\bar{d}(\frac{N}{2})2(n)}{\mu C(1-\bar{d}\Gamma)}$$

With constants μ,C, and N, the above delay can be normalized as

$$T' = \frac{\bar{d}(n)}{1-\bar{d}\ \Gamma}.$$

---

† $\Gamma \equiv \frac{\gamma}{\mu C}$ is the average rate at which messages enter the network per unit of link capacity (γ is the mean number of total messages flowing through the network). This factor reflects the utilization of the communication links. Large values of Γ correspond to heavy load conditions in the network.

3.5 Fault Tolerance Capabilities of the n-cube

3.5.1 Connectivity

The *connectivity* c of a graph is the minimum number of nodes that must be removed to disconnect the graph or reduce it to a solitary node [82]. Frequently, this factor is used as a measure of the fault tolerance capability of a network [71],[75],[83]. For a symmetric network, this factor is the same as degree of the node. The connectivity of the n–cube is n [33]. This implies that up to $n-1$ nodes can fail without disrupting the network †.

3.5.2 Two-Terminal Reliability [84]

This factor (also referred to as path reliability) is an important criterion to evaluate the reliability of a communication network. The Two-Terminal Reliability between two nodes u and v is defined as the probability of finding a path entirely composed of operational edges between u and v. Finding the exact solution for this factor in the n–cube is extremely difficult, since there may be too many paths between two given nodes sharing one or more links. Later in Chapter 5, a lower bound is given for the two-terminal reliability of the n–cube.

3.5.3 Container

A set of node-disjoint paths is termed as a *container* [71]. The *width* of a container is the number of node-disjoint paths it includes. The *length* of a container is the length of the longest path in the container. The best container of a given width between two nodes is the one of the shortest length [71]. Containers have many uses. For instance, by sending a message along different paths, the message will arrive at its

---

† Disrupting here means that an operational node (or more) is disconnected from the rest of the network due to failures.

destination correctly if a majority of the paths have all nodes nonfaulty. Moreover, if a pair of nodes has much communication, congestion can be reduced by sending messages along various paths. In a network with little traffic, long messages can be received earlier by sending individual packets along various paths. With the existence of wide containers, fault tolerance capabilities of a network can be remarkable. In an n–cube, the width of the best container between two arbitrary nodes, namely u and v, is n and the length of this container is $r+2$ where r is the Hamming distance between u and v. This parameter will be quantified for the n–cube in Chapter 5.

3.5.4 f-Fault Diameter

As mentioned in Section 2.6.1, connectivity is a common criterion of fault tolerance of a network. A regular network of connectivity n is guaranteed to remain connected for any f faulty nodes ($f<n$). However, while the connectivity of the network is preserved under f faulty nodes, the resulting diameter of the graph- a measure of communication delay- might significantly increase. An f–fault diameter, d_f, of a graph is defined to be the maximum of the distances over all possible graphs that can occur with at most f faults [43]. In a regular graph of degree n, $d_n=\infty$ (since if all neighbors of any node fail, the graph becomes disconnected). Of particular interest, therefore, is d_{n-1}. Short f–fault diameters are desirable. It has been shown [43] that for an $n-$ cube, $d_{n-1}=n+1$. This indicates that even under highly faulty conditions, the performance of the n–cube will not degrade drastically.

CHAPTER 4

Bridged Hypercubes

As discussed earlier, the diameter of a computer network plays a significant role in its performance, since it establishes a lower bound on the communication delay among the nodes in the network.

A network with few communication links usually has a large diameter. Clearly, one can reduce the diameter by adding some extra links to the network. However, a design with too many links may not be satisfactory since it imposes a high degree of the node and high communication cost on the network. The loop and completely connected networks are two extremes in the spectrum of networks with various diameters and communication costs. For N nodes in the network, the loop has a low communication cost ($O(N)$), but a large diameter ($O(N)$). On the other hand, the completely connected system has a diameter of one but suffers from tremendous communication cost ($O(N^2)$). One can make a compromise between the communication cost and diameter by introducing a cost factor ψ defined as the product of the diameter and number of links in the system †. The objective then, will be to minimize this factor.

In this chapter, the problem of improving the cost of a hypercube-based structure by adding some redundant links to the standard hypercube is considered. A new architecture called *Bridged Hypercube* (BHC) is then introduced. The most attractive feature of the BHC is that its diameter is approximately half the diameter of a hypercube of the same dimensionality. Link redundancy in the BHC is small and tends to

† Some researchers [75] have used the product of the diameter and the degree of the node as a cost factor. This makes most sense in symmetric networks where the degree of the node is the same for every node. The factor ψ introduced here is more adequate since it is applicable to asymmetric networks as well.

zero as the hypercube dimension grows yielding an asymptotically overhead free design. One to One and Broadcast algorithms are developed for this topology and proved to be optimal. The chapter also includes another topology with fewer links and slightly larger diameter. The advantages and disadvantages of the new topologies are compared to those of the n–cube.

4.1. Reducing the Network Diameter

Some researchers have considered improving the diameter of interconnection networks by adding redundant links to the structure. The Chordal Ring [5] and X-Trees [9] are two examples obtained by adding certain links to the ring and tree networks, respectively. These designs are very cost-effective and have a low communication overhead. Clearly, the diameter of the hypercube can be reduced in the same way. However, unlike the ring or tree structures, determining the number of redundant links and their positioning in the hypercube is not very straightforward. Let r be the optimal number of links to be added to an n–cube to halve the diameter. There are $n2^{n-1}$ links in the n–cube. On the other hand, if the cube were fully connected, it would have $2^{n-1}(2^n-1)$ links. Thus, the number of ways to choose r links from the space of possible redundant links in the n–cube is:

$$\begin{bmatrix} 2^{n-1}(2^n-n-1) \\ r \end{bmatrix}$$

This value, even for small n, leads to a combinatorial explosion. Thus, the problem of finding an optimal placement for redundant links is intractable unless some constraints are introduced to the problem (along with a systematic methodology to position the links).

One possible constraint is to limit redundant links to those which would join nodes located at two opposite corners of the cube, i.e. links connecting nodes whose

Hamming distance is n †. Each extra link is referred to as *complementary link* (since it connects two nodes whose labels are complements) or c–link for short. This is indeed a greedy approach to reduce the diameter, for intuitively, the best results may be obtained if extra links connect pairs of nodes with maximum distance. Imposing this constraint on the type of extra links serves two purposes. Firstly, the search space for the optimal or near-optimal extra links in order to reduce the diameter by a certain factor is substantially reduced (this space contains only $\begin{bmatrix} 2^{n-1} \\ r \end{bmatrix}$ choices). Secondly, selection of extra links from a constrained set tends to simplify the routing algorithms substantially. Next a new structure, which is a redundant hypercube with extra links subject to the above constraints, is introduced.

4.2. The Bridged Hypercube (BHC)

As mentioned earlier, the nodes of an n–cube can be partitioned into various classes based on their Hamming weights. More precisely, a class W_i consists of $\binom{n}{i}$ nodes, each having i 1's in its label. Thus, two adjacent nodes belong to two distinct classes of nodes whose weights are two consecutive integers. The adjacency relationship between the two classes mentioned above is shown in Figure 4.1. Observe that each dash between two classes W_i and W_{i+1} represents all links joining nodes of these classes. This does not necessarily mean that every node in W_i is connected to every node in W_{i+1}.

Lemma 1 : There exist exactly $i\binom{n}{i}$ links between the two classes W_{i-1} and W_i, $1 \le i \le n$.

Proof: Each node of W_i is connected to i distinct nodes in W_{i-1} through i distinct links. Each of these nodes is obtained by setting any one of the i 1's in a node of

† This also implies that links connect mutually exclusive nodes, meaning no two extra links will be incident on the same node.

weight i to 0. There exist $\binom{n}{i}$ nodes in W_i. Therefore, the proof is complete. □

In order to have simple routing schemes, the c–links are placed only between two classes of nodes rather than being scattered all over the network. More specifically, the c–links join two classes of weights W_i and W_{n-i} where $0 \leq i \leq n/2$. Pictorially, it appears that the diameter will be halved approximately if the c–links join classes $W_{n/4}$ and $W_{3n/4}$ which are in the middle of intervals $[W_0, W_{n/2}]$ and $[W_{n/2}, W_n]$, respectively (Figure 4.2). This gives rise to a new topology called the *Bridged Hypercube* (BHC) formally defined as follows. A BHC of dimension n is an n–cube with every node $u \in W_{n/4}$ connected to its complement $u' \in W_{3n/4}$ via an extra link, and according to a prefixed labeling scheme. The set of extra links is called a *bridge*. Figure 4.3 shows a BHC of dimension 4 with the bridge established between classes W_1 and W_3. To avoid the clutter, certain links are indicated by labeled arrows. Links connecting pairs of nodes with similar labeled arrows are intended. Next, one to one and broadcast algorithms are developed for this structure. The number of hops needed to perform these routings is at most $n/2 + 1$ (approximately half that required in a conventional n–cube).

4.2.1. One to One Routing in BHC

In the BHC, node classes of various weights can be partitioned into four *Groups* as shown in Figure 4.2:

$$\textit{Group}\ \text{I}: \{W_i:\ 0 \leq i < n/4\}$$

$$\textit{Group}\ \text{II}: \{W_j:\ n/4 \leq j < n/2\}$$

$$\textit{Group}\ \text{III}: \{W_k:\ n/2 \leq k < 3n/4\}$$

$$\textit{Group}\ \text{IV}: \{W_l:\ 3n/4 \leq l \leq n\}$$

Consider an arbitrary pair of nodes, say s and d for source and destination. Let s_i and d_i be the ith bits of the binary labels of s and d, respectively. Define a 2-tuple g_i such that

$$W_0 - W_1 - \ .. - W_i - W_{i+1} - \ .. - W_n$$

Figure 4.1. Adjacency Relationship between W classes in the n—cube.

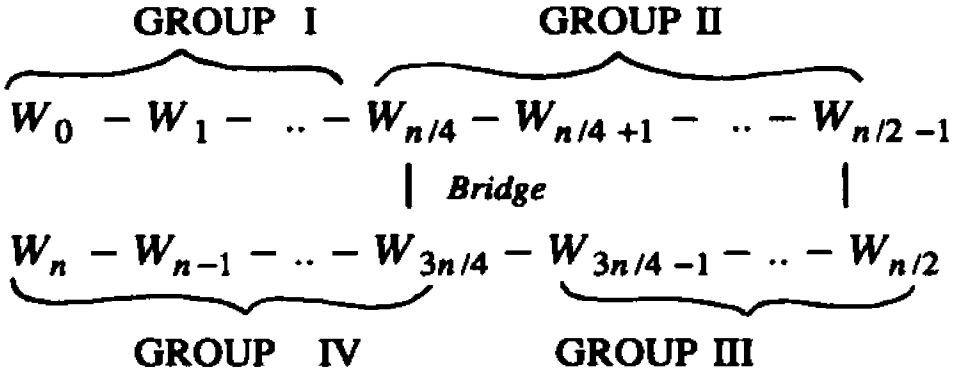


Figure 4.2. Adjacency Relationship between W classes in BHC(n).

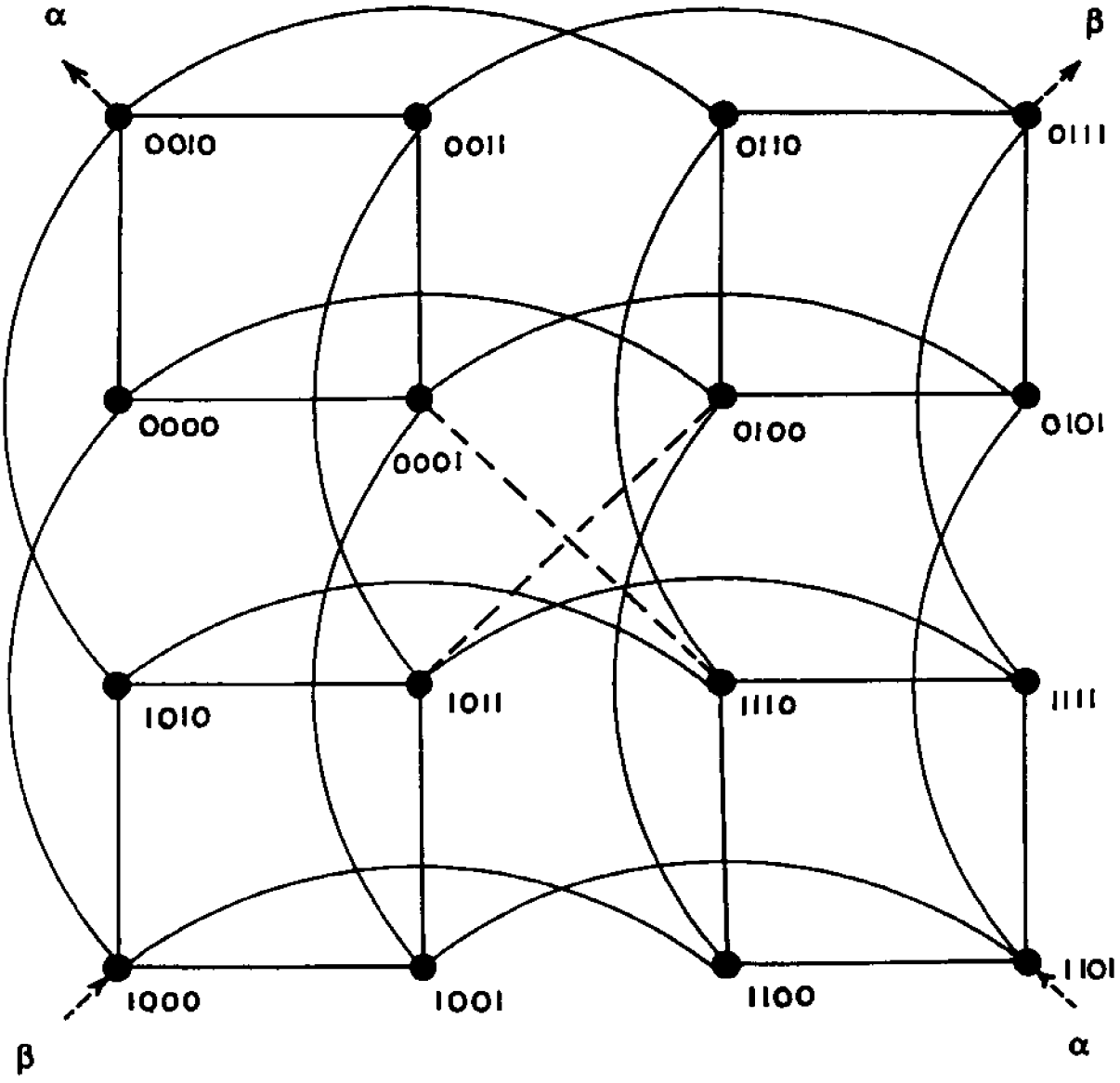


Figure 4.3. The BHC(4) Topology (dashed lines represent the bridge).

$$g_i \equiv (s_i d_i) \qquad 0 \le i \le n-1$$

Clearly, g_i can take on four values, namely 00, 01, 10, and 11. Now define the following sets:

$$M_{00} \equiv \{ g_i : \quad g_i = 00\},$$
$$M_{01} \equiv \{ g_i : \quad g_i = 01\},$$
$$M_{10} \equiv \{ g_i : \quad g_i = 10\},$$
$$M_{11} \equiv \{ g_i : \quad g_i = 11\}, \qquad \text{for } 0 \le i \le n-1.$$

It is further assumed,

$$| M_{00} | = \delta_1, | M_{01} | = \delta_2, | M_{10} | = \delta_3, | M_{11} | = \delta_4,$$

where $|X|$ is the cardinality of set X. Denoting the Hamming distance of s and d by $d_H(s,d)$, one can easily conclude that $d_H(s,d) = \delta_2 + \delta_3$.

Theorem 1: The diameter of the BHC(n) is at most $n/2 + 1$.

Proof: As a proof, algorithms for routing a message between any two nodes are developed. It will be shown that, according to these algorithms, any pair of nodes can communicate in no more than $n/2 + 1$ hops.

If $\delta_2+\delta_3 \le n/2$, the routing is performed as in conventional hypercube by correcting the differing bits in s (relative to d) one at a time. In such cases, the number of hops is thus $\le n/2$. Therefore, it is assumed hereafter that $\delta_2+\delta_3 > n/2$.

The two nodes s and d can belong to any of the four groups defined earlier resulting in 16 different cases. Due to the symmetry of the structure (with respect to the bridge), groups I and II are similar to groups IV and III, respectively (see Figure 4.2). Thus only two cases for which s resides in I or II and d is located anywhere, are considered.

Case (i): s in I and d anywhere.

Define $\Delta \equiv n/4 - (\delta_3 + \delta_4)$. Two subcases are considered:

Subcase (a): $\Delta \leq \delta_1$

Algorithm 1.

input: s and d

Step 1: Complement any Δ bits † of M_{00}.

Step 2: Take the c–link.

Step 3: Complement the remaining differing bits in s and d one at a time.

Time Complexity:

Since s is in I, $\delta_3+\delta_4<n/4$ and $0<\Delta\leq n/4$. By complementing Δ bits of M_{00} in s, a node $s' \in W_{n/4}$ is reached. Note that complementing these Δ bits is necessary to reach a node in $W_{n/4}$ where a c–link (bridge) is available. At this point, $d_H(s',d)=\Delta+\delta_2+\delta_3$. Having arrived at s', the c–link incident on s' is taken to reach a node $d' \in W_{3n/4}$ where $d_H(d',d)=n-(\Delta+\delta_2+\delta_3)$. There are Δ hops in step 1, 1 hop in step 2, and $n-(\Delta+\delta_2+\delta_3)$ hops in step 3. The Number of Hops (NH) needed to reach d from s is thus,

$$NH = \Delta + 1 + n - (\Delta + \delta_2 + \delta_3) = n + 1 - (\delta_2 + \delta_3).$$

Since $\delta_2+\delta_3>n/2$,

$$NH < n + 1 - n/2 \quad or \quad NH \leq n/2.$$

Thus the theorem is correct for this subcase.

Subcase (b): $\Delta > \delta_1$

Assume $\Delta=\delta_1+\epsilon$, where $\Delta\leq n/4$.

---

† The implication of this statement is to use the links of the n–cube corresponding to the specified bits one at a time, in "Δ" hops.

Algorithm 2.

input: s and d.

Step 1: Complement all δ_1 bits of M_{00} in s.

Step 2: Complement ε bits of M_{01} in s.

Step 3: Take the c–link.

Step 4: Complement the differing bits one at a time.

Time Complexity:

In step 1, Δ hops are needed to reach a node $s' \in W_{n/4}$. Changing δ_1 bits of M_{00} will increase the distance from d by δ_1, then changing ε bits in M_{01} will reduce this distance by ε. Therefore, at this point, $d_H = \delta_1+\delta_2+\delta_3-\varepsilon$. In step 2, the c–link is taken to arrive at a node $d' \in W_{3n/4}$ where $d_H = n-(\delta_1+\delta_2+\delta_3-\varepsilon)$. The differing bits in d' and d are complemented one at a time to arrive at d. Total number of hops is therefore,

$$\begin{aligned} NH &= \delta_1 + \varepsilon + 1 + n - (\delta_1 + \delta_2 + \delta_3 - \varepsilon) \\ &= n + 1 + 2\varepsilon - (\delta_2 + \delta_3) \end{aligned}$$

But,

$$\varepsilon = \Delta - \delta_1 = n/4 - (\delta_3+\delta_4) - \delta_1 = n/4 - (\delta_1+\delta_3+\delta_4)$$

and since $\delta_1+\delta_2+\delta_3+\delta_4=n$, $\varepsilon=\delta_2-3n/4$. It follows,

$$NH = \delta_1 + \delta_4 + 2\delta_2 - 3n/2 + 1 = (\delta_2 - \delta_3) - n/2 + 1$$

But $\delta_2-\delta_3<n$. Thus $NH \leq n/2$, and the theorem is correct for this subcase.

Note that in the above algorithm, ε bits which were initially correct, are complemented and then corrected back. This is merely a consequence of not having c–links everywhere. Bits in the source node label must change in order to arrive at the site of c–links, i.e. $W_{n/4}$.

Case (ii): s in II and d anywhere.

Since s is in II, $n/4 \le \delta_3 + \delta_4 < n/2$. Let $\Delta \equiv (\delta_3+\delta_4)-n/4$. Two subcases are distinguished:

Subcase (a): $\Delta \le \delta_4$

Algorithm 3.

input: s and d

Step 1: Complement any Δ bits of M_{11} in s.

Step 2: Take the c–link.

Step 3: Correct the differing bits one at a time.

Time Complexity:

In step 1, Δ bits of M_{11} are complemented to reach $s' \in W_{n/4}$, where $d_H(s',d)=\delta_2+\delta_3+\Delta$. In step 2, the c–link is taken to arrive at $d' \in W_{3n/4}$ where $d_H(d',d)=n-(\delta_2+\delta_3+\Delta)$. The differing bits are then corrected in step 3. Thus,

$$\begin{aligned} NH &= \Delta + 1 + n - (\delta_2 + \delta_3 + \Delta) \\ &= n + 1 - (\delta_2 + \delta_3) \end{aligned}$$

Since $\delta_2 + \delta_3 > n/2$, $NH \le n/2$. Thus the theorem is correct for this subcase.

Subcase (b): $\Delta > \delta_4$

Assume $\Delta = \delta_4 + \epsilon$. Depending on the values of δ_2 and δ_3 two subcases are distinguished:

Subcase (b1): $\delta_3 < \delta_2$

Algorithm 4.

input: s and d.

Step 1: Complement all δ_4 bits of M_{11}.

Step 2: Complement all ϵ bits of M_{10}.

Step 3: Take the c–link.

Step 4: Correct the differing bits.

Time Complexity:

Similar to the discussion for Algorithm 2, it follows,

$$NH = \delta_4 + \varepsilon + 1 + \delta_1 + \varepsilon = \delta_1 + \delta_4 + 2\varepsilon + 1.$$

Since $\varepsilon = \Delta - \delta_4 = (\delta_3 + \delta_4) - n/4 - \delta_4 = \delta_3 - n/4$, NH is given by:

$$\begin{aligned} NH &= \delta_1 + \delta_4 + 2\delta_3 - n/2 + 1 \\ &= n - \delta_2 + \delta_3 - n/2 + 1 \\ &= n/2 + 1 + (\delta_3 - \delta_2). \end{aligned}$$

and since $\delta_3 < \delta_2$, $NH \leq n/2$. Thus, the theorem is correct for this subcase.

