Embeddings, Fault Tolerance and Communication Strategies in $k$-ary $n$-cube Interconnection Networks

Yaagoub A. Ashir

A thesis submitted for the degree of Doctor of Philosophy

Department of Mathematics & Computer Science
University of Leicester

June 5, 1998
Acknowledgment

My sincere thanks and gratitude go to Prof. Iain Stewart, my supervisor, for his guidance, invaluable advice, encouragement, and his patience. I would like to thank the internal examiner, Prof. Rick Thomas, and the external examiner, Prof. Alan Gibbons, for making numerous useful comments and suggestions. Many thanks to University of Wales, Swansea and to Leicester University for giving me the opportunity to pursue my PhD studies during the past three years. My special thanks and gratitude go to University of Bahrain for supporting my postgraduate studies.

I am also grateful to my wife Deena for her help, encouragement and her contribution to the nice atmosphere during my study in UK. Finally, special thanks and appreciation to all my family members for their encouragement. In particular, to my brother Fawzi for his continuous support and help.
Abstract

The $k$-ary $n$-cube interconnection network $Q_n^k$, for $k \geq 3$ and $n \geq 2$, is $n$-dimensional network with $k$ processors in each dimension. A $k$-ary $n$-cube parallel computer consists of $k^n$ identical processors, each provided with its own sizable memory and interconnected with $2n$ other processors. The $k$-ary $n$-cube has some attractive features like symmetry, high level of concurrency and efficiency, regularity and high potential for the parallel execution of various algorithms. It can efficiently simulate other network topologies. The $k$-ary $n$-cube has a smaller degree than that of its equivalent hypercube (the one with at least as many nodes) and it has a smaller diameter than its equivalent mesh of processors.

In this thesis, we review some topological properties of the $k$-ary $n$-cube $Q_n^k$ and show how a Hamiltonian cycle can be embedded in $Q_n^k$ using the Gray codes strategy. We also completely classify when a $Q_n^k$ contains a cycle of some given length.

The problem of embedding a large cycle in a $Q_n^k$ with both faulty nodes and faulty links is considered. We describe a technique for embedding a large cycle in a $k$-ary $n$-cube $Q_n^k$ with at most $n$ faults and show how this result can be extended to obtain embeddings of meshes and tori in such a faulty $k$-ary $n$-cube.

Embeddings of Hamiltonian cycles in faulty $k$-ary $n$-cubes is also studied. We develop a technique for embedding a Hamiltonian cycle in a $k$-ary $n$-cube with at most $4n - 5$ faulty links where every node is incident with at least two healthy links. Our result is optimal as there exist $k$-ary $n$-cubes with $4n - 4$ faults (and where every node is incident with at least two healthy links) not containing a Hamiltonian cycle. We show that the same technique can be easily applied to the hypercube. We also
show that the general problem of deciding whether a faulty $k$-ary $n$-cube contains a Hamiltonian cycle is NP-complete, for all (fixed) $k \geq 3$.

Several communication algorithms for the $k$-ary $n$-cube network are considered; in particular, we develop and analyse routing, single-node broadcasting, multi-node broadcasting, single-node scattering, and total exchange algorithms. We also show how Hamiltonian cycles of the $k$-ary $n$-cube can be exploited to develop fault-tolerant multi-node broadcast and single-node scatter algorithms for the one-port I/O $k$-ary $n$-cube model, and how link-disjoint Hamiltonian cycles of the $k$-ary $n$-cube can be used to develop multi-node broadcast and single-node scatter algorithms for the multi-port I/O $k$-ary $n$-cube model.
Contents

1 Introduction 1
  1.1 Network Topologies ....................................................... 7
  1.2 Communication Aspects of Parallel Computers ............... 17
  1.3 Fault Tolerance ............................................................... 29
  1.4 Embeddings ................................................................. 31

2 The $k$-ary $n$-cube Interconnection Network 34
  2.1 Introduction ............................................................................ 34
  2.2 Definitions and Structures .................................................. 35
  2.3 Topological Properties ......................................................... 37
  2.4 Embeddings in $k$-ary $n$-cubes ........................................ 42

3 Embeddings of Cycles in $k$-ary $n$-cubes 48
  3.1 Introduction ................................................................. 48
  3.2 Recursive Structure of Gray Codes ................................. 49
  3.3 Embedding a Cycle of any Length .................................... 51

4 Fault-Tolerant Embeddings of Cycles, Meshes and Tori
   in $k$-ary $n$-cubes .......................................................... 58
  4.1 Introduction ................................................................. 58
  4.2 Fault-Tolerant Embeddings of Cycles ................. 59
4.2.1 Basic Results .............................................................. 60
4.2.2 Partitioning the Faulty $k$-ary $n$-cube ................... 68
4.3 Embeddings of Meshes and Tori ................................. 71

5  Embeddings of Hamiltonian Cycles in Faulty $k$-ary $n$-
cubes  ................................................................. 74
5.1 Introduction ............................................................... 74
5.2 Tolerating Faults in $k$-ary $n$-cubes ........................... 75
5.3 Tolerating Faults in Hypercubes ................................. 88
5.4 Complexity Issues ..................................................... 94

6  Communication Algorithms .............................................. 98
6.1 Introduction ............................................................... 98
6.2 Dimensional Routing ................................................... 99
6.3 Dimensional Single-node Broadcasting ......................... 101
6.4 Dimensional Multi-node Broadcasting ......................... 105
6.5 Dimensional Single-node Scattering ............................. 106
6.6 Dimensional Total Exchange ....................................... 107

7  Hamiltonian Cycles and Applications to Communication 111
7.1 Introduction ............................................................... 111
7.2 Multi-node Broadcasting for one-port I/O Model ........... 112
7.3 Single-node Scattering for one-port I/O Model ............... 113
7.4 Applications to Fault Tolerance .................................. 114
7.5 Applications to multi-port I/O Model ........................... 116
    7.5.1 Multi-node Broadcasting for multi-port I/O Model 120
    7.5.2 Single-node Scattering for multi-port I/O Model ... 121
8 Conclusions

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.1 Summary</td>
<td>124</td>
</tr>
<tr>
<td>8.2 Future Research</td>
<td>126</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

The original need for fast computation was in a number of contexts involving the solution of partial differential equations (PDEs), such as in computational fluid dynamics and weather forecasting, as well as in structural mechanics and image processing. In these applications, there is a large number of numerical computations to be performed. The desire to solve more and more complex problems has always ran ahead of the capabilities of computers of the time, and has provided a driving force for the development of faster computing machines.

Parallel processing with hundreds or thousands of microprocessors has become a viable alternative to conventional supercomputers and mainframes employing a handful of expensive processors. Several commercial machines with hundreds or thousands of processors have reached the marketplace in the last few years. These systems have spurred research in a number of areas such as the design of efficient network topologies, routing algorithms and protocols, communication interfaces, algorithms, and software tools.

Parallel computers in general can be roughly classified into two types:
multiprocessors with shared memory and multicomputers with non-shared or distributed memory \[11, 57, 80\]. There are also a variety of hybrid designs lying in between. The first type has global shared memory that can be accessed by all processors (see Fig. 1.1). To allow efficient access of the memory by several processors, the memory is divided into several memory banks. A processor can communicate with another processor by writing into the global memory and having the second processor read the same location in the memory using switching systems. The advantage of this architecture is that algorithm design is simple. Moreover, it enables us to make data access transparent to the user who may regard data as being held in a large memory which is readily accessible to any processor. However, as the number of nodes increases, the switching network becomes complex to build. The GF-11 Supercomputer \[9\], the Butterfly

\[\text{Figure 1.1: The Shared Memory Model.}\]
multiprocessor [74], and the Ultracomputer [49] are some examples of multiprocessors with shared memory.

In multicomputers with distributed memory, there is no shared memory and no global synchronization, but rather each processor has its own local memory (see Fig. 1.2). Processors communicate through an interconnection network (i.e., mesh, ring, torus, hypercube, etc.) consisting of direct communication links joining certain pairs of processors. Which processors are connected together is an important design choice. It would be best if all processors were directly linked to each other, but this leads to technological difficulties, or if the processors communicate through a shared bus, but this leads to excessive delays when the number of processors is very large due to bus contention. In multicomputers with distributed memory, communication is achieved by message-passing directly or through some intermediate processors, and computation is data driven. The main advantage of such architectures is the simplicity of

Figure 1.2: The Distributed Memory Model.
their design: the nodes are identical, or are of a few different kinds, and can therefore be fabricated at relatively low cost. Moreover, such machines can easily be made fault tolerant as, for example, healthy nodes can re-route messages so as to avoid failed nodes. Examples of multicomputers with distributed memory include the Cosmic Cube [84], the Ametek S/14 [7], the Ncube [19, 39], the iPSC [38, 39], the CM-200 [21] and the J-machine [27].

Many topologies have been proposed for parallel machines. Among these are the ring, the mesh, the tree, the torus, and the hypercube.

The hypercube, or the binary n-cube, is a popular interconnection network for parallel processing as it possesses a number of topological properties which are highly desirable in the context of parallel processing. For example: it contains a Hamiltonian cycle; many other networks can be efficiently embedded into a hypercube; and its symmetry results in rich communication properties (see, for example, [11, 13, 61, 79] and the references therein). Consequently the hypercube has formed the base topology of a number of parallel machines including the Cosmic Cube [84], the Ametek S/14 [7], the Ncube [19, 39], and the iPSC [38, 39].

However, one drawback to the hypercube is that the number of links incident with each node is logarithmic in the size of the network. While this is not a problem for small hypercubes, it can present some difficulties for very large machines, e.g., machines with tens of thousands of processors. VLSI systems are wire-limited [83] and although hypercubes provide relatively small diameter networks, the property of high degree is not consistent with the realities of VLSI technology. Networks with high degree require more and longer wires than do low degree networks. Thus high degree networks in general cost more and run more slowly.
than low degree networks. It has also been shown that low degree networks achieve lower latency and better hot-spot throughput than do high degree networks [28, 62].

One means proposed to alleviate this problem is to build parallel machines whose underlying topology is that of the $k$-ary $n$-cube $Q_n^k$ (where $k \geq 3$ and $n \geq 2$). A $k$-ary $n$-cube $Q_n^k$ parallel machine consists of $k^n$ identical processors. Each processor has its own local memory and is connected to $2n$ other processors. In order to overcome the problem of the high degree of binary $n$-cubes, we can increase $k$ and decrease $n$ and so obtain lower degree $k$-ary $n$-cubes with the same number of processors. For example, the 4096 nodes in a binary 12-cube with node degree 12 and a total of 24576 links can be interconnected as a 16-ary 3-cube with node degree 6 and a total of only 12288 links.

The $k$-ary $n$-cube $Q_n^k$ has formed the underlying topology of the Mosaic [85], the Cray T3D [70], the iWARP [16] and the J-machine [27] parallel machines (see also [53]). It turns out that many computations in linear algebra and partial differential equations can be performed efficiently on machines having a $k$-ary $n$-cube as their underlying topology.

The $k$-ary $n$-cube $Q_n^k$ is a network with $n$ dimensions and $k$ nodes in each dimension. The $k^n$ nodes of the $k$-ary $n$-cube are indexed by \{0,1,\ldots,k-1\}^n, and there is a link $((x_1,x_2,\ldots,x_n),(y_1,y_2,\ldots,y_n))$ iff there exists $j \in \{1,2,\ldots,n\}$ such that $x_j - y_j = 1 \pmod{k}$ and $x_i = y_i$, for every $i \in \{1,2,\ldots,n\} \setminus \{j\}$ (for example, $Q_2^k$ is a $k \times k$ mesh with wrap-around, $Q_1^k$ is a cycle of length $k$, and $Q_n^2$ is a binary $n$-cube).

In this thesis, we consider various properties of the $k$-ary $n$-cube $Q_n^k$ interconnection network (where $k \geq 3$ and $n \geq 2$). For example, we consider the existence of cycles in both healthy and faulty $k$-ary $n$-cubes.
We derive results that are both of theoretical interest and applicable to communication algorithms. In more detail, this thesis is organized as follows.

- The remainder of this chapter presents some examples of common network topologies and reviews some aspects of communication strategies and fault tolerance. It also reviews some basic terminology of network embedding.

- Chapter 2 includes some structural and topological properties of the $k$-ary $n$-cube network. It also reviews some basic results from the literature on embeddings of common network topologies such as a Hamiltonian cycle (using the Gray code strategy), a binary tree, a mesh, and a hypercube into the $k$-ary $n$-cube.

- Chapter 3 presents a recursive structure of $k$-ary Gray codes and completely classifies when a $k$-ary $n$-cube $Q_n^k$ contains a cycle of some given length.

- Chapter 4 describes a technique for embedding a large cycle, a mesh and a torus in a $Q_n^k$ with at most $n$ faulty nodes or links.

- Chapter 5 develops a technique for embedding a Hamiltonian cycle in a $k$-ary $n$-cube with at most $4n - 5$ faulty links where every node is incident with at least two healthy links. We show in this chapter that the same technique can be easily applied to the hypercube. We also show that given a faulty $Q_n^k$, the problem of deciding whether there exists a Hamiltonian cycle is NP-complete.

- Chapter 6 develops some efficient communication algorithms. In this chapter, we develop and analyse routing, single-node broad-
casting, multi-node broadcasting, single-node scattering, and total exchange. All our algorithms are deterministic and dimensional for one-port I/O \( k \)-ary \( n \)-cube model.

- Chapter 7 shows how Hamiltonian cycles of the \( k \)-ary \( n \)-cube network can be exploited to develop fault-tolerant multi-node broadcast and single-node scatter communication algorithms for the one-port I/O \( k \)-ary \( n \)-cube model. We also show in this chapter how link-disjoint Hamiltonian cycles of the \( k \)-ary \( n \)-cube can be used to develop multi-node broadcast and single-node scatter algorithms for machines that support multi-port I/O model.

- Chapter 8 concludes the thesis by giving a summary of the results and stating some open problems for future research.

### 1.1 Network Topologies

A network is usually modelled by an undirected graph, called the network topology, where the edges represent communication links and the nodes represent either processors or switches. A network is either static or dynamic [38, 95]. In a static (or fixed) network, the nodes represent processors. Examples of static networks include rings, trees, meshes, tori, and hypercubes (all to be defined later). These networks are also sometimes referred to as direct networks because the interconnecting links directly connect nodes as opposed to being switched dynamically. Static networks have been the preferred means of interconnection in distributed memory multicomputers, but have also been used in shared memory multiprocessor interconnection.

In a dynamic (or reconfigurable) network, some nodes represent switches.
Multistage interconnection networks are prime examples of dynamic networks and include the omega, inverse omega, and baseline networks [74, 95].

The topology of an interconnection network determines many architectural features of that machine and affects several performance metrics. Although the actual performance of a network depends on many technological and implementation factors, several topological properties and metrics can be used to evaluate and compare different topologies in a technology-independent manner. Most of these properties are derived from the graph model of the network topology.

Topologies are usually evaluated in terms of their suitability for some standard communication tasks. The following are some typical factors to be considered in the design of interconnection networks [11, 81, 95].

- The node degree.
  The node degree represents the number of I/O ports per processing node. It should be small to reduce the implementation cost.

- The diameter.
  The diameter of a network is the maximum distance between any pair of nodes. Here the distance of a pair of nodes is the minimum number of links that have to be traversed to go from one node to the other.

- The symmetry and the regularity.
  A regular network is defined as a network in which every node has the same degree. A symmetric (or homogeneous) network is one in which the topology looks identical when viewed from every node or every link. This definition gives rise to two types of symmetry:
node symmetry and link symmetry. In graph-theoretic terms, a network is node-symmetric if, for every pair of nodes \( a \) and \( b \), an automorphism of the network can be found that maps \( a \) into \( b \). The definition of link symmetry is identical.

The principal advantage of symmetry in a network lies in the ease of routing data in the network. This allows all nodes to use the same routing algorithm. The task of path-selection is also often simplified. In addition, common data-movement operations such as broadcasts and multicasts can be implemented easily and efficiently on symmetric topologies.

In a regular network, every node has the same degree. Often, networks can be parameterized by \( N_1, N_2, \) and so on, so that the different networks are very closely related; for example, \( N_i \) is usually contained in \( N_j, \) for \( j > i \). We often refer to this set of networks as a network (as we do with the hypercube). In such a case, we say that the network has a constant degree if every \( N_i \) is regular of the same degree (this is not the case with the hypercube). Constant degree networks are easy to expand and are often suitable for VLSI implementation. Also, as we increase the size of the network, i.e., replace \( N_i \) with \( N_j, \) for \( j > i \), the network interface of a node remains unchanged.

- The connectivity.

The connectivity of a network provides a measure of the number of "independent" paths connecting a pair of nodes. The term node connectivity refers to the minimum number of nodes that need to be removed to disconnect the network, and link connectivity is the minimum number of links that need to be removed to achieve the same.
The minimum of node- and link-connectivities is sometimes referred to as the *connectivity* of the network. According to Menger's theorem [90], the node connectivity is equal to the minimum among the maximum number of node-disjoint paths between pairs of nodes. These disjoint paths can be exploited to maintain communication in the face of several node failures. Another important point is that if the network has node connectivity \( k \) then communication between any two nodes can be parallelized by making use of at least \( k \) paths with no pair of these paths having a node in common. Thus, a long message can be sent from node \( a \) to node \( b \) by splitting it into several packets, and by sending these packets in parallel on the disjoint paths connecting \( a \) and \( b \). This reduces the communication time between any pair of nodes by a factor at least \( k \). Moreover, disjoint paths can be exploited to improve performance during normal operation by avoiding congested network elements, and achieve fault-tolerance.

- **The bisection width.**
  
The bisection width of a network is the minimum number of links that must be removed to partition the original network of \( N \) nodes into two subnetworks of \( N/2 \) nodes each (if \( N \) is odd, the subnetworks are of size \((N + 1)/2\) and \((N - 1)/2\)). The bisection width is useful in estimating the area required for a VLSI implementation of the network.

- **The reliability.**
  
The reliability of a network is the probability that all non-faulty nodes can communicate with one another through fault-free paths.
• The scalability and the expandability.

Commercial multiprocessors and multicomputers must usually be designed to allow expandability over a certain range. A network is easy to expand when it requires no changes to any node when more nodes are added. A network is scalable if it continues to yield the same performance per processor as the number of processors increases.

• The flexibility.

The network topology should be rich enough to allow frequently used topologies to be embedded so that algorithms designed for other architectures can be simulated.

• The partitionability.

The partitionability of the network into subnetworks is important for the support of multiusers and multitasking.

We now consider some common network topologies.

Complete Graph

An $N$ processor complete graph is a network where every processor is directly linked to every other processor (see Fig. 1.3). Such a network can be implemented by means of a bus which is shared by all processors, or by means of some type of crossbar switch. The diameter of this network is 1 and the connectivity is $N - 1$. Clearly this is an ideal network in terms of flexibility and fault-tolerance. Unfortunately, when the number of processors is very large, a crossbar switch becomes very costly, and a bus involves large queueing delays. However, complete graphs are
frequently used to connect small numbers of processors in clusters in a hierarchical network, where the clusters are themselves connected via some other type of communication network.

**Linear Processor Array**

A linear processor array architecture consists of $N$ processors, $p_0, p_1, \ldots, p_{N-1}$, and there is a link for every pair of successive processors (see Fig. 1.4). Processor arrays structured for numerical execution have often been employed for large-scale scientific calculations, such as image processing and nuclear energy modelling. The diameter of this network is $N - 1$ and the connectivity is 1. Although the total number of the communication links and the node degree are minimum compared to the other network topologies, the diameter and connectivity properties of
this network are the worst possible. When one processor or link becomes faulty, the network becomes disconnected.

Ring

A ring or a cycle processor architecture (see Fig. 1.5) consists of $N$ processors, $p_0, p_1, \ldots, p_{N-1}$, where $p_i$ is linked to $p_{i+1 \mod N}$. The diameter of this network is $\lfloor N/2 \rfloor$ and the connectivity is 2. The diameter of the ring topology architecture can be reduced by adding chordal connections. Using chordal connections can increase a ring-based architecture's fault tolerance. Ring topologies are most appropriate for a small number of processors executing algorithms not dominated by data communication.

Tree

Tree topology architectures have been constructed to support divide-and-conquer algorithms for searching and sorting, image processing algorithms, and dataflow and reduction programming paradigms. Although a variety of tree-structured topologies have been suggested, complete binary trees are the most analysed variant. A complete binary tree (see Fig. 1.6) with $n$ levels and $N = 2^n - 1$ processors has diameter $2(n - 1)$

Figure 1.5: Ring.
and connectivity 1. The binary tree lends itself to area-efficient implementation in VLSI using the H-tree layout [52].

![Complete Binary Tree](image)

Figure 1.6: Complete Binary Tree.

Although the total number of communication links in a tree network is minimal \((N - 1)\), one disadvantage of a tree is its low connectivity; the failure of any one of its links creates two subsets of processors that cannot communicate with each other. Several strategies have been employed to reduce the communication diameter of tree topologies. Example solutions include adding additional links so that the nodes on the same tree level are connected in the form of a linear array.

Mesh

Mesh-connected processor arrays (see Fig. 1.7) have found wide application in both commercial and experimental machines. The nodes of a \(d\)-dimensional mesh with \(n_i\) points along the \(i\)th dimension are the \(d\)-tuples \((x_1, \ldots, x_d)\) where each of the coordinates \(x_i, i = 1, \ldots, d,\) takes an integer value from 0 to \(n_i - 1\). The links are the pairs \(((x_1, \ldots, x_d), (x'_1, \ldots, x'_d))\) for which there exists some \(i\) such that \(|x_i - x'_i| = 1\) and \(x_j = x'_j\) for all \(j \neq i\). The diameter of a \(d\)-dimensional mesh-connected network, with \(N = \prod_{i=1}^{d} n_i\) processors, is \(\sum_{i=1}^{d} (n_i - 1)\) [95], which can be much smaller than the diameter of a ring and much larger than the diameter of a binary tree with the same number of processors.
A variation with smaller diameter is the torus (see Fig. 1.8) which is a mesh network with wraparound; that is, $|x_i - x'_i| = 1$ in the above definition is relaxed to $|x_i - x'_i| = 1 \mod n_i$. The diameter of a $d$-dimensional torus is $\sum_{i=1}^{d} \lfloor n_i/2 \rfloor$ [11, 95].

Meshes and tori are quite popular networks and have been widely studied. Two-dimensional meshes are ideally suited for many applications such as signal/image processing, matrix computations, numerical solution of differential equations, aerodynamic structure analysis, and computer vision [11, 62].
Hypercube

The binary $n$-cube, commonly called the hypercube, is a popular interconnection network for commercial and experimental systems owing to its relative simplicity and rich interconnection structure (see Fig. 1.9).

Figure 1.9: 4-Dimensional Hypercube.

The first working hypercube system was built at Caltech in 1983 [84]. Since then, many commercial and experimental hypercube computers have been constructed. The hypercube network on $N = 2^n$ nodes is obtained by labelling each node by an $n$-bit binary number and connecting nodes whose addresses differ in one bit exactly. The distance between two nodes along a shortest path in the hypercube (often referred to as the Hamming distance) is the number of bits in which their binary addresses differ. The diameter of a hypercube is $n = \log_2 N$ and the connectivity is also $n$ [78]. The interconnection structure of the hypercube allows the efficient implementation of a large number of parallel algorithms [11, 47, 61, 74]; and the hypercube can simulate a variety of
other common topologies such as rings, meshes [78], and trees [57, 99].

The reader should refer to [41, 47, 61, 74] for other network topologies like the butterfly, cube-connected-cycles, shuffle-exchange, and de Bruijn networks.

1.2 Communication Aspects of Parallel Computers

In this section, we consider some basic communication aspects of multicomputers with distributed memory. In interprocessor communication where several links must be traversed, several issues involving data transmission and routing mechanisms requirements must be addressed. In a network, when node $a$ sends a message to node $b$, a path through the network along which the message will travel must be chosen. This process is called routing.

A message may be broken into one or more segments called packets for transmission. A packet is the smallest unit that contains routing and sequencing information. There are two advantages in dividing a message into packets. Assume that a message is to be transmitted over a path of $k > 1$ communication links and a processor must store the entire message before it can be processed and retransmitted. Also, each packet takes one unit of time to travel along a link. Dividing the message into $m$ packets, namely $p_1, p_2, \ldots, p_m$, and transmitting them sequentially in a pipeline fashion over the $k$-link path will reduce the delay time from $mk$, if the message is transmitted as a whole packet, to $m + k - 1$. This is because if the message is transmitted as a whole packet then it will take $m$ units of time to transmit the message over one link, and hence a total of $mk$
time units. Now assume that each packet is transmitted sequentially in a pipeline fashion over the \( k \)-link path. Then \( p_1 \) will reach the destination in time \( k \); \( p_2 \) will reach the destination in time \( k + 1 \); \( p_3 \) will reach the destination in time \( k + 2 \); and so on. Hence, \( p_m \) will reach the destination in time \( k + m - 1 \).

