Design, Analysis and Routing in Static Interconnection Networks

Doctoral Dissertation

by

Rajib Kumar Das

under the supervision of

Professor Bhabani P. Sinha

INDIAN STATISTICAL INSTITUTE
203, B.T. ROAD, CALCUTTA - 700 035
JULY, 1995

In Memory of

My Beloved Parents

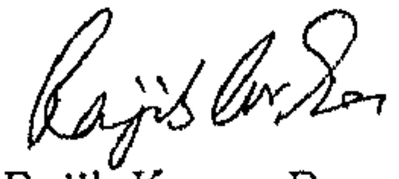
Acknowledgement

Despite the impossibility of giving credit to everyone who has been of help, I should like to single out certain people who have been especially instrumental, directly or indirectly, in the writing of this thesis. It has been my very good fortune to have had contact with exceptional teachers when I was an undergraduate and a graduate student; and to have had stimulating and helpful friends. Finally, it has been my distinct privilege to have had the meaning of research and education exemplified to me by my supervisor Professor B. P. Sinha of Electronics Unit, Indian Statistical Institute, Calcutta.

I am specially grateful to Professor B. B. Bhattacharya of Electronics Unit, ISI, for many things other than reading parts of the thesis and giving valuable suggestions for improvement; to ISI for providing financial support and good library facilities, to Dr. J. Dattagupta, Dr. N. Das and Dr. M. K. Chakrabarti of Electronis Unit, ISI, Dr. S. Sur-Kolay of Jadavpur University and many other workers of the ISI who have helped in some way or the other.

I have enjoyed working with my friends Debasish, Sandip, and Mabhin. Special thanks are due to Krishnendu and Srabani who have made substantial contributions to the contents of this thesis and will always receive my appreciation.

My family has contributed much to this thesis, both tangible and intangible, visible and (except to me) invisible. My sister Mousumi and her husband Dipankar have provided a lot of support and encouragement. Most importantly "they had been there" when I needed their unique help. To all of them, go my deepest gratitude and thanks.



July 25, 1995

Rajib Kumar Das.

Electronics Unit,

Indian Statistical Institute,

Calcutta.

Contents

1 Introduction **1**

1.1 Introduction 1

1.2 Scope of the Thesis 5

1.2.1 Bridged Hypercubes 6

1.2.2 Twisted Hypercubes 6

1.2.3 Fault-Diameter of Hypercubes 7

1.2.4 Distributed Loop Networks 8

1.2.5 Dense Odd Degree Graphs 9

1.2.6 Dense Topology with Multiple Loops 9

2 A Brief Review **10**

2.1 Introduction 10

2.2 Hypercubes 11

2.2.1 Fault Diameter of Hypercubes 14

2.3 Distributed Loop Networks 15

2.4 Star Graphs and Pancake graphs 18

2.4.1 Star graph 20
2.4.2 Pancake Graphs 24
2.5 Dense Regular Topology 25
2.5.1 The Lower Bound 26
2.5.2 Some Well Known Families of Dense Graphs 27

3 Bridged Hypercubes **28**
3.1 Introduction 28
3.2 Notations and Terminologies 29
3.3 Bridged d-Cube with Diameter $\lceil \frac{d}{2} \rceil$ 30
3.4 Average Path Length in Bridged Q_d 35
3.5 Reduction in Diameter by k in Q_d, $(d \geq 2k)$ 39
3.5.1 Reduction by an Even Value 40
3.5.2 Reduction by an Odd Value 44
3.6 Routing Algorithm 48
3.7 Conclusion 49

4 Twisted Hypercubes **50**
4.1 Introduction 50
4.2 Preliminaries 51
4.3 Diameter of Twisted Hypercubes 52
4.3.1 Reduction in Diameter by k, $(k \geq 5)$ 60
4.3.2 Reduction in Diameter by $2m$, $(m \geq 4)$ 61

4.4 Routing Algorithm 62
4.5 Conclusion 65

5 Fault Diameter of Hypercubes 66
5.1 Introduction 66
5.2 Definitions and Notations 67
5.3 Bounds on the Path Lengths 69
5.4 Diameter of Q_n with at most $3n-6$ Faulty Nodes 76
5.5 Algorithm for Routing 84
5.6 Conclusion 92

6 Distributed Loop Networks 94
6.1 Introduction 94
6.2 Some Properties of $G(n;1,s)$ 95
6.2.1 Properties of The Sets S_k 96
6.3 Multinode Broadcast 101
6.3.1 Generation of $A_{i,0}$ 103
6.4 Single Node Scatter 109
6.4.1 Construction of the Spanning Tree 110
6.5 Conclusion 117

7 Dense Odd Degree Graphs 118
7.1 Introduction 118
7.2 The Proposed graph and Some of its Properties 119

7.2.1 Degree Distribution 121
7.2.2 Number of Edges 122
7.3 Path length and Diameter 122
7.3.1 Comparison with the Moore Bound 127
7.4 Routing Algorithm 127
7.4.1 Point-to-Point Communication 127
7.4.2 Single Node Broadcast 131
7.5 Implementation of Algorithms 133
7.5.1 Matrix Transpose 133
7.5.2 Maximum/Minimum/Sum/Average 134
7.5.3 Matrix Multiplication 135
7.5.4 ASCEND/DESCEND Algorithms 136
7.6 Extension to Higher Degree 138
7.7 Conclusions 143

8 Dense Topology with Multiple Loops **145**

8.1 Introduction 145
8.2 Description of the Topology 146
8.3 Diameter of the Network 148
8.3.1 Restricted Redundant Binary Representation 148
8.3.2 Upper Bound on the Diameter 151
8.4 Routing Algorithm 158

8.5 Fault Diameter 160
8.6 Implementation of Algorithms 165
8.7 Conclusion 171

9 Conclusion 172

Bibliography 174

List of Publications 186

Chapter 1

Introduction

1.1 Introduction

Many real-life applications such as image processing, weather forecasting, digital signal processing, etc., require large amount of computations. By distributing the task among several processors, one can appreciably reduce the computation time. To solve complex problems, several computer architectures using multiple processors have been introduced. Recent developments in IC technology have made it economically feasible to construct multiple processor systems consisting of hundreds or thousands of processors.

There are two types of multiprocessor systems [PS87]. One is *tightly coupled*, where the processors share a common clock and/or memory. The other is *loosely coupled*, where each processor runs independently with a local clock and a local memory. The interprocessor communication in such systems plays a vital role in the overall throughput and resource utilization. One possible way of effecting inter-processor communication is through the use of an interconnection network. Thus, the interconnection network has become a critical component in such systems.

Interconnection networks can be classified into two categories : static and dynamic. In a static network, links between two processors are fixed once for all and cannot be reconfigured for direct connection to other processors. Ring, star, mesh

[L93b], hypercubes [H69], etc. are examples of static networks. In a dynamic network, the links can be reconfigured by setting the active switch elements of the network. Examples of such dynamic topologies are baseline, reverse baseline, banyan, omega, Benes, shuffle-exchange, etc.

Various issues related to static interconnection networks include i) design of a suitable network topology, ii) analysis of the topological properties, reliability and fault-tolerance, etc. of a network, iii) developing routing algorithms, iv) mapping of algorithms on a network for different types of applications, etc. Some of these points are briefly explained as follows.

1) Network Topology : A static interconnection network can be modeled by a graph $G = (V, E)$ with V as the set of nodes and E as the set of edges such that the nodes represent the processors and the edges represent the communication links among the processors. Design of a suitable network topology plays a vital role in optimizing various system parameters, e.g., cost of the network, communication delay, fault-tolerance, etc.

2) Reliability and Fault-tolerance : With the introduction of a large number of processors, the probability of failure of the processors and the links increases significantly. Many critical real-life applications using multiprocessors need a very low failure rate. Hence, the design of fault-tolerant multiprocessors systems is an important issue. A fault-tolerant interconnection network can tolerate faults to some degree and still provides gracefully degradable performance.

3) Routing Techniques : The most common form of message passing scheme used in a multicomputer network is *store-and-forward* routing. Packets are the basic units of information in a store-and-forward network. Each node is required to use a packet buffer. A packet is transmitted from a source node to a destination node through a sequence of intermediate nodes. When a packet reaches an intermediate node, it is first stored in the buffer. Then it is forwarded to the next node if the desired output channel and a packet buffer in the receiving channel are both available. Another scheme for routing messages in multicomputer system is

the *wormhole routing* in which a packet is subdivided into smaller flits [DS87]. Unlike the store-and-forward scheme which is controlled by software, the wormhole scheme is entirely hardware routed in a pipelined fashion. Flit buffers are used in the hardware routers attached to the nodes. The communication latency for a wormhole routed network is much smaller than that in a store-forwarded network and almost independent of the distance between communicating nodes.

Various communication requirements may arise in a network, e.g., : i) point-to-point communication, ii) single node broadcast, iii) multinode broadcast iv) single node scatter and v) total exchange [BOS+91]. In point to point communication, a particular node sends or receives data packet(s) from another particular node in the network. In single node broadcast, a particular node sends its packet to all other nodes in the network. In multinode broadcast, each node sends its packet to all other nodes in the network. In single node scatter, a particular node sends different packets to all other nodes in the network. In total exchange, each node sends a packet to every other node in the network (here a node sends different packets to different nodes, in contrast to the multinode broadcast, where a node sends the same packet to every other node).

Every network needs a control strategy to route data from a source to various destinations. In centralized control, a single controller supervizes the flow of data through the network, whereas in distributed control this is done by individual processors.

Fast and easy-to-implement routing algorithm is one of the desirable features of an interconnection network. To facilitate routing, each node must have a unique address. In distributed control, a message is also tagged by its destination address, so that intermediate nodes can decide on the proper routing path.

As already mentioned, the issues concerning routing and fault-tolerance are closely related to the design of a network topology. For example, the routing delay has a lower bound which is governed by the diameter of the network; the fault-tolerance

is dependent on the connectivity of the network.

A static network topology should have the following desirable traits :

i) Low number of links : This is to keep the cost of the network small.

ii) Low degree : The maximum degree of a node in the network graph should be small. This requirement helps to keep the number of I/O ports of a processor to a small value.

iii) Regularity : The network graph should preferably be *regular*, i.e., should have the same number of connections or uniform *degree*.

iv) High bisection bandwidth : The bisection bandwidth is the minimum number of edges along a cut which divides the network into two equal halves. Thus the bisection width provides a good indicator of the maximum communication bandwidth along the bisection of the network [H93a].

v) Low diameter : To minimize the message-passing overhead, the *diameter* of the network graph should be as small as possible.

vi) High degree of fault tolerance : The network should remain operational even in the presence of node/link failures. A common measure of fault-tolerance is the node (edge) connectivity of the graph, which is the minimum number of node (edge) faults that makes the network disconnected. Another useful measure is the f-fault-diameter which is the maximum diameter of the network in the presence of f node/link faults.

vi) Ease of mapping of algorithms : The versatility of a network for efficient implementations of different kinds of parallel algorithms is another desirable feature.

vii) Incremental extensibility : Ideally, the network should be extensible by one node at a time, or at least by a small group of nodes.

Many of the above requirements are mutually conflicting with each other. For example, we would like to have low degree and low number of links. But this will

affect the diameter and fault-tolerance properties of the network graph. Design of a network topology optimizing all the above features is almost intractable. One common approach is to optimize some of the desirable features of the network, keeping other parameters within the acceptable limits. Modifications of existing network topologies to provide improved performance measures are also of sufficient interests.

1.2 Scope of the Thesis

One of the interesting topics of research in the field of interconnection networks is the design of new network topologies. Another aspect for consideration is the analysis of some well known network graphs to find various parameters for performance evaluation, e.g., diameter, mean inter-node distance, fault-tolerance, etc. Finding efficient communication algorithms for different types of networks is also of considerable importance. This thesis addresses some of these problems in the context of static networks, as given below :

i) Modification of the hypercube structure by adding some extra edges, or by exchanging a few pairs of independent links so that the diameter and the average internode distance are both reduced, without adding much complexity to the routing algorithm.

ii) Analysis of fault-diameters of hypercubes and developing routing algorithm in the presence of faulty nodes.

iii) Design of optimal communication algorithms in distributed loop networks.

iv) Design of two new families of dense graphs along with the studies on their routing algorithms, mapping of algorithms for various applications, etc.

1.2.1 Bridged Hypercubes

The hypercube is a very popular network topology with attractive properties such as regularity, symmetry, small diameter, recursive construction and easy routing. Variations of hypercubes have been discovered which further improve some of these properties. These include bridged hypercube [AL90], twisted hypercube [ENS91], cube connected cycle [PV81], generalized hypercube [BA84], incomplete hypercube [K88], etc.

In this thesis, a variant of hypercube has been proposed, in which the diameter of an n-dimensional hypercube Q_n is reduced to $\lceil n/2 \rceil$ by adding some extra links, referred to as *bridges*. The number of extra edges to be added is equal to $\binom{n}{r} + 1$, where $r = \lfloor n/4 \rfloor + 1$. The extra edges constitute a small fraction of the total number of edges. This fraction is $\binom{n}{r} + 1 : n\,2^{n-1}$, which decreases with increase in value of n. The average internode distance for such a bridged hypercube has been computed and an easy algorithm for the shortest path routing has also been developed.

Another part of the work is the reduction in the diameter of Q_n by some arbitrary value k, $k \leq n/2$. It is shown that by adding 8 bridges, to an n-dimensional hypercube ($n \geq 4$), (also termed as an n-cube), its diameter can be reduced by 2. Further, by adding 16 bridges to an n-cube ($n \geq 6$), its diameter can be reduced by 3. For the general case, it is shown that by adding $\binom{4m}{m+1}+1$, ($m \geq 2$) bridges to an n-cube, ($n \geq 4m$), its diameter can be reduced by $2m$; by adding $2\binom{4m-3}{m} + 1$, ($m > 2$) bridges to an n-cube ($n > 4m-2$), its diameter can be reduced by $2m-1$. One important result is that the number of extra bridges necessary to reduce the diameter by k, remains constant for all hypercubes of dimensions greater than $2k$.

1.2.2 Twisted Hypercubes

In a bridged hypercube the degrees of some nodes increase by one. Another technique to reduce the diameter is through *twisting* where the degrees of all nodes

remain the same [ENS91]. A *twist* involves exchanging a pair of independent links (two links are independent if they are not incident on a common node). It is shown that by exchanging 4 pairs of independent links in an n-cube ($n \geq 5$), its diameter can be reduced by 2. Exchange of 16 pairs of independent links would reduce the diameter of an n-cube ($n \geq 7$), by 3. By exchanging 57 pairs of independent links, the diameter can be reduced by 4, for $n \geq 10$. In general, to reduce the diameter of an n-cube by $n/2$, n being an even number ≥ 10, the total number of link pairs to be exchanged is equal to $\binom{n-1}{r}+1$, where $r = \lfloor n/4 \rfloor + 1$. For the general case, a scheme is given which reduces the diameter by k, $k \geq 5$, for hypercubes of dimensions greater than or equal to $2k$. A routing algorithm has also been developed for twisted n-cubes ($n \geq 10$) of diameter $\lceil n/2 \rceil$.

1.2.3 Fault-Diameter of Hypercubes

An important issue related to the hypercube structure is its fault-tolerant capability. To investigate this, we need to understand how the faults affect the structure of a hypercube. Such investigation may help in developing efficient techniques for routing [LH92], [F92], embedding of subcubes [L89], reconfigurability [TSA90], etc. in the presence of faults. For such purposes, it is often very useful to know how the length of the shortest paths between two arbitrary nodes and also the diameter of the hypercube are modified due to faults. It is shown in this thesis that in the presence of $3n-6$ faulty nodes in an n-dimensional cube ($n \geq 5$), with the faults distributed in such a manner that the faulty cube still remains connected, the diameter is at most $n+3$. It is also shown that, in such a situation, for $n > 5$, only those nodes which are at a Hamming distance of $n-3$ may be at the maximum distance of $n+3$. For $n > 5$, all such n-cubes with $3n-6$ faulty nodes and diameter $n+3$ are shown to be isomorphic. An algorithm is developed for routing in such a faulty hypercube. For any two non-isolated nodes u and v which are at a Hamming distance $H(u,v)$, the algorithm gives a path of length at most $H(u,v)+6$ between them. For an n-dimensional hypercube, the time complexity of this algorithm is $f \log n$, where f is the number of faulty nodes.

1.2.4 Distributed Loop Networks

The *loop structure* is one of the most popular network topologies, with the desirable features of simplicity, symmetry and easy routing algorithm. But it is highly vulnerable to faults and can suffer from a large transmission delay. Both these problems can be reduced to some extent by adding a few extra links.

A distributed loop network [BT91] is one possible structure obtained by adding the extra links such that each node on the loop is connected to two additional nodes by direct edges, called as *chords*. If the nodes are numbered as $0, 1, \cdots, n-1$, then a node $i \in \{0, 1, \cdots, n-1\}$ in the network will be adjacent to the nodes $i \pm 1 \bmod n$ and $i \pm s \bmod n$. The corresponding network graph is denoted by $G(n; 1, s)$, s being the chord length.

Some properties of the distributed loop network have been explored. These properties have been utilized in developing algorithms for multinode broadcast and single node scatter in such distributed loop networks. The algorithms are based on the assumption of multiple link availability (MLA), where we assume full duplex communication through each incident link simultaneously. The algorithms can, however, be easily modified for the situation when this assumption is not valid. In multiple node broadcast, each node sends its packet to all other nodes in the network. Since each node has to receive $(n-1)$ packets and can receive at most 4 packets at a time, the minimum time for multinode broadcast is $\lceil (n-1)/4 \rceil$. The minimum number of packet transmissions required for this operation is $n(n-1)$. The proposed algorithm for multinode broadcast takes $\lceil (n-1)/4 \rceil$ units of time and hence is optimal with respect to time. Also, the number of packet transmissions in this algorithm is $n(n-1)$. In single node scatter, a source node sends different data packets to all other nodes in the network. Since the source node has to send a total of $n-1$ data packets and can send only 4 packets at a time, the minimum time required for single node scatter is also $\lceil (n-1)/4 \rceil$. The proposed algorithm for single node scatter takes $\lceil (n-1)/4 \rceil$ units of time and hence, this is also optimal with respect to time.

1.2.5 Dense Odd Degree Graphs

A *new family of graphs* of maximum degree 5, having $N = 4^n$ nodes, $(n > 2)$ and diameter $\leq \lfloor \frac{3}{4} \log_2 N \rfloor + 1$, has been proposed. For even n, these graphs are regular of order 5. For odd n, all but 4 vertices have degree 5 and the remaining 4 vertices have degree 4. This idea is then generalized to define families of such *almost regular* odd degree graphs of maximum degree $2j+1$, $(j > 2)$ with $N = (2j)^n$ nodes, $(n > 2)$ and diameter $\leq \lfloor \frac{3}{4} \log_j N \rfloor + 1$. Algorithms for point-to-point routing and single node broadcast have been developed for such graphs with maximum degree 5. Various algorithms for real-life applications have also been mapped on these topologies.

1.2.6 Dense Topology with Multiple Loops

A *new family of* low diameter networks with multiple loops has been proposed. The corresponding network graph is denoted by $G(m, N)$, where N is the number of processors and m is a parameter of the graph such that $(m-1) \times 2^{\lfloor \frac{m-1}{2} \rfloor + 1} < N \leq m \times 2^{\lfloor \frac{m}{2} \rfloor + 1}$. Also, N is an even multiple of m. The graph $G(m, N)$ is hamiltonian with an average degree of $(3 + \frac{1}{m+1})$ and the maximum node degree of 4. The diameter of $G(m, N)$ is bounded above by $\lfloor \frac{11m}{8} \rfloor + 1$. The graph is incrementally extensible and there exists a simple routing algorithm for point-to-point communication through a path of length less than or equal to the diameter. Mapping of ASCEND/DESCEND class of algorithms [PV81] on these networks has also been discussed.

Chapter 2

A Brief Review

2.1 Introduction

Interconnection networks play a major role in the design of efficient parallel and distributed computing systems. Various topologies for static interconnection networks have been proposed in the literature. Typical examples are linear array [K80], ring [FL72], [RL75], star [S+77], tree [HS77], near-neighbor mesh [B+68], completely connected graph [A72], chordal ring [WL81], distributed loop network [BT91], hypercube [H69], cube-connected cycle [PV81], Moebius graph [LS82], de Bruijn graph [B46], Kautz graph [K69] etc. An extensive survey of both static and dynamic interconnection networks upto 1984 is available in [WF84]. Some of the very recent topologies are Star graph [HAK87], Pancake graph [AK89], Fibonacci cube [H93b] etc. Keeping in mind the objective of this thesis, we present here a brief review on the following popular static networks :

- Distributed loop networks
- Hypercubes
- Star graphs and Pancake graphs
- Regular dense topologies

2.2 Hypercubes

Hypercube is one of the very popular topologies used in parallel and distributed processing. It has many attractive features like regularity, node symmetry, edge symmetry, ease of routing, small diameter, strong connectivity, recursive construction, partitionability and relatively small link complexity. An n-dimensional hypercube Q_n, also termed as an n-cube, is recursively defined as $Q_n = Q_{n-1} \times K_2$, for $n > 0$, where K_2 is a complete graph of order 2, Q_0 consists of a single node and '$\times$' is a Cartesian product on graphs [H69]. Examples of hypercubes of dimensions 1, 2 and 3 are given in Fig. 2.1

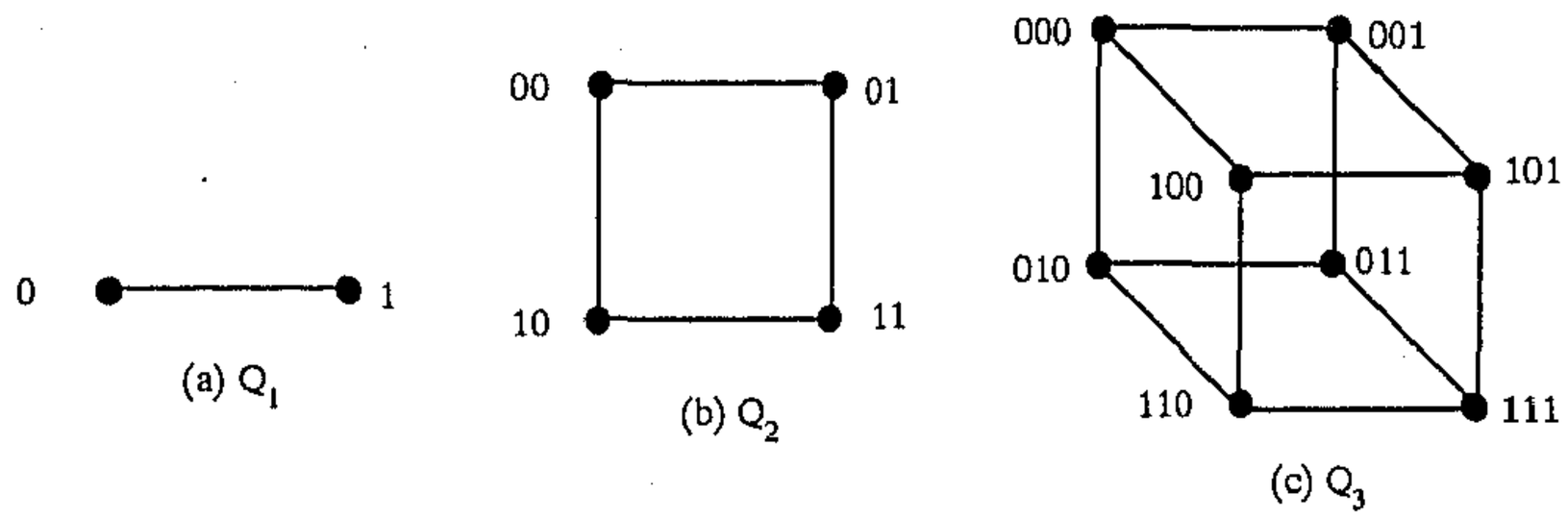


Fig. 2.1 Hypercubes of dimensions 1, 2 and 3

Each node of Q_n can be labeled by a distinct n bit binary number $a_0 a_1 \cdots a_{n-1}$ where two adjacent nodes will have their labels differing in exactly one bit position. Q_n is regular with degree n and has $N = 2^n$ nodes and $n\,2^{n-1}$ edges. The diameter of Q_n is $n = \log_2 N$.

The n-cube has received wide attention of researchers in recent years because of its versatile application in parallel and distributed processing [CS86], [LEN90]. As a result, many interesting properties of the n-cube have been reported in the literature [AG81], [ES79], [F77], [H76], [HHW88], [M80], [SS88]. Various algorithms for hypercubes have been designed and analyzed including routing, broadcasting and related problems [JH89], [NS81], as well as those for real-life applications [LK90].

An n-cube is defined on exactly 2^n nodes, thereby, strongly restricting the possible

system size and leaving a large gap between consecutive allowable sizes. To overcome this difficulty a variation of hypercube called *Incomplete hypercube* was considered [K88]. In [TC92], it was shown that the performance of incomplete hypercubes can be considerably improved by adding extra connections between pairs of nodes. The generalized hypercube structure [BA84] is based on a mixed radix number system and can be designed for any values of N.

The degree of the nodes in hypercubes increases with the increase in size of the network. In [PV81], a graph of constant degree 3, called cube connected cycle is proposed. The diameter of a cube connected cycle having $N = n\ 2^n$ nodes is $\frac{5}{2}logN + O(1)$.

There are other variations of hypercubes which aim at reducing the diameter through some modifications on the original structure. These include the folded hypercube [LA89], bridged hypercube [AL90], twisted cube [ESN91], crossed cube architecture [E92], and varietal hypercube [CC94].

In the folded hypercube [LA89], all the diametrically opposite node pairs are connected by an extra edge. The resulting n-cube has a uniform degree $n+1$ and diameter $\lceil \frac{n}{2} \rceil$. Here, the total number of extra edges is equal to 2^{n-1}. The structure proposed by Amawy and Latiffi requires a far smaller number of extra edges [AL90]. There, it has been shown that by adding $\binom{4m}{m}$ extra edges to a $4m$-dimensional cube ($m \geq 2$), its diameter can be reduced by $2m-1$. As a result, the degree is increased by one for only $2\binom{4m}{m}$ nodes out of 2^{4m} nodes in the network. In [ENS91], it has been shown that by adding only two new edges, the diameter of an n-cube ($n \geq 2$), can be reduced by 1.

Another variation of n-cube to reduce its diameter involves *twisting*, i.e., exchanging a few pairs of independent links. Two edges are called independent if they are not incident on a common node. For example, in Fig. 2.1 the links (010, 110) and (011, 111) are independent. To exchange a pair of independent links (u, v) and (x, y), we delete them and insert two new edges (u, x) and (v, y). In this way, the degree of each node remains the same. Such an exchange of a pair of links is called

called a *twist*. In Fig 2.2, the link pairs (010, 110) and (011, 111) are exchanged.

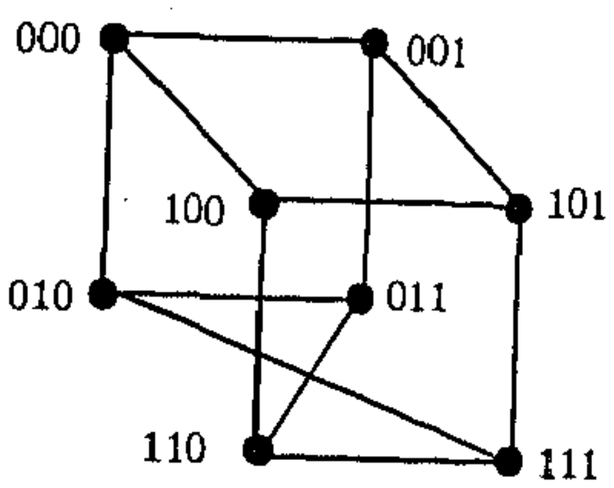


Fig. 2.2 A twisted three-dimensional cube

The idea of reducing diameter by twisting was first introduced by Hilberts *et al* [HKS87]. They have shown that by exchanging $(d-1)2^{d-4}$ link pairs in a d-cube ($d = 2m+1$), its diameter can be reduced to $(d+1)/2$. In [AP91], performance measures of such twisted cubes, in terms of routing, average path length etc. have been studied. In [ENS91] only 4-cycle twists are considered, where the nodes u, v, x and y form a 4-cycle in the hypercube. There, it was shown that a single 4-cycle twist can reduce the diameter of an n-cube ($n > 2$) by 1. In the crossed cube architecture proposed by Efe [E92], the exchange of link pairs are done by introducing a certain number of switches. The diameter of an n-dimensional crossed cube is $\lfloor \frac{n+1}{2} \rfloor$ and the average internode distance is also shorter than that in a hypercube. The number of switches required is equal to $(k-2)2^{2k-1}+2^k$ for a crossed cube of dimension $2k$. For a crossed cube of dimension $2k+1$, the corresponding number is equal to $k2^{2k} - 3(2^{2k-1} - 2^{k-1})$. Algorithms for optimal routing and broadcasting have also been developed for this network.

In [CC94] a new interconnection network topology, called varietal hypercube, has been proposed. The network is a modification of the original hypercube structure keeping the same number of nodes and edges. The diameter of n-dimensional varietal hypercube is $\lfloor \frac{2n}{3} \rfloor$.

2.2.1 Fault Diameter of Hypercubes

An important issue related to the hypercube multiprocessors is their reliability and fault tolerance due to the following reasons : the probability that some processors becomes faulty among such a large number of processors will be high, and a single link or node failure can destroy the regularity and thus may degrade the performance of the hypercube. To investigate the fault tolerant capabilities for hypercubes, we need to understand how the faults affect the structure of a hypercube. Such investigations have led to the development of various techniques for fault tolerance.

Identification of operational subcubes in faulty hypercubes has been considered in [L89]. In [LH92], a fault-tolerant communication scheme that achieves near-optimal routing and broadcasting is proposed. The basic idea is to identify certain fault-free nodes in Q_n that may cause excessive routing delay with the existing fault pattern. For an n-cube, a distributed algorithm with computational complexity $O(n^3)$ is devised to identify all the unsafe nodes. After all the unsafe nodes are identified, a computationally efficient routing algorithm can direct each message to its destination via a path of length no greater than the minimum source-destination path length plus two, provided the number of faulty nodes is no more than $\lceil \frac{n}{2} \rceil$. It has also been shown that broadcasting can be achieved within $n + 1$ steps.

In [F92], algorithms for fault tolerant broadcasting (One-to-all) and gossiping (All-to-all) have been proposed, which do not require any information on the identity of the faulty nodes/links. These algorithms can tolerate upto $n - 1$ faults and are *asymptotically optimal*, that is, they approach the minimum time complexity when the message length increases. Optimal solutions are also proposed for very short messages.

There are several works on the reconfiguration of embedded task graphs in hypercubes when fault occurs [TRS90], [YTR94]. The scheme presented in [YTR94] can tolerate at most n faulty nodes in Q_n, while keeping the task migration time small.

For the purpose of designing efficient fault tolerant techniques, it is often very useful to know how the length of shortest paths between two arbitrary nodes and also the diameter of the hypercube are modified due to faults. It has been shown in [KK87] that there are n disjoint paths between any two nodes, and when the number of faults is less than n the length of the shortest path between two nodes is increased by at most 2, and the diameter of the n-cube is increased by at most 1.

Though an n-cube may become disconnected in the presence of f ($\geq n$) faulty nodes, it is highly unlikely that all the n neighboring nodes of a single node will be faulty. In other words, even with f ($\geq n$) node faults, the cube may remain connected. Therefore, it is useful to study the diameter and shortest path problems in such circumstances as they affect performance of an algorithm running on them.

Esfahanian [E89] introduced the concept of forbidden faulty sets such that all the nodes in a forbidden faulty set cannot be faulty at the same time. In [E89], a forbidden faulty set was constructed by taking all n-neighbors of a node in the cube. Thus there can be 2^n forbidden faulty sets. Under this forbidden faulty set model, Esfahanian has shown that an n-cube remains connected even in the presence of $2n-3$ faults. In [L93a], the fault-diameter of such a faulty n-cube with $f = 2n-3$ has been shown to be at most $n+2$. In [TR93], an algorithm is given for routing between any two non-isolated nodes u and v. The length of the path is less than or equal to the Hamming distance between u and v plus 4. For a hypercube of dimension n, the time complexity of the algorithm is $O(|F| \log n)$ where $|F|$ is total number of faulty nodes.

2.3 Distributed Loop Networks

Processors interconnected in the form of a ring is a very popular network topology. Because of its simplicity and symmetry, we can easily expand the network adding some extra nodes. Also the routing in such networks is very simple. However, the

ring topology suffers from a few disadvantages : 1) it is highly vulnerable to faults in the network and 2) its diameter is high (for a network of n nodes the diameter is $n/2$). Both these problems are reduced to a certain extent when a few more links are added to the network. By adding these links in a uniform way, the desirable properties of symmetry, expandability, uniform building blocks for the switching mechanism and uniform token passing, all are preserved. From the point of view of cost-effectiveness, it is desirable to add as few links as possible. The double loop network or distributed loop network [BT91] is perhaps the simplest of such extensions.

In a distributed loop network, in addition to the ring structure, each node in the ring is connected to two other nodes via chords. Distributed loop network is one of the popular network topologies. A distributed loop network [DHL90], denoted by $G(n; 1, s)$, is defined as follows :

1) the total number of nodes in the network is n,

2) s is an integer such that $1 < s < n/2$,

3) if the n nodes are numbered as 0, 1, 2, ..., $n-1$, then the node i is connected to the nodes $(i \pm 1)$ mod n and $(i \pm s)$ mod n by symmetric (bidirectional) edges for all i, $0 \leq i \leq n-1$.

As an example, the graph $G(14; 1, 6)$ has been shown in Fig. 2.3.

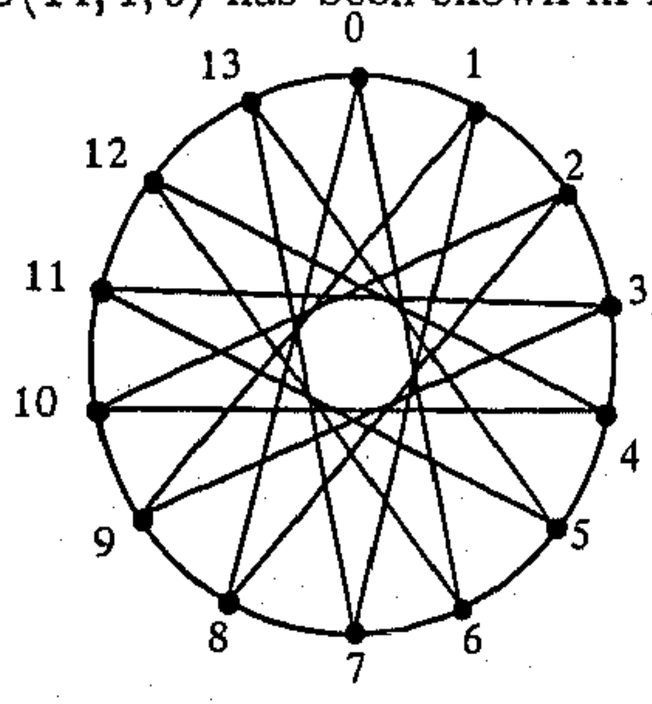


Fig 2.3 $G(14; 1, 6)$

s is referred to as the hop size or just the **hop**. The network $G(N; 1, s)$ has many attractive features like regular structure, constant degree, etc., and is well-studied in literature [BT91], [T91]. This network is a special case of a well-known class of networks called circulant graphs[ET70], which have some good properties in the context of optimal fault-tolerant design [DH91].

A major factor in the computational efficiency of parallel algorithms involving data communications between processors is the **diameter** of the underlying network. Let $D_m(n)$ denote the minimum diameter of a distributed loop network $G(n; 1, s)$.

Two main problems regarding distributed loop networks are

i) Given n, find $D_m(n)$;

ii) Find an s such that the diameter of $G(n; 1, s)$ is equal to $D_m(n)$;

Several authors have given a lower bound $lb(n) = \frac{1}{2}\left(\sqrt{2n-1}-1\right)$, on the minimal diameter $D_m(n)$ [WK74], [BIP85], [BW85], [DHL+90]. In [BT91], Bermond and Tzvieli introduced the notation $R[k] = \{2k^2 - 2k + 2, \cdots, 2k^2 + 2k + 1\}$, so that $lb(n) = k$ if $n \in R[k]$. A network $G(n; 1, s)$ is called optimal if its diameter is equal to this lower bound and suboptimal if its diameter is one more than the lower bound.

All the above mentioned authors noticed that the lower bound is attained when $n = 2k^2 + 2k + 1$. The lower bound k cannot be achieved when $n = 2k^2 + 2k$. The problem of finding the values of n, for which it is possible to design optimal $G(n; 1, s)$ is well studied [T91], [DHL+90]. In [T91], infinite families of optimal networks are identified, along with corresponding optimal values for s. Also, the lower and upper bounds on optimal and suboptimal hops are given.

Routing is another important problem for distributed loop networks. In [MS94] an algorithm for the shortest path routing has been developed. In presence of a single fault this algorithm can be suitably modified for routing between non-faulty nodes such that the path length can be at most one more than the optimal.

2.4 Star Graphs and Pancake graphs

A special class of networks, called *symmetric interconnection networks*, has the property that the network viewed from any node of the network looks the same. In such a network, congestion problems are minimized since the load is distributed uniformly through all the nodes. Moreover, this symmetry allows for identical processors at every node with identical routing algorithms. It is also very useful in designing algorithms that exploit the structure of the network. Hypercubes, cube-connected cycles etc are examples of such symmetric networks. We now give the following definitions that formalize this class of networks.

Definition 2.1 *A graph is node symmetric if for every pair of nodes u and v, there exists an automorphism of the graph that maps u into v.*

Definition 2.2 *A graph is edge symmetric if for every pair of edges a and b in the graph, there exists an automorphism of the graph that maps edge a into edge b.*

Star graphs and pancake graphs fall into a special class of symmetric graphs called *Cayley graphs*. The n-cube can also be viewed as a particular class of Cayley graph as discussed below. This class of graphs uses a group-theoretic approach as a basis for defining graphs. In a *Cayley graph*, each vertex is an element belonging to a group. We consider a group to be the set of permutations generated by a set of generators.

Definition 2.3 *Given a set of generators for a finite group G, we can construct a graph, called Cayley graph, in which the nodes correspond to the elements of the group G and the edges correspond to the action of the generators.*

If u and v are two elements of G and there is a generator g such that $ug = v$ in the group G, then there is an edge from node u to node v. By requiring that the

generators be closed under inverses, we ensure that an edge between u and v will allow communication in both directions, that is, $ug = v$ and $vg = u$ are both true.

Any Cayley graph is fully defined by the set of generators. As an example, consider a group of permutations using the symbols 1, 2, 3 and 4 and the generators 1324, 2143 and 4321. Since these generators are permutations of 4 symbols, they must generate either the group S_4 (*Symmetric group* on 4 symbols, which contain $4! = 24$ permutations) or a subgroup of S_4. In this case the subgroup generated by these three generators consists of the eight permutations : 1234, 2143, 2413, 4231, 3412, 3142, 3142, and 1324. The corresponding Cayley graph arising from the above generators is shown in Fig. 2.4.

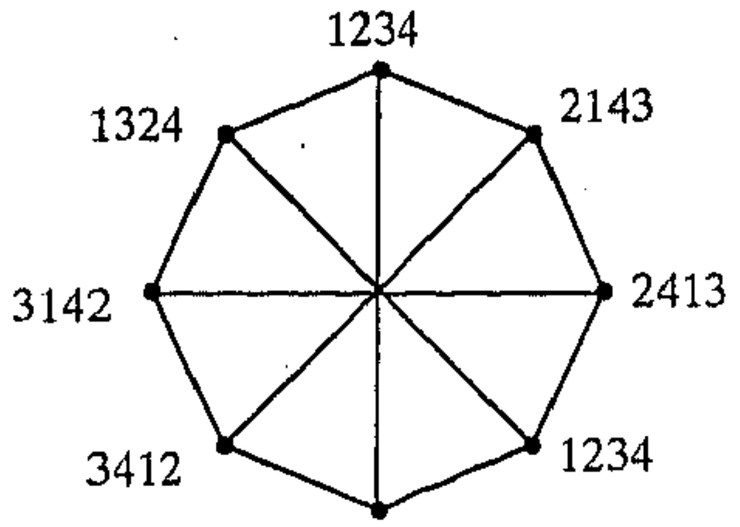


Fig 2.4 A simple Cayley graph.

As another example, consider the generators 213456, 124356, and 123465. The group generated by these generators contain eight permutations and the corresponding Cayley graph, which is the familiar three-dimensional cube is shown in Fig. 2.5.

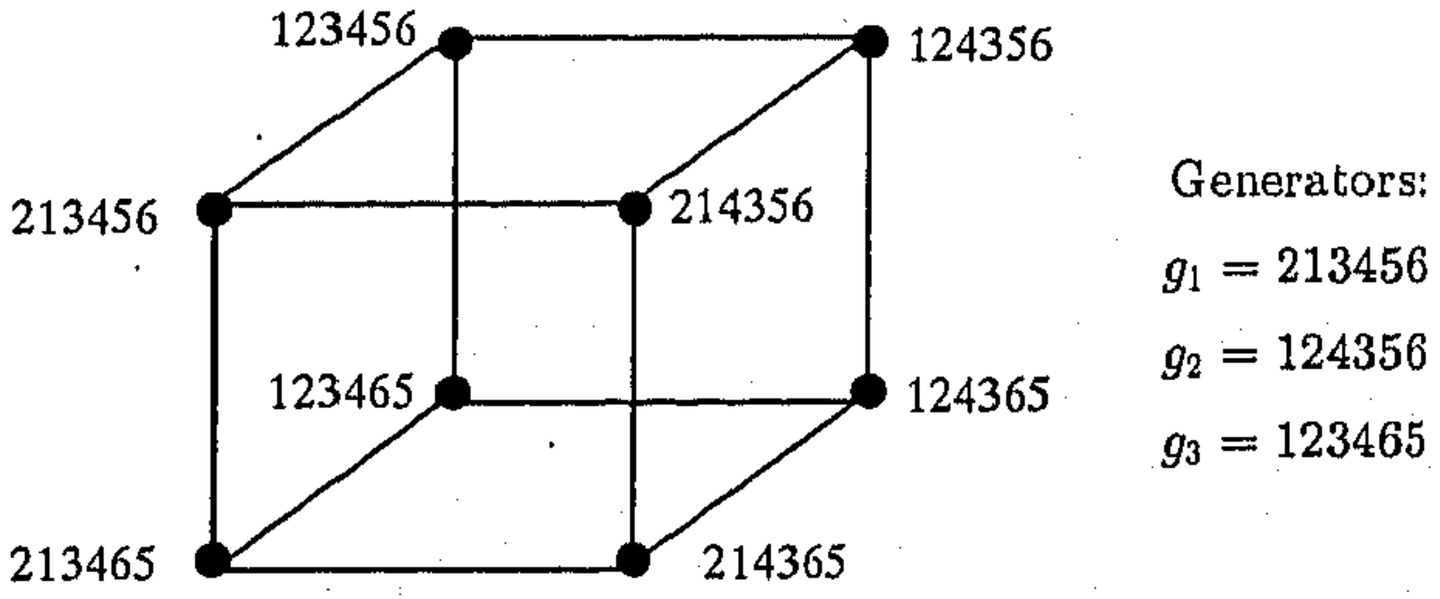


Fig 2.5 Three-cube as a Cayley graph.

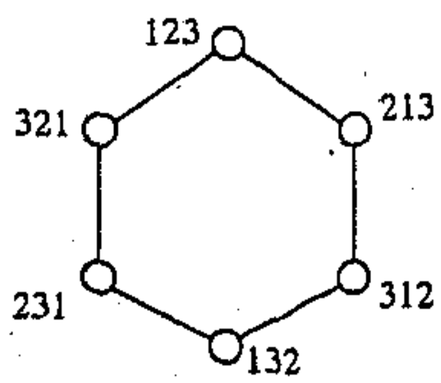
In [AK89], it has been shown that every Cayley graph is node symmetric. One important feature of a node symmetric graph is that routing between two arbitrary nodes reduces to routing from an arbitrary node to a special node. A path from node u to node v can be represented by a sequence of generators $g_1, g_2, \cdots, g_p$, where each g_i is a generator of the Cayley graph. Now, if $g_1, g_2, \cdots, g_p$ is a path from u to v, then it is also a path from $v^{-1}u$ to I, the identity permutation [AK89]. Thus if we want to find a route from u to v we can instead find a route from $v^{-1}u$ to I.

2.4.1 Star graph

Definition 2.4 *The n-star graph is the Cayley graph on the group G consisting of all permutations on n symbols, and the set of generators g defined as follows. The set g consists of* $n-1$ *generators* $\{g_2, g_3, \cdots, g_n\}$ *where* g_i *switches the ith symbol with the first and leaves the remaining symbols in their original positions. So g consists of the following generators :*

$$g_2 = (2134\cdots n)$$
$$g_3 = (3214\cdots n)$$
$$g_4 = (4231\cdots n)$$
$$\vdots$$
$$g_n = (n234\cdots 1)$$

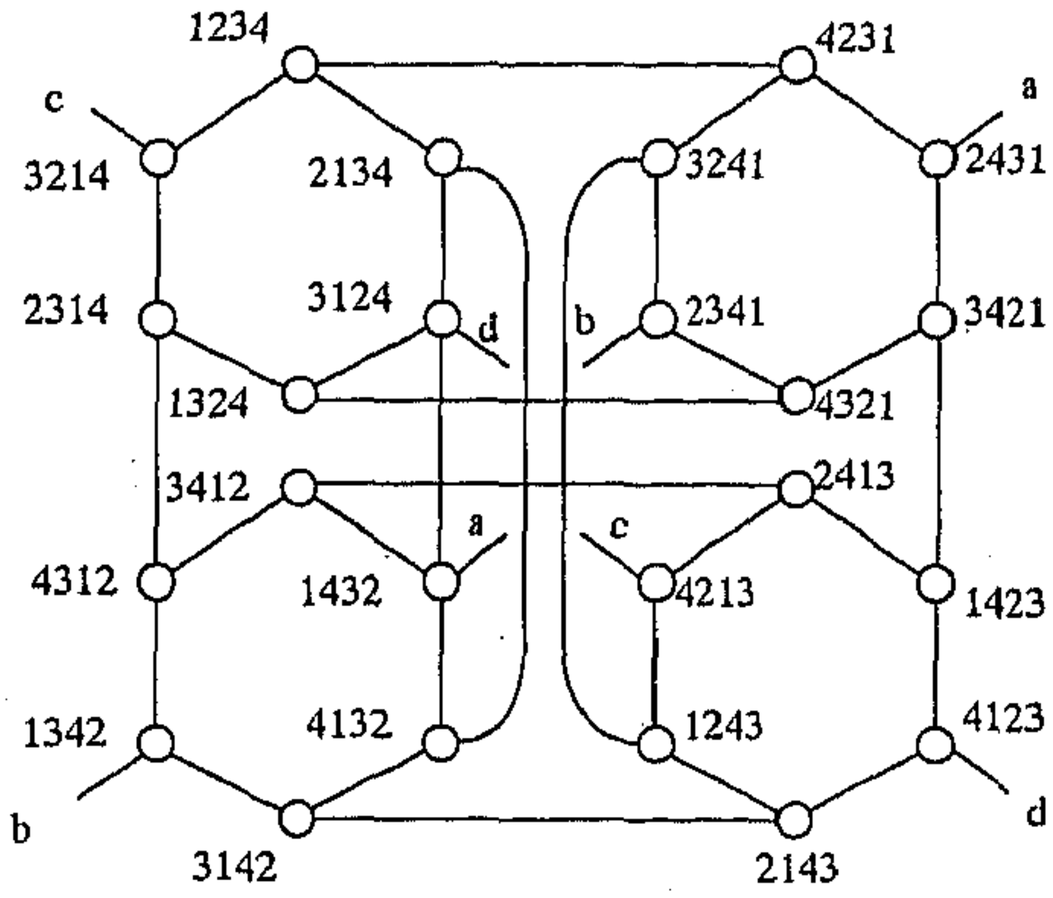
The 3-star and 4-star graphs are shown in Fig. 2.6 and Fig. 2.7 respectively.



Generators : $g_1 = 213$

$g_2 = 321$

Fig 2.6 The 3-star.



Generators:

$g_1 = 2134$

$g_2 = 3214$

$g_3 = 4231$

Fig 2.7 The 4-star.

From Fig. 2.7 it can be noted that the 4-star graph can be decomposed into four 3-star graphs. Again the three-star graph itself is made up of three copies of the two-star graph which is itself a line. More generally, an n-star graph can be viewed as n copies of $(n-1)$-star graphs which are interconnected by edges corresponding to the generator g_n. Furthermore, this decomposition can be carried out recursively, so that each of the $(n-1)$-star graphs is made up of $(n-1)$ copies of $(n-2)$-star graphs, and so on.

A Cayley graph is said to be *hierarchical*, if its generators can be ordered as g_1, $g_2, \cdots, g_d$, such that for each i, $2 \leq i \leq d$, g_i is outside the subgroup generated by the first generators. Every hierarchical Cayley graph has such a recursive decomposition property. The 4-star graph is hierarchical in the sense that the generator g_4 cannot be obtained through any combination of g_2 and g_3.

In [AK89], it is shown that for a Cayley graph defined by a set of generators on n symbols $\{1, 2, 3, \cdots, n\}$, the graph is edge symmetric if and only if for every pair of generators g_1 and g_2, there exists a permutation of the n symbols that maps g_1 into g_2.

Using this, it can easily be shown that the n-star graph is edge symmetric. Some Cayley graphs (particularly, the edge symmetric Cayley graphs) are hierarchical under any ordering of the set of generators. These Cayley graphs are called

strongly hierarchical [AK89]. The n-star graph, being edge symmetric, is strongly hierarchical, that is, it can be recursively decomposed using the generators in any order.

Fault tolerance is another important property of interconnection networks. It has been shown [AK84] that the hierarchical Cayley graphs (with an additional size requirement) are *maximally fault tolerant.* That is, their connectivity is exactly equal to their degree.

In [AK89], an easy algorithm for routing for point to point communication in star graphs has been developed. The diameter of the n-star graph is also shown to be equal to $\lfloor(3(n-1)/2)\rfloor$.

Table 2.1 Comparison of star graph and hypercube.

	Nodes	Degree	Diameter
5-Star	120	4	6
7-Cube	128	7	7
6-Star	720	5	7
9-Cube	512	9	9
7-Star	5040	6	9
12-Cube	4096	12	12
8-Star	40320	7	10
15-Cube	32768	15	15

In Table 2.1 , a comparison of the star graph with the hypercube regarding diameter and average distance is given. It is clear that for a given size, the degree and diameter of the star graph is clearly superior to that of the hypercube. Because of this and other important properties like symmetry, recursive nature etc., the star graph has been considered to be an attractive alternative to the popular hypercube as the network for parallel processing.

A comparative study of the topological properties of the hypercube and star graph can be found in [DA94]. In [QAM94], some useful properties of the star graph as well as some algorithm implementations are discussed. Star graphs are known to be Hamiltonian [KL75]. In [JLD90], it is shown that for all even l such that $6 \leq l \leq n!$, a cycle of length l can be embedded into an n-star graph with unit dilation. Embedding of other useful architectures such as grids [JLD90] and hypercubes [NSK90] are also well studied. Latiffi [L93c] worked on the fault diameter of the star graphs. In [RS93], it is shown that the fault diameter of an n-star graph is only one more than the fault free diameter in presence of upto $n-2$ faults. Since the star graph is not at all incrementally extensible, a variation of the star graph called *incomplete star* has been proposed [LB94] to overcome this difficulty.

The interprocessor communication problem in star graph has drawn considerable interest among the researcher. In [BNL93], a distributed routing algorithm requiring no global knowledge of faults is presented. The algorithm routes through the shortest path if no faults are encountered, i.e., the faults are such that an optimal path still exists. In the absence of an optimal path, the algorithm always finds a path, if any, between two nodes. In an n-star graph, if the number of faults is at most $n-2$, the path length is increased by at most $2i+2$, where i is $O(n)$. Also, an efficient broadcasting algorithm on the star graphs is presented in presence of faults.