Subcase (b2): $\delta_3 \geq \delta_2$

Define $\Delta' \equiv 3n/4 - (\delta_3 + \delta_4)$. The routing is performed as follows:

Algorithm 5.

input: s and d

Step 1: Complement Δ' bits in M_{00} and then in M_{01} if necessary.

Step 2: Take the c-link.

Step 3: Correct the differing bits.

Time Complexity:

If $\Delta' \leq \delta_1$, then,

$$NH = \Delta' + 1 + \delta_1 - \Delta' + \delta_4 = \delta_1 + \delta_4 + 1 \leq n/2.$$

For $\Delta' > \delta_1$, let $\Delta' = \delta_1 + \varepsilon'$. Following Algorithm 5, δ_1 bits and ε' bits are complemented in the sets M_{00} and M_{01}, respectively, to arrive at $s' \in W_{3n/4}$ where $d_H(s',d) = \delta_2 + \delta_3 + \delta_1 - \varepsilon'$. Next, the c-link is taken to arrive at a node d' where $d_H(d',d) = \delta_4 + \varepsilon'$. Finally, in step 3, $\delta_4 + \varepsilon'$ hops are taken to reach d. Therefore,

$$NH = \delta_1 + \varepsilon' + 1 + \delta_4 + \varepsilon' = \delta_1 + \delta_4 + 1 + 2\varepsilon'$$

On the other hand,

$$\epsilon' = \Delta' - \delta_1 = 3n/4 - (\delta_3 + \delta_4) - \delta_1 = \delta_2 - n/4.$$

After some algebraic manipulations, it follows:

$NH = n/2 + 1 + \delta_2 - \delta_3$. But $\delta_2 - \delta_3 \leq 0$, thus $NH \leq n/2 + 1$.

Since $NH \leq n/2 + 1$ for subcases (b1) and (b2), the proof is complete for all cases. □

Observe that in Algorithm 5 (as in Algorithm 2), ϵ' bits are complemented and corrected back in order to use a c−link. Moreover, note that this is the only case where NH can be at most $n/2+1$. The other algorithms require at most $n/2$ hops to route a message between a given pair of nodes. Note that in each algorithm, the message is guaranteed to reach the correct destination. This is true since conventionally in the standard cube, the destination is reached by correcting δ_2 and δ_3 bits belonging to sets M_{01} and M_{10}, respectively. This is indeed equivalent to using the c-link (thus complementing all n bits in the label), and then changing bits belonging to sets M_{00} and M_{11} in the BHC(n).

Theorem 2: The diameter of the BHC(n) is $n/2 + 1$.

Proof: It is sufficient to show that at least one pair of nodes can be found with a minimum distance of $n/2 + 1$. Consider two nodes $s \in W_0$ and $d \in W_n$. Optimal routing between these nodes can be achieved using Algorithm 1. It takes $n/4$ hops to arrive at $W_{n/4}$, i.e. the site of c−links. One hop is required to traverse the c−link, and finally $n/4$ extra hops are needed to reach W_n from $W_{3n/4}$. The total number of hops is therefore, $NH = n/2 + 1$. □

The flowchart in Figure 4.4 depicts the one to one routing algorithm for the BHC.

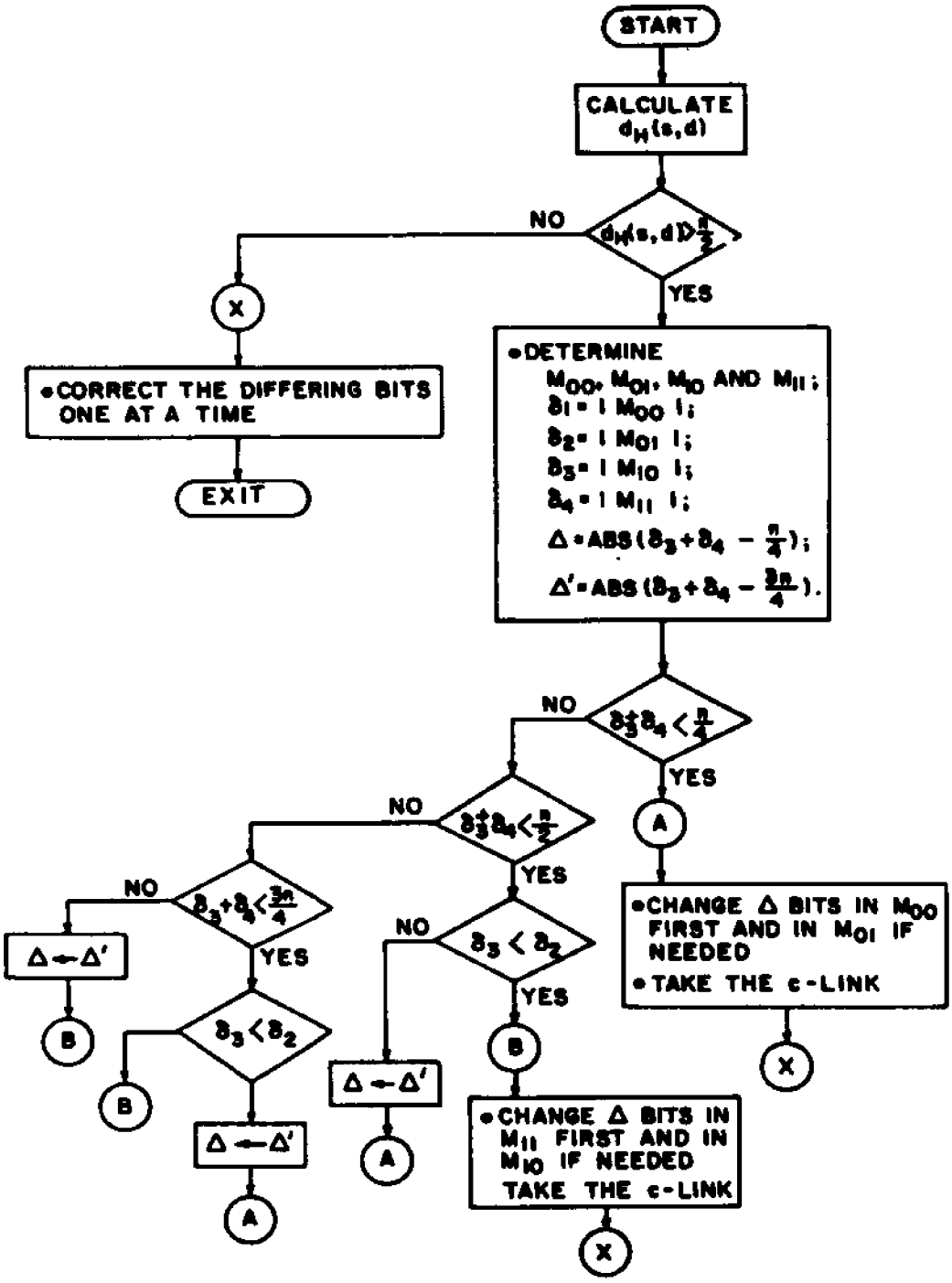


Figure 4.4. Flowchart of One to One Routing in the BHC(n).

(ABS(X) is absolute value of X)

4.2.2 Broadcasting in BHC

By Theorem 2, the diameter of the BHC is $n/2 + 1$. In this section, an optimal broadcasting algorithm is presented which is of complexity $n/2 + 1$. As in the n–cube, the description of the broadcast algorithm will be based on defining a spanning tree for the network graph which has the minimum height. The broadcast tree will be defined by the children (or parent) functions for every node of the network.

The spanning broadcast tree (SBT), described in Chapter 3, has a height of n. This means that broadcasting can be performed in n steps, assuming concurrent operation on ports of every node. Observe that in the ith step of the SBT algorithm, all nodes of weight i are covered. The adjacency relationships between classes of nodes W_i in accordance with SBT algorithm are depicted in Figure 4.5.

Now, suppose a message is to be broadcast from node $0^n \in W_0$ in the BHC(n). The data flow is shown in Figure 4.6. The message is first broadcast such that all nodes belonging to classes W_1 through $W_{n/4}$ are covered. Nodes of $W_{n/4}$ in turn send the message to nodes of classes $W_{n/4+1}$ and $W_{3n/4}$. In the next phase nodes of $W_{n/4+1}$ send the message along the chain until all nodes of $W_{n/4+2}$ through $W_{n/2}$ are covered. At the same time, nodes of $W_{3n/4}$ send the received message to nodes on the right and left corresponding to classes $W_{3n/4-1}$ and $W_{3n/4+1}$, respectively. The broadcasting then continues in both branches until all nodes of weight $W_{n/2+1}$ and W_n are covered. The figure does not give any information regarding the connections among individual nodes in the broadcast tree. In the following, a broadcast algorithm will be devised specifying the relationships among individual nodes. This algorithm conforms with the adjacency relationship between different classes of nodes described above and guarantees that each receiving node gets the message exactly from one node.

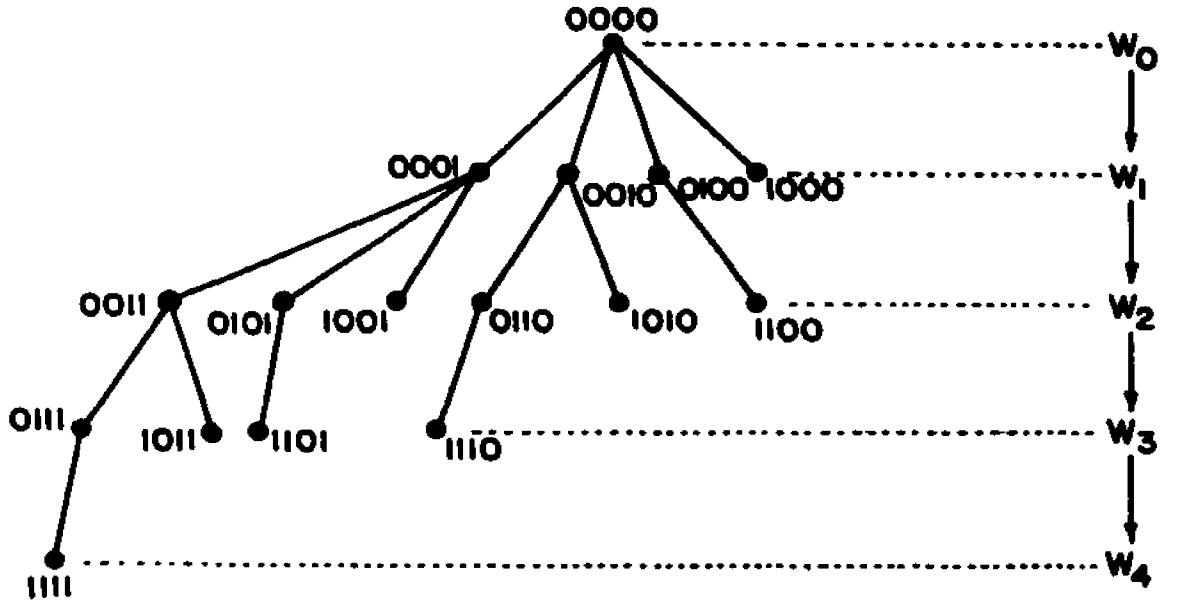


Figure 4.5. Dataflow through W classes according to BST in a 4-cube.

$$W_0 \rightarrow W_1 \rightarrow \ .. \rightarrow W_{n/4} \rightarrow W_{n/4+1} \rightarrow \ .. \ \rightarrow W_{n/2}$$
$$\downarrow$$
$$W_n \leftarrow W_{n-1} \leftarrow \ .. \leftarrow W_{3n/4} \rightarrow W_{3n/4-1} \rightarrow \ .. \rightarrow W_{n/2+1}$$

Figure 4.6. Dataflow through W classes according to BST in the BHC(n).

The broadcast algorithm for the BHC is based on the *Bridged Spanning Tree* (BST). The children and parent functions are to be described for BST. For a node u in the BHC, the following parameters are defined:

k: The highest order bit of u that is 1, or $u_k = 1$ and $u_m = 0$, $\forall\ m > k$ with $k = -1$ for $u = 0$.

k': The highest order bit of u that is 0, or $u_{k'} = 0$ and $u_p = 1$, $\forall\ p > k'$.

Moreover, let *children (u)* be the set of child nodes of node u in the BST and define:

$$M_{BST}(u) \equiv \{n-1\ ,\ ..\ ,\ k+1\}$$

$$P_{BST}(u) \equiv \{n-1\ ,\ ..\ ,\ k'+1\}$$

For $n > 4$, BST can be specified as follows:

a) For $0 \le \|u\| \le n/2$, $3n/4 + 1 \le \|u\| \le n$

$$children_{BST}(u) = \begin{cases} \{(u_{n-1}u_{n-2}..\overline{u}_m..u_0),(\overline{u}_{n-1}\overline{u}_{n-2}..\overline{u}_0)\}, & \|u\| = n/4 \\ \{(u_{n-1}u_{n-2}..\overline{u}_m..u_0)\}, & \|u\| \ne n/4\ ,\ \|u\| \ne n \\ \phi, & \|u\| = n \end{cases}$$

$$\forall\ m \in M_{BST}(u)$$

$$parent_{BST}(u) = \begin{cases} \phi, & \|u\| = 0 \\ (u_{n-1}u_{n-2}..\overline{u}_k..u_0), & \|u\| \ne 0 \end{cases}$$

b) For $n/2 + 1 \le \|u\| \le 3n/4$:

$$children_{BST}(u) = \begin{cases} \{(u_{n-1}u_{n-2}..\overline{u}_m..u_0),(u_{n-1}u_{n-2}..\overline{u}_p..u_0)\}, & \|u\| = 3n/4 \\ \{(u_{n-1}u_{n-2}..\overline{u}_p..u_0)\}, & \|u\| \ne 3n/4\ ,\ \|u\| \ne n/2+1 \\ \phi, & \|u\| = n/2+1 \end{cases}$$

$$\forall\ m \in M_{BST}(u)\ ,\quad \forall\ p \in P_{BST}(u)$$

$$parent_{BST}(u) = \begin{cases} (\overline{u}_{n-1}\overline{u}_{n-2}..\overline{u}_0), & \|u\|=3n/4 \\ (u_{n-1}u_{n-2}..\overline{u}_k..u_0), & \|u\|\neq 3n/4. \end{cases}$$

If $n=4$, the BST is described in the following simple form:

$$children_{BST}(u) = \begin{cases} \{(u_3..\overline{u}_m..u_0)\}, & \|u\|=0 \text{ or } \|u\|=3 \\ \{(u_3..\overline{u}_m..u_0),(\overline{u}_3\overline{u}_2\overline{u}_1\overline{u}_0)\}, & \|u\|=1 \\ \phi, & \|u\|=4 \end{cases}$$

$$\forall\ m \in M_{BST}(u)$$

$$parent_{BST}(u) = \begin{cases} \phi, & \|u\|=0 \\ (\overline{u}_3\overline{u}_2\overline{u}_1\overline{u}_0) & \|u\|=3 \\ (u_3..\overline{u}_k..u_0), & Otherwise. \end{cases}$$

Theorem 3: Broadcasting in the BHC(n) using the BST scheme described above (from the source node $0^n \in W_0$) can be performed in $n/2 + 1$ steps.

Proof: It must be shown that all nodes of the BHC (except the source) uniquely receive the broadcast message in at most $n/2 + 1$ steps. It can be readily seen that the children and parent functions in BST are the same as those in SBT for which the correctness proof is given in [34]-[38]. The only case in BST which is different is case (b) where the message continues to be broadcast from nodes of $W_{3n/4}$ all the way down to $W_{n/2+1}$. Observe that this case is the dual of the case where the message propagates from $W_{n/4}$ all the way up to $W_{n/2-1}$. Based on this principle, k' has been defined by changing 1 to 0 in the definition of k. Attributing the set P_{BST} to k', the children and parent functions can be easily obtained as given before for the interval $[W_{3n/4}, W_{n/2+1}]$.

It takes $n/4$ steps for the message to reach all nodes of $W_{n/4}$. One step is required to cover $W_{n/4+1}$ and $W_{3n/4}$ simultaneously. Broadcasting then continues in the intervals: $[W_{n/4+2}, W_{n/2}]$, $[W_{3n/4-1}, W_{n/2+1}]$, and $[W_{3n/4+1}, W_n]$.

The number of hops to complete broadcasting is therefore,

$$NH = n/4 + 2 + \max\,[\, n/4-2\,,\, n/4-2\,,\, n/4-1\,] = n/2 + 1. \qquad \square$$

In the above analysis, it is implicitly assumed that n is divisible by 4. The extension to cases where n is an arbitrary integer is straightforward. Table 4.1 gives the diameter obtained by adding $\binom{n}{m}$ links to the standard n–cube for an arbitrary n, where $m = \lfloor n/4 \rfloor$. In all architectures, the bridge is established between classes of W_m and W_{n-m}.

Table 4.1. The Diameter and Number of c–links of the BHC(n).

n	*diameter* (d)	ΔL
$4m$	$2m + 1$	$\binom{n}{m}$
$4m + 1$	$2m + 2$	$\binom{n}{m}$
$4m + 2$	$2m + 3$	$\binom{n}{m}$
$4m + 3$	$2m + 4$	$\binom{n}{m}$

4.3. Narrow Bridged Hypercube (NBHC)

The number of links in the BHC bridge may be substantially reduced at the expense of a small increase in diameter and slightly more complex routing algorithms. In the following a network called *Narrow Bridged Hypercube* (NBHC), with fewer bridge links and slightly larger diameter, is presented.

Let p be an integer such that $0 \le p \le n-1$. Define $W'_{n/4}(p) \subset W_{n/4}$ and $W'_{3n/4}(p) \subset W_{3n/4}$ for a given p such that:

$$W'_{n/4}(p) \equiv \{u:\ u_p = 1 \text{ and } u \in W_{n/4}\}$$
$$W'_{3n/4}(p) \equiv \{v:\ v_p = 0 \text{ and } v \in W_{3n/4}\}$$

where u_p and v_p represent the pth bit position in u and v labels respectively. An NBHC(n, p) is constructed from an n–cube by establishing a bridge between $W'_{n/4}(p)$ and $W'_{3n/4}(p)$. Next, the diameter of the NBHC(n, p) will be determined by developing one to one routing algorithms for this network.

4.3.1. One to One Routing in NBHC

Theorem 4: The diameter of the NBHC(n, p) is $n/2 + 3$.

Proof : see Appendix I.

The flowchart for routing between two nodes in the NBHC(n, p) is given in Figure 4.7. Observe the complexity of the routing in this case as compared to that of the BHC. The extension of the above analysis to the case where n is not divisible by 4 is straightforward and can be carried out in the same manner as in the BHC.

4.3.2. Broadcasting in NBHC

The broadcast tree for the NBHC is the same as that of the BHC except for two additional steps. More specifically, according to BST (i.e. broadcast tree for the BHC), at the end of message transfer from $W_{n/4}$ to $W_{3n/4}$ those nodes of $W_{3n/4}$ whose pth bits are 1 have not received the message due to the absence of c–links. However, every node $u \in W_{3n/4}$ and labeled as $u_{n-1}u_{n-2}..1_p..u_1u_0$ can get the message via a path

$$u_{n-1}u_{n-2}..0_p..1_k..u_1u_0 \rightarrow u_{n-1}u_{n-2}..1_p..1_k..u_1u_0 \rightarrow u_{n-1}u_{n-2}..1_p..0_k..u_1u_0$$

where x_y indicates that the yth bit of the label is x ($x \in \{0,1\}$), and k is a bit order in u which is 1. Broadcasting then continues as in BHC. One can readily observe that

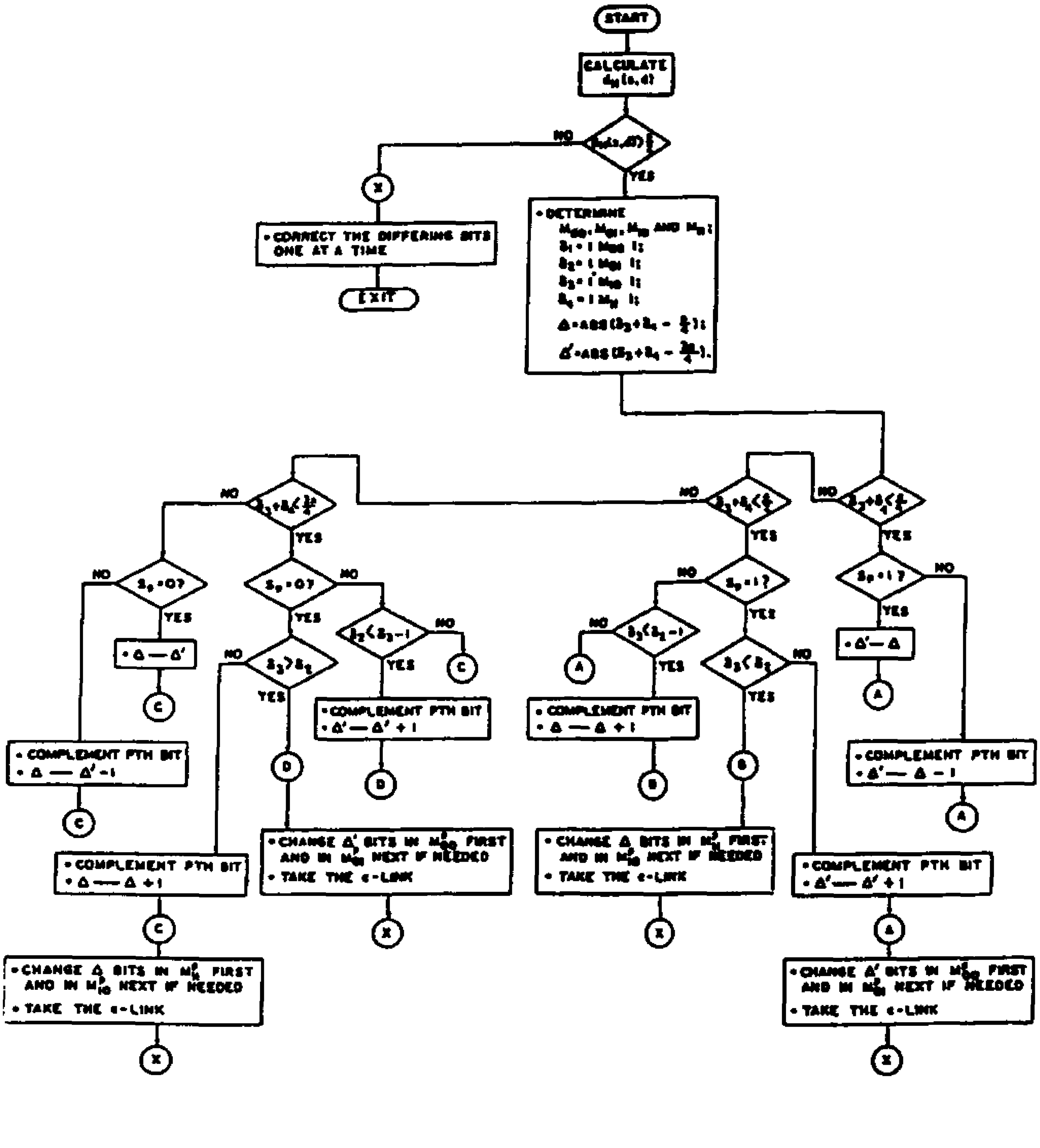


Figure 4.7. Flowchart of One to One Routing in the NBHC(n).
(ABS(X) is absolute value of X)

the complexity of the broadcasting algorithm is $n/2 + 3$. This is indeed the diameter of NBHC which implies that the algorithm is time-optimal.