The second advantage of dividing a message into packets is that if the network has connectivity \( k \) then communication between any two nodes can be parallelized by making use of at least \( k \) paths with no pair of these paths having a node in common. Thus, by splitting the message into several packets, and sending these packets in parallel along the disjoint paths connecting the source and the destination nodes, the communication time between any pair of nodes can be reduced by a factor at least equal to the connectivity of the network. This is useful for large messages (when the number of packets is greater than the accumulated length of the disjoint paths).

**Communication Delays**

One main problem in interconnection networks is to reduce as far as possible the time a message takes to travel from one node to another. Communication delays can be divided into four parts.

(a) *Communication processing time*, which is the time required to prepare information for transmission (i.e., assembling information in packets, appending addressing and control information to the packets, selecting a link on which to transmit each packet, moving the packets to the appropriate buffers, etc.).

(b) *Queueing time*, which is the time the message waits in queue prior to the start of its transmission; for some reason such as waiting
for an unused communication link or ensuring the availability of needed resources (such as buffer space at its destination).

(c) *Transmission time*, which is the time required for transmission of all the bits of the packet.

(d) *Propagation time*, which is the time between the end of transmission of the last bit of the packet at the transmitting processor, and the reception of the last bit of the packet at the receiving processor.

**Message-Routing Schemes**

A network topology must allow every node to send packets to every other node. When the topology is incomplete, routing determines the path selected by a packet to reach its destination. Efficient routing is critical to the performance of multicomputer networks [44, 68].

A routing algorithm is termed *deterministic* if the path selected does not depend on the current network conditions. In deterministic routing, the selected path is entirely determined by the source and destination addresses. That is, the mapping from pairs of source-destination addresses to the path to be followed is a single valued function. This has the advantage of simplicity, but is unable to adapt to network conditions such as congestion and failures.

A routing algorithm is *dimensional* if a path chosen takes the message through one dimension at a time. An example for deterministic dimensional routing is the *row-column routing* (also called *X-Y routing*) in a 2-dimensional mesh which always routes a message along the row first and then along the column, thus using a deterministic path for a given source-destination pair. Another well-known dimensional routing algo-
rithm is the *e-cube* routing algorithm for hypercubes [89]. This algorithm routes packets in a binary *n*-cube in a fixed order of dimensions (usually in increasing or decreasing order).

Alternatively, a routing technique is *adaptive* if, for a given source and destination, the path taken by a particular packet depends on dynamic network conditions, such as the presence of faulty or congested channels. With adaptive routing, the paths can be modified to avoid faulty network elements. Many recent researchers have proposed algorithms for adaptive routing in multicomputer networks [24, 26, 31, 35, 44, 62, 76, 87, 101].

A routing algorithm is said to be *minimal* if the path selected is one of the shortest paths between the source and destination pair. Using a minimal routing algorithm, every channel visited will bring the packet closer to the destination. A *nonminimal* routing algorithm allows packets to follow a longer path, usually in response to current network conditions. This behaviour can lead to a situation known as *livelock*. Livelock occurs when a message circulates endlessly in the network, never reaching its destination. Protocols that misroute in this fashion must have some methods of dealing with livelock [50, 95].

A network consists of many channels and buffers. *Flow control* deals with the allocation of channels and buffers to a packet as it travels along a path through the network. A resource collision occurs when a packet cannot proceed because some resource that it requires is held by another packet. Whether the packet is dropped, blocked in place, buffered, or re-routed through another channel depends on the flow control policy. Flow control techniques attempt to regulate the movement of packets from node to node so as to utilize the network resources as efficiently as possible.
A simple approach to implement network flow control is *store-and-forward*. In this method, the entire packet is buffered at each intermediate node and forwarded to the next node in its path when the desired outgoing link becomes available. This is simple to implement, but the buffering at each intermediate node wastes memory and causes unnecessary delays. An improvement over the store-and-forward approach, called *virtual cut-through*, avoids this problem by buffering packets only when they encounter a busy link [55]. With virtual cut-through, the forwarding of a packet can commence as soon as the header bits are received if the outgoing link requested is free. The packet is buffered at the node only if the requested link is busy. This technique has been shown to result in improved performance over the store-and-forward approach, particularly under light traffic conditions [55]. Cut-through routing was used in the torus routing chip implemented at Caltech [32].

In both the store-and-forward and virtual cut-through approaches, a blocked packet stays in a buffer at an intermediate node, waiting for the outgoing link to be free. Alternatively, the buffering can be reduced to a minimum if the blocked packet is allowed to stay on the partial path it has already traversed. That is, parts of the packet can stay in multiple nodes along the path, holding the links between them. The *wormhole* approach, originally proposed by researchers at Caltech [33], is based on this idea.

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

When the header flit is sent along a communication channel, the remaining flits follow in a pipeline fashion. Should the leading flit be blocked because the communication channel ahead is occupied, the remaining flits in the message are also blocked. An advantage of wormhole routing is that message latency due to transit time (fly time) is less dependent on path length [28] (see [40, 68] for more details about wormhole routing).

Both store-and-forward and wormhole routings are susceptible to deadlock. Deadlock in store-and-forward routing occurs when no message can advance toward its destination because the queues of the message system are full. Consider the example shown in Fig. 1.10. The queue of each node in the 4-cycle is filled with messages destined for the opposite node. No message can advance toward its destination; thus, the cycle is deadlocked. In this locked state, no communication can occur over the deadlocked channels until exceptional action is taken to break the deadlock.

Many deadlock-free routing algorithms have been developed for store-and-forward computer communication networks [46, 51, 65]. These algorithms are based on the concept of a structured buffer pool. The message buffers in each node of the network are partitioned into disjoint classes, and allocating them to packets based on some buffer-allocation scheme. One approach is to allocate the buffers based on the number of hops a packet has travelled [51, 65]. That is, a packet arriving at a node after \(i\) hops is stored in a buffer belonging to class \(i\). This requires at least \(h + 1\) buffer classes in a node, where \(h\) is the maximum number of hops.
A disadvantage of wormhole routing is its use of a blocking buffering scheme. That is, as long as the header can advance, so too do the following flits. If the header cannot advance because, for example, another worm holds the path, all flits in the first message hold their position. This blocks another message from using the path. For this reason, wormhole routing is susceptible to deadlock, particularly in toroidal interconnection networks [18].

An example of deadlock can be seen in Fig. 1.11. This figure illustrates four simultaneous communication requests in a ring network with 4 nodes, one proceeding from node A to node C through node B, the second proceeding from node B to node D through node C, the third proceeding...
from node C to node A through node D, and the fourth request proceeding from node D to node B through node A. If a "reserve-and-hold" policy is used, allowing the partial path to be held while waiting for an outgoing channel, then we have the following situation: node A is holding channel $C_1$ and waiting for channel $C_2$, node B is holding channel $C_2$ and waiting for channel $C_3$, node C is holding channel $C_3$ and waiting for channel $C_4$, and node D is holding channel $C_4$ and waiting for channel $C_1$. This circular wait causes deadlock.

There have been several solutions proposed to the problem of deadlock in wormhole routed networks. One proposal, the Turn Model [48], restricts the directions a worm can turn. This model has the advantage of not requiring extra hardware support, but it can only be used to address the problem of deadlock. A second, more popular, approach is the use
of virtual channels [33]. In this approach, multiple virtual channels are time multiplexed over a physical channel (see Fig. 1.12). Each physical channel has the same number of virtual channels assigned to it, and each virtual channel has its own input and output flit buffer.

![Diagram 1](image1.png)

(a)

![Diagram 2](image2.png)

(b)

Figure 1.12: Virtual Channels: (a) Packet B is blocked behind packet A. (b) Virtual channels provide additional buffers allowing packet B to pass blocked packet A.

Virtual channels increase the hardware complexity of a router because of the need for extra buffers and multiplexor and demultiplexor hardware. However, in addition to preventing deadlock, virtual channels can be used to improve throughput of the network, decrease latency, or provide adaptivity in the routing algorithm [29, 30, 31, 62, 101]. The reader should refer to [47, 67, 94] for other message-routing schemes like the permutation, randomised and hot potato routing schemes.
Common Communication Primitives

Besides one-to-one routing, there are other communication primitives that are of importance in executing certain computational algorithms on a multicomputer network [11, 95]. These communication primitives involve

- **one-to-all** (or **single-node broadcasting**), where one processor wishes to send the same data item to every other processor,
- **one-to-all personalized** (or **single-node scattering**), where one processor wishes to send a different data item to every other processor,
- **all-to-all** (or **multi-node broadcasting**), where every processor wishes to send the same data item to every other processor, and
- **all-to-all personalized** (or **total exchange**), where every processor wishes to send a different data item to every other processor.

Communication algorithms can be implemented in either *d*-port I/O or *multi*-port I/O model. In a *d*-port I/O model, a processor can transmit a packet along at most *d* incident communication links and can simultaneously receive a packet along at most *d* incident communication links, whereas in a *multi*-port I/O model all incident communication links of a processor can be used simultaneously for packet transmission and reception. Broadcasting is a common operation in parallel algorithms. It is used in a variety of linear algebra algorithms such as matrix-vector computations, LU-factorization, transitive closure, and database queries. The reverse of broadcasting is the **global combine** operation, in which each processor has a value which needs to be sent to a specific processor; for example, for finding the maximum or minimum or global sum. In the
case of personalized broadcasting, a single processor has a vector of \( N \) values and the \( i \)th value needs to be sent to the \( i \)th processor. Personalized broadcasting is used in matrix computations, where a column of data stored in one processor is to be distributed to \( N \) other processors. Multi-node broadcasting and total exchange communication algorithms occur in matrix computations and in neural network simulations.

In the case of single-node broadcasting, Sullivan et al. [89] have given what is now the standard algorithm, called the \( e \)-cube algorithm, for broadcasting in hypercube multicomputers. The \( e \)-cube algorithm, where the source processor is \( p_0 \), consists of \( n \) steps and is as follows. In the first step, \( p_0 \) sends the message it intends to broadcast to its adjacent processor in dimension 1. In step \( i \), for \( i = 2, 3 \ldots, n \), each processor sends the broadcast message it has just received to its adjacent processor in dimension \( i \) as does the source. At the end of step \( n \), all the processors of the hypercube will have the message. Since the diameter of the hypercube is \( n \), the \( e \)-cube algorithm is optimal. Fig. 1.13 shows the broadcast tree resulting from the \( e \)-cube algorithm in the 3-dimensional hypercube where the source processor is 000.

Algorithms for the communication primitives in a binary \( n \)-cube were first considered in [79], where the effect of the packet overhead and the data rate on the transmission time is also discussed. In this work, the hypercube links are assumed to be unidirectional and the model is one-port I/O; this increases the algorithm execution times by a factor of \( 2n \) compared to a bidirectional multi-port I/O model.

The communication primitives for the hypercube have also been considered in [10] and [54] under the bidirectional multi-port I/O model.
In [54], optimal and nearly optimal algorithms are given on the basis of a different model of communication. This model differs from the model of [10] in that it quantifies the effects of setup time (or overhead) per packet, while it allows packets to have variable length and to be split and be recombined prior to transmission on any communication link. The model of [10] may be viewed as the special case of the model of [54] in which packets have a fixed length and splitting and combining of packets is not allowed. Under the assumptions of the model of [10], the algorithms given in [54] for single-node scatter, multi-node broadcast, and total exchange are not optimal although some of them are optimal up to a small additive term but are optimal when the dimension of the hypercube $n$ is a prime number (they are also optimal if each node has a multiple of $n$ packets to send to each destination node). In contrast, the corresponding algorithms in [10] are optimal for all $n$ and are unimprovable as far as
time and communication requirements are concerned. For recent research concerning these communication problems, the reader is referred to [8, 15, 88, 91, 93, 97].

1.3 Fault Tolerance

In massively parallel computer systems, as the size of the system increases, so does the probability of component failure especially in operating environments such as mission critical defence applications, spaceborne systems, and environmental controls [37]. Fault-tolerant networks are essential to the reliability of parallel computer systems. A fault-tolerant network has the ability to route information even if certain network components (i.e., processors, switches, and/or communication links) fail.

The techniques often used for network fault tolerance are either: software based, such as adaptive routing, which makes use of multiple source-destination paths to avoid faulty components and multiple passes through the network (often used in omega-like multistage network); or hardware based, such as enhancing the network with additional hardware (such as links and switches). These techniques provide enough redundancy in the original network design to tolerate a certain number of faults [66, 73].

Faults can be classified into three types in terms of their duration: transient, intermittent, and permanent [66]. Transient faults persist only for a finite length of time (usually short) and are non-recurring. Intermittent faults also last for a finite period of time, but are recurring. A permanent fault is an irreversible condition. Different design techniques are sometimes used to tolerate these faults. Transient and intermittent faults, for example, may be tolerated by repeating the operation on the
faulty device until the fault disappears (called *redundancy in time*). Tolerating permanent faults, on the other hand, requires some form of hardware redundancy in the system.

A system is repaired either by replacing the failed component by a spare or by reconfiguring the system structure or work load distribution to circumvent the component. Component replacement restores the system to full operation but requires redundant components not used for normal operations.

Many reconfiguration strategies use all system components to perform useful work. When a fault occurs, system performance is degraded by redistributing the work load among the remaining resources. Or system redundancy can be reduced, affecting subsequent fault tolerance.

A failed component may be physically or logically removed from a system. Logical removal is accomplished by switching off the component’s output into an inactive state, or instructing all units to ignore or bypass it.

Two measures are commonly used to quantify the ability of a system to continue its function in the presence of faults: the *reliability* $R(t)$ is the probability that the system does not fail in the interval $(0, t)$; and the *availability* $A(t)$ is the average fraction of time the system was operational in the interval $(0, t)$. Reliability is important for mission-critical systems, or systems where the result of a system-failure would be catastrophic. Availability is a useful measure for commercial data-processing systems where a repair is feasible in the event of failure.

The objective of fault-tolerant computing is to develop and certify computing systems which perform in a satisfactory fashion in the presence of faults. Also, it is desirable that the system remains available
to execute parallel tasks during the repair and replacement of faulty components. However, there is no broadly accepted methodology for fault tolerant design or analysis [98]. Fault tolerant design requires an awareness of what can go wrong throughout the design process. Failure domains are bigger than design domains. An economic model is needed that recognizes not only the value and cost of functionality, but also the value and cost of dependability.

A large volume of literature exists on the subject of fault-tolerance in interconnection networks. Most of the research on the subject falls into two categories [95]: methods to introduce redundancy in a known topology, motivated by the low connectivity of certain network topologies such as the tree and ring; and exploiting the inherent redundancy of the topology, motivated by the high connectivity and, hence, the existence of multiple routing paths between pairs of nodes such as the hypercube and the torus. For details on this subject the reader is referred to [20, 25, 56, 58, 60, 71, 75].

1.4 Embeddings

The problem of allocating processes to processors in a multicomputer system is known as the mapping problem. A parallel program could be represented as a guest network $G$, where nodes denote processes and links denote communication between processes. A multicomputer system is represented by a host network $H$, where nodes denote processors and links denote communication links between processors. The mapping problem could be modelled as a network embedding problem, mapping statically known networks, or guest networks, onto a fixed-connection network. This section reviews some basic terminology of network embedding.
Let $G$ and $H$ be two network topologies where $G$ is the guest network and $H$ is the host network. Let $V_G, E_G, V_H$ and $E_H$ denote the node and link sets of $G$ and $H$, respectively, and let $P_H$ denote the set of all paths in $H$. That is, $(x_1, x_2, \ldots, x_n) \in P_H$ if $x_i \in V_H$ and $(x_i, x_{i+1}) \in E_H$ for $1 \leq i < n$. Then an embedding of $G$ in $H$ is a pair $(f_V, f_E)$ where $f_V : V_G \rightarrow V_H$ and $f_E : E_G \rightarrow P_H$. Also,

$$(a, b) \in E_G \Rightarrow f_E(a, b) = (x_1, \ldots, x_n)$$

such that

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

Given networks $G$ and $H$ with an embedding $(f_V, f_E)$ of $G$ into $H$, the following terms are used to describe the embedding. For more information, see [61].

- **Dilation.**
  
  The dilation of an embedding is the length of the longest path in $H$ that is associated with a link in $G$ by $f_E$.

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

- **Congestion.**
  
  The congestion of an embedding is the maximum number of times a single link of $H$ belongs to paths in $H$ associated with links in $G$ by $f_E$.

- **Load.**
  
  The load of an embedding is the maximum number of nodes of $G$ associated with a single node of $H$ by $f_V$. 

32
Clearly, the best embeddings are those for which the dilation, expansion, congestion, and load are all small. This is because these four measures bound the speed and efficiency with which $H$ can simulate $G$. If all the four measures are constant, then $H$ will be able to simulate $G$ with constant slowdown. Hence, by developing a good mapping function from one interconnection topology to another, one can simulate the algorithms designed for the former topology on a parallel machine that uses the later topology without much loss of efficiency. If an embedding of a network $G$ into a network $H$ can be found having dilation, congestion and load equal to one, then $G$ is isomorphic to a subnetwork of $H$. 

33
Chapter 2

The \( k \)-ary \( n \)-cube Interconnection Network

2.1 Introduction

In this chapter, we define the \( k \)-ary \( n \)-cube \( Q_n^k \) network and detail its properties. The binary \( n \)-cube has been extensively studied (see [11, 13, 61, 78]), so we restrict ourselves on the \( k \)-ary \( n \)-cube where \( k > 2 \). We give some definitions and describe the recursive structure in Section 2.2. We discuss the topological properties and state the number of node-disjoint paths between any two nodes in a \( Q_n^k \) in Section 2.3. In Section 2.4, we consider some embeddings of common network topologies such as a Hamiltonian cycle (using the Gray code strategy), a binary tree, a mesh, and a hypercube into a \( k \)-ary \( n \)-cube.
2.2 Definitions and Structures

In order to be able to define the \( k \)-ary \( n \)-cube network, we begin this section by introducing some definitions from coding theory [72]. Then, we define the \( k \)-ary \( n \)-cube network and show that it can be built recursively from lower dimensional cubes.

**Definition 2.1** Let \( A = (a_n, a_{n-1}, \ldots, a_1) \) and \( B = (b_n, b_{n-1}, \ldots, b_1) \) be two \( n \)-tuples where \( a_i, b_i \in \{0, 1, \ldots, k - 1\} \). The Hamming distance between \( A \) and \( B \), denoted \( D_H(A, B) \), is the number of positions in which they differ. The Lee distance between \( A \) and \( B \), denoted \( D_L(A, B) \), is defined as:

\[
D_L(A, B) = \sum_{i=1}^{n} \text{dis}(a_i, b_i),
\]

where

\[
\text{dis}(a_i, b_i) = \min(|a_i - b_i|, k - |a_i - b_i|).
\]

**Example 2.2** Let \( k = 5 \) and \( n = 6 \). Let \( A = (3, 0, 1, 2, 3, 4) \) and \( B = (3, 0, 4, 0, 0, 0) \). Then,

\[
D_H(A, B) = 4, \quad \text{and} \quad D_L(A, B) = \text{dis}(4, 0) + \text{dis}(3, 0) + \text{dis}(2, 0) + \\
\text{dis}(1, 4) + \text{dis}(0, 0) + \text{dis}(3, 3) = 1 + 2 + 2 + 2 + 0 + 0 = 7.
\]

Clearly, \( D_L(A, B) = D_H(A, B) \) when \( k = 2 \) or \( 3 \), and \( D_L(A, B) \geq D_H(A, B) \) when \( k > 3 \).

Intuitively, the \( k \)-ary \( n \)-cube \( Q_n^k \) network model is an \( n \)-dimensional, \( k \) nodes in each dimension, mesh-connected network with wraparound connections. In more detail, a \( k \)-ary \( n \)-cube \( Q_n^k \) network is a \( 2n \)-regular
network containing $k^n$ nodes. Each node is labelled with a distinct $n$-tuple $(a_n, a_{n-1}, \ldots, a_1)$ where $a_i \in \{0, 1, \ldots, k-1\}$. Node labels are usually written either as $n$-tuples $(a_n, a_{n-1}, \ldots, a_1)$ or as $(a_n a_{n-1} \ldots a_1)$. Two nodes $U$ and $V$ in the $k$-ary $n$-cube are adjacent if and only if $D_L(U, V) = 1$. Consequently, for $k > 2$, the $k$-ary 1-cube is a cycle of length $k$ and the $k$-ary 2-cube is a $k \times k$ mesh with wraparound. The $i$th digit of the label $a_i$ represents the node's position in the $i$th dimension.

From the definition of Lee distance, it can be seen that every node in $Q_n^k$ shares a link with two nodes in every dimension, resulting in a network of degree $2n$. In addition, the shortest path between any two nodes, $U$ and $V$, has length $D_L(U, V)$. The dimension, $n$, the radix, $k$, and the number of nodes, $N$, have the following relations

$$N = k^n, \quad k = \sqrt[2]{N}, \quad n = \log_k N.$$  

If the nodes of a particular $k$-ary $n$-cube are named with the elements of $\{0, 1, \ldots, k-1\}^n$ (which we always assume they are) then we consider the links to be in one of $n$ different dimensions according to in which component the names of the link's two incident nodes differ (with the rightmost component corresponding to dimension 1). For each $i \in \{1, 2, \ldots, n\}$, we refer to all links whose incident nodes differ in the $i$th component as lying in dimension $i$. Also, for any $i \in \{1, 2, \ldots, n\}$, $Q_n^k$ consists of $k$ disjoint copies of $Q_{n-1}^k$ where corresponding nodes are joined in cycles of length $k$ using links in dimension $i$. When we consider $Q_n^k$ in this way, with the disjoint copies joined by links lying in dimension $i$, we say that we have partitioned $Q_n^k$ over dimension $i$.

Intuitively, we can construct a $k$-ary $n$-cube recursively as follows:

- make $k$ copies of a $k$-ary $(n-1)$-cube
• for each $i = 0, 1, \ldots, k - 1$, rename every node in the $i$th copy by concatenating an $i$ to the left of the node's name in that copy

• for each $j = 1, 2, \ldots, n$, join the nodes whose components differ in only the $j$th component in a cycle of length $k$ (as defined above).

Fig. 2.1 illustrates an example of constructing a 4-ary 3-cube $Q^4_3$ recursively starting from a 4-ary 1-cube.

2.3 Topological Properties

When designing a large multicomputer, one of the most important design decisions involves the topology of the communication structure among the processors. The degree (number of incident links) of each node, the total number of links, and the diameter of the network should be known before choosing the network. In order to hint at why the $k$-ary $n$-cube is popular as an interconnection network for parallel processing, we present in this section some of its topological properties [4, 12, 17].

The degree of each node in the $k$-ary $n$-cube is $2n$; the total number of links is $nk^n$; its diameter is $\lceil k/2 \rceil n$; and it contains $k^{n-1}$ node-disjoint cycles each of length $k$ in each dimension. For example, the four node-disjoint cycles in each dimension for the $Q^4_2$ are as follows:

<table>
<thead>
<tr>
<th>Dimension 1</th>
<th>Dimension 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C_1 = {(0,0), (0,1), (0,2), (0,3)}$</td>
<td>$C_1 = {(0,0), (1,0), (2,0), (3,0)}$</td>
</tr>
<tr>
<td>$C_2 = {(1,0), (1,1), (1,2), (1,3)}$</td>
<td>$C_2 = {(0,1), (1,1), (2,1), (3,1)}$</td>
</tr>
<tr>
<td>$C_3 = {(2,0), (2,1), (2,2), (2,3)}$</td>
<td>$C_3 = {(0,2), (1,2), (2,2), (3,2)}$</td>
</tr>
<tr>
<td>$C_4 = {(3,0), (3,1), (3,2), (3,3)}$</td>
<td>$C_4 = {(0,3), (1,3), (2,3), (3,3)}$</td>
</tr>
</tbody>
</table>
Figure 2.1: The recursive structure of $Q_3^4$: (a) $Q_1^4$; (b) $Q_2^4$; (c) $Q_3^4$. 
Suppose that we are given a set of generators of a finite group \( \Gamma \). If a network can be drawn such that the nodes correspond to the elements of the group \( \Gamma \) and there is a link from an element \( a \) to an element \( b \) if and only if there is a generator \( g \) such that \( ag = b \) in the group \( \Gamma \), then this network is called a Cayley graph. It is required that the set of generators be closed under inverses so that the resulting network can be viewed as being undirected graph. It is well-known, e.g. [1], that every Cayley graph is node-transitive (symmetric).