In [AHK87], a one-to-all broadcasting algorithm is presented for the star graph. The number of communication steps required by their algorithm for an n-star graph is $\leq 3\left(n\log_2 n - \frac{n}{2}\right)$. In [MS92], a one-to-all broadcasting algorithm requiring much less number of steps has been developed. The number of communication steps required by this algorithm is $\sum_{i=2}^{n}(\lceil \log_2 i \rceil + 1)$.

A solution of the deadlock problems for routing is presented in [MJ94], using the concept of virtual channels. An optimal randomized algorithm for permutation routing on the star graphs is developed in [PRW94]. The algorithm runs in $O(D)$ step, D being the diameter of the network.

2.4.2 Pancake Graphs

Before giving a formal definition of the *pancake graph* we consider a combinatorial problem, called the *pancake flicking problem.* The problem is to sort a stack of n pancakes of different sizes by repeatedly flipping top substacks with a spatula. As an example consider a stack of 4 pancakes represented by the permutation 2143. The left end of the permutation is assumed to be the top of the stack. Thus, the sorted stack would look like the identity permutation 1234. If we apply a 2-flip (i.e., flip the top two pancakes with a spatula) to the 2143 permutation, it will be transformed into 1243. A possible sequence of flips to sort the permutation 2143 is given below

$$2143 \rightarrow 4123 \rightarrow 3214 \rightarrow 1234.$$

The pancake flipping problem can be modeled as a class of Cayley graphs using generators representing pancake flips.

Definition 2.5 *The n-pancake graph is the Cayley graph on the group G consisting of all permutations on n symbols 1,2, ..., n and the set of generators g defined as follows. The set g consists of* $n-1$ *generators* $\{g_2, g_3, ..., g_n\}$ *where* $g_i = i(i-1)\cdots 321(i+1)(i+2)\cdots n$.

The pancake graph for $n = 3$ is the same as the 3-star graph. The pancake graph for $n = 4$ is shown in Fig 2.7.

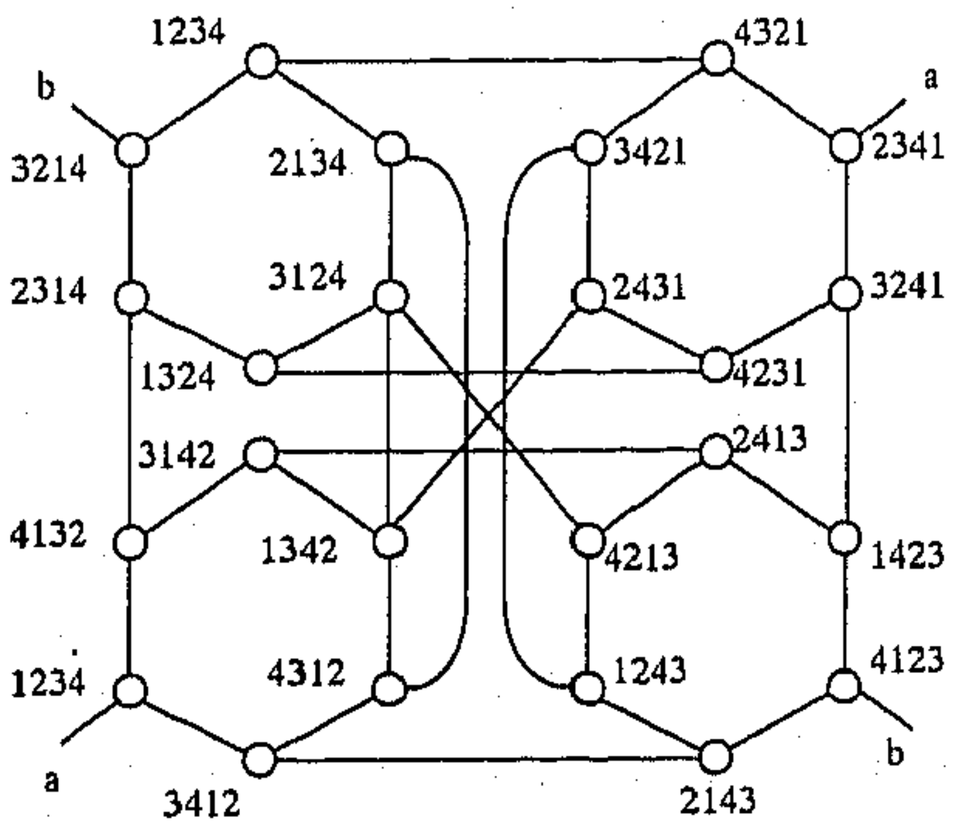


Fig 2.7 The 4-pancake graph.

Finding the diameter of the *pancake graph* is equivalent to finding the maximum number of steps M_n required to sort an arbitrary permutation of n symbols into the identity permutation using only prefix reversal. This problem is still open, and the best known results can be found in [GP79], in which it is shown that $M_n \leq (5n+5)/3$, and for n a multiple of 16, $M_n \geq 17n/16$. However, a simple routing algorithm that routes in at most $(2n-3)$ steps is given in [AK89].

There are certain similarities between the pancake graphs and star graphs. Both have diameters of the order of $O(n)$ with $n!$ nodes. Embedding of cycles of length l, for all even l between 6 and $n!$ with unit dilation, is possible in n-pancake graphs as well [JLD90]. The pancake graph is hierarchical, i.e., recursively decomposable but not strongly so. In particular an n-pancake graph can be decomposed into sub-pancakes only in the order of the generators g_n, g_{n-1}, $\cdots$, g_3, g_2. Also, the star graphs are known to be bipartite [AK89], whereas, the n-pancake graph is not bipartite for $n \geq 4$.

Regarding implementation of parallel algorithms on pancake graphs, it is shown in [AK89] that there is an $O(n \log n)$ algorithm to find the maximal element among $n!$ elements distributed among $n!$ processors located at the nodes of an n-pancake graph. In [AQS93], some fundamental algorithms for the star and pancake graphs with applications to computational geometry, have been developed. In [QAM94], a constant time routing algorithm has been developed for pancake graphs. Algorithms for broadcasting, computing prefix sums, sorting and merging, and algorithms of the ASCEND and DESCEND classes have also been implemented on the pancake graphs.

2.5 Dense Regular Topology

When networks with hundreds of processors are considered, it is desirable from cost and complexity considerations that the network graph is regular and degree of each node is small. Because we wish to minimize the message-passing overhead,

the *diameter* of the graph should be as small as possible. It is further desirable that the graph should be extensible one node at a time, or at least in a small group of nodes. An alternative criterion is that the average internode distance between all pairs of nodes should be minimized.

In summary, the network graph should be a member of a family of regular graphs in which the same small number of edges are incident at each node and the diameter is as small as possible.

2.5.1 The Lower Bound

In designing regular network topologies, there are three variables of concern : the degree d, the diameter k and the number of nodes n. Usually, n is expressed as a function of d and k, i.e., $n = f(d, k)$. For example in a hypercube [H69], $d = k$ and

$$f_{hypercube}(k, k) = 2^k$$

The problem of maximizing $n = f(d, k)$, given d and k was studied by Akers [A65], Elspas [E64], Friedman [F66], Korn [K67] and Storwick [S70]. An optimal example is a *Moore graph* [BI73], [F71], [HS60]. A Moore graph of degree d and diameter k is a regular graph which contains the maximum number $n = f(d, k)$ of nodes, where

$$n = f_{Moore}(d, k) = 1 + d + d(d-1) + \cdots + d(d-1)^{k-1} = \frac{d(d-1)^k - 2}{d-2}$$

However, it has been shown [BI73], [F71], [HS60] that Moore graphs exist only for a few combinations of d and k. For $d > 2$, only the (3,2) [and possibly (57,2)] Moore graphs are realizable. This (usually unattainable) upper bound $f_{\text{Moore}}(d, k)$ is called the *Moore bound.*

While looking for regular networks with number of nodes as close as possible to the Moore bound, it is convenient to pose the problem in a different functional form. That is, given n and d, determine a regular graph of minimum diameter.

From the Moore Bound, a lower bound on the diameter k for a regular graph of degree d with n nodes can be obtained as

$$k_{lb}(d,n) = \left\lceil \log_{d-1} \frac{n(d-2)+2}{d} \right\rceil$$

2.5.2 Some Well Known Families of Dense Graphs

Construction of dense regular graphs, i.e., regular graphs of a given order with small diameters has also received special interest among the researchers. There are already some well known dense regular graphs in the literature, with both even and odd degrees. Among the even degree graphs, the de Bruijn graphs [B46] and Kautz graphs [K69] are two dense tetravalent structures. A de Bruijn graph with 2^n nodes has degree 4 and diameter n. Similarly, a Kautz graph with $2^n + 2^{n-1}$ nodes has degree 4 and diameter n. Generalization of the ideas in the construction of these graphs has also been done leading to higher degree graphs. Thus, radix-k de Bruijn graphs [B46] have been defined for k^n nodes with degree 2k and diameter n. The Kautz graphs are also likewise generalized [K69] for $(k^n + k^{n-1})$ nodes with degree $2k$ and diameter n. Another well known dense graph is the trivalent Moebius graph [LS82] which has 2^n nodes with diameter $\leq \lfloor \frac{3n}{2} \rfloor$.

In general, there are two possible approaches to the construction of new types of dense regular graphs. One is to find a regular graph with as many nodes as possible for a given diameter and degree. The other is to construct a regular graph with as small diameter as possible, when the total number of nodes and also the degree of each node (i.e., the order of regularity) are both specified. Several authors have worked in this area [TS79], [BDQ82]. In [MR82], two new families of graphs were presented. In [D84], Doty presented a new method of constructing interconnection network, based on the generalization of chordal ring network proposed by Arden and Lee [AL81]. Shen and Back [SB93] proposed a new method called recursive expansion, to build large interconnection networks, starting from smaller networks.

Chapter 3

Bridged Hypercubes

3.1 Introduction

The hypercube interconnection scheme is a very popular network topology. An n-dimensional hypercube Q_n (also termed as n-cube) consists of $N = 2^n$ nodes interconnected as follows :

i) each node is labeled by an n-bit binary number $a_0a_1 \cdots a_{n-1}$,
ii) two nodes are connected by an edge if and only if their binary labels differ in exactly one bit position.

The n-cube has become an interesting topic of research in recent years due to its versatile applications in parallel and distributed processing. Many interesting modifications of the n-cube have been reported in the literature [AG90], [AP89], with a view to reducing its diameter, making it incrementally extensible, etc.

In this chapter, we propose a new family of network topologies, by modifying the original hypercube structure, which will have diameters lesser than that of a hypercube, yet retaining the other desirable features of the hypercube, e.g., ease of routing under fault free and faulty situations, high connectivity, i.e., high degree of fault-tolerance, etc. We first show that by adding $\binom{d}{\lfloor \frac{d}{4} \rfloor + 1} + 1$ extra edges, termed as *bridges*, to a d-cube ($d > 4$), its diameter can be reduced by $\lfloor \frac{d}{2} \rfloor$.. Then we

generalize this scheme to add $\binom{4m}{m+1}+1$, $(m \geq 2)$ *bridges* to an n-cube $(n \geq 4m)$ to reduce its diameter by $2m$. To reduce the diameter by $2m-1$, we add $2\binom{4m-3}{m}+1$, $(m \geq 3)$ bridges to an n-cube $(n > 4m-2)$. In both cases, the number of extra edges added is a function of m only, and not dependent on n. We also compute the average path length between the nodes of a bridged Q_d of diameter $\lceil d/2 \rceil$. Finally we give an algorithm for the shortest path routing in a bridged hypercube Q_d of diameter $\lceil d/2 \rceil$.

3.2 Notations and Terminologies

A node in a hypercube is represented by a string of binary digits. A subcube is represented by a string over the alphabet $\{0, 1, *\}$. For example, $01*1*$ represents the subcube formed by the nodes 01010, 01011, 01110, 01111. In order to make the representation more compact we would replace p consecutive occurrences of a given symbol by that symbol raised to the power of p. For example 00011 will be written as $0^3 1^2$. Let $f(x)$ be the number of 1's in the binary representation of a node x. We now define the following sets :

$W_r = \{x | f(x) = r\}$
$A_r = \{x | x \in W_r \text{ and } x \in *^{n-1}0\}$
$B_r = \{x | x \in W_r \text{ and } x \in *^{n-1}1\}$

Also let $H(x, y)$ denote the Hamming distance between two nodes x and y.

If a node x is represented by $x = a_0 a_1 \cdots a_{n-1}$, $a_i \in \{0, 1\}$, a node diametrically opposite to x is denoted by $\bar{x} = \overline{a_0}\ \overline{a_1} \cdots \overline{a_{n-1}}$.

For a source destination pair (s, t) where $s = a_0 a_1 \cdots a_{n-1}$ and $t = b_0 b_1 \cdots b_{n-1}$ we define four sets of bit positions S_{00}, S_{01}, S_{10} and S_{11} as follows :

$S_{00} = \{h | a_h = 0,\ b_h = 0\}$
$S_{01} = \{h | a_h = 0,\ b_h = 1\}$
$S_{10} = \{h | a_h = 1,\ b_h = 0\}$

$S_{11} = \{h | a_h = 1,\ b_h = 1\}$

Example 3.1 Let $s = 01001001$ and $t = 10011011$. Then $S_{00} = \{2, 5\}$, $S_{01} = \{0, 3, 6\}$, $S_{10} = \{1\}$ and $S_{11} = \{4, 7\}$.

We denote the cardinalities of the sets S_{00}, S_{01}, S_{10} and S_{11} by s_0, s_1, s_2 and s_3 respectively. Hence, the number of zeros in $s = (s_0 + s_1)$ and the number of ones $= s_2 + s_3$. Also $H(s, t) = s_1 + s_2$.

3.3 Bridged d-Cube with Diameter $\lceil \frac{d}{2} \rceil$

The extra edges that we propose to add to a cube are always in the form of $(x, \bar{x})$. We refer to these extra edges as *bridges*. In a d dimensional cube ($d > 4$), we take a fixed value of r, $0 < r < \frac{d}{2}$, and then connect bridges from all the nodes in $W_r \cup W_0$. The cardinality of the set W_r is $\binom{d}{r}$. Hence the number of such bridges that will be added to the hypercube is $\binom{d}{r} + 1$. To find an expression for the diameter of the bridged hypercube, we first try to unfold the possible paths between two nodes s and t through a bridge.

Let x be a node at which a bridge is connected. To reach t from s, we can reach x by changing some bits in s. Let $S'_{ij} \subseteq S_{ij}$, $i, j \in \{0, 1\}$, be the sets of bit positions where these changes are made. Let s'_0, s'_1, s'_2 and s'_3 be the cardinalities of the sets $S'_{00}, S'_{01}, S'_{10}$ and S'_{11} respectively. Then we have the following lemma.

Lemma 3.1 *The shortest path between two nodes s and t, via the bridge $(x, \bar{x})$ is of length $p = 1 + 2(s'_1 + s'_2) + s_0 + s_3$.*

Proof : From s, we reach x in $(s'_0 + s'_1 + s'_2 + s'_3)$ steps. Another single step via the bridge leads to the node $\bar{x}$. $\bar{x}$ differs from t in all bits of S'_{01}, S'_{10}, $S_{00} - S'_{00}$ and $S_{11} - S'_{11}$ and nothing else. Thus, the path length p is equal to $(s'_0 + s'_1 + s'_2 + s'_3) + 1 + (s'_1 + s'_2 + s_0 - s'_0 + s_3 - s'_3) = 1 + 2(s'_1 + s'_2) + s_0 + s_3$. Hence, the proof. □

Remark 3.1 *To get a path of shorter length between s and t, we would change as few bits in S_{01} and S_{10} as possible.*

Lemma 3.2 *There exists a path of length $p_0 = d + 1 + s_2 - s_1$ between s and t, via the bridge connected to the node in W_0.*

Proof : Starting from s, we reach the node in W_0 by changing all bits in S_{10} and S_{11}. By Lemma 3.1, $p_0 = 1 + s_0 + s_3 + 2s_2$. Putting $s_0 + s_3 = d - (s_1 + s_2)$ we get $p_0 = d + 1 + s_2 - s_1$. □

Example 3.2 Consider $d = 8$, $s = 00000111$, and $t = 01111001$. Thus $S_{00} = \{0\}$, $S_{01} = \{1, 2, 3, 4, \}$, $S_{10} = \{5, 6\}$, $S_{11} = \{7\}$. Hence, $s_1 = 4$ and $s_2 = 2$. Starting from s, three '1' bits are to be changed to zeros to reach 0^8, and from 1^8 three '1' bits are to be changed to '0' to reach t. Hence path length = 7. Also by Lemma 3.2, $p_0 = 8 + 1 + 2 - 4 = 7$.

Lemma 3.3 *For $s_1 > d - r$ there exists a path of length $p_1 = -d + 2r + 1 + s_1 - s_2$, between s and t, via a bridge connected to some node in B_r.*

Proof : $s_1 > d - r$ implies $r > d - s_1$. If we want to reach a node in B_r from s, where $r > d - s_1 = s_0 + s_2 + s_3$, we must change all bits in S_{00} to 1 and also some more bits in S_{01}. The number of bits in S_{01} to be changed is equal to $s_1' = r - (d - s_1) = s_1 - (d - r)$. Note that if the d-th bit is in S_{01}, these s_1' bits must include the d-th bit to reach a node in B_r. The path length p_1 is equal to $1 + 2s_1' + s_0 + s_3$ (by Lemma 3.1). Putting $s_0 + s_3 = d - (s_1 + s_2)$ and $s_1' = s_1 - (d - r)$, we get $p_1 = 1 + 2r - d + s_1 - s_2$. □

Example 3.3 Take $d = 8$, $r = 3$, $s = 10000010$ and $t = 11111101$. Here, $s_1 = 6$, $s_2 = 1$ and $s_3 = 1$. To reach a node in B_3, we change the 7^{th} bit. Then using the bridge we reach a node 01111100. Another two steps leads to t. Hence path length = 4. Also by Lemma 3.3, $p_1 = 1 + 6 - 8 + 6 - 1 = 4$.

Lemma 3.4 *For $s_1 \leq d - r$ and $s_2 \leq r$, there exists a path of length $p = 1 + s_0 + s_3$ between s and t, via a bridge connected to a node in W_r.*

Proof:

Case I: $s_2 + s_3 \leq r$, i.e., the number of 1's in s is less than or equal to r. The number of 0's in a node in W_r is greater than or equal to s_1, since $s_1 \leq d-r$, by the given condition. Hence starting from s, we can reach a node in W_r by changing all (for $s_1 = d-r$) or some (for $s_1 < d-r$) bits in S_{00}. By Lemma 3.1, $p = 1 + s_0 + s_3$.

Case II: $s_2 + s_3 > r$. By the given condition $s_2 \leq r$, i.e., the number of 1's in a node in W_r is greater than or equal to s_2. Hence starting from s, we can reach a node in W_r by changing some bits in S_{11} only. By Lemma 3.1, path length $p = 1 + s_0 + s_3$.

□

Example 3.4 Consider the pair (s, t) of Example 3.2. Since s is already in W_3, we can use a bridge from s to reach 11111000. Another 2 steps leads to t. Hence path length = 3. Also by Lemma 3.4 $p = 1 + s_0 + s_3 = 3$.

Lemma 3.5 *For $s_2 \leq d - r$ and $s_1 \leq r$, there exists a path of length $p = 1 + s_0 + s_3$, between s and t, via a bridge connected to a node in W_r.*

Proof: Similar to Lemma 3.4, except that we start from t. □

Lemma 3.6 *For $r < s_2$, there exists a path of length $p = d - 2r + 1 + s_2 - s_1$ via a bridge connected to a node in W_r.*

Proof: Since $s_2 > r$, the number of 1's in a node in W_r is less than that in s. Thus to reach a node in W_r from s we have to change some '1' bits to '0'. We can change all bits in S_{11} and some $s_2' = s_2 - r$ bits in S_{10} to reach a node in W_r. By Lemma 3.1, $p = 1 + s_0 + s_3 + 2s_2'$. Putting $s_0 + s_3 = d - (s_1 + s_2)$ and $s_2' = s_2 - r$ we get $p = 1 + d - 2r + s_2 - s_1$. □

Example 3.5 Consider $d = 8$, $r = 3$, $s = 00011111$, and $t = 01100001$. Here, $s_0 = 1$, $s_1 = 2$, $s_2 = 4$, and $s_3 = 1$. By changing the rightmost two bits we reach a node 00011100 in W_3. The bridge leads to 11100011. We need another two steps to reach t. Hence path length 5. By Lemma 3.6, $p = 8 - 6 + 1 + (4 - 2) = 5$.

Theorem 3.1 *A bridged hypercube of dimension d ($d \geq 8$), where bridges are connected to all the nodes in $W_0 \cup W_r$, ($r = \lfloor \frac{d}{4} \rfloor + 1$), has diameter equal to $\lceil \frac{d}{2} \rceil$.*

Proof: We can assume w.l.o.g (without loss of generality) that $s_2 \leq s_1$. Also we need to consider only those s, t for which $H(s,t) > \lceil \frac{d}{2} \rceil$, i.e., $s_0 + s_3 \leq \lceil \frac{d}{2} \rceil - 1$. There can be two possible cases :

Case 1 : $s_1 > d - r$

Case 2 : $r \leq s_1 \leq d - r$

Case 2 can, however, be divided into two subcases :

2a) $s_1 \leq d - r$, $s_2 \leq r$

2b) $r < s_1 < d - r$, $s_2 > r$

Note that the case $s_1 = d - r$ and $s_2 > r$ cannot arise, since in that case $s_1 + s_2 > d - r + r = d$, but the total number of bits is only d.

Case 1: By Lemma 3.2, $p_0 = d + 1 + s_2 - s_1$ and by Lemma 3.3, $p_1 = 1 + 2r - d + s_1 - s_2$.
Hence, $\min(p_0, p_1) \leq (p_0 + p_1)/2 = r + 1 \leq \lceil \frac{d}{2} \rceil$.

Case 2a): By Lemma 3.4, the path length $p = 1 + s_0 + s_3 \leq \lceil \frac{d}{2} \rceil$. When $s_0 + s_3 = \lceil \frac{d}{2} \rceil - 1$, the path length $p = \lceil \frac{d}{2} \rceil \leq$HD(s,t).

Case 2b): By Lemma 3.6, $p = d - 2r + 1 + s_2 - s_1 \leq d - 2r + 1$. For $r = \lfloor \frac{d}{4} \rfloor + 1$, $p \leq \lceil \frac{d}{2} \rceil$.

Hence the proof. □

Example 3.6 An illustrative example is given in Fig. 3.1 with $d = 8$, where the solid lines represent the already existing lines and the dotted lines represent the bridges.

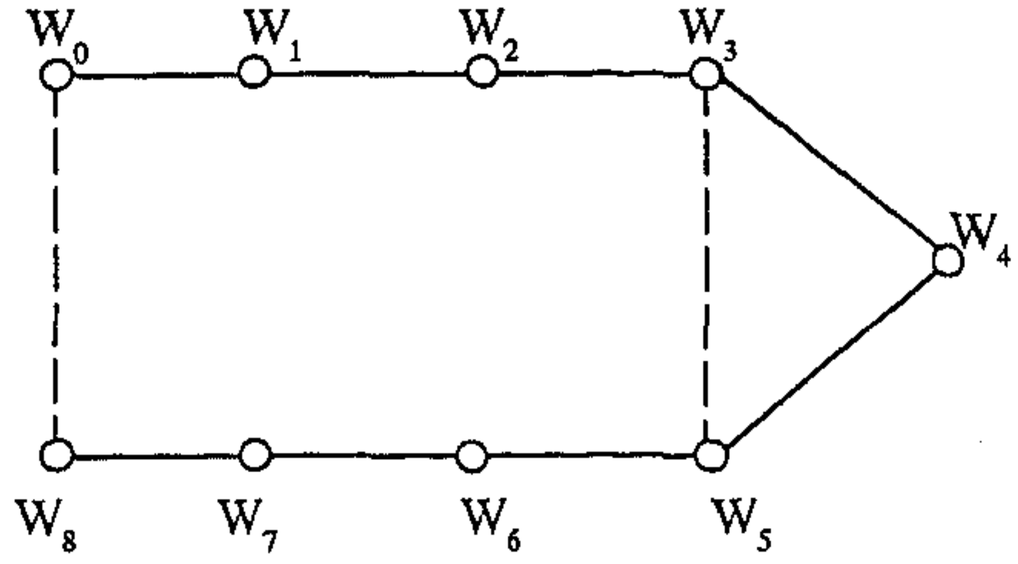


Fig 3.1 A bridged eight-cube

The number of extra edges that we add is small compared to the total number of edges in the d-cube and as we increase d, the ratio $\binom{d}{r}+1 : d.2^{d-1}$ becomes smaller, where $r = \lfloor \frac{d}{4} \rfloor + 1$. The values of this ratio for a few values of d are given in Table 3.1.

TABLE 3.1

Numbers of Bridges Added

d	$\binom{d}{r}+1$	$\binom{d}{r}+1 : d.2^{d-1}$
6	16	.083
8	57	.05
10	121	.02

3.4 Average Path Length in Bridged Q_d

We find the average path length in a bridged hypercube Q_d, $(d > 4)$ which has bridges connected to all the nodes in W_r and W_0, where $r = \lfloor d/4 \rfloor + 1$.

Lemma 3.7 *For any two nodes s and t, the length of a path between them using a bridge is shorter than the hamming distance $H(s,t)$ only if $H(s,t) > \lceil d/2 \rceil$.*

Proof : The length of a path using a bridge of the form $(x, \bar{x})$ is given by $p = 1 + s_0 + s_3 + s'_1 + s'_2$ (by Lemma 3.1). Putting $s_0 + s_3 = d - H(s,t)$, we get $p = 1 + d - H(s,t) + s'_1 + s'_2$. So, for p to be less than $H(s,t)$ we must have

$$1 + d - H(s,t) + s'_1 + s'_2 < H(s,t)$$

i.e., $2H(s,t) > 1 + d$ which implies $H(s,t) > \lceil d/2 \rceil$. □

To find the average path length we consider all possible distinct node pairs (s,t). Evidently there are $\binom{2^n}{2}$ such pairs. We need to consider only the pairs (s,t) for which $s_2 \leq s_1$, since a pair (s,t) with $s_2 > s_1$ is considered as a pair (t,s) with $s_1 > s_2$. First we consider only those pairs (s,t) for which $H(s,t) = s_1 + s_2 > \lceil d/2 \rceil$ so that the shortest path between s and t must be through a bridge. There can be the following two cases :

Case 1 : $s_1 > d - r$

Case 2 : $r \leq s_1 \leq d - r$

For a pair (s,t) in Case 1, the distance between s and t is either $p_0 = d+1-(s_1-s_2)$ (by Lemma 3.1), or $p_1 = 1+2r-d+(s_1-s_2)$ (by Lemma 3.3). If $s_1 - s_2 \geq d-r+1$, then $p_0 < p_1$ and if $s_1 - s_2 \leq d - r$, then $p_1 \leq p_0$. Hence, in Case 1, we consider the following two subcases

Subcase 1a) $s_1 - s_2 \geq d - r + 1$

Subcase 1b) $s_1 - s_2 \leq d - r$

The sum of the distances between s and t for all such pairs in subcases 1a) and 1b) are expressed in the following lemmas.

Lemma 3.8 *The sum of the distances between s and t, for the set of node pairs (s,t) such that $s_1 - s_2 \geq d-r+1$, is given by* $S_1 = \sum_{i=d-r+1}^{d} (d+1-i) f_1(i)$, *where* $f_1(i) = \sum_{k=0}^{\lfloor \frac{(d-i)}{2} \rfloor} \binom{d}{k} \binom{d-k}{k+i} 2^{d-2k-i}$.

Proof : Here, $p_0 < p_1$ and the distance between s and t is equal to $d+1-(s_1-s_2)$. Let $s_1 - s_2 = i$. The number of such node pairs (s,t) for which $s_1 - s_2 = i$ is equal to $f_1(i)$, as given in the statement of the lemma. This can be explained as follows. The number of ways in which k positions for S_{10} and $k+i$ positions for S_{01} can be chosen is given by $\binom{d}{k}\binom{d-k}{k+i}$. The rest $d-2k-i$ positions can be filled up in 2^{d-2k-i} ways. The upper limit of the range of k comes from the condition $k+(k+i) \leq d$. Hence the proof. □

Lemma 3.9 *The sum of the distances between s and t, for all such pairs (s,t), such that $s_1 > d-r$ and $s_1 - s_2 \leq d-r$, is given by*

$$S_2 = \sum_{l=0}^{\lfloor \frac{r-2}{2} \rfloor} \sum_{x=l+1}^{r-1-l} (2+r+l-x) \binom{d}{r-l-1} \binom{r-l-1}{x} 2^{r-l-1-x}$$

Proof : Since $s_1 > d-r$, let $s_1 = d-r+l+1$, where $l \geq 0$. Also, let $s_2 = x$. From the condition $s_1 - s_2 \leq d-r$, we get $x \geq l+1$ and from the condition $s_1 + s_2 \leq d$ we get $x \leq r-1-l$. As $t+1 \leq x \leq r-1-l$, $l \leq \lfloor \frac{r-2}{2} \rfloor$. Here, $p_1 \leq p_0$ and the distance between s and t is equal to $1+2r-d+s_1-s_2 = 1+2r-d+d-r+l+1-x = 2+r+l-x$. The number of such node pairs (s,t) with $s_1 = d-r+l+1$ and $s_2 = x$ is equal to $\binom{d}{d-r+l+1}\binom{r-l-1}{x}2^{r-l-1-x}$. Hence the proof. □

For the set of node pairs (s,t) in Case 2, we consider the following two subcases :

Subcase 2a : $s_2 \leq r$

Subcase 2b : $s_2 > r$.

The next two lemmas give the expression for path length between s and t corresponding to subcases 2a and 2b.

Lemma 3.10 *The sum of the distances between s and t for all pairs (s,t) such that $s_2 \leq r$, $r \leq s_1 \leq d-r$ and $H(s,t) > \lceil d/2 \rceil$, is given by $S_3 = S_3' - \frac{1}{2}(d-2r+1)\binom{d}{r}\binom{d-r}{r}2^{d-2r}$, where $S_3' = \sum_{l=0}^{d-2r}\sum_{x=y}^{r}(d+1-r-l-x)\binom{d}{x}\binom{d-x}{r+l}2^{d-r-l-x}$, and $y = max\ (\lceil d/2 \rceil + 1 - r - l, 0)$.*

Proof : Let $s_1 = r+l$, where $l \geq 0$ and $s_2 = x$. Hence $l \leq d-2r$, for $s_1 \leq d-r$. For $\mathbf{HD}(s,t) = s_1 \dot{-} s_2 > \lceil d/2 \rceil$, $r+l+x \geq \lceil d/2 \rceil + 1$, i.e., $x \geq \lceil d/2 \rceil + 1 - r - l$. By Lemma 3.4, the distance between s and t is equal to $1+s_0+s_3 = 1+d-(s_1+s_2) = 1+d-r-l-x$. The number of node pairs with $s_1 = r+l$ and $s_2 = x$ is equal to $\binom{d}{x}\binom{d-x}{r+l}2^{d-r-l-x}$. This explains S_3'. But for $s_1 = r$ and $s_2 = r$, s and t are interchangeable. To avoid duplication of node pairs (s,t) with $s_1 = r$ and $s_2 = r$, the term $\frac{1}{2}\binom{d}{r}\binom{d-r}{r}2^{d-r}$ is to be subtracted from S_3'. Hence the proof. □

Lemma 3.11 *For the set of node pairs (s,t) such that $s_2 \leq s_1$ and $r < s_1 < d-r$, $r < s_2$, the sum of the distances between them is given by $S_4 = \sum_{i=1}^{d-2r-2}(d-2r+1-i)f_4(i) + \frac{1}{2}(d-2r+1)f_4(0)$, where*

$$f_4(i) = \sum_{k=r+1}^{\lfloor \frac{d-i}{2} \rfloor}\binom{d}{k}\binom{d-k}{k+i}2^{d-2k-i}.$$

Proof : Let $s_2 = k$ and $s_1 = k+i$. Then $i \geq 0$ and as $2k+i \leq d$, $i \leq \lfloor \frac{d-i}{2} \rfloor$. So, the number of node pairs (s,t) with $s_1 - s_2 = i$ is equal to $f_4(i) = \sum_{k=r+1}^{\lfloor \frac{d-i}{2} \rfloor}\binom{d}{k}\binom{d-k}{k+i}2^{d-2k-i}$ for $i > 0$. For $i = 0$, the corresponding number is equal to $\frac{1}{2}f_4(0)$, since when $s_1 = s_2$, s and t are interchangeable. By Lemma 3.6 the distance between s and t is equal to $d-2r+1+s_2-s_1 = d-2r+1-i$. Hence the proof. □

Now we consider the node pairs (s,t) such that $H(s,t) \leq \lceil d/2 \rceil$ and hence the shortest path between s and t does not pass through a bridge.

Lemma 3.12 *For the set of node pairs* (s,t) *such that* $s_2 \leq s_1$ *and* $H(s,t) \leq \lceil d/2 \rceil$, *the sum of the distances between them is equal to* $S_5 = \sum_{i=1}^{\lceil d/2 \rceil} i f_5(i)$, *where* $f_5(i) = \sum_{x=0}^{\lfloor \frac{i-1}{2} \rfloor} \binom{d}{x}\binom{d-x}{i-x} 2^{d-i} + even(i)\frac{1}{2}\binom{d}{i/2}\binom{d-i/2}{i/2}2^{d-i}$ *and* $even(i) = 1(0)$, *if* i *is even(odd).*

Proof : Let $H(s,t) = s_1 + s_2 = i$. By the given condition $i \leq \lceil d/2 \rceil$. The number of node pairs (s,t) with $s_1 + s_2 = i$ is equal to $f_5(i)$. This can be explained as follows. Let $s_2 = x$. Then $s_1 = i - x$ and $x \leq i - x$, i.e., $x \leq \lfloor \frac{i}{2} \rfloor$. When i is odd, $x \leq \frac{i-1}{2}$ and when i is even, maximum value of x is $i/2$. If $x = i/2$, then $s_1 = s_2$ and s and t are interchangeable and the number of node pairs (s,t) with $s_1 = i/2$ and $s_2 = i/2$ is equal to $\frac{1}{2}\binom{d}{i/2}\binom{d-i/2}{i/2}2^{d-i}$. The distance between s and t is equal to $H(s,t) = i$, (by Lemma 3.7). Hence the proof. □

Theorem 3.2 *The average path length in a bridged* Q_d *with bridges connected to all the nodes in* W_0 *and* W_r, $r = \lfloor d/4 \rfloor + 1$, *is equal to* $\bar{P} = \frac{\sum_{i=1}^{5} S_i}{\binom{2^d}{2}}$.

Proof : When $H(s,t) > \lceil d/2 \rceil$, by taking help of the Lemmas 3.8 to 3.11, we can write the sum of the distances between all such pairs (s,t) as $\sum_{i=1}^{4} S_i$. When $H(s,t) \leq \lceil d/2 \rceil$, the sum of the distances between all such node pairs (s,t) is equal to S_5 (by Lemma 3.12). The total number of distinct node pairs (s,t) is $\binom{2^d}{2}$. Hence the proof. □

Example 3.7 *The average path lengths of bridged hypercubes for a few values of d are listed in the following table.*

TABLE 3.2

Average Path Length

dimension	Diameter	Average Path Length
6	3	2.46
8	4	3.29
10	5	4.19
12	6	5.04
16	8	6.85

Here we see that the average path length is very near to the diameter of the bridged Q_d, as against the case in the folded hypercube where the average path length is about $d/4$. But in the folded hypercube bridges are connected to all the nodes in the hypercube whereas in our case the number of bridges constitutes only a small fraction of the total number of links in the original cube. Higher average path length in our case indicates that a large number of node pairs (s, t) have the distances between them nearly equal to the diameter $= \lceil d/2 \rceil$, i.e., the bridges are used most economically.

3.5 Reduction in Diameter by k in Q_d, $(d \geq 2k)$

In the previous section we have considered bridged hypercube whose diameter is reduced to half of its dimension. Now we will consider, how to reduce the diameter of Q_d by some $k \leq \lfloor \frac{d}{2} \rfloor$.

3.5.1 Reduction by an Even Value

First we consider the case when k is even. Assuming that there are bridges connected to all the nodes in W_r, for a given value of r, $0 < r < \frac{d}{2}$, we investigate some more properties regarding paths between s and t.

Lemma 3.13 *For $s_2 \geq r$ and the $(d-1)^{\text{th}}$ bit in S_{01} there is a path of length $p_0 = d - 2r + 1 + s_2 - s_1$, between s and t via a bridge connected to a node in A_r.*

Proof : Starting from s, we can change all bits in S_{11}, and $s'_2 = s_2 - r$ bits in S_{10} to reach a node in A_r. Hence, $p_0 = 1 + s_0 + s_3 + 2s'_2$ (by Lemma 3.1). Putting $s_0 + s_3 = d - (s_2 + s_1)$ and $s'_2 = s_2 - r$ we get $p_0 = d - 2r + 1 + s_2 - s_1$. □

Lemma 3.14 *For $s_1 \geq r$ and the $(d-1)^{\text{th}}$ bit in S_{10} there is a path of length $p_0 = d - 2r + 1 + s_1 - s_2$, between s and t via a bridge connected to a node in A_r.*

Proof : The proof is similar to that of Lemma 3.13, except that we start from t.□

Lemma 3.15 *For $s_2 \geq r$ and the $(d-1)^{\text{th}}$ bit in S_{11} or S_{00} there is a path of length $p_0 = d - 2r + 1 + s_2 - s_1$, between s and t via a bridge connected to a node in A_r.*

Proof : The proof is similar to that of Lemma 3.13. □

Lemma 3.16 *For $s_1 \geq r$ and the $(d-1)^{\text{th}}$ bit in S_{01} there is a path of length $p_1 = d - 2r + 1 + s_1 - s_2$, between s and t via a bridge connected to a node in B_r.*

Proof : Starting from t, we change all bits in S_{11}, and $s'_1 = s_1 - r$ bits in S_{01} (excluding the $(d-1)^{\text{th}}$ bit) to reach a node in B_r. Hence $p_0 = 1 + s_0 + s_3 + 2s'_1$

(by Lemma 3.1). Putting $s_0 + s_3 = d - (s_1 + s_2)$ and $s_1' = s_1 - r$, we get $p_0 = d - 2r + 1 + s_1 - s_2$. □

Lemma 3.17 *For $s_2 \geq r$ and the $(d-1)^{\text{th}}$ bit in S_{10} there is a path of length $p_1 = d - 2r + 1 + s_2 - s_1$, between s and t via a bridge connected to a node in B_r.*

Proof : The proof is similar to that of Lemma 3.16 except that we start from t.□

Lemma 3.18 *For $s_1 \geq r-1$ and the $(d-1)^{\text{th}}$ bit in S_{00} or S_{11}, there is a path of length $p_1 = d - 2r + 3 + s_1 - s_2$, between s and t via a bridge connected to B_r.*

Proof : If the $(d-1)^{\text{th}}$ bit is in S_{11}, we cannot change the $(d-1)^{\text{th}}$ bit to reach a node in B_r. So, starting from t, we can change only $s_3 - 1$ bits in S_{11} and the remaining $s_1' = s_1 - (r-1)$ bits in S_{01}. Hence, $p_1 = 1 + s_0 + s_3 + 2s_1' = d - 2r + 3 + s_1 - s_2$. If the $(d-1)^{\text{th}}$ bit is in S_{00}, we must change it to '1'. So, starting from t we can change all s_3 bits in S_{11} and $s_1' = s_1 - (r-1)$ bits in S_{01} to reach a node in B_r. Hence, $p_1 = d - 2r + 3 + s_1 - s_2$. □

Theorem 3.3 *By adding 8 extra edges to Q_n, $(n \geq 4)$, its diameter can be reduced by 2.*

Proof: **Case I :** $n = 4$. Let us connect all diametrically opposite pairs by bridges. A total of 8 extra edges will be added to the cube in the process. Let us take any two nodes s and t. Let p be the length of the shortest path between s and t. To show that the diameter of this bridged 4-cube is 2 we consider only those (s,t) for which $H(s,t) > 2$. If $H(s,t) = 3$, then $H(\bar{s},t) = 1$ and hence $p = 2$. If $H(s,t) = 4$, then $t = \bar{s}$ and hence $p = 1$.

Case II : $n > 4$. We take a 4-dimensional subcube $C_0 = 0^{n-4}*^4$. We connect bridges within this subcube as in case I, i.e., the bridges are of the form $0^{n-4}x$ to

$0^{n-4}\bar{x}$, where x is a binary string of length 4. Take two nodes $s = ax$ and $t = by$, where x and y are binary strings of length 4, and a and b are binary strings of length $n-4$. Let f be the number of bit positions at which s and t are same. To show that diameter of this bridged n-cube is $n-2$, we consider only those (s,t) for which $H(s,t) > n-2$, i.e., $f \leq 1$.

If $f = 0$, then $x = \bar{y}$. The shortest path between x and y is through C_0 and is of length $n-4+1 = n-3$.
If $f = 1$, there can be two possible cases :

a) If the bit position at which s and t are having the same value is within the rightmost 4 bits, then shortest path between $0^n x$ and $0^n y$ is of length 2 (by case 1). So the shortest path between s and t is through C_0 and is of length $n-4+2 = n-2$.

b) If the bit position at which s and t are having the same value is within the leftmost $n-4$ bits, $x = \bar{y}$. Shortest path between s and t is through C_0 and is of length $n-4$ or $n-2$ depending on whether the bit value is 0 or 1.

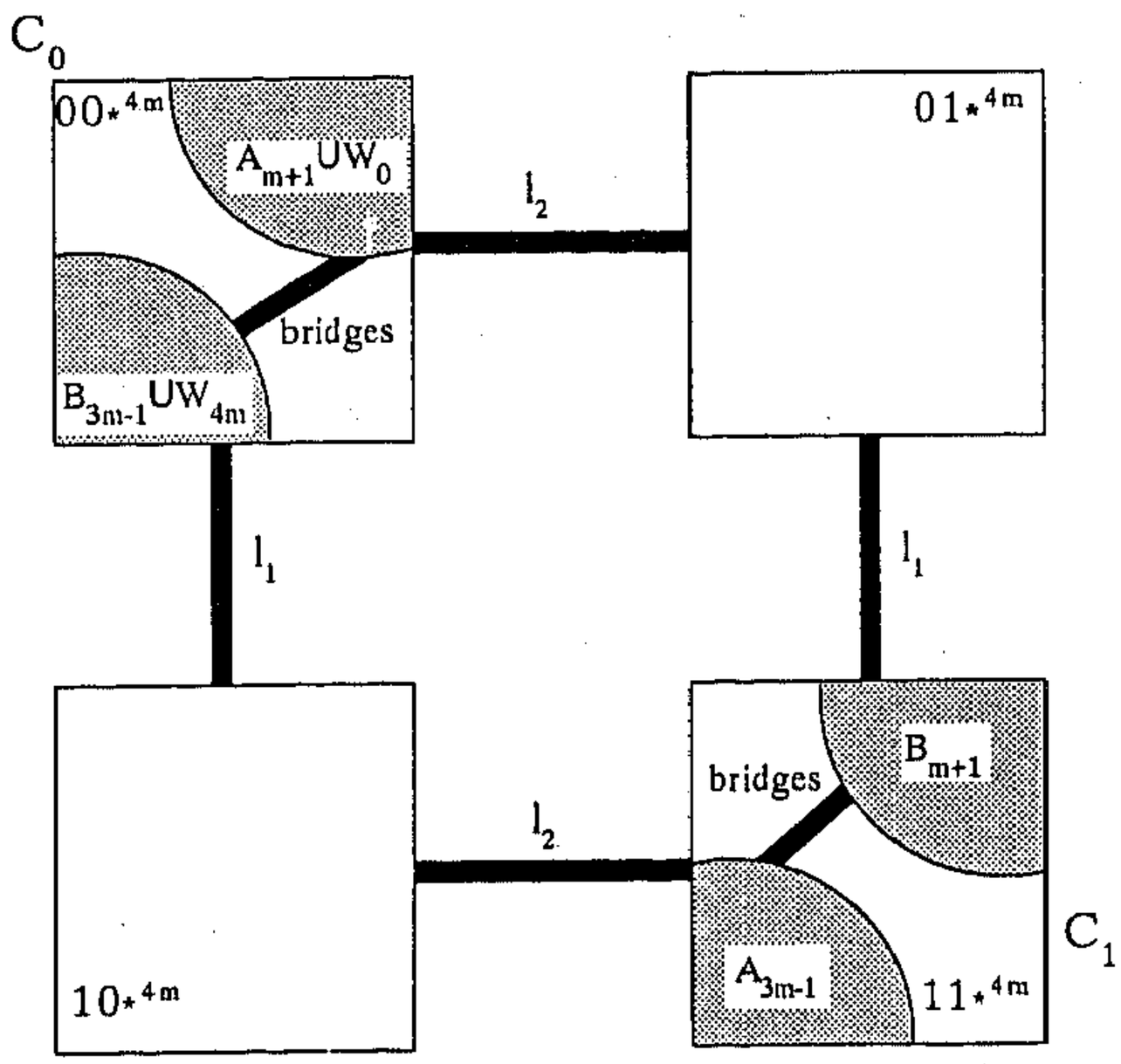
We generalize this idea to reduce the diameter of a hypercube by some even number $2m$ $(m \geq 2)$ as follows. We take an n' cube $(n' = n+4m)$. We take two subcubes $C_0 = 0^n *^{4m}$ and $C_1 = 1^n *^{4m}$. In C_0, we add bridges of the of the form $(0^n x, 0^n \bar{x})$, where x is a bit string of length $4m$, and in C_1 the bridges are of the form $(1^n x, 1^n \bar{x})$. For C_0, C_1 we define the sets W_r, A_r, B_r as for a $4m$-dimensional cube by ignoring the leftmost n bits of the node representation. In C_0, we connect bridges from all the nodes in A_{m+1} and W_0. In C_1, we connect bridges from all the nodes in B_{m+1}. An illustrative example is given in Fig. 3.2 when $n = 2$.

Theorem 3.4 *In an n' cube $(n' = n+4m,\ m \geq 2)$, by adding $\binom{4m}{m+1} + 1$ extra edges in the above mentioned way, we can reduce its diameter by $2m$.*

Proof: We consider two nodes $s = ax$ and $t = by$ in this n' cube, where x and y are binary strings of length $4m$, and a and b are binary strings of length n. Let p_0 be the length of a path between $0^n x$ and $0^n y$ in C_0 and p_1 be that between $1^n x$ and $1^n y$ in C_1.

Among the first n bits from left, let p be the number of bits those are 0 in both s and t, and q be the number of bits those are 1 in both s and t. We define s_0, s_1, s_2 and s_3 for the two strings x and y as before. We consider only those (s,t) for which $H(s,t) > n+2m$, i.e., $p+q+s_0+s_3 \leq 2m-1$. We now consider two paths between s and t, one using a bridge in C_0 and the other using a bridge in C_1. Let the path through C_i be of length P_i, $i \in \{0,1\}$.

We can easily verify that the total number of bits to be changed to get $0^n x$ from s and t from $0^n y$ is equal to $(n-p-q)+2q = n-p+q$. Hence, $P_0 = n-p+q+p_0$. Similarly $P_1 = n+p-q+p_1$. Now, $\min(P_0, P_1) \leq (P_0+P_1)/2 = n+(p_0+p_1)/2$.



l_i : links connecting nodes which differ in the i^{th} bit, $i = 0, 1$

Fig 3.2 A bridged (4m + 2)-cube of diameter 2m + 2

In finding p_0, p_1 we apply the lemmas where $d = 4m$ and $r = m+1$. There can be 4 possible cases as in Theorem 3.1.

Case 1: $s_1 > d - r = 3m - 1$

Case 2a: $s_1 \leq d - r = 3m - 1$, $s_2 \leq r = m + 1$

Case 2b: $m + 1 = r < s_1 < d - r = 3m - 1$, $s_2 > r = m + 1$

Case 3: $s_1 \leq r = m + 1$

Case 1: By Lemma 3.2, $p_0 = 4m + 1 + s_2 - s_1$ and by Lemma 3.3, $p_1 = -2m + 3 + s_1 - s_2$. Hence min $(P_0, P_1) \leq n + m + 2 \leq n + 2m$ since $m \geq 2$.

Case 2a: By Lemma 3.4, at least one of p_0, p_1 is equal to $1 + s_0 + s_3$. Hence at least one of P_0, P_1 is less than or equal to $n + p + q + s_0 + s_3 + 1 \leq n + 2m$, (as $p + q + s_0 + s_3 \leq 2m - 1$).
When $p = q = 0$ and $s_0 + s_3 = 2m - 1$, either P_0 or $P_1 = n + s_0 + s_3 + 1 = n + 2m < H(s, t)$.

Case 2b: If the $(n + 4m - 1)^{\text{th}}$ bit is in S_{00} or S_{11}, $p_0 = 2m - 1 + s_2 - s_1$ (by Lemma 3.15) and $p_1 = 2m + 1 + s_1 - s_2$ (by Lemma 3.18). Hence, min $(P_0, P_1) \leq n + 2m$.
If the $(n + 4m - 1)^{\text{th}}$ bit is in S_{01}, $p_0 = 2m - 1 + s_2 - s_1$ (by Lemma 3.13) and $p_1 = 2m - 1 + s_1 - s_2$ (by Lemma 3.16). Hence, min$(P_0, P_1) \leq n + 2m - 1$.
If the $(n + 4m - 1)^{\text{th}}$ bit is in S_{10}, $p_0 = 2m - 1 + s_1 - s_2$ (by Lemma 3.14) and $p_1 = 2m - 1 + s_2 - s_1$ (by Lemma 3.17). Hence, min$(P_0, P_1) \leq n + 2m - 1$.

Case 3: If $s_2 \leq 3m - 1$, by Lemma 3.5, one of p_0, p_1 is equal to $1 + s_0 + s_3$. Hence at least one of P_0, P_1 is less than or equal to $n + p + q + s_0 + s_3 + 1 \leq n + 2m$ (as $p + q + s_0 + s_3 \leq 2m - 1$).
If $s_2 > 3m - 1$, the case is similar to case 1 if we interchange s and t, hence the proof. □

3.5.2 Reduction by an Odd Value

To reduce the diameter by an odd number say $2m - 1$ $(m \geq 2)$ we take an $n' = n + 4m - 2$ dimensional cube. We add bridges to the subcubes $C_0 = 0^n *^{4m-2}$ and $C_1 = 1^n *^{4m-2}$ as before. Now we will see that this construction does not give a graph of diameter $n + 2m - 1$ for $m > 2$. But for $m = 2$ this construction gives

a graph of diameter $n+2m-1$. To do that, in Theorem 3.4 we substitute d by $4m-2$ and r by m. We note that path length between two points s and t is at most $n+2m-1$ in all cases except in *Case 2b* when the $(n+4m-3)^{\text{th}}$ bit is in S_{00} or S_{11}.

In *Case 2b* when the last bit is in S_{00} or S_{11}

$$P_0 = n - p + q + 2m - \mathbf{1} + s_2 - s_1$$
$$P_1 = n + p - q + 2m + \mathbf{1} + s_1 - s_2$$

If $p-q < s_2 - s_1 - 1$, then $P_1 < n+2m$ and if $p-q > s_2 - s_1 - 1$, then $P_0 < n+2m$. $P_1 = P_0 = n+2m$ only if $p - q = s_2 - s_1 - 1 \cdots$ (a).