4.4 Performance Analysis of BHC and NBHC

Since BHC(n) is an augmented form of the n–cube, it possesses some attractive features of the n–cube and some more. The diameter of the BHC(n) is $n/2+1$. Therefore, for parallel algorithms which heavily rely on the shortest paths between nodes, the BHC may remarkably outperform its hypercube counterpart. In the BHC(n), every c–link is incident on a node of $W_{n/4}$. Thus, the number of links required to establish the bridge is: $\binom{n}{n/4}$, yielding a total number of links $L = n\ 2^{n-1} + \binom{n}{n/4}$. On the other hand, the diameter of the BHC is $n/2 + 1$. Thus, the cost factor ψ, defined earlier in this chapter, for the BHC will be:

$$\psi_{BHC} = (n/2 + 1) \times \left[(n\,2^{n-1}) + \binom{n}{n/4} \right]$$

The NBHC(n) has a diameter of $n/2+3$. The number of links comprising the bridge of the NBHC (extra links) can be obtained by noting that the pth bit of any node of $W_{n/4}$ incident on a c-link is always 1. There exist $\binom{n-1}{n/4-1} = \frac{1}{4}\binom{n}{n/4}$ such links. Observe that reducing the number of links in the BHC bridge by 75% will only increase the diameter by an additive constant of 2. This fact reveals the inherent tradeoff between the link redundancy and the network diameter. Table 4.2 shows the number of c–links in BHC (ΔL_1) and NBHC (ΔL_2) for various network dimensionalities. For the BHC, the link redundancy (n–cube being the reference) is very small and less than 1% for $n>8$. The link redundancy is clearly even less for NBHC. The total number of links in the NBHC is $n\,2^{n-1} + \frac{1}{4}\binom{n}{n/4}$. Thus,

$$\Psi_{NBHC} = (n/2 + 3) \times [\ (n2^{n-1}) + \frac{1}{4} \binom{n}{n/4}]$$

Table 4.3 shows the costs of the standard cube, BHC, and NBHC for various n. Observe that the cost of the BHC tends to half that of the n–cube as n grows. The cost of the NBHC lies between the costs of the n–cube and BHC.

One major disadvantage of the BHC (or NBHC) is its asymmetry. The asymmetry arises since the bridge is set up between a subset of nodes of the network and according to a fixed labeling scheme. For this reason, derivation of other network parameters such as average internode distance or message traffic density is extremely difficult. Intuitively, these two factors are improved in the BHC (or NBHC) due to addition of some extra links.

As illustrated in the flowcharts in Figures 4.4 and 4.7, routing in BHC (or NBHC) is complicated when compared to the simple one to one routing in the standard cube. In addition, broadcast algorithms developed for BHC and NBHC are based on the assumption that the source node is 0^n. Nevertheless, some applications require the data to be broadcast from different source nodes. Devising a general algorithm for a BHC or NBHC may be a tedious task and difficult to implement.

The fault tolerance of the BHC and the NBHC is the same as that of the n–cube if connectivity is to be a criterion to measure this aspect. However, BHC (or NBHC to some extent) is more resilient than the n–cube when factors such as Two-Terminal Reliability are considered. The merits of the n–cube, BHC(n), and NBHC(n) are summarized in Table 4.4. It is worth mentioning that even though performance improvement in the NBHC may not justify its complexity, the analysis of this network is of theoretical value.

Table 4.2. The Link Redundancy in BHC and NBHC.

n	$\Delta L_1 = L_{BHC} - L_{n-cube}$	$\Delta L_2 = L_{NBHC} - L_{n-cube}$	$\Delta L_1(\%)$	$\Delta L_2(\%)$
8	28	7	2.73	0.68
12	220	55	0.89	0.22
16	1820	455	0.34	0.09
20	15504	3876	0.15	0.04
24	134596	33649	0.07	0.02
28	1184040	296010	0.03	0.01

Table 4.3. Cost comparison for the n-cube, BHC, and NBHC.

n	$\psi = d \times L$		
	n-cube	BHC	NBHC
8	8.19×10^3	5.26×10^3	7.22×10^3
12	2.94×10^5	1.74×10^5	2.20×10^5
16	8.38×10^6	4.73×10^6	5.77×10^6
20	2.10×10^8	1.16×10^8	1.37×10^8
24	4.83×10^9	2.61×10^9	3.03×10^9
28	1.05×10^{11}	0.56×10^{11}	0.64×10^{11}

Table 4.4. Merits of BHC and NBHC as compared to those of n–cube.

Merits	BHC	NBHC
Diameter	$n/2 + 1$	$n/2 + 3$
Extra links	$\binom{n}{n/4}$	$\frac{1}{4}\binom{n}{n/4}$
Average message density	slightly lower	slightly lower
Average distance	lower	lower
Cost	approximately half	more than half
Routing	slightly complicated	complicated
Connectivity	same	same
Two-Terminal Reliability	better	better

CHAPTER 5

Folded Hypercubes

Although the BHC has a low diameter, its asymmetry is a major drawback. This chapter investigates the possibility of enhancing the hypercube performance where the symmetry of the network is of concern. A new symmetric network called *Folded Hypercube* or FHC is proposed. Basic routing algorithms for the FHC are developed and shown to be 50% more efficient than those for the n–cube. The network parameters discussed in Chapter 3 are derived for the FHC. For each parameter, the FHC is compared to the n–cube. The redundant links in the FHC can be utilized as spare links to enhance the reliability of the network. Such a network is referred to as *Fault Tolerant Hypercube* or FTH. The Chapter includes some interesting properties of the FTH such as: high communication reliability, high subcube reliability, and the graceful degradation in a faulty environment.

5.1 The Structure of the Folded Hypercube (FHC)

A Folded Hypercube of dimension n, i.e. FHC(n) can be constructed from a standard binary n–cube by connecting each node to the unique node that is farthest from it. To formally describe the structure, the model for standard hypercubes used in Chapter 2 is employed.

5.2 Graph Theoretical Model of FHC.

An n–dimensional FHC can be modeled as a graph $F_n(V,E')$, with the same set of vertices as in $G_n(V,E)$ (the graph of an n–cube) and with the edge set of E' which is a superset of E such that $|E'|=|E|+N/2=(n+1)2^{n-1}$. Clearly, G_n is a spanning subgraph of F_n. In addition, $e=(u,v)\in E(F)$, iff $\|a(u)\oplus a(v)\|=1$ or n. The

requirement $\|a(u) \oplus a(v)\|=n$ implies that in F, each node is connected to its farthest node via an edge (link) hereafter referred to as *complementary* link † or c-link. A link is labeled i if it connects two nodes whose addresses differ in only the ith bit. A c-link is labeled by c (or n interchangeably). A path is represented by the labels of the links composing it. For instance, the path $0 \rightarrow 1 \rightarrow 2$ consists of links whose labels are 0,1, and 2. Figure 5.1 shows an FHC(3) network constructed from a 3-cube. The dashed lines represent the complementary links added to the 3-cube to obtain the FHC(3).

5.3 Topological Properties of the FHC.

Let F represent the topology of an FHC(n). Every node u with address $a(u)$ in F is connected to n nodes at Hamming distance 1 and one node at Hamming distance n. The latter node is reached via a complementary link and its address is obtained by complementing all n bits of $a(u)$. Therefore $deg_F(u)=n+1$, and F is regular of degree $n+1$. Since every node in F has $n+1$ neighbors, it is necessary to remove at least $n+1$ nodes to disconnect F. Thus, it follows:

Theorem 5: The node connectivity of the FHC(n) is $n+1$.

From Theorem 5, the following lemma results:

Lemma 2: The number of node-disjoint paths between any two nodes in the FHC(n) is $n+1$.

Proof: This follows directly from Menger's Theorem [82].

The diameter of FHC(n) is established in the following theorem:

---

† The same definition is used in Chapter 4 for such a link.

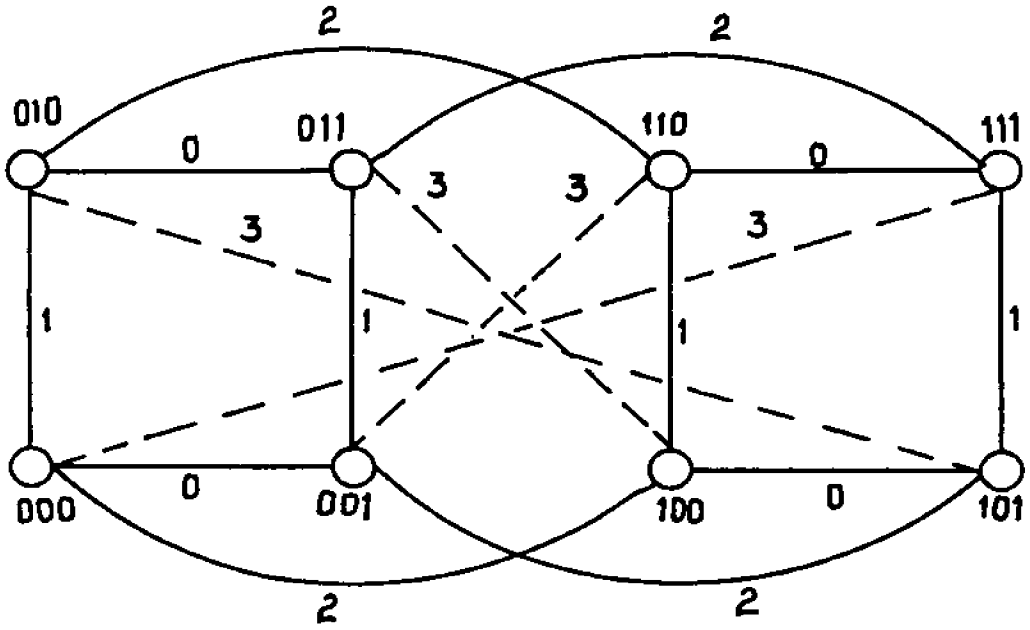


Figure 5.1. The Structure of the FHC(3)
(dashed lines represent the c -links)

Theorem 6: The diameter of FHC(n) is $\lceil n/2 \rceil$ ($\lceil x \rceil$ is the smallest integer greater than or equal to x).

Proof: Consider any pair of nodes $(u,v) \in V$ with $\|a(u) \oplus a(v)\| = \lceil n/2 \rceil + i$ where $-\lceil n/2 \rceil < i \leq \lfloor n/2 \rfloor$ and $\lfloor x \rfloor$ is the greatest integer smaller than or equal to x. If $-\lceil n/2 \rceil < i \leq 0$, u and v can communicate in at most $\lceil n/2 \rceil$ hops by correcting the differing bits in their addresses one at a time. For $0 < i \leq \lfloor n/2 \rfloor$, a path can always be established between u and v by using the c-link which connects u to its farthest node u'. The Hamming distance between u' and v is clearly $n - (\lceil n/2 \rceil + i) = \lfloor n/2 \rfloor - i$. Therefore $d(u,v) = 1 + \lfloor n/2 \rfloor - i$ and since i is assumed to be positive, $d(u,v) \leq \lceil n/2 \rceil$ $\forall\ u,v \in V$. □

Observe that in G, the Hamming distance between two nodes specifies the length of the shortest path between them. This is not necessarily true in F, for two adjacent nodes in F may have a Hamming distance of n. To make a distinction between the Hamming distance and the shortest distance, the Hamming distance between two nodes, say u and v, is denoted hereafter by $d_H(u,v)$. Note that in F

$$d(u,v) = \min\ (\ d_H(u,v)\ ,\ n - d_H(u,v) + 1\ )$$

From the above expression, the following useful property of F can be stated:

$$d(u,v) = d_H(u,v) \qquad \text{for } 0 < d_H(u,v) \leq \lceil n/2 \rceil$$

$$d(u,v) = n - d_H(u,v) + 1 \qquad \text{for } \lceil n/2 \rceil < d_H(u,v) \leq n$$

for every pair of $(u,v) \in V$. This means that in routing algorithms, the shorter path between two nodes can be selected. Avoiding unnecessary communication delays and message congestion are important features of this scheme.

Theorem 7: The number of nodes at distance i from any node in the FHC(n) is $\binom{n+1}{i}$, where $0<i<\lceil n/2 \rceil$. For $i=\lceil n/2 \rceil$, this number is $\binom{n+1}{n/2+1}$ for even n and is $\binom{n}{\lceil n/2 \rceil}$ for odd n.

Proof: Denote the set of nodes in F at distance i from a given node u by $N_i(u)$. Let $|N_i(u)|$ be the cardinality of $N_i(u)$. Note that the FHC, similar to the n–cube, is *vertex-symmetric*, i.e. given any two nodes u and v, there exists an automorphism of the graph F which maps u to v. Thus, without loss of generality, the argument u can be dropped in the above terms. Two cases are considered.

Case 1: $0<i<\lceil n/2 \rceil$

There are $\binom{n}{i}$ nodes in G at Hamming distance i from u (this is the well known property of the n–cube [33]). Since G is a spanning subgraph of F, there exist at least $\binom{n}{i}$ nodes in F at distance i from u. Now consider the node u' incident on the c-link emanating from u. There exist $\binom{n}{i-1}$ nodes at distance $i-1$ from u' (note that the Hamming distance of these nodes from u is $n-i+1$). The distance of such nodes from u (via the c-link) is therefore $i-1+1=i$. Observe that if a c-link is not used in forming a path, the distance of such nodes from u becomes $n-i+1$ which is greater than i. Therefore,

$$|N_i| = \binom{n}{i} + \binom{n}{i-1} = \binom{n+1}{i}$$

Case 2: $i=\lceil n/2 \rceil$. For this case 2 subcases are distinguished:

case (2a) n is even,

$$|N_{n/2}| = \binom{n}{n/2} + \binom{n}{n/2-1} = \binom{n+1}{n/2+1}$$

case (2b) n is odd,

$$|N_{\lceil n/2 \rceil}| = \binom{n}{\lceil n/2 \rceil}$$

This is because when n is odd, any node $v \in N_{\lceil n/2 \rceil}$ would have $d(u,v) = n - \lceil n/2 \rceil + 1 = \lfloor n/2 \rfloor + 1 = \lceil n/2 \rceil = d_H(u,v)$. In other words, the set $N_{\lceil n/2 \rceil}$ contains nodes with the same Hamming distance and actual distance of $i = \lceil n/2 \rceil$ from u. □

The FHC(n) structure is modular in the sense that any lower dimension FHC can be formed from it. To see this, the following theorem is introduced:

Theorem 8: Given an FHC(n) and an integer k such that $0 < k < n$, an FHC(k) can be constructed out of nodes and links available in FHC(n).

Proof: Let $0 < k < n$. Denote the graphs of k-cube and FHC(k) by G_k and F_k, respectively. It is immediate that any G_k is a subgraph of G_n † (i.e. any k-cube can be obtained out of an n-cube). Therefore, only the c-links have to be established between nodes of G_k to obtain an F_k. For clarity, denote a complementary link in a k-dimensional medium by c_k. Label the nodes of the G_n such that all nodes in G_k have all 0's in their $n-k$ left most bit positions. Consider a node u with the address $a(u)=(0^{n-k}u_{k-1}..u_1u_0)$ and its complement node u' with the address $a(u')=(0^{n-k}\overline{u}_{k-1}..\overline{u}_1\overline{u}_0)$ in the G_k. A virtual c_k link can be formed between u and u' by taking any of the following paths:

Path 1: $c_n \rightarrow n-1 \rightarrow \cdots \rightarrow k+1 \rightarrow k$

Path 2: $n-1 \rightarrow n-2 \rightarrow \cdots \rightarrow k \rightarrow c_n$

....

† This is true since any G_n can be decomposed into two G_{n-1} and so on.

$$\text{Path } n-k+1:\ k \rightarrow c_n \rightarrow \cdots \rightarrow k+2 \rightarrow k+1$$

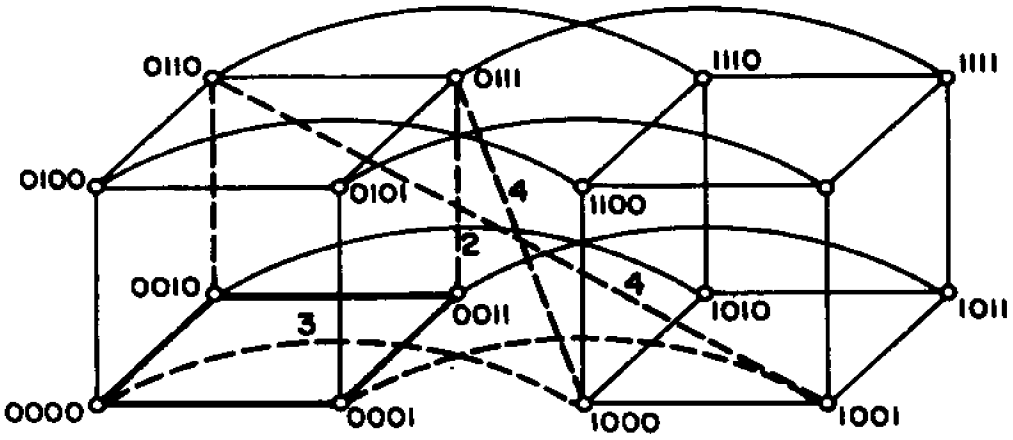
In path 1, traversal of link c_n complements all the bits of u. Then $n-k$ links must be traversed to change the $n-k$ left most bit positions in u' which changed while traversing link c_n. Other paths can be obtained by cyclically shifting the links in path 1 to the left, one at a time.

Note that in forming c_k links, all the used links as well as all intermediate nodes are outside of the G_k; therefore the union of the G_k and all the virtual c_k's such constructed between every pair of nodes in G_k will make up an F_k. But F_k is the graph model of FHC(k). Thus, the proof is reached. □

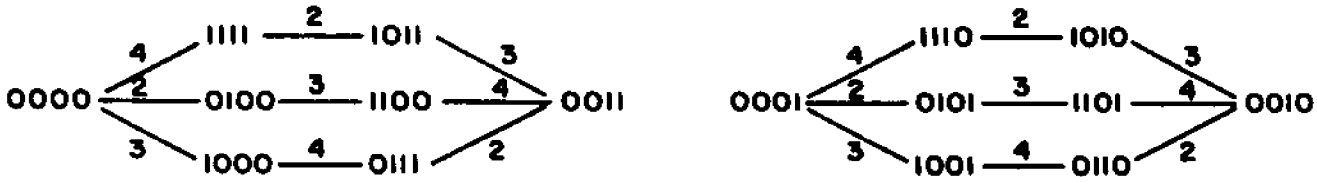
The intermediate nodes act as switches for routing the data. From the above theorem the following lemma results:

Lemma 3: The dilation of every c_k in an FHC(k) is $n-k+1$.

Moreover, observe that there exist exactly $n-k+1$ node-disjoint paths between any node u and its complement u' in the k-cube. Thus the bandwidth of virtual c_k's is $n-k+1$ times that of the other links. Figure 5.2 shows the construction of an FHC(2), consisting of nodes (0000, 0001, 0011, 0010), from an FHC(4). The formation of the virtual c_2's between nodes (0000, 0011) and nodes (0001,0010) is also depicted in the figure. Note that for each c_2 there exist $n-k+1=3$ alternative paths. To avoid the clutter, only two c-links of the FHC(4) are shown.



(a) The Structure



(b) Possible Paths to form the c_2 links.

Figure 5.2. Construction of an FHC(2) from an FHC(4)
(dashed lines represent the virtual c_2 links)

5.4 Routing Algorithms in the FHC

In this section, optimal routing algorithms are developed for the FHC. These algorithms are shown to be remarkably faster than those developed for the standard cube.

5.4.1 One to One Routing.

The following algorithm performs the routing between any pair of nodes, namely u and v, in an FHC(n).

One to One Algorithm:

input: $a(u)$, $a(v)$;

Begin

$a(w) = a(u) \oplus a(v)$

If $\|a(w)\| \leq \lceil n/2 \rceil$ **Then**

route the message sent from u via a path composed of links with labels corresponding to bit positions which are 1's in $a(w)$.

Else

send the message to u' via the c−link;

route the message via a path composed of links with labels corresponding to bit positions which are 0's in $a(w)$.

Endif

End.

Without loss of generality, suppose $\|a(w)\|=r$ and bit positions $w_{r-1}..w_1w_0$ are all 1's and bits $w_{n-1}..w_{r+1}w_r$ are all 0's. If $r \leq \lceil n/2 \rceil$, a path can be formed between u

and v which corresponds to any one of $r!$ permutations on the ordering of $w_{r-1}..w_1w_0$. In case of $r > \lceil n/2 \rceil$, as was seen before, a c-link must be used somewhere along the path. It follows that the shortest path in this case is any permutation of $(c, w_{n-1}..w_{r+1}w_r)$. Observe that in forming a path between u and v, it is immaterial where to use a c–link, for the length of the path such obtained is always the same. Moreover, the number of edges forming the shortest path between any pair of nodes in FHC(n) is at most $\lceil n/2 \rceil$. As can be seen, the above algorithm is optimal.