Bettayeb [12] showed that the \( k \)-ary \( n \)-cube \( Q_n^k \) network can be represented as a Cayley graph with the generating set consisting of all the elements \((a_n, a_{n-1}, \ldots, a_1)\) where \( a_j \in \{1, k-1\} \), for some \( j \), and \( a_i = 0 \) for all \( i \neq j, 1 \leq i, j \leq n \), and the operation is addition of components modulo \( k \). Hence, a \( k \)-ary \( n \)-cube network is node-transitive.

The following lemma shows that the \( k \)-ary \( n \)-cube network \( Q_n^k \) is link-transitive [4]. In other words, we show that for every pair of links \( e_1 \) and \( e_2 \), there is an automorphism of \( Q_n^k \) mapping \( e_1 \) to \( e_2 \).

**Lemma 2.3** Let \( k \geq 3 \) and \( n \geq 1 \). \( Q_n^k \) is link-transitive.

**Proof** We proceed by induction on \( n \). The base case, when \( n = 1 \), is trivial. Suppose that the result holds for \( Q_n^k \), where \( n \geq 1 \). Let

\[
e_1 = ((x_{n+1}, \ldots, x_i, \ldots, x_1), (x_{n+1}, \ldots, x_i + 1, \ldots, x_1)) \text{ and }
e_2 = ((y_{n+1}, \ldots, y_j, \ldots, y_1), (y_{n+1}, \ldots, y_j + 1, \ldots, y_1))
\]

be two links of \( Q_{n+1}^k \), where \( 1 \leq i, j \leq n + 1 \) and addition of components is modulo \( k \) (as it is throughout the proof). Without loss of generality, there are two cases to consider.

**Case (i)** \( i = j = 1 \). For each \( l = 1, 2, \ldots, n + 1 \), let \( \rho_l \) be the automorphism of \( Q_{n+1}^k \) defined via \( \rho_l : (z_{n+1}, \ldots, z_l, \ldots, z_1) \mapsto (z_{n+1}, \ldots, z_l + \ldots) \).
1, ..., 1), and set $\delta_i = y_i - x_i \pmod{k}$. Then the automorphism
$(\rho_{n+1})^{\delta_{n+1}} \cdots (\rho_i)^{\delta_2} (\rho_1)^{\delta_1}$ maps $e_1$ to $e_2$.

Case (ii) $i = 1$ and $j = 2$. Let $\rho_{1,2}$ be the automorphism of $Q_{n+1}^k$
defined via $\rho_{1,2}(z_n+1, ..., z_3, z_2, z_1) \mapsto (z_n+1, ..., z_3, z_1, z_2)$, and set
$\delta_{1,2} = y_2 - x_1 \pmod{k}$ and $\delta_{2,1} = y_1 - x_2 \pmod{k}$. Then the auto-
morphism $(\rho_{n+1})^{\delta_{n+1}} \cdots (\rho_i)^{\delta_2} (\rho_1)^{\delta_1}$ maps $e_1$ to $e_2$. □

The transfer of a large amount of data between two nodes in a multi-
computer may be facilitated by dividing the data into small packets and
sending the packets along different routes. In order to avoid contention,
the packets should travel by routes having no common nodes except the
sending and receiving nodes. Such paths between two nodes $A$ and $B$,
referred to as node-disjoint parallel paths, provide a means of selecting
alternate routes between $A$ and $B$ and increase fault-tolerance. The fol-
lowing theorem states the number and length of disjoint parallel paths
between any two nodes belonging to $Q_n^k$ [17].

**Theorem 2.4** Given $A = (a_n, a_{n-1}, ..., a_1)$ and $B = (b_n, b_{n-1}, ..., b_1)$.
Let $l = D_L(A, B), h = D_H(A, B)$, and $w_i = \text{dis}(a_i, b_i)$ for $1 \leq i \leq n$.
Then, in a $k$-ary $n$-cube, $k > 2$, there are a total of $2n$ node-disjoint parallel paths between $A$ and $B$ of which

(i) $h$ paths have length $l$,

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

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

**Proof** W.l.o.g., assume that the first $h$ digits of the labels of $A$ and $B$
are different while the remaining $n - h$ digits are the same. Then
(i) For each $i, 1 \leq i \leq h$, construct the $i$th path as follows. Starting with the label of $A$, correct digit $i$ using the shortest path in the cycle of dimension $i$ between $a_i$ and $b_i$. Repeat this procedure for the remaining digits of $A$, proceeding sequentially through dimensions $i+1, i+2, \ldots, h, 1, \ldots, i-1$. This produces $h$ paths of length $l$.

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

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

It can be seen that none of these paths share any nodes except $A$ and $B$. □
Note the theorem above is not valid when $k = 2$. This is because adding one to a bit is the same as subtracting one from a bit in the binary case. Example 2.5 shows the six disjoint parallel paths between two nodes labelled $(0,1,3)$ and $(0,3,4)$ in a $Q_5^3$.

**Example 2.5** The six disjoint parallel paths between $(0,1,3)$ and $(0,3,4)$ in a $Q_5^3$ are:

Path 1: $(0,1,3) \rightarrow (0,1,4) \rightarrow (0,2,4) \rightarrow (0,3,4)$

Path 2: $(0,1,3) \rightarrow (0,2,3) \rightarrow (0,3,3) \rightarrow (0,3,4)$

Path 3: $(0,1,3) \rightarrow (1,1,3) \rightarrow (1,1,4) \rightarrow (1,2,4) \rightarrow (1,3,4) \rightarrow (0,3,4)$

Path 4: $(0,1,3) \rightarrow (4,1,3) \rightarrow (4,1,4) \rightarrow (4,2,4) \rightarrow (4,3,4) \rightarrow (0,3,4)$

Path 5: $(0,1,3) \rightarrow (0,1,2) \rightarrow (0,2,2) \rightarrow (0,3,2) \rightarrow (0,3,1) \rightarrow (0,3,0) \rightarrow (0,3,4)$

Path 6: $(0,1,3) \rightarrow (0,0,3) \rightarrow (0,0,4) \rightarrow (0,4,4) \rightarrow (0,3,4)$

### 2.4 Embeddings in $k$-ary $n$-cubes

This section considers some examples from the literature [17] on embeddings of common network topologies into a $k$-ary $n$-cube. The topologies considered are a Hamiltonian cycle, a mesh, a binary tree, and a binary hypercube ($B_n$). All our embedding problems considered throughout this thesis have dilation, congestion and load equal to one.

**Embedding a Hamiltonian Cycle**

Let $A = \langle A_1, A_2, \ldots, A_N \rangle$ be a sequence of distinct node labels in a $Q_n^k$. If $A$ forms a cycle (or a ring) of length $N = k^n$, then $Q_n^k$ is *Hamiltonian*. Since the Lee distance between any two successive labels and the Lee
distance between $A_N$ and $A_1$ must be 1, the sequence of node labels, $A$, forms a Gray code.

The preceding information suggests that one means of generating a Hamiltonian cycle is to generate a Gray code. A Gray code for a $Q_n^k$ can be generated using the $k$-ary Gray code presented in [17] which is as follows.

**Theorem 2.6** Let $S = (0,1,\ldots,k^n - 1)$ be a sequence of $n$ digit, radix $k$ numbers. Let $f : \{0,1,\ldots,k - 1\}^n \rightarrow \{0,1,\ldots,k - 1\}^n$ be such that

$$f(a_n, a_{n-1}, \ldots, a_1) = a_n, a_{n-1} - a_n(\text{mod} k), \ldots, a_1 - a_2(\text{mod} k).$$

Then the sequence

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

forms a $k$-ary Gray code for a $Q_n^k$.

Below is an example of a Gray code in a $Q_4^2$ obtained by the above theorem. See [17] for more Gray code strategies.

<table>
<thead>
<tr>
<th>$S$</th>
<th>$S^*$</th>
<th>$S$</th>
<th>$S^*$</th>
<th>$S$</th>
<th>$S^*$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>10</td>
<td>13</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>11</td>
<td>10</td>
<td>21</td>
<td>23</td>
</tr>
<tr>
<td>02</td>
<td>02</td>
<td>12</td>
<td>11</td>
<td>22</td>
<td>20</td>
</tr>
<tr>
<td>03</td>
<td>03</td>
<td>13</td>
<td>12</td>
<td>23</td>
<td>21</td>
</tr>
</tbody>
</table>

**Embedding a Mesh**

The Gray code presented in Theorem 2.6 can be used to embed meshes or tori of certain dimensions into $Q_n^k$ [17]. Let $M$ be a $K^{n_1} \times K^{n_2} \times \ldots \times K^{n_m}$-dimensional mesh or torus, where $n = \sum_{i=1}^{m} n_i$. The following construction shows how to embed $M$ into $Q_n^k$. 43
Assume that each dimension $i$ of $M$ is labelled with a radix $k$, $n_i$ digit number $0, 1, \ldots, k^{n_i} - 1$. Using Theorem 2.6, relabel each dimension with the corresponding Gray code sequence. Now each node of $M$ can be identified with an $m$-tuple whose $i$th component is the node’s location in the $i$th dimension, $1 \leq i \leq m$. If $x$ is a node of $M$ with label $(x_1, x_2, \ldots, x_m)$, then define $f_V(x) = x_1, x_2, \ldots, x_m$, the concatenation of $x_1, x_2, \ldots, x_m$.

It should be clear that if $x$ and $y$ are any two adjacent nodes in $M$, then $f_V(x)$ and $f_V(y)$ are adjacent in $Q^k_n$. For if $x$ and $y$ are adjacent in $M$, their labels differ only in some dimension $i$. Since each dimension is labelled with a Gray code, the Lee distance between $x$ and $y$ in dimension $i$ is one. Therefore, $D_L(f_V(x), f_V(y)) = 1$, and $x$ and $y$ are adjacent in $Q^k_n$. As an example, Fig. 2.2 shows a $3^2 \times 3$ mesh embedded into a $Q^3_3$. In the mesh, $x(1, 2, 1)$ is adjacent to four other nodes: $A(1, 2, 0), B(0, 2, 1), C(1, 2, 2), \text{ and } D(1, 0, 1)$. It is easily verified that the Lee distance between $x$ and $A, B, C, \text{ or } D$ is 1. Therefore, $x$ is adjacent to the other four nodes in the $Q^3_3$.

![Figure 2.2: Embedding a $3^2 \times 3$ mesh into a $Q^3_3$.](image)
Embedding a Binary Tree

The following lemma shows how a complete binary tree, denoted $T_h$, of height $h$ (where the root is at height 0) and $N = 2^{h+1} - 1$ nodes can be easily embedded into a $Q^k_n$.

**Lemma 2.7** A $k$-ary $n$-cube $Q^k_n$ network, for $k \geq 3$ and $n \geq 2$, contains a complete binary tree $T_n$ of height $n$ as a subnetwork.

**Proof** We proceed by induction on $n$. When $n = 2$, then Fig. 2.3(a) shows a complete binary tree $T_2$ of height 2 embedded into a $Q^2_2$. Assume that a $Q^k_n$, for $k \geq 3$ and for some $n \geq 2$, contains a $T_n$ as a subnetwork. Now consider $Q^k_{n+1}$ partitioned over dimension $n + 1$ into $k$ disjoint isomorphic copies of $k$-ary $n$-cubes $Q^k_n(0), Q^k_n(1), \ldots, Q^k_n(k-1)$ (where the isomorphism is the natural one) with corresponding nodes linked in a cycle of length $k$.

By induction hypothesis, $Q^k_n(0)$ and $Q^k_n(2)$ contain isomorphic copies of $T_n$. By connecting the root of $T_n$ in $Q^k_n(0)$ and the root of $T_n$ in $Q^k_n(2)$ to their corresponding node $R$ in $Q^k_n(1)$, the resulting construction is a $T_{n+1}$ rooted at $R$ (i.e., see Fig. 2.3(b)). □

Embedding a Hypercube

It is easy to show by induction on $n$ that a binary $n$-cube $B_n$ network is a subnetwork of a $k$-ary $n$-cube $Q^k_n$. As an induction base case, note that any link in a $Q^k_1$ is a $B_1$. Suppose that the result holds for some $n \geq 1$. Partitioning $Q^k_{n+1}$ over dimension 1 yields $k$ copies of $Q^k_n$, namely $Q^k_n(0), Q^k_n(1), \ldots, Q^k_n(k-1)$. By the induction hypothesis, $Q^k_n(0)$ and $Q^k_n(1)$ contain isomorphic copies of $B_n$. The binary $(n+1)$-cube $B_{n+1}$ of
Figure 2.3: Embedding a complete binary tree $T_n$ into a $Q_n^k$:

(a) a $T_2$ embedded into a $Q_2^k$. (b) a $T_{n+1}$ embedded into a $Q_{n+1}^k$.

$Q_{n+1}^k$ can be obtained by taking the disjoint union of the copies of $B_n$ in $Q_n^k(0)$ and $Q_n^k(1)$ and joining corresponding nodes.

The following lemma shows that a binary $2d$-cube $B_{2d}$ is isomorphic to a 4-ary $d$-cube $Q_d^4$ [17].

**Lemma 2.8** Let $k = 4$ and $d \geq 1$. Then a binary $2d$-cube $B_{2d}$ is isomorphic to a 4-ary $d$-cube $Q_d^4$.

**Proof** We proceed by induction on $d$. Let $f : \{0,1\}^2 \rightarrow \{0,1,2,3\}$ where $f(0,0) = 0, f(0,1) = 1, f(1,1) = 2,$ and $f(1,0) = 3$. Then clearly $f$ maps $B_2$ to $Q_1^4$. Suppose that the result holds for some $d \geq 1$. Partitioning $Q_{d+1}^4$ over dimension 1 yields 4 copies of $Q_d^4$, namely $Q_d^4(0), Q_d^4(1), Q_d^4(2), \text{and} Q_d^4(3),$ with corresponding nodes linked in a cycle of length $k$. 
By the induction hypothesis, each of $Q_d^4(0), Q_d^4(1), Q_d^4(2),$ and $Q_d^4(3)$ is isomorphic to a copy of $B_{2d}$. By taking the disjoint union of the copies of $B_{2d}$ in $Q_d^4(0)$ and $Q_d^4(1)$ and joining corresponding nodes, we obtain a binary $(2d + 1)$-cube $B_{2d+1}$. Denote this hypercube by $B'$. Similarly, by taking the disjoint union of the copies of $B_{2d}$ in $Q_d^4(2)$ and $Q_d^4(3)$ and joining corresponding nodes, we obtain another binary $(2d + 1)$-cube $B_{2d+1}$. Denote this hypercube by $B''$. Now, by taking the disjoint union of the copies of $B'$ and $B''$ and joining corresponding nodes, we obtain a binary $(2d + 2)$-cube $B_{2(d+1)}$. □

A straightforward result from the above lemma is that a binary $n$-cube $B_n$ network is a subnetwork of a 4-ary $\lceil n/2 \rceil$-cube $Q_4^{\lceil n/2 \rceil}$. As an example, Fig. 2.4 shows a binary 4-cube $B_4$ embedded into a 4-ary 2-cube $Q_2^4$.

![Figure 2.4: A binary 4-cube $B_4$ embedded into a $Q_2^4$.](image)

47
Chapter 3

Embeddings of Cycles in
$k$-ary $n$-cubes

3.1 Introduction

We have shown in Chapter 2 that the $k$-ary $n$-cube can simulate a cycle of length $k^n$. In this chapter, we completely classify when a $k$-ary $n$-cube $Q^k_n$, for $k \geq 3$ and $n \geq 2$, contains a cycle of some given length. We start this chapter by giving in Section 3.2 a generation of $k$-ary Gray codes of dimension $n$. The $k$-ary Gray codes have been used to obtain Hamiltonian cycles in $k$-ary $n$-cubes (see, e.g., [17]). To a certain extent, we are repeating what was done in [17] where two methods of generating $k$-ary Gray codes of dimension $n$ were given, one of which is recursive as is ours. However, the recursive method in [17] yields a Hamiltonian cycle in a $k$-ary $n$-cube only when $k$ is even: our recursive method yields a Hamiltonian cycle for every $k \geq 2$. Recursive generating methods might prove useful when it comes to implementation.

In Section 3.3, we ascertain exactly when a cycle of length $m$, where
3 \leq m \leq k^n$, can be embedded in $Q_n^k$. Our analysis yields an algorithm for generating a cycle of length $m$ in $Q_n^k$, when one exists, thus answering a question posed in [17]. Broeg [18] had previously shown how to embed an even length cycle in a general toroidal network using Gray codes, a totally different approach to ours; and even then only when at least one of the generating cycles in the direct product has even length. He did not attempt to classify when cycles exist in toroidal networks (or even in $k$-ary $n$-cubes) and a general classification has yet to be obtained. As can be seen from our proofs, the embedding of even length cycles in $k$-ary $n$-cubes is straightforward when compared with the much more interesting case of embedding odd length cycles.

3.2 Recursive Structure of Gray Codes

As defined in Chapter 2, a $k$-ary Gray code of dimension $n$ is an ordering of the elements of $\{0, 1, \ldots, k-1\}^n$ such that the Lee distance between consecutive elements in the list is 1, as is the Lee distance between the first and last elements. It is not immediately apparent that such Gray codes exist (but they do, as we shall see).

Suppose that $G_k(i)$ is a $k$-ary Gray code of dimension $i$. Then $Q_k(i)$ is defined to be the list obtained from $G_k(i)$ by only including those elements of $G_k(i)$ whose rightmost digit is 0, and $S_k(i)$ is defined to be the list obtained from $G_k(i)$ by only including those elements of $G_k(i)$ whose rightmost digit is different from 0. $Q_k^r(i)$ (resp. $S_k^r(i)$) is the list obtained from $Q_k(i)$ (resp. $S_k(i)$) by reversing the order of the elements in the list. For any $j \in \{0, 1, \ldots, k-1\}$, $jQ_k(i)$ is the list obtained by prefixing every element of $Q_k(i)$ with the digit $j$, and the same goes for $jS_k(i)$, $jQ_k^r(i)$ and $jS_k^r(i)$. Note that the elements of $jQ_k(i)$, $jS_k(i)$,
$jQ_k(i)$ and $jS_k(i)$ are $(i + 1)$-tuples.

Define the $k$-ary Gray code of dimension $1$ $G_k(1)$ as $(0, 1, \ldots, k - 1)$; so,

$$Q_k(1) = (0) \text{ and } S_k(1) = (1, 2, \ldots, k - 1).$$

Suppose that $G_k(i)$ is a $k$-ary Gray code of dimension $i$ such that $G_k(i)$ is the concatenation, $Q_k(i); S_k(i)$, of $Q_k(i)$ and $S_k(i)$ (this is true for $G_k(1)$). Define $G_k(i + 1)$ as

$$0Q_k(i); 1Q_k(i); 2Q_k(i); \ldots; (k - 1)Q_k(i); (k - 1)S_k(i); (k - 2)S_k(i); \ldots$$

$$\ldots; 1S_k(i); 0S_k(i),$$

if $k$ is even, and

$$0Q_k(i); 1Q_k(i); 2Q_k(i); \ldots; (k - 1)Q_k(i); (k - 1)S_k(i); (k - 2)S_k(i); \ldots$$

$$\ldots; 1S_k(i); 0S_k(i),$$

if $k$ is odd. Then $G_k(i + 1)$ is a $k$-ary Gray code of dimension $i + 1$ such that $G_k(i + 1)$ is the concatenation of $Q_k(i + 1)$ and $S_k(i + 1)$. Clearly, $|G_k(n)| = k^n$. Two examples are given below. In Example 3.1, a Hamiltonian cycle for a $Q^4_2$ is given, and in Example 3.2, a Hamiltonian cycle for a $Q^3_3$ is given.

**Example 3.1** A recursive structure of Gray codes in a $Q^4_2$.

<table>
<thead>
<tr>
<th>$G_4(1)$</th>
<th>(0, 1, 2, 3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_4(1)$</td>
<td>(0)</td>
</tr>
<tr>
<td>$S_4(1)$</td>
<td>(1, 2, 3)</td>
</tr>
<tr>
<td>$G_4(2)$</td>
<td>(00, 10, 20, 30, 33, 32, 31, 21, 22, 23, 13, 12, 11, 01, 02, 03)</td>
</tr>
</tbody>
</table>
Example 3.2  A recursive structure of Gray codes in a $Q_3^n$.

<table>
<thead>
<tr>
<th>$G_3(1)$</th>
<th>(0, 1, 2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q_3(1)$</td>
<td>(0)</td>
</tr>
<tr>
<td>$S_3(1)$</td>
<td>(1, 2)</td>
</tr>
<tr>
<td>$G_3(2)$</td>
<td>(00, 10, 20, 21, 22, 12, 11, 01, 02)</td>
</tr>
<tr>
<td>$Q_3(2)$</td>
<td>(00, 10, 20)</td>
</tr>
<tr>
<td>$S_3(2)$</td>
<td>(21, 22, 12, 11, 01, 02)</td>
</tr>
<tr>
<td>$G_3(3)$</td>
<td>(000, 010, 020, 120, 110, 100, 200, 210, 220, 221, 222, 212, 211, 201, 202, 102, 101, 111, 112, 122, 121, 021, 022, 012, 011, 001, 002)</td>
</tr>
</tbody>
</table>

3.3  Embedding a Cycle of any Length

In this section, we ascertain exactly when a cycle of length $m$, where $3 \leq m \leq k^n$, can be embedded in $Q_n^k$. Our analysis yields an algorithm for generating a cycle of length $m$ in $Q_n^k$, when one exists, thus answering a question posed in [17]. We first consider the $k$-ary $n$-cube $Q_n^k$ when $k \geq 3$ is odd.

**Lemma 3.3**  Let $k \geq 3$ be odd. $Q_2^k$ contains a cycle of length $m$, for each $m$ such that $k \leq m \leq k^2$.

**Proof**  Suppose that $m$ is odd. Then $m = k + 2\alpha(k - 1) + 2\beta$, where $0 \leq 2\alpha \leq k - 1$ and $0 \leq \beta \leq k - 2$. There are two cases to consider.

**Case (i)**  $0 \leq 2\alpha \leq k - 3$. The cycle of length $m$ is as in Fig. 3.1 (where not all the links of $Q_2^k$ are shown: the nodes of each row and each column should be joined in a cycle of length $k$).
Case (ii) $2\alpha = k - 1$ and $0 \leq 2\beta \leq k - 1$. The cycle of length $m$ is as in Fig. 3.2.

Suppose that $m$ is even. Consider the tiled grid in Fig. 3.3. By taking the appropriate number of tiles and regarding the perimeter of these tiles as a cycle in $Q_2^k$, we can easily find a cycle of even length $m$, for every even $m$ such that $4 \leq m \leq k^2$. □
Theorem 3.4 Let $k \geq 3$ be odd and let $n \geq 2$. $Q^n_k$ contains a cycle of length $m$, for each $m$ such that $k \leq m \leq k^n$.

Proof We proceed by induction on $n$. When $n = 2$ the result follows by Lemma 3.3. Suppose that the result holds for $Q^n_k$, where $n \geq 2$. Consider $Q^n_{k+1}$ and let $m$ be such that $k \leq m \leq k^{n+1}$. Then $m$ can be written as $m = \alpha_1 k^n + \alpha_2$, where $0 \leq \alpha_1 \leq k$ and $0 \leq \alpha_2 \leq k^n - 1$.

Case (i) $\alpha_2 \geq k$ or $\alpha_2 = 0$. $Q^n_{k+1}$ is built from $k$ copies of $Q^n_k$ with corresponding nodes joined in cycles of length $k$. We can build a cycle $C$ of length $\alpha_1 k^n$ in the first $\alpha_1$ copies of $Q^n_k$ as follows. By the induction hypothesis, $Q^n_k$ is Hamiltonian. Take a Hamiltonian cycle $H_1$ in the first copy of $Q^n_k$, and let $H_i$ be the isomorphic copy of $H_1$ in the $i$th copy of $Q^n_k$, for $i = 2, 3, \ldots, \alpha_1$ (we refer here to the natural isomorphism). Let $w_1, x_1, y_1$ and $z_1$ be distinct nodes in $H_1$ such that $(w_1, x_1)$ and $(y_1, z_1)$ are links of $H_1$, and let $w_i, x_i, y_i$ and $z_i$ be their isomorphic copies in $H_i$, for $i = 2, 3, \ldots, \alpha_1$ (note that $Q^n_k$ has at least 9 nodes). If $\alpha_1 = 1$ then $C$ is $H_1$. If $\alpha_1 > 1$ is odd then $C$ is built from $H_1, H_2, \ldots, H_{\alpha_1}$.
by removing the links of \( \{(w_i, x_i) : i = 1, 2, \ldots, \alpha_1 - 1\} \cup \{(y_i, z_i) : i = 2, 3, \ldots, \alpha_1\} \), and including the links of \( \{(w_i, w_{i+1}), (x_i, x_{i+1}) : i = 1, 3, \ldots, \alpha_1 - 2\} \cup \{(y_i, y_{i+1}), (z_i, z_{i+1}) : i = 2, 4, \ldots, \alpha_1 - 1\} \). If \( \alpha_1 \) is even then \( C \) is built from \( H_1, H_2, \ldots, H_{\alpha_1} \) by removing the links of \( \{(w_i, y_i) : i = 1, 2, \ldots, \alpha_1 - 1\} \cup \{(w_i, z_i) : i = 2, 3, \ldots, \alpha_1 - 2\} \cup \{(y_{i+1}, z_{i+1}) : i = 1, 3, \ldots, \alpha_1 - 2\} \).