For $H(x,y) > n+2m-1 = n+3$, $p+q+s_0+s_3 \leq 2$. Also $s_0+s_3 \geq 1$, since the $(n+4m-3)^{\text{th}}$ bit is in S_{00} or S_{11}. Hence, $p+q \leq 1$. Then $p+q$ can be 1 or zero. If $p+q=1$, $p-q$ is an odd number. But s_0+s_3 is equal to 1. Hence, s_2-s_1 is an odd number. So condition (a) does not hold. If $p+q=0$, i.e., $p=q=0$ we can take either of the path through C_0, which are of lengths $n+2m-1+s_2-s_1$ and $n+2m-1+s_1-s_2$.

Lemma 3.19 *By adding* 16 *edges to an* n*-cube,* $(n \geq 6)$*, we can reduce its diameter by 3.*

Proof : The proof follows from the above discussion where $m=2$, and the number of edges added is equal to $\binom{4m-2}{m} + \mathbf{1} = \binom{6}{2} + 1 = 16$. □

Now we investigate some more properties regarding path lengths in a d-cube which has bridges connected to all the nodes in W_r, $1 < r < d/2$.

Lemma 3.20 *For* $s_1 \leq d-r-1$, $s_2 \leq r$ *and the* $(d-1)^{\text{th}}$ *bit of* s *is zero, there exists a path of length* $p_0 = 1+s_0+s_3$*, between* s *and* t*, via a bridge connected to a node in* B_r.

Proof :

Case i) $s_2+s_3 \leq r$. The number of zeros in s is less than or equal to $d-r-1$ by

the given condition. Hence we can reach a node in A_r by changing at most $s_0 - 1$ bits in S_{00} to '1'. Hence we never change the $(d-1)^{\text{th}}$ bit even if it is in S_{00}. By Lemma 3.1, $p_0 = 1 + s_0 + s_3$.

Case ii) $s_2 + s_3 > r$. Starting from s, we change some bits in S_{11} to reach a node in A_r and hence $p_0 = 1 + s_0 + s_3$ (by lemma 1) □

Lemma 3.21 *For $s_1 \leq d-r$, $s_2 \leq r$ and the $(d-1)^{\text{th}}$ bit of s is 1, there exists a path of length $p_1 = 1 + s_0 + s_3$, between s and t, via a bridge connected to a node in B_r.*

Proof :

Case i) $s_2 + s_3 \leq r$. Starting from s, we change some bits in S_{00} to reach a node in B_r and hence $p_1 = 1 + s_0 + s_3$ (by Lemma 3.1).

Case ii) $s_2 + s_3 > r$. Starting from s, we change at most $s_3 - 1$ bits in S_{11} to reach a node in B_r as $s_2 < r$. Hence we never change the last bit even if it is in S_{11}. Thus, $p_1 = 1 + s_0 + s_3$ (by Lemma3.1) □

To reduce the diameter of a hypercube by an odd number $2m-1$ $(m > 2)$ we make a slight modification to our previous scheme. We describe this scheme as follows. We take n' $(n' = n + 4m - 2)$ cube. We take two subcubes $C_0 = 0^n *^{4m-2}$ and $C_1 = 1^n *^{4m-2}$. In C_0, we add bridges of the form $(0^n x, 0^n \bar{x})$, where x is a bit string of length $4m-2$, and in C_1 the bridges are of the form $(1^n x, 1^n \bar{x})$. For C_0, C_1 we define the sets W_r, A_r and B_r as for a $4m-2$ dimensional cube by ignoring the leftmost n bits of the node representation. In C_0, we connect bridges from all the nodes in A_m and W_0. In C_1, we connect bridges from all the nodes in B_{m+1}. Now we come to the following theorem.

Theorem 3.5 *In an n'-cube, where $n' = n+4m-2$ $(m > 2)$ by adding $2\binom{4m-3}{m}+1$ bridges in the above mentioned way we can reduce its diameter by $2m-1$.*

Proof: We consider two nodes $s = ax$ and $t = by$ in this n' cube, where x and y

are binary strings of length $4m-2$, and a and b are binary strings of length n. We define p, q, p_0, p_1, s_0, s_1, s_2, s_3, P_0 and P_1 as in Theorem 3.4.

We consider only those (s,t) for which $H(s,t) > n+2m-2$, i.e., $p+q+s_0+s_3 \leq 2m-2$. There can be 4 possible cases as in Theorem 3.4.

Case 1 : $s_1 > 3m-3$

Case 2a : $m < s_1 \leq 3m-3$, $s_2 \leq m$

Case 2b : $m < s_1 \leq 3m-3$, $s_2 > m$

Case 3 : $s_1 \leq m$

Case 1 : In Lemma 3.2 we put $d = 4m-2$ and $r = m+1$, to get $p_0 = 4m-1+s_2-s_1$. Similarly from Lemma 3.3, we get $p_1 = -2m+5+s_1-s_2$. Hence, $\min(P_0, P_1) \leq n+m+2 \leq n+2m-1$, as $m > 2$.

Case 2a : If the $(n+4m-3)^{\text{th}}$ bit of s is zero, putting $d = 4m-2$ and $r = m$ in Lemma 3.20, we have $p_0 = 1+s_0+s_3$. Hence, $P_0 \leq n+p+q+s_0+s_3+1 \leq n+2m-1$. Alternatively, if the $(n+4m-3)^{\text{th}}$ bit of s is '1', putting $d = 4m-2$ and $r = m+1$ in Lemma 3.21, we have $p_1 = 1+s_0+s_3$. Hence, $P_0 \leq n+2m-1$. When $p = q = 0$ and $s_0+s_3 = 2m-2$, $P_1 = n+2m-1 < H(s,t)$.

Case 2b : If the $(n+4m-3)^{\text{th}}$ bit is in S_{01}, $p_0 = 2m-1+s_2-s_1$ (by Lemma 3.13) and $p_1 = 2m-3+s_1-s_2$ (by Lemma 3.16). Hence, $\min(P_0, P_1) \leq n+2m-2$.

If the $(n+4m-3)^{\text{th}}$ bit is in S_{10}, $p_0 = 2m-1+s_1-s_2$ (by Lemma 3.14) and $p_1 = 2m-3+s_2-s_1$ (by Lemma 3.17). Hence, $\min(P_0, P_1) \leq n+2m-2$.

If the $(n+4m-3)^{\text{th}}$ bit is in S_{00} or S_{11}, $p_0 = 2m-1+s_2-s_1$ (by Lemma 3.15) and $p_1 = 2m-1+s_1-s_2$ (by Lemma 3.18). Hence, $\min(P_0, P_1) \leq n+2m-1$.

Case 3 : If $s_2 > 3m-3$, we interchange s and t so that Case 1 becomes applicable. If $s_2 \leq 3m-3$, we interchange s and t so that Case 2a becomes applicable and hence the proof. □

3.6 Routing Algorithm

In this section we describe the shortest path routing strategy in a bridged hypercube of dimension d, $(d > 4)$. A packet to be routed from the source node s to the destination node t, is transmitted along with a tag as shown in Fig 3.3. Here, b is a flag bit, C is a d-bit vector and t is the destination node. $b = 1$ indicates that a bridge is to be used and $b = 0$ indicates that we will route the packet without using any bridge. C is called the routing vector. Each bit of C is associated with a specific dimension of the hypercube so that we can transmit the packet along an edge of the hypercube in the i^{th} dimension, provided that the i^{th} bit of C is 1.

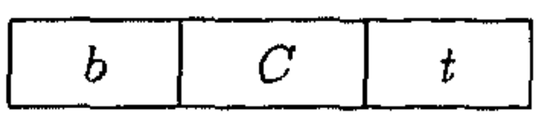


Fig 3.3 Tag used for routing

The algorithm for setting the tag is now described below :

Step 1 : If $s_1 + s_2 \leq \lceil d/2 \rceil$, then set $b \leftarrow 0$ and set $C \leftarrow s \oplus t$; stop
else go to Step 2.

Step 2 : set $b = 1$; /* Here, $s_1 + s_2 > \lceil d/2 \rceil$ */.
/* we have to find the node x from which a bridge $(x, \bar{x})$ will be used.*/
Enumerate s_0, s_1, s_2 and s_3;
Enumerate either p or p_0 and p_1 as done in Theorem 3.1;
Find the minimum path length;
To find the node x to achieve this minimum path length, suitably change certain bits in s in a way as described in the corresponding lemmas;
Set $C \leftarrow s \oplus x$; stop.

The computation of the tag takes $O(n)$ time. Once the tag is attached to the packet, routing at a ny intermediate node u (including s) is done by the following algorithm :

Algorithm $\mathbf{R_b}$;

Step 1 : If $C \neq 0$, then find any i such that the i^{th} bit of C is non-zero.
Set bit i of C to 0 and transmit along dimension i.

Step 2 : If $C = 0$ then check b;
If $b = 0$ then accept the packet /* destination is reached */
else begin
set $b \leftarrow 0$; $C \leftarrow \bar{u} \oplus t$;
transmit to $\bar{u}$ using a bridge;
end;

The algorithm $\mathbf{R_b}$ is very simple and its complexity is no more than that of a routing algorithm in an ordinary hypercube.

3.7 Conclusion

Hypercubes have various applications in parallel processing. This chapter describes different methods for reducing the diameter of a hypercube by adding some extra edges, called bridges. The addition of bridges will not only reduce the diameter but will also reduce the average internode distance. In this chapter, we aimed at reducing the diameter, by adding as few edges as possible. We have added $\binom{d}{r} + 1$ bridges (where $r = \lfloor d/4 \rfloor + 1$) to a d-cube ($d > 4$) to reduce its diameter by $\lfloor d/2 \rfloor$. The extra edges constitute a small fraction of the total number of edges and this fraction is $\binom{d}{r} + 1 : d.2^{d-1}$, which decreases with increase in d. Also, we have given a simple algorithm for shortest path routing in such bridged hypercubes. Another important result is that the number of bridges to be added to reduce the diameter by k, remains constant for all hypercubes of dimension greater than $2k$.

Chapter 4

Twisted Hypercubes

4.1 Introduction

In the previous chapter we have considered the reduction of the diameter of a hypercube by adding a few extra edges, called bridges. Use of such bridges, however, increases the number of links and also makes the structure irregular. We now aim at reducing the diameter of the hypercube without changing the degree of the network. The approach used here involves twisting, i.e., exchanging pairs of independent links [ENS91]. Two edges of a hypercube are called *independent* if they are not incident on a common node.

In what follows, we show that by exchanging 4 pairs of independent edges in a d-cube ($d \geq 5$), we can reduce its diameter by 2. To reduce the diameter by 3, we need to exchange only 16 pairs of independent edges in a d-cube ($d \geq 7$). Also, by exchanging 57 pairs of independent edges, the diameter can be reduced by 4 for $d \geq 10$. To reduce the diameter by $\lfloor \frac{d}{2} \rfloor$, where $d \geq 10$, we need to exchange $\binom{d-1}{r} + 1$ pairs of independent edges, where $r = \lfloor \frac{d}{4} \rfloor + 1$; the number of links exchanged constitutes a small fraction of the total number of links in the hypercube. Also, compared with the twisted hypercubes described in [HK87], the number of link pairs exchanged to reduce the diameter of a d-cube by $\lfloor d/2 \rfloor$ is much smaller in our scheme. Further, it may be noted that the nature of the

twists in our scheme is different from that used in [HK87].

A simple algorithm for point-to-point routing through a shortest path in the proposed twisted hypercube has also been presented.

4.2 Preliminaries

The definitions and notations introduced in the previous chapter are also used here. In addition to those, we define below a twist of type r applied on a d-cube, $0 \leq r < d/2$, as follows :

Definition 4.1 Let y be a bit string of length $(d-1)$ such that $y0 \in A_r$ (A_r is as defined in Chapter 3). A twist of type r involves deleting the link pair $(y0, y1)$ and $(\bar{y}0, \bar{y}1)$ and inserting the two links $(y0, \bar{y}1)$ and $(y1, \bar{y}0)$ for all such y in the hypercube. Such a twist of type r will be denoted by T_r.

It may be noted that a twist of type r is a more general type of twists than the 4-cycle twist considered in [ENS91]. The total number of link pairs exchanged in a twist of type r is equal to $\binom{d-1}{r}$. An illustrative example is shown in Fig. 4.1.

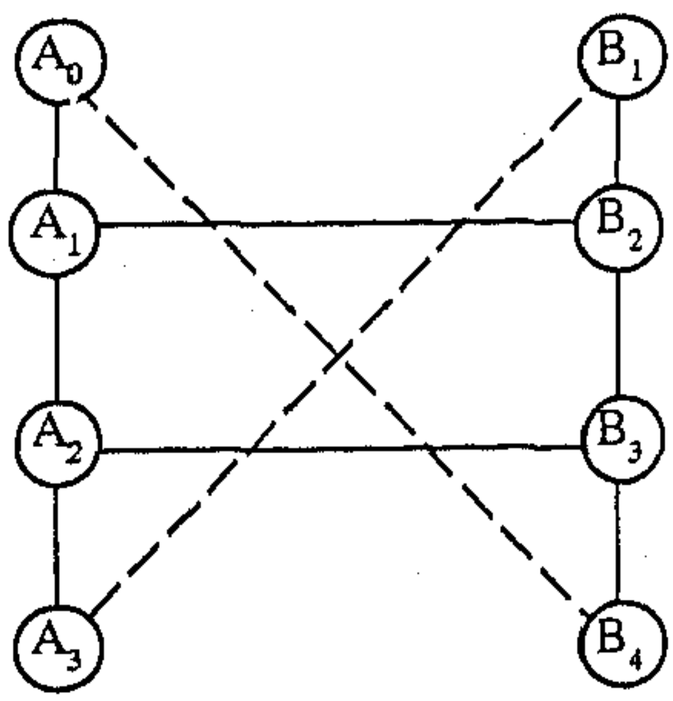


Fig. 4.1 A 4-cube with a twist of type 0

Lemma 4.1 *If we delete links of the type* $(y0, y1)$ *where* $y0 \in A_r$, *for some given* r, *then the path length between any two nodes* s *and* t *remains equal to* $H(s,t)$, *provided* $H(s,t) > 1$.

Proof : Suppose there is a path of length $H(s,t)$ from s to t in the original cube as

$$s \rightarrow a_1 \rightarrow a_2 \rightarrow \cdots \rightarrow a_i \rightarrow a_j \rightarrow \cdots \rightarrow a_k \rightarrow t$$

Suppose the link (a_i, a_j) is deleted. For $H(s,t) > 1$, $s = a_i$ and $t = a_j$, are simultaneously not possible. We can assume w.l.o.g that $s \neq a_i$. We can take an alternative path from s to t by changing the rightmost bit in s first, and then changing the required bits. □

Remark 4.1 If we delete links of type $(y0, y1)$, $y0 \in A_r$ for $r = r_1, r_2, \cdots, r_k$ where

$$\begin{aligned} r_2 - r_1 &> 1 \\ r_3 - r_2 &> 1 \\ &\vdots \\ r_k - r_{k-1} &> 1, \end{aligned}$$

then for two nodes s and t, where $H(s,t) > 1$, we have a path between s and t of length $H(s,t)$.
If $H(s,t) = 1$ and the link (s,t) is deleted, then let $s = y0$ and $t = y1$. There exists a node $y'0$ such that $H(s, y'0) = 1$ and the link $(y'0, y'1)$ is intact. Now we can get a path $s \rightarrow y_1 0 \rightarrow y_1 1 \rightarrow t$, which is of length 3.

4.3 Diameter of Twisted Hypercubes

In what follows we consider the effect of different type of twists on the diameter of Q_d, starting from $d = 4$.

Lemma 4.2 *If we apply a type 0 twist on a 4-cube its diameter becomes equal to 3.*

Proof : Fig. 4.1 shows a 4-cube with a twist of type 0, where new edges are shown by dotted lines. Let s and t be any two nodes in the above cube. If $H(s,t) > 1$, there always exists a path of length $H(s,t)$ after the twist. To show that the diameter of this bridged cube is equal to 3, we need to consider only the cases where $H(s,t) = 1$ or 4. We can assume w.l.o.g that s is in A_k, $0 \leq k \leq 3$.

Case 1 : $H(s,t) = 1$.

The link (s,t) is deleted only for $s \in A_0$ or $s \in A_3$. If $s \in A_0 = \{0000\}$ then t is equal to 0001 for $H(s,t) = 1$ and the link (s,t) is deleted. We take the path $s \rightarrow 0010 \rightarrow 0011 \rightarrow 0001 = t$ which is of length 3. The proof is similar for the case when $s \in A_3$.

Case 2 : $H(s,t) = 4$.

We assume s to be in A_k, $0 \leq k \leq 3$.

If $s \in A_0$ or $s \in A_3$, s and t are directly connected by a link added due to the twist.

If $s \in A_1$, then $t \in B_3$. The path $s \rightarrow 0^4 \rightarrow 1^4 \rightarrow t$ is of length 3. The proof is similar for the case when $s \in A_2$. □

Lemma 4.3 *If we apply a type 1 twist on a 4-cube its diameter becomes equal to 3.*

Proof : Referring to the Fig. 4.2 let s and t be any two nodes in the 4-cube. Path length between s and t can be greater than $H(s,t)$ only if s and t differ in the last bit. We can assume w.l.o g that $s \in A_k, 0 \leq k \leq 3$. Due to symmetry of the structure we consider $s \in A_1$ and $s \in A_0$ only.

a) Let $s \in A_1$

If $t \in B_2$, there can be two cases. If $H(s,t) = 1$, we take the path $s \rightarrow 0^4 \rightarrow 0^3 1 \rightarrow t$ which is of length 3. If $H(s,t) = 3$, we take the path $s \rightarrow \bar{s} \rightarrow t$, which is of length 2.

If $t \in B_3$, there can be two cases. If $H(s,t) = 4$, path length is equal to 1 owing to a direct link provided by the twist. If $H(s,t) = 2$, then we take the path

$s \rightarrow \bar{s} \rightarrow 1^4 \rightarrow t$, which is of length 3.

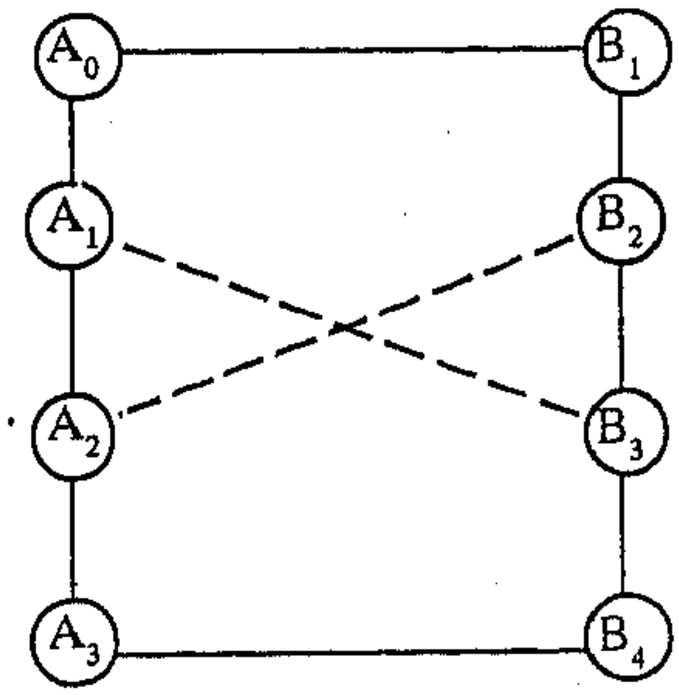


Fig 4.2 A 4-cube with a twist of type 1

b) Let $s \in A_0$

If $t \neq \bar{s}$, there is always a path between s and t of length $H(s,t)$. If $t = \bar{s} = 1111$, we take the path $0^4 \rightarrow 0^3 1 \rightarrow 1^3 0 \rightarrow t$ which is of length 3. □

Theorem 4.1 *By exchanging 4 link pairs in Q_{n+4}, $(n > 0)$ we can reduce its diameter by 2.*

Proof : We apply T_0 within a 4-dimensional subcube $C_0 = 0^n *^4$. This involves exchange of 1 link pair. We apply T_1 within another 4-dimensional subcube $C_1 = 1^n *^4$. This involves exchange of 3 link pairs.

Take any two nodes s and t in Q_{n+4}. Let $s = ax$ and $t = by$ where a and b are strings of length n. Let the path length between s and t be represented by $p(s,t)$.

Case I. If $a = b = 0^n$, $p(s,t) \leq 3$ (by Lemma 4.2);

Case II. If $a = b = 1^n$, $p(s,t) \leq 3$ (by Lemma 4.3);

Case III. If $H(s,t) \leq n+2$, then $p(s,t) \leq n+2$

Case IV. $H(s,t) = n+4$, then $x = \bar{y}$ and there exists a path of length $n+1$ from s to t via one of C_0 and C_1 .

Case V. $H(s,t) = n+3$. In this case s and t must agree in one bit position.

1) Suppose the bit common to s and t is within the leftmost n bit positions, i.e., $x = \bar{y}$. Either $0^n x$ is connected to $0^n \bar{x}$ or $1^n x$ is connected to $1^n \bar{x}$. Hence the path length is at most $n+1+1 = n+2$.

2) The bit common to s and t is within the rightmost 4 bit positions, i.e., $a = b$.

Subcase a) If $\bar{x}$ and y do not differ in the rightmost bit, then let $z \in \{0,1\}$ and the path
$s = ax \rightarrow z^n x \rightarrow z^n \bar{x} \rightarrow z^n y \rightarrow by = t$, is of length $n+2$.

Subcase b) If $\bar{x}$ and y differ in the rightmost bit, then there may be the following cases.

i) If $b \neq 0^n$ or 1^n, then the path $s \rightarrow z^n x \rightarrow z^n \bar{x} \rightarrow b\bar{x} \rightarrow by$ is of length $n+2$

ii) If $b = 1^n$, then for $x \in A_0 \cup A_3 \cup B_1 \cup B_4$, y is in $A_0 \cup A_3 \cup B_1 \cup B_4$ and the path $s = 0^n x \rightarrow 0^n \bar{x} \rightarrow 10^{n-1} \bar{x} \rightarrow 10^{n-1} y \rightarrow 1^n y$ is of length $n+2$.

On the other hand for $x \in A_1 \cup A_2 \cup B_2 \cup B_3$, y is in $A_1 \cup A_2 \cup B_2 \cup B_3$ and the path
$s = 0^n x \rightarrow 0^n \bar{y} \rightarrow 1^n \bar{y} \rightarrow 1^n y$ is of length $n+2$.

iii) If $a = 1^n$, it can be similarly proved that $p(s,t) = n+2$. □

Before investigating the effect of a twist of type r on a d-cube, we give a lemma regarding the path between two nodes s and t, when new links of the form $(x, \bar{x})$ have been inserted for all $x \in A_r$.

Lemma 4.4 *For $s_1 > d-r-1$ and the $(d-1)^{\text{th}}$ bit in s is 0, there exists a path of length $p_1 \leq 2r+2$, between s and t, via a link of the form $(x, \bar{x})$ for some $x \in A_r$.*

Proof : For $s_1 > d-r-1$, $r \geq d-s_1$. If the $(d-1)^{\text{th}}$ bit is in S_{01}, we can reach a node in A_r from s, by changing all bits in S_{00} to 1 and also some $s_1' = s_1 - (d-r)$ bits in S_{01}. The path length $p_1 = 1 + 2s_1' + s_0 + s_3$ (by Lemma 3.1). Putting

$s_0 + s_3 = d - (s_1 + s_2)$ and $s_1' = s_1 - (d - r)$, we get $p_1 = 1 + 2r - d + s_1 - s_2$. Since $s_1 - s_2 \leq d$, $p_1 \leq 2r + 1$. If the $(d-1)^{\text{th}}$ bit is in S_{00}, we change all but the $(d-1)^{\text{th}}$ bit in S_{00} and $s_1' + 1$ bits in S_{01} to reach a node in A_r. Thus $p_1 = 2r - d + 3 + s_1 - s_2$. In this case $s_1 - s_2 \leq d - 1$. Hence $p_1 \leq 2r + 2$. □

Lemma 4.5 *In a d-cube (d being an even number), if we apply twists of type r, where r is an integer between 1 and* $\frac{d}{2} - 2$, *path length between any two nodes s and t is less than or equal to* $max(2r + 2, d - 2r + 1, d/2, 4)$.

Proof: First consider that only new links are added, but no links are deleted. We can assume w.l.o.g. that $s_1 \leq s_2$. Because of symmetry of the twisted cube we assume s to be in some $A_k, 0 \leq k \leq d - 1$ (Proof for the other case is similar).

We define s_0, s_1, s_2 and s_3 as done in the previous Chapter. There can be 4 possible cases as in Theorem 3.1.

Case 1 : $s_1 > d - r - 1$

Case 2a : $s_1 \leq d - r - 1$, $s_2 \leq r$

Case 2b : $r < s_1 < d - r - 1$, $s_2 > r$

Case 3 : $s_1 \leq r$

Case 1 : By Lemma 4.4, the path length $p \leq 2r + 2$.

Case 2a : By Lemma 3.20, the path length $p = 1 + s_0 + s_3$. If $H(s,t) > d/2$, then $p \leq d/2$.

Case 2b : If the rightmost bit in $s \in S_{01}$, by Lemma 3.13 and 3.16, we can show that the path length $p \leq d - 2r$. If the rightmost bit in $s \in S_{00}$, by using Lemma 3.15 and 3.18, we can show that the path length $p \leq d - 2r + 1$.

Case 3 : If $s_2 > d - r - 1$, we interchange s and t so that Case 1 becomes applicable. If $s_2 \leq d - r - 1$, we interchange s and t so that Case 2a becomes applicable.

Now it follows that the path length is less than or equal to max $(2r + 2, d - 2r + 1, d/2)$.

Regarding the deletion of links we make the following observations :

Observation 1: For $H(s,t) > 1$, there is always a path of length $H(s,t)$ between s and t (by Lemma 4.1).

Observation 2: For $H(s,t) = 1$, and the $(d-1)^{\text{th}}$ bit of s and t are identical, the link (s,t) is not deleted.

Observation 3: For $H(s,t) = 1$, and the link (s,t) is deleted, there exists a path of length 3, between s and t. In this case s is connected to $\bar{s}$ and t is connected $\bar{t}$.

Now, consider a path between s and t via a link $(x,\bar{x})$ which is of length $H(s,x) + 1 + H(\bar{x},t)$ before deletion. If $H(s,x) > 1$ and $H(\bar{x},t) > 1$, the path length does change after deletion (by obs. 1). If $H(s,x) = 1$ and the link (s,x) is deleted then s is connected to $\bar{s}$. If $H(\bar{s},t) > 1$, then there is a path of length $1 + H(\bar{s},t)$ between s and t (by obs. 1), which is less than or equal to $d/2$ for $H(s,t) > d/2$. If $H(\bar{s},t) = 1$, the path between s and t is of length at most 4 (by obs. 3). □

Lemma 4.6 *In a d-cube (d being an even number), if we apply twists of type 0 and type r, where r is an integer between 1 and* $d/2-1$, *path length between any two nodes s and t is less than or equal to* $max(r+2, d-2r+1, d/2, 5)$.

Proof : First consider that only new links are added but no link is deleted. We can assume w.l.o.g that $s_1 \leq s_2$. Because of symmetry of the twisted cube we assume s to be in some A_k, $0 \leq k \leq d-1$ (proof for the other case is similar).

We define s_0, s_1, s_2 and s_3 as before. There can be 4 possible cases as in Theorem 3.1.

Case 1 : $s_1 > d-r-1$

Case 2 : $s_1 \leq d-r-1$, $s_2 \leq r$

Case 2b : $r < s_1 < d-r-1$, $s_2 > r$

Case 3 : $s_1 \leq r$

Case 1 : By using Lemma 3.2 and 3.3 we can show that the path length $p \leq r+2$. Other cases are treated as in Lemma 4.5.

Now it follows that the path length is less than or equal $\max(r+2, d-2r+1, d/2)$.

Regarding deletion of links we have the following observations.

Observation 1. For $H(s,t) > 2$, there is always a path of length $H(s,t)$ between s and t.

Observation 2. For $H(s,t) \leq 2$, there is no path of length $H(s,t)$, between s and t, only if both the links $(s,\bar{s})$ and $(t,\bar{t})$ are present.

Observation 3. For $H(s,t) = 1$, and if the links $(s,\bar{s})$ and $(t,\bar{t})$ are present, then there exists a path of length at most 3, between s and t.

Observation 4. For $H(s,t) = 2$, and if the links $(s,\bar{s})$ and $(t,\bar{t})$ are present, then there exists a path, of length at most 4, between s and t.

Now, consider a path between two nodes s and t via a link $(x,\bar{x})$ which is of length $H(s,x) + 1 + H(\bar{x},t)$ before deletion. If $H(s,x) > 2$ and $H(\bar{x},t) > 2$, the path length does not change after deletion (by obs. 1). Consider $H(s,x) \leq 2$ and s is connected to $\bar{s}$. If $H(\bar{s},t) > 2$, then there is a path of length $1 + H(\bar{s},t)$ between s and t (by obs. 1), which is less than or equal to $d/2$ for $H(s,t) > d/2$. If $H(\bar{s},t) \leq 2$, the path between s and t is of length at most 5 (by obs. 3 and 4). □

Lemma 4.7 *If we apply a twist of type 1, on a d-cube (d being an even number ≥ 6), its diameter is reduced by 2.*

Proof : By Lemma 4.5 the path length is less than or equal to $\max(d-2r+1, d/2, 2r+2, 4)$. Putting $r = 1$ we get maximum path length of $d-1$ corresponding to $d-2r+1$. But the path length is less than $d-2r+1$ if the $(d-1)^{\text{th}}$ bit of s and t are different. If the $(d-1)^{\text{th}}$ bit is in S_{00} or S_{11} the path length is $p_0 = d-2r+1+s_2-s_1$ (by Lemma 3.15) and $p_1 = d-2r+1+s_1-s_2$ (by Lemma 3.18). In this case the path length becomes equal to $d-2r+1$ only if $s_1 = s_2$, i.e., when $s_1 + s_2$ is an even number. Then $H(s,t) \leq d-2$. □

Lemma 4.8 *In a six-cube, if we apply twist of type 0 and type 2, maximum path length between any two nodes is less than or equal to 4.*

Proof : According to Lemma 4.6 the maximum path length is 5 for $d = 6$ and $r = 2$. But the path length can be 5, only if $H(\bar{s}, t) = 2$. In that case $H(s, t) = 4$. Hence path length is at most 4. □

We take a $d = n + 6$ cube ($n \geq 1$). In this d-cube we take two subcubes, $C_0 = 0^n*^6$ and $C_1 = 1^n*^6$. We apply twist of type 0 and 2 in C_0 and twist of type 1 in C_1. The total number of link pairs exchanged in the process is 16. Now we have the following theorem.

Theorem 4.2 *By applying twists on a d-cube as above ($d \geq 7$), we can reduce its diameter by 3.*

Proof : Take two nodes (s, t) in the above cube. When both s and t are in C_0 or C_1, there is a path of length at most 4 between them. We can assume w.l.o.g that t is outside C_0, C_1. From the discussions in section 3.5.2, it follows that if no links were deleted in C_0, C_1, then the d-cube would have had diameter $d - 3$. Now also this result holds, as whatever links are deleted are of the type which change the $(d-1)^{\text{th}}$ bit. If required, we can change the $(d-1)^{\text{th}}$ bit outside C_0, C_1. □

Lemma 4.9 *In an 8-cube, if we apply T_0 and T_3 its diameter becomes equal to 5.*

Proof : Follows from Lemma 4.6. □

Lemma 4.10 *If we apply a twist of type 2 in an 8-cube, its diameter becomes equal to 6.*

Proof : Follows from Lemma 4.5. □

Lemma 4.11 *If we apply T_0 and T_3 its diameter becomes equal to 5.*

Proof : Follows from Lemma 4.6 if we replace $d/2$ by $\lceil d/2 \rceil$. □

In a $d = n + 8$ cube ($n \geq 2$) we exchange 57 link pairs in the following way. We apply twists of type 0 and 3 within one 8-dimensional subcube 0^n*^8. We apply twist of type 2 within another subcube 1^n*^8. Now we have the following theorem.

Theorem 4.3 *The above d-cube has diameter equal to $d - 4$.*

Proof : The proof is similar to that of Theorem 4.2. □

Theorem 4.4 *In a d-cube (d being an even number ≥ 10) if we apply twists of type 0 and r, where $r = \lfloor d/4 \rfloor + 1$, its diameter becomes equal to $d/2$.*

Proof : The proof follows from Lemma 4.6. □

The number of link pairs exchanged is small compared to the total number of links in the d-cube. The following table shows the number of links exchanged l_e, the total number of links L and also the ratio l_e/L. It is shown that the ratio l_e/L decreases with increase in d.

Table 4.1: Numbers of links exchanged

d	l_e	L	l_e/L
10	170	5120	.033
12	662	24576	.027
14	1432	114688	.012

4.3.1 Reduction in Diameter by k, $(k \geq 5)$ in Q_{n+2k} $(n > 0)$

If we want to reduce the diameter by k in Q_{n+2k}, we apply some twists in a $2k$-dimensional subcube $C_0 = 0^n*^{2k}$ so that any two nodes within this subcube have

a path between them of length at most k. In another $2k$-dimensional subcube $C_1 = 1^n*^{2k}$ we apply some twists so that any two nodes within this subcube have a path between them of length at most $k+1$. Now we state the following theorem.

Theorem 4.5 *The $(n+2k)$-cube twisted as above will have diameter equal to $n+k$.*

Proof : We define x, y, p_0, p_1, p and q as in Theorem 3.4. Hence $p_0 \leq k$ and $p_1 \leq k+1$. If $p \geq q$, the path between s and t via C_0 is of length $n-p+q+p_0 \leq n+k$. If $p < q$, the path between s and t via C_1 is of length $n+p-q+p_1 < n+k+1$.□

4.3.2 Reduction in Diameter by $2m$ $(m \geq 4)$ in Q_{n+4m} $(n > 0)$

We take two $4m$-dimensional subcubes $C_0 = 0^n*^{4m}$ and $C_1 = 1^n*^{4m}$. We apply T_0 and T_{m+1} within C_0 so that any two nodes in C_0 are apart by a distance of at most $2m$. We apply some twists within C_1 so that any two nodes in C_1 are apart by a distance of at most $2m+2$. We call the resultant graph Q'_{n+4m} and claim that the diameter of Q'_{n+4m} is $n+2m$.

Theorem 4.6 *The $(n+4m)$-cube twisted as above will have diameter equal to $n+2m$.*

Proof : We represent two nodes s and t as $s = ax$ and $t = by$ where x and y are strings of length $4m$, and a and b are strings of length n. We define p, q, p_0, p_1, s_0, s_1, s_2, s_3, P_0 and P_1 as in Theorem 3.4. Hence $p_0 \leq 2m$ and $p_1 \leq 2m+2$. We consider only those (s,t) for which $H(s,t) > n+2m$, i.e., $p+q+s_0+s_3 \leq 2m-1$.

i) $p_0 \neq 1+s_0+s_3$. Here, $p_0 \leq \max(m+2, 2m-1) \leq 2m-1$. Hence, if $q-p \leq 1$, then $P_0 = n+(q-p)+p_0 \leq n+2m$. If $q-p \geq 2$, then $P_1 = n-(q-p)+p_1 \leq n+2m$.
ii) $p_0 = 1+s_0+s_3$. Here, $P_0 = n-p+q+1+s_0+s_3 \leq n+p+q+1+s_0+s_3 \leq n+2m$.
□

Let $X(k)$ denote the number of link pairs to be exchanged to reduce the diameter by k in Q_{n+2k} for $(n > 0)$. Then we can write,

for $k = 2m$, and $m > 2$, $X(k) = \sum_{i=3}^{m} \binom{4m-1}{m+1} + 57$. [Note that, $X(4) = 57$].

for $k = 2m+1$, and $m > 2$, $X(k) = X(2m) + \binom{4m+1}{m+1} + 1$

4.4 Routing Algorithm

We now describe the routing strategy for a twisted hypercube Q_d^T, $(d \geq 10)$ on which we apply T_0 and T_r $(r = \lfloor \frac{d}{4} \rfloor + 1)$.

Note that there are two types of links present in the twisted hypercube Q_d :
i) The links those were present in the original hypercube, i.e., the links connecting two nodes differing in exactly one bit position.
ii) The exchanged links of the form $(x, \bar{x})$.

The algorithm gives the shortest route between two nodes s and t which would involve at most one exchanged link. An exchanged link will be included in the route only when the distance between s and t is more than $\lfloor d/2 \rfloor$. If the shortest path does not involve any exchanged link and $H(s,t) > 1$, then the routing is done in the same way as in the original hypercube except that a suitable ordering of the dimensions of the links is made for traversal. Otherwise, if s and t are such that $H(s,t) = 1$, but the original link connecting s and t is exchanged, then the shortest path will be of length 3.

In the remaining cases, when the shortest path involves an exchanged link of the form $(x, \bar{x})$, routing from s to x and $\bar{x}$ to t are done in the same way as in the original hypercube (here, $H(s,x) < d/2$ and $H(\bar{x},t) < d/2$) with appropriate ordering of the dimensions as mentioned above.

A packet destined to a node t from a source node s is associated with a tag like this :

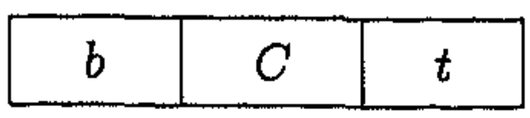


Fig 3.3 Tag used for routing

Here, C is an n-bit routing vector such that if the i^{th} bit of C ($i \neq d-1$) is 1, then the packet will be transmitted through the link of dimension i. b is a single bit information which is set to 1, if the path between s and t passes through an exchanged link of the form $(x, \bar{x})$ and 0, otherwise. The algorithm for setting the tag is described below.

Step 1 : If $H(s,t) \leq \lceil d/2 \rceil$, then set $b \leftarrow 0$; $C \leftarrow s \oplus t$; stop.
else set $b \leftarrow 1$ and go to Step 2

Step 2 : /* Here, the path involves a link of the form $(x, \bar{x})$ and our aim is to find x. */
A) If $s \in A_k$, $0 \leq k \leq d-1$ then find-x;
B) If $s \in B_k$, $1 \leq k \leq d$ then
begin
/* As the structure is symmetric, x can be found this way */
$r \leftarrow 0$; /* r is a d bit vector */
set the $(d-1)^{\text{th}}$ bit of r to 1.
$s' \leftarrow s \oplus r$; $t' \leftarrow t \oplus r$;
$y \leftarrow$ find-x(s', t'); $x \leftarrow y \oplus r$;
end;

Step 3 if t and $\bar{x}$ differs only in the $d-1$th bit then
$x \leftarrow x \oplus r$;
$C = s \oplus x$;

Procedure **find-x** is used to determine the node x at which the exchanged link is to be taken.

Procedure find-x(s,t);

i) Enumerate s_0, s_1, s_2 and s_3.

ii) Find the minimum path length p as done in the proof of Lemma 4.5.
iii) Find the node x to achieve this minimum path length, by suitably changing certain bits in s, as described in the corresponding lemmas.

Note that computation of the tag requires $O(n)$ time because of Step i) in the Procedure **find-x**. Once the tag is attached to the packet, routing at each individual node u (including s) is done by the following algorithm :

Algorithm R_T;

Step 1: If $C = 0$ then go to step 4

Step 2: If the $(d-1)^{th}$ bit of C is 0 then /* Routing as in original cube */

find any i such that the i^{th} bit of C is *non-zero* ;

else /* the $(d-1)^{th}$ bit of C is 1 */

if the link along dimension $d-1$ is not exchanged then $i \leftarrow d-1$

else

begin

if $\exists$ any $j \neq d-1$ such that the j^{th} bit of C is non-zero then $i \leftarrow j$;

else $i \leftarrow 0$;

/* Here though $H(s,t) = 1$, distance between s and t in the twisted cube is 3. */

/* Note that i can be set to any value other than $d-1$ */

end;

Step 3: Flip the i^{th} bit of C and transmit along dimension i.

Step 4: check b; If $b = 0$, then accept the packet

else begin

set $C \leftarrow \bar{u} \oplus t$;

$b \leftarrow 0$; transmit to $\bar{u}$.

end

The complexity of the algorithm R_T is of the same order as that of a routing algorithm in the original hypercube.

4.5 Conclusion

This chapter describes a different method of reducing the diameter by exchanging some pairs of independent edges (without adding extra edges) or *twisting*. Twisting of hypercubes has an extra advantage of reducing the diameter without changing the degree of nodes. Also, the number of link pairs exchanged to reduce the diameter by a given value is much small compared to that given in [HKS87]. For example, we apply twists of type 0 and type 3 in an 11 cube to get a graph of diameter 6. The number of link pairs exchanged in the process is 121, which is much smaller than that required by the method given in [HKS87] which is equal to $10.2^7 = 1280$. It may also be noted that the twists which we have proposed in this chapter are different in nature from those given in [HKS87]. A simple algorithm for point-to-point routing in the twisted hypercube has also been developed.

Chapter 5

Fault Diameter of Hypercubes

5.1 Introduction

One of the important concerns in dealing with hypercube interconnection scheme is its fault-tolerance. The diameter of a faulty n-cube in the presence of at most $n-1$ node failures has been considered in [KK87]. Though an n-cube may be disconnected in the presence of f ($\geq n$) faulty nodes, it is highly unlikely that all the n neighboring nodes of a single node will be faulty. In other words, even with f ($\geq n$) node faults, the cube may remain connected. In such cases, finding the diameter of the faulty cube is an interesting topic of research. Esfahanian [E89] introduced the concept of forbidden faulty sets such that all the nodes in a forbidden faulty set cannot be faulty at the same time. In [E89], a forbidden faulty set was constructed by taking all n-neighbors of a node in the cube. Thus there can be 2^n forbidden faulty sets. Under this forbidden faulty set model, it has been shown that an n-cube remains connected even in the presence of $2n-3$ faults. In [L93a], the fault-diameter of such a faulty n-cube with $f = 2n-3$ has been shown to be at most $n+2$.

In this chapter, we have extended the earlier results and considered the presence of $3n-6$ faulty nodes in an n-dimensional cube for $n \geq 4$. In our case, we define the set of the forbidden faulty sets as follows : 1) all n neighbors of a node constitute

a forbidden faulty set as defined in [E89] and also, 2) all the $2n-2$ neighbors of a pair of adjacent nodes form a forbidden faulty set.

With this definition of the set of forbidden faulty sets, we can argue as follows that in an n-cube, a set of 3 nodes can be disconnected in the presence of $3n-5$ faulty nodes. Consider a path consisting of 3 nodes u, v and w, where v is adjacent to both u and w. Note that u and w cannot be adjacent. The node u has $n-1$ neighbors other than v, the node v has $n-2$ neighbor nodes other than u and w, and the node w has $n-1$ neighbors other than v of which one is a common neighbor to u. So u, v and w have a total of $3n-5$ neighbor nodes. Hence, even under the assumed forbidden faulty set model, a set of 3 nodes can be disconnected in the presence of $3n-5$ faulty nodes.

We will show that under the above model of forbidden faulty sets, the n-cube will remain connected even with $3n-6$ faulty nodes and the fault diameter will be $n+3$. Further, only those nodes which are at a Hamming distance of $n-3$, can be at the maximum distance of $n+3$ in the presence of faults. It will also be shown that all such n-cubes with diameter $n+3$ are isomorphic for $n > 5$. We give an algorithm for routing in such a faulty hypercube which computes a path between two nodes u and v such that the length of the path is at most 6 more than the Hamming distance between u and v. The time complexity of this algorithm is $|F| \log n$, where F is the set of faulty nodes in the n-cube.

5.2 Definitions and Notations

A hypercube of dimension n consists of 2^n nodes and $n.2^{n-1}$ edges. Each node is labeled by a binary string $a_{n-1}\ a_{n-2} \ldots a_1\ a_0$ and two nodes are adjacent if and only if their labels differ in exactly one bit position. The links are labeled from 0 to $n-1$, where the link i connects two nodes which differ in the i-th bit, the rightmost bit being the 0-th bit.

For any two nodes u and v, the Hamming distance between them is denoted by

$H(u,v)$, which is the length of a shortest path between u and v in a fault-free hypercube. The actual distance between two nodes u and v, in the presence of faults is denoted by $d(u,v)$.
A node adjacent to u along the link i (or the dimension i) is denoted by u_i.

For a source-destination pair (u,v) with $H(u,v)=m$, we define the following :
$S(u,v)$: set of bit positions in which u and v differ, say $\{x_1, x_2, \cdots, x_m\}$.
$E(u,v)$: set of bit positions in which u and v are the same, say $\{e_1, e_2, \cdots, e_{n-m}\}$.
$N(u)$: set of fault-free neighbors of u.
$A(u)$: set of dimensions accessible from u, i.e., $A(u) = \{i|$, such that u_i is fault-free$\}$.
$C(u,v) = A(u) \bigcap A(v)$
n_b : total number of faulty neighbors of u and v.

We can reach v from u by traversing along the links x_1, x_2, $\cdots$, x_m in any order. A path between u and v is denoted by a string of the form $(x_1\ x_2\ \cdots\ x_k)$, which is obtained by traversing along the links of these dimension in this order. We use the notation Cir$(x_1\ x_2 \cdots x_m)$ to denote the set of paths obtained by a circular permutation of the string $x_1x_2\cdots x_k$. For example Cir(0 1 2) denotes the set of paths (0 1 2), (1 2 0) and (2 0 1). Similarly, (0 Cir(1 2) 3) denotes the set of paths (0 1 2 3) and (0 2 1 3). Also the notation Cir$(x_1\ x_2\ \ldots\ \bar{x}_i\ \ldots\ x_m)$ will be used to denote the set of paths obtained by the circular permutations of all the literals x_j's in the string except x_i which will be kept fixed at the i-th position. For example Cir(0 1 $\bar{2}$ 3) denotes the set of three paths (0 1 2 3), (1 3 2 0) and (3 0 2 1).

Paths between two nodes u and v are said to be node-disjoint if they do not have any common node except u and v. However, following the idea in [L93a], the definition of node-disjoint will be extended in our discussion as follows : Paths between u and v are said to be node-disjoint if they have no node in common which is potentially faulty.

We use the notation $F(Q_n)$ to denote the set of faulty nodes within Q_n. The

fault-diameter D_f of a hypercube with f faults is defined as the maximum of the diameters of all possible hypercubes which remain connected with f faulty nodes.

5.3 Bounds on the Path Lengths

In what follows, we take the help of some useful results from [L93a] regarding $d(u,v)$ for any two nodes u and v in Q_n.

Result 5.1 If $|F(Q_n)| \leq n-1$, then $d(u,v) \leq H(u,v)+2$ and $d(u,v) = n$ when $H(u,v) = n$.

If $|F(Q_n) \leq 2n-3$, for $|N(u)| \geq 1$ and $|N(v)| \geq 1$, the following results hold :

Result 5.2 $d(u,v) \leq H(u,v)+4$.

Result 5.3 For $H(u,v) = n$, $d(u,v) = n$.

Result 5.4 For $H(u,v) = n-1$, $d(u,v) \leq n+1$.

Result 5.5 $d(u,v) = n+2$ only if $H(u,v) = n-2$ and there are $2n-3$ faulty nodes distributed around one of u and v, say u, such that
i) $A(u) = \{j\}$, $j \in E(u,v)$ and
ii) $A(u_j) = \{j,k\}$, $k \in E(u,v) - \{j\}$.

Theorem 5.1 *For $n \geq 5$, if $|F(Q_n)| \leq 3n-6$, the distance between any two nodes u and v which are not isolated is at most $H(u,v)+6$.*

Proof : Let $H(u,v) = m$. Also, let $S(u,v) = \{x_1, x_2, \cdots x_m\}$ and $E(u,v) = \{e_1, e_2 \cdots e_{n-m}\}$. There can be two cases :
Case I : $C(u,v) \neq \phi$
Case II : $C(u,v) = \phi$

Case I : Here, we consider two subcases :

Subcase Ia : $C(u,v) \bigcap S(u,v) \neq \phi$: Let $i \in C(u,v) \bigcap S(u,v)$. We split Q_n along dimension i into two subcubes Q_{n-1} and Q'_{n-1} as shown in Fig. 5.1. Let $u \in Q_{n-1}$. Then $v \in Q'_{n-1}$. Also $u_i \in Q'_{n-1}$ and $v_i \in Q_{n-1}$.

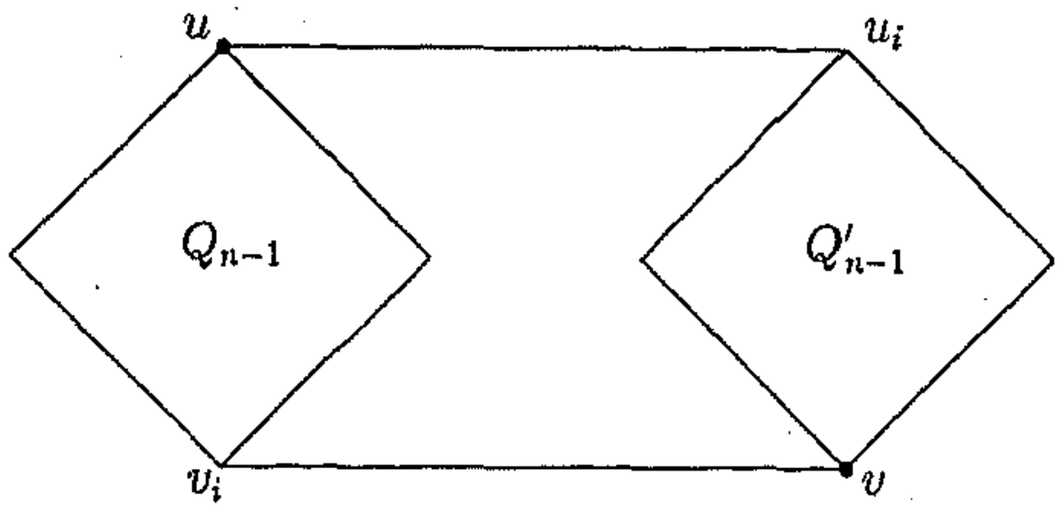


Figure 5.1: Q_n split along dimension i

A) If $|F(Q'_{n-1})| \leq n-2$, then $d(u_i,v) \leq H(u_i,v)+2$ (by Result 5.1). As $H(u_i,v) = H(u,v)-1$ and $d(u,v) \leq d(u_i,v)+1$, it follows that $d(u,v) \leq H(u,v)+2$

B) If $|F(Q'_{n-1})| > n-2$, then $|F(Q_{n-1})| \leq 2n-5 = 2(n-1)-3$.

i) If $|N(u)| \geq 2$ and $|N(v_i)| \geq 2$, then $d(u,v_i) \leq H(u,v_i)+4$, by Result 5.2. Thus in this case $d(u,v) \leq H(u,v)+4$.

ii) If $|N(u)| = 1$, then $|F(Q'_{n-1})| \leq 2n-5$ and by Result 5.2 $d(u_i,v) \leq H(u_i,v)+4$, provided $N(v) \geq 2$.

iii) If $|N(u)| = 1$ and $|N(v)| = 1$, then $n_b = 2n-2$. Let $S(u_i,v) = \{y_1, y_2, \cdots, y_{m-1}\}$.

Now we have the following sets of paths between u and v for $n > m$.

Set 1. $m-2$ paths among m paths of the form $(i \;\; \mathrm{Cir}(x_1 \;\; \bar{e}_1 \;\; x_2 \;\; \cdots x_{m-1} \;\; \bar{e}_1 \;\; x_m) \;\; i)$ for $m \geq 4$

Set 2. $n-m-1$ paths as $(i \;\; e_j \;\; y_1 \;\; i \;\; \cdots \;\; y_{m-1} \;\; e_j \;\; i)$ for $2 \leq j \leq n-m$

For $m \geq 4$, the total number of paths is equal to $n-3$ and when $m < 4$, we have $n-m$ paths as $(i \;\; e_j \;\; y_1 \;\; i \;\; \cdots \;\; y_{m-1} \;\; e_j \;\; i)$ for $1 \leq j \leq n-m$. Here, $d(u,v) \leq m+4$.

For $m = n$ we have the following set of paths between u and v.