5.4.2 Broadcasting

As discussed in Chapter 3, the problem of broadcasting in a network can be modeled as one of finding a spanning tree of the graph representing the network. For the FHC, a spanning tree of the graph F is to be obtained. In F, there exist many broadcast trees of which those with minimum height are desirable. Note that the height of the tree specifies the number of steps required to perform the broadcasting. The lower bound on the height of such trees is $\lceil n/2 \rceil$, the diameter of the FHC.

5.4.2.1 Spanning Binary Tree for the FHC (SBTF)

To express the broadcast tree with minimum height in F, all node addresses are modified by defining a mapping A from previous address space to a new address space. Let the root be located at node s and $a(v)=a(u) \oplus a(s)$ for a given node u in F. Then A is specified as follows:

$$a_A(v) = A(v_{n-1}..v_1v_0) = \begin{cases} 0v_{n-1}..v_1v_0 & \text{if } \|v\| \le \lceil n/2 \rceil; \\ 1\overline{v}_{n-1}..\overline{v}_1\overline{v}_0 & \textit{otherwise.} \end{cases}$$

Table 5.1 shows the address mapping for a 4-cube. Having mapped the addresses, define the set of integers $M_{SBTF}(u,s) = \{k+1,..,n-1,n\}$ where k is position of the highest order bit of $v=u \oplus s$ that is 1 as specified before. Then,

Table 5.1. Address Mapping for a 4-cube

$a(u)$	$a_A(u)$	$a(u)$	$a_A(u)$
0000	0 0000	1111	1 0000
0001	0 0001	1110	1 0001
0010	0 0010	1101	1 0010
0100	0 0100	1011	1 0100
1000	0 1000	0111	1 1000
0011	0 0011		
0101	0 0101		
1001	0 1001		
0110	0 0110		
1010	0 1010		
1100	0 1100		

$$children_{SBTF}(u, s) = \{(u_n u_{n-1}..\overline{u}_m..u_0)\} \quad \forall\ m \in M_{SBTF}(u, s).$$

$$parent_{SBTF}(u, s) = \begin{cases} \phi, & u = s; \\ (u_n u_{n-1}..\overline{u}_k..u_0), & u \neq s. \end{cases}$$

Figures 5.3 and 5.4 show a spanning tree generated by the children (or parent) function for FHC(4) and FHC(5), respectively. As can be seen, the height of SBTF is $\lceil n/2 \rceil$ which is approximately half the height of SBT [38], the broadcast tree for the n-cube. In the next theorem it shall be proven that SBTF covers all nodes in F.

Theorem 9: SBTF is a spanning tree of F.

Proof: Graphs G and F have the same set of vertices, namely V. For the proof, it is sufficient to show that all nodes covered by SBT (spanning tree of G_n) are also covered by SBTF (spanning tree of F_n). The graph model of FHC(n), namely F_n is symmetric. Therefore, without loss of generality, let s be the source node (i.e. the root of SBTF) such that $a(s)=0^n$. Consider a given node u covered by SBT in G_n with the address $u_{n-1}u_{n-2}..u_0$ such that $\|a(u)\| \leq \lceil n/2 \rceil$. Applying the transformation A to $a(u)$ yields the address $a_A(u)=0u_{n-1}u_{n-2}..u_0$. Assume k is the highest order bit in u which is one, and $M_{SBT}(u)=\{k+1,..,n-1\}$. Since the nth order bit of $a_A(u)$ is 0, thus k is also the highest order bit in $a_A(u)$ which is one, and $M_{SBTF}(u)=\{k+1,..,n-1,n\}$. Observe that $M_{SBTF}(u)=M_{SBT}(u) \cup \{n\}$, and every node u covered by SBT is also covered by SBTF if $\|a(u)\| \leq \lceil n/2 \rceil$. In addition, one of the children of node u in SBTF corresponding to $\{n\} \in M_{SBTF}(u)$ is the node u' which can be reached from node u by traversing link n. But traversal of link n results in complementation of all n bits in $a_A(u)$, i.e. $a_A(u')=1\overline{u}_{n-1}\overline{u}_{n-2}..\overline{u}_0$. Clearly, $\|a(u')\|=n-\|a(u)\| > \lceil n/2 \rceil$. The complement of node u uniquely specifies the node u'. Therefore, every node u' with $\|a(u')\| > \lceil n/2 \rceil$ is covered by SBTF. On the other hand, every node u' with $\|a(u')\| > \lceil n/2 \rceil$ is a leaf-node in SBTF since $a_A(u')=1\overline{u}_{n-1}\overline{u}_{n-2}..\overline{u}_0$ and $k=n$, and therefore $M_{SBTF}(u')=\phi$. Thus all nodes of F with weight i, $0<i \leq n$ are uniquely covered by SBTF and the proof is reached. □

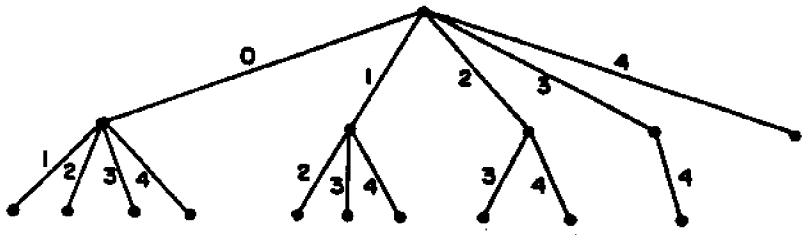


Figure 5.3. The Spanning Broadcast Tree (SBTF) for the FHC(4).

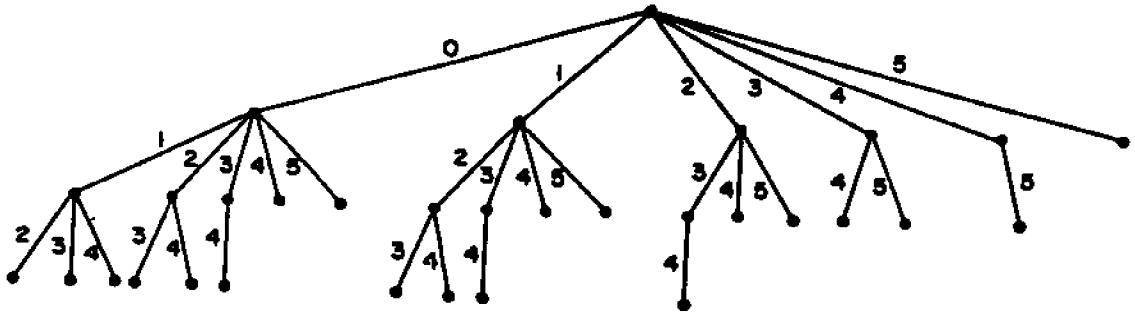


Figure 5.4. The Spanning Broadcast Tree (SBTF) for the FHC(5).

5.4.2.2 SBTF Performance

The number of steps required by SBTF broadcast algorithm is $\lceil n/2 \rceil$ which is approximately half the steps required by SBT algorithm. Now assume the communication is packet switched. Following the notation used in [38], let M denote the number of elements to be received by a node, t_c the transfer time for an element along a link, and τ the start-up time for the communication of a packet of maximum B elements. The broadcast time such obtained for SBT and SBTF, assuming all $\lceil M/B \rceil$ packets are pipelined through the cube, will be:

$$t_{SBT} = (\lceil M/B \rceil + n - 1) (\tau + Bt_c)$$

$$t_{SBTF} = (\lceil M/B \rceil + \lceil n/2 \rceil - 1) (\tau + Bt_c)$$

Clearly, the response time is still reduced by about 50% in SBTF. Nevertheless, for $\lceil M/B \rceil \gg n$, SBTF does not do much better than SBT. To improve the broadcast time for long messages, the concept of Multiple Broadcast Tree for FHC (MSBTF) will be introduced next. MSBTF is similar to MSBT [18], and efficiently employs all links in the FHC network to achieve optimal broadcast time.

5.4.2.3 Multiple Spanning Binary Tree for the FHC (MSBTF)

The MSBTF is a directed graph composed of $n+1$ SBTF's with each subtree rooted in each of the nodes adjacent to the source node. The root of each subtree is reached by any one of the links $0,1,..,n-1,n$. Moreover, each link appears twice in MSBTF implying bidirectional communication among nodes. The construction of MSBTF is very similar to that of MSBT. Each subtree rooted at a node with incoming link i (which is also incident on the root of MSBTF) is constructed from the subtree rooted at the node with incoming link 0 by right shifting all the link labels i times cyclically in the sequence $(0, 1, .., n-1, n)$. Figure 5.5 shows the MSBTF for a 4-cube. Note that data flow is always toward the leaf nodes and therefore all link arrows are eliminated. Every node in MSBTF is repeated exactly $n+1$ times. Thus if at each cycle, $n+1$ different packets of data are fed through the cube with all nodes' ports working concurrently, a speedup of $n+1$ over SBTF is achievable.

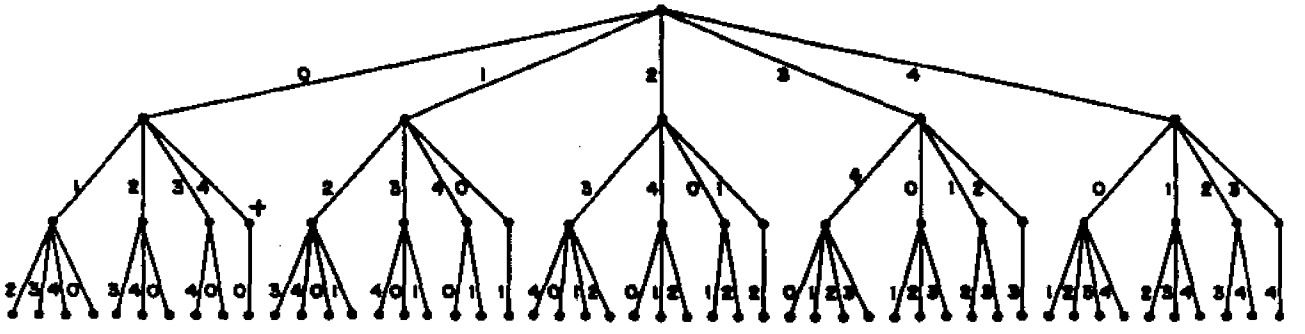


Figure 5.5. The Multiple Spanning Broadcast Tree (MSBTF) for FHC(4).

As an example in Figure 5.5 every node is repeated exactly 5 times once in each subtree. The node marked in the figure can be reached via the following paths:

Path	0→4	in the 0th subtree
Path	1→2→3	in the 1st subtree
Path	2→3→1	in the 2nd subtree
Path	3→1→2	in the 3rd subtree
Path	4→0	in the 4th subtree

To formally define the functions associated with MSBTF, the previously defined mapping scheme A is employed. For simplicity, it is assumed the root to be at the address 0^n. Let u be an arbitrary node with the address $(u_n u_{n-1} \cdots u_0)$ and k be such that $u_k=1$, and $u_m=0$, $\forall\ m \in M_{MSBTF}(i,j)$ where

$M_{MSBTF} = \{(k+1) mod (n+1),(k+2) mod (n+1),...,(j-1) mod (n+1)\}$. Hence, bit k is the first bit to the right of bit j $(0 \le j \le n)$ cyclically which is equal to one, if $k \ne j$. For the special case of $i=0$, define $k=-1$. For the jth subtree in the MSBTF graph with source node 0, the set of child nodes of node u is defined as follows:

$children_{MSBTF}\ (u,j,0) =$

$$\begin{cases} (u_n u_{n-1}..\overline{u}_j..u_0), & \text{if } k=-1; \\ \{(u_n u_{n-1}..\overline{u}_m..u_0)\}, & \\ \forall\ m \in M_{MSBTF}(i,j), & \text{if } u_j=1,\ k=j; \\ \{(u_n u_{n-1}..\overline{u}_m..u_0)\}, & \\ \forall\ m \in M_{MSBTF}(i,j) \cup j, & \text{if } u_j=1,\ k \ne j; \\ \{(u_n u_{n-1}..\overline{u}_m..u_0)\}, & \\ \forall\ m \in M_{MSBTF}(i,j) \cup \{j\} - \{j-1\}, & \text{if } u_j=1,\ k \ne j,\ \|a(u)\| = \lceil n/2 \rceil,\ n \text{ is odd}; \\ \phi, & \text{if } u_j=0,\ k \ne -1. \end{cases}$$

$$parent_{MSBTF}(u,j,0) = \begin{cases} \phi, & \text{if } k=-1; \\ (u_n u_{n-1}..\overline{u}_j..u_0), & \text{if } u_j=0,\ k\neq -1; \\ (u_n u_{n-1}..\overline{u}_k..u_0), & \text{if } u_j=1. \end{cases}$$

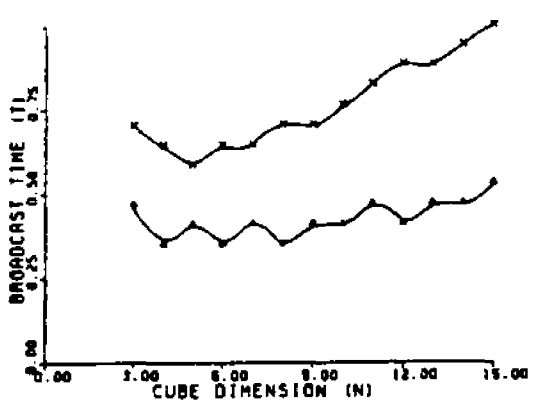
5.4.2.4 MSBTF Performance

With all ports working concurrently, $(n+1)$ different packets can be sent from the source in each cycle. It thus turns out that:

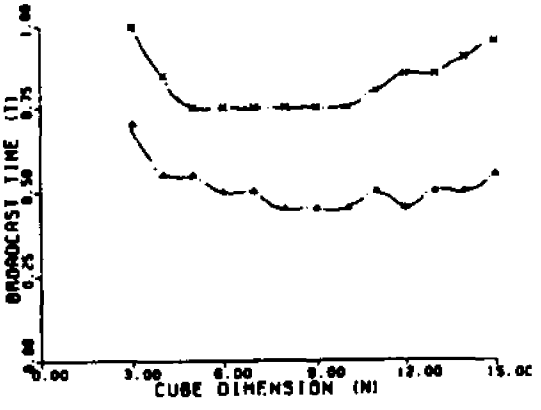
$$t_{MSBT} = \left(\left\lceil \frac{M}{Bn} \right\rceil + n \right)(\tau + Bt_c) \qquad [38]$$

$$t_{MSBTF} = \left(\left\lceil \frac{M}{B(n+1)} \right\rceil + \lceil n/2 \rceil \right)(\tau + Bt_c)$$

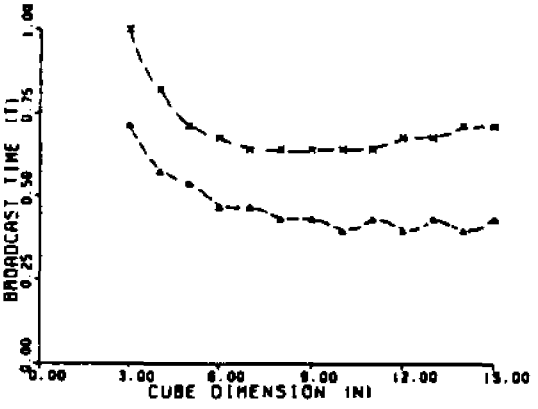
Figure 5.6 depicts the variation of broadcast time under MSBT and MSBTF schemes vs. the dimensionality of the cube and also the number of packets to be broadcasted. It is readily seen that upto 50% improvement is achieved in broadcast time indicating the superiority of the FHC over n–cube.



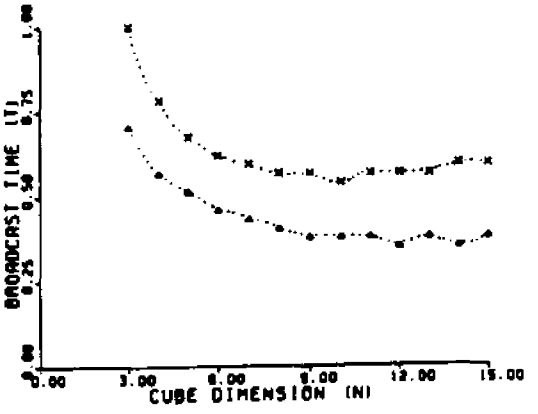
(a)



(b)



(c)



(d)

Figure 5.6. Comparison of Broadcast Time for n-cube and FHC(n).

n-cube: × FHC(n): Δ

(a) $\lceil M/B \rceil$: 25 (b) $\lceil M/B \rceil$: 50

(c) $\lceil M/B \rceil$: 75 (d) $\lceil M/B \rceil$: 100

5.5 Evaluation of the FHC Parameters

The network parameters derived for the n–cube in Chapter 3 are evaluated for the FHC in this section. For each parameter, the performance of the FHC is compared to that of the n–cube.

5.5.1 Average Distance ($\bar{d}$).

The average distance for the FHC can be easily obtained using the result of Theorem 7.

$$\bar{d} = \frac{\sum_{i=1}^{i=\lceil n/2 \rceil} i\,|N_i|}{N - 1}$$

The variation of $\bar{d}$ vs. n is plotted in Figure 5.7. For a wide range of n the average distance in FHC is at least 15% less than that of the n–cube.

5.5.2 Diameter (d).

By Theorem 6 the diameter of the FHC network is $\lceil n/2 \rceil$ which is half the diameter of the n–cube.

5.5.3 Cost (ξ).

By Theorem 5 the degree of the node for the FHC is $n+1$ and by Theorem 6 the diameter of this network is $\lceil n/2 \rceil$. Thus,

$$\xi = d\,(n+1) = \lceil n/2 \rceil\,(n+1)$$

The plot of ξ vs. n is shown in Figure 5.8. The cost factor for FHC is appreciably less than that of the n–cube and tends to half for large n.

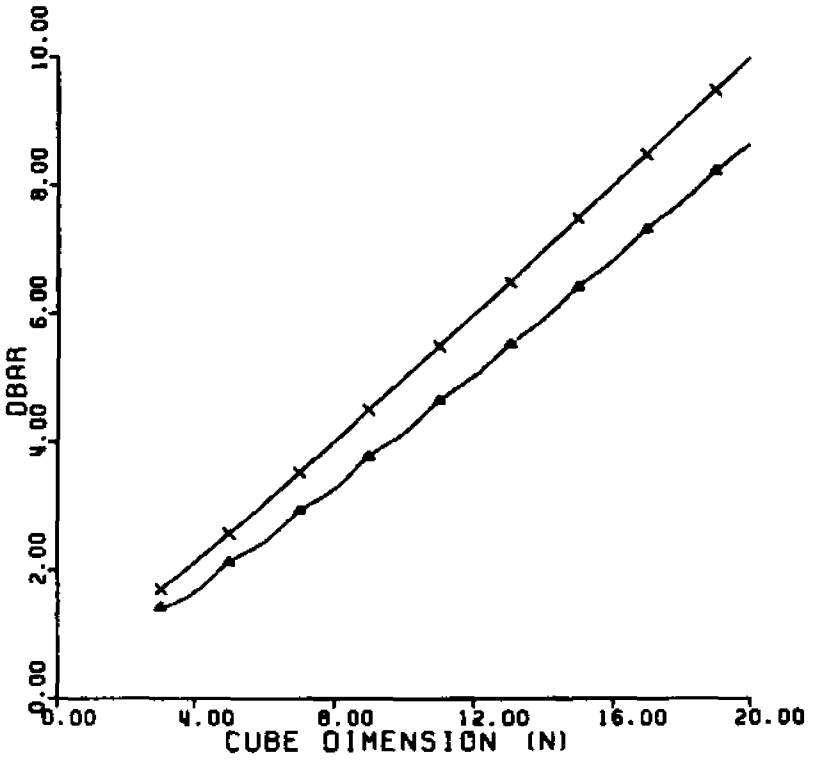


Figure 5.7. Comparison of Average Distance ($\bar{d}$)

n-cube: × FHC(n): Δ

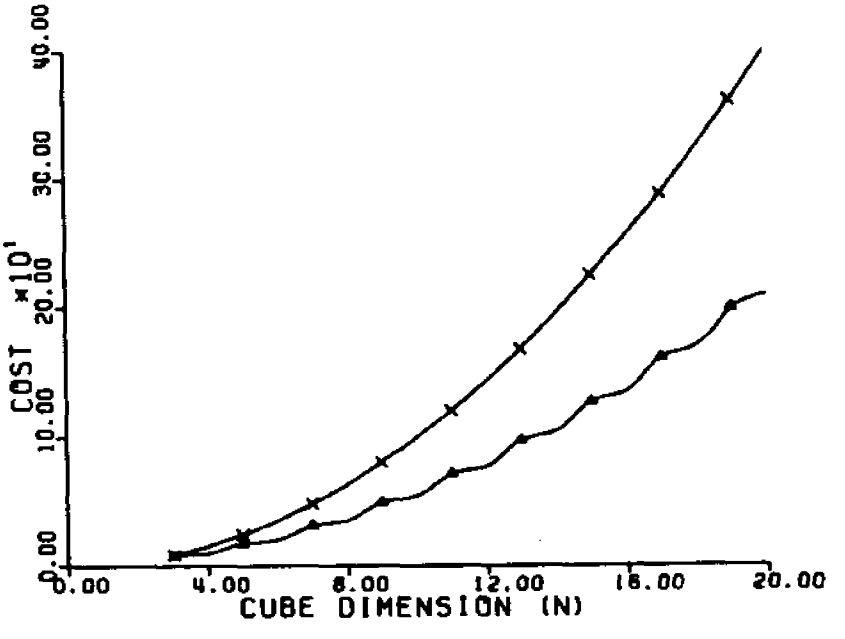


Figure 5.8. Comparison of Cost (ψ)

n-cube: × FHC(n): Δ

5.5.4 Message Traffic Density (ρ).

Using the same notation as in Chapter 3, for the FHC it follows:

$$\rho = \frac{\bar{d}\,N}{(n+1)N/2} = 2\,\frac{\bar{d}}{n+1}$$

Figure 5.9 depicts the variation of ρ vs. n for both structures. While this measure is 1 for the n–cube, for FHC it is close to 0.75 and increases slightly by n. Nevertheless, ρ does not exceed 0.80 even for large systems with $n = 20$ ($N > 10^6$).