If \( \alpha_2 = 0 \) then we are done; so we may assume that \( \alpha_2 \geq k \). By the induction hypothesis, there is a cycle \( D \) of length \( \alpha_2 \) in the \((\alpha_1 + 1)\)th copy of \( Q_n^k \). As \( Q_n^k \) is link-transitive, by Lemma 2.3, we may assume that the link \((w_{\alpha_1+1}, x_{\alpha_1+1})\) is a link of \( D \), if \( \alpha_1 \) is odd, and \((y_{\alpha_1+1}, z_{\alpha_1+1})\) is a link of \( D \), if \( \alpha_1 \) is even (where \( w_{\alpha_1+1}, x_{\alpha_1+1}, y_{\alpha_1+1} \) and \( z_{\alpha_1+1} \) are the isomorphic copies of \( w_1, x_1, y_1 \) and \( z_1 \) in the \((\alpha_1 + 1)\)th copy of \( Q_n^k \)). The cycle of \( Q_{n+1}^k \) obtained from \( C \) and \( D \) by removing the links \((w_1, x_1)\) and \((w_{\alpha_1+1}, x_{\alpha_1+1})\), and including the links \((w_{\alpha_1}, w_{\alpha_1+1})\) and \((x_{\alpha_1}, x_{\alpha_1+1})\), if \( \alpha_1 \) is odd, and by removing the links \((y_{\alpha_1}, z_{\alpha_1})\) and \((y_{\alpha_1+1}, z_{\alpha_1+1})\), and including the links \((y_{\alpha_1}, y_{\alpha_1+1})\) and \((z_{\alpha_1}, z_{\alpha_1+1})\), if \( \alpha_1 \) is even, has length \( m \).

Case (ii) \( 0 < \alpha_2 < k \). Again, note that \( Q_{n+1}^k \) is built from \( k \) copies of \( Q_n^k \) with corresponding nodes joined in cycles of length \( k \). We must have that \( \alpha_1 > 0 \), and so rewrite \( m \) as \( m = \beta_1 k^n + \beta_2 + \beta_3 \), where \( \beta_1 = \alpha_1 - 1 \), \( \beta_2 = k^n - k \) and \( \beta_3 = k + \alpha_2 \). As \( n \geq 2 \) and \( k \geq 3 \), we have that \( 0 \leq \beta_1 \leq k - 2 \), \( k < \beta_2 < k^n \) and \( k < \beta_3 < k^n \). If \( \beta_1 > 0 \) then build a cycle \( C \) of length \( \beta_1 k^n \) in the first \( \beta_1 \) copies of \( Q_n^k \); a cycle \( D \) of length \( \beta_2 \) in the \((\beta_1 + 1)\)th copy of \( Q_n^k \); and a cycle \( E \) of length \( \beta_3 \) in the \((\beta_1 + 2)\)th copy of \( Q_n^k \), as in Case (i). We can now join \( C \), \( D \) and \( E \) as in Case (i) (using Lemma 2.3) to obtain a cycle of length \( m \) in \( Q_{n+1}^k \). 

\( \square \)
Proposition 3.5 Let \( k \geq 3 \) be odd and let \( n \geq 2 \). There are no cycles of odd length less than \( k \) in \( Q^k_n \).

Proof Suppose that \( C \) is a cycle in \( Q^k_n \) of odd length less than \( k \). By [12], \( Q^k_n \) is node-transitive, and so we may assume that the node \((0,0,\ldots,0)\) is in \( C \). Consider starting at \((0,0,\ldots,0)\) and moving along \( C \). Suppose we traverse a link of \( C \) taking us from a node of the form \((x_1,\ldots,x_{i-1},0,x_{i+1},\ldots,x_n)\) to the node \((x_1,\ldots,x_{i-1},k-1,x_{i+1},\ldots,x_n)\): then we say that this link is a flip of dimension \( i \). In the reverse situation, we say that the link is an inverse flip of dimension \( i \). Note that as \( C \) has length less than \( k \), for every flip (resp. inverse flip) of dimension \( i \), there must correspond an inverse flip (resp. flip) of dimension \( i \). That is, flips and inverse flips come in pairs. Define the parity of a node \((x_1,x_2,\ldots,x_n)\) to be 0 (resp. 1) if the sum \( x_1 + x_2 + \ldots + x_n \) is even (resp. odd). Note that the only links of \( C \) the traversal of which preserve the parity of nodes are flips and inverse flips. Hence, as flips and inverse flips come in pairs, we obtain a contradiction. \( \square \)

Corollary 3.6 Let \( k \geq 3 \) be odd and let \( n \geq 2 \). There is a cycle of length \( m \) in \( Q^k_n \), for all even \( m \) such that \( 4 < m < k - 1 \).

Proof \( Q^k_n \) contains \( Q^k_2 \) as a subnetwork, and so proceed as in the proof of Lemma 3.3. \( \square \)

Now we consider the existence of cycles in the \( k \)-ary \( n \)-cube \( Q^k_n \) when \( k \geq 4 \) is even. In such networks, every link of \( Q^k_n \) joins a node of even parity with a node of odd parity, and so \( Q^k_n \) is bipartite and can not have odd length cycles.

Lemma 3.7 Let \( k \geq 4 \) be even. \( Q^k_2 \) contains a cycle of length \( m \), for every even \( m \) such that \( 4 \leq m \leq k^2 \).
Proof Consider the tiled grid in Fig. 3.4. By taking the appropriate number of tiles and regarding the perimeter of these tiles as a cycle in $Q^k_2$, we can find a cycle of even length $m$, for every even $m$ such that $4 \leq m \leq k^2$.

$\square$

Figure 3.4: The tiled grid of Lemma 3.7.

**Theorem 3.8** Let $k \geq 4$ be even and let $n \geq 2$. $Q^k_n$ contains a cycle of length $m$, for every even $m$ such that $4 \leq m \leq k^n$.

**Proof** We proceed by induction on $n$. The base case of the induction, when $n = 2$, follows by Lemma 3.7. Suppose that the result holds for $Q^k_n$, where $n \geq 2$. Consider $Q^k_{n+1}$. We can write $m$ as $m = \alpha_1 k^n + \alpha_2$, where $0 \leq \alpha_1 \leq k$ and either $\alpha_2 = 0$ or $4 \leq \alpha_2 \leq k^n - 1$, or as $m = \alpha_1 k^n + (k^n - 2) + 4$, where $0 \leq \alpha_1 \leq k - 2$. Either way, by proceeding as in the proof of Theorem 3.4, the result follows. $\square$

Drawing together the above results yields the following.

**Corollary 3.9** Consider the $k$-ary $n$-cube $Q^k_n$, where $k \geq 3$ and $n \geq 2$. When $k$ is odd:
\begin{itemize}
\item \(Q_n^k\) contains no cycles of odd length less than \(k\) but contains a cycle of length \(m\), for every even \(m\) such that \(4 \leq m \leq k\).
\item \(Q_n^k\) contains a cycle of length \(m\), for every \(m\) such that \(k \leq m \leq k^n\).
\end{itemize}

When \(k\) is even:

\begin{itemize}
\item \(Q_n^k\) contains no odd length cycles but contains a cycle of length \(m\), for every even \(m\) such that \(4 \leq m \leq k^n\).
\end{itemize}

Theorem 3.4 yields an alternative proof of the result established in Section 3.2 that \(Q_n^k\) is Hamiltonian, for \(k \geq 3\) and \(n \geq 2\). Also, the proof of Theorem 3.4 yields an algorithm for generating a cycle of length \(m\) in \(Q_n^k\), if one exists, thus answering a question posed in [17].
Chapter 4

Fault-Tolerant Embeddings of Cycles, Meshes and Tori in $k$-ary $n$-cubes

4.1 Introduction

In massively parallel systems, as the size of the system increases, so does the probability of component failure. Fault-tolerant networks are essential to the reliability of parallel computer systems. A fault-tolerant network has the ability to simulate other network topologies even if certain network components (i.e., nodes and/or communication links) fail.

We have examined in Chapter 3 the capability of the non-faulty $k$-ary $n$-cube network of simulating cycle-structured networks. In this chapter, we investigate the existence of cycles, meshes and tori in a $k$-ary $n$-cube $Q_n^k$ with a limited number of node and link faults. The existence of cycles, meshes and tori in faulty hypercubes has been reasonably well studied (see, e.g., [22, 23, 59, 92, 96]) whereas for $k$-ary $n$-cubes the situation is
nowhere near as clear with most research on faulty \( k \)-ary \( n \)-cubes having focussed on routing and broadcasting (e.g., [14, 18, 62, 76]). As regards the existence of cycles, meshes and tori in faulty hypercubes, all of the literature mentioned above except [92] considers the presence of either only faulty links or only faulty nodes but not both. However, Tseng [92] showed that there exists a cycle of length at least \( 2^n - 2\nu \) in a hypercube (of dimension \( n \)) with \( \nu \leq n - 1 \) faulty nodes and \( \lambda \leq n - 4 \) faulty links where the total number of faulty nodes and faulty links, \( \nu + \lambda \), does not exceed \( n - 1 \). We take a similar stance to Tseng and show that in a \( k \)-ary \( n \)-cube \( Q_n^k \), where \( k \geq 3 \) and \( n \geq 2 \), with \( \nu \) faulty nodes and \( \lambda \) faulty links where \( \nu + \lambda \leq n \), there exists a cycle of length at least \( k^n - \nu \omega \), where \( \omega = 1 \) if \( k \) is odd and \( \omega = 2 \) if \( k \) is even (in fact, we also show that in some circumstances when \( k \) is even, there is still a cycle containing all the healthy nodes in such a faulty \( k \)-ary \( n \)-cube). We extend our main result to obtain embeddings of meshes and tori in such a faulty \( k \)-ary \( n \)-cube.

4.2 Fault-Tolerant Embeddings of Cycles

This section examines the existence of a cycle of length at least \( k^n - \nu \omega \) in a \( k \)-ary \( n \)-cube \( Q_n^k \) with \( \lambda \) faulty links and \( \nu \) faulty nodes where \( \lambda + \nu \leq n \) (throughout this chapter, \( \omega = 1 \) if \( k \) is odd and \( \omega = 2 \) if \( k \) is even). Note that if a node in \( Q_n^k \) is faulty then we regard all of its \( 2n \) incident links as faulty: the \( \lambda \) faulty links alluded to above are faulty links between healthy nodes.

We begin by proving a basic result, which we shall use later, and by proving some results involving faulty \( k \)-ary 2-cubes. We then develop a partitioning scheme to decompose a \( k \)-ary \( n \)-cube into a collection of
k-ary 2-cubes so that these k-ary 2-cubes can be linked together to form our large cycle.

4.2.1 Basic Results

The following result involves the k-ary n-cube $Q^k_n$ with only faulty links.

**Lemma 4.1** Let $k \geq 3$ and $n \geq 2$. If $Q^k_n$ contains at most $n$ faulty links then $Q^k_n$ is Hamiltonian.

**Proof** We proceed by induction on $n$. The induction scheme used to prove this lemma uses the following technique. Partition the $Q^k_n$ over some dimension and consider the $k$ disjoint copies of $Q^k_{n-1}$ with corresponding nodes joined in cycles of length $k$. These disjoint copies are denoted by $Q^k_{n-1}(0), Q^k_{n-1}(1), \ldots, Q^k_{n-1}(k-1)$. Throughout this proof, if $u$ is a node of $Q^k_{n-1}(i)$, say, then we often denote it by $u_i$ and we refer to its corresponding node in $Q^k_{n-1}(j)$ as $u_j$. Assume that $Q^k_{n-1}(i)$ contains a Hamiltonian cycle $C_i$ for all $0 \leq i \leq k-1$. If there exists a link $(x_0, y_0)$ in $C_0$ such that $(x_0, x_1)$ and $(y_0, y_1)$ are both healthy and $(x_1, y_1) \in C_1$, then $C_0$ can be joined to $C_1$ by removing the links $(x_0, y_0)$ and $(x_1, y_1)$, and including the links $(x_0, x_1)$ and $(y_0, y_1)$ to form a cycle $D_2$ of length $2k^{n-1}$. If there exists a link $(u_1, v_1)$ in $D_2 \setminus \{(x_0, x_1), (y_0, y_1)\}$ such that $(u_1, u_2)$ and $(v_1, v_2)$ are both healthy and $(u_2, v_2) \in C_2$, then $D_2$ can be joined to $C_2$ by removing the links $(u_1, v_1)$ and $(u_2, v_2)$, and including the links $(u_1, u_2)$ and $(v_1, v_2)$ to form a cycle $D_3$ of length $3k^{n-1}$. Continuing in this way eventually yields a cycle $D_k$ of length $kk^{n-1} = k^n$ which is a Hamiltonian cycle of $Q^k_n$.

Returning to our induction scheme. If $n = 2$ then $Q^k_2$ contains at most 2 faulty links. There exists some dimension, say dimension 1, that
contains at least 1 faulty link. Partitioning $Q^k_2$ over dimension 1 yields $k$ disjoint copies $Q^k_1(0), Q^k_1(1), \ldots, Q^k_1(k - 1)$ of $Q^k_1$ with corresponding nodes joined in cycles of length $k$, where the faulty links contained in $Q^k_1(0), Q^k_1(1), \ldots, Q^k_1(k - 1)$ total at most 1. We have the following cases.

Case (i) All faulty links are in dimension 1.

Note that in this case every cube, $Q^k_1(i)$, is a cycle of length $k$. Consider a faulty link $f_e$ falling between $Q^k_1(i)$ and $Q^k_1(i+1)$. W.l.o.g. we can assume that $i = 0$. If the second faulty link falls between $Q^k_1(0)$ and $Q^k_1(1)$ then join $Q^k_1(0)$ to $Q^k_1(k - 1)$ by removing $(x_0, y_0)$ from $Q^k_1(0)$ and $(x_{k-1}, y_{k-1})$ from $Q^k_1(k - 1)$, and including the links $(x_0, x_{k-1})$ and $(y_0, y_{k-1})$. Denote this cycle by $D_2$. Then $D_2$ is of length $2k$ and contains every node of $Q^k_1(0)$ and $Q^k_1(k - 1)$ exactly once, and no other nodes. As there are at least 2 links of $Q^k_1(k - 1)$ in $D_2$, by a similar fashion, $D_2$ can be joined to $Q^k_1(k - 2)$ to form a cycle $D_3$ of length $3k$ and so on. continuing in this way eventually yields a Hamiltonian cycle in $Q^k_2$.

If $f_e$ is the only faulty link falling between $Q^k_1(0)$ and $Q^k_1(1)$ then as $k > 2$ there exists a link $(x_0, y_0)$ in $Q^k_1(0)$ such that the links $(x_0, x_1)$ and $(y_0, y_1)$ are both healthy: join $Q^k_1(0)$ and $Q^k_1(1)$ by removing $(x_0, y_0)$ from $Q^k_1(0)$ and $(x_1, y_1)$ from $Q^k_1(1)$, and including the links $(x_0, x_1)$ and $(y_0, y_1)$ to form a cycle $D_2$ of length $2k$. As $2\lfloor k/2 \rfloor > 1$, the second faulty link in dimension 1, there exists $(u_1, v_1)$ in $Q^k_1(1)$ such that the links $(u_1, u_2)$ and $(v_1, v_2)$ are both healthy, or $(u_0, v_0)$ in $Q^k_1(0)$ such that the links $(u_0, u_{k-1})$ and $(v_0, v_{k-1})$ are both healthy. Then $D_2$ can be joined to either $Q^k_1(2)$ or $Q^k_1(k - 1)$ to form a cycle of length $3k$. By proceeding in this way, we eventually obtain a Hamiltonian cycle of $Q^k_2$.

Case (ii) Dimension 1 contains exactly one faulty link.
W.l.o.g. we may assume that the only faulty link not falling in dimension 1 is \((x_0, y_0)\) in \(Q_1^k(0)\). Either the links \((x_0, x_1)\) and \((y_0, y_1)\) are both healthy or the links \((x_0, x_{k-1})\) and \((y_0, y_{k-1})\) are both healthy. We can join \(Q_1^k(0)\) to \(Q_1^k(1)\) or to \(Q_1^k(k - 1)\), respectively, as in Case (i), and extend this cycle to a Hamiltonian cycle of \(Q_2^k\).

As our induction hypothesis, assume that the lemma holds for \(Q_n^k\), for some \(n \geq 2\) and for all \(k \geq 3\), and consider \(Q_{n+1}^k\) with at most \(n + 1\) faulty links. There is a dimension, say dimension 1, that contains at least 1 faulty link. Partition \(Q_{n+1}^k\) over dimension 1 and consider the \(k\) copies of \(Q^k\) having faulty links total at most \(n\).

Let a new \(Q_n^k\) contain all the \(n\) faulty links of the \(k\) copies. Then by the induction hypothesis, the new \(Q_n^k\) is Hamiltonian. Copy the Hamiltonian cycle of the new \(Q_n^k\) to all the \(k\) copies. This yields \(k\) isomorphic cycles \(C_0, C_1, \ldots, C_{k-1}\) each of length \(k^n\) (where the isomorphism is the natural one). As \(\lfloor k^n/2 \rfloor > n+1\), the total number of faulty links, for all \(k \geq 3\) and \(n \geq 2\), there exists some link \((x_0, y_0)\) in \(C_0\) such that the links \((x_0, x_1)\) and \((y_0, y_1)\) are both healthy. Denote by \(D_2\) the cycle obtained by removing \((x_0, y_0)\) from \(C_0\) and \((x_1, y_1)\) from \(C_1\), and including the links \((x_0, x_1)\) and \((y_0, y_1)\). Then \(D_2\) contains every node of \(C_0\) and \(C_1\) exactly once, and no other nodes.

All links of \(D_2\) except for \((x_0, x_1)\) and \((y_0, y_1)\) are links in \(C_0\) or \(C_1\). Hence, there is potential to join \(D_2\), as above, to \(C_2\). Again, as the maximum number of faulty links in dimension 1 is strictly less than \(\lfloor k^n/2 \rfloor\), there exists some link \((u_1, v_1)\) in \(D_2 \setminus \{(x_0, x_1), (y_0, y_1)\}\) such that \((u_1, u_2)\) and \((v_1, v_2)\) are both healthy. By proceeding as above, we can obtain a cycle \(D_3\) containing every node of \(C_0, C_1\) and \(C_2\) exactly once and no other nodes. Continuing in this way eventually yields a
Hamiltonian cycle of $Q_{n+1}^k$.

The following results involve the $k$-ary 2-cube $Q_2^k$.

**Lemma 4.2** Let $k \geq 3$. If $Q_2^k$ has $\nu \leq 2$ faulty nodes then $Q_2^k$ contains a cycle of length at least $k^2 - \nu \omega$.

**Proof** Suppose that $Q_2^k$ has 2 faulty nodes. Partition $Q_2^k$ over some dimension in which the labels of the 2 faulty nodes differ. This results in $k$ copies, $C_0, C_1, \ldots, C_k$, of cycles of length $k$ where the nodes of $C_i$ are $\{(i, j) : j = 0, 1, \ldots, k - 1\}$ and where corresponding nodes in these cycles are joined in cycles of length $k$. W.l.o.g. we may assume that node $(0, 0) \in C_0$ is faulty and that the other faulty node $v$ is not in $C_0$ or $C_{k-1}$.

**Case (i) $k$ is odd.**

The 2 different possibilities for $k = 3$ (up to isomorphism) and the cycles of length 7 are as shown in Fig. 4.1 (the nodes of the cycle $C_j$ are in the $j$th column with $(j, 0)$ at the bottom and $(j, 2)$ at the top).

![Faulty nodes](image)

*Figure 4.1: The cycles when $k = 3$.*

Suppose that $k \geq 5$ (and that $k$ is odd). Let $D_{j_0}^0, D_{j_1}^1, D_{j_2}^2$ and $D_{j_3}^3$ be the cycles of $Q_2^k$ depicted in Fig. 4.2, for some $j \in \{0, 1, \ldots, k - 1\}$,
involving all the nodes of \( C_j \) and \( C_{j+1} \) except for the nodes shown (here, addition is modulo \( k \)). Note that no matter which "row" a faulty node lies in, the cycles can be shifted vertically so as to avoid the faulty node.

Let the fault \( v \) be in \( C_a \). As \( k \geq 5 \), we may assume that \( a \neq k - 2 \).

Form the cycles:

- \( D_{k-1}^{0} \)
- \( D_{a}^{1} \), if \( a \) is odd
- \( D_{a-1}^{0} \), if \( a \) is even

Figure 4.2: The cycles \( D_{j}^{0}, D_{j}^{1}, D_{j}^{2} \) and \( D_{j}^{3} \).
• \(D_j^2\), for all odd \(j\) such that \(0 < j < k - 2\) and \(j \neq a\)

• \(C_{k-2}\).

By "joining" these cycles using links joining consecutive \(C_j\)'s (e.g., by replacing the links \(((k - 1, 0), (k - 1, 1))\) and \(((k - 2, 0), (k - 2, 1))\) with the links \(((k - 2, 0), (k - 1, 0))\) and \(((k - 2, 1), (k - 1, 1))\), we can form a cycle of length \(k^2 - 2\).

Case (ii) \(k\) is even.

Let \(P(i, j, m)\) denote the path

\[(i, j), (i, j + 1), (i + 1, j + 1), (i + 1, j + 2), (i, j + 2), (i, j + 3),
   (i + 1, j + 3), (i + 1, j + 4), (i, j + 4), \ldots, (i + 1, m - 1), (i + 1, m)\]

(if \(j = m\) then \(P(i, j, m)\) is the empty path) and let \(Q(i, j)\) denote the path

\[(i, j), (i, j + 1), (i, j + 2), \ldots, (i, j - 2), (i, j - 1), (i + 1, j - 1),
   (i + 1, j - 2), \ldots, (i + 1, j + 2), (i + 1, j + 1), (i + 1, j)\]

(-addition is modulo \(k\)). Let \(\hat{P}(i, j, m)\) denote the reversal of \(P(i, j, m)\).

Suppose that the fault \(v = (a, b)\).

Case (ii)(a) Either \(a\) is odd and \(b\) is even, or \(a\) is even and \(b\) is odd.

W.l.o.g., we may assume that \(a\) is odd and \(b\) is even (as the second case is isomorphic). If \(b \neq 0\) then the concatenation of the following paths forms a fault-avoiding cycle of length \(k^2 - 2\) in \(Q_2^2\):

\[
P(k - 1, 0, k - 2), (k - 1, k - 2), (k - 1, k - 1), (0, k - 1),
   Q(1, k - 1), Q(3, k - 1), \ldots, Q(a - 2, k - 1), (a, k - 1),
   \hat{P}(a, b + 1, k - 1), (a + 1, b + 1), (a + 1, b), \hat{P}(a, 0, b), (a + 1, 0),
   Q(a + 2, 0), Q(a + 4, 0), \ldots, Q(k - 3, 0), (k - 1, 0)
\]
and if \( b = 0 \) then the concatenation of the following paths forms a fault-avoiding cycle of length \( k^2 - 2 \) in \( Q_2^k \):

\[
P(k - 1, 0, k - 2), (k - 1, k - 2), (k - 1, k - 1), (0, k - 1),
\]
\[
Q(1, k - 1), Q(3, k - 1), \ldots, Q(a - 2, k - 1), (a, k - 1),
\]
\[
\hat{P}(a, b + 1, k - 1), (a + 1, b + 1), (a + 1, b), Q(a + 2, 0),
\]
\[
Q(a + 4, 0), \ldots, Q(k - 3, 0), (k - 1, 0)
\]

(remember, \( v \) is not in \( C_{k-1} \) or \( C_0 \)).