Set 1. $m-1$ paths as $(i \;\; \mathrm{Cir}(y_1 \;\; y_2 \;\; \tilde{i} \;\; y_3 \cdots y_{m-2} \;\; y_{m-1}) \;\; i)$, for $m \geq 5$. Here,

$d(u,v) = n+2$

Considering all cases, $d(u,v) \leq H(u,v)+4$.

iv) If $|N(v_i)| = 1$, then $|F(Q'_{n-1})| \leq 2n-5$ and by Result 5.2 $d(u_i,v) \leq H(u_i,v)+4$, provided $N(u_i) \geq 2$.

v) If $N(v_i) = 1$ and $N(u_i) = 1$, then the path between u_i and v_i must be through u and v. By the same reasoning as in *iii)* $d(u_i,v_i) \leq H(u_i,v_i)+4$. Hence, $d(u,v) \leq H(u,v)+2$.

Subcase Ib : $C(u,v) \bigcap S(u,v) = \phi$: Let $i \in C(u,v)$. We split Q_n along the dimension i into two subcubes Q_{n-1} and Q'_{n-1} as shown in Fig. 5.2. Let $u \in Q_{n-1}$. Then $v \in Q_{n-1}$ and $u_i, v_i \in Q'_{n-1}$.

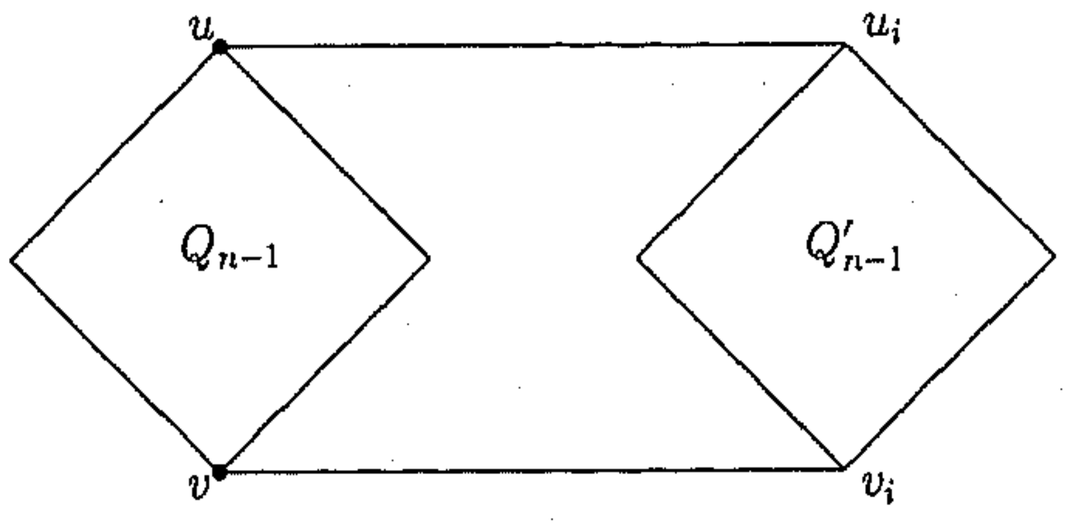


Figure 5.2: Q_n split along dimension i

If $|F(Q'_{n-1})| \leq n-2$, then $d(u_i,v_i) \leq H(u_i,v_i)+2$ by Result 5.1 and hence $d(u,v) \leq H(u,v)+4$.

If $|F(Q'_{n-1})| > n-2$, then $|F(Q_{n-1})| \leq 2n-5 = 2(n-1)-3$. Then by Result 5.3, $d(u,v) \leq H(u,v)+4$ provided $|N(u)| \geq 2$ and $|N(v)| \geq 2$.

When $|N(u)| = 1$ then $|F(Q'_{n-1})| \leq 2n-5$ and for $|N(v_i)| \geq 2$, we have $d(u_i,v_i) \leq H(u_i,v_i)+4$. Hence, $d(u,v) \leq H(u,v)+6$.

If $|N(v_i) = 1$ then for the pair (u,v_i) Subcase Ia is applicable and hence $d(u,v_i) \leq H(u,v_i)+4$. Since the path between u and v_i must be through the node v for this case, it follows that $d(u,v) \leq H(u,v)+4$

Case II : $C(u,v) = \phi$. Let $|A(u)| = p$ and $|A(v)| = q$. Let $B_1 = A(u) \bigcap S(u,v)$, $B_2 = A(u) \bigcap E(u,v)$, $B_3 = A(v) \bigcap S(u,v)$, and $B_4 = A(v) \bigcap E(u,v)$. Also, let p_1, p_2, q_1 and q_2 denote the cardinalities of the sets B_1, B_2, B_3 and B_4 respectively. Hence $p_1 + p_2 = p$, and $q_1 + q_2 = q$. Let $i \in A(u)$ and $j \in A(v)$. Then we consider the following cases :

i) At least one of u_{ij} or v_{ij}, say u_{ij} is fault-free.

Now $A(u_i) \bigcap A(v) \neq \phi$.
If $j \in S(u,v)$, then from the subcase Ia above, $d(u_i,v) \leq H(u_i,v) + 4$. Hence, if $i \in S(u,v)$, then $d(u,v) \leq H(u,v) + 4$ and if $i \notin S(u,v)$ then $d(u,v) \leq H(u,v) + 6$.

On the other hand, if $j \notin S(u,v)$, then by subcase Ib above, $d(u_i,v) \leq H(u_i,v) + 4$, provided $|N(v)| > 1$. If $|N(v)| = 1$, then $d(u_i,v) \leq H(u_i,v) + 6$.

Thus, when u_{ij} is fault-free, the only case for which it remains to be shown that $d(u,v) \leq H(u,v) + 6$ is when $|N(v)| = 1$, $A(v) = \{j\}$, $j \notin S(u,v)$ and if u_{ij} is fault-free for some $i \in A(u)$, then $i \notin S(u,v)$.
If $|N(u)| = 1$, then we proceed as follows :

Split Q_n along the dimension j into two subcubes Q_{n-1} and Q'_{n-1}. Let Q_{n-1} contain u and v. As $|N(u)| = 1$ and $|N(v)| = 1$, $|F(Q_{n-1})| \geq 2n - 4$ and hence $|F(Q'_{n-1})| \leq n - 2$. By Result 5.1, $d(u_{ij}, v_j) \leq H(u_{ij}, v_j) + 2$. Hence $d(u,v) \leq H(u,v) + 6$.

For $|N(u)| > 1$, consider the pair (v_j, u). If $A(v_j) \bigcap A(u) \neq \phi$, then by the same reasoning as in Case I, $d(u,v) \leq H(u,v) + 6$. On the other hand, if $A(v_j) \bigcap A(u) = \phi$, then we proceed as follows.

Let $|A(v_j)| = g$, and let $G_1 = A(v_j) \bigcap S(u,v)$ and $G_2 = A(v_j) \bigcap E(u,v)$. Also let g_1 and g_2 denote the cardinalities of the sets G_1 and G_2 respectively.

Now u has $n - p$ faulty neighbors. There are another p_1 faulty nodes of the form u_{ij}, for each $i \in B_1$. Hence, there are at least $n - p + p_1 = n - p_2$ faulty nodes within distance 2 from u. Also, there are $(n-1)+(n-g) = 2n-g-1$ faulty nodes around v and v_j. Hence we have already considered $3n - g - p_2 - 1$ faulty nodes.

Hence, the remaining number of faulty nodes is at most $g+p_2-5$. Between u and v_j, there are $(g-1)p_2$ paths as follows :

Set 1. $p_2(g_2-1)$ paths of the form $(i \;\; e_b \;\; j \;\; x_1 \;\; \cdots \;\; x_m \;\; i \;\; e_b)$, $e_b \in G_2 - \{j\}$, $i \in B_2$. Note that $B_2 \bigcap G_2 = \phi$ because $A(v_j) \bigcap A(u) = \phi$.

Set 2. $p_2 g_1$ paths among $p_2 m$ paths of the form $i \;\; \mathrm{Cir}(x_1 \;\; \bar{j} \;\; \bar{i} \;\; \cdots x_m)$, $i \in B_2$. Note that only g_1 nodes of the form v_{jk}, $k \in \{x_1, x_2 \cdots x_m\}$, are fault-free.

Since $(g-1)p_2 > g+p_2-5 = (g-1)+p_2-4$, we have $H(u,v) \leq H(u,v)+6$.

ii) All the nodes of the form u_{ij} and v_{ij}, $i \in A(u)$, $j \in A(v)$, are faulty. Now depending on $A(u)$ and $A(v)$ there can be the following cases.

A) There exists some i, j such that $i \in A(u)$, $j \in A(v)$ and $i, j \in S(u,v)$.

$H(u_i, v_j) = H(u,v)-2$. By splitting Q_n along dimensions i and j we get 4 subcubes of dimension $n-2$ (refer to Fig. 5.3). One of them, say Q_{n-2}, contains both u_i and v_j. Now there are at least $(n-p)+(n-q)+2pq = 2n-p-q+2pq$ faulty nodes within distance 2 from u and v. These include q faulty neighbors of u_i and p faulty neighbors of v_j. As v_{ij} and v_{ij} are outside Q_{n-2}, only $p+q-2$ of those $2n-p-q-2pq$ faulty nodes lie within Q_{n-2}. Hence, $|F(Q_{n-2})| \leq 3n-6-((2n-p-q+2pq)-(p+q-2)) = 3n-6-(2n-2p-2q+2pq+2) = n-2(pq-p-q+4)$. As $pq-p-q+4 = (p-1)(q-1)+3 \geq 3$, $|F(Q_{n-2})| < n-2$ and by Result 5.1, $d(u_i, v_j) \leq H(u_i, v_j)+2$, i.e., $d(u,v) \leq H(u,v)+2$.

B) There exists some i, j such that $i \in A(u) \bigcap S(u,v)$, $j \in A(v)$ where $A(v) \subseteq E(u,v)$.

Let $S(u_i, v) = \{z_1, z_2, \cdots, z_{m-1}\}$. We split Q_n along dimensions i and j (refer to Fig. 5.4). We can get $n-p-q-2$ paths between u_i and v_j, which do not pass through any of the assumed faulty nodes as follows :

Set 1. $n-m-(p_2+q)$ paths of the form $(e_k \;\; j \;\; z_1 \;\; \cdots \;\; z_{m-1} \;\; e_k)$ where $e_k \in E(u,v) - B_2 - A(v)$.

We choose e_k in this way because out of $n-m$ dimensions in $E(u,v)$, we exclude

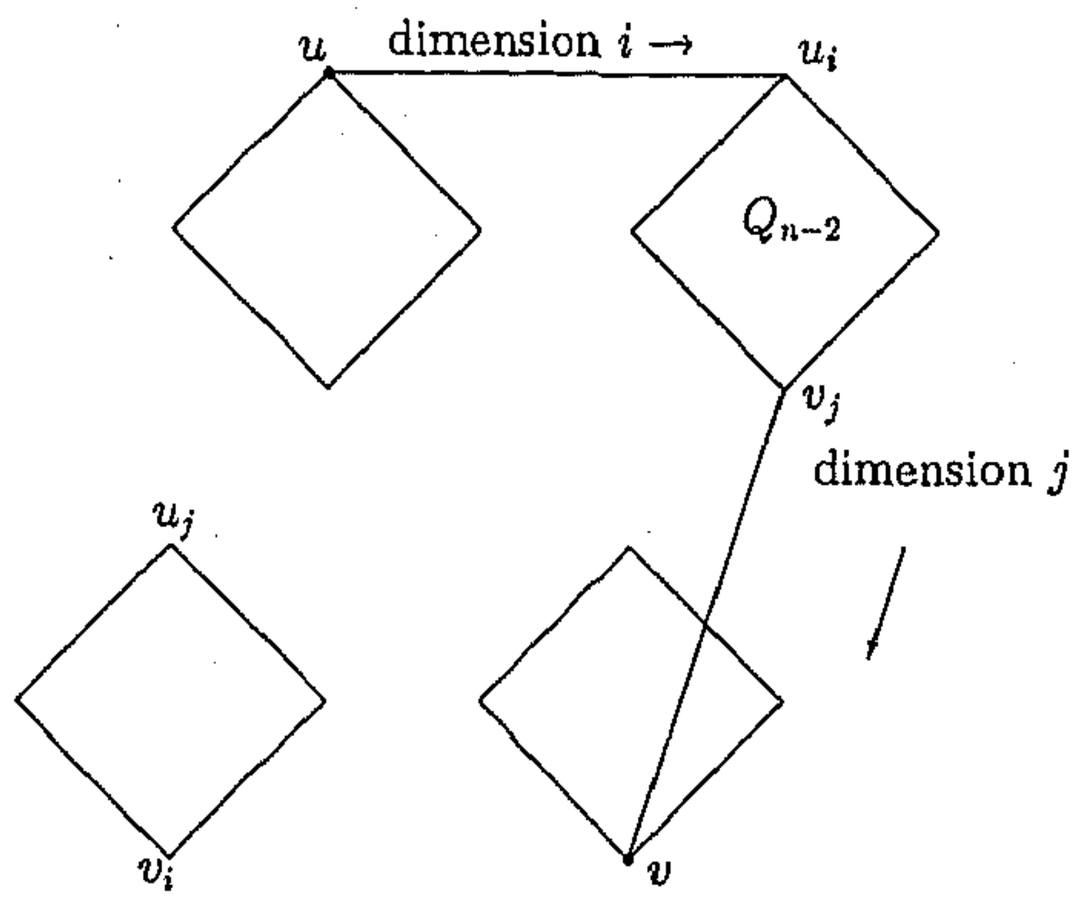


Figure 5.3: Q_n split along dimensions i and j

$A(v)$ to avoid the faulty nodes of the form u_{ij} and B_2 to avoid the faulty nodes of the form v_{ij}.

Set 2. $m-1-p_1$ paths among $(m-1)$ paths of the form $(\mathrm{Cir}(z_1\ \bar{j}\ \cdots\ z_{m-1}))$.

The total number of paths $= n-p-q-1 > 3n-6-(2n-p-q+2pq)$ as $2pq-2p-2q+5 > 0$. Hence, $d(u,v) \leq H(u,v)+4$.

C) There exists some i, j such that $i \in A(u)$ and $j \in A(v)$, where $A(u) \bigcap S(u,v) = \phi$ and $A(v) \bigcap S(u,v) = \phi$.

We show $n-p-q$ paths between u_i and v_j, which do not pass through any of the assumed faulty nodes, as follows :

Set 1. $n-m-(p+q)$ paths as $(e_k\ j\ \ x_1\ \ \cdots\ \ x_m\ \ i\ e_k)$, where $e_k \in E(u,v) - A(u) - A(v)$.

Set 2. m paths of the form $\mathrm{Cir}(x_1\ \bar{j}\ \cdots\ \ \bar{i}\ x_m)$.

But $n-p-q-2 > 3n-6-(2n-p-q+2pq)$. Hence, $d(u_i,v_j) \leq H(u_i,v_j)+2 \leq H(u,v)+4$.

Hence, the proof.

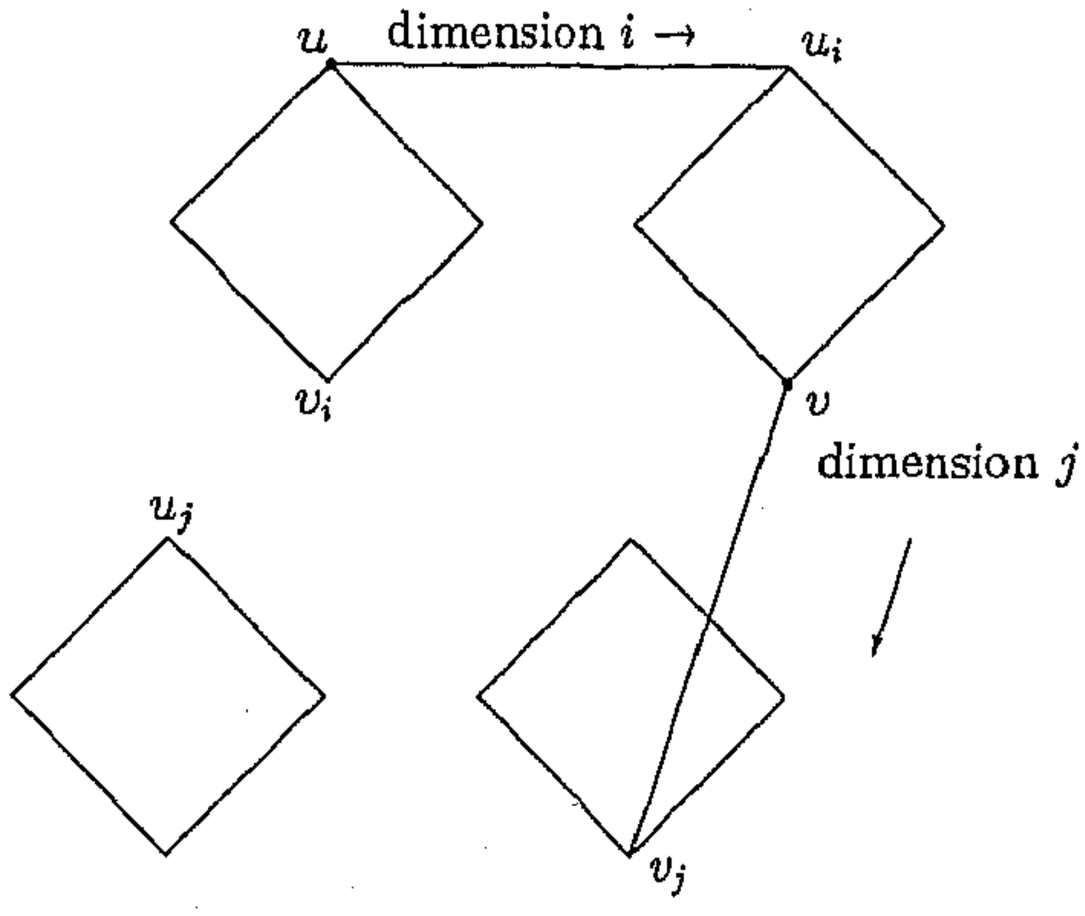


Figure 5.4: Q_n split along dimensions i and j

Remark 5.1 *It is noted that when* $C(u,v) \neq \phi$, $d(u,v)$ *can be equal to* $H(u,v)+6$ *only if one of* $|N(u)|$ *and* $|N(v)|$ *is equal to* 1 *and* $C(u,v) \subseteq E(u,v)$.

Corollary 5.1 *If* $|F(Q_n) \leq 3n-6$, *then for any two nodes* u *and* v *which are not isolated, if* $A(u) \bigcap S(u,v) \neq \phi$ *and* $A(v) \bigcap S(u,v) \neq \phi$, *then* $d(u,v) \leq H(u;v)+4$.

Proof : If $C(u,v) \neq \phi$, then by Remark 5.1 and the conditions given above, $d(u,v)$ cannot be equal to $H(u,v)+6$. Hence, we need to consider only the case when $C(u,v) = \phi$. Hence, $n_b \geq n$.

Let $i \in A(u) \bigcap S(u,v)$ and $j \in A(v) \bigcap S(u,v)$. We split Q_n along dimensions i and j into 4 subcubes of dimension $n-2$ each. Let one subcube Q_{n-2} contain u_i and v_j. Also, if u_{ij} or v_{ij} is fault-free, then by Theorem 5.1, $H(u,v) \leq d(u,v)+4$. If these two nodes are faulty, then $|F(Q_{n-2})| \leq 3n-6-(n+2) = 2n-8 < 2(n-2)-3$. Hence, $d(u_i,v_j) \leq H(u_i,v_j)+4$, provided $|N(u_i)| > 1$ and $|N(v_j)| > 1$. If, say $|N(u_i)| = 1$, then the path from u_i to v must be through u and in that case, as $d(u_i,v) \leq H(u_i,v)+6 = H(u,v)+5$, we must have $d(u,v) \leq H(u,v)+4$. Hence, the proof. □

5.4 Diameter of Q_n with at most $3n-6$ Faulty Nodes

Theorem 5.2 *If* $|F(Q_n)| \leq 3n-6$, *the following results hold for any two nodes* u *and* v *which are not isolated and at a hamming distance* n.
i) $d(u,v) = n+2$, *only if* $|N(u)| = |N(v)| = 1$ *and* $A(u) = A(v)$
ii) $d(u,v) = n$, *in all other cases.*

Proof : We proceed along the line followed in the proof of Theorem 5.1.

Case 1 : $C(u,v) \neq \phi$. Note that the situation like Subcase Ib in the proof of Theorem 5.1, cannot arise as $E(u,v) = \phi$. Moreover, in all the cases where we have applied Result 5.2, we can use Result 5.3 instead. Then we find that $d(u,v) = H(u,v) = n$ in all cases except when $|N(u)| = |N(v)| = 1$.

When $|N(u)| = |N(v)| = 1$, as shown Subcase Ia of the proof of Theorem 5.1, $d(u,v) = n+2$

Case 2 : $C(u,v) = \phi$. Let $i \in A(u)$. u_i and v are two diametrically opposite nodes in a subcube Q_{n-1}. If $|N(u)| = 1$, then $|F(Q_{n-1})| \leq 2n-5 = 2(n-1)-3$, and by Result 5.3 $d(u_i, v) = n-1$. Similarly for $|N(v)| = 1$ we can show that $d(u,v) = n$. Next, we consider the case when $|N(u)| > 1$ and $|N(v)| > 1$.

For $i \in A(u)$ and $j \in A(v)$ we consider two cases as in Theorem 5.1.

i) u_{ij} is fault-free. Let $j,k \in A(v)$. Consider as before a subcube Q_{n-1} which contains u_i and v (refer to Fig 5.5).

As $A(u) \bigcap A(v) = \phi$, u_j, u_k and v_i are faulty. All these three nodes lie outside Q_{n-1} and hence $|F(Q_{n-1})| \leq 3n-9 = 3(n-1)-6$. Since $C(u,v) = \phi$, for each dimension $l \in \{0,1,\cdots n-1\}$, either u_l or v_l or both must be faulty. Hence, $n_b \geq n$. To isolate the the pair (u_i, u_{ij}) from the remaining vertices in Q_{n-1}, we would require at least $2n-4$ more faults. Hence, u_i cannot be isolated in Q_{n-1}. To isolate the set of three nodes v_j, v and v_k from the remaining vertices in Q_{n-1} we would require

$3(n-1)-5$ faulty nodes in Q_{n-1}. Hence, the node v is not isolated in Q_{n-1}. Thus, for the pair (u_i, v) in Q_{n-1}, Case 1 above is applicable as $j \in C(u_i, v)$. Hence, $d(u_i, v) \leq n-1$.

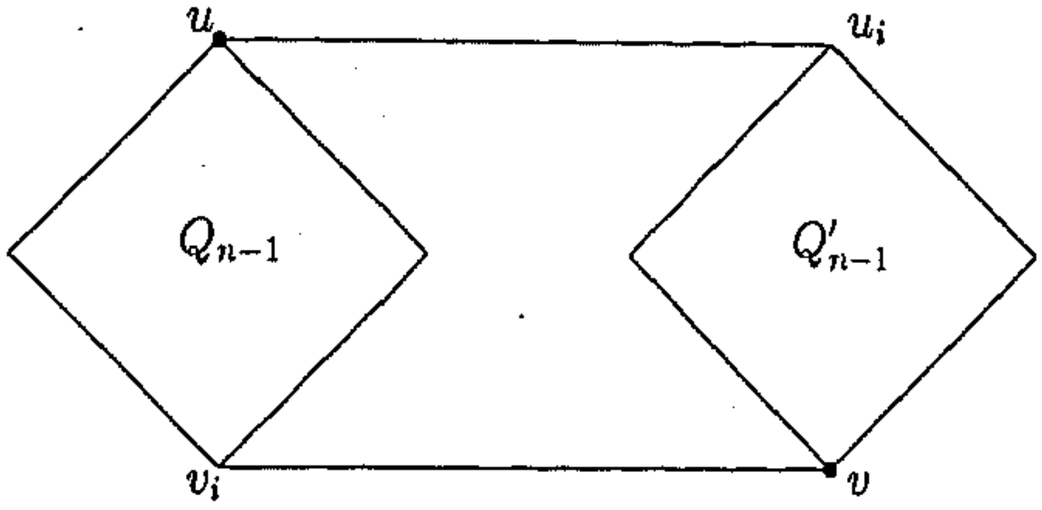


Figure 5.5: Q_n split along dimension i

ii) All the nodes of the form u_{ij} and v_{ij} are faulty for $i \in A(u)$ and $j \in A(v)$. As $E(u, v) = \phi$, only the corresponding Subcase A) needs to be considered. Hence, in Q_{n-2}, $d(u_i, v_j) = n-2$ by Result 5.1.

Hence the theorem. □

Theorem 5.3 *If* $|F(Q_n) \leq 3n-6$, *and* $(n > 4)$, *the following results hold for any two nodes* u *and* v *which are not isolated and at a Hamming distance of* $n-1$.
i) $d(u, v) \leq n+1$ *for* $n > 5$
ii) $d(u, v) \leq n+3$ *for* $n = 5$.

Proof : As $H(u, v) = n-1$, $E(u, v)$ must contain one and only one element. We can assume w.l.o.g (without loss of generality) that $E(u, v) = \{0\}$.

Now, if either of u_0 or v_0, say u_0 is fault-free, then $H(u_0, v) = n$ and Theorem 5.2 is applicable for the pair (u_0, v). Hence, $d(u_0, v) = n$ or $n+2$. If $d(u_0, v) = n$, then $d(u, v) \leq n+1$. But $d(u_0, v) = n+2$, only if $|N(u_0)| = 1$. In that case the path from u_0 to v must pass through u. Hence $d(u, v) \leq n+1$.

Next, we consider the case when both u_0 and v_0 are faulty. We proceed as in the proof of Theorem 5.1.

Case 1 : $A(u) \bigcap A(v) \neq \phi$. This case corresponds to Case I of Theorem 5.1. Let $i \in A(u) \bigcap A(v)$. Hence, $H(u_i, v) = H(u, v_i) = n - 2$. Now, if we apply Result 5.4 whenever we have applied Result 5.2 in the proof of Theorem 5.1, we will get $d(u, v) \leq H(u, v) + 2$ except when $|N(u)| = |N(v)| = 1$. When $|N(u)| = |N(v)| = 1$, we have shown that $d(u, v) \leq H(u, v) + 4$. But for $n > 5$, we can avail of $n - 2$ paths between u and v as $(i \quad \mathrm{Cir}(y_1 \quad y_2 \quad \bar{i} \quad \cdots \quad y_{n-2}) \quad i)$ where $S(u_i, v) = \{y_1 \quad y_2 \quad \cdots \quad y_{n-2}\}$. Hence, for $n > 5$, $d(u, v) \leq n + 1$.

Case 2 : $A(u) \bigcap A(v) = \phi$. This situation is similar to Case II of Theorem 5.1

i) Let there exist some $i \in A(u)$ and $j \in A(v)$ such that u_{ij} is fault-free. Consider a subcube Q_{n-1}, which contains both u_i and v. Note that u_0, u_j and v_i are faulty nodes. All these three nodes lie outside Q_{n-1}. Hence, $|F(Q_{n-1})| \leq 3(n-1) - 6$. As $j \in A(u_i) \bigcap A(v)$, for the pair (u_i, v), we can apply Case 1 above, resulting in $d(u_i, v) \leq H(u_i, v) + 2 = n - 2 + 2 = n$.

ii) Let all the nodes of the form u_{ij} and v_{ij} be faulty for $i \in A(u)$ and $j \in A(v)$. In this case, only the Subcase A) as given in the corresponding part of Theorem 5.1 needs to be considered. Using the results thereof, we get $d(u, v) \leq H(u, v) + 2$.

Hence, the theorem. □

Theorem 5.4 *If* $|F(Q_n) \leq 3n - 6$, *and* $(n > 4)$, *then for any two nodes* u *and* v *which are not isolated and at a Hamming distance of* $n-2$, $d(u, v) \leq n + 2$.

Proof : If $A(u) \bigcap E(u, v) \neq \phi$ or $A(u) \bigcap E(u, v) \neq \phi$, then let $i \in A(u) \bigcap E(u, v)$. Now, $H(u_i, v) = n-1$ and by Theorem 5.3, $d(u_i, v) \leq n+1$. Hence, $d(u, v) \leq n+2$.

On the other hand, if $A(u) \subseteq S(u, v)$ and $A(v) \subseteq S(u, v)$, then by Corollary 5.1 $d(u, v) \leq H(u, v) + 4 = n + 2$.

Hence the theorem. □

Theorem 5.5 *With the assumed forbidden faulty set model, if* $|F(Q_n)| \leq 3n-6$, *then the diameter of* Q_n *is atmost* $n+3$.

Proof : From Theorems 5.2 to 5.4, it follows that $d(u,v) \leq n+3$ for $H(u,v) \geq n-2$. Again, by Theorem 5.1, $d(u,v) \leq H(u,v)+6$. Hence, for $H(u,v) \leq n-3$, $d(u,v) \leq n+3$. Hence the proof. □

Theorem 5.6 *For any two nodes* u *and* v *in* Q_n, $(n \geq 6)$ *such that* $H(u,v) = n-3$, *the distance between them can be equal to* $n+3$ *only if all the* $3n-6$ *faulty nodes are distributed around one of* u *and* v, *say* u, *such that*

i) $A(u) = \{j\}$, $j \in E(u,v)$;

ii) $A(u_j) = \{j,k\}$, $k \in E(u,v) - \{j\}$;

iii) $A(u_{jk}) = \{k,l\}$, $l \in E(u,v) - \{j,k\}$

Proof : Consider a pair (u,v) such that $H(u,v) = n-3$, i.e. $|E(u,v)| = 3$. We can assume w.l.o.g. that $E(u,v) = \{0,1,2\}$. Also, by Corollary 5.1, $d(u,v) \leq H(u,v)+4$ if $A(u) \bigcap S(u,v) \neq \phi$ and $A(v) \bigcap S(u,v) \neq \phi$. Hence, if $d(u,v)$ has to be equal to $n+3$, then at least one of $A(u)$ and $A(v)$ must be a subset of $E(u,v)$. W.l.o.g., let $A(u) \subseteq E(u,v)$, i.e., $N(u)$ can be at most equal to 3.

The following cases can arise :

Case I : $A(u) \bigcap A(v) = \phi$.

Case II : $A(u) \bigcap A(v) \neq \phi$.

Case I : $A(u) \bigcap A(v) = \phi$. Here, $n_b \geq n$. Let the remaining number of faulty nodes be denoted by n_r. Hence, $n_r \leq 2n-6$. We consider the following subcases :

Subcase Ia) $A(v) \subseteq S(u,v)$.

i) If $|N(v)| = 1$, then let $A(v) = \{i\}$, $i \in S(u,v)$. We now split Q_n along dimension i into two subcubes Q_{n-1} and Q'_{n-1}. Let $v \in Q_{n-1}$. Then $v_i, u \in Q'_{n-1}$, and $|F(Q'_{n-1})| \leq 2n-5$. By Result 5.2, $d(u,v_i) \leq n$ and hence, $d(u,v) \leq n+1$.

ii) If $|N(v)| \geq 2$, v will have at least two fault-free neighbors say v_3 and v_4.

A) When $|N(u)| = 3$, we can establish the following sets of paths from v to u :

Set 1. $n-4$ paths as (3 Cir(4 $\bar{0}$ 5 $\cdots$ $n-1$) 0)

Set 2. $n-5$ paths as (4 Cir(5 $\bar{1}$ $\bar{3}$ 6 $\cdots$ $n-1$) 1)

These paths are valid only for $n > 6$ because of the following reasons. $A(u) \subseteq E(u,v) = \{0,1,2\}$, Hence, u_3, u_4 and u_5 are all faulty, and so for $n = 6$, the above paths through u_3, u_4 and u_5 are all inaccessible.

Set 3. 1 path as (3 1 0 4 $\cdots$ $n-1$ 1 0)

Set 4. 1 path as (4 1 2 5 $\cdots$ $n-1$ 3 2 1)

Set 5. 1 path as (3 2 4 $\cdots$ $n-1$ 2)

Set 6. 1 path as (4 2 5 $\cdots$ $n-1$ 3 2)

Hence, for $n > 6$ we have a total of $2n-5$ paths, with $d(u,v) \leq n+1$.

Now we consider the case for $n = 6$, for which $n_r \leq 6$. If $|N(v)| = 3$, then we have 7 paths as follows :

Path 1. (3 0 4 5 0)

Path 2. (3 1 4 5 1)

Path 3. (3 2 4 5 2)

Path 4. (4 0 5 3 0)

Path 5. (4 1 5 3 1)

Path 6. (4 2 5 3 2)

Path 7. (5 0 1 3 4 1 0)

If $|N(v)| = 2$, then $n_r \leq 5$ and hence we will discard the path of the type 'path 7'above. Thus, $d(u,v) \leq 7$.

B) $|N(u)| = 2$. W.l.o.g. we can assume $A(u) = \{0,1\}$. Consider a subcube Q_{n-2} which has u_0 and v as diametrically opposite nodes. When $Q_n - Q_{n-2}$ contains only two faulty nodes other than u_2, v_1 and v_2, then we can always have a path from u to v out of the following 4 paths :

Set 1. 2 paths as (0 j 4 $\cdots$ $n-1$ 0 j 3), $j \in \{1,2\}$.

Set 2. 2 paths as (1 j 3 5 $\cdots$ $n-1$ 0 j 4), $j \in \{1,2\}$.

If $Q_n - Q_{n-2}$ contains more than two faulty nodes in addition to u_2, v_1 and v_2, then $|F(Q_{n-2})| \leq 3(n-2)-6$. Hence, $d(u_0, v) \leq n-2$ (Theorem 5.2), provided u_0 is not isolated within Q_{n-2}. If u_0 and u taken together were isolated in Q_{n-2}, all the $2(n-3)$ nodes around them in Q_{n-2} must have been faulty. Apart from these, there will be 4 more faulty nodes, viz., u_2, v_0, v_1 and v_2. Then we can establish a path from u to v out of the $n-3$ paths of length $n-1$ between u and v as $(1\ \mathrm{Cir}(3\ \cdots\ n-2\ \bar{1}\ n-1))$.

C) $|N(u)| = 1$. W.l.o.g. we can assume $A(u) = \{0\}$. Consider a subcube Q_{n-2} as in Case B) above. The faulty nodes u_1, u_2, v_1, and v_2 are outside Q_{n-2}. If $Q_n - Q_{n-2}$ contains only one more faulty node, then we can establish a path from u to v out of the following two possibilities :

$(0\ j\ 4\ \cdots\ n-1\ 0\ j\ 3),\ j \in \{1,2\}$.

If $Q_n - Q_{n-2}$ contains more than one faulty node in addition to u_1, u_2, v_1 and v_2, then $|F(Q_{n-2})| \leq 3(n-2)-6$. Hence, $d(u_0, v) \leq n-2$ (Theorem 5.2), provided u_0 is not isolated within Q_{n-2}. If u_0 and u taken together were isolated in Q_{n-2}, all the $2(n-3)$ nodes around them in Q_{n-2} must have been faulty. Apart from these, there will be 5 more faulty nodes viz. u_1, u_2, v_0, v_1, and v_2. Also, either of u_{01} or u_{02}, say u_{01} must be fault-free for the existence of a path between u and v. Then we can establish a path from u to v out of $n-3$ paths given by $(0\ 1\ \mathrm{Cir}(3\ \cdots\ n-2\ \bar{0}\ \bar{1}\ n-1))$. In this case, $d(u,v) \leq n+1$.

Subcase Ib) $A(v) \bigcap E(u,v) \neq \phi$.

Let $i \in A(u)$ and $j \in A(v) \bigcap E(u,v)$. Then $i \neq j$ because of the initial assumption. W.l.o.g., we can assume that $i = 0$ and $j = 1$. If u_2 is fault-free we can establish $2n-5$ paths from u to v as follows :

Set 1. $n-3$ paths as $(0\ \mathrm{Cir}(3\ \cdots\ \bar{0}\ n-1))$

Set 2. $n-3$ paths as $(2\ \mathrm{Cir}(3\ \bar{1}\ \cdots\ \bar{2}\ n-1)\ 1)$

Set 4. 1 path as $(0\ 1\ 3\ \cdots\ (n-1)\ 0\ 1)$.

Since $n_r \leq 2n-6$, there always exists a path from u to v with $d(u,v) \leq n+1$.

Next we consider the case when u_2 is faulty. As the dimension 1 is in $A(v)$ and $A(u) \bigcap A(v) = \phi$, u_1 must be faulty. Hence, $|N(u)| = 1$.

We split Q_n along dimension 2 into two subcubes Q_{n-1} and Q'_{n-1}. u, v, u_0 and v_1 all will belong to one of these subcubes. Let this subcube be Q_{n-1}.

If $|F(Q_{n-1})| \leq 3(n-1) - 6$, then $d(u_0, v_1) \leq n-1$, provided u_0 and v_1 are not isolated in Q_{n-1}. But to isolate v_1 in Q_{n-1}, we need $2n-4$ faulty nodes around v and v_1. But there are already $n-1$ faults around u and hence, the node v_1 cannot be isolated in Q'_{n-1}. If u_0 and u taken together are isolated in Q_{n-1}, then u_{02} must be fault-free for the existence of a path between u and v. In this case, there are already $(2n-3)$ (about u and u_0) $+ 1$ (the node v_0) $= 2n-2$ faults. Now, we can establish a path between u and v out of the $n-3$ paths given by $(0\ \ 2\ \ \mathrm{Cir}(3\ \ \cdots\ n-2\ \ \bar{0}\ \ \bar{2}\ \ n-1))$. Hence, $d(u, v) \leq n+1$.

If on the other hand, $|F(Q_{n-1})| > 3(n-1) - 6$, then $|F(Q'_{n-1})| \leq 2$. Now we consider the situations corresponding to i) v_2 is faulty and ii) v_2 is fault-free.

i) v_2 is faulty. Two faulty nodes in Q'_{n-1} are u_2 and v_2. If $|N(v)| = 1$, i.e., only fault-free neighbor of v is v_1, then $n_b = 2n-2$ and we can establish $n-3$ paths from u to v as $(0\ \ \mathrm{Cir}(3\ \ \bar{1}\ \ 4\ \ \cdots\ \ n-2\ \ \bar{0}\ \ n-1)\ \ 1)$. Hence, $d(u, v) \leq n+1$. If $|N(v)| > 1$, then we must have some $k \in A(v) \bigcap S(u, v)$. Now, u_{02} and v_{k2} are both in Q'_{n-1} in which there can be at most 2 faults. Hence, we have a path of length $n-3$ between u_{02} and v_{k2} in Q'_{n-1}. Thus, $d(u, v) \leq n+1$.

ii) v_2 is fault-free. If u_{02} is fault-free, then we have a path of length $n-2$ between u_{02} and v_2 in Q'_{n-1}. Hence, $d(u, v) \leq n+1$. If u_{02} is faulty, then two faulty nodes in Q'_{n-1} are u_2 and u_{02}. Now, if for some $k > 2$, $k \in A(u_0)$, then there is a path of length $n-3$ between u_{0i2} and v_2 in Q'_{n-1}. Otherwise there will be a total of $2n-3$ faults around u and u_0 and at least one faulty node (v_0) around v. Also the node u_{01} must be fault-free for the existence of a path between u and v. Then we have $n-3$ paths from u to v as $(0\ \ 1\ \ \mathrm{Cir}(3\ \ \cdots\ \ \bar{0}\ \ n-1)\ \ 1)$

Considering all the cases, $d(u, v) \leq n+1$.

Case II : $A(u) \bigcap A(v) \neq \phi$. W.l.o.g., we can assume that $0 \in A(u) \bigcap A(v)$.

Split Q_n along dimension 0 into two subcubes Q_{n-1} and Q'_{n-1}. Also let $u, v \in Q_{n-1}$. Then $u_0, v_0 \in Q'_{n-1}$. In this case $d(u,v)$ can be equal to $n+3 = H(u,v)+6$ only if either $|N(u)| = 1$ or $|N(v)| = 1$ (by Remark 5.1).

If $|N(v)| = 1$, then $n_b \geq 2n-4$ with at least $n-3$ faults around u and $n-1$ faults around v. Hence, $|F(Q'_{n-1})| \leq n-2$ and $d(u_0, v_0) \leq H(u_0, v_0)+2$ (by Result 5.1). Hence, $d(u,v) \leq n+1$. Thus, for $d(u,v)$ to be equal to $n+3$ we must have $|N(u)| = 1$.

When $|N(u)| = 1$, $|F(Q'_{n-1})| \leq 2n-5$. We now consider the following subcases :

Subcase 1 : $|N(v_0)| = 1$. Total number of faults around u and v_0 is equal to $2n-2$. We can establish $n-3$ paths from u and v as (0 Cir(3 4 0 5 $\cdots$ $n-1$))
Hence, $d(u,v) \leq n-1$.

Subcase 2 : $|N(v_0)| > 1$. u_0 must have at least one fault-free neighbor in Q'_{n-1}. For $d(u,v)$ to be equal to $n+3$, we must have $d(u_0, v_0) \geq n+1$. Since $H(u_0, v_0) = n-3 = (n-1)-2$, by Result 5.2, $d(u_0, v_0) = H(u_0, v_0)+4 = n+1$. Also, by Result 5.5, $d(u_0, v_0) = n+1 = (n-1)+2$, only if

i) $A(u_0) = \{0, j\}$, $j \in \{1,2\}$ and $A(u_{0j}) = \{j, k\}$, $k \in \{1,2\} - \{j\}$,
or ii) $A(v_0) = \{0, j\}$, $j \in \{1,2\}$ and $A(v_{0j}) = \{j, k\}$, $k \in \{1,2\} - \{j\}$.

If the situation like ii) occurs, we can show that $d(u,v) \leq n+1$ as follows. There are $n-1$ faulty nodes around u, and the other $2n-5$ faulty nodes are distributed as above. Now we can establish a path between u and v as (0 3 1 0 4 $\cdots$ $n-1$ 1) which is of length $n+1$ and does not pass through any of the faulty nodes. But if the situation like i) occurs, then the path between u and v must pass through the node u_{012}. Then $d(u,v) = n+3$.

We have considered all the possibilities and found that the only situation when $d(u,v) = n+3$ is as given above. □

Corollary 5.2 *All n-cubes with $3n-6$ faulty nodes and the corresponding diameter $D_{3n-6} = n+3$ are isomorphic for $n > 5$.*

Proof : Follows from Theorem 5.6. □

5.5 Algorithm for Routing

An algorithm for routing in a faulty hypercube Q_n, with fault set F has been developed in [TR93]. There, $|F| \leq 2n-3$ and the time complexity of the algorithm is $O(|F| \log n)$. For routing in Q_n with $|F| \leq 3n-6$ we use a function *Route* from [TR93] as described below :

Route (u, v, F, Q_k, M) : This function returns a path between u and v which lies within a k-subcube Q_k and is of length at most $H(u,v)+4$, provided $|F| \leq 2k-3$, M being an n-bit vector which contains 1's at the bit positions corresponding to the dimensions which will not be used in the path. If $|F| > 2k-3$ and this procedure fails then it returns a **null** path.

Before describing the algorithm for routing with at most $3n-6$ faulty nodes, we further describe a function **Access** (u) and a procedure **Findpath**, which are based on the ideas of [TR93]. In what follows we represent bitwise AND, OR and Exclusive-Or operations by $\wedge$, $\vee$ and $\oplus$ respectively. We assume that the weight of a binary number consisting of n digits can be performed in $\log n$ time.

function Access(u); /* returns a vector which has its i^{th} bit set to 1, only if u_i is fault-free. */
begin
a $\leftarrow$ 0; /* a is an n bit vector */
for each $f \in F$ **do**
begin
$\delta \leftarrow f \oplus u$;
if $weight(\delta) = 1$ **then** a $\leftarrow$ a $\vee \delta$

```
        end;
        return(ā)
end;
```

The time complexity of the function **Access** is $|F| \log n$ [TR93].

The procedure **Findpath** will be used to find a fault-free path among a set of m paths of the form $\mathrm{Cir}(x_1 \ \ x_2 \ \ \cdots \ \ x_m)$ following the ideas of [TR93] as follows : Let $S(u,v) = \{x_1, x_2, \cdots, x_m\}$, where $x_1 < x_2 < \cdots < x_m$. Note that $\mathrm{Cir}(x_1 \ \ x_2 \ \ \cdots \ \ x_m)$ is the set of paths obtained by rotating $(x_1 \cdots x_m)$ cyclically m times. The path $(x_r \cdots x_m x_1 \cdots x_{r-1})$, $r \leq m$, is described by the nodes which differ from u successively in the bit positions corresponding to x_r, x_{r+1}, $\cdots$, x_m, x_1, $\cdots$, x_{r-1}. Therefore, to check whether a faulty node $f \in F$ blocks one of the m paths, we examine if f differs from u in the bit positions which form a subset of $\{x_1, x_2, \cdots, x_m\}$ and which are taken from the sequence $< x_1, x_2, \cdots, x_m >$ in a cyclic order (i.e., wrap around fashion). The procedure **Findpath** performs this check as explained below.

Let $\alpha = u \oplus v$ and $Q_m(u,v)$ denote the m-subcube spanning the dimensions x_1, x_2, $\cdots$, x_m in which u and v are two diametrically opposite nodes. For a given faulty node $f \in F$, we define $\beta = f \oplus u$. f can block one of the m paths only if $f \in Q_m(u,v)$, which implies $\alpha \vee \beta = \alpha$. If β has a single block of successive 1's of $S(u,v)$ in cyclic order, only then it blocks a path. We use an array of integers $Avail[0..n-1]$ to indicate whether a path belonging to $\mathrm{Cir}(x_1 \ \ x_2 \ \ \cdots \ \ x_m)$ is blocked or not. If a path starting with the dimension i is blocked then we set $Avail[i]$ to 0. Initially $Avail[i]$ is set to 1 for all i. β is checked as follows. First, construct an array $Index\ [0..n-1]$, where $Index\ [i] = 0$, if $i \notin S(u,v)$ and k if $i = x_k \in S(u,v)$. Find the positions of the rightmost 1-bit and the leftmost 1-bit of β and let these be r and l respectively. If $weight(\beta) = Index[r] - Index[l] + 1$, then β is a straight block of successive 1's starting from the bit position l without any wrap-around from x_m to x_1. We then set $Avail[l]$ to 0. Otherwise, complement the bits of β in all positions of $S(u,v)$ to get β', and repeat the above procedure with β' instead of β to check if β is a wrap-around block. If β' is found to have a

single block of successive 1's of $S(u,v)$, with the position of the rightmost 1 equal to r' and if $r' = x_i$ (say), then set $Avail[x_{i+1}]$ to 0.

ProcedureFindpath(u, v, f);
begin
$\beta \leftarrow u \oplus f$; $\alpha \leftarrow u \oplus v$;
if $\beta \vee \alpha = \alpha$ **then** /* $f \in Q_m(u,v)$ */
begin
$l \leftarrow lmb(\beta)$; $r \leftarrow rmb(\beta)$;
/* lmb and rmb return the leftmost and rightmost 1-bits of β respectively /*
if $weight(\beta) =$ Index[r] − Index[l] + 1 **then** $Avail[l] \leftarrow 0$; /* a straight block */
else
begin
$\beta' \leftarrow \beta \oplus \alpha$; /* Complementing the $S(u,v)$ bits of β */
$l' \leftarrow lmb(\beta)$; $r' = rmb(\beta)$;
if $weight(\beta')$ = Index[r'] − Index[l'] + 1 **then**
begin
find $x_i \in S(u,v)$ such that $x_i = r'$;
$Avail[x_{i+1}] \leftarrow 0$;
end;
end;
end;
end;

The running time of the function **Findpath** (u, v, f) is $O(\log n)$. Applying **Findpath** for each $f \in Q_m(u,v)$, if we find that for some $i \in S(u,v)$, $Avail[i] = 1$, then the path starting with that dimension i is not blocked.

Example 5.1 Consider $\alpha = 0100110101$ and $\beta = 0100000101$. Here, $x_1 = 1$, $x_2 = 4$, $x_3 = 5$, $x_4 = 7$ and $x_5 = 9$. The successive element of the array *Index* will be initialized with 0, 1, 0, 0, 2, 3, 0, 4, 0, and 5 respectively. Also, $r = 9$ and $l = 2$. Index[9] − Index[2] + 1 = 5 − 2 + 1 = 4 $\neq$ $weight\beta = 3$ and hence,

β is not a straight block. Now $\beta' = 0000110000$. Here, $r' = 5$, $l' = 4$ and Index[5] − Index[4] + 1 = 2 = $weight\beta'$. Since $r' = 5 = x_3$, the path $x_4\ x_5\ \cdots$ is blocked and hence only $Avail[7]$ is to be set to 0.

The procedure **Findpath** will also be used to find a fault-free path between u and v among the set of paths $\mathrm{Cir}(x_1\ \tilde{e}_1\ x_2\ \cdots\ x_{m-1}\ \bar{e}_1\ x_m)$, where $S(u,v) = \{x_1, x_2, \cdots, x_m\}$ and $e_1 \in E(u,v)$. To check if one of these paths are blocked we proceed as follows :

if f is a neighbor of u or v **then Findpath**(u,v,f) else **Findpath**(u_{e_1}, v_{e_1}, f).

Sometimes we need to find the number of faulty nodes contained in a given m-subcube Q_m spanned along dimensions $S(u,v)$. For this we need to check only the non-$S(u,v)$ bits of the faulty nodes. Therefore, $\forall f \in F$, we try to match the non-$S(u,v)$ bits of f with the non-$S(u,v)$ bits of Q_m. This can be done in $O(|F| \log |F|)$ time. Let $E(u,v) = \{e_1, e_2, \cdots, e_{n-m}\}$, where $e_1 < e_2 < \cdots < e_{n-m}$. Consider a subcube Q_m^k which is the image of $Q_m(u,v)$ along dimension e_k, $k \geq 1$. We use a procedure **Findnumbers**$(Q_m(u,v), E(u,v) - \{e_1\})$ to find out the number of faulty nodes in each of the subcubes Q_m^k for $k = 2, 3, \cdots, n-m$.

When $|A(u)| = |A(v)| = \{i\}$, $i \in S(u,v)$, we use a function **Route1** (u, v, f, i) which returns a path between u and v.

Function Route1(u, v, f, i);
begin
    Findnumbers$(Q_m(u,v), E(u,v) - \{e_1\})$;
    Set $\{y_1, y_2, \cdots, y_{m-1}\} = S(u,v) - \{i\}$;
    for $k = 2$ to $n-m$ **do if** $|F(Q_m^k)| = 2$ **then** return $(i\ e_k\ y_1\ i\ y_2\ \cdots y_{m-1}\ e_k\ i)$;
    if $(m < 4)$ **then** return $(i\ e_1\ y_1\ i\ y_2\ \cdots y_{m-1}\ e_1\ i)$;
    $\forall i$ set $Avail[i]$ to 1;
    for each $f \in F$ **do**
    begin
        if f is a neighbor of u_i or v_i **then Findpath** (u_i, v_i, f)
            else Findpath(u_{ic_1}, v_{ic_1}, f);

```
        end;
        return a path using the array Avail;
end;
```

In some cases we have to check if there exists any fault-free node of the form u_{ij} or v_{ij} for $i \in A(u)$ and $j \in A(v)$. For this, we describe below a procedure **Nextnode** which outputs an array of integers $Count[0..n-1]$ so that $Count[i]$ gives the number of faulty neighbors of u_i along all possible dimensions j, $j \in A(v)$. It is assumed that $\mathbf{a} = \mathbf{Access}(u)$ and $\mathbf{b} = \mathbf{Access}(\mathrm{v})$;

```
Procedure Nextnode(u, F, Count);
begin
    Set Count[i] to 0 ∀i;
    for each f ∈ F do
    begin
        δ ← f ⊕ u;
        if weight(δ) = 2, then
        begin
            if (weight(δ ∧ a) = 1 and weight(δ ∧ b) = 1) then
            begin
                i = lmb(δ ∧ a);
                Count[i] = Count[i] + 1;
            end;
        end;
    end;
end;
```

Now it follows that if for a given $i \in A(u)$, u_{ij} is faulty for all $j \in A(v)$, then $Count[i]$ will be equal to $weight(\mathbf{b})$; Otherwise, Count[i] < $weight(\mathbf{b})$. The time complexity of the procedure **Nextnode** is $O(|F| \log n)$.