5.5.5 Average Message Delay (T).

Under conditions specified in Chapter 3 and adopting the same notation, the delay of the FHC is given by:

$$T = \frac{\bar{d}(\frac{N}{2})2(n+1)}{\mu C\,(1-\bar{d}\,\Gamma)}$$

With constants μ,C, and N, the above delay can be normalized as

$$T' = \frac{\bar{d}(n+1)}{1-\bar{d}\,\Gamma}.$$

Figure 5.10 shows the variation of T' vs. Γ for both n–cube and FHC. Note that despite the increase in number of links in the FHC, the total capacity of the network (C) is kept constant. As utilization grows (higher traffic), time delay increases exponentially in the n–cube but very slightly in FHC. The improvement of T' in FHC for heavy loads (large Γ's) is significant.

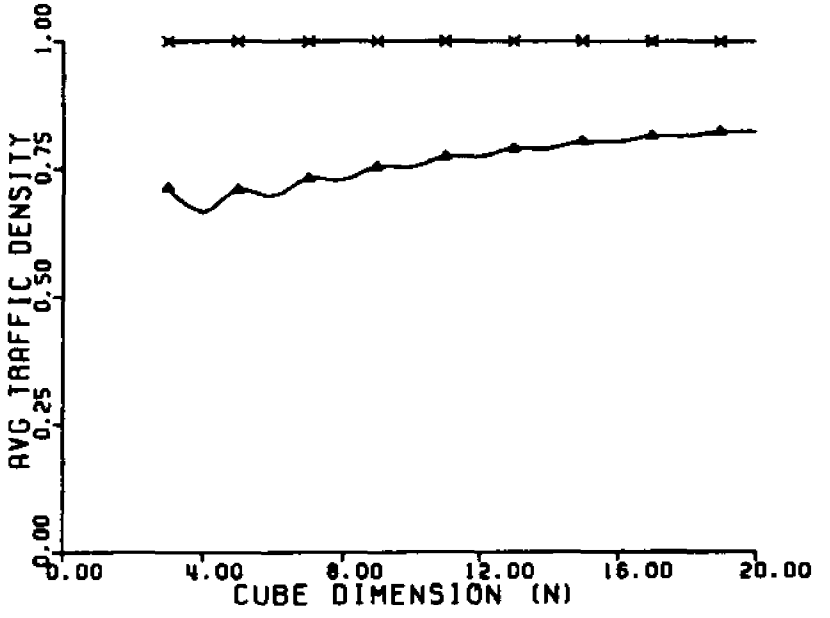


Figure 5.9. Comparison of Message Traffic Density (ρ)

n-cube: $\times$ FHC(n): Δ

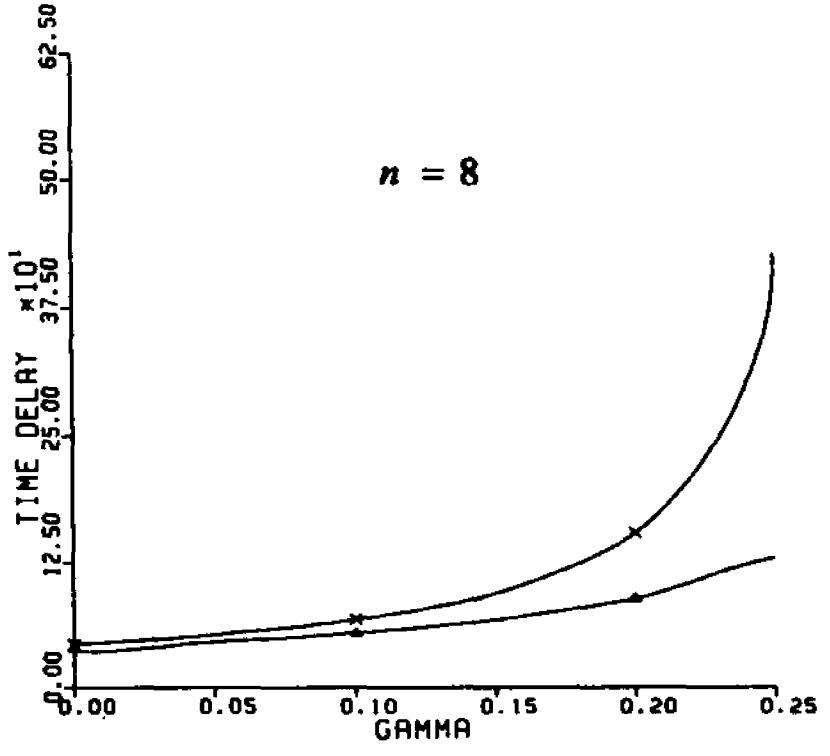


Figure 5.10. Comparison of Average Message Delay (T')

n-cube: $\times$ FHC(n): Δ

5.6 Fault Tolerance Capabilities of FHC

5.6.1 Connectivity

By Theorem 5, the connectivity of FHC is $n+1$. This implies that up to n nodes can fail without disrupting the network. As such, connectivity does not seem to reflect the practical fault tolerance capability of a network. The only way to remove n nodes of the n–cube and disconnect the network is to remove all the n neighbors of any one node - a very unlikely event. Similarly, all $n+1$ neighbors of any one node in FHC (n) must fail to disrupt the network. Denote such events by A_S and A_F for the n–cube and FHC, respectively. Also, let $P(X)$ represent the probability of occurrence of event X. Assuming a fault set of n nodes in an n–cube, there exist $\binom{2^n}{n}$ different fault sets out of which only 2^n result in disconnecting the network. Each of these sets is simply the set of all neighbors of a particular node. The same argument can be applied to the FHC network. It follows that,

$$P(A_S) = \frac{2^n}{\binom{2^n}{n}} \quad , \quad P(A_F) = \frac{2^n}{\binom{2^n}{n+1}}$$

and from the above,

$$P(A_F) = \frac{n+1}{2^n - n} P(A_S) \approx \frac{n}{2^n} P(A_S)$$

Based on the above expression, one may conceivably believe that even though the connectivity of the FHC is only slightly better than that of the n–cube, the probability of disrupting the FHC network, given a certain pattern of faulty nodes, seems to be far less than that of the n–cube subjected to the same fault pattern. The above analysis is not conclusive, however, since it only considers the special case, case of n–faults.

5.6.2 Two-Terminal Reliability [84]

This factor (also referred to as path reliability) is an important criterion to evaluate the reliability of a communication network. The Two-Terminal Reliability between two nodes u and v is defined as the probability of finding a path entirely composed of

operational edges between u and v. The aim is to determine this factor for both n–cube and FHC. Unfortunately, for these two networks, the number of paths between two nodes may be quite large. Moreover, many paths may have one or more links in common making the reliability analysis intractable. Therefore, a lower bound on Two-Terminal Reliability shall be derived by considering a subset of all available paths between two nodes in the structure.

The problem is modeled as one of finding the probability of having a path between 2 given nodes in G and F corresponding to n–cube and FHC, respectively. To simplify the analysis, it is assumed that link failures are statistically independent and occur randomly in time. Denote the link operational probability by p and consider any two arbitrary nodes in G or F with Hamming distance of r (i.e. $r = \|a(u) \oplus a(v)\|$). The number of edge disjoint paths between u and v by Menger's Theorem [82] and for $n>2$ is:

a) n–cube

r paths of length r

$n-r$ paths of length $r+2$

b) FHC(n)

case (i) $0 < r \leq \lceil n/2 \rceil$

r paths of length r

$n+1-r$ paths of length $r+2$

case (ii) $\lceil n/2 \rceil < r \leq n$

$n+1-r$ paths of length $n+1-r$

r paths of length $n+3-r$

Clearly, a path of length l consists of l links and has the reliability of p^l. Define the following terms:

$A(i,j)$: The event of having at least one path operational out of i paths of length j

RT_s: The Two-Terminal Reliability in n–cube

RT_F: The Two-Terminal Reliability in FHC

$P[X]$: The probability of occurrence of X

To derive $P[A(i,j)]$, note that the probability of failure of all i paths of length j is $[\,1-p^j\,]^i$. Thus,

$$P[A(i,j)] = 1 - [\,1-p^j\,]^i$$

The Two-Terminal reliability for the n–cube ($n>2$) then can be derived as follows:

$$RT_S > P[A(r,r)] + P[A(n-r,r+2)] - P[A(r,r)]P[A(n-r,r+2)]$$

And for the FHC(n) ($n>2$), RT_F can be obtained as follows:

For $0 < r \le \lceil n/2 \rceil$:

$$RT_F > P[A(r,r)] + P[A(n+1-r,r+2)] - P[A(r,r)]P[A(n+1-r,r+2)]\ ;$$

and for $\lceil n/2 \rceil < r \le n$:

$$RT_F > P[A(n+1-r,n+1-r)] + P[A(r,n+3-r)] - P[A(n+1-r,n+1-r)]P[A(r,n+3-r)]$$

The Two-Terminal Reliability vs. r is plotted in Figure 5.11 for both n–cube and FHC. While the path reliability in the n–cube drops off as distance increases, this curve for FHC is an inverted bell with the minimum occurring at $\lceil n/2 \rceil$. This is because Two-Terminal Reliability is minimum for nodes with largest distance which is $\lceil n/2 \rceil$ in FHC.

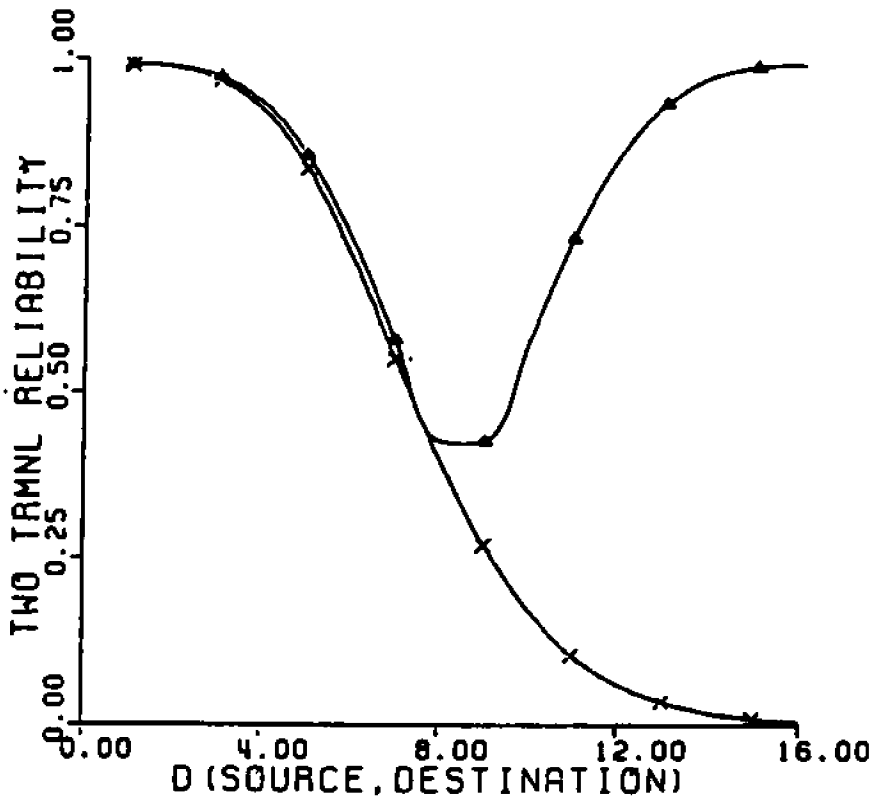


Figure 5.11. Two-Terminal Reliability (n=16 , p=0.6).
n–cube: × FHC(n): Δ

5.6.3 Container

In an n–cube, the width of container between two arbitrary nodes, namely u and v is n and the length of the container is $r+2$ where r is the Hamming distance between u and v. In the FHC between any two nodes, a container of width $n+1$ can be established. Even though for $0<r \leq \lceil n/2 \rceil$ the length of such a path is the same as that of n–cube, for $\lceil n/2 \rceil < r \leq n$ this length is at most $\lceil n/2 \rceil+2$. Figure 5.12 shows the container between two nodes in both n–cube and FHC(n) for $n=5$ and $r=4$. To quantify the quality of a container, a factor $CQ(u,v)$ (Container Quality) is attributed to the best container existing between nodes u and v. Due to the vertex symmetry property of n–cube and FHC, it suffices to designate CQ by r, where r is the Hamming distance between u and v. Since short and wide containers are desirable, $CQ(r)$ can be formulated as follows:

$$CQ\ (r)\ =\ \frac{\textit{no. of node disjoint paths between two nodes of Hamming distance r}}{\textit{average length of all node disjoint paths between two nodes}}$$

High values of $CQ(r)$ are favourable. Considering the number of node-disjoint paths between two nodes discussed in the previous section for the n–cube and FHC(n), CQ(r) shall be evaluated as follows:

a) n–cube

$$CQ_S(r) \equiv \frac{n^2}{n(r+2) - 2r}$$

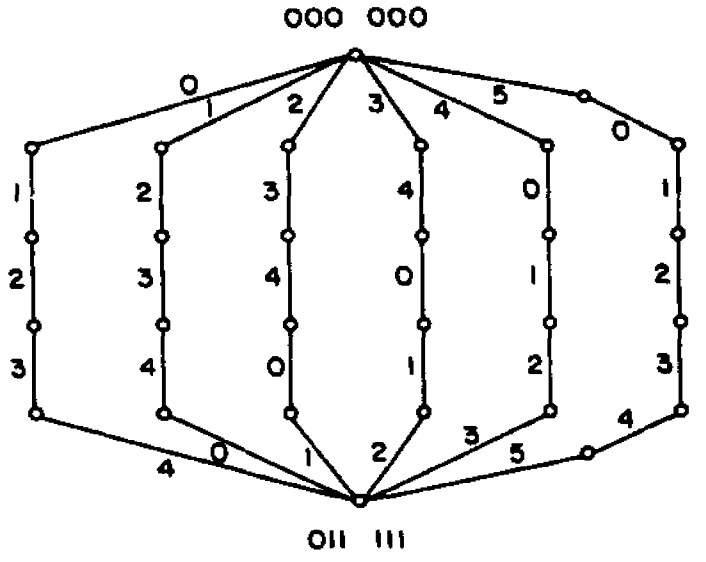
b) FHC

case (i) $0<r \leq \lceil n/2 \rceil$

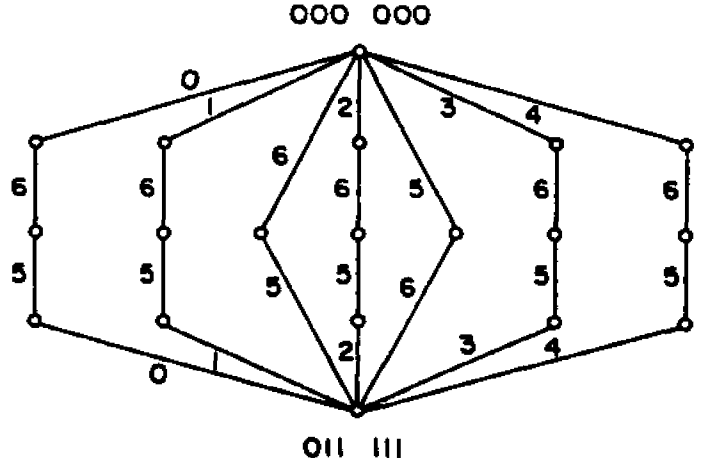
$$CQ_F(r) = \frac{(n+1)^2}{n(r+2) +2 - r}$$

case (ii) $\lceil n/2 \rceil < r \leq n$

$$CQ_F(r) = \frac{(n+1)^2}{(n+1-r)^2 + r(n+3-r)}$$



(a)



(b)

Figure 5.12. Container between two nodes ($n=6, r=5$).
(a) n-cube (b) FHC(n)

Note that in case (i), no significant improvement is achieved for CQ. However, in case (ii) the improvement is remarkable since the length of the container is reduced by using complementary edges. To demonstrate the improvement obtained for CQ, define:

$$CQ(r)\ Improvement \equiv \frac{CQ_F(r) - CQ_S(r)}{CQ_S(r)}$$

The variation of $CQ(r)$ *Improvement* vs. r is plotted for n=10 and n=20 in Figure 5.13.

5.6.4 f-Fault Diameter

As mentioned in Section 2.6.4, f –fault diameter, d_f, of a graph is defined to be the maximum of the distances over all possible graphs that can occur with at most f faults. For the FHC(n), $d_n = \lceil n/2 \rceil + 1$, and corresponds to a graph obtained from F_n after removing n neighbors of any node.

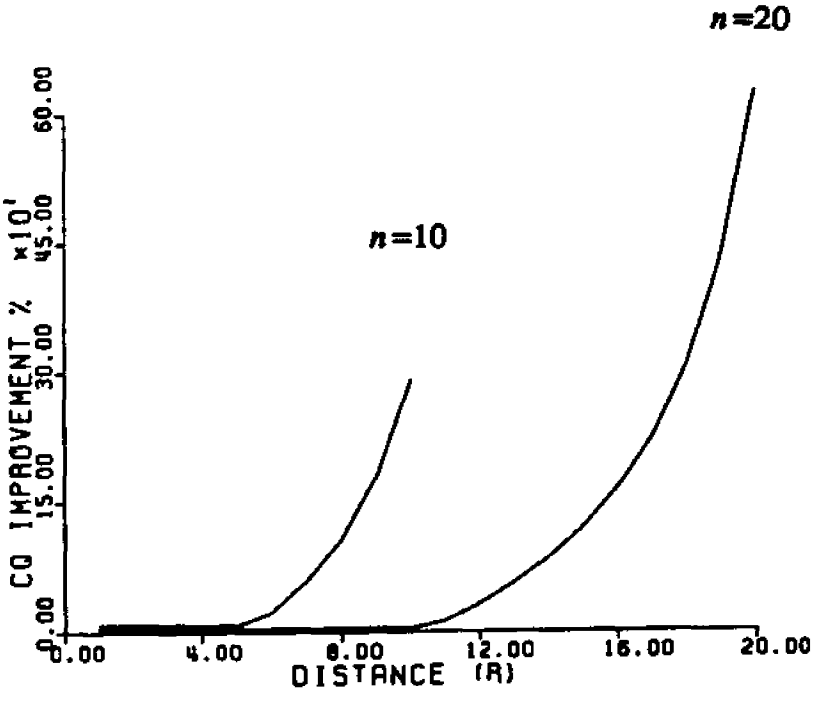


Figure 5.13. CQ Improvement (%) vs Distance between two Nodes.

5.7 Enhancing the Fault Tolerance of the Hypercube

An important criterion used to judge a computer network is its fault tolerance. In the n–cube, the high degree of connectivity prevents the nonfaulty processors from being disconnected (see Chapter 3). Moreover, the n–cube has the feature that if the number of faulty nodes does not exceed n, the diameter of the surviving network can be reasonably bounded [63]. But despite this high resilience, the universality of the hypercube may be lost if, as a consequence of the topology changes, the fundamental algorithms must be reorganized in a nonuniform and inefficient way.

Fortunately, most of the basic algorithms for universal networks, such as the n–cube, CCC, Butterfly or Shuffle-Exchange, have the property that they can be formulated with the dimension of the network as a parameter of the algorithm. More precisely, these algorithms can be performed on any nonfaulty $d-$ dimensional substructure with a slow down factor of 2^{n-d} (see the remarks about "limited parallelism" in [73] and the simulations between various universal networks in [17]). This motivates the question, how many dimensions may be lost, when the n–cube contains f faulty processors. Clearly, failure of a single node destroys one dimension. Failure of as few as two processors may destroy two dimensions (in this case the address of one failed node is the bitwise complement of the other's). A rough answer to the above question has been obtained by determining the bounds on the number of lost dimensions [64]. The objective in the following sections is to enhance the fault tolerance of the hypercube through adding some extra links such that fewer dimensions will be destroyed as a result of some node failures.

As mentioned earlier, the degradation in performance of the n–cube due to failure of a single component (link or node) is very high. For instance, if one node or link fails in an n–cube, the best one can do is to extract an operational $(n-1)$–cube from the damaged n–cube. This leaves approximately half the nodes and links of the network underutilized. This problem can be substantially alleviated by augmenting the n–cube with more interconnection links. The resultant network is referred to as *Fault Tolerant Hypercube*, or FTH. By offering many more available subcubes, the FTH

can utilize the network components more efficiently in a faulty environment. The following sections address the reliability and fault tolerance issues for the FTH, and illustrate the superiority of this network over the conventional standard hypercube.

5.8 The Structure of the Fault Tolerant Hypercube (FTH)

The FTH network is constructed from a standard hypercube by establishing spare links between each node and its farthest node. The spare link joining one node and its farthest node is called a *complementary link* or c–link. The number of c–links is clearly 2^{n-1} and thus the link redundancy is $1/n$. Note that from a topological point of view both FTH and FHC are exactly the same. The main difference between the two is that in the FTH, complementary links serve as spares and are not active in a fault free environment, whereas in the FHC these links are active as long as they are non-faulty.

Let $< x_0 , x_1 , ..., x_{n-1} >$ represent the dimensions of an n-cube. The FTH can be visualized as a network with $n+1$ dimensions $< x_0 , x_1 , ..., x_{n-1} , x_n >$ where x_n is the dimension corresponding to c–links. Therefore, all links along dimension x_i, called collectively *sheaf i*, connect nodes whose labels differ in the ith bit (for $0 \leq i \leq n-1$) or nodes whose labels are complements (for $i=n$).

Theorem 10: The *sheaf c* (set of c–links) can replace any single *sheaf i*, $0 \leq i \leq n-1$, in the FTH to yield an n–cube.