Case (ii)(b) Either \( a \) is odd and \( b \) is odd, \( a \) is even and \( b \) is even.

W.l.o.g., we may assume that \( a \) is odd and \( b \) is odd (as the second case is isomorphic). Consider the following cycles in \( Q_2^k \):

- \( D_{k-1}^3 \)
- \( D_a^3 \)
- \( D_j^2 \), for all odd \( j \) such that \( 1 \leq j \leq k - 3, j \neq a \).

By "joining" them as we did in Case (i), we obtain a fault-avoiding cycle of length \( k^2 - 4 \) in \( Q_2^k \).

The cases when \( Q_2^k \) has 1 fault are similar. \( \square \)

Note that for any even \( k \), it may be the case (but not necessarily always is) that \( Q_2^k \) has 2 faulty nodes and the longest fault-avoiding cycle has length \( k^2 - 4 \). This is because when \( k \) is even, \( Q_2^k \) is bipartite and if the two faulty nodes happen to lie on the same side of the partition then any cycle must necessarily omit at least 2 nodes from the other side of the partition.

Lemma 4.3 Let \( k \geq 3 \). If \( Q_2^k \) has exactly 1 faulty node and exactly 1 faulty link then there exists a cycle of length at least \( k^2 - \omega \).
Proof Adopting the notation of Lemma 4.2, we may assume that the faulty node is (0,0). Partition $Q_k^2$ over the dimension in which the faulty link, $(x,y)$, say, lies. W.l.o.g. we may assume that $x \in C_i$ and $y \in C_{i+1}$, where $i \neq 0$.

If $k$ is odd then we can take the cycles $D_0^1, D_2^2, D_4^2, \ldots, D_{k-3}^2$ and $C_{k-1}$, ensuring that the faulty link $(x,y)$ is not used in any $D_j^2$, and join them as in Lemma 4.2 (again ensuring that we do not use $(x,y)$ as a joining link) to obtain a cycle of length $k^2 - 1$.

If $k$ is even then we can take the cycles $D_0^3, D_2^2, D_4^2, \ldots, D_{k-4}^2$ and $D_{k-2}^2$, ensuring that the faulty link $(x,y)$ is not used in any $D_j^2$, and join them as in Lemma 4.2 (again ensuring that we do not use $(x,y)$ as a joining link) to obtain a cycle of length $k^2 - 2$. □

Lemma 4.4 Let $k \geq 3$. If $Q_k^2$ has exactly 1 faulty node, no faulty links and $(x,y)$ is a (healthy) link of $Q_k^2$ then there exists a cycle of length at least $k^2 - \omega$ which includes the link $(x,y)$.

Proof W.l.o.g. we may assume that the faulty node is (0,0) and that $(x,y)$ lies in dimension 1. There are 4 cases to consider: when $(x,y)$ joins $C_0$ and $C_1$, and when $k$ is odd and even; and when $(x,y)$ joins $C_i$ and $C_{i+1}$, where $i \in \{1,2,\ldots,k-2\}$, and when $k$ is odd and even.

Suppose that $(x,y)$ joins $C_0$ and $C_1$ and that $k$ is even. Then form a cycle of length $k^2 - 2$ by “joining” $D_{k-1}^3, D_1^2, D_3^2, \ldots, D_{k-5}^2, D_{k-3}^2$, in the sense of Lemma 4.2, ensuring that the link $(x,y)$ is used in the joining process.

Suppose that $(x,y)$ joins $C_0$ and $C_1$ and that $k$ is odd. Then form a cycle of length $k^2 - 1$ by “joining” $D_{k-1}^2, D_1^2, D_3^2, \ldots, D_{k-4}^2, C_{k-2}$, ensuring that the link $(x,y)$ is used in the joining process.

The remaining cases proceed similarly. □
4.2.2 Partitioning the Faulty $k$-ary $n$-cube

Having established some preliminary lemmas in the previous section, we
now use these lemmas to construct a long cycle in a faulty $k$-ary $n$-cube
in which the total number of (node and link) faults is at most $n$.

Let $d_1 \in \{1, 2, \ldots, n\}$ be some dimension of (the healthy) $Q_n^k$. Partitioning $Q_n^k$ over dimension $d_1$ yields $k Q_{n-1}^k$'s, namely $Q_{n-1}^k(0), Q_{n-1}^k(1), \ldots, Q_{n-1}^k(k-1)$, where the nodes of $Q_{n-1}^k(i)$ are named:

$$\{u \in \{0, 1, \ldots, k-1\}^n : \text{the } d_1\text{th component of } u \text{ is } i\}.$$  

Partitioning each $Q_{n-1}^k(i)$ over some dimension $d_2 \in \{1, 2, \ldots, n\} \setminus \{d_1\}$ yields $k Q_{n-2}^k$'s, namely $Q_{n-2}^k(i, 0), Q_{n-2}^k(i, 1), \ldots, Q_{n-2}^k(i, k-1)$, where the nodes of $Q_{n-2}^k(i, j)$ are named:

$$\{u \in \{0, 1, \ldots, k-1\}^n : \text{the } d_1\text{th component of } u \text{ is } i \text{ and the } d_2\text{th component of } u \text{ is } j\};$$

and so on. Proceeding in this fashion for $n - 2$ phases yields $k^{n-2}$ copies of $Q_2^k$.

A simple induction, allied with this proposed decomposition of $Q_n^k$, yields the following structural result.

**Lemma 4.5** Let $Q_n^k$ be healthy, where $k \geq 3$ and $n \geq 2$, and let $m$ be such that $1 \leq m \leq n$. Then $Q_n^k$ can be constructed from $Q_m^k$ as follows.

(i) Replace every node of $Q_m^k$ by a copy of $Q_{n-m}^k$ (all copies are disjoint).

(ii) If $(x, y)$ is a link of $Q_m^k$ then include a link from every node of the copy of $Q_{n-m}^k$ corresponding to $x$ to its corresponding node in the copy of $Q_{n-m}^k$ corresponding to $y$.  \[\Box\]
Now, suppose that our initial $Q_n^k$ is faulty where the number of faulty links $\lambda$ and the number of faulty nodes $\nu$ are such that $\lambda + \nu \leq n$. Suppose further that we apply the partitioning algorithm above so that at every stage $m$, $0 \leq m \leq n - 3$, the dimension over which we partition is chosen according to the following rules:

if there is a faulty link in some $Q_{n-m}^k$ lying in some as yet unused dimension $d$ then
partition every $Q_{n-m}^k$ over dimension $d$
else
if there are 2 faulty nodes in some $Q_{n-m}^k$ whose names differ in the as yet unused dimension $d$ then
partition every $Q_{n-m}^k$ over dimension $d$
else
partition every $Q_{n-m}^k$ over any as yet unused dimension $d$.

Apply the above partitioning algorithm $n - 2$ times. Let the (possibly) faulty $Q_{n-2}^k$, denoted $\pi(Q_n^k)$, be obtained from $Q_n^k$ as follows. Using Lemma 4.5, replace every $Q_2^k$ in our faulty $Q_n^k$ by a node and include a link $(x, y)$ iff every link joining the copy of $Q_2^k$ corresponding to $x$ and the copy of $Q_2^k$ corresponding to $y$ is healthy.

Now for our main result.

**Theorem 4.6** Let $k \geq 3$ and $n \geq 2$, and let $Q_n^k$ contain $\lambda$ faulty links and $\nu$ faulty nodes where $\lambda + \nu \leq n$. Then there exists a cycle of length at least $k^n - \nu \omega$.

**Proof** If $k \geq 3$ and $n = 2$ then the result follows by Lemmas 4.1, 4.2 and 4.3. Moreover, if $\lambda = n$ then the result follows by Lemma 4.1. Hence, we may assume that $\lambda \leq n - 1, k \geq 3$ and $n \geq 3$. 

69
Case (i) $\lambda \leq n - 2$.

Apply the partitioning algorithm above to yield $k^{n-2}$ (possibly) faulty $Q_2^k$’s and the (possibly) faulty $k$-ary $(n - 2)$-cube $\pi(Q_n^k)$. There are essentially two cases: (a) one $Q_2^k$ has two faulty nodes, $\nu - 2$ $Q_2^k$’s have one faulty node and no $Q_2^k$ has a faulty link; and (b) $\nu$ $Q_2^k$’s have one faulty node and no $Q_2^k$ has a faulty link.

By Lemma 4.1, as $\lambda \leq n - 2$, $\pi(Q_n^k)$ has a Hamiltonian cycle $H$. Also, by Lemma 4.2, any $Q_2^k$ with 1 faulty node has a cycle of length $k^2 - \omega$ and any $Q_2^k$ with 2 faulty nodes has a cycle of length $k^2 - 2\omega$; and, by Theorem 2.6, every healthy $Q_2^k$ has a Hamiltonian cycle. Using the Hamiltonian cycle in $\pi(Q_n^k)$, we can “join” the cycles in each $Q_2^k$ together, in the sense of the proof of Lemma 4.2, as follows.

In case (a), by Lemma 4.2, the copy of $Q_2^k$ containing 2 faulty nodes has a cycle $C$ of length $k^2 - 2\omega$. Consider the node $x$ of $\pi(Q_n^k)$ corresponding to this copy of $Q_2^k$ and let $y$ be the next node in the Hamiltonian cycle $H$ of $\pi(Q_n^k)$. Choose a link $(u, v)$ in the cycle $C$, in the copy of $Q_2^k$ corresponding to $x$, such that the nodes corresponding to $u$ and $v$ in the copy of $Q_2^k$ corresponding to $y$ are both healthy; call these nodes $u$ and $v$ also. By Lemma 4.4, the copy of $Q_2^k$ corresponding to $y$ has a cycle $D$ of length $k^2 - \omega$ which includes the link $(u, v)$, and we can “join” $C$ and $D$ over the links $(u, v)$. We can proceed in this way, continually using Theorem 2.6, Lemma 4.4 and the fact that any healthy $Q_n^k$ is link-transitive, so as to obtain a cycle of length $k^n - \nu\omega$. In case (b), similar reasoning yields the result.

Case (ii) $\lambda = n - 1$.

Apply the partitioning algorithm above to yield $k^{n-2}$ (possibly) faulty copies of $Q_2^k$ and the (possibly) faulty $k$-ary $(n - 2)$-cube $\pi(Q_n^k)$. There
are essentially two cases: (a) one $Q_k^2$ has one faulty node, one $Q_k^5$ has one faulty link and all other $Q_k^i$'s are healthy; and (b) one $Q_k^5$ has one faulty node and one faulty link, and all other $Q_k^i$'s are healthy. By proceeding as we did in case (i), using the above lemmas and results as appropriate, the result follows. □

Of course, there are some circumstances when there is in fact a longer cycle in the faulty $Q_n^k$ than is given by Theorem 4.6.

4.3 Embeddings of Meshes and Tori

In this section, we extend the main theorem of the previous section to embed a mesh or a torus in a faulty $k$-ary $n$-cube $Q_n^k$.

**Proposition 4.7** Let $k \geq 3$ and $n \geq 2$, and suppose that the $k$-ary $n$-cube $Q_n^k$ contains $\lambda$ faulty links and $\nu$ faulty nodes where $1 \leq f = \lambda + \nu \leq n - 1$.

(i) If $f \geq 2$ then there exists a mesh and a torus of size $(k^f - \nu \omega) \times k^{(n-f)}$.

(ii) If $f = 1$ and this fault is a faulty node then there exists a mesh and a torus of size $(k^i - \omega) \times k^{(n-i)}$, for each $i = 2, 3, \ldots, n - 1$.

(iii) If $f = 1$ and this fault is a faulty link then there exists a mesh and a torus of size $k^i \times k^{(n-i)}$, for each $i = 2, 3, \ldots, n - 1$.

**Proof** Suppose that $f \geq 2$. Partition $Q_n^k$ over a set $D$ of $n - f$ different dimensions so that each dimension of $D$ does not contain a faulty link. This results in $k^{(n-f)}$ disjoint copies of $Q_n^k$ such that the total number of faults in all copies is $f$. Build a new copy, $P$, of $Q_n^k$ by superimposing all
faults in the copies of $Q_k^f$ in $P$. Consequently, $P$ is a $k$-ary $f$-cube with $\nu$ faulty nodes and $\lambda$ faulty links where $\nu + \lambda = f$. By Theorem 4.6, $P$ contains a cycle $C$ of length at least $k^f - \nu \omega$. Also, every disjoint copy of $Q_k^f$, as above, contains the cycle $C$ (that is, all nodes and links in the isomorphic copy of $C$ in each copy of $Q_k^f$ are healthy).

The process of obtaining the $k^{(n-f)}$ disjoint copies of $Q_k^f$, above, results in a copy $\pi(Q_n^k)$ of $Q_{(n-f)}^k$, as in Lemma 4.5 (with the notation as in the paragraph preceding Theorem 4.6), that has no faulty nodes or links. By Theorem 2.6, $\pi(Q_n^k)$ has a Hamiltonian cycle. The links of $Q_n^k$ corresponding to the links of this Hamiltonian cycle in $\pi(Q_n^k)$, together with the cycles $C$ in each of the $Q_k^f$'s, result in a torus of size $(k^f - \nu \omega) \times k^{(n-f)}$.

If $Q_n^k$ has exactly 1 faulty node then by proceeding as above (using Theorem 4.6), $Q_n^k$ contains a torus of size $(k^i - \nu \omega) \times k^{(n-i)}$, for each $i = 2, 3, \ldots, n - 1$. If $Q_n^k$ has exactly 1 faulty link then by proceeding as above (using Lemma 4.1), $Q_n^k$ contains a torus of size $k^i \times k^{(n-i)}$, for each $i = 2, 3, \ldots, n - 1$. □

Bose et al. [17] and Bettayeb [12] have shown the existence of a mesh of size:

$$k^{n_1} \times k^{n_2} \times \ldots \times k^{n_s},$$

where $n = \sum_{i=1}^{s} n_i$, in a healthy $k$-ary $n$-cube $Q_n^k$. By proceeding as in the proof of Proposition 4.7, we can extend this result as follows.

**Proposition 4.8** Let $k \geq 3$ and $n \geq 2$, and suppose that the $k$-ary $n$-cube $Q_n^k$ contains $\lambda$ faulty links and $\nu$ faulty nodes where $1 \leq f = \lambda + \nu \leq n - 1$.

(i) If $f \geq 2$ then there exists a mesh and a torus of size $(k^f - \nu \omega) \times k^{n_1} \times \ldots \times k^{n_s}$, where $n - f = \sum_{i=1}^{s} n_i$.  

72
(ii) If \( f = 1 \) and this fault is a faulty node then there exists a mesh and a torus of size \((k^{n_1} - \omega) \times k^{n_2} \times \ldots \times k^{n_s}\), where \( n = \sum_{i=1}^{s} n_i \) and \( n_1 \neq 1 \).

(iii) If \( f = 1 \) and this fault is a faulty link then there exists a mesh and a torus of size \( k^{n_1} \times k^{n_2} \times \ldots \times k^{n_s}\), where \( n = \sum_{i=1}^{s} n_i \) and \( n_1 \neq 1 \).
Chapter 5

Embeddings of Hamiltonian Cycles in Faulty $k$-ary $n$-cubes

5.1 Introduction

We have developed in Chapter 4 a technique for embedding a cycle, a mesh and a torus in a $k$-ary $n$-cube with both faulty nodes and faulty links. The main objective of this chapter is to examine the number of link faults that a $k$-ary $n$-cube $Q_n^k$ can tolerate so that there is still a Hamiltonian cycle (of course, we assume that every node is incident with at least 2 healthy links). In particular, we show that a $k$-ary $n$-cube $Q_n^k$ where at most $4n - 5$ links are faulty and where every node is incident with at least two healthy links has a Hamiltonian cycle, but that there exist $k$-ary $n$-cubes with $4n - 4$ faults (and where every node is incident with at least two healthy links) not containing a Hamiltonian cycle. We also show that the general problem of deciding whether a faulty $k$-ary $n$-cube contains a Hamiltonian cycle is NP-complete, for all (fixed) $k \geq 3$.

Our results can be regarded as direct analogies of those in [22] for $k$-
ary $n$-cubes as opposed to binary $n$-cubes, although the proofs are more complicated, given the two parameters $k$ and $n$ as opposed to just the one, $n$. However, it should be pointed out that our approach is more general (with slight modifications, our arguments work for the binary $n$-cube also as we will show in Section 5.3) and simpler in that the base cases in Chan and Lee's paper involve binary $n$-cubes, where $n = 3, 4, 5$, yet our base cases involve $k$-ary $n$-cubes only where $n = 2$ and $k \geq 4$, and $n = 3$ and $k = 3$. When our approach is applied to binary $n$-cubes, the base case is only where $n = 3$.

5.2 Tolerating Faults in $k$-ary $n$-cubes

This section shows the existence of a Hamiltonian cycle in a $k$-ary $n$-cube $Q^k_n$, where $k \geq 3$ and $n \geq 2$, with at most $4n - 5$ faulty links where each node is incident with at least 2 healthy links. The proof of our main theorem of this section is by induction and is structured as follows. We begin by proving the inductive step and then we return to the base cases of the induction. In order to avoid repetition, we refer to reasoning used in the proof of the inductive step whilst proving our base cases (note that this introduces no circularity to our arguments).

**Theorem 5.1** Let $k \geq 4$ and $n \geq 2$, or let $k = 3$ and $n \geq 3$. If $Q^k_n$ has at most $4n - 5$ faulty links and is such that every node is incident with at least 2 healthy links then $Q^k_n$ has a Hamiltonian cycle.

**Proof** The proof proceeds by induction on $n$. We handle the base cases, when $n = 2$ and $k \geq 4$, and when $n = 3$ and $k = 3$ later. As our induction hypothesis, assume that the result holds for $Q^k_n$, for some $n \geq 2$ and for all $k \geq 4$, or for some $n \geq 3$ and $k = 3$. Let $Q^k_{n+1}$ have $4n - 1$ faults and be
such that every node is incident with at least 2 healthy links. Then there
exists some dimension, say dimension 1, which contains at least 3 faults.
We can partition \( Q^k_{n+1} \) over dimension 1 and consider \( Q^k_{n+1} \) to consist of
\( k \) disjoint copies \( Q_1, Q_2, \ldots, Q_k \) of \( Q^k_n \) with corresponding nodes joined
in cycles of length \( k \), where the faults contained in \( Q_1, Q_2, \ldots, Q_k \) total
at most \( 4n - 4 \) (see Fig. 5.1). Throughout this proof, if \( u \) is a node of
\( Q_i \), say, then we often denote it by \( u_i \), and we refer to its corresponding
node in \( Q_j \) as \( u_j \).

![Figure 5.1: The \( k \) copies of \( Q^k_n \).](image)

Case (i) Each \( Q_i \) is such that every node is incident with at least 2
healthy links and no \( Q_i \) contains \( 4n - 4 \) faults.

W.l.o.g. we may assume that \( Q_1 \) has most faults from amongst \( Q_1, Q_2, \ldots, Q_k \). Hence, each of \( Q_2, Q_3, \ldots, Q_k \) has at most \( 2n - 2 \) faults. By the
induction hypothesis, \( Q_1 \) has a Hamiltonian cycle \( C_1 \). As the maximum
number of faults in dimension 1 (that is, \( 4n - 1 \)) is strictly less than
\( 2[k^n/3] \), w.l.o.g. we may assume that there exist links \((x_1, y_1)\) and \((y_1, z_1)\)
of \( C_1 \) such that the links \((x_1, x_2), (y_1, y_2), (z_1, z_2), (x_2, y_2)\) and \((y_2, z_2)\) are
all healthy. If \( y_2 \) is incident with 2 or 3 healthy links in \( Q_2 \) then leave \( Q_2 \).
unchanged. Otherwise, by making previously healthy links in $Q_2$ that are incident with $y_2$ faulty, ensure that $y_2$ is incident with exactly 3 healthy links in this amended $Q_2$, 2 of which are $(x_2, y_2)$ and $(y_2, z_2)$: denote this amended $Q_2$ by $\tilde{Q}_2$. As $Q_2$ has at most $2n - 2$ faults, $\tilde{Q}_2$ has at most $4n - 5$ faults. Suppose that some node $w_2$ is incident with exactly 1 healthy link in $\tilde{Q}_2$. This must have been because $(y_2, w_2)$ was a healthy link in $Q_2$ and it was removed to form $\tilde{Q}_2$. Alter the construction of $\tilde{Q}_2$ so that $(y_2, w_2)$ is the third healthy link incident with $y_2$. As $Q_2$ has at most 1 node which is incident with only 2 healthy links, the resulting $\tilde{Q}_2$ is such that every node is incident with at least 2 healthy links. By the induction hypothesis applied to $Q_2$, if $y_2$ is incident with 2 or 3 healthy links, or to $\tilde{Q}_2$ otherwise, $Q_2$ has a Hamiltonian cycle $C_2$ containing at least 1 of the links $(x_2, y_2)$ and $(y_2, z_2)$. W.l.o.g. we may assume that $(x_2, y_2) \in C_2$. Denote by $D_2$ the cycle obtained by removing $(x_1, y_1)$ from $C_1$ and $(x_2, y_2)$ from $C_2$, and including the links $(x_1, x_2)$ and $(y_1, y_2)$ (this method of "joining" two cycles will be used extensively throughout). Then $D_2$ contains every node of $Q_1$ and $Q_2$ exactly once, and no other nodes.

All links of $D_2$ except for $(x_1, x_2)$ and $(y_1, y_2)$ are links in $Q_1$ or $Q_2$. Hence, there is potential to join $D_2$, as above, to a Hamiltonian cycle in $Q_3$ or $Q_k$. Again, as the maximum number of faults in dimension 1 is strictly less than $2[k^n/3]$, there exist links $(u, v)$ and $(v, w)$ in $D_2 \setminus \{(x_1, x_2), (y_1, y_2)\}$ such that $(u_2, u_3), (v_2, v_3), (w_2, w_3), (u_3, v_3)$ and $(v_3, w_3)$ are all healthy, if $(u, v)$ and $(v, w)$ are in $Q_2$, and $(u_1, u_k), (v_1, v_k), (w_1, w_k), (u_k, v_k)$ and $(v_k, w_k)$ are all healthy, if $(u, v)$ and $(v, w)$ are in $Q_1$. By proceeding as above, we can obtain a cycle $D_3$ containing either every node of $Q_1$, $Q_2$ and $Q_3$ or every node of $Q_1$, $Q_2$ and $Q_k$ exactly
once and no other nodes. Continuing in this way eventually yields a Hamiltonian cycle of $Q_{n+1}^k$.

Case (ii) Each $Q_i$ is such that every node is incident with at least 2 healthy links and some $Q_j$ has exactly $4n - 4$ faults.

W.l.o.g. we may assume that $j = 1$. Suppose that there is some fault $(x_1, y_1)$ of $Q_1$ such that $(x_1, x_2)$ and $(y_1, y_2)$ are healthy. Amend $Q_1$ so that $(x_1, y_1)$ is healthy and denote this amended $Q_1$ by $\tilde{Q}_1$. By the induction hypothesis applied to $\tilde{Q}_1$, $\tilde{Q}_1$ has a Hamiltonian cycle $C_1$ (which may or may not contain $(x_1, y_1)$). The cycle $C_1$ has an isomorphic copy $C_i$ in each $Q_i$, for $i = 2, 3, \ldots, k$. If $(x_1, y_1)$ is in $C_1$, the cycle $C_1$ can be joined to $C_2$ using the healthy links $(x_1, x_2)$ and $(y_1, y_2)$. Otherwise, because there are exactly 3 faults in dimension 1 and $\lceil k^n/2 \rceil > 3$, there is a link $(u_1, v_1)$ of $C_1$ such that $(u_1, u_2)$ and $(v_1, v_2)$ are healthy, and $C_1$ can be joined to $C_2$ using these links. If we denote the new cycle by $D_2$ then $D_2$ can be joined to $C_3$ in the same manner, and so on until we obtain a Hamiltonian cycle of $Q_{n+1}^k$.

On the other hand, suppose that for every fault $(x_1, y_1)$ of $Q_1$, at least one of $(x_1, x_2)$ and $(y_1, y_2)$, and at least one of $(x_1, x_k)$ and $(y_1, y_k)$ are faulty. Let $(x_1, y_1)$ be some fault of $Q_1$. As there are exactly 3 faults in dimension 1, every fault in $Q_1$ must be incident with either $x_1$ or $y_1$. But as the nodes $x_1$ and $y_1$ are incident with at most $4n - 5$ faults in $Q_1$ between them, this yields a contradiction.

Case (iii) There exists some $Q_i$ in which there is a node incident with exactly 1 healthy link in $Q_i$.