We now describe the routing algorithm which closely follows the proof of Theorem 5.1. The required path is expressed in the form of a string of dimensions. The

operator '$\circ$' is used to concatenate two strings of dimensions.

Algorithm R (u, v, F) ;

Step 1 : $\mathbf{a} \leftarrow$ Access(u); $\mathbf{b} \leftarrow$ Access(v); $\mathbf{c} = \mathbf{a} \wedge \mathbf{b}$;
if weight$(\mathbf{c}) > 0$ **then** go to Step 2
else go to Step 3;

Step 2 : $\mathbf{d} \leftarrow \mathbf{c} \wedge (u \oplus v)$;
if $\mathbf{d} \neq 0$, **then** go to Step 2a **else** go to Step 2b;

Step 2a : $M \leftarrow 0$; /* M is an n-bit vector */
set i-th bit of M to 1;
$i \leftarrow lmb(d)$. /* $i \in C(u,v) \bigcap S(u,v)$ */
$P \leftarrow Route(u, v_i, F \bigcap Q_{n-1}, Q_{n-1}, M)$; /* P is a path */
/* Q_{n-1} is the subcube containing u and v_i */
if $P \neq$ null then return$(P \circ i)$;
else
begin
$P \leftarrow Route(u_i, v, F \bigcap Q'_{n-1}, Q'_{n-1}, M)$;
/* Q'_{n-1} is the subcube containing u_i and v */
if $P \neq$ null **then** return$(i \circ P)$
end
else /* either $N(u) = N(v) = 1$ or $N(u_i) = N(v_i) = 1$. */
if $weight(\mathbf{a}) = 1$ **then** /* $N(u) = N(v) = 1$ */
Route1(u, v, f, i)
else /* $N(u_i) = N(v_i) = 1$ */
begin
$P \leftarrow$ **Route1**(u_i, v_i, f, i);
return a path obtained by removing i from the leftmost and rightmost end of P;
end;

Step 2b : Techniques similar to Step 2a are applied.

Step 3 : **Nextnode**$(u, F, Count1)$; **Nextnode**$(v, F, Count2)$;

if for some $i \in A(u)$, $Count1[i] < weight(\mathrm{b})$ **then**

go to Step 1 with $u = u_i$;

if for some $j \in A(v)$, $Count[j] < weight(\mathrm{a})$ **then**

go to Step1 with $v = v_j$;

/* If no such i or j can be obtained, we have to consider 3 subcases as in Theorem 5.1. */

$p_1 \leftarrow \mathrm{a} \wedge \alpha$; $q_1 \leftarrow \mathrm{b} \wedge \alpha$;

if $p_1 \neq 0$ and $q_1 \neq 0$ **then** go to Step 3a

else if $p_1 \neq 0$ and $q_1 = 0$ **then** go to Step 3b

else go to Step 3c;

Step 3a : $i \leftarrow lmb(p_1)$; $j \leftarrow lmb(q_1)$; $M \leftarrow 0$;

set i^{th} and j^{th} bit of M to 1;

$P \leftarrow$ **Route**$(u_i, v_j, F \bigcap Q_{n-2}, M)$

/* Q_{n-2} spans all the dimensions except i and j and contains u_i and v_j. */

return $(i \circ P \circ j)$;

Step 3b : $i \leftarrow lmb(p_1)$; $j \leftarrow lmb(\mathrm{b})$;

Findnumbers$(Q_m(u_i, v_j), E(u,v) - B_2 - A(v))$;

for each $e_k \in E(u,v) - p_2 - A(v)$ do

begin

$Q'_m \leftarrow$ the image of $Q_m(u_i, v_j)$ along dimension e_k

if $|F(Q'_m)| = 1$ **then** /* v_{e_k} is the only faulty node in Q'_m */

begin

$P \leftarrow$ a path in Q'_m between u_{ie_k} and v_{je_k};

return $(i \ e_k \circ P \circ e_k \ j)$;

end;

end;

for $i = 1$ to $n-1$ do $Avail[i] \leftarrow 1$;

for each $f \in F$ do

begin

if f is a neighbor of u_i then **Findpath** (u_i, v, f)

else Findpath (u_{ij}, v_j, f);

end;

return a path using the array $Avail$;

Step 3c : $i \leftarrow lmb(\mathrm{a})$; $j \leftarrow lmb(\mathrm{b})$;

Findnumbers$(Q_{m+2}(u_i, v_j), E(u,v) - A(u) - A(v))$;

for each $e_k \in E(u,v) - A(u) - A(v)$ **do**

begin

$Q'_{m+2} \leftarrow$ the image of $Q_{m+2}(u_i, v_j)$ along dimension e_k;

if $|F(Q'_{m+2})| = 2$ **then**

/* u_{e_k} and v_{e_k} are the only faulty nodes in Q'_{m+2} */

begin

$P \leftarrow$ a path in Q'_{m+2} between u_{ie_k} and v_{je_k};

return $(i \;\; e_k \circ P \circ e_k \;\; j)$;

end;

end;

for $i = 1$ to $n-1$ **do** $Avail[i] \leftarrow 1$;

for each $f \in F$ **do**

begin

if f is a neighbor of u_i **then Findpath** (u_i, v_i, f)

else if f is a neighbor of v_j **then Findpath** (u_j, v_j, f)

else Findpath (u_{ij}, v_{ij}, f);

end;

return a path using the array $Avail$;

Remark 5.2 For a given fault instance F, either Step 2 or Step 3 is executed. Moreover, in Step 3, we need to perform only one of Steps 3a, 3b and 3c.

The overall complexity of the algorithm R is $O(|F| \log n)$.

5.6 Conclusion

We have shown that in the presence of $3n-6$ faults in an n-cube ($n > 5$) under the assumed forbidden faulty set model, the diameter can increase by at most 3. For $n > 5$, only those nodes which are at a Hamming distance of $n-3$ may be at the maximum distance of $n+3$. It is also shown that all such hypercubes with diameter $n+3$ are isomorphic. In [L93a], it was conjectured that the diameter of Q_n would increase by at most β in presence of $\beta(n-2)+1$ faulty nodes ($\beta \leq n-2$), provided the hypercube remains connected. Our results establish that for $\beta = 3$, the fault-diameter part of this conjecture is correct, but not the number of faulty nodes (which is $3n-5$ according to the conjecture instead of $3n-6$).

Since the above results are applicable under the assumption of forbidden faulty sets, it will be useful to find the probability that Q_n remains connected in the presence of $f \geq n$ faulty nodes. It is easy to verify that the probability of Q_n being disconnected, i.e., $\text{prob}(D_f(Q_n) = \infty) = \frac{2^n\binom{2^n-(n+1)}{f-n}}{\binom{2^n}{f}}$, when $n \leq f < 2n-2$. This is so because any one of the 2^n nodes can have all of its neighbors as faulty and the remaining $f-n$ faulty nodes can be distributed over the rest of Q_n. When $f \geq 2n-2$ we need to consider also the case that all $2n-2$ neighbors of two adjacent nodes can be faulty. As there are $n2^{n-1}$ links in Q_n we can express the corresponding probability of Q_n becoming disconnected as

$$\text{prob}(D_f(Q_n) = \infty) = \frac{2^n\binom{2^n-(n+1)}{f-n} + n2^{n-1}\binom{2^n-2n}{f-2n+2}}{\binom{2^n}{f}}, \text{ when } 2n-2 \leq f < 3n-5$$

For large n, the second term in the numerator becomes very small compared to the first term. We have found that for $n = 6$, $\text{prob}(D_{2n-3}(Q_n) = \infty) = 6.8 \times 10^{-5}$ and $\text{prob}(D_{3n-6}(Q_n) = \infty) = 7.1 \times 10^{-4}$.

It can be seen that with $f \geq 3n-6$, the probability that Q_n will be disconected, will continue to increase. Hence, generalization of our results for larger number of faults may not have much practical significance. However, for the sake of theoretical interest, we conjecture that the fault-diameter of Q_n will be $n+\beta$ in the presence $\beta(n-3)+3$ faulty nodes such that the cube remains connected. Here, the forbidden

faulty set will consist of the set of nodes adjacent to a set of i nodes forming a path, $1 \leq i \leq \beta$. But Q_n becomes disconnected if all the $\binom{n}{2}$ nodes at a distance 2 from a particular node is faulty and hence we must have $\binom{n}{2} > \beta(n-3)+3$, i.e., $(n-3)(n-(2\beta-2)) > 0$. From this inequality we get a restriction on the value of β given as $n > 2\beta - 2$.

Chapter 6

Distributed Loop Networks

6.1 Introduction

This chapter deals with certain problems arising from the studies in distributed loop network $G(n; 1, s)$. This structure has certain advantages over the simple ring network regarding diameter, connectivity etc. Since the distributed loop network is a regular and symmetric structure, it should be possible to develop efficient communication algorithms exploiting the underlying symmetry of the structure. The communication algorithms which we focus on are multinode broadcast and single node scatter algorithm.

Our proposed algorithms for multinode broadcast and single node scattering are based on the model of multiple link availability (MLA) [DOS+91], where we assume that each node can receive or send packets through all of its adjacent links simultaneously. Links are bidirectional and full duplex communication is possible through each link. But if there are more than one packet to be transmitted through the same link in the same direction at the same time, only one packet can be sent, and the remaining packets have to wait in a queue. We also assume that each packet transmission takes one unit of time, i.e., packets are all of same length. Packet transmissions are assumed to be error free. Under this model, our proposed algorithms for multiple node broadcast and single node scatter are both

optimal in time. As an overhead, implementation of these algorithms requires a buffer of size n at each node to store the packets from all the nodes of the network.

6.2 Some Properties of $G(n; 1, s)$

Let S_k, $1 \leq k \leq d$, be the set of nodes at a distance k from the node 0, where d is the diameter of the graph $G(n; 1, s)$. A shortest path between any two nodes consists of either one of $+s$ and $-s$ links and/or either one of $+1$ and -1 links. Hence, any node iu S_k, can be written in the form $a_1.s + a_2$, where a_1 and a_2 are integers and $|a_1| + |a_2| = k$. Here $|a_1|$ is the number of $+s$ or $-s$ links and $|a_2|$ is the number of $+1$ or -1 links used in the path. For a node u in S_k, the node $n - u$ is also in S_k. The nodes in S_k can be easily enumerated from those in S_{k-1}. For $k = 1, 2, \cdots, d$, we generate the elements of S_k in pairs, in the following order:

$(k, -k)$, $(s + k - 1, -s - (k - 1))$, $(s - (k - 1), -s + (k - 1))$, $\cdots$,

$(j.s + (k - j), -j.s - (k - j))$, $(j.s - (k - j), -j.s + (k - j))$, $\cdots$, $(k.s, -k.s)$

where all numbers are taken modulo n, the node pair $(k, -k)$ are reached using k number of $+1$ and -1 links respectively, and so on.

It may be noted that in the above method of generation, some elements of S_k may be duplicated and in that case, the corresponding term will be indicated by a dash. To facilitate our later discussions, we construct a $d \times 2d$ matrix M with these generated pairs of S_k's as follows :

1. The elements of S_k are entered in the k-th row of the matrix, each pair generated as above being placed in one column, The pairs are placed in the successive columns in the above order of their generation, starting from the column 1.
2. The entries in columns $2k + 1$ through $2d$ are all filled up by a null ($\emptyset$) entry.
3. If an element is already generated in some S_p, $p \leq k$, then we keep it out to avoid duplication and mark the corresponding position in S_k by a dash (-).

An illustrative example is given below.

Example 6.1 Consider the network $G(14;1,6)$ of Fig. 5.1. The diameter d of this graph is 3. Hence, M will be a 3×6 matrix with the pairs of S_1, S_2 and S_3 as follows :

S_1	:	$(1,13)$	$(6,8)$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
S_2	:	$(2,12)$	$(7,-)$	$(5,9)$	$(-,-)$	$\emptyset$	$\emptyset$
S_3	:	$(3,11)$	$(-,-)$	$(4,10)$	$(-,-)$	$(-,-)$	$(-,-)$

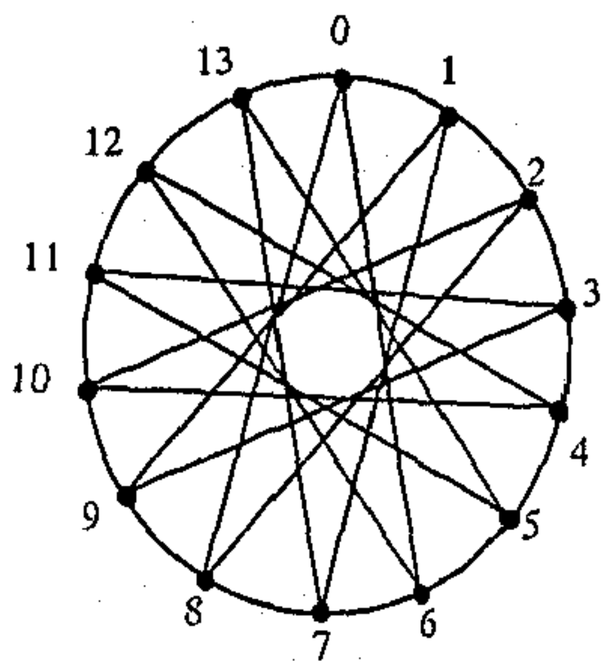


Fig. 6.1 $G(14;1,6)$

Whenever a pair of generated nodes in S_k will be encountered in our discussion, we will treat that as an ordered pair.

6.2.1 Properties of The Sets S_k

We observe the following properties of the sets S_k's (all node numbers are taken **modulo** n) :

Property 1 : The pairs generated as above are always in the form $(u, n-u)$.

Property 2 : In a generated pair, If only u is present, but not $n-u$, then u must be equal to $n/2$. In other words the nodes are duplicated in pairs (except the node $n/2$).

Property 3 : Let j be an integer, $0 < j < k$ and let $k \geq 2$. If an element $j.s+(k-j)$ is present in S_k, then S_{k-1} must contain both the nodes $(j-1)s+(k-j)$ and $j.s + (k - j - 1)$. Similar property also holds for the elements of the form $j.s - (k - j)$, $-j.s - (k - j)$, and $-j.s + (k - j)$ in S_k.

Property 4 : For $k \geq 2$, if the node k is present in S_k, then S_{k-1} must contain the node $k - 1$.

Property 5 : For $k \geq 2$, if the node $k.s$ is present in S_k, then $(k - 1).s \in S_{k-1}$.

Property 6 : In the process of sequentially generating the pairs of S_k, each S_k will contain $2k$ pairs of nodes, until any duplication occurs.

Property 7 : If the first duplication occurs in some S_p, either k will not be present in all S_k, for $k > p$ or $k.s$ will not be present in S_k, for $k \geq p$.

Proof : A duplication occurs in S_p, if there exist some integers i, j, i' and j' so that a node $(js + i) \in S_p$ is already generated as $(j's + i')$ in S_{p-q} for some $q \geq 0$. That is, $(js + i) = (j's + i') \bmod n$, which leads to an equation of the form $a.s + b = 0 \pmod n$, where a and b are integers. Also $|j| + |i| = p$ and $|j'| + |i'| = p - q$. There can be two different cases :

i) $a > 0, b > 0$ and ii) $a > 0, b \leq 0$.

Case I : $a > 0, b > 0$.

There can be two possible subcases : 1) $a < b$ and 2) $a \geq b$.

Subcase 1 : If $a < b$, then $a < (a + b)/2 < b$.

If $a + b$ is odd, then from the equation $a.s + b = 0 \pmod n$, we can write,

$$\lceil (a + b)/2 \rceil = -\{a.s + \lfloor (b - a)/2 \rfloor\}$$

Thus for $p = \lceil (a + b)/2 \rceil$, the node $\lceil (a + b)/2 \rceil$ will be generated in S_{p-1} with $j' = -a$ and $i' = -\lfloor (b - a)/2 \rfloor$. Hence, the pair $(p, -p)$ will be absent in S_p and hence, the pair $(k, -k)$ will be absent in S_k for all $k \geq p$ (by property 4).

If $a+b$ is even, then we can write,

$$(a+b)/2 = -\{a.s + (b-a)/2\}$$

Thus the node $-a.s-(b-a)/2$ will be generated as $(a+b)/2$ in S_p for $p=(a+b)/2$. Now, $(a+b)/2+1 = -\{a.s+(b-a)/2-1\}$. That means, the node $(a+b)/2+1$ has already been generated in S_{p-1}. Hence, the pair $(k,-k)$ will be absent in S_k for all $k \geq p$ (by property 4).

Subcase 2 : If $a \geq b$ then $a \geq (a+b)/2 \geq b$.

If $a+b$ is odd, then from $a.s+b=0 \pmod n$, we have

$$\lfloor (a-b)/2 \rfloor .s + b = -\lceil (a+b)/2 \rceil .s$$

So for $p = \lceil (a+b)/2 \rceil$, the node $p.s = \lceil (a+b)/2 \rceil .s$ will be a duplicate generation in S_p and hence for $k > p$, the pair $(k.s, -k.s)$ will be absent in S_k. If $a+b$ is even, then the first duplication occurs in S_p for $p=(a+b)/2$ in the form $(a-b)/2.s+b = -(a+b)/2.s$. Then also, the pair $(k.s,-k.s)$ will be absent in all S_k for $k \geq p$.

Case II : $a>0$, $b<0$. The proof is similar to above.

Example 6.2 For $n=13$ and $s=3$ we have an equation $s-3=0$ and the pair $(3,-3)$ is absent in S_3 as evident from the following matrix.

S_1	:	$(1,12)$	$(3,10)$			
S_2	:	$(2,11)$	$(4,9)$	$(-,-)$	$(6,7)$	
S_3	:	$(-,-)$	$(5,8)$	$(-,-)$	$(-,-)$	$(-,-)$	$(-,-)$

Example 6.3 For $n=12$ and $s=5$ we have an equation $2s+2=0 \pmod{12}$ and the pair $(2s,-2s)$ is absent in S_2 (duplicated in S_2 itself) as can be seen from the following matrix.

S_1	:	$(1,11)$	$(5,7)$			
S_2	:	$(2,10)$	$(6,-)$	$(4,8)$	$(-,-)$	
S_3	:	$(3,9)$	$(-,-)$	$(-,-)$	$(-,-)$	$(-,-)$	$(-,-)$

Property 8 : If n is even and the node $n/2$ occurs in S_k in any form other than $(k.s)$ then all 4 nodes adjacent to the node $n/2$ must be in S_{k-1}.

Proof : Since $s > 1$, the diameter d of $G(n; 1, s)$ will be less than $n/2$. Hence the set $S_{n/2}$ will not be generated in the above process, i.e., $n/2$ cannot appear either as k or as $-k$ in any S_k. Suppose $n/2$ can be expressed in the form $j.s + (k - j)$, $0 < j < k$. Then, $n/2$ can also be expressed in the form $-j.s - (k - j)$. Hence all 4 nodes adjacent to it will be in S_{k-1}. Similarly, we can prove for the case when $n/2$ is of the form $j.s - (k - j)$.

Property 9 : If for some k, $|S_k| < 4$, then k is the diameter of the graph.

Proof : The only way $|S_k|$ can be equal to 1 or 3, is the case when one of the generated pairs in S_k has only one dash $(-)$ in it, i.e., when the node $n/2$ is in S_k (by property 2). Thus for odd n, $|S_k|$ cannot be equal to 1 or 3. We now consider the following cases.

Case I : n is odd.

In this case, $|S_k| = 2$. The elements in S_k must be a pair of the form $(u, n - u)$. Suppose S_{k+1} is non-empty and contains a pair in any form other than $(k+1, -k-1)$ or $((k+1).s, -(k+1).s)$. Then, S_k must have two pairs of nodes (by property 3), which is a contradiction. So if S_{k+1} is non-empty, the elements in it must be either $(k+1, -k-1)$ or $((k+1).s, -(k+1).s)$.

a) Let the pair in S_{k+1} be $(k+1, -k-1)$. Then $S_p = \{(p, -p)\}$, for $k+1 \leq p \leq d$. If a node $j \in S_k$, then the 4 nodes adjacent to j must be in $S_{k-1} \cup S_k \cup S_{k+1}$. Now if $d > k$, we cannot find four nodes adjacent to the node $d \in S_d$, as S_{d-1} and S_d together can have 4 nodes including d, i.e., only 3 nodes adjacent to the node d. So S_{k+1} must be empty.

b) Similarly we can prove for the case when the pair is of the form $((k+1).s, -(k+1).s)$.

Hence if $|S_k| = 2$, then $k = d$.

Case II : n is even.

The proof is similar to that in case I, when $|S_k| < 3$. So, we consider only the case when $|S_k| = 3$. This is possible only when $n/2 \in S_k$.

First we give a brief outline of the proof. We show that $|S_{k+1}| \leq 2$, which will imply $d \leq k+1$ (arguments as in Case Ia above applies). Then we show that for a node in S_{k+1}, we cannot get 4 nodes adjacent to this node in $S_k \cup S_{k+1}$ leading to a contradiction.

i) $n/2$ is of the form $j.s + (k-j)$ or $j.s - (k-j)$, where $0 < j < k$.
All 4 nodes adjacent to $n/2$ are in S_{k-1} (by property 8). Since there is duplication in S_k, either $k+1$ or $(k+1).s$ will be absent in S_{k+1}. Hence, if S_{k+1} contains more than one pair of nodes, then at least one of them will be in a form other than $(k+1, -k-1)$ and $((k+1)s, -(k+1)s)$. Then S_k must have two pairs of nodes which is a contradiction. Thus, S_{k+1} can have only one pair. Since $n/2$ is not adjacent to any node in S_{k+1}, the proof follows as in Case Ia.

ii) $n/2$ is of the form $k.s$. Then the pair $((k+1).s - (k+1).s) \notin S_{k+1}$ as $k.s + s = -k.s + s = -(k-1).s \pmod{n}$, which has already appeared in S_{k-1}. Also, $k.s - 1 = -k.s - 1$, and hence the pair $(k.s-1, -k.s+1)$ will be absent in S_{k+1}, as this pair will already be generated as $(k.s+1, -k.s-1)$ in the process of sequential generation of pairs defined earlier. Now, for $s = 2$, $k.s + 1 = -k.s + 1 = -(k-1)s - 1 \in S_k$. Hence, if $(k.s + 1, -k.s - 1) \in S_{k+1}$, then s must be greater than 2 and also $((k-1)s+1, -(k-1).s-1)$ must be present in S_k. Then S_k cannot have any other pair. But we cannot find 4 nodes adjacent to the node $(k.s+1) \bmod n = n/2+1$, as there are only 3 nodes in S_k and the node $(-k.s-1) \bmod n = n/2-1$ in S_{k+1} cannot be adjacent to $(n/2+1)$. Thus the pair $(k.s+1, -k.s-1) \notin S_{k+1}$.

If S_{k+1} contains any pair other than $(k+1, -k-1)$ or $((k+1).s, -(k+1).s)$, then S_k must have 2 pairs (by property 3). Hence S_{k+1} can only contain a pair of the form $(k+1, -k-1)$. The node $k+1$ is connected to node $k \in S_k$ by link of type -1 and it cannot be connected to $k.s$ by links of type $+s$ or $-s$. Hence the two nodes which are adjacent to $k+1$ by links of type $+s$ and $-s$ must be $-k$ and $-(k+1)$. This implies $2(k+1) = (-k-(k+1)) \bmod n$. Then n must be odd,

which is a contradiction and hence the proof. □

Property 10 : The diameter d of $G(n; 1, s)$ is less than or equal to $\lceil (n-1)/4 \rceil$.

6.3 Multinode Broadcast

Since the degree of a node in a distributed loop network is 4, a node can receive at most 4 packets at a time. Since each node has to receive a total of $(n-1)$ packets for multiple node broadcast, the minimum time for multiple node broadcast is $\lceil (n-1)/4 \rceil$. The minimum number of packet-transmissions required for this operation is $n(n-1)$. An algorithm for multiple node broadcast that requires $\lceil (n-1)/4 \rceil$ time, will be optimal with respect to time.

For broadcast from a single node, we first construct a spanning rooted tree from the given network, with the source node placed at the root. The source node starts sending its packet at time $t = 0$ through the links included in the spanning tree. Subsequently, each of the remaining nodes receives a packet from its parent node in the spanning tree and sends it to its children nodes. Let $A_{t,0}$ denote the set of links of the spanning tree those are used for communication at the time instant t (the suffix 0 stands for the source node 0), $t = 1, 2, \cdots, T$. Also, let $[v, w]$ denote the link connecting the nodes v and w. If a packet is sent from a node v to a node w at the time instant t, then the directed link $[v, w]$ connecting v to w will be an element of $A_{t,0}$. Let E_t denote the set of the terminal nodes of all the links in $A_{t,0}$. Then we have the following :

1) All the links in $A_{1,0}$ must have the same starting node 0.

2) Starting nodes of the links in $A_{t,0}$ must be in the set $\bigcup_{j=1}^{t-1} E_j$

3) E_t must cover all the nodes in the network, i.e., $\bigcup_{t=1}^{T} E_t$.

For multiple node broadcast, we can extend the above idea to construct n spanning trees, each for one source node. When we broadcast along all these spanning trees

simultaneously, we have to ensure that a single link transmits only one packet at a time in a given direction and at the same time, the broadcast is completed in optimal time. For every source node u, $0 < u \leq n-1$, we now define the sets $A_{t,u}$ for different t, in exactly the same way as we have defined the sets $A_{t,0}$ for the source node 0. That is, corresponding to every link $[v, w] \in A_{t,0}$, the link $[(v+u) \bmod n, (w+u) \bmod n]$ is included in $A_{t,u}$ for all $u > 0$. This is possible as the network is vertex symmetric with respect to modulo n rotation. This idea of using homogeneous algorithm for multiple node broadcast has been applied to hypercubes before [BOS+91].

If we start broadcasting from all the nodes simultaneously according to the sets $A_{t,u}$, we have to ensure that for all t, $A_{t,u} \bigcap A_{t,v} = \emptyset$ when $u \neq v$. Otherwise, if two sets $A_{t,u}$ and $A_{t,v}$ have a link in common, we say that a conflict has occurred, in which case a single link would be required to transmit more than one packet in a given direction at the time instant t, which is contrary to our assumed model.

In a distributed loop network, we have defined the type of a link $[v, w]$ connecting v to w as $\Delta(v, w)$, where $w = (v + \Delta(v, w)) \bmod n$. The links can be of four types namely $+1$, -1, $+s$ and $-s$.

Example 6.4 Consider the distributed loop network $G(14; 1, 6)$ as shown in Fig. 6.1.

The type of the link [1, 2] is $+1$ and that of the link [1, 7] is $+6$. The types of the links [13, 0] and [12, 4] are also $+1$ and $+6$ respectively. The types of the links [6, 7], [0, 13], [1, 7] and [6, 0] are $+1$, -1, $+6$ and -6 respectively.

Suppose we construct the sets $A_{t,0}$ in such a way that for all t, no two links in $A_{t,0}$ are of the same type. Thus for all t, $A_{t,u}$ can have a maximum of 4 links, all of different types. We claim that no conflict would arise for such a choice of the links in $A_{t,u}$, i.e., $\forall t, A_{t,u} \bigcap A_{t,v} = \emptyset$ for $v \neq u$. We prove this claim by contradiction as follows. If possible, let a link in $A_{t,u}$ be in common with a link in $A_{t,v}$, for $v \neq u$. Then, for some $[w_1, x_1] \in A_{t,0}$ and $[w_2, x_2] \in A_{t,0}$, $[(w_1 + v) \bmod n$, $(x_1 + v) \bmod$

$n] = [(w_2 + u) \bmod n, (x_2 + u) \bmod n]$ This implies that $v - u = (w_2 - w_1) \bmod n = (x_2 - x_1) \bmod n$. Then, $(x_1 - w_1) \bmod n = (x_2 - w_2) \bmod n$, i.e., we must have two links of the same type in $A_{t,0}$, which is a contradiction.

In what follows we shall describe a method for generating $A_{t,0}$'s such that for all t, $A_{t,0}$ contains at most one link of each type. We will use these $A_{t,0}$'s to devise an algorithm for multiple node broadcast algorithm in optimal time.

6.3.1 Generation of $A_{t,0}$

Definition 6.1 Let (w, x) be a pair in S_k. For (w, x) in any form other than $(k.s, -k.s)$ we can choose a pair (w_1, x_1) from S_{k-1}, such that either $\Delta(w_1, w) = 1$, $\Delta(x_1, x) = -1$ or $\Delta(w_1, w) = -1$, $\Delta(x_1, x) = 1$. Similarly for any pair (w, x) in S_k which is not of the form $(k, -k)$, we can select a pair (w_s, x_s) from S_{k-1}.

Definition 6.2 We define an ordered set S'_k, by selecting the non-null elements of the k-th row of the matrix M, in order from left to right, excluding the dashes(-) (each element being a pair of nodes or a singleton when the node is $n/2$).

Example 6.5 For $G(14; 1, 16)$, $S'_2 = \{(2, 12), (7), (5, 9)\}$, $S'_3 = \{(3, 11), (4, 10)\}$.

Remark 6.1 The purpose of S'_k is to keep track of the nodes which are not yet included in the spanning tree.

We first give an overview of the generation process. $A_{t,0}$'s are generated in the increasing order of t, starting from $t = 1$. In S_1, there are two pairs $(1, -1)$ and $(s, -s)$. $A_{1,0}$ consists of the links joining the node 0 to the pairs in S_1. We generate the subsequent $A_{t,0}$'s as follows.

We note that we have generated the elements in S_k in the following order :
$(k, -k)$, $(s + k - 1, -s - (k - 1))$, $(s - (k - 1), -s + (k - 1))$, ..., $(js + (k - j)$, $-js - (k - j))$, $(js - (k - j), -js + (k - j))$, ..., $(k.s, -k.s)$.

If there is no duplication upto the generation of S_k, then there will be $2k$ pairs in S_k. We select a set of 4 nodes $\{w, x, y, z\}$ from S_k as follows. We select the leftmost pair in S'_k. Let this pair be (w, x). We also select the rightmost pair in S'_k. Let this pair be (y, z). We can always find $A_{t,0}$ connecting the nodes in the sets $\{w, x, y, z\}$ satisfying the requirement of different link types as $\{[w_1, w], [x_1, x], [y_s, y], [z_s, z]\}$. Next we delete the pairs (w, x) and (y, z) from S'_k and select the leftmost and rightmost pairs in the new S'_k. This way we go on until all the nodes in the network are included in the spanning tree.

Example 6.6 We give an example for computing the sets $A_{t,0}$ for $n = 13$ and $s = 3$.

$$\begin{array}{lcllll} S_1 & : & (1,12) & (3,10) & & \\ S_2 & : & (2,11) & (4,9) & (-,-) & (6,7) \\ S_3 & : & (-,-) & (5,8) & & \end{array}$$

Here, $S_1 = \{1, 12, 3, 11\}$ and $A_{1,0} = \{[0, 1], [0, 12], [0, 3], [0, 10]\}$. From S_2, $(w, x) = (2, 11)$, $(y, z) = (6, 7)$ and $A_{2,0} = \{[1, 2], (12, 11), (3, 6), (10, 7)\}$ The remaining pair in $S_2 = (4, 9)$ and $S_3 = (5, 8)$. Hence, $A_{3,0} = \{[1, 4], [12, 9], [6, 5], [7, 8]\}$

If $(n-1)$ mod $4 = 0$, all the 4 link types will be present in each $A_{t,0}$. If $(n-1)$ mod $4 = 2$, we make all but the last $A_{t,0}$ with 4 links. If $(n-1)$ mod $4 = 3$, one of the $A_{t,0}$'s will have 3 links and the rest 4 links each. If $(n-1)$ mod $4 = 1$, we can make either one $A_{t,0}$ with 1 link and the rest with 4 links each or one $A_{t,0}$ having 3 links, another $A_{t,0}$ having 2 links and the rest with 4 links each. We now consider below a few more points in detail for odd and even values of n.

For odd n

Since n is odd, the elements of S_k will always occur in pairs. If a duplication occurs in some S_p, for $p \leq k$, then some pairs will be absent in S_p and there may be odd number of pairs in S_k. In that case we have to select $A_{t,0}$ by taking one pair from S_k and one pair from S_{k+1}. By property 7, either the pair $(k+1, -k-1)$ or $((k+1)s, -(k+1)s)$ will be absent in S_{k+1}.

Consider the case when the pair $(k+1, -k-1)$ is absent in S_{k+1}, and S'_k contains one pair say (w, x). Since S_{k+1} is non-empty S_k must have more than one pair and since we have started to connect the nodes in S_k by selecting the leftmost and rightmost pairs, (w, x) cannot be of the form $(k, -k)$ or $(k.s, -k.s)$. We select the rightmost pair from S_{k+1} say (y, z). If the pair (y, z) is of the form $((k+1)s, -(k+1)s)$, we construct $A_{t,0}$ as $\{[w_1, w], [x_1, x], [y_s, y], [z_s, z]\}$. Here, $(y_s, z_s) = (k.s, -k.s)$ which are already included in the spanning tree. If (y, z) is not $((k+1)s, -(k+1)s)$, then we make $A_{t,0} = \{[w_1, w], [x_1, x], [y_s, y], [z_s, z]\}$ provided $(w, x) \neq (y_s, z_s)$, otherwise we make $A_{t,0} = \{[w_s, w], [x_s, x], [y_1, y], [z_1, z]\}$.

If the pair $(k+1, -k+1)$ is present in S_{k+1} we select the pair (y, z) from the leftmost end of S_{k+1} and proceed as before.

For even n

After any duplication has occurred, the number of elements in S_k will be odd only when $n/2$ is present in S_k, and even otherwise. The generation process is same as that for odd n, except when the node $n/2$ is to be connected to the tree. When the node $n/2$ is to be connected and there remains another pair (w, x) unconnected to the spanning tree, we make $A_{t,0}$ consisting of 3 links, connecting the nodes $(w, x, n/2)$ and also maintaining the property of different link types.

If no other node remains unconnected to the tree, then we make $A_{t,0}$ consisting of only 1 link connecting $n/2$ to the tree.Algorithm A to generate $A_{t,0}$ for $t = 1, 2, \cdots, \lfloor (n-1)/4 \rfloor$ is described below.

Algorithm A

Step 1 : Generate the sets $S_1, S_2, \ldots, S_d$;
/* properties 3, 4, 5 and 9 are used to reduce calculation */
for $k = 1, 2, \cdots, d$, set $S'_k \leftarrow S_k$;

Step 2 : $t \leftarrow 1$; $k \leftarrow 1$;

Step 3 : **If** $k > d$ **then** stop.

If $|S'_k| > 4$ then

/* S'_k denotes the set of nodes in S_k which are not yet connected */

begin

if leftmost element in S'_k does not occur in pair then

begin

$w \leftarrow n/2$;

(y, z) rightmost pair in S'_k; $A_{t,0} \leftarrow \{[w_1, w], [y_s, y], [z_s, z]\}$

end

else if rightmost element in S'_k does not occur in pair **then**

begin

$y \leftarrow n/2$; $(w, x) \leftarrow$ leftmost pair in S'_k;

$A_{t,0}. \{[w_1, w], [x_1, x], [y - s, y]\}$;

end

else

begin

$(w, x) \leftarrow$ leftmost pair in S'_k; $(y, z) \leftarrow$ rightmost pair in S'_k;

$A_{t,0} \leftarrow \{[w_1, w], [x_1, x], [y_s, y], [z_s, z]\}$;

end;

update S'_k; $t \leftarrow t + 1$; go to Step 3;

end;

If $|S'_k| = 3$ then /* S'_k contains node $n/2$ */

begin

$(w, x) \leftarrow$ the pair in S'_k; $y \leftarrow n/2$; $A_{t,0} \leftarrow \{[w_1, w], [x_1, x], [y - s, y]\}$;

update S'_k; $t \leftarrow t + 1$; $k \leftarrow k + 1$; go to step 3;

end;

If $|S'_k| = 2$ then

begin

$(w, x) \leftarrow$ the pair in S'_k; $k \leftarrow k + 1$;

If $k > d$ then

begin

$A_{t,0} \leftarrow \{[w_1, w], [x_1, x]\}$; stop;

end;

If $(k, -k)$ is absent in S_k then

```
if rightmost element in S_k occurs as a pair then
begin
    (y, z) rightmost pair in S_i;
    If (w, x) ≠ (y_s, z_s) then A_{t,0} ← {[w_1, w], [x_1, x], [y_s, y], [z_s, z]}
        else A_{t,0} ← {[w_s, w], [x_s, x], [y_1, y], [z_1, z]};
end;
else                    /* rightmost element in S_i is n/2 */
begin
    y ← n/2;
    If (w, x) ≠ (y − s) then A_{t,0} ← {[w_1, w], [x_1, x], [y − s, y]}
        else A_{t,0} ← {[w_s, w], [x_s, x], [y − 1, y]}
end;
else                    /* (k, −k) is present in S_k */
begin
    (y, z) ← leftmost pair in S_k; A_{t,0} ← {[w_s, w], [x_s, x], [y_1, y], [z_1, z]};
end;
    t ← t + 1; update S'_k; go to step 3
end;
If |S'_k| = 1 then /* S'_k contains the element n/2 = y */
begin
    y ← n/2; k ← k + 1;
    if k > d then
    begin
        A_{t,0} ← [y − 1, y]; stop;
    end
    else if (k, −k) is absent in S_k then
    begin
    (w, x) ← the rightmost pair in S'_k;
    if {w_s, x_s} ⋂ {y} = ∅ then A_{t,0} ← {[w_s, w], [x_s, x], [y − 1, y]}
        else A_{t,0} ← {[w_1, w], [x_1, x], [y − s, y]};
end
else                    /* (k, −k) is present in S_k */
begin
```

$(w, x) \leftarrow$ the leftmost pair in S'_k; $A_{t,0} \leftarrow \{[w_1, w], [x_1, x], [y - s, y]\}$

end;

$t \leftarrow t + 1$; update S'_k; go to step 3;

end; /* $|S'_k| = 0$*/

Step 4 : $k \leftarrow k + 1$; go to step 3.

Example 6.7 We give an example for computing the sets $A_{t,0}$ for $n = 14$ and $s = 3$.

$$\begin{array}{lcllll} S_1 & : & (1,13) & (3,11) & & \\ S_2 & : & (2,12) & (4,10) & (-,-) & (6,8) \\ S_3 & : & (-,-) & (5,9) & (7,-) & \end{array}$$

Here, $A_{1,0} = \{[0, 1], [0, 13], [0, 3], [0, 11]\}$. From S_2, $(w, x) = (2, 12)$ and $(y, z) = (6, 8)$ and $A_{2,0} = \{[1, 2], [13, 12], [3, 6], [11, 8]\}$. The remaining pair in $S_2 = (4, 10) = (w, x)$ and $y = 7$. Here $(w_1, x_1) = (3, 11)$ and $(w_s, x_s) = (1, 13)$ and $y_1 = 6$. Hence, $A_{3,0} = \{[1, 4], [13, 10], [6, 7]\}$ The remaining pair in $S_3 = (5, 9)$ and $A_{4,0} = \{[4, 5], [10, 9]\}$

We now give an overall idea for the multiple node broadcast algorithm. At each node, a buffer of size n is needed to store data packets from n processors. Each of these buffers has n locations with addresses, ranging from 0 to $n - 1$. The packet originated from node r will be placed in the location $(r - u)$ of the buffer of node u. The overall arrangement can be shown as

	$P(u)$	$P(u+1)$	$P(u+2)$	. . . $P(r)$ . . .	$P(u-1)$
address	0	1	2	$r - u$	$n - 1$

$P(r)$: Data packet originated from node r.

The sets $A_{1,0}$, $A_{2,0}$, ..., $A_{T,0}$ are determined beforehand once for all, and we store for each instant of time, the address of the buffer from which a data packet is to be sent and the link along which that packet is to be sent. Let $[w, x] \in A_{t,0}$. Then $[w+u, x+$

$u] \in A_{t,u}$. Hence at time t, the node $(w+u)$ must send the packet originated from the node u, i.e., $P(u)$ which is stored in location $(n-w)$ of its buffer, to the node $(x+u)$. To implement this, we need to store the buffer address $(n-w)$ and the link type $\Delta(w+u, x+u)$ which is same as the link type $\Delta(w, x)$. Thus the information regarding the link $[w, x]$ in $A_{t,0}$ is sufficient to effect transmission of data packets from all the nodes in the network. If $A_{t,0} = \{[w_1, w], [x_1, x], [y_s, y]), (z_s, z)\}$ we store the t-th record consisting of four pairs $(b_1, l_1), (b_2, l_2), (b_3, l_3), (b_4, l_4)$ for any node $(w+u)$ as follows

$(n-w_1)$	$\Delta(w_1, w)$	$(n-x_1)$	$\Delta(x_1, x)$	$(n-y_s)$	$\Delta(y_s, y)$	$(n-z_s)$	$\Delta(z_s, z)$
b_1	l_1	b_2	l_2	b_3	l_3	b_4	l_4

There will be $\lceil (n-1)/4 \rceil$ such records. For $t = 0, 1, \cdots \lceil (n-1)/4 \rceil - 1$, each node will fetch the t-th record, and transmit the packet in location b_i along the link of type l_i.

6.4 Single Node Scatter

In scattering, a node has to send $(n-1)$ different packets to each of the other nodes in the network. Since a node can transmit at most 4 packets at a time, the minimum time required for single node scatter is $\lceil (n-1)/4 \rceil$. We will present now a time-optimal algorithm for single node scatter which requires $\lceil (n-1)/4 \rceil$ units of time.

To describe our scattering algorithm, we assume that the node 0 is the source node. The packets will be transmitted from the node 0, along a spanning tree T rooted at node 0. T consists of 4 subtrees T_{+1}, T_{-1}, T_{+s} and T_{-s} rooted at the nodes $+1$, -1, $+s$ and $-s$ respectively. Each of the 4 subtrees contains at most $\lceil (n-1)/4 \rceil$ nodes.

With such a construction of the spanning tree, all the nodes will receive their

packets within time $\lceil (n-1)/4 \rceil$, if the following rule for transmission of packets is obeyed [BOS+91].

Node 0 sends packets to distinct nodes in the subtree (using only the links in T), giving priority to the nodes farthest away from node 0 (breaking ties arbitrarily).

We also ensure that each packet travels along the shortest path to its destination by making T a shortest path tree.

6.4.1 Construction of the Spanning Tree

We find the sets S_k's for the graph $G(n; 1, s)$ as before. We maintain the property that if a node u of a generated pair $(u, n-u)$ is in T_{+1}, then the node $(n-u)$ will be in T_{-1} or if u is in T_{+s}, then $(n-u)$ will be in T_{-s}. We divide the total set of $(n-1)$ nodes into two partitions of nearly equal size : partition I, consisting of the pairs which will be included in the trees T_{+1} and T_{-1} , and partition S, consisting of the pairs which will be included in the trees T_{+s} and T_{-s}.

Before going into the details of partitioning the nodes, we make the following observations on the matrix M.

Obs. 1. In row k, the pair in column 1 is of the form $(k, -k)$. So we put all the pairs in column 1 in partition I.

Obs. 2. All the pairs of the form $(k.s, -k.s)$ will be put in the partition S.

Obs. 3. If a node u of a pair $(u, n-u)$ in S_k, is adjacent to some node u' in S_{k-1} then $(n-u)$ is adjacent to the node $(n-u')$ in S_{k-1}.

The method of grouping the nodes for partition I and partition S is almost identical for odd and even values of n. First we describe the procedure for odd n.

For odd n

Since n is odd, there will be a total of $(n-1)/2$ pairs in all the sets S_k's. We collect the pairs for partition I as follows. We leave out the pairs of the form $(k.s, -k, s)$. We take all the pairs in column 1. The maximum number of such pairs is $\lceil (n-1)/4 \rceil$. If the number of pairs in column 1 is $\lceil (n-1)/4 \rceil$ then we put all these pairs in partition I and the rest in partition S. Otherwise, from successive columns we select pairs starting at the bottom of that column and move upwards until we get $\lceil (n-1)/4 \rceil$ pairs [See Example 6.8]. Later on, we will show that it is indeed possible to collect $\lceil (n-1)/4 \rceil$ pairs in this way.

The pairs in partition I are connected in such a way that if one node of a pair is connected to T_{+1}, then the other node of that pair is connected to T_{-1}. Now we have the following lemmas.

Lemma 6.1 *Suppose $(u, n-u)$ is a pair in partition I in some column c. Then the pair $(u, n-u)$ can always be connected to the subtrees T_{+1} and T_{-1}.*

Proof : The pairs in column 1 are of the form $(k, -k)$. The node k is included in T_{+1} and the node $-k$ is included in T_{-1}.

Let the pair be of the form $(j.s + i, -j.s - i)$ in S_k. Then the pair $(i, -i)$ is in column 1 of which i is included in T_{+1} and $-i$ is included in T_{-1} respectively. All the pairs $(p.s + i, -p.s - i)$, where $1 \leq p \leq j$ are in partition I and the node $p.s + i$ can be connected to T_{+1} by a link of type $+s$ and the node $-p.s - i$ can be connected to T_{-1} by a link of type -1.
The proof is similar for a pair of the form $(j.s - i, -j.s + i)$. □

Lemma 6.2 *The pairs in partition S can be connected to the subtrees T_{+s} and T_{-s}, after we connect the pairs in partition I to T_{+1} and T_{-1}.*

Proof : Let a pair in partition S be of the form $(j.s + (k-j), -j.s - (k-j))$. The pair $(j.s, -j.s)$ will be in partition S. So the node $j.s$ will be connected to

T_{+s} and the node $-j.s$ will be connected to T_{-s} respectively. Also all the pairs $(j.s+i, -j.s-i)$, $0 < i < k-j$, are in partition S and can be connected to T_{+s} and T_{-s} by links of type $+1$ and -1 respectively.

The proof is similar for a pair of the form $(j.s-(k-j), -j.s+(k-j))$. □

Lemma 6.3 *It is possible to get at least* $\lfloor (n-1)/4 \rfloor$ *pairs in partitions I.*

Proof : We have shown that the diameter d of the graph is less than or equal to $\lceil (n-1)/4 \rceil$. Total number of pairs for odd n is $(n-1)/2$. Since we have to leave out only the pairs of the form $(k.s, -k.s)$ for partition S, the maximum number of pairs that we can have in partition I is equal to $(n-1)/2$ − number of such pairs of the form $(k.s, -k.s)$, which is same as $(n-1)/2$ − number of rows in the matrix M $= (n-1)/2 - d \geq (n-1)/2 - \lceil (n-1)/4 \rceil = \lfloor (n-1)/4 \rfloor$ □

For even n

The construction of the spanning tree in this case is slightly different. When we encounter the nodes in pairs the same procedure is followed as before. Only the node $n/2$ will not occur in pair. The following cases can occur:

Case I. $n/2$ is in row k and is of the form $j.s+(k-j)$ or $j.s-(k-j)$, $0 < j < k$:

a) $(n-1) \bmod 4 = 3$.

We make partition I consisting of $\lceil (n-1)/4 \rceil$ pairs provided that the node $n/2$ is not encountered in the process. Otherwise, partition I will have $\lfloor (n-1)/4 \rfloor$ pairs and the node $n/2$. If $n/2$ is included in partition I, then we can connect this node to either T_{+1} or T_{-1} tree. If $d < \lceil (n-1)/4 \rceil$, we can always get the required number of pairs in partition I, as the number of pairs of the form $(p.s, -p.s)$ is at most equal to d. If $d = \lceil (n-1)/4 \rceil$, only the d-th row of the matrix will have 3 nodes and all other rows will have 4 nodes each (by property 9). Hence the node $n/2$ will occur in the last row. Partition S will have d pairs only if all the rows

contain pairs of the form $(p.s, -p.s)$. In that case we would encounter the node $n/2$ to form partition I after $\lfloor (n-1)/4 \rfloor$ pairs.

b) $(n-1) \bmod 4 = 1$.

We make partition I consisting of $(n-1)/4$ pairs. If the node $n/2$ is encountered in the process it will be also included. By the same reasoning as above we will always get at least $(n-1)/4$ pairs.

Case II. $n/2$ is in row k and of the form $k.s$:

In this case $n/2$ appears in column $2k$ and all columns $c > 2k$ will be empty, as shown in the proof of property 9. Here $n/2 \in S$. If the pair $(k.s+1, -k.s-1)$ is present in row $(k+1)$ and is in partition S, to connect this pair properly we have to keep the pair $((k-1).s+1, -(k-1).s-1))$ in partition S (the node $k.s. = n/2$ becomes a leaf node). We show below that we get at least $\lfloor (n-1)/4 \rfloor$ pairs required for Partition I.

a) $(n-1) \bmod 4 = 3$.

If $d = \lceil (n-1)/4 \rceil$, the node $n/2$ will be in the d-th row. Since we have one pair of the form $(p.s, -p.s)$ in rows 1, 2, ..., $d-1$, the number of such pairs is $\lfloor (n-1)/4 \rfloor$ and we can get $\lceil (n-1)/4 \rceil$ pairs for partition I. If $d < \lceil (n-1)/4 \rceil$, we leave out the pair $((k-1).s+1, -(k-1).s-1)$ in the k-th row while making partition I. Thus we have to keep the node $n/2$ and k pairs for partition S. As $k \leq d < \lceil (n-1)/4 \rceil$, we can certainly get $\lceil (n-1)/4 \rceil$ pairs for partition I.

b) $(n-1) \bmod 4 = 1$.

By the same reasoning as given above we will get at least $\lfloor (n-1)/4 \rfloor$ pairs for partition I. □

Example 6.8 For $n = 14$, $s = 4$, we first construct the matrix M as follows :

$S1$	:	*(1,13)*	$(4,10)$			
$S2$	:	*(2,12)*	*(5, 9)*	$(3,11)$	$(8,6)$	
$S3$	:	$(-,-)$	$(-,-)$	$(-,-)$	$(-,-)$	$(7,-)$	$(-,-)$

The 3 underlined pairs are in partition I and others are in partition S. The spanning tree is shown in Fig. 6.2.

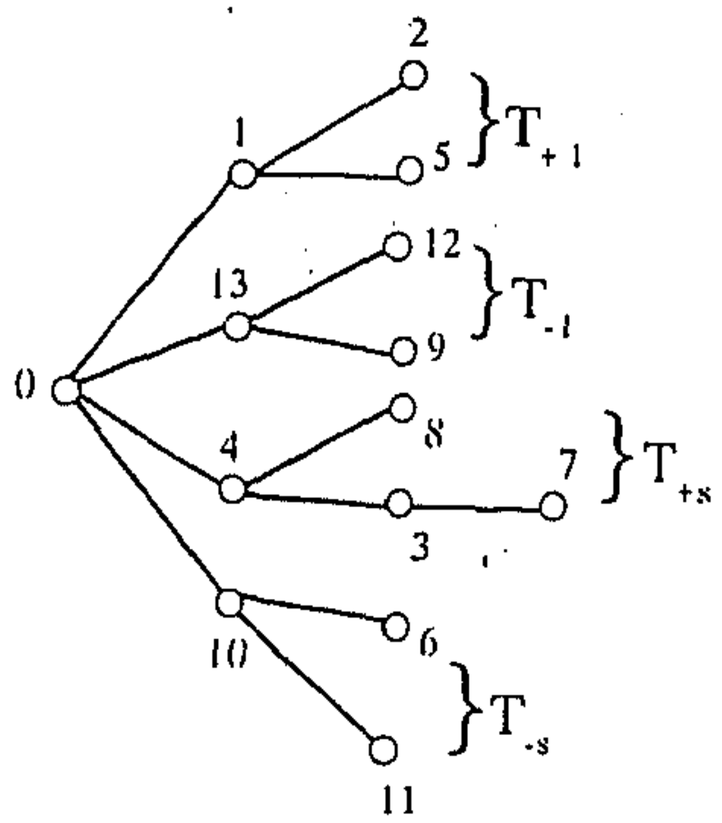


Fig. 6.2 Spanning tree for single node scatter.