Proof: Consider an n–cube whose nodes are assigned binary numbers from 0 to 2^n-1. Moreover, a link is labeled i if it connects two nodes whose labels differ only in the i–th bit. The n–cube can be visualized as two subcubes of dimension $n-1$ joined by sheaf i ($0 \leq i \leq n-1$). Each node of one subcube is connected to a unique node in the other subcube via a link labeled i. Now assume some links of sheaf i are broken (i.e. faulty). The n–cube can be split into two $(n-1)$–cubes by removing all links of sheaf i. Now the set of c-links is introduced to the structure. Each link is incident on two nodes, say u and u', whose labels differ in all bits including the ith bit. Therefore, each of the 2^{n-1} c–links connects one node of one $(n-1)$–cube to a unique node of the other $(n-1)$–cube. It can be readily seen that the resultant network is an n–cube

with its i th sheaf replaced by sheaf c. □

This attractive feature can be exploited in the FTH to tolerate certain communication link failures. More specifically, if any single *sheaf i* is not operational, the new hypercube with dimensions $< x_0 , x_1 ,..., x_{i-1} , x_n , x_{i+1} ,..., x_{n-1} >$ can emulate the standard cube without any degradation in performance. When replacing a sheaf i by the sheaf n, a transformation T must be performed on the set of labels. For a given node u with label $u_{n-1}u_{n-2}..u_i..u_1u_0$, T can be specified as follows:

$$T(u_{n-1}u_{n-2}..u_i..u_1u_0) = \begin{cases} u_{n-1}u_{n-2}..u_i..u_0 & \text{if } u_i = 0 \\ \overline{u}_{n-1}..\overline{u}_{i+1}u_i\overline{u}_{i-1}..\overline{u}_0 & \text{if } u_i = 1 . \end{cases}$$

Figure 5.14 shows a 3-cube along with its augmented network (FTH) in which sheaf c has replaced sheaf 1.

5.9 Analysis of Communication link Reliability in FTH

To simplify the analysis, it is assumed that link failures are statistically independent and occur randomly in time. Let p be the reliability of a link at a given instance. The standard n-cube is composed of n sheaves each sheaf having 2^{n-1} links. Denote the reliability of each sheaf by R_{sh}. Thus

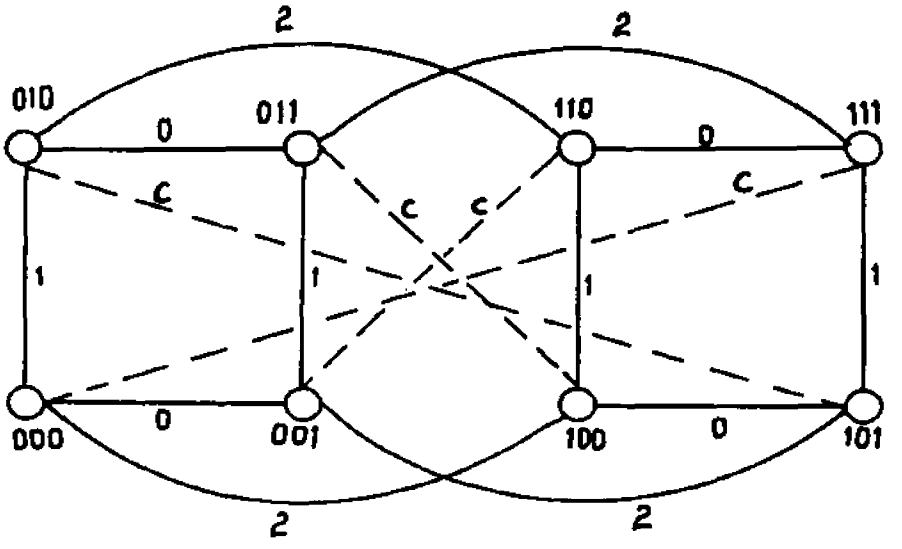
$$R_{sh} = (p)^{2^{n-1}}$$

On the other hand, all n sheaves must be operational in the standard hypercube whereas in the FTH, any n operational sheaves (out of $n+1$) will do. Denoting the collective link reliability of the standard and the FTH by R_s and R_a respectively, it follows:

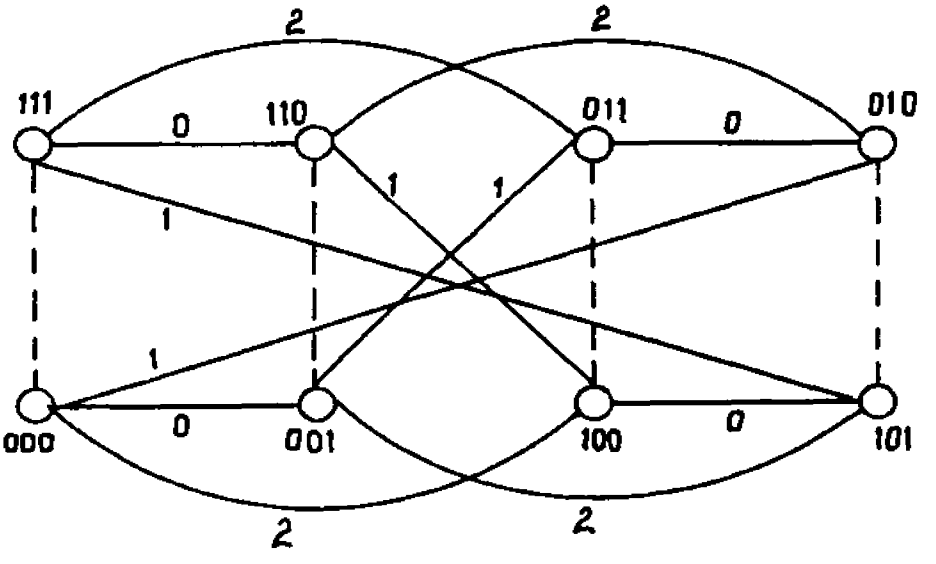
$$R_s = R_{sh}{}^n$$

$$R_a = \binom{n+1}{n} R_{sh}{}^n(1 - R_{sh}) + R_{sh}{}^{n+1}$$

The link reliability vs. cube dimension is plotted in Figure 5.15, for both standard and the FTH, and for p=0.99. The plot indicates the superiority of the augmented cube over the standard cube.



(a) All Sheaves are operational.



(b) Sheaf 1 is replaced by Sheaf c .

Figure 5.14. Link Replacement in the FTH(3).

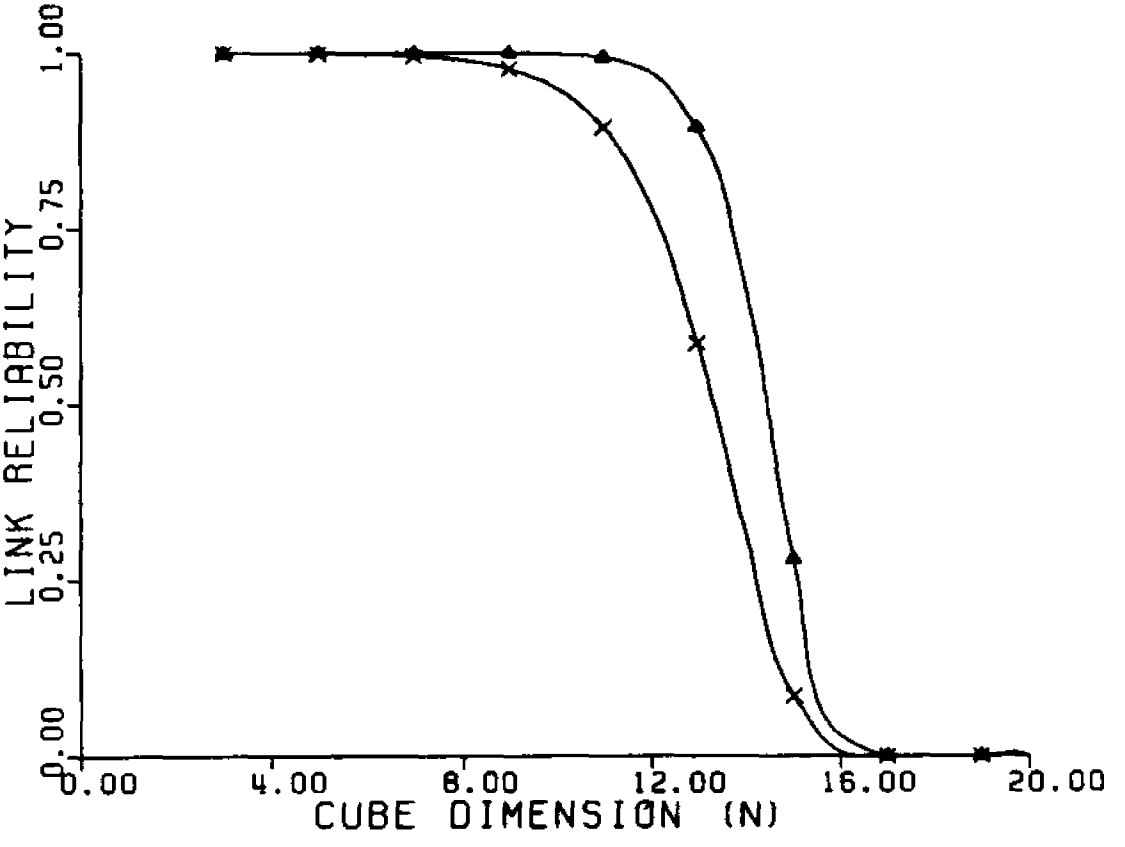


Figure 5.15. Collective Link Reliability (p=0.99).

n–cube : × FTH(n) : Δ

5.10 Subcube Reliability of the FTH

Partitioning a fault free hypercube into subcubes that can be allocated to different tasks is a common practice [85],[86]. Partitioning also provides a good measure of the ability of a faulty structure to degrade gracefully. Intuitively, the FTH should offer a better fault resilience compared to the n–cube, since the robustness of the FTH is improved due to c–links. The following lemmas provide a quantification of the ability of these networks to embed subcubes of various dimensions in a fault free environment.

Lemma 4: The number of k–cubes in any n–cube is $\binom{n}{k} 2^{n-k}$, where $0<k<n$.

Proof: Each k–cube can be represented by an n–tuple where $(n-k)$ bits are fixed and the remaining k bits specify the dimensions spanned by the k–cube. For a given set of k bit positions, there exist 2^{n-k} different ways to label the remaining $(n-k)$ bits. On the other hand, there are $\binom{n}{k}$ ways to choose k bits out of n bits. Thus the proof is reached. □

Lemma 5: The number of k–cubes in any FTH(n) is $\binom{n+1}{k} 2^{n-k}$, where $0\le k<n$.

Proof: By Theorem 10, the set of spare links or sheaf c can replace any set of links in a given dimension. This means that in the FTH, there exist $(n+1)$ dimensions from which one can choose the k dimensions required to construct a k–cube. Thus the proof follows. □

From Lemmas 4 and 5, the number of k–cubes in the FTH(n) is $\alpha_k = \frac{n+1}{n-k+1}$ times the number of k–cubes in the n–cube. Observe that $\alpha_n = n+1$. This implies that there exist $(n+1)$ n–cubes in the FTH(n). While all these n–cubes share the same set of nodes, they are distinguished by the n sheaves they possess. More formally, the ith n-cube possess all the sheaves $< x_0 , x_1 , .. , x_n >$ except for the sheaf x_i, where $0\le i\le n$. The significance of this property of the FTH lies in the fact that there is no degradation in network performance if one link fails. In case of a node failure, degradation is unavoidable for there is no spare to replace the failed nodes.

Observe that α_k increases with k. For instance, there are twice as many 2-cubes in the FTH(3) as there are in the 3-cube. This observation is important since in case of some node failures in the system, the FTH degrades to a cube of lower dimension more gracefully than does the n–cube. Next a node failure model is applied to both n–cube and FTH, and the the subcube reliability of these structures are comparatively analyzed.

5.10.1 A Node Failure Model

As mentioned previously, a single node failure in an n–cube destroys one dimension. Subsequent node failures may or may not destroy further dimensions depending on the fault sites. In the worst case, if the node at the opposite corner of the first faulty node fails, two dimensions will be lost. Given an arbitrary set of node failures, a fault free $(n-1)$-subcube exists if and only if all the faulty nodes can fit in an i–cube, $i<n$ (Necessity follows from the above example ; sufficiency becomes obvious when considering that an n–cube may be divided into two disjoint $(n-1)$-cubes and the faulty nodes placed totally within one of the subcubes).

The expression for subcube reliability of an n–cube has been derived by Abraham and Padmana [65] using a node failure model based on Markov chain. The same model can be applied to the FTH in order to determine the subcube reliability of this network. But before doing so, the derivation of the reliability expression as well as the mean time to failure [65] are briefly reviewed. Let S_i be the system state in which all node failures in the cube are contained in an i–cube, but not in an $(i-1)$-cube for $0 \le i < n$. In terms of functional subcubes, state S_i can be characterized as embedding exactly $n-i$ disjoint subcubes of order $n-1, n-2, .., i$. This sequence of functional subcube sizes is unique if one insists on a maximal disjoint subcube at each point in the sequence (Note that while the size sequence is unique, there may be many ways to generate a subcube of a particular size). For example, the functional subcubes existing in an 8-cube are shown in Figure 5.16, where a cube is represented as a rectangle. In the figure, the set of faulty nodes are contained in a 4-cube and, therefore, the system is in state S_4.

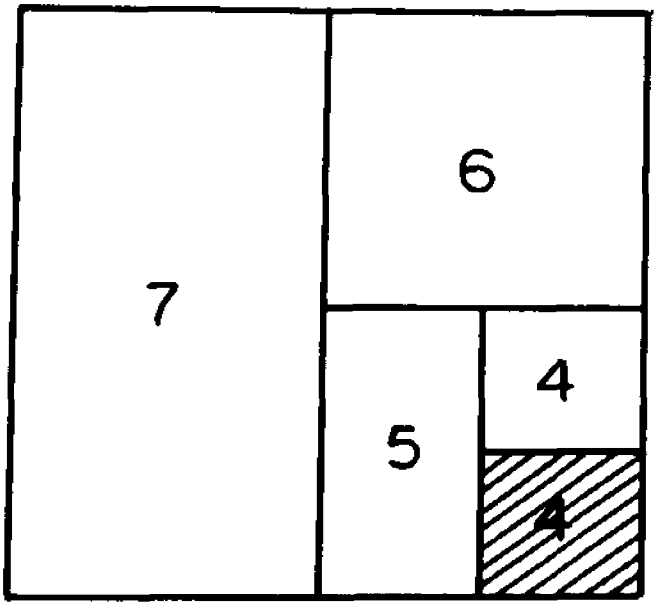


Figure 5.16. Functional Subcubes existing in an 8-cube.
(each box and each number represent
a cube and its dimension respectively).

In state S_n no embedding of a functional $(n-1)$-cube is possible. This state may be defined as the fatal case since no operational $(n-1)$-cube can be obtained in this state. Finally state S_G represents the initial fault free state of the n-cube. All nodes are assumed to have an identical exponential failure distribution with constant rate λ.

The transition from S_i to S_{i+j} $(0<j\leq n-i)$ occurs when an additional node has failed outside the damaged i-cube. In this case the new damaged cube will be of size $i+j$. To determine the transition rate, imagine the n-cube as split across i dimensions such that each node in the original damaged i-cube is in a separate partition. Each of the resultant 2^i partitions is an $(n-i)$-cube containing exactly one node from the damaged i-cube. The new failure must be in one of these partitions, say C. Further, all paths within partition C cross none of the i dimensions used during the splitting process. Therefore, if the new failure in C is distance j away from the node in C that belongs to the damaged i-cube, all damaged nodes in the system may be contained in an $(i+j)$-cube. There are $\binom{n-i}{j}$ such nodes in each C, leading to a transition rate of $\lambda 2^i \binom{n-i}{j}$.

Let $P_G(t)$ be the probability that the system is in state S_G. Generally, denoting the probability of being in state S_i by $P_i(t)$, the state equations for the system are given by [65]:

$$\frac{\partial P_G(t)}{\partial t} = -\lambda N P_G(t)$$

$$\frac{\partial P_0(t)}{\partial t} = -\lambda(N-1)P_0(t) + \lambda N P_G(t)$$

$$\frac{\partial P_i(t)}{\partial t} = -\lambda(N-2^i)P_i(t) + \sum_{j=0}^{i-1} \lambda\, 2^j \binom{n-j}{i-j} P_j\,, \quad 0 < i \leq n.$$

The initial conditions are $P_G(0) = 1$, and $P_i(0) = 0$ for all i.

The cumulative probabilities $R_i(t)$, $0<i<n$, as

$$R_G(t) = P_G(t)$$

$$R_0(t) = P_0(t) + R_G(t)$$

$$R_i(t) = P_i(t) + R_{i-1}(t).$$

where $R_G(t)$ is the cumulative probability that the system is good at t (no failures up to t).

Thus, $R_i(t)$ is the probability that all node failures (if any) up to time t are contained within an i–cube, leaving $n-i$ functional subcubes of order $n-1,..,i$. Let T_i represent the system's mean time to failure corresponding to state S_i. Then T_i can be evaluated by integrating the expression for $R_i(t)$.

$$T_i = \int_0^\infty R_i(t)dt \quad , \quad 0 \le i < n.$$

The transition rates for the FTH can be obtained by deriving the related expressions for an FTH(3) and generalizing them to the case of FTH(n). But before doing so, a definition and a lemma are in order.

Definition: Consider an n–cube and an integer k, $0<k<n$. This cube can be envisaged as a k–cube whose vertices are each a cube of dimension $(n-k)$. The conversion of dimension from n to k in this manner is called the cube *compression*, and the resulting k–cube is referred to as the *compressed* cube. The nodes and links of the compressed cube are called *vertices* and *bundles*, respectively.

Lemma 6: An FTH(n) can be compressed to an FTH(k) by replacing each $(n-k)$-cube by a single vertex, and the set of 2^k links between any two adjacent vertices by a single bundle.

Proof: The proof is obvious.

Lemma 7: Consider any transition $S_i \rightarrow S_{i+j}$ in FTH(n) for $0 \le i < n$. Then $j \le \lceil \frac{n-i}{2} \rceil$.

Proof: Proof is by contradiction. Assume $j > \lceil \frac{n-i}{2} \rceil$, and compress the FTH($n$) to obtain an FTH($n-i$). Clearly, prior to the transition, one of the vertices in the ($n-i$)-cube is faulty (this corresponds to the faulty i-cube in the original cube). Therefore, the transition $S_i \rightarrow S_{i+j}$ in the n-cube is similar to the transition $S_0 \rightarrow S_j$ in the ($n-i$)-cube. (failure of one node of the n-cube is sufficient to make the vertex containing it faulty). But the latter transition must have occurred due to the failure of a vertex, j hops away from the first failed vertex. This is impossible since the diameter of the FTH($n-i$) is at most $\lceil \frac{n-i}{2} \rceil$. □

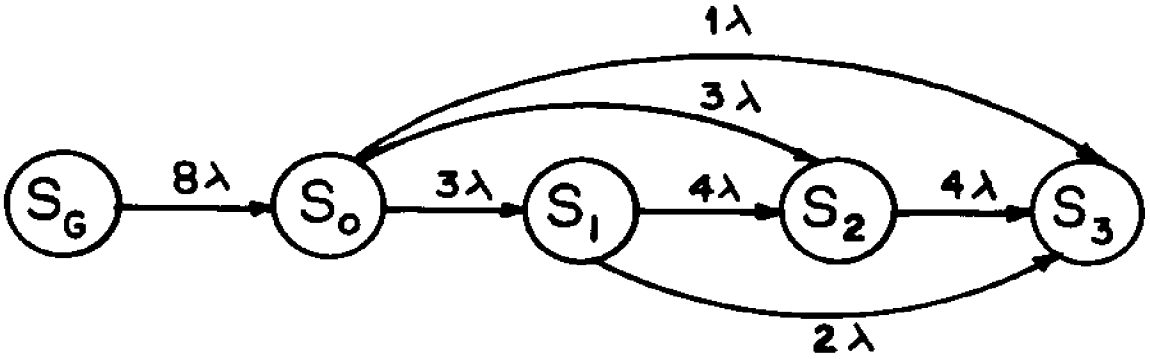
Now consider the transition $S_0 \rightarrow S_j$ for a given FTH(n). Since the topology of the FTH(n) is similar to that of FHC(n), according to Theorem 7 there are N_j nodes at distance j from the first failed node. Failure of any one of these nodes can result in the above transition. Therefore, the transition rate is λN_j. To derive the rates for the case $i > 0$, consider the following lemma.

Lemma 8: The rate for transition $S_i \rightarrow S_{i+j}$ in an FTH(n) is 2^i times that for transition $S_0 \rightarrow S_j$ in an FTH($n-i$).

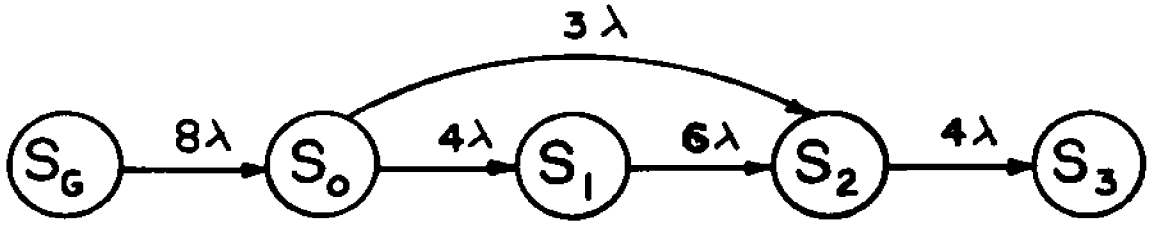
Proof: The FTH(n) can be compressed to form an FTH($n-i$). State S_i (a faulty i-cube) in the FTH(n) corresponds to state S_0 (a faulty vertex) in the FHC($n-i$). The transition $S_i \rightarrow S_{i+j}$ occurs if a node fails in such a way that j dimensions, other than those spanned by the faulty i-cube, are destroyed resulting in a faulty $i+j$-cube in the FTH(n). This resembles the transition $S_0 \rightarrow S_j$ caused by the failure of one vertex, j hops away from the first failed vertex in the FTH($n-i$). But there are 2^i nodes in each vertex which can make that vertex faulty. Thus, the proof is complete. □

The transition rates for any FTH(n) can be obtained using Lemmas 7 and 8. Figure 5.17 shows the state transition for both 3-cube and FTH(3). As an example, suppose nodes 000 and 001 in the FTH(3) are faulty (see Figure 5.1). Thus, the network is in State S_1 since the faulty nodes can be embedded in a 1-cube. Subsequent failure of any one of the remaining nodes will bring the network to State S_2. Note that transition $S_1 \rightarrow S_3$ is impossible according to Lemma 7. Table 5.2 summarizes the

transition rates for the FTH(n), $4 \leq n \leq 8$. The cumulative probabilities $R_i(t)$ for the FTH(n) can be obtained using the same approach as in the n–cube. However, it appears that no closed form for these terms can be obtained. Figure 5.18 shows the cumulative probabilities for both the n–cube and the FTH, for various n, under the node failure model. The T_{n-1}'s are shown in Table 5.3 for both structures, and the improvement in this factor is shown in Table 5.4. The theoretical results have been verified by Monte Carlo simulation.