W.l.o.g. we may assume that the node $x_1$ in $Q_1$ is incident with exactly 1 healthy link, $(x_1, y_1)$, in $Q_1$. As $x_1$ is incident with $2n - 1$ faults: each $Q_i$, for $i = 2, 3, \ldots, k$, contains at most $2n - 3$ faults; there is no other node
in any $Q_i$, for $i = 2, 3, \ldots, k$, which is incident with less than 3 healthy links in that $Q_i$; and apart from $x_1$, there is no other node in $Q_1$ which is incident with less than 2 healthy links in $Q_1$. Also, as $x_1$ is incident with at least 2 healthy links in $Q_{n+1}^k$, we may suppose that $(x_1, x_2)$ is healthy. Consider $w_1$, one of the $2n - 1$ potential neighbours of $x_1$ in $Q_1$ for which the link $(x_1, w_1)$ is faulty. There are two scenarios.

**Case (iii)(a) $(w_1, w_2)$ is a healthy link.**

Make the previously faulty link $(x_1, w_1)$ healthy, and denote the amended $Q_1$ by $\tilde{Q}_1$. By the induction hypothesis applied to $\tilde{Q}_1$, $Q_1$ has a Hamiltonian path $P_1$ from $x_1$ to $w_1$. By making previously healthy links in $Q_2$ that are incident with $x_2$ faulty, and by making the link $(x_2, w_2)$ healthy (if necessary), ensure that $x_2$ is incident with exactly 2 healthy links in this amended $Q_2$, one of which is $(x_2, w_2)$; and denote this amended $Q_2$ by $\tilde{Q}_2$. $\tilde{Q}_2$ has at most $4n - 5$ faults and every node in $\tilde{Q}_2$ is incident with at least 2 healthy links in $\tilde{Q}_2$. Hence, by the induction hypothesis applied to $\tilde{Q}_2$, there exists a Hamiltonian path $P_2$ in $Q_2$ from $x_2$ to $w_2$. Join $P_1$ and $P_2$ using the healthy links $(x_1, x_2)$ and $(w_1, w_2)$ to form a cycle $D_2$. By proceeding as we did earlier, $D_2$ can eventually be extended to a Hamiltonian cycle of $Q_{n+1}^k$.

**Case (iii)(b) All links from every such $w_1$ to its corresponding node $w_2$ in $Q_2$ are faulty.**

This accounts for another $2n - 1$ faults in $Q_{n+1}^k$. Also, if $(x_1, x_k)$ is healthy then by symmetry we are in Case (iii)(a) (as all but at most 1 link of the form $(w_1, w_k)$ is healthy). Hence, we may assume that $(x_1, x_k)$ is faulty and this accounts for all the faults in $Q_{n+1}^k$. Consequently, $(y_1, y_2)$ and $(y_1, y_k)$ are both healthy links (recall that $(x_1, y_1)$ is the only healthy link of $Q_1$ incident with $x_1$). Let $w_1$ be some potential neighbour of $x_1$ in $Q_1$.
for which the link \((x_1, w_1)\) is faulty. Amend \(Q_1\) by making the faulty link \((x_1, w_1)\) healthy, and by the induction hypothesis applied to this amended \(Q_1\), there is a Hamiltonian path \(P_1\) in \(Q_1\) from \(x_1\) to \(w_1\). Rename the nodes of \(P_1\) as \(x_{1,1} = x_1, x_{1,2} = y_1, x_{1,3}, \ldots, x_{1,k^n} = w_1\), and note that in each \(Q_i, i \geq 2\), there is a corresponding Hamiltonian path \(P_i\) which can be extended to a Hamiltonian cycle \(C_i\) of \(Q_i\) (as \((x_i, w_i)\) is healthy in \(Q_i\)). Rename the nodes of \(C_i\) as \(x_{i,1} = x_i, x_{i,2} = y_i, x_{i,3}, \ldots, x_{i,k^n} = w_i\), for each \(i \geq 2\).

For ease of notation, denote \(k^n\) by \(m\). Suppose \(k\) is even. Then the following is a Hamiltonian cycle in \(Q_{n+1}^k\):

\[
(x_{1,1}, x_{2,1}, \ldots, x_{k,1}, x_{k,2}, x_{k,3}, x_{1,3}, x_{1,4}, \ldots, x_{1,m}, x_{k,m}, x_{k-1,m}, \ldots, x_{2,m},
\]

\[
x_{2,m-1}, x_{3,m-1}, \ldots, x_{k,m-1}, x_{k,m-2}, x_{k-1,m-2}, \ldots, x_{2,m-2}, x_{2,m-3},
\]

\[
x_{3,m-3}, \ldots, x_{k,m-3}, x_{k,m-4}, \ldots, x_{k,4}, x_{k-1,4}, \ldots, x_{2,4}, x_{2,3}, x_{3,3}, \ldots,
\]

\[
x_{k-1,3}, x_{k-1,2}, x_{k-2,2}, \ldots, x_{2,2}, x_{1,2}, x_{1,1})
\]

(see Fig. 5.2 where some of the healthy links between the \(Q_i\)'s are shown and bold links denote the links of the Hamiltonian cycle). If \(k\) is odd then the following is a Hamiltonian cycle in \(Q_{n+1}^k\):

\[
(x_{1,1}, x_{2,1}, \ldots, x_{k,1}, x_{k,2}, x_{k-1,2}, \ldots, x_{2,2}, x_{2,3}, x_{3,3}, \ldots, x_{k,3}, x_{k,4}, x_{k-1,4}, \ldots,
\]

\[
x_{2,4}, x_{2,5}, \ldots, x_{2,m}, x_{3,m}, \ldots, x_{2,m}, x_{k,m}, \ldots, x_{1,m}, x_{1,m-1}, \ldots, x_{1,2}, x_{1,1})
\]

(see Fig. 5.3).

Case (iv) There exists some \(Q_i\) in which there is a node incident with no healthy links in \(Q_i\).

W.l.o.g. we may assume that \(x_1\) is incident with no healthy links in \(Q_1\). As \(x_1\) is incident with at least 2 healthy links in \(Q_{n+1}^k\), the links \((x_1, x_2)\) and \((x_1, x_k)\) must be healthy. There are at least \(2n\) faults in \(Q_1\), and so
there must be at most $2n - 4$ faults distributed amongst $Q_2, Q_3, \ldots, Q_k$. Hence, apart from $x_1$ there are no nodes which are incident with less than 3 healthy links in their respective copy of $Q^k_n$. The node $x_1$ has $2n$ potential neighbours in $Q_1$. A simple counting argument yields that there exist distinct potential neighbours $y_1$ and $z_1$ of $x_1$ in $Q_1$ such that $(y_1, y_2)$ and $(z_1, z_k)$ are healthy. Amend $Q_1$ so that the previously faulty links $(x_1, y_1)$ and $(x_1, z_1)$ are now healthy. Applying the induction hypothesis to this amended $Q_1$ yields a path $P_1$ in $Q_1$ from $y_1$ to $z_1$ upon which every node of $Q_1$ appears exactly once, except for $x_1$ which does not appear at all.
By making previously healthy links in $Q_2$ that are incident with $x_2$ faulty, and by making the link $(x_2, y_2)$ healthy (if necessary), ensure that $x_2$ is incident with exactly 2 healthy links in this amended $Q_2$, one of which is $(x_2, y_2)$; and denote this amended $Q_2$ by $\tilde{Q}_2$. As $Q_2$ has at most $2n - 4$ faults, $\tilde{Q}_2$ has at most $4n - 6$ faults and every node in $\tilde{Q}_2$ is incident with at least 2 healthy links in $\tilde{Q}_2$. Hence, by the induction hypothesis applied to $\tilde{Q}_2$, there exists a Hamiltonian path $P_2$ in $\tilde{Q}_2$ from $x_2$ to $y_2$. Similarly, there is a Hamiltonian path $P_k$ in $Q_k$ from $x_k$ to $z_k$. Let $D$ be the cycle obtained by joining $P_1$, $P_2$ and $P_k$ using the healthy links $(x_1, x_2)$, $(y_1, y_2)$, $(x_1, x_k)$ and $(z_1, z_k)$. By proceeding as above, using the
fact that the maximum number of faults in dimension 1 (that is, $2n - 1$) is strictly less than $[k^n/2]$, $D$ can eventually be extended to a Hamiltonian cycle of $Q_{n+1}^k$.

It remains to show that the result holds for the base cases of the induction; namely, when $n = 2$ and $k \geq 4$, and when $n = 3$ and $k = 3$.

**Lemma 5.2** If $Q_2^k$, where $k \geq 4$, has 3 faulty links and is such that every node is incident with at least 2 healthy links then $Q_2^k$ has a Hamiltonian cycle.

**Proof** There exists some dimension, say dimension 1, that contains at least 2 faults. Partition $Q_2^k$ over dimension 1 to obtain $k$ copies of $Q_1^k$, namely $Q_1, Q_2, \ldots, Q_k$.

**Case (i)** All faults are in dimension 1.

Consider the cycle $Q_1$ of length $k$. As there are 3 faults in dimension 1, w.l.o.g. there exists a link $(x_1, y_1)$ of $Q_1$ such that the links $(x_1, x_2)$ and $(y_1, y_2)$ are both healthy: join $Q_1$ and $Q_2$ using these links. By proceeding in this way with $Q_3, \ldots, Q_k$, we obtain a Hamiltonian cycle of $Q_2^k$.

**Case (ii)** Dimension 1 has exactly two faults.

W.l.o.g. the only fault not in dimension 1 may be assumed to be $(x_1, y_1)$ in $Q_1$. If the links $(x_1, x_2)$ and $(y_1, y_2)$ are both healthy or the links $(x_1, x_k)$ and $(y_1, y_k)$ are both healthy then we can join $Q_1$ with $Q_2$ or $Q_k$, respectively, as in Case(i), and extend this cycle to a Hamiltonian cycle of $Q_2^k$.

Hence, w.l.o.g. we may assume that the links $(x_1, x_2)$ and $(y_1, y_k)$ are both faulty. If $k$ is even then there exists a Hamiltonian cycle in $Q_2^k$ as pictured in Fig. 5.2 (in that picture, $x_{1,3}$, $x_{1,2}$, $x_{2,3}$ and $x_{k,2}$ play the roles of $x_1$, $y_1$, $x_2$ and $y_k$, respectively). If $k$ is odd then there exists a
Hamiltonian cycle in $Q^5_2$ as pictured in Fig. 5.3 (in that picture, $x_{1,m}$, $x_{1,1}$, $x_{2,m}$ and $x_{k,1}$ play the roles of $x_1$, $y_1$, $x_2$ and $y_k$, respectively). □

**Lemma 5.3** If $Q^3_2$ has 3 faulty links and is such that every node is incident with at least 2 healthy links then $Q^3_2$ has a Hamiltonian cycle unless these 3 faulty links form a cycle of length 3.

**Proof** There exists some dimension, say dimension 1, that contains at least 2 faults. Partition $Q^3_2$ over dimension 1 to obtain 3 copies of $Q^3_1$, namely $Q_1$, $Q_2$ and $Q_3$. We may assume that either $Q_1$ contains 1 fault or all faults are in dimension 1. Denote the nodes of $Q_i$ by $x_i$, $y_i$ and $z_i$, for $i = 1, 2, 3$.

**Case (i)** $Q_1$ contains 1 fault.

W.l.o.g. we may assume that the fault in $Q_1$ is $(x_1, y_1)$.

**Case (i)(a)** The links $(x_1, x_2)$ and $(y_1, y_2)$ are healthy.

Form the cycle $C = (x_1, z_1, y_1, y_2, z_2, x_2, x_1)$ in $Q^3_2$. There are 2 possibilities: either one of the sets of pairs

$$\{(x_1, x_3), (z_1, z_3)\}, \{(y_1, y_3), (z_1, z_3)\}, \{(x_2, x_3), (z_2, z_3)\}, \{(y_2, y_3), (z_2, z_3)\}$$

consists of 2 healthy links or the faulty links in dimension 1 are $(z_1, z_3)$ and $(z_2, z_3)$. In the former case, the cycle $C$ can be joined to the cycle $(x_3, y_3, z_3, x_3)$ using the pair of healthy links to obtain a Hamiltonian cycle in $Q^3_2$: in the latter case, we can define the cycle $C'$ to be $(x_1, z_1, y_1, y_3, z_3, x_3, x_1)$ and, by symmetry, the former case applies.

**Case (i)(b)** At least one of the links $(x_1, x_2)$ and $(y_1, y_2)$ is faulty.

By symmetry, we may also assume that at least one of $(x_1, x_3)$ and $(y_1, y_3)$ is faulty (as otherwise we are in Case (i)(a)); so this accounts for all faults in $Q^3_2$. The only configuration possible, up to isomorphism, is that
in Fig. 5.4(a), and so there is a Hamiltonian cycle as depicted in that figure. (In Fig. 5.4(a), the nodes $x_1, y_1$ and $z_1$ of $Q_1$ form the central column, with the other 2 columns similarly depicting the nodes of $Q_2$ and $Q_3$. Faults are denoted by missing links and links of the Hamiltonian cycle are drawn in bold.)

Case (ii) All faults are in dimension 1.

Up to isomorphism, there are 6 different configurations possible, as depicted in Fig. 5.4(b-g), with Hamiltonian cycles as shown except for Fig. 5.4(g) where no such Hamiltonian cycle exists. □

![Fig. 5.4: The different configurations for $Q_2^3$.](image-url)
Lemma 5.4 If $Q_3^3$ has 7 faulty links and is such that every node is incident with at least 2 healthy links then $Q_3^3$ has a Hamiltonian cycle.

Proof Case (i) $Q_3^3$ contains faults forming a cycle $C$ of length 3.

All of the faults in $C$ must appear in the same dimension, dimension 1 say. Partition $Q_3^3$ across dimension 1 to obtain 3 copies of $Q_3^3$, namely $Q_1$, $Q_2$ and $Q_3$, and let the faulty links in $C$ be $(x_1, x_2)$, $(x_2, x_3)$ and $(x_3, x_1)$. We may assume that $Q_1$ contains the most faults amongst these copies, then $Q_2$ and then $Q_3$.

Case (i)(a) $Q_1$ contains faults forming a cycle $D$ of length 3.

If $Q_1$ has a node $y_1$ incident with less than 2 healthy links in $Q_1$ then $y_1$ must appear on the cycle $D$ and $y_1 \neq x_1$ (as otherwise $x_1$ would be incident with less than 2 healthy links in $Q_3^3$). In this case, make a previously faulty link $(y_1, z_1)$ of the cycle $D$ healthy, where $z_1 \neq x_1$, and denote this amended $Q_1$ by $\bar{Q}_1$ (note that all nodes in $\bar{Q}_1$ are incident with at least 2 healthy links in $\bar{Q}_1$).

If every node of $Q_1$ is incident with at least 2 healthy links in $Q_1$ then make a link $(y_1, z_1)$ of the cycle $D$ that is not incident with $x_1$ healthy, and denote this amended $Q_1$ by $\bar{Q}_1$.

By Lemma 5.3, $\bar{Q}_1$ has a Hamiltonian cycle $E_1$. Moreover, as $Q_1$ contains at least 3 faulty links, either $(y_1, y_2)$ and $(z_1, z_2)$ are both healthy or $(y_1, y_3)$ and $(z_1, z_3)$ are both healthy: w.l.o.g. we may assume that it is $(y_1, y_2)$ and $(z_1, z_2)$. By making previously healthy links in $Q_2$ faulty and possibly by making the faulty link $(y_2, z_2)$ healthy (if it is indeed faulty), ensure that $y_2$ is incident with exactly 2 healthy links in this amended $Q_2$, one of which is $(y_2, z_2)$, so that this amended $Q_2$ does not contain faults forming a cycle of length 3: we denote this amended $Q_2$ by $\bar{Q}_2$. By Lemma 5.3, $\bar{Q}_2$ has a Hamiltonian cycle $E_2$. Join $E_1$ and $E_2$ using the
links \((y_1, y_2)\) and \((z_1, z_2)\) to obtain a cycle \(F\) of \(Q_3^3\) consisting entirely of healthy links.

Let \(E_3\) be the isomorphic copy of \(E_2\) in \(Q_3\) (note that \(Q_3\) has no faults). As \(E_3\) has length 9 and dimension 1 contains at most 4 faults, w.l.o.g. we may assume that there are links \((u_2, v_2)\) and \((u_3, v_3)\) in \(F\) and \(E_3 \setminus \{(y_3, z_3)\}\), respectively, such that \((u_2, u_3)\) and \((v_2, v_3)\) are healthy. Join \(F\) and \(E_3\) using the links \((u_2, u_3)\) and \((v_2, v_3)\) to obtain a Hamiltonian cycle of \(Q_3^3\).

Case (i)(b) \(Q_1\) does not contain faults forming a cycle \(D\) of length 3.

**Case (i)(b) \(Q_1\) does not contain faults forming a cycle \(D\) of length 3.**

Note that the proofs of Cases (i), (ii), (iii) and (iv) of the main theorem hold for \(Q_3^3\) except that: throughout, instead of appealing to an inductive hypothesis, we use Lemma 5.3; in Case (i), we assume that dimension 1 contains at most 5 faults; and in Case (iii(a)), when building \(Q_2\) we must ensure that we do not introduce a cycle of faults of length 3 (this can be done as \(Q_2\) has at most 1 fault). Consequently, we are left with one scenario to consider: when each \(Q_i\) is such that every node is incident with at least 2 healthy links and when dimension 1 contains 6 or 7 faults.

Let (a new) 3-ary 2-cube \(Q_2^3\) be such that there is a fault \((x, y)\) in \(Q_2^3\) if and only if there is a fault \((x_i, y_i)\) in \(Q_i\), for some \(i \in \{1, 2, 3\}\). Then \(Q_2^3\) has at most 2 faults and by Lemma 5.3, there is a Hamiltonian cycle \(C\). For each \(i \in \{1, 2, 3\}\), let \(C_i\) be the isomorphic copy of \(C\) in \(Q_i\) (note that each \(C_i\) consists entirely of healthy links). Even if dimension 1 (of our original \(Q_3^3\)) contains 7 faults, there exists a pair of healthy links \\{\((u_1, u_2), (v_1, v_2)\)\} or \\{\((u_1, u_3), (v_1, v_3)\)\}, where \((u_1, v_1)\) is a link of \(C_1\): w.l.o.g. we may assume that these healthy links are \((u_1, u_2)\) and \((v_1, v_2)\). We can join \(C_1\) and \(C_2\) using these healthy links and then proceed similarly to join the resulting cycle to \(C_3\) and obtain a Hamiltonian cycle...
of $Q_3^3$.

Case (ii) $Q_3^3$ does not contain faults forming a cycle of length 3.

There exists a dimension, dimension 1 say, containing at least 3 faults. Partition $Q_3^3$ across dimension 1 to obtain 3 copies of $Q_3^3$, namely $Q_1$, $Q_2$ and $Q_3$. Let $Q_1$ contain the most faults amongst these copies, then $Q_2$ and then $Q_3$. Proceeding as in Case (i)(b) yields the result. \hfill \Box

The main theorem now follows by induction. \hfill \Box

The result in Theorem 5.1 is optimal in the following sense. Let $a$, $b$, $c$ and $d$ be 4 nodes in $Q_n^k$, where $k \geq 4$ and $n \geq 2$, or $k = 3$ and $n \geq 3$, such that there are links $(a, b)$, $(b, c)$, $(c, d)$ and $(d, a)$. Let the faults of $Q_n^k$ consist of those links incident with $a$ that are different from $(a, b)$ and $(a, d)$, and those links incident with $c$ that are different from $(b, c)$ and $(c, d)$. In particular, $Q_n^k$ has $4n - 4$ faults and every node is incident with at least 2 healthy links, but the four links $(a, b)$, $(a, d)$, $(c, b)$, and $(c, d)$ form a cycle by themselves, making a Hamiltonian cycle impossible in this $Q_n^k$. Thus, making our result optimal.

5.3 Tolerating Faults in Hypercubes

As regards hypercubes, it was shown in [22] that there exists a Hamiltonian cycle in a binary $n$-cube $B_n$ with at most $2n - 5$ faulty links where every node is incident with at least 2 healthy links. The base cases of the induction in Chan and Lee’s paper are where $n = 3, 4,$ and 5. In this section, we will prove the same result using the approach of the previous section. However, our approach is simpler and reduces the induction base to only one, where $n = 3$.
Theorem 5.5 Let $n \geq 3$. If $B_n$ has at most $2n - 5$ faulty links and is such that every node is incident with at least 2 healthy links then $B_n$ has a Hamiltonian cycle.

Proof The proof is by induction on the dimension $n$. Note that given any binary $n$-cube $B_n$, we can partition $B_n$ over some dimension $i \in \{1, 2, \ldots, n\}$ and consider $B_n$ to consist of two isomorphic disjoint copies $B'$ and $B''$ of $B_{n-1}$ (where the isomorphism is the natural one) with corresponding nodes joined by $2^{n-1}$ links lying in dimension $i$. Throughout the proof of the theorem, if $x$ is a node in $B'$, say, then we often denote it by $x'$, and we refer to its corresponding node in $B''$ as $x''$.

We begin the proof of the theorem with a lemma.

Lemma 5.6 Let $n \geq 3$. If $B_n$ has at most $n - 2$ faulty links then $B_n$ has a Hamiltonian cycle.

Proof We proceed by induction on the dimension $n$. When $n = 3$ and $B_n$ contains 1 faulty link, partition $B_3$ over the dimension that contains this faulty link. This results in two healthy disjoint copies $B'$ and $B''$ of $B_2$. As each copy is a cycle of length 4 and $4/2 > 1$, there exists some link $(x', y')$ in $B'$ with the property that $(x', x'')$ and $(y', y'')$ are both healthy. The Hamiltonian cycle of $B_3$ consists of the Hamiltonian path from $x'$ to $y'$ in $B'$, the Hamiltonian path from $x''$ to $y''$ in $B''$, and the two links $(x', x'')$ and $(y', y'')$.

Assume that the lemma holds for $B_n$ for some $n \geq 3$. Let $B_{n+1}$ have $(n + 1) - 2 = n - 1$ faulty links. Then there exists some dimension, say dimension 1, which contains at least 1 fault. Partition $B_{n+1}$ over this dimension and consider the two disjoint copies $B'$ and $B''$ of $B_n$ with faults total at most $n - 2$. Let a new $B_n$ contain all the $n - 2$ faulty
links of the $B'$ and the $B''$. Then by the induction hypothesis, the new $B_n$ is Hamiltonian. Copy the Hamiltonian cycle of the new $B_n$ to $B'$ and $B''$. This yields 2 isomorphic cycles $C'$ in $B'$ and $C''$ in $B''$ each of length $2^n$. As $\left\lfloor \frac{2^n}{2} \right\rfloor > n - 1$, the total number of faulty links, for all $n \geq 3$, there exists some link $(x', y')$ in $C'$ with the property that $(x', x'')$ and $(y', y'')$ are both healthy. The Hamiltonian cycle of $B_{n+1}$ consists of the Hamiltonian path from $x'$ to $y'$ in $C'$, the Hamiltonian path from $x''$ to $y''$ in $C''$, and the two links $(x', x'')$ and $(y', y'')$. □

Next, consider a binary hypercube $B_n$ with at most $2n - 5$ faulty links and is such that every node is incident with at least 2 healthy links. To show that this binary hypercube is Hamiltonian, we proceed by induction on $n$. When $n = 3$ and $B_n$ has 1 faulty link, we can proceed as the induction base of Lemma 5.6.

Assume that the result holds for $B_n$, for some $n \geq 3$. Let $B_{n+1}$ have at most $2(n+1) - 5 = 2n - 3$ faulty links and be such that every node is incident with at least 2 healthy links. Assume w.l.o.g. that dimension 1 contains the most faults from amongst the $n+1$ dimensions. Partition $B_{n+1}$ over dimension 1 and consider the two disjoint copies $B'$ and $B''$ of $B_n$.

Case (i) Dimension 1 contains at least $n - 1$ faults.

In this case, $B'$ and $B''$ will have faults total at most $n - 2$. By proceeding as in Lemma 5.6, we can construct two isomorphic disjoint Hamiltonian cycles $C'$ in $B'$ and $C''$ in $B''$. As $\left\lfloor \frac{2^n}{2} \right\rfloor > 2n - 3$ for all $n \geq 3$, there exists some link $(x', y')$ in $C'$ with the property that $(x', x'')$ and $(y', y'')$ are both healthy. The Hamiltonian cycle of $B_{n+1}$ consists of the Hamiltonian path from $x'$ to $y'$ in $C'$, the Hamiltonian path from $x''$ to $y''$ in $C''$, and the two links $(x', x'')$ and $(y', y'')$. 90
Case (ii) Dimension 1 contains at least 1 fault and at most \( n - 2 \) faults.