Let C_t denote the set of nodes which are destinations of the packets sent from the node 0 at time t. After constructing the partitions I and S, the sets C_t's are generated by the following Algorithm.

Algorithm C :

Step 1 : The nodes in partition I and S are arranged in decreasing order of their distance from node 0.

Step 2 : For $t = 0, 1, ..., \lceil (n-1)/4 \rceil - 1$, do the following :

If n is odd then a pair of nodes from partition I and a pair of nodes from partition S, taken in the above order, constitute C_t.

If n is even, then C_t's are generated in the same way as above except when the node $n/2$ is encountered. Let the node $n/2$ is encountered at $t = t_1$. If another pair of nodes remain to be included in C_t's, then $|C_{t_1}| = 3$, otherwise $|C_{t_1}| = 1$. When, $|C_{t_1}| = 3$, if $n/2 \in I$, then C_{t_1} consists of $n/2$ and a pair in partition S. If $n/2 \in S$, then C_{t_1} consists of $n/2$ and a pair in partition I.

To effect the transmission of data packets from node 0, each packet will be associated with a *tag* depending on its destination node. If a node is in partition I and

appears in the matrix M in the form $j.s + k$, $j, k > 0$ the packet destined to this node will have a *tag* like this :

link type				partition type
$+s$	$-s$	$+1$	-1	
j	0	k	0	I

Similarly a packet destined to a node $-j.s + k$ in partition S will have a *tag* like this :

link type				partition type
$+s$	$-s$	$+1$	-1	
0	j	k	0	S

The transmission of a packet at any node will be made only along a link of type for which there is a non-zero entry in the tag associated with the packet. Once it is decided to transmit a data packet along a particular link, the corresponding link type entry in its *tag* is decremented by unity.

We have to ensure that the packets destined to the nodes in partition I, always travel through T_{+1} or T_{-1} and packets destined to the nodes in partition S always travel through T_{+s} or T_{-s}. We do this by sending every packet destined for a node in partition I along the links of type $+1$ or -1 first, until the corresponding link type's entry in the tag of the packet becomes zero. Then only, we transmit the packet using $+s$ or $-s$ links. The reverse procedure is followed for the packets destined to the nodes in partition S. The algorithm for single node scatter can now be described as follows.

Algorithm S :

```
Parbegin
    node 0 :
            begin
        for t = 0, 1, ..., (n − 1)/4 − 1 do
        begin
            take the data packets destined to the nodes in C_t;
            generate the tags for each of these packets;
            decide the appropriate links along which these packets are to be
            transmitted, depending on the partition type and the non- zero
            link type entries of the tags;
            update the entries of the tag of each packet accordingly;
            transmit the packets;
        end
end
node i(i > 0) :
        begin
            if packets are received then
            begin
                check the tag of the last received packets;
                if all link type entries of the tag of a packet are zero, then
                consume the packet (the packet has reached its destination);
                else
                begin
                    update the entries of the tag of each packet and transmit these
                    packets along the decided link;
                end
            end
        end;
Parend ;
```

6.5 Conclusion

In this chapter we have given algorithms for multiple node broadcast and single node scatter in a distributed loop network. Our algorithms are based on the assumptions of multiple link availability (MLA) and full duplex communication along each link. Also, to complete the multiple node broadcast in time $\lceil (n-1)/4 \rceil$, we need proper synchronization among all the nodes of the network. There is no overhead associated with the running of this algorithm except an $O(n)$ storage requirement at each node. The required information for packet broadcasting will be computed once for all and will be stored in all the nodes of the network. The single node scattering algorithm also runs in time $\lceil (n-1)/4 \rceil$ and hence is optimal with respect to time. The required information for packet scattering is computed once for all and stored at the source node. The resulting overhead is an $O(n)$ storage at the source node. The total exchange or multiple node scatter is another important communication problem. Further work is being carried out to find if their exists a time optimal algorithm for this problem in a distributed loop network.

Dense Odd Degree Graphs

7.1 Introduction

Construction of dense regular graphs, i.e., regular graphs of a given order with small diameters constitutes an important topic of recent research. Examples of some well known dense regular graphs are Moebius graph [LS82], de Bruijn graphs [NGB46], Kautz graphs [WHK69], etc. Moebius graph graph is a trivalent structure with 2^n nodes and diameter $\leq \lfloor \frac{3n}{2} \rfloor$. A de Bruijn graph with 2^n nodes has degree 4 and diameter n. For the same degree and diameter the total number of nodes in the Kautz graph is 3.2^{n-1}. Generalization of the ideas in the construction of these graphs have also been done, leading to higher degree graphs. Thus a radix-k de Bruijn graph has k^n nodes with degree $2k$ and diameter n. Likewise, the Kautz graph with the same degree and diameter has $k^n + k^{n-1}$ nodes.

There are basically two approaches to construct such dense graphs. One is, given the degree and diameter, find a graph with the number of nodes as close to the Moore bound [BI73]. Another is, given the total number of nodes and degree, find a graph with as small a diameter as possible. Our work is of the first kind. In this chapter, we propose a new family of graphs of maximum degree 5, having $N = 4^n$ nodes, $(n \geq 2)$ and diameter $\leq \lfloor \frac{3n}{2} \rfloor + 1 = \lfloor \frac{3}{4} \log_2 N \rfloor + 1$. For even n, these graphs will be regular of order 5. For odd n, all but 4 vertices will have

degree 5 and the remaining 4 vertices will have degree 4. That is, these graphs are regular (for even n) or almost regular (for odd n) dense ones. We then generalize the underlying ideas of the construction process to define families of such *almost regular* odd degree graphs of maximum degree $2j+1$, $(j > 2)$ with $N = (2j)^n$ nodes, $(n \geq 2)$ and diameter $\leq \lfloor \frac{3n}{2} \rfloor + 1 = \lfloor \frac{3}{4} \log_j N \rfloor$. A heuristic algorithm for point to point routing in such *almost* 5-regular graphs has also been suggested. Simulation of these algorithm shows that the length of the path computed by this algorithm does not exceed the shortest distance between them by 1.03, 1.27, and 1.51 on an average for $n = 256$, 1024 and 4096 respectively.

For the graphs of maximum degree 5, an algorithm for single node broadcast has been presented. This algorithm requires $3n$ time for a graph with 4^n nodes.

Implementation of various algorithms for real-life applications, e.g., matrix transpose, matrix multiplication, finding the sum/average/maximum/minimum of a set of data elements and ASCEND/DESCEND types of algorithms [PV81] have also been discussed.

7.2 The Proposed graph and Some of its Properties

We define the proposed graph $G = (V, E)$ with V as the vertex set and E as the set of undirected edges in the following way :

1) Consider a string $v_1 v_2 \cdots v_i \cdots v_n$, $v_i \in \{0, 1, 2, 3\}$, $1 \leq i \leq n$. Every distinct string of the form $v_1 v_2 \cdots v_i \cdots v_n$ has a corresponding distinct vertex $v \in V$. Clearly, $|V| = 4^n$.

2) For $v \in V$ and $k \in \{1, 2\}$, we define functions $f_k(v)$, $\varphi_k(v)$ and $g(v)$ as follows :

$$f_k(v_1 v_2 \cdots v_n) = u_1 u_2 \cdots u_n, \text{ where } u_i = \begin{cases} v_{i+1}, & \text{for } 1 \leq i \leq n-1 \\ (v_1 + k) \bmod 4, & \text{for } i = n \end{cases}$$

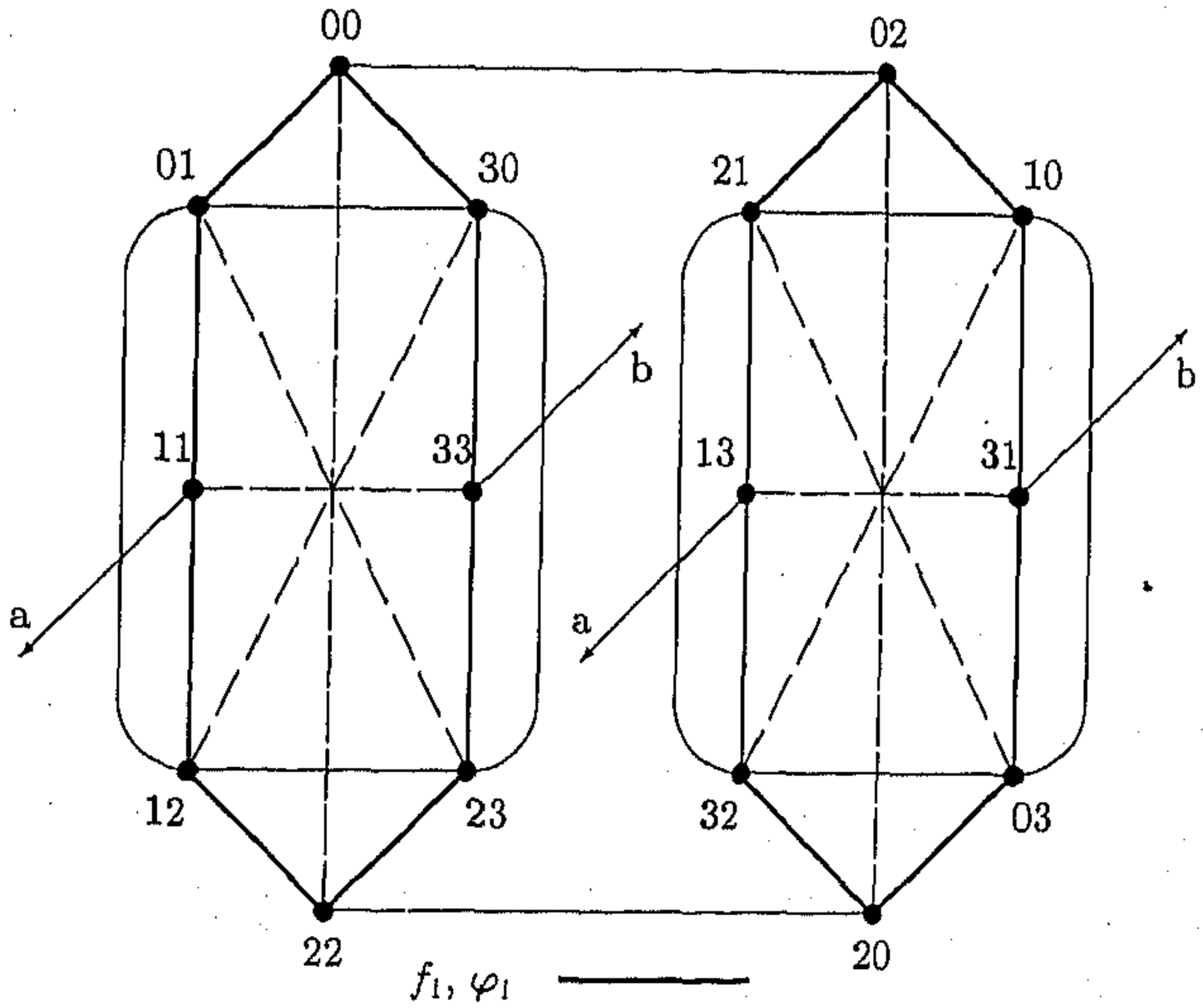


Figure 7.1: A graph with 16 nodes

$$\varphi_k(v_1v_2\cdots v_n) = u_1u_2\cdots u_n, \text{ where } u_i = \begin{cases} v_{i-1} & \text{for } 2 \le i \le n \\ (v_n - k) \bmod 4 & \text{for } i = 1 \end{cases}$$

$$g(v_1v_2\cdots v_n) = u_1u_2\cdots u_n, \text{ where } u_i = \begin{cases} v_i & \text{for } 1 \le i \le n-2 \\ (v_i + 2) \bmod 4 & \text{for } i = n-1, n \end{cases}$$

Note that $f_k^{-1} = \varphi_k$ and $g^{-1} = g$.

Now the following steps complete the definition of the graph $G(V, E)$:

i) Connect an edge from u to v if and only if $f_k(u) = v$ or $\varphi_k(u) = v$ or $g(u) = v$.

ii) remove the directions of all the edges.

It is clear that the maximum degree of a vertex $v \in V$ is 5. A graph with 16 nodes is shown in figure 7.1.

In what follows, $v_1v_2\cdots v_n$ will denote the string representation of a vertex v. Any

digit in a string will be regarded as a modulo 4 number, and the corresponding arithmetic involving such digits will also be in modulo 4, unless otherwise mentioned.

7.2.1 Degree Distribution

Let $e = \{f_1, f_2, \varphi_1, \varphi_2, g\}$. The degree of a node u can be less than the maximum value of 5, if for some $h_1, h_2 \in e$, and $h_1 \neq h_2$, we have

$$h_1(u) = h_2(u) \qquad (A)$$

Note that $f_1(u)$ can never be equal to $f_2(u)$. Similarly $\varphi_1(u) \neq \varphi_2(u)$. Suppose (A) holds for $h_1 = f_i$, $i \in \{1,2\}$ and $h_2 = g$. For $u = u_1u_2 \cdots u_n$, $f_i(u) = u_2u_3 \cdots u_n(u_1 + i)$ and $g(u) = u_1u_2 \cdots u_{n-2}(u_{n-1} + 2)(u_n + 2)$. Then we get the following relations :

i) $u_1 = u_2 = u_3 = \ldots = u_{n-2} = u_{n-1}$

ii) $u_{n-1} + 2 = u_n$

iii) $u_n + 2 = u_1 + i$

But $u_n + 2 = (u_{n-1} + 2) + 2 = u_{n-1} \pmod 4$, i.e., $u_1 + i = u_1$. This implies $i = 0 \pmod 4$, which is a contradiction. Hence, $f_i(u) \neq g(u)$. Similarly, $\varphi_i(u) \neq g(u)$.

On the other hand, if $f_i(u) = \varphi_j(u)$, then

$$u_2u_3 \cdots u_n(u_1 + i) = (u_n - j)u_1u_2 \cdots u_{n-2}u_{n-1}.$$

So, we get the following relations :

1. When n is an even number say $2m$, then $u_1 = u_3 = \cdots = u_{2m-1} = u_1 + i$, which is a contradiction. That is, $f_i(u) \neq \varphi_j(u)$ for even n.

2. When n is an odd number say $2m - 1$, then

i) $u_1 = u_3 = \cdots = u_{2m-1}$

ii) $u_2 = u_4 = \cdots = u_{2m-2} = u_1 + i$

iii) $u_2 = u_{2m-1} - j = u_1 - j$

Hence, $u_1 + i = u_1 - j$, i.e., $(i + j) \bmod 4 = 0$. This is possible only if $i = j = 2$. So $f_2(u) = \varphi_2(u)$ only when n is odd and u is of the form $u_1(u_1 + 2)u_1(u_1 + 2)u_1 \cdots (u_1 + 2)u_1$, where $u_1 \in \{0, 1, 2, 3\}$.

Now we get the following lemma :

Lemma 7.1 *In $G(V, E)$, with $|V| = 4^n$, i) all nodes will be of degree 5 when n is even, and ii) only 4 nodes will be of degree 4 and the rest of degree 5, when n is odd.*

7.2.2 Number of Edges

When n is even the total number of edges $|E|$ is equal to $\frac{4^n \times 5}{2} = 10 \times 4^{n-1}$. For odd values of n $|E|$ is equal to $\frac{(4n-4)\times 5 + 4\times 4}{2} = 10 \times 4^{n-1} - 2$.

7.3 Path length and Diameter

We now consider a path from a source node $s = s_1 s_2 \cdots s_n$ to a destination node $t = t_1 t_2 \cdots t_n$. Since the links of the graph $G(V, E)$ are characterized by the functions f_i, φ_i and g, a path of length k can be represented by a string $p_1 p_2 \cdots p_k$, where $p_i \in \{f_1, f_2, \varphi_1, \varphi_2, g\}$. First we consider the paths which do not involve φ_1 or φ_2. Also, as $g^2 = \mathrm{Id}$, the identity function, we would not apply two g's in succession. Before investigating the nature of such paths we define the following :

Definition 7.1 *For two digits $a, b \in \{0, 1, 2, 3\}$, we define an operation $\bullet$ as follows :*

$$b \bullet a = \begin{cases} 0, & \text{if } (b - a) \bmod 4 = 1 \text{ or } 2 \\ 1, & \text{if } (b - a) \bmod 4 = 3 \text{ or } 0 \end{cases}$$

We consider a path P, starting from the node s and of the form $h_1 g_1 h_2 g_2 \cdots h_i g_i \cdots h_n g_n$, where $h_i \in \{f_1, f_2\}$ and $g_i = \mathrm{Id}$ or g. Let the path P lead to a node

$u = u_1 u_2 \cdots u_n$. Corresponding to the path P we define two strings $c = c_1 c_2 \cdots c_n$, $c_i \in \{1, 2\}$ and $x = x_1 x_2 \cdots x_n$, $x_i \in \{0, 1\}$ as follows :

i) If $h_i = f_k$, $k \in \{1, 2\}$, then $c_i = k$

ii) If $g_i = \mathrm{Id}$, then $x_i = 0$ and if $g_i = g$ then $x_i = 1$.

Now we give the following lemma :

Lemma 7.2 *For $1 \leq i \leq n$, the following relation holds :*

$$u_i = \begin{cases} (s_i + c_i), & \text{if } x_i \oplus x_{i+1} = 0 \\ (s_i + c_i) + 2, & \text{if } x_i \oplus x_{i+1} = 1 \end{cases}$$

where $x_{n+1} = x_1$.

Proof : The function $h_i (= f_k)$ operated on a string modifies the first digit, say a, of the string to $a + k$, i.e., $a + c_i$, and puts it in the nth digit position of the resulting string. Since the total number of h_i's in P is same as the total number of digits in s, it follows that $u_i = s_i + c_i$ or $s_i + c_i + 2$. Addition of the term c_i is due to the function f_i and absence or presence of the term 2 is determined by the values of x_i and x_{i+1}, as each g function involves addition by 2. In fact, there can be 4 possible cases.

i) $x_i = 0$, $x_{i+1} = 0$. No g function is involved. Hence, $u_i = s_i + c_i$.

ii) $x_i = 0$, $x_{i+1} = 1$. Only one g function is involved and hence $x_i = 1$, $u_i = s_i + c_i + 2$.

iii) $x_i = 1$, $x_{i+1} = 0$. $u_i = s_i + c_i + 2$.

iv) $x_i = 1$, $x_{i+1} = 1$. Two g functions are involved and hence $u_i = s_i + c_i + 2 + 2 = s_i + c_i$.

Hence, the proof. □

Note that $b \bullet a$ can take on only binary values, i.e., 0 or 1. From the result of Lemma 7.2 we arrive at the following results.

Lemma 7.3 *For $1 \leq i \leq n$, $x_i \oplus x_{i+1} = u_i \bullet s_i$, where $x_{n+1} = x_1$.*

Proof : If $x_i \oplus x_{i+1} = 0$, then $u_i - s_i = c_i$ (from Lemma 7.2). Note that c_i can be either 1 or 2. Hence, $u_i \bullet s_i = 0$ in this case.

Again, when $x_i \oplus x_{i+1} = 1$, $u_i - s_i = c_i + 2$. In this case $u_i \bullet s_i = 1$. □

Lemma 7.4 $x_1 = (\sum (u_i \bullet s_i)) \oplus x_1$, *where* $\sum$ *stands for modulo 2 sum over* $i = 1, 2 \cdots, n$.

Proof : From Lemma 7.3, $x_i \oplus x_{i+1} = u_i \bullet s_i$ for $1 \leq i \leq n$. We can write this equation as $x_{i+1} = (u_i \bullet s_i) \oplus x_i$. Now we get a set of n equations as

$$x_2 = (u_1 \bullet s_1) \oplus x_1 \tag{1}$$
$$x_3 = (u_2 \bullet s_2) \oplus x_2 \tag{2}$$
$$x_4 = (u_3 \bullet s_3) \oplus x_3 \tag{3}$$
$$\vdots$$
$$x_n = (u_{n-1} \bullet s_{n-1}) \oplus x_{n-1} \tag{$n-1$}$$
$$x_1 = (u_n \bullet s_n) \oplus x_n \tag{n}$$

Solving these equations, we get $x_1 = (\sum (u_i \bullet s_i)) \oplus x_1$. □

Corollary : It follows that $\sum (u_i \bullet s_i) = 0$.

Lemma 7.5 *For any pair of nodes* (s, t) *such that* $\sum t_i \bullet s_i = 0$, *there exists a path* P *between* s *and* t, *where* $P = h_1 g_1 h_2 g_2 \cdots h_i g_i \cdots h_n g_n$, $h_i \in \{f_1, f_2\}$.

Proof : The proof involves finding the strings c and x which will uniquely characterize the path P. By Lemma 7.2, $t_i = s_i + c_i$ or $s_i + c_i + 2$. Hence, we find c_i by the following equation :

$$c_i = \begin{cases} 1, & \text{if } t_i - s_i = 1 \text{ or } 3 \\ 2, & \text{if } t_i - s_i = 2 \text{ or } 0 \end{cases}$$

To find the string x we use the result of Lemma 7.3, i.e., $x_i \oplus x_{i+1} = t_i \bullet s_i$. This equation must be satisfied for all i, which leads to the relation $x_1 = (\sum (t_i \bullet s_i)) \oplus x_1$.

For $\sum (t_i \bullet s_i) = 0$, we will always be able to find a string x satisfying this relation. Hence, we start with an arbitrary value of $x_1 \in \{0, 1\}$. Then successively applying the equation $x_i \oplus x_{i+1} = t_i \bullet s_i$, for $i = 1, 2, \cdots, n-1$, we will get x_2, x_3, $\cdots$, x_n.□

Let $l(P)$ be the length of a path P of the above form. Now we give the following theorem.

Theorem 7.1 *For any pair of nodes (s,t) such that $\sum (t_i \bullet s_i) = 0$, there exists a path P between s and t of the above form such that $l(P) \leq \lfloor \frac{3n}{2} \rfloor$.*

Proof : $l(P) = n + w(x)$, where $w(x)$ is the number of 1's in the binary string x. Now, we can get two strings for x satisfying the equations $x_i \oplus x_{i+1} = t_i \bullet s_i$, one with $x_1 = 0$, and the other with $x_1 = 1$. These two strings are complement to each other and hence for one of them the number of 1's is less than or equal to $\lfloor \frac{n}{2} \rfloor$. Hence, the proof. □

Now we consider a path P of the form $g_1 h_1 g_2 h_2 \cdots h_{n-1} g_n$ where h_i's and g_i's are as defined earlier. Let the path P from s lead to a node $u = u_1 u_2 \cdots u_n$. Corresponding to the path P we define two strings $c = c_1 c_2 \cdots c_{n-1}$ and $x = x_1 x_2 \cdots x_n$ as before. Now we state the following lemmas :

Lemma 7.6 *For $2 \leq i \leq n$,* $u_i = \begin{cases} (s_{i-1} + c_{i-1}), & \text{if } x_i \oplus x_{i+1} = 0 \\ (s_{i-1} + c_{i-1}) + 2, & \text{if } x_i \oplus x_{i+1} = 1 \end{cases}$

Also, $u_1 = \begin{cases} s_n, & \text{if } x_1 \oplus x_2 = 0 \\ s_n + 2, & \text{if } x_1 \oplus x_2 = 1 \end{cases}$

Proof : The function $h_i (= f_k)$ operated on a string modifies the first digit, say a, of the string to $a + k$, i.e., $a + c_i$, and puts it in the nth digit position of the resulting string. Since the total number of f_i's in P is one less than the total number of digits in s, the functions f_i's operate only on the digits s_1 to s_{n-1} and only the digits u_2 to u_n are affected by f_i's. It follows that either $u_i = s_{i-1} + c_{i-1}$ or $s_{i-1} + c_{i-1} + 2$, for $2 \leq i \leq n$. The absence or presence of the term 2 is determined by the value of $x_i \oplus x_{i+1}$ as explained in Lemma 7.2. Moreover, $u_1 = s_n$ or $s_n + 2$, depending on the value of $x_1 \oplus x_2$. Hence, the proof. □

Lemma 7.7 *For* $2 \leq i \leq n$, $x_i \oplus x_{i+1} = u_i \bullet s_{i-1}$, *where* $x_{n+1} = x_1$.

Proof : From Lemma 7.6, for $2 \leq i \leq n$, if $x_i \oplus x_{i+1} = 0$, then $u_i - s_{i-1} = c_{i-1}$. Note that c_{i-1} can be either 1 or 2. Hence in this case $u_i \bullet s_{i-1} = 0$.

Again, when $x_i \oplus x_{i+1} = 1$, then $u_i - s_{i-1} = c_{i-1} + 2$. In this case $u_i \bullet s_{i-1} = 1$. □

Lemma 7.8 *For any pair of nodes* (s,t) *there exists a path of the form* P *between* s *and* t', *where* $t' = t'_1 t_2 t_3 \cdots t_n$, t'_1 *being equal to* s_n *or* $s_n + 2$.

Proof : As in Lemma 7.5, we will first find the strings c and x which will uniquely characterize the path P. By Lemma 7.6, for $2 \leq i \leq n$, $t_i = s_{i-1} + c_{i-1}$ or $s_{i-1} + c_{i-1} + 2$. Hence, for $1 \leq i \leq n-1$, we find c_i by the following equation :

$$c_i = \begin{cases} 1, & \text{if } t_{i+1} - s_i = 1 \text{ or } 3 \\ 2, & \text{if } t_{i+1} - s_i = 2 \text{ or } 0 \end{cases}$$

To find the string x we use the result of Lemma 7.7, i.e., we must have $x_i \oplus x_{i+1} = t_i \bullet s_{i-1}$ for $i = 2, 3, \cdots, n$. Summing up all these equations we get $x_1 \oplus x_2 = \sum_{i=2}^{n} (t_i \bullet s_{i-1}) = S$, say. Hence, we start with some arbitrary value for x_2 and apply the equations successively for $i = 2, 3, \cdots, n$, to get x_3, x_4, $\cdots$, x_n and x_1. Here, the value of $S = x_1 \oplus x_2$ only determines the first digit of t', i.e., t'_1 (Lemma 7.6). If $S = 0$, then $t'_1 = s_n$ and if $S = 1$ then $t'_1 = s_n + 2$. □

Let $l(s,t')$ denote the length of the path P between s and t'.

Theorem 7.2 $l(s,t') \leq n - 1 + \lfloor \frac{n}{2} \rfloor$.

Proof : The proof is similar to that of Theorem 7.1. □

Theorem 7.3 *For any pair of nodes* (s,t), *there exists a path of length at most* $\lfloor \frac{3n}{2} \rfloor + 1$ *between* s *and* t.

Proof : By Theorem 7.2, $l(s,t') \leq \lfloor \frac{3n}{2} \rfloor - 1$. If $t'_1 = t_1$, then we are done. Otherwise, we consider the following cases.

Case i) $t'_1 = t_1 \pm 1$. If $t'_1 = t_1 + 1$, then we apply $f_2\varphi_1$ on t' to reach t. If $t'_1 = t_1 - 1$, then we apply $f_1\varphi_2$ on t' to reach t.

Case ii) $t'_1 = t_1 + 2$. Note that the value of t'_1 is determined by $S = \sum_{i=2}^{n} (t_i \bullet s_{i-1})$. If we could complement S by some means, then the value of t'_1 could be changed from $t_1 + 2$ to t_1. Now consider a node $t_1 t_2 \cdots t_{n-1} t'_n$ such that $t'_n \bullet s_{n-1} = (t_n \bullet s_{n-1}) \oplus 1$. Such a t'_n can always be found out by setting

$$t'_n = \begin{cases} t_n - 1, & \text{for } (t_n - s_{n-1}) = 1 \text{ or } 3 \\ t_n + 1, & \text{for } (t_n - s_{n-1}) = 0 \text{ or } 2 \end{cases}$$

Thus, starting from s, we can always reach the node $t_1 t_2 \cdots t_{n-1} t'_n$ by a path of the form $g_1 h_1 g_2 \cdots g_{n-1} h_{n-1} g_n$ having length at most $\lfloor \frac{3n}{2} \rfloor - 1$. Applying now $f_1\varphi_2$ (if $t'_n = t_n + 1$) or $f_2\varphi_1$ (if $t'_n = t_n - 1$), we can reach $t = t_1 t_2 \cdots t_n$. Hence, the proof.□

7.3.1 Comparison with the Moore Bound

According to the Moore bound [BI73], for a graph of maximum degree d and diameter D, the total number of nodes $N \leq (d(d-1)^D - 2)/(d-2)$, i.e., $D \geq \log_{d-1} N$ (approximately). For $d = 5$, we get $D \geq \log_4(\frac{3N+2}{5}) \approx 0.5 \log N = D_{\min}$ (say). In our graph the diameter is close to $0.75 \log N$ which is roughly equal to 1.5 times $D_{\min}$. This may be compared with the diameter of other constant degree graphs, e.g., Moebius graph, Cube-Connected Cycle (CCC), De Bruijn graph, as in Table 7.1 where N is the number of nodes in each graph.

7.4 Routing Algorithm

7.4.1 Point-to-Point Communication

We can always find a path between a source node s and a destination node t, with a path length less than or equal to the diameter, by employing the method described in the earlier section. The routing algorithm in that case will be simple

Table 7.1: Comparison of diameters of 3 common graphs with Moore bound

Topology	Degree	Diameter (D) (approx.)	Moore bound ($D_{\min}$) (approx.)	$D/D_{\min}$ (approx.)
Moebius	3	$\frac{3}{2}\log_2 N$	$\log_2 N$	1.5
CCC	3	$\frac{5}{2}\log_2 N$	$log_2 N$	2.5
de Bruijn	4	$\log_2 N$	$\log_3 N$	1.6
Proposed graph	5	$\frac{3}{4}\log_2 N$	$\log_4 N$	1.5

and its complexity will be no more than that in a Moebius graph [LS82]. But there may exist a path of much shorter length between s and t than that obtained by this method. This is illustrated in the following example.

Example 7.1 Let $s = 01213$ and $t = 01012$. The method given in section 3 gives a path of length 7 as $s \rightarrow 12132 \rightarrow 21323 \rightarrow 21301 \rightarrow 13010 \rightarrow 30103 \rightarrow 01030 \rightarrow 01012$.

We may note that there exists a path of length only 2 between s and t given by $s \rightarrow 10121 \rightarrow 01012 = t$.

In the above example, a path of shorter length was possible because the first 3 digits of s match with the last 3 digits of t. Whenever such a matching exists we can exploit this to get a path of length shorter than that obtained by the method in section 7.3. Guided by this observation, we discuss below the basic ideas of a near-optimal routing technique.

Let j be the largest integer, $j \geq 1$, such that one of the following conditions hold

1) $s_{n-i+1} = t_{j-i+1}$, for all $i \in \{1, 2, \cdots, j\}$

2) $s_{j-i+1} = t_{n-i+1}$, for all $i \in \{1, 2, \cdots, j\}$

Case 1 : $s_{n-i+1} = t_{j-i+1}$, for all $i \in \{1, 2, \cdots, j\}$. Since j bits are already in position, we need only $n - j$ shifts. We consider a path P of the form $h_1 g_1 h_2 g_2 \cdots h_{n-j} g_{n-j}$,

where $h_i \in \{f_1, f_2\}$ and $g_i = \mathrm{Id}$ or g. The path P is uniquely characterized by the two strings $c = c_1 c_2 \cdots c_{n-j}$ and $x = x_1 x_2 \cdots x_{n-j}$ as before. For $1 \leq i \leq n-j$, we find c_i by the following equation

$$c_i = \begin{cases} 1, & \text{if } t_{i+j} - s_i = 1 \text{ or } 3 \\ 2, & \text{if } t_{i+j} - s_i = 2 \text{ or } 0 \end{cases}$$

Now to find the string x, we set $x_1 = 0$ and then by successively applying the equation $t_{i+j} \bullet s_i = x_i \oplus x_{i+1}$ for $1 \leq i \leq n-j-1$, we compute $x_2, x_3, \cdots, x_{n-j}$. Then the path P will lead to the destination node t, if $t_n \bullet s_{n-j} = x_{n-j}$. Otherwise, we will reach a node $t' = t_1 t_2 \cdots t'_n$, where $t'_n = t_n + 2$. Hence, we consider the following subcases.

Subcase 1a): $t_n \bullet s_{n-j} = x_{n-j} \oplus 1$. We will be able to reach the node t by suitably modifying P as done in the Case ii) of the proof of Theorem 7.3. Hence the path length l_1 is equal to $n - j + w(x) + 2$.

There is also an alternative path from s to t. In finding the upper bound on the diameter, we had taken another string x with $x_1 = 1$ and it was obtained by complementing the bits of x. But here we cannot take such a string, as that will also change the bit t_n. Instead, we employ a different method. Let m, $(m > 1)$ be the first position from the left at which a 1 occurred in the above string x, i.e., $x_1 = x_2 = \cdots = x_{m-1} = 0$ and $x_m = 1$. This implies that, if $h_{m-1} = f_1$, then $t_{j+m-1} = (s_{m-1} + 3)$. We now modify the different string x as follows :
we replace h_{m-1} by $f_2 \varphi_1 f_2$, and set x_m to 0 so that t_{j+m-1} is again equal to $(s_{m-1}+3)$. Similarly if $h_{m-1} = f_2$, then $t_{j+m-1} = s_{m-1}$. We would replace f_{m-1} by $f_1 \varphi_2 f_1$, and set x_m to 0, so that $t_{j+m-1} = a_{m-1}$ again.

Thus, x_m is now 0, whereas earlier it was 1. Keeping $h_m, h_{m+1}, \cdots, h_{n-j}$ unaltered, we can now get a different string for x as $x_1 x_2 \cdots x_{m-1} \overline{x_m}\ \overline{x_{m+1}} \cdots \overline{x_{n-j}}$. Moreover, we can directly reach the vertex t by this process and do not need the extra two steps at the end as it was necessary corresponding to the earlier string x.

Number of 1's in this new string for x is equal to $(n-j-m) - w(x) + 1$. Hence, the new path length l_2 is equal to $n - j + 2 + n - j - m - w(x) + 1$. The minimum

of l_1 and l_2 is less than or equal to $(l_1 + l_2)/2 = (3(n - j + 1) - m)/2 + 1 = (3n)/2 + 1 - (3j + m - 3)/2$ with $j \geq 1$ and $m > 1$. Hence, the path length obtained is always within the bound given for the diameter and *decreases with increase in the value of j*. Also, $l_2 < l_1$ if,

$$n - j - m - w(x) + 1 < w(x),$$

i.e., $$w(x) > (n - j - m + 1)/2.$$

Subcase 1b): $t_n \bullet s_{n-j} = x_{n-j}$. Here, the path length $l_1 = n - j + w(x)$. An alternative path can also be obtained as in Subcase 1a) and the complementary string can also be similarly generated. But in this case, we will need extra 2 steps at the end to reach t. So the new path length $l_2 = n - j + 2 + (n - j - m) - w(x) + 3$. The minimum of l_1 and l_2 is again less than or equal to $(l_1 + l_2)/2 = (3(n - j + 1) - m)/2 + 1$. Also, $l_2 < l_1$ if,

$$(n - j - m) - w(x) + 5 < w(x),$$

i.e., $$w(x) > (n - j - m + 5)/2.$$

Case 2 : Routing is similar to that in Case 1 if we start from t to reach s.

Example 7.2 Let $s = 00000$ and $t = 01011$.

By following the method given in section 7.3, we find a path of length 8 as

$s \rightarrow 00001 \rightarrow 00012 \rightarrow 00121 \rightarrow 00103 \rightarrow 01032 \rightarrow 01010 \rightarrow 30101 \rightarrow 01011 = t$.

Here $j = 1$, as $s_5 = t_1$ and Case 1 is applicable. By following the routing technique given above, we find a path of length 6 as $s \rightarrow 00001 \rightarrow 00011 \rightarrow 30001 \rightarrow 00010 \rightarrow 00101 \rightarrow 01011 = t$.

Remark : The path of length 2 between s and t of example 1 also follows from the near-optimal routing technique described above.

Although the path obtained by our routing algorithm may not be the shortest, its length is definitely less than or equal to the bound on diameter of the graph. Also, our routing algorithm requires $O(\log N)$ time for a graph with N nodes. Further, we have computed the shortest paths between every pair of nodes and found that

the number of extra links needed by following our routing algorithm, is very small on the average. The results of these computations are given in the following table.

Table 7.2: Average difference from the shortest path lengths

Number of nodes	Average number of extra links
16	0.4417
64	0.7961
256	1.0300
1024	1.2665
4096	1.5138

7.4.2 Single Node Broadcast

To explain the broadcast algorithm, we use the following notations :
$a_1a_2\cdots a_{n-1}*$ will represent 4 nodes, where '$*$' can take all possible 4 values 0, 1, 2 and 3. Similarly, by using more than one '$*$' in different positions, a larger set of nodes can be represented. If x is the number of '$*$'s in a symbolic notation, then the number of nodes represented will be equal to 4^x. Also, m successive occurrences of the symbol '$*$' will be represented by $*^m$.

To broadcast a message from a source node $a_1a_2\cdots a_n$, we would traverse the nodes in the following sequence :

$$a_1a_2\cdots a_n \rightarrow a_2a_3\cdots a_n* \rightarrow a_3a_4\cdots a_n** \rightarrow \cdots \rightarrow *^n$$

It is to be noted that a single arrow '$\rightarrow$' in the above sequence may involve more than one communication step.

To transfer a message from $a_1a_2\cdots a_n$ to $a_2a_3\cdots a_n*$, we note that,

$$f_1(a_1a_2\cdots a_n) = a_2a_3\cdots a_n(a_1+1)$$
$$f_2(a_1a_2\cdots a_n) = a_2a_3\cdots a_n(a_1+2)$$
$$f_1\varphi_2 f_1(a_1a_2\cdots a_n) = a_2a_3\cdots a_na_1$$
$$f_2\varphi_1 f_2(a_1a_2\cdots a_n) = a_2a_3\cdots a_n(a_1+3)$$

Hence, it is clear that 3 communication steps will suffice to perform the operations involved in each of the above broadcast steps indicated by an arrow '$\rightarrow$'. The total number of steps required is equal to $3n$.

To execute the broadcast algorithm in a distributed way, a tag is attached with the message. The tag has a *link* field which can take the values 1 or 2 and a *count* field which is initially set to $3n$ and decremented by one before each communication step.

The following procedure is executed by the source node as well as every other node on receiving a message, where x is the current node.

```
Procedure Broadcast(x : node)
begin
    while count > 0 do
    begin
        case (count mod 3) of
        0 : for i = 1,2 do
              set link ← 3 − i; count ← count − 1; transmit to f_i(x);
        2 : set i ← link, link ← 3 − link; count ← count − 1; transmit to φ_i(x);
        1 : count ← count − 1; i ← link; transmit to f_i(x);
        endcase;
    end;
end;
```

Remark 7.1 We could have as well broadcast in a reverse way following the sequence as

$$a_1a_2\cdots a_n \rightarrow *a_1a_2a_3\cdots a_{n-1} \rightarrow **a_1a_2\cdots a_{n-2} \rightarrow \cdots \rightarrow *^n .$$

For this we just need to replace f_i by φ_i and vice versa in the above procedure.

7.5 Implementation of Algorithms

In all discussions that follow, we would represent a processing node interchangeably by its label $a_1a_2\cdots a_n$ or by a unique integer i in the range 0 to $4^n - 1$, where $i = \sum a_k 4^{n-k}$. Actually, the integer i corresponds to a linear numbering of the nodes with proper weights of the digits a_i's.

7.5.1 Matrix Transpose

To transpose a $4^m \times 4^m$ matrix A we consider a network of 4^{2m} nodes. Initially the element $A(i,j)$ of the matrix is stored in the node $4^m \times i + j$. After transposing, $A(i,j)$ should be in the node $4^m \times j + i$. In other words, we need to move the element in the node $a_1a_2\cdots a_ma_{m+1}a_{m+2}\cdots a_{2m}$ to the node $a_{m+1}a_{m+2}\cdots a_{2m}a_1a_2\cdots a_m$. We consider the following two cases :

Case 1 : m is even.

We note that $f_2f_2g(a_1a_2\cdots a_{2m}) = a_3a_4\cdots a_{2m}a_1a_2$. Thus, moving the data from a node $a_1a_2\cdots a_{2m}$ to the node $a_2a_3\cdots a_{2m}a_1a_2$ will involve 3 communication steps. Applying this sequence of 3 operations successively $m/2$ times, the whole matrix can be transposed. The total number of steps required will be $3(m/2)$.

Case 2 : m is odd. Let $m = 2x + 1$.

By applying the sequence of 3 operations (as in Case 1) successively for $\lfloor \frac{m}{2} \rfloor$ times, the data at the node $a_1a_2\cdots a_{2m}$ can be moved to the node $a_ma_{m+1}\cdots a_{2m}a_1a_2\cdots a_{m-1}$.

As $f_1\varphi_2f_1(a_ma_{m+1}\cdots a_{2m}a_1a_2\cdots a_{m-1}) = a_{m+1}a_{m+2}\cdots a_{2m}a_1a_2\cdots a_m$, 3 more communication steps will suffice to move the data to $a_{m+1}a_{m+2}\cdots a_{2m}a_1a_2\cdots a_m$. The total number of communication steps required in this case is equal to $3\lfloor \frac{m}{2} \rfloor + 3 = 3\lceil \frac{m}{2} \rceil$.

We describe the procedure that will be executed by a node x for transposing the matrix only for the case when m is even. The procedure can easily be extended for the case when m is odd. A tag is attached to the data which has only a *count* field. Initially the *count* field is set to $3(m/2)$.

Procedure Transpose(x : node)
begin
  while *count* > 0 **do**
  begin
    if (*count* mod $3 \neq 1$) **then**
      set *count* $\leftarrow$ *count* $- 1$; transmit data to the node f_2(x);
    else
      set *count* $\leftarrow$ *count* $- 1$; transmit data to the node g(x);
  end;
end;

7.5.2 Maximum/Minimum/Sum/Average

We first consider only the summing up of 4^n elements, where each node contains exactly one element. The maximum / minimum / average can be computed in the sane way. We plan to store the final sum in the node $4^n - 1$, i.e. $33\cdots3$. We first show below that we can sum 4 elements initially stored in the nodes $*a_2a_3\cdots a_n$ and put the result in $a_2a_3\cdots a_n3$, in 3 steps.

Note that,

$$\left.\begin{aligned} f_2(0a_2a_3\cdots a_n) &= a_2a_3\cdots a_n2 \\ f_1(1a_2a_3\cdots a_n) &= a_2a_3\cdots a_n2 \end{aligned}\right\} \qquad (A)$$

Hence, after one communication step, the data stored at the nodes $0a_2a_3\cdots a_n$ and $1a_2a_3\cdots a_n$ will move to the node $a_2a_3\cdots a_n2$. Now we can compute their sum at $a_2a_3\cdots a_n2$.

Also note that,

$$\left.\begin{aligned} f_2(2a_2a_3\cdots a_n) &= a_2a_3\cdots a_n0 \\ f_1(3a_2a_3\cdots a_n) &= a_2a_3\cdots a_n0 \end{aligned}\right\} \qquad (B)$$

Hence, the other two data at $2a_2a_3\cdots a_n$ and $3a_2a_3\cdots a_n$ can be summed at $a_2a_3\cdots a_n0$.

These two sets of operations (A) and (B) can be executed in parallel.

Also,

$$\varphi_1(a_2a_3\cdots a_n2) = 1a_2a_3\cdots a_n$$
$$\varphi_2(a_2a_3\cdots a_n0) = 2a_2a_3\cdots a_n$$

Hence, by another communication step, the partial sums can be moved to the nodes $1a_2a_3\cdots a_n$ and $2a_2a_3\cdots a_n$ respectively.

Now, applying the steps as in (A), we can obtain the sum in $a_2a_3\cdots a_n3$.

Summing of 4^n elements involves a sequence of n such operations, which can be represented as follows :

$$*^n \rightarrow *^{n-1}3 \rightarrow \cdots \rightarrow *33\cdots3 \rightarrow 33\cdots3$$

We can also proceed in the reverse way as

$$*^n \rightarrow 3*^{n-1} \rightarrow \cdots \rightarrow 33\cdots3* \rightarrow 33\cdots3$$

Hence, the sum of 4^n elements can be computed in $3n$ steps of which $2n$ steps involve both communication and computation and the other n steps involve communication only.

7.5.3 Matrix Multiplication

Multiplication of matrices on the proposed graph can be done in almost the same way as in a hypercube [A89]. To multiply two matrices A and B of size $4^m \times 4^m$, we take a graph consisting of 4^{3m} nodes. We would represent the processor at the node $a_1a_2\cdots a_mb_1b_2\cdots b_mc_1c_2\cdots c_m$ by $P(x,y,z)$ where $x = \sum a_i4^{m-i}$, $y =$

$\sum b_i 4^{m-i}$, and $z = \sum c_i 4^{m-i}$. The notation $P(*, y, z)$ is used to denote the set of 4^m processors $P(d, y, z)$, where d assumes all values from 0 to $4^m - 1$. The outline of the multiplication algorithm is given below.

Initially, the elements of the matrix $A[j, k]$ and $B[j, k]$ are stored at the processor $P(0, j, k)$. The elements $C[j, k]$ of the final product matrix $C = AB$ will be stored in the processor $P(r, k, j)$, where $r = 4^m - 1$.

The first step involves a broadcast of $A[j, k]$ and $B[j, k]$ from $P(0, j, k)$ to $P(j, k, *)$ for all j, k, which takes $3m$ steps (each step involves sending both $A[j, k]$ and $B[j, k]$). Now, $P(j, i, i)$ contains $A[j, i]$ and $P(i, k, i)$ contains $B[i, k]$.

Next, we broadcast (running the procedure broadcast in reverse way) $A[j, i]$ from $P(j, i, i)$ to $P(*, j, i)$ and $B[i, k]$ from $P(i, k, i)$ to $P(*, i, k)$. Hence the processor $P(k, j, i)$ will contain $A[j, i]$ for all k and the processor $P(j, i, k)$ will contain $B[i, k]$ for all j. These two broadcast steps can also be done in $3m$ steps.

Next, the data $B[i, k]$ is moved from $P(j, i, k)$ to $P(k, j, i)$. This involves a shuffle of m bits and by following the method as for matrix transposition, we can do this in time $3\lceil \frac{m}{2} \rceil$.

Now the processor $P(k, j, i)$ contains $A[j, i]$ and $B[i, k]$. Hence, the product of $A[j, i]$ and $B[i, k]$ can be computed in parallel for all values of i, j and k. To compute the sum of these products over all i, we require another $3m$ steps and the sums will be available at $P(r, k, j)$, i.e., $P(r, k, j)$ will contain $C[j, k] = \sum A[j, i] \times B[i, k]$.

Hence the multiplication can be done in $O(m) = O(\log N)$ steps, where $N \times N$ is the size of each of the matrices A and B.

7.5.4 ASCEND/DESCEND Algorithms

A class of parallel algorithms, called ASCEND and DESCEND types of algorithm [PV81] have many areas of application such as FFT, bitonic sorting, etc. Here we will describe the implementation of DESCEND type of algorithms only. We

consider N data elements $d[0], d[1], \cdots d[N-1]$ stored, respectively at storage locations $x[0], x[1], \cdots x[N-1]$. We denote a basic operation in the DESCEND type of algorithms by $\mathbf{Oper}(m, i, x[m], x[m+2^i])$, where the operation involves the two data elements at locations $x[m]$ and $x[m+2^i]$ whose binary representations differ in the ith bit. $\mathbf{Oper}(m, i, x[m], x[m+2^i])$ computes two values R_0 and R_1, which are stored at $x[m]$ and $x[m+2^i]$ respectively. Let $N = 4^n = 2^2n$. Then the DESCEND algorithm runs as follows :

for $i = q-1$ to 0 **do**

  for all m, $0 \leq m < N$ **do** in parallel

    if $\text{bit}_i(m) = 0$ **then** $\mathbf{oper}(m, i, x[m], x[m+2^i])$;

Initially, we store the data $d[k]$ at the node $a_1 a_2 \cdots a_n$ where a_i's are related to the the binary representation $b_1 b_2 \cdots b_{2n}$ be of k as follows :

$$a_i = 2b_{2i} + b_{2i-1}, \quad 1 \leq i \leq n.$$

We now give a brief overview of implementing the DESCEND algorithm by our method. We denote by $w[a_1 a_2 \cdots a_n]$ the data stored at the node $a_1 a_2 \cdots a_n$. Note that,

$$f_2(0*^{n-1}) = *^{n-1}2$$
$$f_1(1*^{n-1}) = *^{n-1}2$$

Hence, after one communication step the data stored at $0*^{n-1}$ and $1*^{n-1}$ can be moved to $*^{n-1}2$. Then the operation **Oper** can be performed on them. Another communication step will bring the computed results R_0 and R_1 to $0*^{n-1}$ and $1*^{n-1}$ respectively. Similarly the data stored at $2*^{n-1}$ and $3*^{n-1}$ are operated upon and the results R_0 and R_1 are moved to $3*^{n-1}$ and $2*^{n-1}$ respectively. These two steps complete the operation **Oper** corresponding to the bit b_1. The next step of the DESCEND algorithm corresponding to the bit b_2 will involve data pairs in the nodes $(0*^{n-1}, 3*^{n-1})$ and $(1*^{n-1}, 2*^{n-1})$ respectively.

We note that,

$$f_2(1*^{n-1}) = *^{n-1}3$$
$$f_1(2*^{n-1}) = *^{n-1}3$$

Thus, the data in $1*^{n-1}$ and $2*n-1$ can be operated upon and stored at $*^{n-1}3$. Now one of the computed results, i.e., R_0 is moved to $*^{n-1}2$ in 2 communication steps. As,

$$\varphi_2 f_1(*^{n-1}3) = *^{n-1}2$$

Simultaneously with these steps the data stored at $3*^{n-1}$ and $0*^{n-1}$ are brought $*^{n-1}1$ to be operated upon, and the corresponding result R_0 is brought to $*^{n-1}1$.

Hence, after 5 steps, the operation of the DESCEND algorithm corresponding to the most significant two bits b_1 and b_2 of the binary representation of m can be completed. The above process is repeated n times for completion of the DESCEND algorithm. The steps are formally described as follows :

Algorithm DESCEND

Step 1 : $i := n$;

Step 2 : move data pair $(w[0*^{n-1}], w[1*^{n-1}])$ to $*^{n-1}2$

move data pair $(w[2*^{n-1}], w[3*^{n-1}])$ to $*^{n-1}0$

Perform **Oper** at $*^{n-1}2$ and $*^{n-1}0$

Step 3 : move (R_0, R_1) to $(0*^{n-1}, 1*^{n-1})$

move data pair (R_1, R_0) to $(2*^{n-1}, 3*^{n-1})$

Step 4 : move data pair $(w[1*^{n-1}], w[2*^{n-1}])$ to $*^{n-1}3$

move data pair $(w[0*^{n-1}], w[3*^{n-1}])$ to $*^{n-1}1$

Perform operations at $*^{n-1}3$ and $*^{n-1}1$

Step 5 : move R_0 at $*^{n-1}3$ to $*^{n-1}2$

move R_0 at $*^{n-1}1$ to $*^{n-1}0$

Step 6 : $i \leftarrow i-1$; if $i > 0$ then go to Step 1 else Stop.

The total time required is equal to $5n$ for 4^n data elements.

7.6 Extension to Higher Degree

We now propose a family of graphs G of maximum degree $2j+1, j \geq 2$ and having $(2j)^n$ nodes, $n \geq 2$. Each node v of the graph is represented by a distinct string

of length n, i.e., $v_1v_2\cdots v_n$, $v_i \in \{0, 1, \cdots, 2j-1\}$. For $v \in V$, we define functions $f_k(v)$, $\varphi_k(v)$, $(k \in \{1, 2, \cdots, j\})$ and $g(v)$ as follows :

$$f_k(v_1v_2\cdots v_n) = u_1u_2\cdots u_n, \text{ where } u_i = \begin{cases} v_{i+1}, & \text{for } 1 \le i \le n-1 \\ (v_1+k) \bmod 2j, & \text{for } i = n \end{cases}$$

$$\varphi_k(v_1v_2\cdots v_n) = u_1u_2\cdots u_n, \text{ where } u_i = \begin{cases} v_{i-1} & \text{for } 2 \le i \le n \\ (v_n-k) \bmod 2j, & \text{for } i = 1 \end{cases}$$

$$g(v_1v_2\cdots v_n) = u_1u_2\cdots u_n, \text{ where } u_i = \begin{cases} v_i & \text{for } 1 \le i \le n-2 \\ (v_i+j) \bmod 2j, & \text{for } i = n-1, n \end{cases}$$

The edge set E of the graph is defined as follows : $(u, v) \in E \Leftrightarrow f_k(u) = v$ or $\phi_k(u) = v$ or $g(u) = v$.