3-cube



FTH(3)

Figure 5.17. State Transitions under Node Failure Model.

Table 5.2. Transition Rates for the Node Failure Model †.

Transition	$n=3$	$n=4$	$n=5$	$n=6$	$n=7$	$n=8$
$S_0 \to S_1$	$\binom{4}{1}$	$\binom{5}{1}$	$\binom{6}{1}$	$\binom{7}{1}$	$\binom{8}{1}$	$\binom{9}{1}$
$S_0 \to S_2$	$\binom{3}{2}$	$\binom{5}{2}$	$\binom{6}{2}$	$\binom{7}{2}$	$\binom{8}{2}$	$\binom{9}{2}$
$S_0 \to S_3$	-	-	$\binom{5}{3}$	$\binom{7}{3}$	$\binom{8}{3}$	$\binom{9}{3}$
$S_0 \to S_4$	-	-	-	-	$\binom{7}{4}$	$\binom{9}{4}$
$S_1 \to S_2$	$2^1\binom{3}{1}$	$2^1\binom{4}{1}$	$2^1\binom{5}{1}$	$2^1\binom{6}{1}$	$2^1\binom{7}{1}$	$2^1\binom{8}{1}$
$S_1 \to S_3$	$2^1\binom{3}{2}$	$2^1\binom{3}{2}$	$2^1\binom{5}{2}$	$2^1\binom{6}{2}$	$2^1\binom{7}{2}$	$2^1\binom{8}{2}$
$S_1 \to S_4$	-	-	-	$2^1\binom{5}{3}$	$2^1\binom{7}{3}$	$2^1\binom{8}{3}$
$S_1 \to S_5$	-	-	-	-	-	$2^1\binom{7}{4}$
$S_2 \to S_3$	$2^2\binom{1}{1}$	$2^2\binom{3}{1}$	$2^2\binom{4}{1}$	$2^2\binom{5}{1}$	$2^2\binom{6}{1}$	$2^2\binom{7}{1}$
$S_2 \to S_4$	-	-	$2^2\binom{3}{2}$	$2^2\binom{5}{2}$	$2^2\binom{6}{2}$	$2^2\binom{7}{2}$
$S_2 \to S_5$	-	-	-	-	$2^2\binom{5}{3}$	$2^2\binom{7}{3}$
$S_3 \to S_4$	-	$2^3\binom{1}{1}$	$2^3\binom{3}{1}$	$2^3\binom{4}{1}$	$2^3\binom{5}{1}$	$2^3\binom{6}{1}$
$S_3 \to S_5$	-	-	-	$2^3\binom{3}{2}$	$2^3\binom{5}{2}$	$2^3\binom{6}{2}$
$S_3 \to S_6$	-	-	-	-	-	$2^3\binom{5}{3}$
$S_4 \to S_5$	-	-	$2^4\binom{1}{1}$	$2^4\binom{3}{1}$	$2^4\binom{4}{1}$	$2^4\binom{5}{1}$
$S_4 \to S_6$	-	-	-	-	$2^4\binom{3}{2}$	$2^4\binom{5}{2}$
$S_5 \to S_6$	-	-	-	$2^5\binom{1}{1}$	$2^5\binom{3}{1}$	$2^5\binom{4}{1}$
$S_5 \to S_7$	-	-	-	-	-	$2^5\binom{3}{2}$
$S_6 \to S_7$	-	-	-	-	$2^6\binom{1}{1}$	$2^6\binom{3}{1}$
$S_7 \to S_8$	-	-	-	-	-	$2^7\binom{1}{1}$

† Each entry must be multiplied by λ.

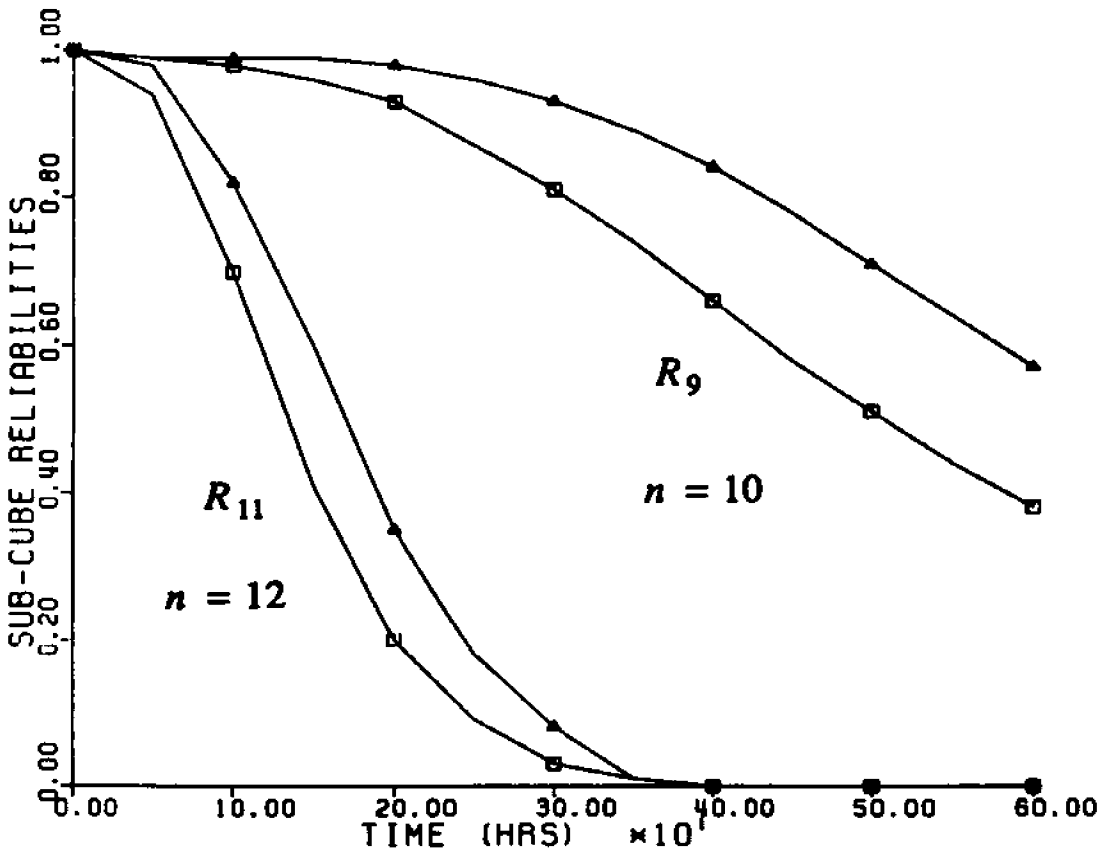


Figure 5.18. Cumulative Probabilities

n–cube : □ FTH : Δ

Table 5.3. Subcubes' Mean Time to Failure † ($\lambda = 10^{-5}/hrs$).

n	T_G	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7
4	6250	12917	14821	19107	28155				
	6250	12917	15298	22441	34941				
5	3125	6351	6888	8194	10460	14982			
	3125	6351	6996	8955	12142	18392			
6	1563	3150	3303	3726	4468	5611	7861		
	1563	3150	3329	3920	5220	6579	9592		
7	781	1569	1612	1750	2014	2400	2968	4094	
	781	1569	1619	1802	2224	2750	3528	5091	
8	391	783	795	839	934	1079	1272	1554	2117
	391	783	797	854	996	1257	1466	1882	2663

† In each entry, top number corresponds to the n-cube and the bottom one corresponds to FTH(n).

Table 5.4. Improvement in Mean Time to Failure in FTH(n) †.

n	T_{n-1} (n-cube)	T_{n-1} (FTH)	ΔT_{n-1} (%)
4	28155 (27718)	34941 (34200)	24.1 (23.4)
5	14982 (14153)	18392 (18531)	22.8 (30.9)
6	7861 (7923)	9592 (10020)	22.0 (26.4)
7	4094 (4234)	5091 (5555)	24.3 (31.1)
8	2117 (2288)	2663 (2860)	25.7 (25.0)
9	1090 (1190)	1363 (1438)	25.0 (20.8)
10	559 (621)	700 (788)	25.2 (26.9)
11	286 (318)	358 (405)	25.1 (27.4)
12	146 (160)	183 (206)	25.6 (28.8)

† The numbers in parenthesis are the simulation results.

5.11 Discussion

In this chapter, a new interconnection topology, the Folded Hypercube (FHC), has been introduced and investigated. The basic properties of this topology have been derived and compared to those of the standard n–cube. It has been shown that the folded hypercube has many of the appealing features of the n–cube such as node and edge symmetry. In addition, the folded hypercube has been shown to be superior to the n–cube in many communication aspects. In particular, halved diameter, better average distance, shorter delay in communication links, less message traffic density, and lower cost make this new structure very promising. Furthermore, the FHC outperforms the n–cube in terms of time complexity of its routing algorithms for one to one and broadcasting.

An FHC may be thought of as a fault tolerant hypercube augmented with some spare links. Communication link reliability in such a case is fairly improved. The presence of spare links makes an extra dimension available in the hypercube, resulting in the availability of many more subcubes in the network than those existing in the nonredundant hypercube. This attractive feature makes the fault tolerant hypercube degrade gracefully in a faulty environment.

The FHC suffers from two shortcomings, however. Unlike the standard hypercube, the FHC can not be partitioned such that the unit dilation (i.e. the distance between logically adjacent nodes) is preserved. Partitioning of the FHC can be only done virtually which results in some lengthy links. Also, a single node failure causes the FHC to degrade to i) a standard $(n-1)$-cube, or ii) a virtual FHC$(n-1)$. Depending on the application and the specific design goals, FHC's attractive features may indeed outweigh its drawbacks. Generally speaking, FHC offers the designer a viable design choice that cannot be overlooked in a wide range of multiprocessing environments.

CHAPTER 6

Conclusions and Future Directions

Devising a highly efficient network to interconnect many processing elements is an essential and nontrivial task. This work presented and investigated a class of hypercube-based networks for parallel processing applications. The efficiency of these networks has been improved by reducing the communication time among the processing elements. The networks in this class have one thing in common. They all can be obtained by selectively and systematically adding some extra links to the standard hypercube.

The Bridged Hypercube (BHC) can be established by introducing a small amount of link redundancy to the standard hypercube. The analysis of this network has been based on grouping the nodes according to their Hamming weights and then connecting two particular groups of nodes via extra links. The BHC can lend itself to algorithms which rely mainly on the short paths between nodes. For such algorithms, any graph with low diameter will suffice. In addition, basic routing algorithms (i.e. one to one and broadcasting) have been developed for the BHC and shown to be 50% faster than algorithms devised for the standard hypercube. The main drawback of the BHC, however, is its asymmetry. Therefore, algorithms developed for this network are more complex.

It has been further shown that some low-diameter structures employing smaller amount of redundancy can be utilized at the expense of more complicated routing algorithms. For instance, the Narrow Bridged Hypercube (NBHC) has been shown to have 1/4 as many extra links as BHC and a slightly larger diameter. The open problem in this regard is that of finding an optimal number of extra links and their positioning to halve the diameter of the hypercube.

The Folded Hypercube (FHC) introduced in Chapter 5 is an attractive network. Like the n–cube, the FHC has been shown to be vertex symmetric, edge symmetric, and strongly resilient. Beyond sharing these important network characteristics with the n–cube, the FHC has such additional virtues as: low diameter, low average distance, and low average message delay. One disadvantage of the FHC is that it cannot be partitioned into FHC's of lower dimension with dilation one. Nevertheless, partitioning can be done virtually resulting in some lengthy links in the structure.

The extra links in the FHC can serve as spares for the hosting hypercube. In such a case, the extra links have been shown to provide many subcubes within the original hypercube. For this reason, the FTH exhibits a graceful degradation in performance when components are prone to failures. In a fault free environment, where subcubes of the original cube are to be allocated to incoming tasks, the spare links can provide more available subcubes by introducing an extra dimension to the system. In this regard, an efficient algorithm for allocation and deallocation of the subcubes in a hypercube of given dimension is to be devised.

Clearly, there are many open problems on the FHC to be addressed. While some algorithms involving binary trees (i.e. broadcasting) have been developed to run on the FHC with the speedup of 50%, many other classical algorithms have not been addressed. Sorting, for instance, is a fundamental problem whose solution could well be useful in a variety of applications. Intuitively, this problem can be solved more efficiently on the FHC than on the n–cube due to the low diameter of the FHC. Load balancing is another important task which seemingly can be carried out faster on the FHC.

REFERENCES

[1] K.J. Thurber, "Interconnection networks- A survey and assessment," *Proceedings of the national computer conference*, AFIPS, vol. 43, pp. 909-919, 1974.

[2] G.A. Anderson and E.D. Jensen, "Computer interconnection structures: Taxonomy characteristics, and examples," *ACM Computing Survey*, vol. 7, No. 4, pp. 197-213, December 1975.

[3] H.J. Siegel, "Interconnection networks for SIMD machines," *IEEE Computer*, vol. 12, pp. 57-66, June 1979.

[4] K.E. Batcher, "Design of a massively parallel processor," *IEEE Transactions on Computers*, vol. c-29, pp. 836-840, September 1980.

[5] B.W. Arden and H. Lee, "Analysis of chordal ring network," *IEEE Transactions on Computers*, pp. 291-295, April 1981.

[6] L.D. Wittie, "Communication structures for large multimicrocomputer systems," *IEEE Transactions on Computers*, pp. 264-273, April 1981.

[7] H.J. Siegel and R.J. McMillen, "The multistage cube: a versatile interconnection network," *Computer*, vol. 14, pp. 65-76, December 1981.

[8] D.A. Reed and R.M. Fujimoto, "Multicomputer networks: message-based parallel processing," *MIT Press, Cambridge*, Mass., 1987.

[9] D. A. Reed and D. C. Grunwald, "The performance of multicomputer interconnection networks," *Computer*, pp. 63-73, June 1987.

[10] P.H. Enslow, Jr., "Multiprocessor organization-a survey," *ACM Computing survey*, vol. 9, pp. 103-129, March 1977.

[11] H.J. Siegel, R.J. McMillen, and P.T. Mueller, Jr., "A survey of interconnection methods for reconfigurable parallel processing systems," *Proceedings of the national computer conference, AFIPS*, vol. 48, pp. 387-400, 1979.

[12] H.J. Siegel and P.T. Mueller, Jr., "A survey of interconnection methods for reconfigurable parallel processing systems," *AFIPS Conference Proceedings 1979 National Computer Conference*, pp. 529-542, June 1979.

[13] K.J. Thurber, G.M. Masson, "Distributed-Processor communication architecture," Lexington, Mass. : Lexington Books, 1979.

[14] H.J. Siegel, Editor, "Special issue on interconnection networks for parallel and distributed processing, 124 pp., April 1980.

[15] T. Feng, "A survey of interconnection networks," *IEEE Computer*, vol. 14, pp. 12-27, December 1981.

[16] C.L. Wu, Editor, "Special issue on interconnection networks,"*IEEE Computer*, December 1981.

[17] H.J. Siegel, "Interconnection networks for large scale parallel processing," *Theory and Case Studies*, Lexington Books, 1985.

[18] S. Somasegar, "Methods for improving the simulation time of different types of interconnection networks," Master Thesis, LSU, December 1988.

[19] C.L. Seitz, "The Cosmic Cube," *Communication of the ACM*, vol. 28, pp. 22-33, Jan. 1985.

[20] P. Wiley, "A parallel architecture comes of age at last," *Computer*, pp. 46-50, June 1987.

[21] J. Tuazan, J. Peterson, M. Paniel, and D. Lieberman, "Caltech/JPL Mark II hypercube Concurrent Processor," *Proc. Int'l. Conf. on Parallel Processing*, pp. 666-673, 1985.

[22] J.C. Peterson, J.O. Tuazon, D. Lieberman, and M. Paniel, "The Mark III Hypercube-Ensemble Concurrent Processor," *Proc. Int'l. Conf. on Parallel Processing*, pp. 71-73, Aug. 1985.

[23] J.S. Squire and S.M. Palais, "Physical and logical design of a highly parallel computer," *Technical Note*, Dept. of Elec. Eng., Univ. of Michigan, Oct. 1962.

[24] J.S. Squire and S.M. Palais, "Programming and design considerations for a highly parallel computer," *Proc. Spring Joint Computer Conference*, pp. 395-400, 1963.

[25] H. Sullivan and T.R. Bashkow, "A large scale, homogeneous fully distributed parallel machine, I", *Proc. Computer Architecture Symposium*, pp. 105-117, 1977.

[26] H. Sullivan and T.R. Bashkow, "A large scale, homogeneous fully distributed parallel machine, II" *Proc. Computer Architecture Symposium*, pp. 118-124, 1977.

[27] M.C. Pease, "The indirect binary n-cube multiprocessor array," *IEEE Trans. on Computers*, vol. C-26, pp. 458-473, May 1977.

[28] G. Fox, "The performance of the Caltech hypercube in scientific calculation," *Caltech Report*, CALT-68-1298, April 1985.

[29] J.P. Hayes, T.N. Mudge, Q.F. Stout, "Architecture of a hypercube supercomputer," *Proc. Int'l Conf. on Parallel Processing*, pp. 653-660, 1986.

[30] J.J. Dongarra and I.S. Dutt, "Advanced Architecture Computers," *Technical Memorandum 57*, Argonne National Laboratory, 1985.

[31] NCUBE Corp., *NCUBE Handbook*, version 1.0, Beaverton, Ore., April 1986.

[32] W. D. Hillis, "The Connection Machine," *Cambridge*, Mass., MIT Press, 1985.

[33] Y. Saad and M.H. Schultz, "Topological properties of hypercubes," *Res. Rep.*, YALEU/DCS/RR-389, 1985.

[34] Y. Saad and M.H. Schultz, "Data Communication in Hypercubes," *Technical Report YALEU/DCS/RR-428*, Dept. of Computer Science, Yale University, Oct. 1985.

[35] S.L. Johnsson, "Communication Efficient Basic Linear Algebra Computing on Hypercube Architectures," *Technical Report YALEU/CSD/rr-361*, Dept. of Computer Science, Yale University, Jan. 1985.

[36] G.C. Fox and S.W. Otto, "Matrix algorithms on a hyperube I: Matrix multiplication," *Parallel Computing* 4, North-Holland, pp. 17-31, 1987.

[37] O.A. McBryan and E.F. Van de Velde, "Hyperucbe algorithms and implementations," *2nd SIAM Conference on Parallel Computing*, Nov. 1985.

[38] C.T. Ho and S.L. Johnsson, "Distributed Routing Algorithms for Broadcasting and Personalized Communication in Hypercubes," *Proc. Int'l Conf. on Parallel Processing*, pp. 640-648, Aug. 1986.

[39] Y.L. Lan, A. Esfahanian, and L.M. Ni, "Multicast in hypercube multiprocessors," *Proc. of the third Int'l Conf. on Supercomputing*, May 1988.

[40] S.R. Deshpande and R.M. Jenevein, "Scalability of a Binary Tree on a Hypercube," *Proc. Int'l Conf. on Parallel Processing*, pp. 661-668, 1986.

[41] S.N. Bhatt and I.C.F. Ipsen, "How to embed trees in hypercubes," *Res. Rep.*, YALEU/DCS/RR-443, December 1985.

[42] A.Y. Wu, "Embedding of tree networks into hypercubes," *Journal of Parallel and distributed computing 2*, pp. 238-249, 1985.

[43] M.Y. Chan and F.Y.L. Chin, "On Embedding rectangular grids in hypercubes," *IEEE Trans. on Computers*, vol. 37, no. 10, pp. 1285-88, Oct. 1988.

[44] E.M. Reingold, J. Nievergelt, N. Deo, *Combinatorial Algorithms.*, Princeton-Hall, Englewood Cliffs, NJ, 1977.

[45] S.L. Johnsson, "Communication efficient basic linear algebra computations on hypercube architectures," *Journal of parallel* and *distributed computing* 4, pp. 133-172, 1987.

[46] J.E. Brandenburg and D.S. Scott, "Embeddings of communication trees and grids into the hypercubes," *Int'l Scientific Comput. Report* 280182–001, 1985.

[47] T.F. Chan, F. Chin, *Private Communiations*, 1985.

[48] C.T. Ho and S.L. Johnsson, "On the embedding of arbitrary meshes in boolean cubes with expansion two, dilation two," *Proc. Int'l Conf. on Parallel Processing*, pp. 188-191, 1988.

[49] H. J. Siegel, "Interconnection Networks for Large Parallel Processing" Lexington, Mass.: Lexington Books, c 1985.

[50] S.L. Johnsson and C.T. Ho, "Algorithms for multiplying matrices of arbitrary shapes using shared memory primitives on boolean cubes", *YALEU/DCS/TR-569* October 1987.

[51] C.T. Ho and S.L. Johnsson, "Matrix multiplication on boolean cubes using shared memory primitives," *YALEU/DCS/TR*–636, July 1988.

[52] L.M. Ni, C.T. King, and P. Prins, "Parallel algorithm design considerations for hypercube multiprocessors," *Proc. Int'l. Conf. on Parallel Processing*, pp. 717-720, 1988.

[53] C.T. Ho and S. L. Johnsson, "Algorithms for matrix transposition on boolean n–cube configured ensemble architectures," *Proc. Int'l. Conf. on Parallel Processing*, pp. 621-629.

[54] C. Aykanat, F. Ozguner, F. Ercal, and P. Sadayappan, "Iterative algorithms for solution of large sparse systems of linear equations on hypercubes," *IEEE Trans. on Computers* vol. 37, no. 12, pp. 1554-68, Dec. 1988.