In this case, \( B' \) and \( B'' \) will have faults total at most \( 2n - 4 \). W.l.o.g. we may assume that \( B' \) has more faults than \( B'' \).

**Case (ii)(a)** Each of the \( B' \) and \( B'' \) is such that every node is incident with at least 2 healthy links and no of the two cubes contains \( 2n - 4 \) faults.

In this case, as \( B' \) has more faults than \( B'' \), \( B'' \) has at most \( n - 2 \) faults. By the induction hypothesis, \( B' \) has a Hamiltonian cycle \( C' \). As \( \lceil 2n/3 \rceil > n - 2 \) for all \( n \geq 3 \), there exists links \((x', y')\) and \((y', z')\) of \( C' \) with the property that \((x', z''), (y', y'')\) and \((z', z'')\) are all healthy. If at least one of the links \((x'', y'')\) and \((y'', z'')\) is faulty (we may assume w.l.o.g. that \((x'', y'')\) is the faulty one), then by making the link \((x'', y'')\) healthy the amended \( B'' \) will have at most \( n - 3 \) faulty links and each node of the amended \( B'' \) is incident with at least 3 healthy links. By making previously healthy links in the amended \( B'' \) that are incident with \( x'' \) faulty, ensure that \( x'' \) is incident with exactly two healthy links in this amended \( B'' \), one of which is \((x'', y'')\); and denote this amended \( B'' \) by \( \tilde{B}'' \). As \( B'' \) has at most \( n - 2 \) faults, \( \tilde{B}'' \) has at most \( 2n - 5 \) faults and every node in \( \tilde{B}'' \) is incident with at least 2 healthy links in \( \tilde{B}'' \). Hence, by the induction hypothesis applied to \( \tilde{B}'' \), there exists a Hamiltonian path in \( \tilde{B}'' \) from \( x'' \) to \( y'' \). The Hamiltonian cycle of \( B_{n+1} \) consists of the Hamiltonian path from \( x' \) to \( y' \) in \( B' \), the Hamiltonian path from \( x'' \) to \( y'' \) in \( B'' \), and the two links \((x', x'')\) and \((y', y'')\).

Otherwise (both \((x'', y'')\) and \((y'', z'')\) are healthy), if \( y'' \) is incident with 2 or 3 healthy links in \( B'' \) then leave \( B'' \) unchanged. Otherwise, by making previously healthy links in \( B'' \) that are incident with \( y'' \) faulty, ensure that \( y'' \) is incident with exactly 3 healthy links in this amended
$B''$, 2 of which are $(x'',y'')$ and $(y'',z'')$: denote this amended $B''$ by $\tilde{B}''$. As $B''$ has at most $n-2$ faults, $\tilde{B}''$ has at most $2n-5$ faults. Suppose that some node $w''$ is incident with exactly 1 healthy link in $\tilde{B}''$. This must have been because $(y'',w'')$ was a healthy link in $B''$ and it was removed to form $\tilde{B}''$. Alter the construction of $\tilde{B}''$ so that $(y'',w'')$ is the third healthy link incident with $y''$. As $B''$ has at most 1 node which is incident with only two healthy links, the resulting $\tilde{B}''$ is such that every node is incident with at least two healthy links. By the induction hypothesis applied to $B''$, if $y''$ is incident with 2 or 3 healthy links, or to $\tilde{B}''$ otherwise, $B''$ has a Hamiltonian cycle $C''$ containing at least one of the links $(x'',y'')$ and $(y'',z'')$. W.l.o.g. we may assume that $(x'',y'') \in C''$. The Hamiltonian cycle of $B_{n+1}$ consists of the Hamiltonian path from $x'$ to $y'$ in $B'$, the Hamiltonian path from $x''$ to $y''$ in $B''$, and the two links $(x',x'')$ and $(y',y'')$.

Case (ii)(b) Each of $B'$ and $B''$ is such that every node is incident with at least 2 healthy links and $B'$ has exactly $2n-4$ faults.

W.l.o.g. we may assume that for every fault $(x',y')$ of $B'$, at least one of $(x',x'')$ and $(y',y'')$ is faulty. Let $(x',y')$ be some fault of $B'$. As there is exactly 1 fault in dimension 1, we may assume that this fault is $(x',x'')$ and every fault in $B'$ is incident with $x'$. As the node $x'$ is incident with at most $n-2$ faults in $B'$, this yields a contradiction. Hence, there exists a fault $(u',v')$ of $B'$ such that $(u',u'')$ and $(v',v'')$ are healthy. Make the previously faulty link $(u',v')$ of $B'$ healthy, and denote this amended $B'$ by $\tilde{B}'$. By the induction hypothesis applied to $\tilde{B}'$, there is a Hamiltonian cycle $C'$ in $\tilde{B}'$. Since $B''$ is healthy, there is an isomorphic copy of $C'$ in $B''$. Denote this cycle by $C''$. If $(u',v') \in C'$ then there is a Hamiltonian path in $B'$ from $u'$ to $v'$. The Hamiltonian cycle of $B_{n+1}$ consists of the
Hamiltonian path from \( u' \) to \( v' \) in \( C' \), the Hamiltonian path from \( u'' \) to \( v'' \) in \( C'' \), and the two links \((u', u'')\) and \((v', v'')\). Otherwise (If \((u', v') \notin C'\)), as \([2^n/2] > 1\), there exists some link \((s', t')\) in \( C' \) such that \((s', s'')\) and \((t', t'')\) are both healthy. The Hamiltonian cycle of \( B_{n+1} \) consists of the Hamiltonian path from \( s' \) to \( t' \) in \( C' \), the Hamiltonian path from \( s'' \) to \( t'' \) in \( C'' \), and the two links \((s', s'')\) and \((t', t'')\).

**Case (ii)(c)** There is a node \( u' \) in \( B' \) such that \( u' \) is incident with exactly 1 healthy link.

As \( u' \) is incident with \( n - 1 \) faults in \( B' \), \( B'' \) contains at most \( n - 3 \) faults; there is no other node in \( B'' \) which is incident with less than 3 healthy links in \( B'' \); and apart from \( u' \), there is no other node in \( B' \) that is incident with less than 2 healthy links in \( B' \). Also, as \( u' \) is incident with at least 2 healthy links in \( B_{n+1}, (u', u'') \) has to be healthy. Since \( u' \) is incident with exactly \( n - 1 \) faulty links in \( B' \) and \( n - 1 > n - 2 \), there exists some faulty link \((u', v')\) in \( B' \) such that both \((u', u'')\) and \((v', v'')\) are healthy. Make the previously faulty link \((u', v')\) of \( B' \) healthy, and denote this amended \( B' \) by \( B' \). By the induction hypothesis applied to \( B' \), there is a Hamiltonian path from \( u' \) to \( v' \) in \( B' \). By making previously healthy links in \( B'' \) that are incident with \( u'' \) faulty and by making the link \((u'', v'')\) healthy (if necessary), ensure that \( u'' \) is incident with exactly 2 healthy links in this amended \( B'' \), one of which is \((u'', v'')\); and denote this amended \( B'' \) by \( B'' \). As \( B'' \) has at most \( n - 3 \) faults, \( B'' \) has at most \( 2n - 5 \) faults and every node in \( B'' \) is incident with at least 2 healthy links in \( B'' \). Hence, by the induction hypothesis applied to \( B'' \), there exists a Hamiltonian path in \( B'' \) from \( u'' \) to \( v'' \). The Hamiltonian cycle of \( B_{n+1} \) consists of the Hamiltonian path from \( u' \) to \( v' \) in \( B' \), the Hamiltonian path from \( u'' \) to \( v'' \) in \( B'' \), and the two links \((u', u'')\) and \((v', v'')\). □
Now consider a $B_n$ with node $a$ is incident with only 2 healthy links, $(a, b)$ and $(a, d)$, and node $c$ is incident with only 2 healthy links, $(c, b)$ and $(c, d)$. Then $B_n$ contains $2n - 4$ faulty links and every node in $B_n$ is incident with at least 2 healthy links. However, the four links $(a, b), (a, d), (c, b),$ and $(c, d)$ form a cycle by themselves, making a Hamiltonian cycle impossible for $n \geq 3$. Thus, making the above result optimal.

### 5.4 Complexity Issues

As regards complexity, it was shown in [22] that the problem of deciding whether a faulty binary $n$-cube has a Hamiltonian cycle is NP-complete. In more detail, let HCFH denote the problem whose instances of size $N$ are faulty hypercubes on $N$ nodes and whose yes-instances are faulty hypercubes which have a Hamiltonian cycle (note that HCFH has no instances of size $N$ when $N$ is not a power of 2). It was shown in [22] that there is a polynomial-time reduction from the well-known NP-complete problem 3-Satisfiability (see [43]) to HCFH.

Let HCFH($k$) denote the problem whose instances of size $N$ are faulty $k$-ary $n$-cubes on $N$ nodes (and so $N = k^n$) and whose yes-instances are faulty $k$-ary $n$-cubes which have a Hamiltonian cycle. Note that there is one problem HCFH($k$) for each $k > 2$ (with HCFH(2) being a reformulation of HCFH).

**Theorem 5.7** The problem HCFH($k$) is NP-complete, for each $k \geq 2$.

**Proof** We begin with a lemma.

**Lemma 5.8** When $k \geq 3$ and $n \geq 2$, the nodes of $Q^k_n$ can be partitioned as the disjoint union $U^k_n \cup V^k_n$ such that:
• \(|U^k_n| = 2^n\) and \(|V^k_n| = k^n - 2^n\)

• the subgraph of \(Q^k_n\) induced by \(U^k_n\) contains \(B_n\) as a subgraph

• the subgraph of \(Q^k_n\) induced by \(V^k_n\) contains a path \(P^k_n\) of length \(k^n - 2^n - 1\) as a subgraph, with terminal nodes \(x'\) and \(y'\) (where no node appears more than once on \(P^k_n\))

• there exists a link \((x, y)\) of \(B_n\) such that \((x, x')\) and \((y, y')\) are links of \(Q^k_n\).

**Proof** We proceed by induction on \(n\): the base case when \(n = 2\) is straightforward (no matter whether \(k\) is odd or even). Suppose that the result holds for some \(n \geq 2\). Partitioning \(Q^k_{n+1}\) over dimension 1 yields \(k\) copies of \(Q^k_n\), namely \(Q_1, Q_2, \ldots, Q_k\). By the induction hypothesis, \(Q_1\) contains a copy of \(B_n\) and a path \(P^k_n\), as in the statement of the lemma, with \(Q_2, Q_3, \ldots, Q_k\) containing isomorphic copies (where the isomorphism is the natural one).

Consider the binary \((n+1)\)-cube \(B_{n+1}\) of \(Q^k_{n+1}\) obtained by taking the disjoint union of the copies of \(B_n\) in \(Q_1\) and \(Q_2\) and joining corresponding nodes. Consider the path \(P^k_{n+1}\) built as follows.

• Join the paths \(P^k_n\) in \(Q_1, Q_2, \ldots, Q_k\) by starting from \(x'\) of the path \(P^k_n\) in \(Q_1\) and including the appropriate links from \(y'\) of one path to \(y'\) of the next, or \(x'\) of one path to \(x'\) of the next.

• Augment this path with the link \((x', x)\) or \((y', y)\), depending on whether \(k\) is even or odd, respectively, where \(x'\) and \(y'\) are the terminal nodes of the path \(P^k_n\) in \(Q_k\) and \((x, y)\) is the link in the copy of \(B_n\) in \(Q_k\).
• Augment this path with Hamiltonian paths in the copies of $B_n$ in $Q_k, Q_{k-1}, \ldots, Q_3$ from $x$ to $y$, or *vice versa*, as appropriate, with these paths joined using the appropriate links from $y$ of one path to $y$ of the next, or $x$ of one path to $x$ of the next.

Whether $k$ is even or odd, $p_{n+1}^k$ is a path of length $k^{n+1} - 2^{n+1} - 1$ from the node $x'$ of the path $p_n^k$ in $Q_1$ to the node $x$ of the binary $n$-cube $B_n$ in $Q_3$ (the construction can be visualised in Fig. 5.5). If we choose our link in $B_{n+1}$ to be that joining the node $x$ in the copy of $B_n$ in $Q_1$ with the node $x$ in the copy of $B_n$ in $Q_2$ then the lemma follows by induction.

□

![Figure 5.5: The construction in the proof of Lemma 5.8.](image)

Next, fix $k \geq 3$. In the reduction from 3-Satisfiability to HCFH in [22], note that the faulty hypercube constructed from an instance of 3-Satisfiability always has healthy links joining nodes of degree 2. Consequently, there is a polynomial-time algorithm which takes as input an instance of 3-Satisfiability and produces as output a faulty binary $n$-cube, for some $n$, and a healthy link $(x, y)$ with the property that the instance of 3-Satisfiability is a yes-instance if and only if

• the faulty binary $n$-cube has a Hamiltonian cycle
• a Hamiltonian cycle exists in the faulty binary \( n \)-cube if and only if there is a Hamiltonian cycle containing the link \((x, y)\).

Let \( B_n \) be a faulty binary \( n \)-cube that has a Hamiltonian cycle if and only if the healthy link \((x, y)\) appears in a Hamiltonian cycle. Let the faulty \( k \)-ary \( n \)-cube \( Q^k_n \) be defined as follows. The nodes of \( Q^k_n \) are partitioned as \( U^k_n \cup V^k_n \) such that:

- the subgraph of \( Q^k_n \) induced by the nodes of \( U^k_n \) is the faulty binary \( n \)-cube \( B_n \)
- the subgraph of \( Q^k_n \) induced by the nodes of \( V^k_n \) is a path of length \( k^n - 2^n - 1 \) from node \( x' \) to node \( y' \) (upon which no node appears more than once)
- the only other links are \((x, x')\) and \((y, y')\) (where \((x, y)\) is the specified link in \( B_n \)).

Such a partition exists by Lemma 5.8 and can clearly be constructed from \( B_n \) in polynomial-time (that is, time polynomial in \( N = 2^n \)). Consequently, the faulty \( B_n \) has a Hamiltonian cycle if and only if the faulty \( Q^k_n \) has a Hamiltonian cycle, and the result follows from the facts that 3-Satisfiability is NP-complete and HCFH(\( k \)) is in NP. \( \square \)
Chapter 6

Communication Algorithms

6.1 Introduction

One of the most important aspects of any large-scale general-purpose parallel computer is the speed and efficiency of its communication algorithms. This is because most large-scale general-purpose machines spend a large portion of their resources making sure that the right data gets to the right place within a reasonable amount of time. In this chapter, we consider the problems of: moving a data item from one processor to another processor; a single processor broadcasting the same data item to every other processor; a single processor sending different data items to every other processor; the simultaneous broadcast of the same data item from every processor to every other processor; and the simultaneous exchange of different data items between every pair of processors.

Most of the (varied) algorithms for such problems in the literature, e.g., [10, 13, 54, 79], relate to hypercubes. Consequently, we restrict ourselves to the $k$-ary $n$-cube where $k > 2$. All the algorithms presented in this chapter are dimensional. We mean by dimensional that at any one unit...
of time, data items are transmitted along links of only one dimension of the \( k \)-ary \( n \)-cube \( Q_n^k \).

We work under the following assumptions. Each processor has a copy of the same program and computation is synchronous. The size of each message to be transmitted is one packet and all packets have roughly equal size. The time taken to cross any link is the same for all packets and we take it to be one unit of time. All local computation can be done in negligible time and each processor has unlimited storage space. Packets can be transmitted along a link in one direction at any one time and their transmission is error free.

All of our algorithms are developed under the assumption of one-port I/O communication and store-and-forward routing. Whilst other models of parallel processing assume multi-port I/O communication (indeed, many modern routing algorithms have been proposed for a variety of machines under this assumption), most existing machines only support one-port I/O communication in hardware and modern routing algorithms, designed for multi-port systems, have not yet been implemented in commercial systems [64]. One-port machines require less storage capacity and the design of their processors is not as complicated as in the multi-port case. Moreover, the start-up time to initiate multiple links may be longer than the time to initiate only one link. It was also shown in [42] that for short messages multi-port communication algorithms can be slower than one-port communication algorithms.

### 6.2 Dimensional Routing

Consider the problem of a source processor \( s = (s_n, s_{n-1}, \ldots, s_1) \) wishing to send a data item to a destination processor \( d = (d_n, d_{n-1}, \ldots, d_1) \).
The routing algorithm presented in [33] is optimal for unidirectional $k$-ary $n$-cubes. With a simple modification, it can be improved to make it optimal for the $k$-ary $n$-cube model considered in this chapter as follows. The algorithm below sends a packet from processor $s$ to processor $d$ in time equal to the Lee distance, $D_L(s, d)$, between $s$ and $d$ by modifying the digits of $s$ one by one in order to transform the label $s$ into the label $d$. The algorithm is as follows:

\begin{verbatim}
let the dimensions in which $s$ and $d$ differ be \{i_1, \ldots, i_m\};
for every $i \in \{i_1, \ldots, i_m\}$ do
  for $j := 1$ to $D_L(s_i, d_i)$ do
    if $d_i - s_i = D_L(s_i, d_i)$ or $s_i - d_i = k - D_L(s_i, d_i)$ then
      send the packet from its current processor $(a_n, a_{n-1}, \ldots, a_i, \ldots, a_1)$ to processor
      $(a_n, a_{n-1}, \ldots, a_i + 1 \mod k, \ldots, a_1)$;
    else
      send the packet from its current processor $(a_n, a_{n-1}, \ldots, a_i, \ldots, a_1)$ to processor
      $(a_n, a_{n-1}, \ldots, a_i - 1 \mod k, \ldots, a_1)$;
  endfor
endfor
\end{verbatim}

Clearly this algorithm is optimal.

For example, in a 6-ary 3-cube, if a data item needs to be moved from the source processor $s = (0, 3, 5)$ to the destination processor $d = (4, 5, 1)$ then according to the above algorithm the progression will be:

\[(0, 3, 5) \rightarrow (0, 3, 0) \rightarrow (0, 3, 1) \rightarrow (0, 4, 1) \rightarrow (0, 5, 1) \rightarrow (5, 5, 1) \rightarrow (4, 5, 1)\]
6.3 Dimensional Single-node Broadcasting

Consider the problem of a source processor wishes to send its own data item to every other processor. Our single-node broadcast algorithm consists of \( n \) stages. In stage 1, the algorithm partitions the \( k \)-ary \( n \)-cube \( Q_n^k \) over dimension 1 into \( k \) isomorphic disjoint copies of \( Q_{n-1}^k \), where the corresponding nodes are joined in a cycle of length \( k \). Let the subcube that contains the source processor be named the source cube and denoted \( Q(0) \). Let the subcube of distance \( i \) from the left of \( Q(0) \) be denoted \( Q_l(i) \), and the subcube of distance \( i \) from the right of \( Q(0) \) be denoted \( Q_r(i) \).

Let \( \beta_l = \lfloor k^n/2 \rfloor \), and let \( \beta_r = \lfloor k^n/2 \rfloor \) if \( k \) is odd and \( \beta_r = \lfloor k^n/2 \rfloor - 1 \) if \( k \) is even. The most remote subcubes from \( Q(0) \) are \( Q_l(\beta_l) \) and \( Q_r(\beta_r) \) and they are adjacent. Note that if \( k \) is even then there is only one subcube of distance \( \lfloor k/2 \rfloor \) from \( Q(0) \). Therefore, we consider it on the left of \( Q(0) \). Let the source processor be denoted \( s(0) \), and its corresponding processor in \( Q_l(i) \) (resp. \( Q_r(i) \)) be denoted \( s_l(i) \) (resp. \( s_r(i) \)). After partitioning the \( Q_n^k \), stage 1 of our single-node broadcast algorithm proceeds as follows:

let the packet processor \( s(0) \) intends to broadcast
be denoted \( p \);
processor \( s(0) \) sends packet \( p \) to processor \( s_l(1) \);
do in parallel:
  • for \( i := 1 \) to \( \beta_l - 1 \) do
    processor \( s_l(i) \) sends packet \( p \)
    to processor \( s_l(i + 1) \);
  • for \( j := 0 \) to \( \beta_r - 1 \) do
    processor \( s_r(j) \) sends packet \( p \)
to processor $s_r(j + 1)$;

enddo

At the end of stage 1, processor $s_l(i)$, for $i = 1, 2, \ldots, \beta_l$, and processor $s_r(j)$, for $j = 1, 2, \ldots, \beta_r$, contain the broadcast packet $p$. The problem is now reduced to subcubes of dimension $n - 1$ ($Q(0)$ with source processor $s(0)$, $Q_l(i)$ with source processor $s_l(i)$ and $Q_r(j)$ with source processor $s_r(j)$).

In stage $i$, for $i = 2, 3, \ldots, n$, each resulting subcube from the previous stage performs in parallel the above algorithm. The algorithm clearly achieves its objective.

To analyse the time complexity for each stage of this algorithm, there are two cases to consider.

**Case(i) $k$ is even.** The time taken by each stage is

$$\max(1 + (\beta_l - 1), 1 + \beta_r) = \lfloor k/2 \rfloor.$$

**Case(ii) $k$ is odd.** The time taken by each stage is

$$\max(1 + (\beta_l - 1), 1 + \beta_r) = 1 + \lfloor k/2 \rfloor.$$

Therefore, the total time taken by this algorithm is $\lfloor k/2 \rfloor n$ if $k$ is even, and $n + \lfloor k/2 \rfloor n$ if $k$ is odd. Since the diameter of a $Q_n^k$ is also $\lfloor k/2 \rfloor n$, the above single-node broadcast algorithm is optimal when $k$ is even. While this can be shown to be non-optimal when $k$ is odd, it is within $n$ units of time of the optimal and has the virtue of being implemented simply.

However, if we assume that the system supports 2-port I/O communication, where a processor can transmit and receive data items along at most two incident links at any one unit of time, then with a simple
modification to stage 1, and hence the following stages, of the above al-
algorithm, the resulting single-node broadcast algorithm becomes optimal
for any $k > 2$. The modified stage 1 of the above algorithm is as follows:

let the packet processor $s(0)$ intends to broadcast
be denoted $p$;
do in parallel:
• for $i := 0$ to $\beta_t - 1$ do
  processor $s_t(i)$ sends packet $p$
  to processor $s_t(i + 1)$;
• for $j := 0$ to $\beta_r - 1$ do
  processor $s_r(j)$ sends packet $p$
  to processor $s_r(j + 1)$;
enddo

Since in the following stages, each resulting subcube from the previous
stage performs in parallel the above modified algorithm, the time taken
by each stage is $\max(\beta_t, \beta_r) = \lfloor k/2 \rfloor$. Therefore, the total time taken by
this modified single-node broadcast algorithm is $\lfloor k/2 \rfloor n$ which is optimal
for any $k > 2$. Fig. 6.1 illustrates the steps taken by the single-node
broadcast algorithm for the two models in a 5-ary 2-cube $Q_5^2$ where the
source processor $s(0) = 00$. Fig. 6.1(a) illustrates the steps taken by the
algorithm using one-port I/O and Fig. 6.1(b) illustrates the steps taken
by the algorithm using 2-port I/O. Note that in this figure, by using the
2-port I/O the algorithm reduces the steps from 6, which are taken by
the algorithm using one-port I/O, to 4.
Figure 6.1: The time steps \( t \) taken by the single-node broadcasting in \( Q_2^5 \) using: (a) one-port I/O (b) 2-port I/O.
6.4 Dimensional Multi-node Broadcasting

Consider the problem of each processor wishes to send its own data item to every other processor (essentially, every processor wishes to do a single-node broadcast simultaneously). Our multi-node broadcast algorithm consists of $n$ stages. In stage 1, each cycle in dimension 1 performs the following "daisy-chain" algorithm of [79]:

Each processor sets its own local packet named 'current' to be the data item it intends to broadcast;

for $i := 1$ to $k - 1$ do
    processor $(a_n, a_{n-1}, \ldots, a_1)$ sends the packet 'current' to processor $(a_n, a_{n-1}, \ldots, a_1 + 1 \ mod \ k)$;
    the packet 'current' of processor $(a_n, a_{n-1}, \ldots, a_1)$ is reset to be the packet just received by processor $(a_n, a_{n-1}, \ldots, a_1)$ and this packet is also retained locally;
endfor

In stage $i$, for $i = 2, 3, \ldots, n$, each cycle in dimension $i$ performs the above algorithm amongst its own processors except that as well as sending on its own data item, it sends on all the packets retained from the previous stage. The algorithm clearly achieves its objective.