We consider a path from a source node $s = s_1s_2\cdots s_n$ to a destination node $t = t_1t_2\cdots t_n$. First we consider the paths which do not involve φ_1 or φ_2. Also, as $g^2 = \mathrm{Id}$, the identity function, we would not apply two g's in succession. Before investigating the nature of such paths we define the following :

Definition 7.2 *For two digits $a, b \in \{0, 1, \cdots, 2j-1\}$, we define an operation $\bullet$ as follows :*

$$b \bullet a = \begin{cases} 0, & \text{if } (b-a) \bmod 2j \in \{1, 2, \cdots, j\} \\ 1, & \text{if } (b-a) \bmod 2j \in \{0, j+1, j+2, \cdots, 2j-1\} \end{cases}$$

We consider a path P, starting from the node s and of the form $h_1g_1h_2g_2\cdots h_ig_i\cdots h_ng_n$, where $h_i = f_k$, $k \in \{1, 2, \cdots, j\}$ and $g_i = \mathrm{Id}$ or g. Let the path P lead to a node $u = u_1u_2\cdots u_n$. Corresponding to the path P we define two strings $c = c_1c_2\cdots c_n$, $c_i \in \{1, 2, \cdots, j\}$ and $x = x_1x_2\cdots x_n$, $x_i \in \{0, 1\}$ as follows :

i) If $h_i = f_k$, $k \in \{1, 2, \cdots, j\}$, then $c_i = k$

ii) If $g_i = \mathrm{Id}$, then $x_i = 0$ and if $g_i = g$ then $x_i = 1$.

Now we give the following lemma :

Lemma 7.9 *For $1 \le i \le n$, the following relation holds :*

$$u_i = \begin{cases} (s_i + c_i), & \text{if } x_i \oplus x_{i+1} = 0 \\ (s_i + c_i) + j, & \text{if } x_i \oplus x_{i+1} = 1 \end{cases}$$

where $x_{n+1} = x_1$.

Proof : The function $h_i(= f_k)$ operated on a string modifies the first digit, say a, of the string to $a + k$, i.e., $a + c_i$, and puts it in the nth digit position of the resulting string. Since the total number of h_i's in P is same as the total number of digits in s, it follows that $u_i = s_i + c_i$ or $s_i + c_i + j$. Addition of the term c_i is due to the function f_i and absence or presence of the term j is determined by the values of x_i and x_{i+1}, as each g function involves addition by j. In fact, there can be 4 possible cases.

i) $x_i = 0$, $x_{i+1} = 0$. No g function is involved. Hence, $u_i = s_i + c_i$.

ii) $x_i = 0$, $x_{i+1} = 1$. Only one g function is involved and hence $x_i = 1$, $u_i = s_i + c_i + j$.

iii) $x_i = 1$, $x_{i+1} = 0$. $u_i = s_i + c_i + j$.

iv) $x_i = 1$, $x_{i+1} = 1$. Two g functions are involved and hence $u_i = (s_i + c_i + j + j) \bmod 2j = s_i + c_i$.

Hence, the proof. □

Lemma 7.10 *For* $1 \leq i \leq n$, $x_i \oplus x_{i+1} = u_i \bullet s_i$, *where* $x_{n+1} = x_1$.

Proof : If $x_i \oplus x_{i+1} = 0$, then $u_i - s_i = c_i$ (from Lemma 7.2). Note that $c_i \in \{1, 2, \cdots, j\}$. Hence, $u_i \bullet s_i = 0$ in this case.

Again, when $x_i \oplus x_{i+1} = 1$, $u_i - s_i = c_i + j$. In this case $u_i \bullet s_i = 1$. □

Lemma 7.11 $x_1 = (\sum (u_i \bullet s_i)) \oplus x_1$, *where* $\sum$ *stands for modulo 2 sum over* $i = 1, 2 \cdots, n$.

Proof : The proof is similar to that of Lemma 7.4. □

Corollary : It follows that $\sum (u_i \bullet s_i) = 0$.

Lemma 7.12 *For any pair of nodes* (s, t) *such that* $\sum (t_i \bullet s_i) = 0$, *there exists a path* P *between* s *and* t, *where* $P = h_1 g_1 h_2 g_2 \cdots h_i g_i \cdots h_n g_n$, $h_i = f_k$, $k \in \{1, 2, \cdots, j\}$.

Proof : The proof involves finding the strings c and x which will uniquely characterize the path P. By Lemma 7.9, $t_i = s_i + c_i$ or $s_i + c_i + j$. Hence, we find c_i by the following method.

First find $t_i - s_i \bmod 2j$. If this number $\in \{1, 2, \cdots, j\}$, then $c_i = (t_i - s_i) \bmod 2j$. Otherwise, $c_i = (t_i - s_i - j) \bmod 2j$.

To find the string x we use the result of Lemma 7.10, i.e., $x_i \oplus x_{i+1} = t_i \bullet s_i$. This equation must be satisfied for all i, which leads to the relation $x_1 = (\sum (t_i \bullet s_i)) \oplus x_1$. For $\sum (t_i \bullet s_i) = 0$, we will always be able to find a string x satisfying this relation. Hence, we start with an arbitrary value of $x_1 \in \{0, 1\}$. Then successively applying the equation $x_i \oplus x_{i+1} = t_i \bullet s_i$, for $i = 1, 2, \cdots, n-1$, we will get x_2, x_3, $\cdots$, x_n.□

Let $l(P)$ be the length of a path P of the above form. Now we give the following *theorem*.

Theorem 7.4 *For any pair of nodes (s, t) such that $\sum (t_i \bullet s_i) = 0$, there exists a path P between s and t of the above form such that $l(P) \leq \lfloor \frac{3n}{2} \rfloor$.*

Proof : The proof is similar to that of Theorem 7.1. □

Now we consider a path P of the form $g_1 h_1 g_2 h_2 \cdots h_{n-1} g_n$ where h_i's and g_i's are as defined earlier. Let the path P from s lead to a node $u = u_1 u_2 \cdots u_n$. Corresponding to the path P we define two strings $c = c_1 c_2 \cdots c_{n-1}$ and $x = x_1 x_2 \cdots x_n$ as before. Now we state the following lemmas :

Lemma 7.13 *For* $2 \leq i \leq n$, $u_i = \begin{cases} (s_{i-1} + c_{i-1}), & \text{if } x_i \oplus x_{i+1} = 0 \\ (s_{i-1} + c_{i-1}) + j, & \text{if } x_i \oplus x_{i+1} = 1 \end{cases}$

Also, $u_1 = \begin{cases} s_n, & \text{if } x_1 \oplus x_2 = 0 \\ s_n + j, & \text{if } x_1 \oplus x_2 = 1 \end{cases}$

Proof : The function $h_i (= f_k)$ operated on a string modifies the first digit, say a, of the string to $a + k$, i.e., $a + c_i$, and puts it in the nth digit position of the resulting string. Since the total number of f_i's in P is one less than the total

number of digits in s, the functions f_i's operate only on the digits s_1 to s_{n-1} and only the digits u_2 to u_n are affected by f_i's. It follows that either $u_i = s_{i-1} + c_{i-1}$ or $s_{i-1} + c_{i-1} + j$, for $2 \leq i \leq n$. The absence or presence of the term j is determined by the value of $x_i \oplus x_{i+1}$ as explained in Lemma 7.9. Moreover, $u_1 = s_n$ or $s_n + j$, depending on the value of $x_1 \oplus x_2$. Hence, the proof. □

Lemma 7.14 *For $2 \leq i \leq n$, $x_i \oplus x_{i+1} = u_i \bullet s_{i-1}$, where $x_{n+1} = x_1$.*

Proof : From Lemma 7.13, for $2 \leq i \leq n$, if $x_i \oplus x_{i+1} = 0$, then $u_i - s_{i-1} = c_{i-1}$. Note that $c_{i-1} \in \{1, 2, \cdots, j\}$ Hence in this case $u_i \bullet s_{i-1} = 0$.

Again, when $x_i \oplus x_{i+1} = 1$, then $u_i - s_{i-1} = c_{i-1} + j$. In this case $u_i \bullet s_{i-1} = 1$. □

Lemma 7.15 *For any pair of nodes (s, t) there exists a path of the form P between s and t', where $t' = t'_1 t_2 t_3 \cdots t_n$, t'_1 being equal to s_n or $s_n + j$.*

Proof : As in Lemma 7.12, we will first find the strings c and x which will uniquely characterize the path P. By Lemma 7.13, for $2 \leq i \leq n$, $t_i = s_{i-1} + c_{i-1}$ or $s_{i-1} + c_{i-1} + j$. Hence, for $1 \leq i \leq n-1$, we find c_i by the following method.

First find $(t_{i+1} - s_i) \bmod 2j$. If this number $\in \{1, 2, \cdots, j\}$, then $c_i = (t_{i+1} - s_i) \bmod 2j$. Otherwise, $c_i = (t_{i+1} - s_i - j) \bmod 2j$.

To find the string x we use the result of Lemma 7.14, i.e., we must have $x_i \oplus x_{i+1} = t_i \bullet s_{i-1}$ for $i = 2, 3, \cdots, n$. Summing up all these equations we get $x_1 \oplus x_2 = \sum_{i=2}^{n} (t_i \bullet s_{i-1}) = S$, say. Hence, we start with some arbitrary value for x_2 and apply the equations successively for $i = 2, 3, \cdots, n$, to get x_3, x_4, $\cdots$, x_n and x_1. Here, the value of $S = x_1 \oplus x_2$ only determines the first digit of t', i.e., t'_1 (Lemma 7.13). If $S = 0$, then $t'_1 = s_n$ and if $S = 1$ then $t'_1 = s_n + j$. □

Let $l(s, t')$ denote the length of the path P between s and t'.

Theorem 7.5 $l(s, t') \leq n - 1 + \lfloor \frac{n}{2} \rfloor$.

Proof : The proof is similar to that of Theorem 7.1. □

Theorem 7.6 *For any pair of nodes (s,t), there exists a path of length at most $\lfloor \frac{3n}{2} \rfloor + 1$ between s and t.*

Proof : By Theorem 7.5, $l(s,t') \leq \lfloor \frac{3n}{2} \rfloor - 1$. If $t'_1 = t_1$, then we are done. Otherwise, we consider the following cases.

Case i) $t'_1 = t_1 \pm k$, $k \in \{1, 2, \cdots, j-1\}$. If $t'_1 = t_1 + k$, then we apply $f_{k+1}\varphi_1$ on t' to reach t. If $t'_1 = t_1 - k$, then we apply $f_1\varphi_{k+1}$ on t' to reach t.

Case ii) $t'_1 = t_1 + j$. Note that the value of t'_1 is determined by $S = \sum_{i=2}^{n} (t_i \bullet s_{i-1})$. If we could complement S by some means, then the value of t'_1 could be changed from $t_1 + j$ to t_1. Now consider a node $t_1 t_2 \cdots t_{n-1} t^*_n$ such that $t^*_n \bullet s_{n-1} = (t_n \bullet s_{n-1}) \oplus 1$. Such a t^*_n can always be found out by setting

$$t^*_n = \begin{cases} t_n - k, & \text{for } (t_n - s_{n-1}) = k \text{ or } j + k,\ k \in \{1, 2, \cdots, j-1\} \\ t_n + 1, & \text{for } (t_n - s_{n-1}) = 0 \text{ or } j \end{cases}$$

Thus, starting from s, we can always reach the node $t_1 t_2 \cdots t_{n-1} t^*_n$ by a path of the form $g_1 h_1 g_2 \cdots g_{n-1} h_{n-1} g_n$ having length at most $\lfloor \frac{3n}{2} \rfloor - 1$. Applying now $f_1\varphi_2$ (if $t^*_n = t_n + 1$) or $f_{k+1}\varphi_1$, $k \in \{1, 2, \cdots j-1\}$ (if $t^*_n = t_n - k$), we can reach $t = t_1 t_2 \cdots t_n$. Hence, the proof. □

7.7 Conclusions

We have proposed a new family of graphs with $(2j)^n$ nodes, $j \geq 2$, $n \geq 2$, with a maximum node degree of $2j + 1$. We have shown that the diameter of this graph is bounded above by $\lfloor \frac{3n}{2} \rfloor + 1$. This upper bound is however not tight. For the graphs of maximum degree 5, we have exhaustively checked that except for $n = 5$, the diameter is one less than this bound for all values of n less than 7. For $n = 5$, the diameter is equal to this bound and the diameter may be equal to or less than this bound for $n > 6$.

For the graphs of maximum degree 5, a heuristic algorithm for point-to-point routing has been developed. The time complexity of this algorithm is $O(\log N)$ and for N upto 4096 the path computed by this algorithm is no longer than the shortest path by an amount 1.52 on the average. An $O(\log N)$ algorithm for single node broadcast has also been proposed and implementation of various algorithms have been discussed. Future works may include studying the fault-tolerance properties of this topology and embedding of other popular networks such as trees, meshes, etc.

Chapter 8

Dense Topology with Multiple Loops

8.1 Introduction

Ring topology is widely used due to its structural symmetry and simplicity. The routing algorithm in a ring is very simple. But the diameter of a ring with N nodes is $\lfloor \frac{N}{2} \rfloor$, which leads to a large communication delay. The diameter can be reduced if additional links are introduced within a ring. Some of the topologies resulting from such modifications are chordal ring [AL81], distributed loop network [BT91], etc. In a chordal ring, the degree of every node is 3, while the diameter is $O(\sqrt{N})$, N being the total number of nodes in the graph. In a distributed loop network the lower bound on the diameter is $\frac{\sqrt{2N-1}-1}{2}$, with increased uniform node degree of 4.

In this chapter, we propose a new family of network topologies, by adding a few links over the ring. Our proposed topology has nodes with degrees 2, 3, and 4 (average node degree is less than or equal to 3.25). The diameter of the proposed topology is $O(\log N)$ where N is the total number of nodes. The diameter of the network is expressed in terms of a parameter m ($m \geq 3$) where, N is an even multiple of m and $(m-1) \times 2^{\lfloor \frac{m-1}{2} \rfloor + 1} < N \leq m \times 2^{\lfloor \frac{m}{2} \rfloor + 1}$. The topology is denoted by a graph $G(N, m)$. The diameter of $G(N, m)$ is bounded above by $\lfloor \frac{11m}{8} \rfloor + 2$. In respect of diameter, this topology is thus superior to both the chordal ring and the distributed loop network. The total number of links used in this network

is less than that in a distributed loop network and is no more than 1/12 th of that in a chordal ring. The *Ascend* and *Descend* types of algorithms [PV81] can also be very efficiently implemented on this topology. Moreover, the successive values of N, for which the proposed topology can be defined, are at an interval of $2m < 4 \log N$. That is, if N and N', are two successive values of the total number of nodes with $N' \geq N$, then $N' - N = 2m$. This may be contrasted with other fixed degree topologies having $O(\log N)$ diameter, e.g., Mobius graph [L S82], de Bruijn graph [B46], cube-connected cycle [PV81], etc., for which the successive values of N are at much larger intervals. For all such graphs, N' is at least $2N$. The proposed network graph is hamiltonian and an easy routing scheme on this topology has also been presented.

8.2 Description of the Topology

We define a graph $G(m, N)$, with the following characteristics:

a) N is the total number of nodes in the graph. Let the nodes be numbered as $0, 1, \cdots N-1$.

b) m is a parameter of the graph such that $m \geq 3$.

c) N is an even multiple of m, i.e., $N = 2k \times m$, for some positive integer k.

d) $(m-1) \times 2^{\lfloor \frac{m-1}{2} \rfloor + 1} < N \leq m \times 2^{\lfloor \frac{m}{2} \rfloor + 1}$.

e) The nodes are connected by the following three types of edges (all operations below are treated under modulo N, unless otherwise mentioned) :

1) The node i is connected to the nodes $(i+1)$ and $(i-1)$. Thus the nodes are connected in the form of a cycle. We call these edges as c-edges (cyclic edges).

2) For $0 \leq i \leq 2k-1$, the node $i.m$ is connected to the diametrically opposite node $(i.m + N/2)$. We call these edges as d-edges (diagonal edges).

After introducing d-edges, there are $2k$ number of nodes in the network, which are

of degree 3. These degree 3 nodes divide the cycle, formed in (1), into $2k$ parts. We call each of these parts as a sector of length m. The sector j consists of the nodes, $\{j.m,\ j.m+1,\ j.m+2,\ \cdots,\ (j+1)m\}$, $0 \leq j \leq (2k-1)$.

3) Nodes in different sectors are interconnected by a third type of edges, called as hops or h-edges, as described below.

Starting from the node $j.m+1$, in sector j, $0 \leq j \leq 2k-1$, h-edges are introduced at every alternate node of degree 2 (there are $\lfloor \frac{m}{2} \rfloor$ such nodes in every sector). These connections are done by the following ways depending on the value of m.

Let $\lfloor m/2 \rfloor - 1 = r$.

Case 1 : r is even.

i) The node $j.m+i$ is connected to the nodes $[(j.m+i) \pm m \times 2^{r-(i-1)}]$. These h-edges are termed as hops of type $r-(i-1)$, $i = 1,\ 3,\ 5,\ \cdots,\ r+1$, which are denoted by $h_{r-(i-1)}$. The type is actually determined by the length of the hop.

ii) The node $j.m+i+r$ is connected to the nodes $[(j.m+i+r) \pm m \times 2^{i-2}]$. These edges are termed as hops of type $i-2$, $i = 3, 5, \cdots, r+1$ and are denoted by h_{i-2}.

Case 2 : r is odd.

i) The node $j.m+i$ is connected to the nodes $[(j.m+i) \pm m \times 2^{r-(i-1)}]$, by the hops of type $r-(i-1)$, $i = 1, 3, 5, \cdots, r$, which are denoted by $h_{r-(i-1)}$.

ii) The node $j.m+i+r$ is connected to the nodes $[(j.m+i+r) \pm m \times 2^{i-2}]$, by the hops of type $i-2$, $i = 2, 4, \cdots, r+1$ and are denoted by h_{i-2}. □

An example of $G(5, 40)$ is given in Fig. 8.1. It is to be noted here that the largest hop originated from a sector, is of type $\lfloor m/2 \rfloor - 1$. Hops of even (odd) type will be referred to as *even (odd)* hops. The connection pattern shows that even and odd hops originate from different halves of a sector.

Average node degree of this graph is $3+1/(m+1)$, which approaches to 3 for large N. Thus the total number of edges in the graph is asymptotically $\sim 1.5N$.

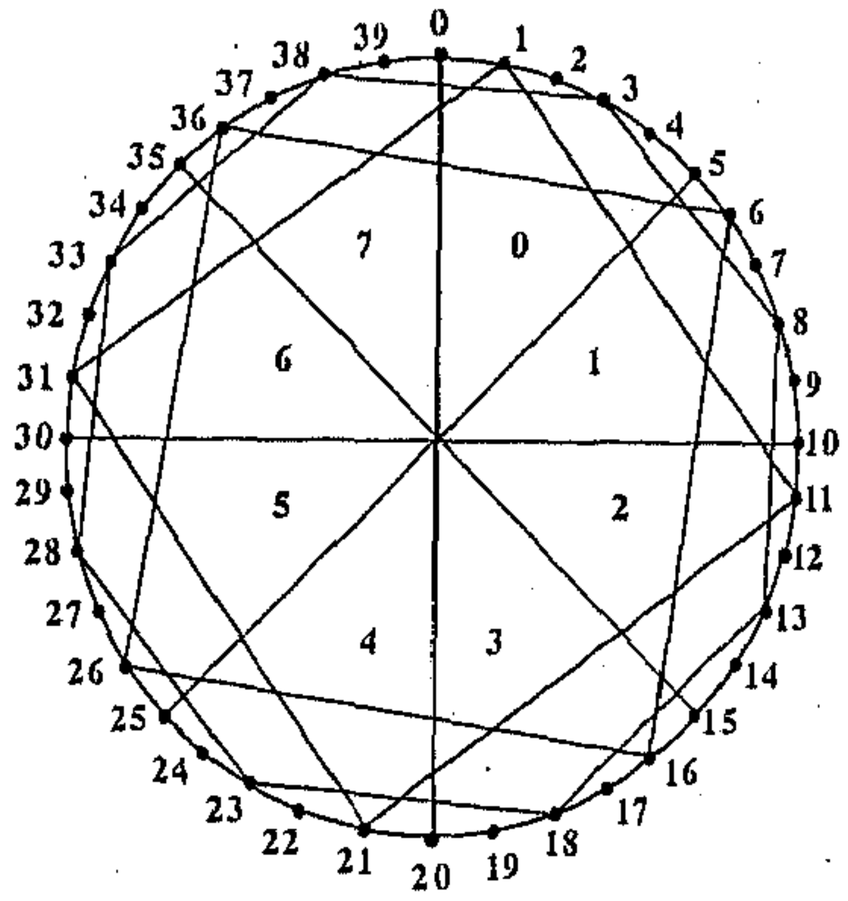


Figure 8.1: The proposed graph G(5, 40)

8.3 Diameter of the Network

In this section, we find an upper bound on the diameter D of the graph $G(m, N)$. To start with, we discuss about a restricted redundant binary number system to represent a node of the graph.

8.3.1 Restricted Redundant Binary Representation

Definition 8.1 A *Redundant Binary* representation of a number $k = k_{r-1}\ k_{r-2} \cdots k_0$, is one in which each digit k_i, $0 \leq i \leq r-1$, is an element of $\{0,\ 1,\ \bar{1}\}$ and $K = \sum_{i=0}^{r-1} 2^i k_i$.

Naturally, k does not have a unique representation in redundant binary, e.g., the binary number 0111011 has the equivalent representations $100\bar{1}10\bar{1}$ and $1000\bar{1}0\bar{1}$ in redundant binary.

Definition 8.2 A redundant binary representation, in which there is at least one zero bit in between two non-zero bits, will be termed as an RRB *Restricted Redundant Binary* representation.

Example 8.1 For the binary string 0111011, its RRB representation is $1000\bar{1}0\bar{1}$.

There is a simple way of obtaining the RRB representation of a number from its binary representation. Start scanning from the least significant bit (lsb) of the binary representation. As soon as a continuous run of 1's of length l, $l \geq 1$, is encountered, replace that by a new string $1000\cdots0\bar{1}$, of length $(l+1)$. Again, start scanning from the most significant bit (msb), i.e., '1', of the replaced string and repeat the same process until we finally get the RRB representation. The process will clearly take $O(r)$ time.

Example 8.2 The string 1111011101011 can be converted to the equivalent RRB representation by the following steps :
$1111011101011 \rightarrow 111101110110\bar{1} \rightarrow 11110111101\bar{1}0\bar{1} \rightarrow$
$11111000\bar{1}0\bar{1}0\bar{1} \rightarrow 10000\bar{1}00\bar{1}0\bar{1}0\bar{1}$.

We state the following results about the RRB representation.

Lemma 8.1 *The number of non-zero bits in the RRB representation of a number k is less than or equal to $\left\lceil \frac{\lceil \log k \rceil + 1}{2} \right\rceil$.*

Proof : When K is a power of 2, its RRB representation consists of a single '1' at the left most position. Otherwise, binary representation of K requires $\lceil \log k \rceil$ number of bits. In RRB representation, the number of bits required to represent k may be one more than that, i.e., $\lceil \log k \rceil + 1$. Since in between two consecutive non-zero bits there should be at least one zero bit in RRB representation, the maximum number of non-zero bits in this representation of k will be $\left\lceil \frac{\lceil \log k \rceil + 1}{2} \right\rceil$. □

Lemma 8.2 *The RRB representation of a number is unique.*

Proof : It is clear that $(00\cdots0)$ is the unique representation of 0.
Now, if possible let $a = (a_k a_{k-1} \cdots a_1 a_0)$ and $b = (b_k b_{k-1} \cdots b_1 b_0)$ be two distinct

RRB representation of a number, say x. Then $a-b$ must be 0. Since a and b are distinct representations, they must differ by at least one bit position. Let $a_i \neq b_i$. If *only one of* a_i *and* b_i *is non-zero*, then the i^{th} bit of $a-b$ will be non zero and since in RRB representation every non-zero bit is padded by at least one zero bit from both the sides, no carry bit can make this i^{th} bit zero in the final representation of $(a-b)$.

When both of a_i and b_I are non-zero, without loss of generality, *let us suppose* $a_i = 1$ and $b_i = -1$. In that case, the i^{th} bit of $a-b$ will be 0 and the carry 1 will be propagated to the next bit. Now as we are considering both a_{i+1} and b_{i+1} will be 0. Thus the carry of the i^{th} bit will make this $(i+1)^{th}$ bit of $(a-b)$ non-zero. Thus we have seen that $a-b$ can not be equal to 0, which contradicts our assumption. Therefore RRB *representation of a number is unique.* □

Lemma 8.3 *In the RRB representation, the largest number M that can be obtained using b number of bits is given by :*

$$M = \begin{cases} \frac{2}{3}(2^b - 1), & \text{if } b \text{ is even;} \\ \\ \frac{2}{3}(2^b - 1) + \frac{1}{3}, & \text{otherwise.} \end{cases}$$

Proof : It is clear that for the largest number, every alternate bit starting from the most significant bit will be 1.

If b is even, then

$$L = 2^{b-1} + 2^{b-3} + \cdots + 2 = \frac{2}{3}(2^b - 1)$$

Otherwise,

$$L = 2^{b-1} + 2^{b-3} + \cdots + 2^0 = \frac{2}{3}(2^b - 1) + \frac{1}{3}$$

□

Let w_x be the total number non-zero bits in the RRB representation of a number x. Further suppose that w_x^e and w_x^o be the number of non-zero bits corresponding to the even ('0' is excluded) and odd powers of 2 respectively, in the RRB representation of x.

Example 8.3 For $x = 100\bar{1}0\bar{1}01$, $w_x = 4$, $w_x^e = 2$ and $w_x^o = 1$.

When x is odd, $w_x = w_x^e + w_x^o + 1$, otherwise $w_x = w_x^e + w_x^o$

8.3.2 Upper Bound on the Diameter

For convenience of later discussion, we denote two hops of type l originating from a vertex v by $(+h_l)$ and $(-h_l)$, according to their other end points being $v+2^l \times m$ and $v - 2^l \times m$, respectively. To find the diameter it is enough to consider the paths from a node s in sector 0, to a node $d \leq \frac{N}{2}$, since the connections among the nodes are symmetric.

Suppose, d belongs to the sector δ, $0 \leq \delta \leq \frac{N}{2m}$. Starting from sector 0, we usè hops of different types, to reach sector δ. To find a path from s to d, we would first identify the appropriate hops and then we would find a specific order in which these hops are to be accessed so that the total path length will be small. The method is described as follows:

Case 1 : $0 \leq d \leq \frac{N}{4}$, i.e., $0 \leq \delta \leq \lfloor \frac{N}{4m} \rfloor$.

First we find the RRB representation of δ. Hops corresponding to the non-zero bits of this representation will be selected. It means that if the j^{th} bit of the RRB representation is 1 ($\bar{1}$) then the path includes a hop h_{j-1} ($-h_{j-1}$), $1 \leq j \leq \lfloor \frac{m}{2} \rfloor$. It can be verified that the h-edges present in the network would support this selection process.

Lemma 8.4 *The number of hops required to reach a sector δ, starting from the sector 0 is at most $\lfloor \frac{m+2}{4} \rfloor$, $\forall \delta, 0 \leq \delta \leq \lfloor \frac{N}{4m} \rfloor$.*

Proof : It is clear that the hops required to reach sector δ is exactly equal to w_δ and $w_\delta \leq \lceil \frac{log\delta+1}{2} \rceil \leq \lceil \frac{log\lfloor \frac{N}{4m} \rfloor+1}{2} \rceil \leq \lceil \frac{\lfloor \frac{m}{2} \rfloor}{2} \rceil \leq \lfloor \frac{m+2}{4} \rfloor$. Thus we may need at most $\lfloor \frac{m+2}{4} \rfloor$ hops. □

Since even and odd hops originate from different halves of a sector, let us restrict ourselves to use either even or odd hops only. h_0 may be included in both the cases. Since accessing a hop of type l is equivalent to accessing two hops of type $(l-1)$, $\lfloor \frac{m}{2} \rfloor - 1 \geq l \geq 1$, it is always possible to get such a collection of hops. The conversion of the hops into only one type, even or odd, can be done as follows:

According to the RRB representation of δ, w_δ is the total number of hops required to reach sector δ. Among these, w_δ^o hops are odd and w_δ^e number of hops are even. Without loss of generality, let $w_\delta^e \geq w_\delta^o$. Our strategy is to replace these w_δ^o number of odd hops by even hops. We count below the maximum number of hops required to reach sector δ, under this restriction.

For odd δ, $w_\delta^e + w_\delta^o + 1 = w_\delta \Rightarrow w_\delta^e + w_\delta^o = w_\delta - 1 \leq \lfloor \frac{m+2}{4} \rfloor - 1$. After conversion, we need $2w_\delta^o + w_\delta^e + 1$ number of hops. Now, $2w_\delta^o + w_\delta^e + 1 \leq \lfloor \frac{3}{2}(w_\delta^o + w_\delta^e) \rfloor + 1$ $\leq \lfloor \frac{3}{2}(\lfloor \frac{m+2}{4} - 1) \rfloor + 1$, by Lemma 8.4. Thus the maximum number of hops required to reach sector δ (using either even or odd hops along with some hops of type 0) is $\lfloor \frac{3}{2}(\lfloor \frac{m+2}{4} \rfloor - 1) \rfloor + 1$.

For even δ, $w_\delta^e + w_\delta^o = w_\delta \leq \lfloor \frac{m+2}{4} \rfloor$. Here, the lsb of the RRB representation of δ is 0. Two cases may arise : (1) in the RRB representation of δ, it may not be possible to get $\lfloor \frac{m+2}{4} \rfloor$ number of non-zero bits, (2) the RRB representation of δ contains $\lfloor \frac{m+2}{4} \rfloor$ number of non-zero bits, in which case the only possibility is that every alternate bit, starting from the second least significant bit, is non-zero. In the latter case we do not need any conversion because all the hops will be odd. Therefore, when we need any conversion, in the worst case, the RRB representation of δ will contain $(\lfloor \frac{m+2}{4} \rfloor - 1)$ number of non-zero bits. Thus, after conversion the maximum number of hops required is $2w_\delta^o + w_\delta^e \leq \lfloor \frac{3}{2}(\lfloor \frac{m+2}{4} \rfloor - 1) \rfloor$.

Hence, the maximum number of hops required to reach sector δ, under the restriction that either even or odd hops (h_0 may be included in both the cases) can be used, is $\lfloor \frac{3}{2}(\lfloor \frac{m+2}{4} \rfloor - 1) \rfloor + 1$.

Case 2 : $\lfloor \frac{N}{4m} \rfloor \leq \delta \leq \lfloor \frac{N}{2m} \rfloor$.

To reach this part of our network, we may utilize diagonal edges. But if we include a diagonal edge in our path, we would not use any hop of type $\lfloor \frac{m}{2} \rfloor - 1$.

From sector 0, if we use a d-edge we will reach the sector $\lfloor \frac{N}{2m} \rfloor$. From there we have to cover a distance of j sectors to reach sector δ, where $j = \lfloor \frac{N}{2m} \rfloor - \delta$. To traverse this distance we would not use hops of type $\lfloor \frac{m}{2} \rfloor - 1$. Here, by Lemma 8.3 we have, $0 \leq j \leq \frac{2}{3}(2^{\lfloor \frac{m}{2} \rfloor - 1} - 1) + \frac{1}{3}$, which implies, $\lfloor \frac{N}{2m} \rfloor - \frac{1}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 1) \leq \delta \leq \lfloor \frac{N}{2m} \rfloor$. For such an δ, the total number of hops required to reach the sector δ can be counted as follows.

j can be represented in RRB using a maximum of $(\lfloor \frac{m}{2} \rfloor - 1)$ bits, among which the number of non-zero bits can be at most $\lceil \frac{\lfloor \frac{m}{2} \rfloor - 1}{2} \rceil = \lfloor \frac{m}{4} \rfloor$. With the restriction that only odd or even hops can be used, the maximum number of hops required to cover j number of sectors will be $\lfloor \frac{3}{2}(\lfloor \frac{m}{4} \rfloor - 1) \rfloor + 1$ by the same reasoning as in case 1. Thus, if $\lfloor \frac{N}{2m} \rfloor - \frac{1}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 1) \leq \delta \leq \lfloor \frac{N}{2m} \rfloor$, the maximum number of hops (including a diagonal edge) required to reach sector δ will be $\lfloor \frac{3}{2}(\lfloor \frac{m}{4} \rfloor - 1) \rfloor + 2$.

Now, we are left with the case when $\lfloor \frac{N}{4m} \rfloor \leq \delta \leq \lfloor \frac{N}{2m} \rfloor - \frac{1}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 1) - 1$. Using the upper bound on $\lfloor \frac{N}{2m} \rfloor$, the above range is redefined as $\lfloor \frac{N}{4m} \rfloor \leq \delta \leq \frac{2}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 2)$. By Lemma 8.3, the farthest sector (from sector 0), whose RRB representation contains $\lfloor \frac{m}{2} \rfloor$ bits is $\frac{2}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 1)$. This shows that in this range when $\delta \leq \frac{2}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 1)$, we need not use d-edges. Only hops are sufficient here, as we have discussed in Case 1. The range $\lfloor \frac{N}{4m} \rfloor \leq \delta \leq \lfloor \frac{N}{2m} \rfloor - \frac{1}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 1) - 1$ is totally contained in the range $0 \leq \delta \leq \frac{2}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 1)$. Thus in this case, the number of hops required to reach the destination sector δ is same as the bound obtained in Case 1.

Hence for $0 \leq \delta \leq \frac{N}{2m}$, the number of hops required to reach sector δ, is at most $\max(B_1, B_2)$, where $B_1 = \lfloor \frac{3}{2}(\lfloor \frac{m+2}{4} \rfloor - 1) \rfloor + 1$ and $B_2 = \lfloor \frac{3}{2}(\lfloor \frac{m}{4} \rfloor - 1) \rfloor + 2$. Now, for $m = 8p + r$, $0 \leq r \leq 7$,

$$\max(B_1, B_2) = \begin{cases} \lfloor \frac{3}{2}(\lfloor \frac{m+2}{4} \rfloor - 1) \rfloor + 1, & \text{when } r = 2, 3; \\ \\ \lfloor \frac{3}{2}(\lfloor \frac{m}{4} \rfloor - 1) \rfloor + 2, & \text{otherwise.} \end{cases}$$

□

So far we have discussed about how to reach sector δ, starting from sector 0. But actually our task is to reach the node d in sector δ, starting from the node s in

sector 0. After using a hop, to get the next one we must traverse at least two c-edges. Moreover, a few c-edges may be required to get the first hop after starting from s and also to reach d in the final sector, after using all the hops. In what follows, we will count the total number of edges involved in the path from s to d other than the required hops as discussed above. Before that, we introduce the following notations.

1) x_j denotes the node in the sector j, from which hops of type 0 are originated. Thus x_j divides the sector j into two halves. Even and odd hops are emanated from these two halves.

2) By $[a, b]$ we mean the set of the nodes $a,\ a+1,\ a+2,\ \cdots,\ b$.

3) Let H_δ be the set of hops of either even or odd types, required to reach sector δ. Suppose $|H_\delta| = p$. Let $u_j^1,\ u_j^2,\ \cdots,\ u_j^p$ be the nodes in sector j such that the hops in H_δ originate from these nodes and $u_j^1 < u_j^2 < \cdots < u_j^p$. Again, we call $[u_*^1, u_*^p]$ as the range of H_δ in sector j.

With this notation, it is clear that $w_\delta^e + w_\delta^o \leq \lceil \frac{u_i^p - u_i^1 + 1}{2} \rceil$. Thus, the number of hops required to reach sector δ would be at most $\lfloor \frac{3}{2} \lceil \frac{u_i^p - u_i^1 + 1}{2} \rceil \rfloor \leq \frac{3}{4} u_0^p - \frac{3}{4} u_0^1 + \frac{5}{2}$.

4) Let the *projection* of d onto sector j be a node $\bar{d}_j$ in sector j, which is related to d by the equation $d = (\delta - j) \times m + \bar{d}_j$

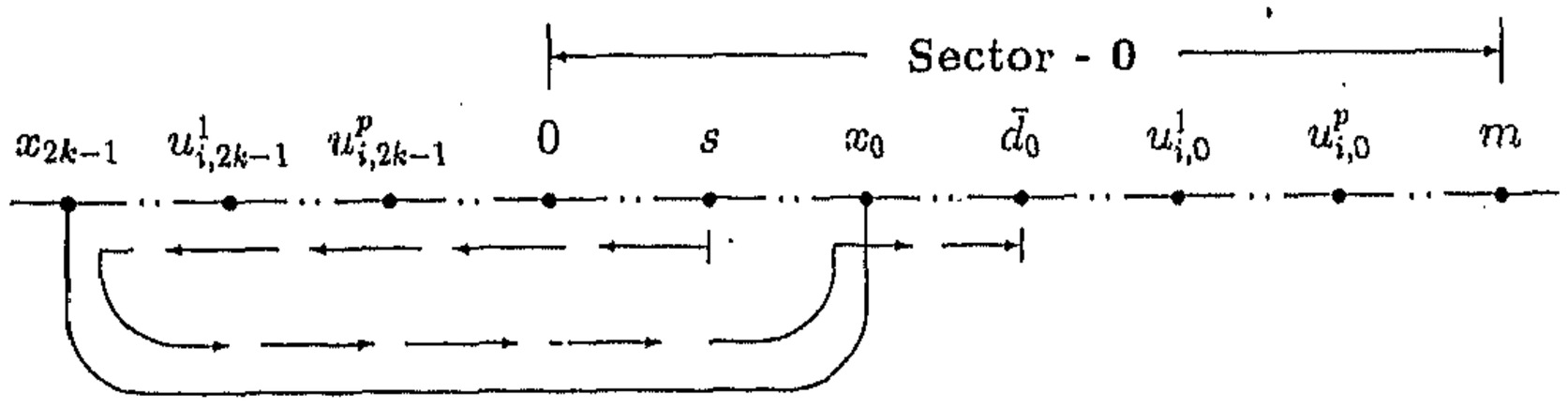
It is clear that to use the hops in H_δ (some of which may be repeated in the actual path), $\forall j, 1 \leq j \leq p$, we must traverse through u_r^j at least once for some r, $0 \leq r \leq 2k-1$. Now, counting the number of c-edges involved in this traversal can be equivalently mapped to the problem of counting the number of c-edges required to reach $\bar{d}_0$ starting from s, with the restriction that any one of $\{u_0^j,\ u_{2k-1}^j\}$ is traversed at least once. We will now show that by appropriately choosing the order of the hops, this number can always be made less than or equal to m, by using *at most* one additional hop of type 0.

Without loss of generality let us assume that $s \leq \bar{d}_0$.

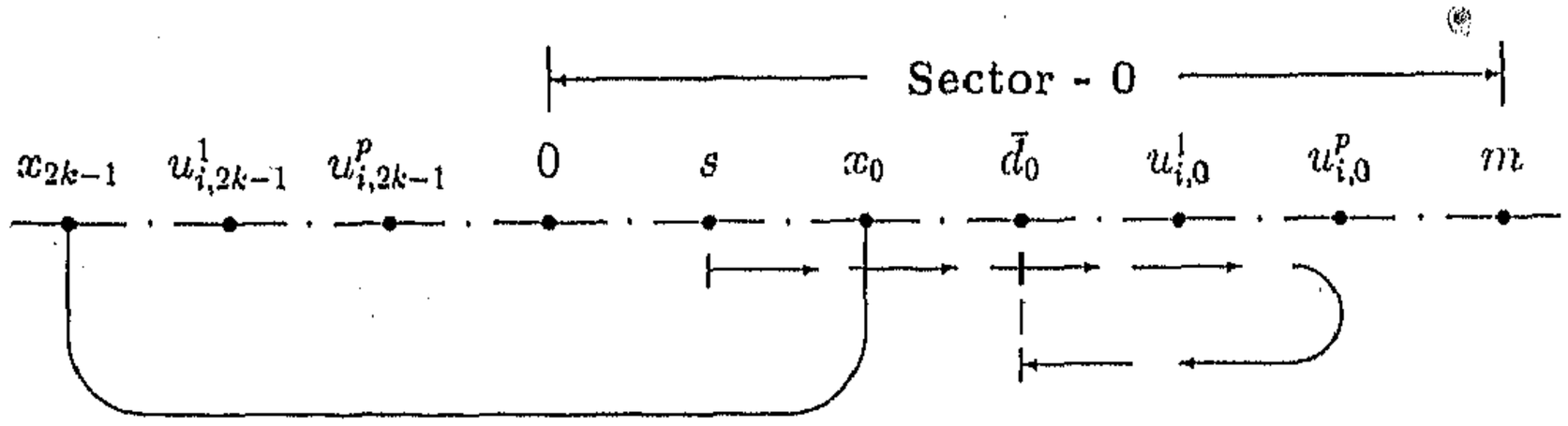
Case 1 : s and $\bar{d}_0$ are in the different halves of the sector 0. Let us consider the

following two subcases separately :

Subcase 1a : $[\, s, \bar{d}_0 \,] \cap (u_0^1, u_0^p) = \phi$.
In this case either $(u_0^1, u_0^p) \subseteq [\bar{d}_0 + 1, m]$ or $(u_0^1, u_0^p) \subseteq [0, s-1]$.
Suppose, $(u_0^1, u_0^p) \subseteq [\bar{d}_0 + 1, m]$. Consider the situation in Fig. 8.2.



(a) Traversal 1



(b) Traversal 2

Figure 8.2: Traversal 1 and Traversal 2

For $s + \bar{d}_0 \leq m$, use of the hops in H_δ requires the traversal from s to x_{2k-1} using c-edges, from x_{2k-1} to x_0 using an additional hop of type 0 and then from x_0 to $\bar{d}_0$ using c-edges. Hence, the number of edges other than those in H_δ is $s + \lfloor \frac{m}{2} \rfloor + 1 + \bar{d}_0 - \lfloor \frac{m}{2} \rfloor = s + \bar{d}_0 + 1$, with at most m number of c-edges.
We call this traversal as *traversal-1.*

For $s + \bar{d}_0 \geq m$, the required traversal is from s to u_0^p and from u_0^p to $\bar{d}_0$, using c-edges only. Thus the number of edges other than those in H_δ, is

$$u_0^p - s + u_0^p - \bar{d}_0 \leq 2m - (s + \bar{d}_0) \leq m$$

Let us call this traversal as *traversal-2*.

For $(u_0^1, u_0^p) \subseteq [0, s-1]$, it can similarly be shown that the number of c-edges can be at most m, with at most one additional hop of type 0.

Subcase 1b : $[\ s, \bar{d}_0\ ] \cap (u_0^1, u_0^p) \neq \phi$. If (u_0^1, u_0^p) is totally contained in $[\ s, \bar{d}_0\ ]$, then the required path is direct from s to $\bar{d}_0$, using c-edges and the path length is $\bar{d}_0 - s$. Otherwise, as shown in Fig. 8.3, it can be tackled in a similar way as we have done in subcase 1a. In both the situations the number of edges other than those in H_6, will be at most $m + 1$.

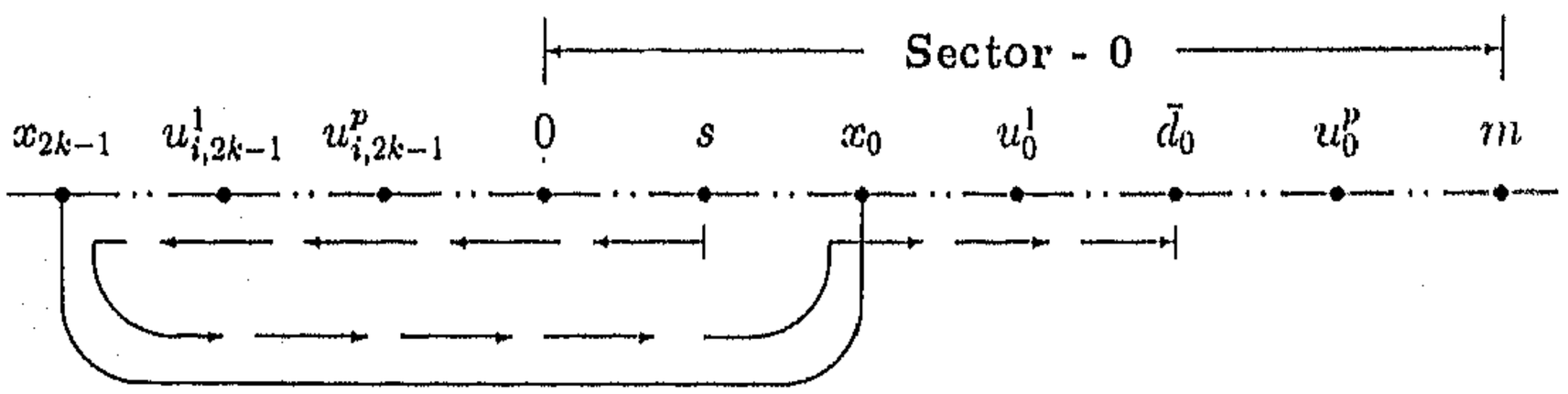


Figure 8.3: Traversal 1 for Case 2

Case 2 : s and $\bar{d}_0$ are in the same half of the sector.

Let us consider the situation when (u_0^1, u_0^p) is also in the same half and $[\ s, \bar{d}_0\ ] \subseteq (u_0^1, u_0^p)$, as shown in Fig. 8.4. Here, the traversal along the c-edges is from s to u_0^1, from u_0^1 to u_0^p and finally from u_0^p to $\bar{d}_0$. The total number of c-edges is thus equal to $(s - u_0^1) + (u_0^p - u_0^1) + (u_0^p - \bar{d}_0) = 2u_0^p - 2u_0^1 - (\bar{d}_0 - s) \leq m - (\bar{d}_0 - s) \leq m$.

All other cases can be treated in a similar way.

Combining all the above results, the length $\hat{L}$, of the longest path between any two nodes is given by

$$\hat{L} = \begin{cases} \lfloor \frac{3}{2}(\lfloor \frac{m+2}{4} \rfloor - 1) \rfloor + 2 + m, & \text{when } \delta = 2, 3; \\ \\ \lfloor \frac{3}{2}(\lfloor \frac{m}{4} \rfloor - 1) \rfloor + 3 + m, & \text{otherwise.} \end{cases}$$

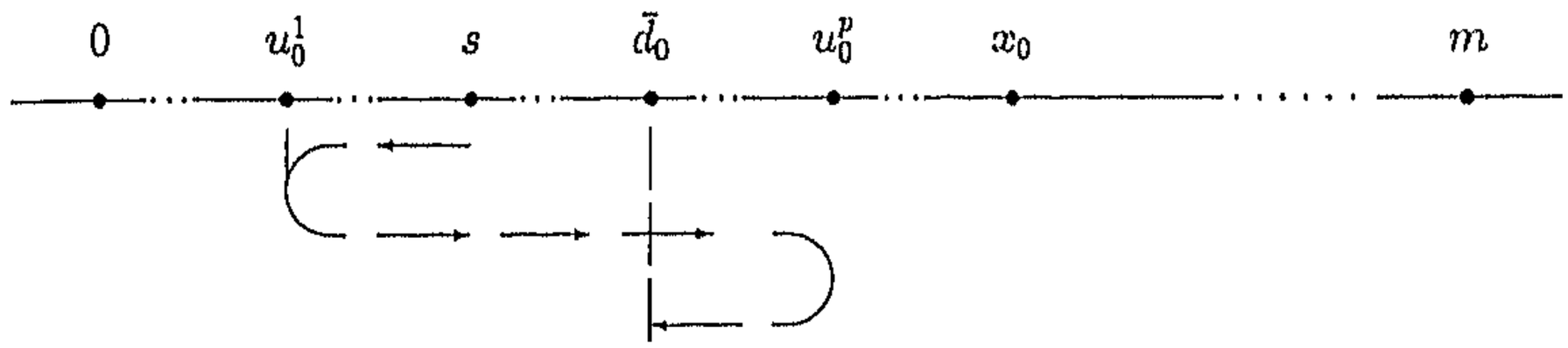


Figure 8.4: Traversal for Case 2

A little algebraic manipulation yields the following result.

Theorem 8.1 *The diameter D of the graph $G(m, N)$ is given by $D \leq \hat{L} \leq \lfloor \frac{11m}{8} \rfloor + 2$* □

Remark 8.1 When s and $\tilde{d}_0$ are in the same half of a sector, a path of smaller length, other than that we have established above, can always be found between s and d. In this case we will use both even and odd types of hops corresponding to the RRB representation of δ, except those whose origins correspond to the points in $[s+1, \tilde{d}_0 - 1]$. We will replace each of these hops in $[s+1, \tilde{d}_0 - 1]$ by exactly two hops of smaller length. The total number of hops required is thus at most $\lceil \frac{\lfloor \frac{m}{2} \rfloor}{2} \rceil + \lfloor \frac{\tilde{d}_0 - s}{2} \rfloor$. The corresponding traversal from s to $\tilde{d}_0$ is from s to x_{2k-1} using c-edges, from x_{2k-1} to x_0 using h_0 and finally from x_0 to $\tilde{d}_0$ using c-edges. Therefore the total path length will be

$$\begin{aligned} & \lceil \tfrac{\lfloor \frac{m}{2} \rfloor}{2} \rceil + \lfloor \tfrac{\tilde{d}_0 - s}{2} \rfloor + m - (\tilde{d}_0 - s) + 1 \\ \leq \quad & \lceil \tfrac{\lfloor \frac{m}{2} \rfloor}{2} \rceil + m - \tfrac{\tilde{d}_0 - s}{2} + 1 \\ \leq \quad & \lceil \tfrac{\lfloor \frac{m}{2} \rfloor}{2} \rceil + m + 1 \\ \leq \quad & \lceil \lfloor \tfrac{5m}{4} \rfloor + 2 \end{aligned}$$

Let us call this traversal as *traversal-3*.

8.4 Routing Algorithm

Input : 1) the source node s and the destination node d

2) the two paramotors m and N of the topology

Output : A path from s to d

Procedure RRB(sector-diff, E, O)

/* E and O are two linear arrays of maximum size $\lfloor \frac{m}{4} \rfloor$. These arrays are used to store even and odd bits of the RRB representation separately. 'sector-diff is the difference between the sector numbers of d and s */ **Step 1 :** Obtain the RRB representation of sector-diff.

Let it be $(b_L\ b_{L-1} \cdots b_2\ b_1\ b_0)$

Step 2 : for i = 1 to $\lfloor \frac{m}{4} \rfloor$ do

begin

E[i] = b_{2i};

O[i] = b_{2i-1};

end;

$b \leftarrow b_0$;

Procedure **Hops-SH**

/* Obtain hops in a specific order in which they will be used to reach d, starting from s, when $\bar{s}_0$ and $\bar{d}_0$ are in the same half */ **Step 1 :** convert the hops which are originated from $\{\bar{s}_0 + 1, \bar{d}_0 - 1\}$ and

accordingly modify the elements of E and O. The elements of these two arrays

give the number of times these hops are to be used in the path.

Step 2 : Following the traversal-3 (as in section 3), find the order in which these hops are to be used.