[55] S.L. Johnsson, C.T. Ho, M. Jacquemin, and A. Ruttenberg, "Computing Fast Fourier Transforms on boolean cubes and related networks," *YALEU/DCS/TR-598*, October 1987.

[56] S.L. Johnsson and C.T. Ho, M. Jacquemin, and A. Ruttenberg, "Systolic FFT algorithms on boolean cube networks," *YALEU/DCS/TR*–619, March 1988.

[57] C.T. Ho and S.L. Johnsson, "Stable dimension permutations on boolean cubes," *YALEU/DCS/TR*–617, October 1988.

[58] S.L. Johnsson and C.T. Ho, "Shuffle permutations on boolean cubes," *YALEU/DCS/TR*–653, October 1988.

[59] Y. Won and S. Sahni, "Maze routing on a hypercube multiprocessor computer," *Proc. Int'l Conf. on Parallel Processing*, pp. 630-637, 1988.

[60] J.F. Jenq and S. Sahni, "All pairs shortest paths on a hypercube multiprocessor," *Proc. Int'l. Conf. on'Parallel Processing*, pp. 713-716, 1988.

[61] A.R. Omondi and J.D. Brock, "Implementing a dictionary on hypercube machines," *Proc. Int'l Conf. on Parallel Processing*, pp. 707-709, 1988.

[62] J.R. Armstrong and F.G. Gray, "Fault diagnosis in a boolean n–cube array of multiprocessors," *IEEE Trans. on Computers*, vol. c-30, no. 8, pp. 587-590, Aug. 1981.

[63] M.S. Krishnamoorthy and B. Krishnammorthy, "Fault diameter of interconnection networks," *Comput. Math. Applications*, vol. 13, no. 5/6, pp. 577-582, 1987.

[64] B. Becker and H.U. Simon, "How robust is the n–cube?", *IEEE 27th Annual Symposium on Foundation of Computer Science*, pp. 283-291, 1986.

[65] S. Abraham and K. Padmanabhan, "Reliability of the hypercube multicomputers," *Proc. 1988 Int'l. Conf. on Parallel Processing*, pp. 90-93, August 1988.

[66] D.A. Rennels, "On implementing fault-tolerance in binary hypercubes," *IEEE Fault Tolerant Computing*, pp. 344-349, 1985.

[67] S. Latifi and A. El-Amawy, "Enhancing communication reliability in hypercube networks," *21st Southeastern Conf. on System Theory*, pp. 234-237, March 1989.

[68] S. Latifi and A. El-Amawy, "Nonplanar VLSI arrays with high fault tolerance capabilities," *IEEE Trans. on Reliability*, Special issue on Parallel and Distributed Systems, pp. 51-57, April 1989.

[69] M.S. Chen and K.G. Shin, "Message routing in an injured hypercube," *Proc. Third Conf. Hyperucbe Concurrent Comput. Appl.*, to be published.

[70] T.C. Lee and J.P. Hayes, "Routing and broadcasting in faulty hypercube computers," *Proc. Third Conf. Hypercube Concurrent Comput. Appl.*, to be published.

[71] F.J. Meyer and D.K. Pradhan, "Flip-Trees: Fault Tolerant Graphs with Wide Containers," *IEEE Transactions on Computers*, vol. 37, no. 4, pp. 472-478, April 1988.

[72] P. Ramanathan and K.G. Shin, "Reliable broadcast in hyperucbe multicomputers," *IEEE Transactions on Computers*, vol. 37, no. 12, pp. 1654-57, Dec. 1988.

[73] F.P. Preparata and J. Vuillemin, "The Cube-Connected Cycles: A versatile network for Parallel Computation," *Communication of the ACM*, vol. 24, no. 5, pp. 300-309, May 1981.

[74] L.D. Wittie, "Communication Structures for large networks of microcomputers," *IEEE Transactions on Computers*, vol. C-30, no.4, pp. 264-273, April 1981.

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

[76] P.W. Dowd and K. Jabbour, "Spanning multiaccess channel hypercube computer interconnection," *IEEE Transactions on Computers*, vol. 37, no. 9, Sep. 1988.

[77] G.H. Barnes, "The Illiac IV computer," *IEEE Transactions on Computers*, vol. C-17, pp. 746-757, Aug. 1968.

[78] K.V.S. Bhat, "On the properties of arbitrary hypercubes," *Comp. and Math. with Appl.*, vol. 8, no. 5, pp. 339-342, 1982.

[79] A.H. Esfahanian, L.M. Ni, and B.E. Sagan, "On enhancing hypercube multiprocessors," *Proc. Int'l Conf. on Parallel Processing*, pp. 86-89, 1988.

[80] F. Harary, "Graph Theory," *Addison Wesley,*, Reading, Mass., 1969.

[81] L. Kleinrock, "Queueing Systems: Vol. II, Computer Applications," New York: Wiley, 1976.

[82] J. Bondy and U. Murthy, "Graph Theory with Applications," New York: American Elsevier, 1976.

[83] S.B. Akers, D. Harel, and B. Krishnamurthy, "The Star Graph: An attractive alternative to the n-cube," *Proc. 1988, IInt'l. Conf. on Parllel Processing*, pp. 393-400.

[84] C.J. Colbourn, "The Combinatorics of Network Reliability," New York : Oxford, Oxford University Press, 1987.

[85] M.S. Chen and K.G. Shin "Embedment of interacting task modules into a hypercube multiprocessor," *Proc. Second Hypercube Conf.*, pp. 121-129, October 1986.

[86] M.S. Chen and K.G. Shin, "Processor allocation in an n-cube multiprocessor using gray codes," *IEEE Trans. on Computers*," vol. c-36, no. 12, pp. 1396-407, December 1987.

APPENDIX I

Proof of Theorem 4

Theorem 4: The diameter of NBHC(n, p) of dimension n is $n/2+3$.

Proof: Similar to the case of the BHC network, the NBHC network can be partitioned into groups I, II, III, and IV as defined before. The two nodes s and d can belong to any of these four groups resulting in 16 different cases. Distinguished cases, however, are those in which s resides in group I or II (d can be anywhere). The proof is carried out by showing that any arbitrary pair of nodes can communicate in at most $n/2+3$ hops. The communication between a pair of nodes takes place following an algorithm whose complexity is shown to be at most $n/2+3$. The basic idea in routing between two nodes is to arrive (when profitable) at the site of a c-link by changing the weight of the source to $n/4$ or $3n/4$, whichever takes fewer steps. However, care must be taken such that the pth bit of the source is changed to 1 (0) if it is already 0 (1) and if a node in $W_{n/4}$ ($W_{3n/4}$) is to be reached. To formally describe these algorithms some definitions are in order.

Let p be an integer where $0 \le p \le n-1$, and M be a set such that $M \in \{ M_{00}, M_{01}, M_{10}, M_{11} \}$. Define:

$$M^p = \begin{cases} M & \text{if } g_p \notin M \\ M-\{g_p\} & \text{if } g_p \in M \end{cases}$$

In addition, the following variables are defined:

$$\Delta \equiv \delta_3 + \delta_4 - n/4$$
$$\Delta' \equiv 3n/4 - (\delta_3 + \delta_4)$$
$$\varepsilon \equiv \Delta - \delta_4$$
$$\varepsilon' \equiv \Delta' - \delta_1$$

The routing algorithms also depend on the position of the pth bit. More specifically,

p may be such that g_p belongs to one of the sets: M_{00}, M_{01}, M_{10}, or M_{11}. Thus, for every subcase, all these four possibilities are considered.

Case (i): s in I and d anywhere.

Define $\Delta = n/4 - (\delta_3+\delta_4)$. The idea here is to increase the weight of s such that a node in $W_{n/4}$ is reached which is incident on a c–link. Two subcases are considered:

Subcase (a): $\Delta \leq \delta_1$

1 - $g_p \in M_{10}$ or $g_p \in M_{11}$ †.

The pth bit of s is already 1 and does not need to change when a node in $W_{n/4}$ is being reached. Thus, Algorithm 1 (see Chapter 4) can be employed and the time complexity is $NH \leq n/2$.

2 - $g_p \in M_{00}$

Algorithm 7.

input: s and d

Step 1: Complement the pth bit in M_{00}

Step 2: Complement $\Delta-1$ bits of M_{00}^p.

Step 3: Take the c–link.

Step 4: Correct the differing bits.

In Step 1, the pth bit is changed to 1. This will increase the weight of s by 1. Thus, in Step 2, $\Delta-1$ bits of M_{00} (rather than Δ) are complemented. At the end of Step 2, a node $s' \in W_{n/4}$ is reached which is incident on a c–link (since $s_p=1$. Thus, the c–link can be used in Step 3. Moreover, the time complexity of Algorithm 7 is the same as that of Algorithm 1. The only difference between Algorithm 1 and Algorithm 7 is that in the former any Δ bits in M_{00} can be complemented in the first step, whereas in the latter these Δ bits must include the pth bit. Thus $NH \leq n/2$.

---

† This also means that s_p (i.e. the pth bit of label s) is equal to 1.

3 - $g_p \in M_{01}$

Algorithm 8.

input: s and d

Step 1: Complement the p th bit in M_{01}

Step 2: Complement any $\Delta-1$ bits in M_{00}.

Step 3: Take the c –link.

Step 4: Correct the differing bits.

The c –link is available in Step 3, since the p th bit is changed to one in Step 1. By complementing the p th bit of M_{01} and $\Delta-1$ bits of M_{00}, a node $s' \in W_{n/4}$ is reached. At this point, $d_H(s',d)=(D\ 1-1)+\delta_2-1+\delta_3$. The c –link is then taken to arrive at $d' \in W_{3n/4}$, where $d_H(d',d)=n-(\Delta-1+\delta_2-1+\delta_3)$. Thus,

$NH=1+\Delta-1+1+n-(\Delta-1+\delta_2-1+\delta_3)$. Subject to the condition $\delta_2+\delta_3>n/2$, it follows: $NH \leq n/2+2$.

Subcase (b): $\Delta>\delta_1$

1 - $g_p \in M_{10}$ or $g_p \in M_{11}$

Algorithm 2 can be employed for this subcase. The c –link is available in Step 2, because the p th bit which is initially 1 does not change in the first step of the algorithm. Clearly, $NH \leq n/2$.

2 - $g_p \in M_{00}$

Here also, Algorithm 2 can be used. Since in Step 1 of the Algorithm 2 all δ_1 bits of M_{00} including the p th bit are changed to one's, the c –link is available in Step 2. Thus, the time Complexity is $NH \leq n/2$.

3 - $g_p \in M_{01}$

Algorithm 9.

input: s and d.

Step 1: Complement all δ_1 bits of M_{00} in s.

Step 2: Complement the p th bit

Step 3: Complement $\varepsilon-1$ bits of M_{01}.

Step 4: Take the c –link.

Step 5: Complement the differing bits.

Comparison of this algorithm with Algorithm 2 reveals that the time complexity of the two are the same. The only change in Algorithm 9 is that the p th bit is selectively complemented in the domain of M_{01}, in order to reach a node of $W_{n/4}$ which is incident on the bridge. This, however, does not affect the time complexity. Thus, $NH \leq n/2$.

Case (ii): s in II and d anywhere.

Subcase (a): $\Delta<\delta_4$

In this subcase, the weight of s is reduced to $n/4$ by changing some bits in its label from 1's to 0's. Nevertheless, the p th bit of the label must be 1 prior to using a c –link.

1 - $g_p \in M_{10}$ or $g_p \in M_{11}$

Algorithm 10.

input: s and d

Step 1: Complement any Δ bits of M_{11}^p in s.

Step 2: Take the c –link.

Step 3: Correct the differing bits one at a time.

This algorithm is similar to Algorithm 3. If $p \in M_{11}$, since $\Delta<\delta_4$, the p th bit can be sidestepped when complementing Δ bits of M_{11}. Thus the c –link will be available in Step 2. If $p \in M_{10}$, the p th bit remains the same during Step 1 and the c –link is available in Step 2. The complexity of Algorithm 10 is therefore the same as that of Algorithm 3, i.e. $NH \leq n/2$.

2 - $g_p \in M_{00}$

Algorithm 11.

input: s and d

Step 1: Complement the p th bit in M_{00}

Step 2: Complement $\Delta-1$ bits in M_{11}.

Step 2: Take the c –link.

Step 3: Correct the differing bits one at a time.

The c –link is available since the p th bit is changed to 1 in Step 1. Let s' be the node reached after Step 2. Thus, $d_H(s',d)=1+\Delta-1+\delta_2+\delta_3$. After the c –link is taken and node d' is reached, $d_H(d',d)=n-(d_H(s',d))$. Thus, the time complexity is: $NH=1+\Delta-1+1+n-(\Delta+\delta_2+\delta_3)$. After applying the constraint $\delta_2+\delta_3>n/2$ and some algebraic manipulations, it follows: $NH \leq n/2$.

3 - $p \in M_{01}$

Algorithm 12.

input: s and d

Step 1: Complement the p th bit in M_{01}

Step 2: Complement $\Delta-1$ bits in M_{11}.

Step 3: Take the c –link.

Step 4: Correct the differing bits one at a time.

The c –link is available, for the p th bit is complemented prior to Step 3. At the end of Step 1, a node s' is reached such that $d_H(s',d)=\Delta-1+\delta_3+\delta_2-1$. The number of remaining bits to be corrected in Step 4 is therefore, $n-d_H(s',d)$. Thus: $NH=1+\Delta-1+1+n-(\Delta-1+\delta_3+\delta_2-1)$. But $\delta_2+\delta_3>n/2$, and $NH \leq n/2+2$.

Subcase (b): $\Delta=\delta_4$

1 - $g_p \in M_{11}$ or $g_p \in M_{10}$

Algorithm 1 can be employed here. The p th bit is 1 and remains the same before

using the c–link. Also $NH \leq n/2$.

2 - $g_p \in M_{00}$

Once again, Algorithm 1 can be used. By complementing all δ_4 bits of M_{00}, the pth bit changes to 1 making a c–link available before Step 2.

3 - $g_p \in M_{01}$

Here, Algorithm 8 with complexity of $NH \leq n/2+1$ can be used.

Subcase (c): $\Delta > \delta_4$

Subcase (c1): $\delta_3 < \delta_2 - 1$

1 - $g_p \in M_{11}$

Algorithm 13.

input: s and d.

Step 1: Complement all $\delta_4 - 1$ bits of M^p_{11}.

Step 2: Complement $\varepsilon+1$ bits of M_{10}.

Step 3: Take the c–link.

Step 4: Correct the differing bits.

Note that in the first step the pth bit does not change and remains 1 providing a c–link before Step 3. After the first step, a node $s' \in W_{n/4}$ is reached. Moreover, $d_H(s',d) = \delta_4 - 1 + \delta_3 - (\varepsilon + 1) + \delta_2$. A simple analysis will reveal that $NH = n/2+3+\delta_3-\delta_2$. But since $\delta_3 < \delta_2 - 1$, the complexity of Algorithm 13 is : $NH \leq n/2+1$.

2 - $g_p \in M_{10}$

Algorithm 14.

input: s and d.

Step 1: Complement all δ_4 bits of M_{11}.

Step 2: Complement ε bits of M^p_{10}.

Step 3: Take the c–link.

Step 4: Correct the differing bits.

Algorithm 14 is similar to Algorithm 4 and has the same complexity. The only difference is that in Algorithm 14, when complementing ε bits of M_{10}, the bit p must remain intact. Thus, the time complexity of Algorithm 14 is : $NH \leq n/2$.

3 - $g_p \in M_{00}$

Algorithm 15.

input: s and d.

Step 1: Complement the p th bit of M_{00}.

Step 2: Complement all δ_4 bits of M_{11}.

Step 3: Complement $\varepsilon+1$ bits of M_{10}.

Step 4: Take the c –link.

Step 5: Correct the differing bits.

Complementing the p th bit of s in Step 1 guarantees that a c –link will be available when the node s' is reached prior to Step 4. However, this increases the weight of s by one. To compensate for this, $\varepsilon+1$ bits (rather than ε bits) of M_{10} are changed in Step 3). The distance between s' and d is thus: $d_H(s',d)=\delta_4+\delta_3-(\varepsilon+1)+\delta_2+1$. After taking the c –link a node d' is reached such that: $d_H(d',d)=n-d_H(s',d)$. The time complexity therefore will be:

$NH=1+\delta_4+(\varepsilon+1)+1+(n-\delta_4-\delta_3+\varepsilon-\delta_2) = n/2+3-\delta_2+\delta_3$

Noting that $\delta_3<\delta_2-1$, it follows: $NH \leq n/2+1$.

4- $g_p \in M_{01}$

Algorithm 16.

input: s and d.

Step 1: Complement the p th bit of M_{01}.

Step 2: Complement all δ_4 bits of M_{11}.

Step 3: Complement $\varepsilon+1$ bits of M_{10}.

Step 4: Take the c–link.

Step 5: Correct the differing bits.

The pth must be changed to 1 in order to have a c–link available at the site of $W_{n/4}$. Thus ε+1 bits of M_{10} must be changed. The distance between s', the node at which the message arrives after Step 3, and d is: $d_H(s',d)=\delta_4+\delta_3-(\varepsilon+1)+\delta_2-1$. After taking the c–link, a node d is reached such that: $d_H(d',d)=n-(d_H(s',d))$. Therefore, the time complexity of Algorithm 16 is:

$NH=1+\delta_4+\varepsilon+1+1+n-(\delta_4+\delta_3-(\varepsilon+1)+\delta_2-1)$. Simplifying the expression and considering the constraints, it follows:

$NH=n/2+5+\delta_3-\delta_2$ or $NH \leq n/2+3$.

Subcase (c2): $\delta_3 \geq \delta_2-1$

In this subcase it is advantageous to increase the weight of the label s to arrive at $W_{3n/4}$ (rather than decrease the weight and arrive at $W_{n/4}$). Since the pth bit of any node of $W_{3n/4}$ incident on a $c-$ link is 0, before using the c–link the pth bit of the label must be 0.

$1 - g_p \in M_{11}$

Algorithm 16.

input: s and d

Step 1: Complement the pth bit of M_{11}

Step 2: Complement δ_1 bits of M_{00}

Step 3: Complement ε'+1 bits of M_{01}.

Step 4: Take the c–link.

Step 5: Correct the differing bits.

The pth bit which is initially 1 must change to 0. This will decrease the weight of the label by one. Thus, ε'+1 bits (rather than ε' bits) of M_{01} must be complemented. After Step 3, a node s' is arrived such that:

$d_H(s',d)=\delta_1+\delta_2-(\varepsilon'+1)+\delta_3+1$. Then the c-link then can be taken to arrive at a node d' such that: $d_H(d',d)=n-(\delta_1+\delta_2+(\varepsilon'+1)-\delta_3-1)$. The complexity of this algorithm is: $NH=1+\delta_1+\varepsilon'+1+1+d_H(d',d)=3+2\varepsilon'+\delta_1+\delta_4$. But $\varepsilon'=\delta_2-n/4$, and $\delta_3\geq\delta_2-1$. It follows:

$NH\leq n/2+3$.

2 - $g_p\in M_{10}$

Algorithm 17.

input: s and d

Step 1: Complement the pth bit of M_{10}

Step 2: Complement δ_1 bits of M_{00}

Step 3: Complement $\varepsilon'+1$ bits of M_{01}^p.

Step 4: Take the c-link.

Step 5: Correct the differing bits.

Using the same reasoning as before, the complexity of Algorithm 17 is given by: $NH=1+\delta_1+\varepsilon'+1+1+n-(\delta_1+\delta_2-(\varepsilon'+1)+\delta_3-1)$. It follows that:

$NH\leq n/2+3$.

3 - $g_p\in M_{00}$

Algorithm 18.

input: s and d

Step 1: Complement δ_1-1 bits of M_{00}^p

Step 2: Complement $\varepsilon'+1$ bits of M_{01}.

Step 4: Take the c-link.

Step 5: Correct the differing bits.

All the bits of M_{00} must be complemented except for the pth bit which must remain 0. The complexity of Algorithm 18 will be given by:

$NH = \delta_1 - 1 + \epsilon' + 1 + 1 + n - (\delta_1 - 1 + \delta_2 - (\epsilon' + 1) + \delta_3)$. It follows that $NH \leq n/2 + 1$.

4 - $g_p \in M_{01}$

Algorithm 19.

input: s and d

Step 1: Complement δ_1 bits of M_{00}

Step 2: Complement ϵ' bits of M^p_{01}.

Step 3: Take the c-link.

Step 4: Correct the differing bits.

From the above algorithm, the complexity is given by:

$NH = n/2 + 1 + \delta_2 - \delta_3$.

After some simplifications it follows: $NH \leq n/2 + 1$.

The time complexity (i.e. NH) of all the above algorithms is no more than $n/2+3$. This implies that any arbitrary pair of nodes in an n-dimensional NBHC can communicate in at most $n/2+3$ hops. On the other hand, one can easily find a pair of nodes which can communicate in exactly (no fewer than) $n/2+3$ †. Thus the proof is reached. □

The flowchart for routing between two nodes in an n-dimensional NBHC is shown in Figure. In the flowchart, s_p represents the p th bit of label s.

† For instance, a pair of nodes which satisfy the upperbound of NH in part 1 or 2 of Subcase (c2).

VITA

Shahram Latifi received the Master of Science degree in Electrical Engineering from the University of Teheran, Iran in 1980. From 1980 to 1984 he worked as a Technical Manager in Khazar Company, Teheran, Iran. Since 1984, he has been a graduate student at Louisiana state University where he received his Master of Science degree in Electrical and Computer Engineering in 1986. He has held a teaching assistantship during his graduate study at LSU. His research interests include computer architecture, fault tolerant computing, parallel processing, and hypercube networks. He is a student member of IEEE Computer Society. Presently, he is a candidate for the degree of Doctor of Philosophy in Electrical Engineering.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: Shahram Latifi

Major Field: Electrical Engineering

Title of Dissertation: Hypercube-based Topologies with Incremental Link Redundancy

Approved:

A Elamawy
Major Professor and Chairman

F. Glen Hamburg
Dean of the Graduate School

EXAMINING COMMITTEE:

J. Burk Jones

Okan K. Sen

Subhash Kak

Suresh Rai

Richard Tapia

Date of Examination:

7/7/89