Procedure **Hops-DH**

/* Obtain hops in a specific order, in which they will be used to reach d, starting

from s, when $\bar{s}_0$ and $\bar{d}_0$ are in the different halves */ Step 1 : Obtain w_δ^e and w_δ^o.
(Where δ = sector-diff and w_δ^e, w_δ^o represent the same parameters as in section 3.)

Step 2 : if $(w_\delta^e > w_\delta^o)$ then convert all the odd hops to even hops and modify the elements of E and O

else convert all the even hops to odd hops and modify the elements of E and O accordingly.

Step 3 : /* To find the order in which the hops are to be used */

if $(\bar{s}_0 + \bar{d}_0 < m)$ then use the traversal-1

else use the traversal-2 (as discussed in section 3)

Procedure Find-Path (*distance*)

/* This procedure finds the hops which are to be included in the path and the order in which they are to be used to reach d starting from s. */

Step 1 : sector-diff $\leftarrow \lfloor \frac{distance}{m} \rfloor$;

Step 2 : range $\leftarrow \lceil \frac{2}{3}(2^{\lfloor \frac{m}{2} \rfloor} - 1) \rceil$;

if (sector-diff = 0) then walk along the outer circle from s to d;

if $(0 <$ sector-diff $\leq$ range) then RRB(sector-diff, E, O)

else

begin

diag-edge $\leftarrow$ 'yes';

/* to indicate whether a diagonal edge is to be included */

sector-diff $\leftarrow -(\frac{N}{2m} - 1 -$ sector-diff');

RRB(sector-diff, E, O);

end;

Step 3 : $\bar{s}_0 \leftarrow s - \lfloor \frac{s}{m} \rfloor \times m$;

$\bar{d}_0 \leftarrow d - \lfloor \frac{d}{m} \rfloor \times m$;

Step 4 : /* assume that $\bar{s}_0 \leq \bar{d}_0$ */

if ($\bar{s}_0$ and $\bar{d}_0$ are in the same half of the sector) then Hops-SH;

else Hops-DH;

```
/* Main Programme */

begin
   distance  ← (d − s + N) mod N;
   if  distance ≤  N/2 then
      Find-Path (distance)
    else
      begin
        Find-Path (N - distance);
        Keeping the order same, change the sign of all the hops obtained;
      end;
end.
```

8.5 Fault Diameter

It is clear that the graph is biconnected. To find the diameter in presence of a single faulty node, we proceed as follows.

Let P be the path between two nodes s and d, obtained by the method discussed in section 3 with the path length $L(P) \leq \lfloor \frac{11m}{8} \rfloor + 2$. In presence of a single faulty node, if the faulty node f lies on the path P, then there may be two possible cases : (A) the faulty node lies on a sector other than the sector 0 and the destination sector, (B) the faulty node is in the destination sector. The case for the faulty node lying in sector 0 can be treated in the same way as the case (B).

Case A :

Let h_q and h_r $(q < r)$ be two consecutive hops to be accessed in the path P and suppose while traversing along P, we enter the sector j by using the hop h_q and leave that sector by using the hop h_r. Let v_q and v_r be the respective nodes in sector j from where h_q and h_r originate.

The fault would affect the path if any one of the following occurs :

Case 1 : The node v_r itself is faulty.

Case 2 : The node v_r is a live node but the hop h_r would take us to such a node which is faulty.

Case 3 : An intermediate node within the range $[v_q, v_r]$ is faulty.

We do not consider the case when the node v_q is faulty. Because in that case, the situation will be identical to the case (2) above for the sector from which we enter sector δ. We consider below each of these cases separately.

Case 1 : Instead of using a hop h_r, we can use four hops of type $(r-2)$. Thus, in this case the path length would be increased by three. Therefore, the path length would be at most $L(P) + 3 \leq \hat{L} + 3$.

Case 2 : In this situation, the only restriction is on the hop h_r, originated from v_r in sector j. There is no difficulty in using the hop $-h_r$ from v_r. Now, $2^r = -3 \times 2^r + 2^{r+2}$. Instead of using a hop h_r from sector j, we can use $-h_r$ three times consecutively, starting from sector j, and then h_{r+2} once. Thus in P, the hop h_r is actually replaced by a sequence of four hops, namely, $\{-h_r, -h_r, -h_r, h_{r+2}\}$. Moreover, if h_r is to be used twice in P, then it will be replaced by six $-h_r$ hops followed by two h_{r+2} hops. Thus, the path length would be at most $L(P) + 6 \leq \hat{L} + 6$.

Case 3 : We consider two subcases.

Subcase 3a : The faulty node is of degree 2.

Let v_l and v_{l+2} be the two neighbors of the faulty node f, and $\pm h_l$ and $\pm h_{l+2}$ be the hops originated from these two nodes respectively. To bypass f, a sequence of five hops, namely, $\{-h_l, -h_l, -h_l, -h_l, h_{l+2}\}$ will be accessed in between h_q and h_r. Thus the path length would be at most $L(P) + 5 \leq \hat{L} + 5$.

Subcase 3b : The faulty node is of degree 4.

This situation is illustrated in Fig. 8.5. Let the hops that are originated from f be of type l. Also, let $q \neq l-2$ and $r \neq l+2$. Hence, to bypass f, a sequence of hops $\{-h_{l-2}, -h_{l-2}, -h_{l-2}, -h_{l-2}, -h_l, -h_l, -h_l, h_{l+2}\}$ can be used in between h_q and h_r. Thus, we are inserting 8 more hops in the original sequence of hops in P.

A point should be noted here that the upper bound $\hat{L}$, on the length of the path P is a conservative estimate. While computing the value of $\hat{L}$, we assumed that

the path P contains all the even or odd types of hops at least once. But in this particular case, we see that there are at least three types of hops, namely, h_{l-2}, h_l and h_{l+2} which were not included in P. This reduces the upper bound on $L(P)$ to $\hat{L} - 3$. Therefore, in presence of such a faulty node, the path length would be at most $L(P) + 8 \leq \hat{L} + 5$.

If $q = l - 2$, then the sequence $\{h_q, h_r\}$ in P will be replaced by the sequence $\{-h_q, -h_q, -h_q, -h_l, -h_l, -h_l, h_{l+2}, h_r\}$. If $r = l + 2$, then the sequence $\{h_q, h_r\}$ in P will be replaced by the sequence $\{h_q, h_{l-2}, h_{l-2}, h_{l-2}, h_{l-2}, h_l, h_l, h_l\}$. Thus, the path length will be increased by 6.

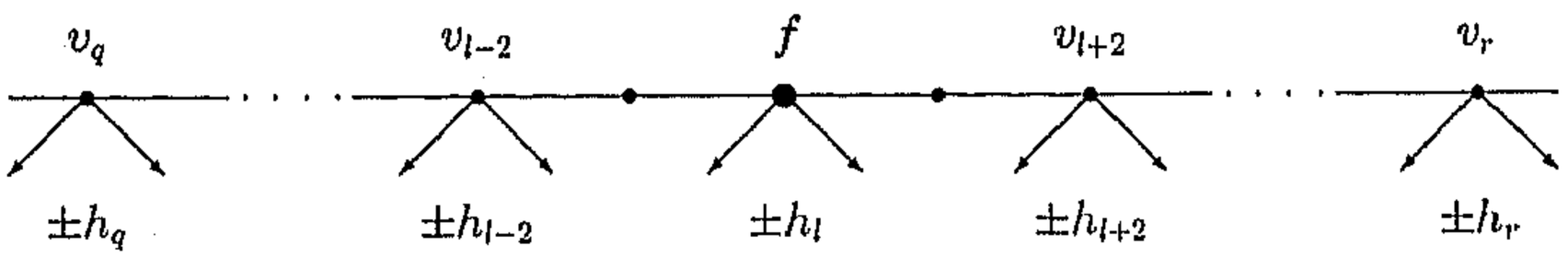


Figure 8.5: Faulty node is of degree 4

Case B :

Let us now consider the case when the faulty node lies in the destination sector. Of course, we assume that $f \neq d$. Actually, the fault would affect the path P only if it lies in that part of the destination sector which is included in P. Let us call that part as the *restricted portion* of the destination sector. These restricted portions will be different for different traversals from s to $\bar{d}_0$ as discussed in the last part of section 3. We shall consider those cases separately.

Case 1 : When s and $\bar{d}_0$ are in the different halves of the sector 0.

Subcase 1a : $[\, s, \bar{d}_0\,] \cap (u_0^l, u_0^r) = \phi$.

i) $s + d < m$

Here, in the destination sector (that is, in sector δ) the restricted portion is from x_δ to d.

If d is a degree-4 node, then let us assume that the hops that originating from d

be of type q. To bypass the fault in the restricted portion, along with the hops in P, we shall use two additional hops $+h_q$ and $-h_q$. The resulting traversal will be as follows. Start traversing along P, as discussed in section 7.3. During this traversal, we would have passed through a node $\bar{d}_r$, the projection of d on some sector r, $r \neq \delta$. As soon as we reach such a node for the first time, we use the hop $+h_q$, which is not originally in P. Then, again we shall follow the same order of using hops and c-edges as it was in the path P. Because of using an additional hop in the new traversal, after using all the other hops in P we shall reach the sector $(\delta + 2^q)$. Finally, from the node $\bar{d}_{\delta+2^q}$ in that sector, we would take the hop $-h_q$ to come back to the destination sector δ directly through the node d. Thus, in this case, we can bypass the fault at the expense of two additional edges.

If d is not a degree-4 node, then let us assume that the hops originating from the node $d+1$ be of type q. Here also we shall use two additional hops $+h_q$ and $-h_q$ in the new traversal so that as soon as we reach the projection of $d+1$ on some sector r for the first time, $r \neq \delta$, we take $+h_q$ and finally we would enter the destination sector through the node $d+1$ by using a $-h_q$ hop. From there, we have to come back to the node d using a c-edge. Thus the path length will be increased by 4.

ii) $s + d \geq m$

Here the restricted portion is from u_s^1 to d. This situation can be tackled in a way similar to the case (i) above, by taking two additional hops of type same as that originating from either d or $d-1$. Thus, here we bypass the fault at the expense of at most 4 additional edges.

Subcase 1b : $(u_0^1, u_0^p) \subseteq [s, \bar{d}_0]$

Here, corresponding to the path P, the restricted portion in the destination sector is from u_s^p to d. Let us consider the following two cases.

Case (a) : If $\bar{d}_0$ and (u_0^1, u_0^p) are in the different halves of the sector then to avoid the restricted portion in the destination sector, we will choose any one of the following two options :

Option 1 : Without loss of generality, let us assume that the path P contains only even type of hops. Thus, the total number of hops included in P will be at most $2w_\delta^o + w_\delta^e + 1$. The new traversal can be done as follows. Starting from s, use only c-edges to reach x_0. From there use h_0 to reach x_1 in sector 1. From sector 1, start using hops in increasing order of magnitude. After using all the hops in H_δ, use only c-edges to reach d in sector δ. Thus, in this case, the total path length will be

$$L(P_1) = \frac{m}{2} - s + 1 + \frac{m}{2} + m - \bar{d}_0 + 2w_\delta^o + w_\delta^e + 1$$

Option 2 : In this alternative path, we will use only odd type of hops. As a result, the range of H_δ will now be in the other half of the sector. That is, (u_0^1, u_0^p) and $\bar{d}_0$ are now in the same half of the sector. But at this point (u_0^1, u_0^p) may not be a subset of $[s, \bar{d}_0]$. The number of hops in this case will be $2w_e^0 + w_\delta^o + 1$. Here, we shall bypass the fault by the same technique (at the expense of at most 4 edges) as we have taken in the first part of the subcase 1a. The total path length will be

$$L(P_2) = s + \frac{m}{2} + 1 + \bar{d}_0 - \frac{m}{2} + 2w_e^0 + w_\delta^o + 1 + 4$$

Now, $\min(L(P_1), L(P_2)) \leq \frac{L(P_1)+L(P_2)}{2} \leq \frac{11m}{8} + 5$. This shows that the path length may increase by at most 3, in presence of such a fault.

Case (b) : If $\bar{d}_0$ and (u_0^1, u_0^p) are in the same half of the sector then if we interchange s and d, the situation will be identical to the case (a) above.

Subcase 1c : $(u_0^1, u_0^p) - [\, s, \bar{d}_0 \,] \neq \phi$

This situation can be treated in a similar way as we have done in subcase 1a.

Case 2 : s and $\bar{d}_0$ are in the same half of the sector.

With respect to the path P, discussed in the remark made in section 3, the restricted region in the destination sector will be included in the portion from x_δ to d.

Let the hops that originate from either $\bar{d}_0$ or $\bar{d}_0 - 1$ be of type q according to the situation whether d is a node of degree 4 or not, respectively. The fault in the

destination sector can be bypassed by using two additional hops of type $q-1$ and one of type q as follows. If we traverse along P, we would have passed through a node from which h_{q-1} originates. As soon as we enter such a node we will take h_{q-1} twice so that after using all the hops in P we will reach the sector $\delta + 2^q$. From there we will take a $-h_q$ hop to reach the destination d. Considering two additional c-edges at most, the fault can be bypassed at the expense of a maximum of five edges.

From the above discussion, we can conclude that in presence of a single fault, the diameter of the topology can be increased at most by 6.

8.6 Implementation of Algorithms

When N is a power of 2, a class of parallel algorithms, known as ASCEND and DESCEND types of algorithms [PV81], can easily be implemented on the proposed network topology. For brevity, we shall discuss here only the ASCEND type of algorithms.

Suppose, $N = 2^q$ and the input data $a_0, a_1, \cdots, a_{N-1}$ are stored in the processors $P[0], P[1], \cdots, P[N-1]$, respectively. An algorithm is in the ASCEND class if a sequence of operations is carried out between a pair of data that are successively $2^0, 2^1, \cdots, 2^{q-1}$ processor locations apart. We shall now discuss the implementation of such algorithms in our case. For the ease of discussion, let us first re-number the nodes of $G(m, N)$.

Re-numbering of nodes : Let $N = 2^q$ and $m = 2^r$. Therefore, $q = 2^{r-1} + r + 1$. Since there are $N/m = 2^{q-r}$ sectors, $G(m, N)$ contains 2^{q-r} cycles each of length $m + 1$, containing m c-edges and a single hop h_0. Let us number these cycles as $0, 1, \cdots, 2^{q-r} - 1$, so that the cycles i and $(i + 1)$ mod 2^{q-r} have one node in common. The node in cycle 0 from which the hop h_0 is originated, is now renumbered as 0. The remaining nodes are numbered from 1 to $N - 1$ along the largest cycle of length N, so that the cycle i now consists of the renumbered nodes

$\{im,\ im+1,\ \cdots,\ (i+1)m\}$. We would also represent any such renumbered node by an ordered pair (l,p), $0 \leq l \leq 2^{q-r}-1, 0 \leq p \leq m-1 = 2^r-1$, where l represents the cycle number which this node belongs to and p represents its distance from the node lm. Note that, since $p < m$, every node will have a unique representation by such an ordered pair. Thus, to address any of the N nodes, we require q bits in which the most significant $q-r$ bits would represent the cycle number and the least significant r bits would represent the position of the node in the corresponding cycle. Also, if a node is renumbered as n, then, $n = l.2^r + p$.

In our later discussions, we refer to a node by this renumbered value.

Before going to discuss the implementation, we now describe an ASCEND type of algorithm in the following two steps, where a basic operation between the two processors $P[i]$ and $P[r]$ has been indicated by OPER(i, j, $P[i]$, $P[r]$).

Proc ASCEND

```
Step 1 :    /* Process data elements within each cycle of length m + 1 */
            begin
              for each l, 0 ≤ l ≤ 2^(q-r) - 1,  do in parallel
              begin
                for j = 0 to r - 1   do
                begin
                  for each p, 0 ≤ p ≤ 2^r - 1   do in parallel
                        if bit_j (p) = 0 then
                          OPER(p, j, P[(l, p)], P[(l + 2^j, p)])
                end;
              end;
            end;

Step 2 :    /* Processes data elements across the cycles */
            begin
              for j = r to q - 1   do
              begin
              for each i, 0 ≤ i ≤ N - 1   do in parallel
```

```
            if bit_j (i) = 0 then
                OPER(i, j, P[i], P[i + 2^j])
        end;
    end;
```

Step 1 can be implemented in a similar way as it was discussed in [PV81]. This step can be executed on $G(m, N)$ in time linear in cycle length, that is, in $O(m)$ time. We shall now discuss the implementation of step 2 on $G(m, N)$. As an example, the hop distribution in cycle 0 of $G(2^3, 2^8)$ is illustrated in Fig. 8.6.

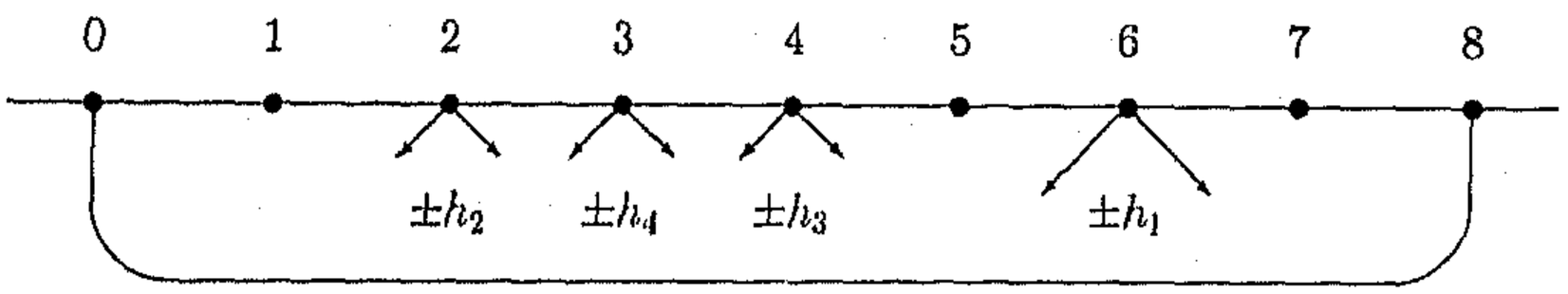


Figure 8.6: Hop distribution

Initially, let us start with the assumption that the data element a_i is stored in the processor $P[i]$, $\forall\ i, 0 \leq i \leq N-1$. We shall now concentrate only on those operations which involve a_0, the data element initially stored in the processor $P[0]$. Step 2 of ASCEND algorithm demands that the operations between the pairs of data elements stored in the processors $(P[0], P[8])$, $(P[0], P[16])$, $(P[0], P[32])$, $(P[0], P[64])$, $(P[0], P[128])$ should be carried out successively. At first, using the hop h_0, the processor $P[0]$ and $P[8]$ can communicate with each other so that any operation can be carried out between a_0 and a_8. To perform the later operations, the data elements are to be shifted to some other nodes where the suitable hops are available. This can be made possible by successive shifting of each data element through all the nodes in the cycle, which it belongs to. From this point it is clear that the corresponding positions within a cycle must be same for the data elements among which operations are to be carried out. Now, by three successive shifts, the data element which is actually stored in $P[0]$ can be brought to the processor

$P[6]$ from where the hop h_1 (which is actually a direct connection between cycle 0 and cycle 2) can be used to perform an operation with the element which was supposed to be stored in the processor $P[16]$ initially. Then to avail the next hop originating from the node 4, again two successive shifts are needed. The next available hop is h_3 which connects cycle 0 and cycle 8. With the help of this hop, an operation can be carried out between the elements which were initially stored in the processors $P[0]$ and $P[64]$. But, to fulfill the requirement of ASCEND algorithm the third operation must be between the data which were initially stored in the processors $P[0]$ and $P[32]$. Therefore, initially a_{32} must be stored in the processor $P[64]$. Similarly, a_{64} should be placed initially in $P[128]$ and a_{128} in $P[32]$. Thus, we see that some appropriate permutation must be specified for initial distribution of data elements among the processors. In general, the required permutation is described below in the inputting scheme.

Inputting of data elements : Let the sequence of hops originated from the nodes of the cycle l, $0 \leq l \leq 2^{q-r} - 1$, in the order $(l,1)$, $(l,2)$, $\cdots$, $(l,m-1)$, $(l+1,0)$ be designated as $< h_{i_1}, h_{i_2}, \cdots, h_{i_\alpha} >$, where $\alpha = 2^{r-1}$. Clearly, $h_{i_\alpha} = h_0$. For example, the sequence of hops in any cycle of $G(2^3, 2^8)$ is $< h_2, h_4, h_3, h_1, h_0 >$ (here, h_4 is the diagonal edge of length 2^7). The sequence of hops $< h_{i_1}, h_{i_2}, \cdots, h_{i_\alpha} >$ can be alternatively designated by a sequence of numbers $< i_1, i_2, \cdots, i_\alpha >$. Thus, for $G(2^3, 2^8)$, the sequence of hops is denoted by $< 2, 4, 3, 1, 0 >$.

We now define a permutation π_m, associated with $G(m, N)$ as follows:

$$\pi_m = \begin{pmatrix} q-r & q-r-1 & \cdots & 0 \\ i_1 & i_2 & \cdots & i_\alpha \end{pmatrix}$$

For $G(2^3, 2^8)$,
$$\pi_{2^3} = \begin{pmatrix} 4 & 3 & 2 & 1 & 0 \\ 2 & 4 & 3 & 1 & 0 \end{pmatrix}$$

If we scan the top row of π_m from the right end, we get the figures correspond to the types of the hops to be successively taken for executing step 2 of the ASCEND algorithm. The bottom row of π_m, on the other hand indicates the order of the hop types actually existing in the network $G(m, N)$, when scanned from one end of a cycle of length m.

Let the processor $P[i]$ be placed at the node i. The N data elements $a_0,\ a_1 \cdots,\ a_{N-1}$ will be distributed among the processors $P[0]$, $P[1]$, $\cdots$, $P[N-1]$, in such a way that the element a_i will be stored in the processor $P[j]$, where, $i = (l, p)$, $j = (l', p)$ and the binary representation of l' is obtained by taking the permutation π_m on the binary representation of l.

Let us now discuss the implementation of step 2 on $G(m, N)$:

For a fixed j the computation corresponding to the *for* loop in step 2 can not be executed in one parallel step, because within a cycle only one node is actually connected to a node at 2^j distance apart, $\forall j,\ r \leq j \leq q-1$. Therefore, by means of successive circular shifts all the data elements in the cycle should be brought to that node, so that OPER(., j, ., .) can be executed. For each j, this step requires $2^r + 1$ $(= m + 1)$ units of time. However, this computation can be pipelined and the total time required to execute step 2 can be reduced to only $O(m)$.

Step 2 can be explained in the following way.

bf Code section for the processor $(l,\ p)$ in Step 2 :

/* Assume that $\bar{l} = \pi_m^{-1}(l)$, where π_m^{-1} denotes the inverse permutation of π_m. That is, the binary representation of $\bar{l}$ is obtained by taking the permutation π_m^{-1} on the binary representation of l. Moreover, if $(l,\ p)$ is a degree-4 node then assume that $\pm h_k$ originate from that point. Let b represent the k^{th} bit of $\bar{l}$. The value of b can be either 0 or 1. */

for $i = 1$ *to* $2(m+1)$ **do**
begin
Step (2a) :
if ($(l,\ p)$ is a degree 4 node) **then**
if $m - p + 2 \leq i \leq 2m - p + 1$ **then**
if $(b = 0)$ **then**
use $+h_k$ to communicate with the processor $(l + 2^k,\ p)$ to perform any operation on the data elements stored in these two processors;

/* if ($p = 0$) then perform this operation on the data which the processor (l, 0) has obtained from (l, 1). */

else

use -h_k to communicate with the processor ($l - 2^k$, p) to perform any operation on the data elements stored in these two processors;

/* if ($p = 0$) then perform this operation on the data which the processor (l, 0) has obtained from (l, 1). */

Step (2b) :

if ($p \neq 0$) then send the data to (l, $p - 1$)

else

send the data which was obtained from (l, 1) to ($l + 1$, 0) and that which was obtained from ($l - 1$, 0) to the processor ($l - 1$, $m - 1$);

end

Since the nodes $i.m$ ($0 \leq i \leq 2^{q-r} - 1$) are at the joint of two successive cycles, cycle i and $(i - 1)$ mod $2^{q-r} - 1$, care should be taken for proper pipeline of data elements within a cycle, by keeping two separate registers for the two cycles.

The sequence of events corresponding to the *for* loop in step 2 is illustrated in Table 8.1, for $G(2^3, 2^8)$. In any cycle l of $G(2^3, 2^8)$, $0 \leq l \leq 31$, there are 8 nodes, namely, (l, p), $0 \leq p \leq 7$. For each of these processors, the time units during which they execute step 2a are marked "Y". The *for* loop in step 2 is executed $2m + 2$ times. Moreover, for each i, $(1 \leq i \leq 2m)$, it takes two units of time, one for step 2a and another for step 2b. Therefore, the time required to execute step 2 is 4m + 2 units, that is $O(\log N)$ time. So, we can conclude that the ASCEND class of algorithms can be implemented on $G(m, N)$ in $O(\log N)$ units of time, when N is a power of 2.

Table 8.1:

Node	Time units															
$(l, .)$	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
(., 0)	Y	Y	Y	Y	Y	Y	Y	Y								
(., 7)																
(., 6)				Y	Y	Y	Y	Y	Y	Y	Y					
(., 5)																
(., 4)						Y	Y	Y	Y	Y	Y	Y	Y			
(., 3)							Y	Y	Y	Y	Y	Y	Y	Y		
(., 2)								Y	Y	Y	Y	Y	Y	Y	Y	
(., 1)																

8.7 Conclusion

A new family of graphs with constant node degree and low diameter has been introduced. With a slight modification in the definition, this family of graphs can be constructed for all possible values of N. Given N, if N' is the nearest number to N, $N \leq N'$, on which the graph is defined, then construct a graph with N' nodes. Delete $N' - N$ ($< 2m$) number of degree 2 nodes, at most one from each sector. The diameter, $D(N)$, would be upper bounded by $D(N')$ and the maximum node degree would remain same as in the original graph.

Chapter 9

Conclusion

The overall throughput and resource utilization of a multiprocessor system using a network for interprocessor communication, depends largely on the topology of the network. In this thesis we have concentrated on the static interconection networks only. In the first few chapters we have addressed several problems related to two widely used interconnection networks, e.g., hypercubes and distributed loop networks. Later on we have introduced some new topologies and analysed their properties. Various algorithm have also been mapped on these topologies.

We have first considered the hypercube topology. Due to its rich connectivity and regular structure, hypercube has become one of the very popular topologies. We have considered reduction in diameter of a hypercube by bridging (adding some extra edges) or twisting (exchanging some pairs of independent links). The objective was to keep the number of links added or exchanged very small and thus preserving the original structure of the hypercube as far as possible. The number of links added constitute a small fraction of the total number of links originally present in the hypercube. For example, in a 10-dimensional cube this fraction was 0.02. The average internode distance for such a bridged hypercube has been calculated. We have also developed algorithms for the shortest path routing in such bridged and twisted hypercubes. Future work on these structures includes developing algorithms for other communication problems, such as single node broadcast, multinode broadcast, scattering, total exchange, etc. It will be

interesting to study the implementation of different algorithms on these modified hypercubes.

We have studied the fault-diameter of an n-cube in presence of $f \leq 3n-6$ faulty nodes. It is shown that the fault-diameter of an n-cube is $n+3$ with $3n-6$ faulty nodes provided that the hypercube remains connected. An efficient algorithm for routing in such situations has also been developed.

Another popular network that we have taken up for analysis is the distributed loop network. We have investigated various properties of distributed loop network in detail. Exploiting the underlying symmetry of the network structure, optimal time communication algorithms for multinode broadcast and single node scatter have been developed.

Construction of dense regular graphs for multiprocessor systems have attracted the interests of many researchers in the field of interconnection networks. We have introduced two new families of network topologies for this purpose.

1) A graph of maximum degree 5 with $N = 4^n$ $(n \geq 2)$ nodes and diameter $\leq \lfloor \frac{3 \log N}{4} \rfloor + 1$ has been proposed. These graphs are regular of order 5 for even n. When n is odd, only 4 nodes are of degree 4 and the remaining nodes are of degree 5. We have then generalized this idea to define families of such almost regular odd degree graphs of maximum degree $2j+1$, $(j > 2)$, with $(2j)^n$ nodes $(n \geq 2)$, and diameter $\leq \lfloor \frac{3 \log N}{4} \rfloor + 1$. Various algorithms were also implemented on these topologies.

2) The second structure is a modification of the ring network with the addition of a few links on the ring. Maximum degree of this network graph is 4 and average degree is less than or equal to 3.25. The network graph with N nodes is denoted by $G(m, N)$, where m is a parameter of the graph such that $m \geq 3$ and N is an even multiple of m. The diameter of the proposed topology is of the order of $O(\log N)$. The successive values of N, for which the proposed topology can be defined, are at an interval of less than $4 \log N$. When N is a power of 2, *Ascend* and *Descend* class of algorithms [PV81] can be implemented on this topology in $O(\log N)$ time.

Bibliography

[A89] S. G. Akl, *The Design and Analysis of Parallel Algorithms.* NJ : Prentice - Hall, 1989.

[A65] S. B. Akers, "On the construction of (d, k) graphs," *IEEE Trans. Electron. Comput.*, p. 448, June 1965.

[AG81] R. Armstrong, and F. G. Gray, "Fault diagnosis in a boolean cube of microprocessors," *IEEE Trans. Comput.* vol. C-30, no. 8, pp. 587 - 590, Aug. 1981.

[AHK87] S. B. Akers, D. Harel and B. Krishnamurthy, "The star graph : An attractive alternative to the n-cube," *Proc. Int. Conf. on Parallel Processing*, pp. 393-400, 1987.

[AK89] S. B. Akers and B. Krishnamurti, "Group graphs as interconnection networks," in*Proc. Int. Conf. Fault tolerant Comput.*, 1984, pp. 422-427.

[AK89] S. B. Akers and B. Krishnamurti, "A group theoretic model for symmetric interconnection network," *IEEE Trans. on Comput.*, vol. 36, pp. 555-566, Apr. 1989.

[AL81] B. W. Arden and H. Lee, "Analysis of chordal ring network," *IEEE Trans. on Comput.*, vol. C-30, pp. 291-295, Apr. 1981.

[AL90] A. E. Amawy, and S. Latifi, "Bridged hypercube networks," *Journal of Parallel and Distributed Computing*, pp. 90-95 , Sep. 1990.

[AL90] A. E. Amawy, and S. Latifi, "Properties and performances of folded hypercubes," *IEEE Trans. Paralled and Distributed Systems*, vol. 2, pp. 31-42, Jan. 1991.

[AM89] M. Alam and R. Melhem, "How to use an incomplete binary hypercube for fault tolerance," in *Hypercube and Distributed Computers*, F. Andre and J. Verjus, Eds. Amsterdam, The Netherlands; North-Holland, 1989, pp. 329-341.

[AP89] S. Abraham, and K. Padmanabhan, "Performance of direct binary n-cube network for multiprocessors," *IEEE Trans. Comput.* vol. C-38, no. 7, pp. 1000 - 1011, Jul. 1989.

[AP91] S. Abraham, and K. Padmanabhan, "Twisted cube : A study in asymmetry," *Journal of Parallel and Distributed Computing*, vol. 13, pp. 104 - 110, Nov. 1991.

[AQS93] S. G. Akl, K. Qiu, and I. Stojmenovic, "Fundamental Algorithms for the star and pancake interconnection networks with application to computational geometry," *Networks*, vol. 23, no. 4, pp. 215-225, Jul. 1993.

[B46] N. G. de Bruijn, "A combinatorial problem, "*Koninklijke Nederlande Academie van Watenschapen Proc.*, vol. A49, pp. 758-764, 1946.

[BI73] E. Bannai and T. Ito, "On finite Moore graphs," *J. Fac. Sci.*, Tokyo Univ., pp. 191-208, 1973.

[BIP85] J. C. Bermond, G. Illiades, and C. Peyrat, "An optimization problem in distributed loop computer networks," *Proc. 3rd Int. Conf. on Combinatorial Mathematics*, 1985.

[BA84] L. N. Bhuyan and D. P. Agrawal, "Generalized hypercube and hypercube structures for a computer network," *IEEE Trans. Comput.*, vol. C-33, no. 4, pp. 323-333, Apr. 1984.

[BDQ82] J. -C. Bermond, C. Delrome and J. -J. Quisquater, "Tables of Large Graphs with given degree and diameter," *Inform. Processing. Lett.*, vol. 15, pp. 10-13, Aug. 1982.

[BNL93] N. Bagherzadeh, N. Nassif, and S. Latifi, "A routing and broadcasting scheme on faulty star graphs," *IEEE Trans. on Computers*, vol. 42, no. 11, pp. 1398-1403, Nov. 1993.

[BOS+91] D. P. Bertsekas, C. Ozveren, G. D. Stamoulis, P. Tseng, and J. N. Tsitsiklis, "Optimal Communication Algorithm for Hypercubes", *Journal of Parallel and Distributed Computing*, vol. 11, pp 263 - 275, April 1991.

[BT84] F. T. Boesch and R. Tindell, "Circulants and their connectivies," *J. Graph Theory* vol. 8, pp. 487-499, 1984.

[BT91] J. -C. Bermond, and D. Tzvieli, "Minimal-Diameter Double-Loop Networks : Dense optimal Family", *Networks*, Vol. 21, pp 1 - 9, Jan. 1991.

[BW85] F. T. Boesch and J. -F. Wang, "Reliable circulant networks with minimum transmission delay," *IEEE Trans. Circuits and Systems*, vol. CAS-32, pp. 1286-1291, 1985.

[CC94] S. Y. Cheng and J. H. Chuang, "Varietal hypercube - A new interconnection network topology for large scale multicomputer," *Proc. 1994, Int. Conf. on Parallel and Distributed Systems*, pp. 703-708, 1994.

[CL93] M. Y. Chan and S. Lee, "Fault-Tolerant Embedding of Complete Binary Trees in Hypercubes," *IEEE Trans. on Comput.*, vol. 4, no. 3, Mar. 1993.

[CS86] T. F. Chan and Y. Saad, "Multigrid algorithms on the hypercube multiprocessor," *IEEE Trans. Comput.*, vol. C-35, no. 11, pp. 969-977, Nov. 1986.

[CS88] M. Chen and K. Shin, "Message routing in an injured hypercube," in *Proc. 3rd Conf. Hypercube Concurrent Comput. and Appl.*, G. Fox Ed., ACM Press, pp. 312-317.

[CS90] M. Chen and K. Shin, "Depth-first search approach for fault-tolerant routing in hypercube microcomputers," *IEEE Trans. Parallel Distributed Syst.*, vol. 1, no. 2, pp. 152-159, 1990.

[D79] P. J. Davis, *Circulant Matrices*, John Wiley, New York (1979).

[D84] K. W. Doty, "New Designs for Dense Processor Interconnection Networks," *IEEE Trans. Comput.*, vol. C-33, pp. 447-450, May, 1984.

[DA94] K. Day and A. Tripathi, "A comparative study of topological properties of hypercubes and star graphs," *IEEE Trans. Parallel and Distributed systems*, vol. 5, no. 1, pp. 31-38, Jan. 1994.

[DGD89] S. K. Das, J. Ghosh and N. Deo, "Stirling Networks : A Versatile Combinatorial Topology for Multiprocessor Systems," *Technical Report Number N-89-003*, June 1989.

[DHL90] D. Du, D. Hsu, Q. Li and J. Xu, "A combinatorial problem related to distributed loop networks," *Networks*, vol. 20, pp. 173-180, 1990.

[DH91] S. Dutt and J. P. Hayes, "Designing fault-tolerant systems using Automorphisms," *Journal of Parallel and Distributed Computing*, vol. 12, pp. 249-268, 1991.

[DS87] W. J. Dally and C. L. Seitz, "Deadlock-free meassage routing in multiprocessor interconnection network," *IEEE Trans. Compt.*, vol. 36, no. 5, pp. 547-553, May 1987.

[E64] B. Elspas, "Topological constraints on interconnection-limited logics," *Switching Circuit Theory Logical Design*, pp. 133-147, Oct. 1964.

[E89] A. H. Esfahanian, "Generalized measures of fault tolerance with application to N-cube networks, " *IEEE Trans. Comput.*, vol. 38, no. 11, pp. 1586-1591, Nov. 1989.

[E92] K. Efe, "The crossed cube architecture for parallel computation," *IEEE Trans. Comput.*, vol. 3, no. 5, pp. 513-524, Sep. 1992.

[ENS91] A. Esfahanian, L. M. Ni, and B. E. Sagan, "The twisted N-cube with application to microprocesssing," *IEEE Trans. Comput.* vol. 40, no. 1, pp 88-93, Jan. 1991.

[ES79] P. Erdos and J. Spencer, "Evolution of the n-cube," *J. Comput. Math. Appl.*, vol. 5, pp 307-312, Mar. 1983.

[ET70] B. Elspas and J. Turner, "Graphs with Circulant Adjacency Matrices", *Journal of Combinatorial Theory*, no. 9, pp. 297-307, 1970.

[F66] H. Friedman, "A design for (d, k) graphs," *IEEE Trans. Electron. Comput.*, pp. 253-254, Apr. 1966.

[F71] H. Friedman, "On the impossibility of certain Moore graph," *J. Combinatorial Theory (B)*, pp. 245-252, Apr. 1966.

[F77] S. Foldes, "A characterization of hypercubes," *J. Discrete Math.*, vol. 17, pp. 155-159, 1977.

[F92] P. Fraigniaud, "Asymptotically optimal broadcasting and gossiping in faulty hypercube microprocessors," *IEEE Trans. Comput.*, vol. 41, no. 11, pp. 1410-1419, Nov. 1992.

[FH91] P. Fraigniaud and C. -T. Ho, "Arc-disjoint spanning trees on cube connected cycles networks," in *Proc. Int. Conf. Parallel Processing*, 1991.

[FL72] D. J. Farber and K. C. Larson, "The system architecture of the distributed computer system - the communication systems," *Proc. Symp.*

Computer Comm. Networks and Teletraffic, Brooklyn Polytechnic Press, pp. 21-27, Apr. 1972.

[FP91] P. Fraigniaud and C. Peyrat, "Broadcasting in hypercubes when calls fail," *Inform. Processing. Lett.*, vol. 39, pp. 115-119, 1991.

[FYA84] M. A. Fiol, J. L. A. Yebra, and I. Alegre, "Line digraph iterations and the (d, k) digraph problem," *IEEE Trans. Comput.*, vol. C-33, pp. 400-403, May 1984.

[GP79] W. H. Gates and C. H. Papadimitriou, "Bounds for sorting by prefix reversal," *Discrete Math*, vol. 27, pp. 47-57, 1979.

[GS81] J. R. Goodman and C. H. Sequin, "Hypertree : A Multiprocessor Interconnection Topology," *IEEE Trans. on Comput.*, pp. 923-933, Dec. 1981.

[GS88] J. M. Gordon and Q. F. Stout, "Hypercube message routing in presence of faults," in *Proc. 3rd Conf. Hypercube Concurrent Comput. and Appl.*, Jan. 1988, pp. 318-327.

[GS89] A. Ghafoor and P. Sole, "Performance of fault-tolerant diagnostics in the hypercube systems," *IEEE Trans. Comput.*, vol. 38, no. 8, pp. 1164-1172, 1989.

[H69] F. Harary, *Graph theory*. New York: Addison-Wisley, 1969.

[H76] S. Hart, "A note on the edges of the n-cube," *J. Discrete Maths.*, vol. 14, pp. 157-163, 1976.

[H93a] K. Hwang, *Advanced computer architecture : parallelism, scalibility, progammability*, McGraw-Hill, 1993.

[H93b] W. Hsu, "Fibonacci Cubes - A New Interconnection Topology," *IEEE Trans. on parallel and distributed systems*, vol. 4, no. 1, Jan. 1993.

[HAK87] D. Harel, S. B. Akers and B. Krishnamurthy, "The Star Graph : An attractive alternative to the n-cube, " *Proc. 1987 Conf. ICPP*, pp. 393-400.

[HHW88] F. Haray, J.P. Hayes and H. Wu, " A survey of the theory of hypercube graphs," *Comp. Math. Applic.* vol 15, no.4, pp. 277-289, 1988.

[HKS87] P. A. J. Hilberts, M. R. J. Koopman, and J. L. A. van de Snepscheut, "The twisted cube," *Parallel Architectures and Languages Europe, Lecture notes in Computer Science*, pp. 152-159, Jun. 1987.

[HK88] K. Hwang and D. Kim, "Generalization of orthogonal multiprocessors for massively parallel computation," in *Proc. 2nd Frontiers MPC*, Oct. 1988, pp. 391-398.

[HS60] A. J. Hoffman and R. R. Singleton, "On Moore graphs with diameter 2 and 3," *IBM J. Res. Devolop.*, pp. 497-504, 1960.

[II81] M. Imase and M. Itoh, "Design to minimize diameter on building block network," *IEEE Trans. Comput.*, vol. C-30, pp. 439-442, Jun. 1981.

[II83] M. Imase and M. Itoh, "A design for directed graphs with minimum diameter," *IEEE Trans. Comput.*, vol. C-32, pp. 782-784, Aug. 1983.

[JH89] S. L. Johnson and C. -T. Ho, "Optimum broadcasting and personalized communication in hypercubes," *IEEE Trans. Comput.*, vol. 38, no. 9, pp. 1249-1268, Sep. 1989.

[JLD90] J.-S. Jwo, S. Lakshmivarahan, and S. K. Dhall, "Embeddings of cycles and grids in star graphs," *Symp. on Parallel and Distrib. Processing*, pp. 540-547, 1990.

[K67] I. Korn, "On (d, k) graphs," *IEEE Trans. Electron. Comput.*, p. 90, Feb. 1967.

[K69] W. H. Kautz, "Design of optimal interconnection networks for multiprocessors," in *Architecture and Design of Digital Computers*, Nato Advanced Summer Institute, 1969, pp. 249-272.

[K80] H. T. Kung, "The structure of parallel algorithms," in *Advances in Computers*, vol. 19, M. C. Yovits, ed., Academic Press, N. Y., 1980.

[K88] H. P. Katseff, "Incomplete Hypercubes," *IEEE Trans. Comput.*, vol. C-37, pp. 604-608, May 1988.

[K93] K. Hwang, *Advanced Computer Architecture*. New York: McGraw-Hill, 1993.

[KK87] M. S. Krishnamoorthy and B. Krishnamurthy, "Fault diameter of interconnection networks," *Comput. and Math. with Appl.*, vol. 13, no. 5/6, pp. 577-582, 1987.

[L89] S. Latifi, "Identification of operational subcubes in faulty hypercube," Research report, Univ. Nevada, Las Vegas, 1989.

[L93a] S. Latifi, "Combinatorial analysis of the fault-diameter of the n-cube," *IEEE Trans. Comput.*, vol. 42, no. 1, pp. 27-33, Jan. 1993.

[L93b] F. T. Leighton, *Parallel Architectures and Algorithms : Arrays, Trees and hypercubes*. San Mateo, California: Morgan Kaufmann Publishers, 1993.

[L93c] S. Latifi, "On the fault-diameter of the star graph," *Information Processing Letters*, vol. 46, pp. 143-150, 1993.

[LA89] S. Latifi and A. Al-Amawy, "On folded hypercubes," *Proc. 1989 Int. Conf. Parallel Processing*, pp. 180-187, 1989.

[LB94] S. Latifi and N. Bagherzadeh, "Incomplete Star : An alternative scalable network based on the star graph," *IEEE Trans. on Parallel and Distributed Systems*, vol. 5, no. 1, pp. 97-102, Jan. 1994.

[LEN90] Y. Lan, A. H. Esfahanian and L. M. Ni, "Multicast in hypercube multiprocessors," *J. Parallel Distributed Comput.*, pp. 30-41, Jan. 1990.

[LF90] C. Li and W. Fuchs, "Graceful degradation on hypercube multiprocessors using data distribution," in *Proc. DMCC5*, D. Walker and Q. Stout, Eds., IEEE Computer Society Press, 1990, pp. 1446-1454.

[LH90] T. -C. Lee and J. F. Hayes, "One-step-degradable fault tolerant hypercube," in *Proce. Distributed Memory Comp. Conf.*, 1990.

[LH92] T. -C. Lee and J. P. Hayes, "A fault-tolerant communication scheme for hypercube computers," *IEEE Trans. Comput.*, vol. 41, no. 10, pp. 1242-1256, Oct. 1992.

[LK90] W. -M. Lin and V. K. P. Kumar, "Efficient histogramming on hypercube SIMD machines," *Comput. Vision, Graphics, and Image Processing*, vol. 49, pp. 104-120, 1990.

[LS82] W. E. Leland and M. H. Solomon, "Dense trivalent graphs for processor interconnection," *IEEE Trans. on Comput.*, vol. C-31, no. 3, pp. 219-222, Mar. 1982.

[M80] M. Mulder, "*n*-cube and median graphs," *J. Graph Theory*, vol. 4, pp. 107-110, 1982.

[MJ94] J. Misic and Z. Jovanovic, "Routing functions and deadlock avoidance in a star graph interconnection network," *Journal of Parallel and Distributed Computing*, vol. 22, no. 2, pp. 216-228, 1994.

[MR82] G. Memmi and Y. Railard, "Some new results about the (d, k) graph problem," *IEEE Trans. Comput.*, vol. C-31, pp. 748-791, Aug. 1982.

[MS91] K. Mukhopadhyaya, and B. P. Sinha, "Optimal Design and Routing of Distributed Loop Networks", *Proceedings of International Symposium on Circuits and Systems*, Singapore, Aug. 1991.

[MS92] V. E. Mendia and D. Sarkar, "Optimal broadcasting on the star graph," *IEEE Trans. on Parallel and Distributed Systems*, vol. 3., no. 4, pp. 389-396, Jul. 1992.

[NS81] D. Nassimi and S. Sahni, "Data Broadcasting in SIMD computers," *IEEE Trans. Comput.*, vol. 30, Feb. 1981.

[NSK90] M. Nigam, S. Sahni and B. Krishnamurthy, "Embedding hamiltonians and hypercubes in star interconnection graphs," *in Int. Conf. on Parallel Processing*, pp. III 340-343, 1990.

[PR82] D. K. Pradhan and S. M. Reddy,"A Fault-tolerant Communication Architecture for distributed systems," *IEEE Trans. on Comput.*, vol. C-31, pp. 863-870, Sept. 1982.

[PRW94] M. A. Palis, S. Rajasekaran and D. S. L. Wei, "Packet routing and PRAM evalutation on star graphs and leveled networks," *Journal of Parallel and Distributed Computing*, vol. 20, no. 2, pp. 145-157, Feb. 1994.

[PS87] J. L. Peterson and A. Silberschatz, Operating System Concepts, Reading, Massachusetts, Addison-Wesely, 1987.

[PV81] F. P. Preparata and J. Vuillemin, "The cube-connected cycles : A versatile network for parallel computation," *CACM*, vol. 24, pp. 300-309, May. 1981.

[QAM94] K. Qiu, S. G. Akl, and H. Meijer, "On some properties and algorithms for the star and pancake interconnection networks," *Journal of Parallel and Distributed Computing*, vol. 22, pp. 16-25, 1995.

[RL75] C. C. Reames and M. T. Liu, "A loop network for simultaneous transmission of variable length messages," *Proc. Second Symp. Computer Architecture*, pp. 7-12, Jan. 1975.

[RRK83] S. M. Reddy, P. Raghavan and J.G. Kuhl, "A class of graphs for processor interconnection," in *Proc. 1983 Conf. ICPP*, pp. 154-157.

[RS88] P. Ramanathan and K. G. Shin, "Reliable broadcast in hypercube multicomputers," *IEEE Trans. Comput.*, vol. 37, no. 12, pp. 1654-1657, Dec. 1988.

[RS93] Yordon Rouskov, and Pradip K. Srimani, "Fault diameter of star graphs," *Information Processing Letters*, vol. 48, pp. 243-251, 1993.

[S70] R. M. Storwick, "Improved construction technique for (d, k) graphs," *IEEE Trans. Electron. Comput.*, pp. 1214-1216, Dec. 1970.

[S82] L. Snyder, "Introduction to the Configurable, Highly Parallel Computer," *IEEE Trans. on Comput.*, pp. 47-56, Jan. 1982.

[SB93] H. Shen and R-J. Back, "Construction of Large-Size Interconnection Networks with High Performance," *Networks*, vol. 23, No 4, pp. 399-414, July 1993.

[SS85] Y. Saad and M. H. Shultz, "Data communications in hypercubes," Dep. Comput. Sci., Yale Univ. Res. Rep. 428/85., 1985.

[SS88] Y. Saad and M. H. Shultz, "Topological properties of hypercubes," *IEEE Trans. Comput.*, vol. 37, no. 7. pp. 867-872, July 1988.

[T78] C. D. Thompson, "Generalized connection networks for parallel processor intercommunication," *IEEE Trans. Comput.*, vol. C-24, no. 12, pp. 1119 - Dec. 1978.

[T91] D. Tzvieli, "Mimimal Diameter Double-Loop Networks I. Large Infinite Families," *Networks*, vol. 21, pp 387-415, Jul. 1991.

[TR93] S. B. Tien and C. S. Raghavendra, "Algorithms and Bounds for Shortest Paths and Diameter in Faulty Hypercubes," *IEEE Trans. Comput.* vol. 4, pp. 713-718, Jun. 1993.

[TRS90] S. B. Tien, C. S. Raghavendra and M. A. Sridhar, "Reconfiguring embedded task graphs in faulty hypercubes by automorphism," in *Proc. Hawaii Int. Conf. Syst. Sci.*, Jan. 1990, pp. 91-100.

[TS79] Toueg and K. Steiglitz, "The design of small diameter networks by local search," *IEEE Trans. Comput.*, vol. C-28, pp. 537-542, Jul. 1979.

[TW91] N. -F. Tzeng, and S. Wei, "Enhanced hypercubes," *IEEE Trans. Comput.*, vol. 40, no. 3, pp 284-293, Mar. 1991.

[WC74] C. K. Wong and D. Coppersmith, "A combinatorial problem related to multimodule memory organization," *J. ACM*, vol. 21, no. 3, pp. 392-401, 1974.

[WF84] C. Wu and T. Feng. Eds., *Interconnection Networks for parallel and distributed processing.* Los Alamitos, CA:IEEE Computer Society, 1984.

[YN90] A. S. Yussef and B. Narahari, "Banyan-hypercube networks," *IEEE Trans. Parallel Distributed Syst.*, vol. 1, no. 2, pp. 160-169, Apr. 1990.

[YTR94] P.-J. Yang, S.-B. Tien, and C. S. Raghavendra, "Reconfiguration of rings and meshes in faulty hypercubes," *Journal of Parallel and Distributed Computing*, vol. 22, no. 1, pp. 96-106, Jul. 1994.

[YWL+88] C. Yang, J. Wang, J. Lee, and F. Boesch, "Graph theoretic reliability analysis for the Boolean n-cube networks," *IEEE Trans. Circuits Syst.*, vol. 32, no. 9, pp. 1175-1179, 1988.

LIST OF PUBLICATIONS OF THE AUTHOR ON SOME OF WHICH THIS THESIS IS PARTIALLY BASED

1. R. K. Das, K. Mukhopadhyaya and B. P. Sinha, "A new family of bridged and twisted hypercubes," *IEEE Transactions on Computers*, vol 43, no. 10, pp 1240-1247, Oct. 1994.

2. R. K. Das and B. P. Sinha, "Optimal communication algorithms in distributed loop networks," *Journal of Parallel and Distributed Computing*, to appear.

3. R. K. Das, K. Mukhopadhyaya and B. P. Sinha, "Bridged and twisted hypercubes with reduced diameters," in *Proc. International Conference on Parallel Processing*, pp. I-72 - I-75, Aug 17-21, 1992.

4. R. K. Das and B. P. Sinha, "Dense odd degree graphs," in *Proc. Fourth National Seminar on Theoretical Computer Science*, pp. 91 - 102, June 16 - 18, 1993.

5. S. Sen Gupta, R. K. Das, K. Mukhopadhyaya and B. P. Sinha, "A new family of low-diameter network topologies with multiple loops," in *Proc. First International Workshop on Parallel Processing*, pp. 41-46, December 26-31, 1994, Bangalore.