Stage 1 is completed in time $k - 1$; stage 2 is completed in time $k(k - 1)$; stage 3 is completed in time $k^2(k - 1)$; and so on. In general, the time taken by stage $i$, $1 \leq i \leq n$, is $(k - 1)k^{i-1}$. Therefore, the total time taken by this algorithm is

$$\sum_{i=1}^{n} (k - 1)k^{i-1} = k^n - 1.$$

This algorithm is optimal as each processor can only receive at most one packet per unit of time and there are $k^n$ processors in total.
6.5 Dimensional Single-node Scattering

Consider the problem of a source processor wishes to send a different data item to every other processor. Our single-node scatter algorithm consists of \( n \) stages. In stage 1, the algorithm partitions the \( k \)-ary \( n \)-cube \( Q_n^k \) over dimension 1 into \( k \) isomorphic copies of \( Q_{n-1}^k \), namely \( Q_{n-1}^k(1), Q_{n-1}^k(2), \ldots, Q_{n-1}^k(k) \). Let the \( j \)th processor, \( 1 \leq j \leq k^{n-1} \), in \( Q_{n-1}^k(i), 1 \leq i \leq k \), be denoted \( s_{(i,j)} \); and its corresponding processor in \( Q_{n-1}^k(d) \) be denoted \( s_{(d,j)} \). After partitioning the \( Q_n^k \), stage 1 of our single-node scatter algorithm, where the source processor is \( s_{(1,1)} \), proceeds as follows:

let the packet intended for processor \( s_{(i,j)} \)
be denoted \( p_{(i,j)} \);
for \( j := 1 \) to \( k^{n-1} \) do
  for \( i := k \) downto 2 do
    processor \( s_{(1,1)} \) sends packet \( p_{(i,j)} \) to processor \( s_{(2,1)} \)
    and for every processor \( s_{(d,1)}, 2 \leq d \leq k-1 \), having retained a copy of any packet, \( p_{(r,j)} \), just received,
    sends this packet on to processor \( s_{(d+1,1)} \) if \( r > d \);
  endfor
endfor

At the end of stage 1, each processor \( s_{(i,1)} \), for \( i = 1, 2, \ldots, k \), contains the packets intended for every processor in \( Q_{n-1}^k(i) \). The problem is now reduced to \( Q_{n-1}^k(i) \) where the source processor is \( s_{(i,1)} \).

In stage \( i \), for \( i = 2, 3, \ldots, n \), each resulting subcube from the previous stage performs the above algorithm. Again the algorithm clearly achieves its objective.
The time taken by stage 1 is \((k - 1)k^{n-1}\). In general, the time taken by stage \(i\), \(1 \leq i \leq n\), is \((k - 1)k^{n-i}\). Therefore, the total time taken by this algorithm is
\[
\sum_{i=1}^{n}(k - 1)k^{n-i} = k^n - 1.
\]
This algorithm is optimal since the source processor must send out the \(k^n - 1\) different packets over one communication link at a time.

### 6.6 Dimensional Total Exchange

Consider the problem of every processor wishes to send a different data item to every other processor (in contrast to the multi-node broadcast where every processor wishes to send the same data item to every other processor). Let \(d(a, b)\) be the data item to be sent from processor \(a\) to processor \(b\). We assume that the resulting packet contains the data item \(d(a, b)\) and also details of the destination processor \(b\). Our total exchange algorithm consists of \(n\) stages and is as follows:

for \(i := 1\) to \(n\) do
  for every packet \([d(a, b), b]\) retained so far by processor \(c\) or originating at processor \(c\) do
    if the \(i\)th digit of \(b\) is \(b_i\) then
      route the packet \([d(a, b), b]\) from processor \(c\) to processor \((c_n, c_{n-1}, ..., b_i, ..., c_1)\) so that no interim processor, including processor \(c\), retains a copy of \([d(a, b), b]\);
  endfor
endfor

We have been intentionally vague as to how the routing, above, is achieved and we shall address this in more detail presently. Suffice to
say, however we choose to route the packets, by following the progress of
one particular packet in an execution of the above algorithm, it is clear
that the packet eventually ends up at its intended destination (note that
no copies of packets are ever made).

Returning to how we route the packets, let us first note that after
completion of stage $i$, exactly the packets going from the source processor
$(c_n, \ldots, c_{i+1}, x_i, \ldots, x_1)$, for some $x_i, \ldots, x_1$, to the destination processor
$(y_n, \ldots, y_{i+1}, c_i, \ldots, c_1)$, for some $y_n, \ldots, y_{i+1}$, are located at processor
$(c_n, \ldots, c_{i+1}, c_i, \ldots, c_1)$. Consequently, after every stage there are $k^n - 1$
packets located at each processor. In order to accomplish the routing
in stage $i + 1$, we use the routing algorithm in Section 6.2. In stage
$i + 1$, each processor routes packets around a cycle in dimension $i + 1$,
where the direction is dictated by whichever is the shortest path to the
intended destination. If we were to perform routings simultaneously in an
ad hoc fashion then we might find that processors had incoming packets
on two different incident links at the same time. So as to avoid such a
circumstance, we proceed as follows.

For any processor, every packet located at this processor has a unique
associated label $(x, y)$, where $x$ denotes the direction, $+1$, 0 or $-1$, around
the cycle it is to be routed in stage $i + 1$ (0 denotes “no move”) and $y$
denotes the length of the path up to its destination (if a packet can be
labelled $(+1, k/2)$ or $(-1, k/2)$, where $k$ is even, then we always choose
the label $(+1, k/2)$ so as to make any label unique). By symmetry, there
are exactly the same number of packets with identical labels located at
every processor. Hence, the routing in stage $i + 1$ proceeds as follows:

for each label $(x, y)$ do
  for each packet $p$ at processor $c$ with label $(x, y)$ do
route $p$ (according to the label $(x,y)$);
endfor
endfor

Note that routing packets simultaneously in this way ensures that there are no "input collisions" as described above.

Prior to stage $i+1$, the total number of packets labelled $(x,y)$ located at some processor $c$ is $k^{n-1}$, if $y \neq 0$, and $k^{n-1} - 1$ otherwise. Hence, the time taken to complete stage $i+1$ is

$$\sum_{j=0}^{k-1} k^{n-1} D_L(0,j) = k^{n-1} \sum_{j=0}^{k-1} D_L(0,j) = \begin{cases} k^{n-1}(k^2 - 1)/4 & \text{if } k \text{ is odd} \\ k^{n+1}/4 & \text{if } k \text{ is even} \end{cases}$$

and so the time taken by the above algorithm to complete a total exchange is $nk^{n-1}(k^2 - 1)/4$, if $k$ is odd, and $nk^{n+1}/4$, if $k$ is even.

In order to obtain a lower bound for the time taken to complete a total exchange, consider the processor $s$. It must necessarily send $k^n - 1$ packets, one to every other processor. Hence, the total number of packets sent, over all processors, in order to get the data items initially at processor $s$ to their destinations is

$$\sum_d D_L(s,d),$$

where $d$ ranges over all processors. This holds for every processor $s$, and so the total number of packets sent in order that a total exchange is completed is at least

$$k^n \sum_d D_L(s,d).$$

At any one time, any processor sends at most one packet, and so the
time taken to complete a total exchange is at least

\[ k^n \sum_d D_L(s, d)/k^n = \sum_d D_L(s, d). \]

By symmetry, it suffices to compute the above summation when \( s = (0, 0, \ldots, 0) \). Arrange the names of the processors in an \( k^n \times n \) matrix. Also by symmetry, every \( i \in \{0, 1, \ldots, k-1\} \) appears an identical number of times in the matrix and so

\[ \sum_d D_L(s, d) = (nk^n/k) \sum_{j=0}^{k-1} D_L(0, j), \]

which yields that our total exchange algorithm is optimal.
Chapter 7

Hamiltonian Cycles
and Applications to
Communication

7.1 Introduction

We show in this chapter how the Hamiltonian cycle of the $k$-ary $n$-cube network can be exploited to develop multi-node broadcast and single-node scatter communication algorithms for one-port I/O $k$-ary $n$-cube model. Although the algorithms presented in Section 7.2 and 7.3 are not dimensional, they complete the process in time equal to the time of those algorithms presented in Chapter 6. The dimensional multi-node broadcast algorithm of Section 6.4 requires all the $nk^n$ communication links of the $Q_n^k$ to complete the process while the multi-node broadcast algorithm of this chapter requires only $k^n$ communication links to complete the process. Moreover, the two algorithms are fault-tolerant as we will show later in this chapter.
All the communication problems are studied under the store-and forward communication model, i.e., a processor must store the entire message before it can be processed and retransmitted. We consider a model where splitting and recombining of messages is allowed. We assume that each processor has a copy of the same program and computation is synchronous. The time taken to cross any link is the same for all packets and we take it to be one unit of time. All local computation can be done in negligible time and each processor has unlimited storage space. All packets have roughly equal size. Packets can be transmitted along a link in one direction at any one time and their transmission is error free.

It was shown in [42] that for short messages multi-port communication algorithms can be slower than one-port communication algorithms. Therefore, we assume that messages of one-port I/O model are short and they are of size one packet, and messages of multi-port I/O model are long and they are of size $M$ packets where $M \geq n$.

Some parallel machines, e.g., the J-machine [27], support multi-port I/O model. We show in this chapter that the $k$-ary $n$-cube network can be decomposed into $n$ link-disjoint Hamiltonian cycles and then we show how these cycles can be used to develop multi-node broadcast and single-node scatter algorithms for machines that support multi-port I/O model.

### 7.2 Multi-node Broadcasting for one-port I/O Model

The following multi-node broadcast algorithm exploits the Hamiltonian cycle of the $k$-ary $n$-cube to perform the "daisy-chain" algorithm as fol-
each processor generates the $k$-ary Gray codes of dimension $n$ as detailed in Section 3.2; let the resulting Hamiltonian cycle be $s_0, s_1, \ldots, s_{k^n - 1}$ where node $s_i$ is linked to node $s_{i+1} \mod k^n$; each processor sets its own local packet named 'current' to be the data item it intends to broadcast; for $j := 1$ to $k^n - 1$ do 

processor $s_i$ sends the packet 'current' to processor $s_{i+1} \mod k^n$; the packet 'current' of processor $s_i$ is reset to be the packet just received by processor $s_i$ and this packet is also retained locally; endfor

The algorithm clearly achieves its objective. The total time taken to complete the multi-node broadcast is $k^n - 1$ which is optimal. Note that this algorithm requires only $k^n$ links for data transmission whereas the dimensional multi-node broadcasting of Section 6.4 requires the whole $nk^n$ links of the $k$-ary $n$-cube to complete the process.

### 7.3 Single-node Scattering for one-port I/O Model

Let the source processor be $s_0$. Our single-node scatter algorithm utilizes the Hamiltonian cycle of the $k$-ary $n$-cube for data transmission and is as follows:
each processor generates the $k$-ary Gray codes of dimension $n$ as detailed in Section 3.2;
let the resulting Hamiltonian cycle be $s_0, s_1, \ldots, s_{k^n - 1}$
and let the packet intended for processor $s_i$ be denoted $p_i$;
for $i := k^n - 1$ downto 1 do
  processor $s_0$ sends packet $p_i$ to processor $s_i$ and
  processor $s_j$, for $j \neq 0$, having retained a copy of
  any packet just received, sends this packet on to
  processor $s_{j+1}$;
endfor

Clearly, the time taken by this algorithm is $k^n - 1$ which is optimal.

7.4 Applications to Fault Tolerance

The multi-node broadcast and the single-node scatter algorithms for one-port I/O model of the previous sections can be implemented on a $k$-ary $n$-cube $Q_k^n$ with faulty links. For example, to implement the multi-node broadcast algorithm for one-port I/O model on a $Q_k^n$ with at most $\lambda$ faulty links, it is enough to construct a Hamiltonian cycle in this faulty $Q_k^n$ and perform the multi-node broadcast algorithm presented in Section 7.2 (i.e., Theorem 5.1 shows the existence of a Hamiltonian cycle in a $Q_k^n$ with $\lambda = 4n - 5$ faulty links).

As any single-node scatter algorithm for one-port I/O $k$-ary $n$-cube model requires every processor, except the source, receive one packet on at most one incident link at any one time, it is necessary for the degree of each node in the $Q_k^n$ to be at least 1. Assume that a Hamiltonian cycle can be constructed in a $Q_k^n$ with at most $\lambda$ faulty links. Then the
single-node scatter algorithm of Section 7.3 can be implemented in a $Q^k_n$ with at most $\lambda + 1$ faulty links as follows.

**Corollary 7.1** If a Hamiltonian cycle can be constructed in a $Q^k_n$, where $k \geq 3$ and $n \geq 2$, with at most $\lambda$ faulty links, then a Hamiltonian path can be constructed in a $Q^k_n$ with at most $\lambda + 1$ faulty links.

**Proof** Let $Q^k_n$, where $k \geq 3$ and $n \geq 2$, contain at most $\lambda + 1$ faulty links and let $f_e$ be any faulty link in this $Q^k_n$. Then by making $f_e$ healthy, the resulting $Q^k_n$ will contain at most $\lambda$ faulty links and a Hamiltonian cycle $HC$ can be constructed in this $Q^k_n$.

If $f_e \notin HC$ then keep $HC$ unchanged. Otherwise, remove $f_e$ from $HC$. In both cases the resulting $HC$ contains a Hamiltonian path $HP$ where every link in $HP$ is healthy.

Let the nodes of $HP$ of the above result be denoted $s_l(L), s_l(L-1), \ldots, s_l(1), s(0), s_r(1), s_r(2), \ldots, s_r(R)$ where $s(0)$ is the source processor and $s_l(i)$ (resp. $s_r(i)$) is a node of distance $i$ on the left (resp. right) of $s(0)$. The single-node scatter algorithm is as follows:

let the packet intended for processor $s_l(i)$ (resp. $s_r(i)$) be denoted $p_l(i)$ (resp. $p_r(i)$);

for $i := L$ downto 1 do

processor $s(0)$ sends packet $p_l(i)$ to processor $s_l(1)$

and processor $s_l(j)$, for $j \neq 0$, having retained

a copy of any packet just received, sends this packet

on to processor $s_l(j+1)$;

endfor

for $i := R$ downto 1 do

processor $s(0)$ sends packet $p_r(i)$ to processor $s_r(1)$
and processor $s_r(j)$, for $j \neq 0$, having retained
a copy of any packet just received, sends this packet
on to processor $s_r(j+1)$;
endfor

The algorithm clearly achieves its objective. The time taken by this
algorithm is $L + R = k^n - 1$ which is optimal.

As a result, it should be clear that the multi-node broadcast algorithm
can be implemented in a $k$-ary $n$-cube with faulty nodes whenever a
cycle containing all the healthy nodes can be constructed. Also, the
single-node scatter algorithm can be implemented in such a faulty $k$-ary
$n$-cube whenever a linear array containing all the healthy nodes can be
constructed.

7.5 Applications to multi-port I/O Model

In this section, we develop multi-node broadcast and single-node scat­
tter communication algorithms for multi-port $k$-ary $n$-cube model. Our
algorithms utilize the incident links of each source processor by decom­
posing the $k$-ary $n$-cube $Q_n^k$ into $n$ link-disjoint Hamiltonian cycles and
performing the algorithms of Section 7.2 and 7.3 on each Hamiltonian
cycle.

Let $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ be two network topologies, where
$V_1$ and $V_2$ are the sets of nodes and $E_1$ and $E_2$ are the sets of links.

**Definition 7.2** Given two network topologies $G_1 = (V_1, E_1)$ and $G_2 =
(V_2, E_2)$, define the cross product of $G_1$ and $G_2$ denoted by $G_1 \otimes G_2$, as
the network topology $G = (V, E)$, where

$$V = \{(x, y) | x \in V_1, y \in V_2\}$$
\[ E = \{(x_1, y_1), (x_2, y_2) \mid (x_1, x_2) \in E_1 \text{ and } y_1 = y_2, \]
or \((x_1 = x_2 \text{ and } (y_1, y_2) \in E_2)\} \]

See [34] for properties of cross product of interconnection networks.

Alspach et al. [2, Corollary 3] showed that \( C_{i_1} \otimes C_{i_2} \otimes \ldots \otimes C_{i_n} \), where \( C_{i_i} \) is a cycle of length \( i_i \), has a Hamiltonian decomposition. Using this result, the following theorem follows.

**Theorem 7.3** \( Q_n^k \) can be decomposed into \( n \) Hamiltonian cycles.

**Proof** Let \( C_k \) be a cycle of length \( k \), and each node in \( C_k \) is labelled with a radix \( k \) number \( 0, 1, \ldots, k - 1 \). There is a link between nodes \( u \) and \( v \) iff \( D_L(u, v) = 1 \). Thus, a \( k \)-ary \( n \)-cube \( Q_n^k \) can be defined as a cross product of cycles as follows.

\[
Q_n^k = C_k \otimes C_k \otimes \ldots \otimes C_k = \otimes_{i=1}^{n} C_k.
\]

The result follows from [2, Corollary 3]. □

Given the node labels of the \( k \)-ary \( n \)-cube \( Q_n^k \), the problem of developing an efficient algorithm to find \( n \) link-disjoint Hamiltonian cycles (Gray codes) is open [18]. The following algorithm constructs 2 link-disjoint Hamiltonian cycles (Gray codes), \( HC_1 \) and \( HC_2 \), in a \( k \)-ary 2-cube \( Q_2^k \) for any \( k \geq 3 \) given the node labels of the \( k \)-ary 2-cube (note that the addition is modulo \( k \)):

\[
HC_1 := \phi;
\]
\[
HC_2 := \phi;
\]
for \( i := 0 \) to \( k - 1 \) do
    for \( j := 0 \) to \( k - 1 \) do
\[ HC_1 := HC_1 \cup \{((i,j),(i,j+1))\} \]
\[ HC_2 := HC_2 \cup \{((j,i),(j+1,i))\} \]
\text{endfor}
\text{endfor}

for \( i := 0 \) to \( k-2 \) do
\[ HC_1 := HC_1 \cup \{((i,i),(i+1,i)),((i+1,i),(i+1,i+1))\} \]
\[ \{((i,i),(i,i+1)),((i+1,i),(i+1,i+1))\} \]
\[ HC_2 := HC_2 \cup \{((i,i),(i,i+1)),((i+1,i),(i+1,i+1))\} \]
\[ \{((i,i),(i+1,i)),((i+1,i),(i+1,i+1))\} \]
\text{endfor}

**Lemma 7.4** The resulting \( HC_1 \) and \( HC_2 \) from the above algorithm are link-disjoint Hamiltonian cycles in \( Q_2^k \).

**Proof** \( HC_1 \) first contains the \( k \) disjoint cycles of dimension 1 each of length \( k \), namely \( C_0, C_1, \ldots, C_{k-1} \), and \( HC_2 \) contains the \( k \) disjoint cycles of dimension 2 each of length \( k \), namely \( C'_0, C'_1, \ldots, C'_{k-1} \). The algorithm then 'joins' \( C_0 \) to \( C_1 \), then \( C_1 \) to \( C_2 \), and so on to form \( HC_1 \), and 'joins' \( C'_0 \) to \( C'_1 \) and \( C'_1 \) to \( C'_2 \), and so on to form \( HC_2 \). The algorithm 'joins' \( C_i \) to \( C_{i+1} \) and \( C'_i \) to \( C'_{i+1} \) by performing the following steps:

- it selects 2 links from \( HC_1 \) and 2 links from \( HC_2 \) as follows: \( e_1 = ((i,i),(i,i+1)) \) of \( C_i \); \( e_2 = ((i+1,i),(i+1,i+1)) \) of \( C_{i+1} \); \( e'_1 = ((i,i),(i+1,i)) \) of \( C'_i \); and \( e'_2 = ((i,i+1),(i+1,i+1)) \) of \( C'_{i+1} \),
- it removes links \( e_1 \) and \( e_2 \) from \( HC_1 \) and adds them to \( HC_2 \), and
- it removes links \( e'_1 \) and \( e'_2 \) from \( HC_2 \) and adds them to \( HC_1 \).

In each 'joining' process, the algorithm ensures that the two links removed from \( HC_1 \) are added to \( HC_2 \), and the two links removed from \( HC_2 \)
are added to $HC_1$. Thus making $HC_1$ and $HC_2$ link-disjoint Hamiltonian cycles.

Fig. 7.1 illustrates the resulting 2 link-disjoint Hamiltonian cycles (Gray codes) when the above algorithm is applied to a $Q_4^4$.

![Diagram of link-disjoint Hamiltonian cycles]

<table>
<thead>
<tr>
<th>$HC_1$</th>
<th>$HC_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00 33</td>
<td>00 33</td>
</tr>
<tr>
<td>10 23</td>
<td>01 32</td>
</tr>
<tr>
<td>13 20</td>
<td>31 02</td>
</tr>
<tr>
<td>12 21</td>
<td>21 12</td>
</tr>
<tr>
<td>22 11</td>
<td>22 11</td>
</tr>
<tr>
<td>32 01</td>
<td>23 10</td>
</tr>
<tr>
<td>31 02</td>
<td>13 20</td>
</tr>
<tr>
<td>30 03</td>
<td>03 30</td>
</tr>
</tbody>
</table>

Figure 7.1: The 2 link-disjoint Hamiltonian cycles in a $Q_4^4$.

Having constructed the $n$ link-disjoint Hamiltonian cycles in the $k$-ary $n$-cube $Q_n^k$, we show in the following how we can exploit these Hamiltonian cycles to develop efficient multi-node broadcast and single-node scatter communication algorithms for the multi-port I/O model.
7.5.1 Multi-node Broadcasting for multi-port I/O Model

Let the size of the message each processor intends to broadcast be $M$ packets where $M \geq n$. Each processor can send $n$ packets over $n$ incident links and can simultaneously receive $n$ packets over the other $n$ incident links. Therefore, the lower bound for any multi-node broadcast algorithm for the multi-port I/O model is

$$T_{MNB} = \left\lceil \frac{(k^n - 1)}{n} \right\rceil M.$$ 

Our multi-node broadcast algorithm for multi-port I/O model divides each message into $n$ parts and distributes these parts over the $n$ link-disjoint Hamiltonian cycles. The algorithm then performs in parallel the multi-node broadcast for one-port I/O of Section 7.2 on each Hamiltonian cycle. The algorithm achieves 100% utilization of the network links and is as follows:

1. Each processor divides its broadcast message into $n$ parts, namely $P_1, P_2, \ldots, P_n$, each of size at most $\lceil M/n \rceil$ packets, and let the $j$th packet of $P_i$ be denoted $p_{(i,j)}$.
2. Each processor generates the $n$ link-disjoint Hamiltonian cycles of the $Q_n^k$, namely $HC_1, HC_2, \ldots, HC_n$, as detailed above.
3. For $j := 1$ to $\lceil M/n \rceil$ do
   a. Each processor broadcasts packet $p_{(i,j)}$ along $HC_i$,
   b. For $i = 1, 2, \ldots, n$, using the multi-node broadcast algorithm of Section 7.2;
4. Endfor
The total time taken to complete the multi-node broadcast algorithm of Section 7.2 is $k^n - 1$. Therefore, the total time taken to complete our algorithm is

$$(k^n - 1) \left\lceil \frac{M}{n} \right\rceil \simeq T_{MNB}.$$  

7.5.2 Single-node Scattering for multi-port I/O Model

Let the size of each message the source processor $s(0)$ intends to send be $M$ packets where $M \geq n$. The source processor $s(0)$ can send different packets simultaneously over the $2n$ incident links in each unit of time. Therefore, the lower bound for any single-node scatter algorithm for multi-port I/O model is

$$T_{SNS} = \left\lceil \frac{(k^n - 1)}{2n} \right\rceil M.$$  

We first develop single-node scatter algorithm for 2-port I/O $k$-ary $n$-cube model where the size of each message the source processor $s(0)$ wishes to send is one packet. Let $\beta_l = \lfloor k^n/2 \rfloor$, and let $\beta_r = \lceil k^n/2 \rceil$ if $k$ is odd and $\beta_r = \lfloor k^n/2 \rfloor - 1$ if $k$ is even. Then our single-node scatter algorithm for the 2-port I/O model is as follows:

1. each processor generates the $k$-ary Gray codes of dimension $n$ as detailed in Section 3.2;
2. let the resulting Hamiltonian cycle be $s_l(\beta_l), s_l(\beta_l - 1), \ldots, s_l(1), s(0), s_r(1), \ldots, s_r(\beta_r)$ and let the packet intended for processor $s_l(i)$ (resp. $s_r(i)$) be denoted $p_l(i)$ (resp. $p_r(i)$);
3. do in parallel:
   - for $i := \beta_l$ downto 1 do
     - processor $s(0)$ sends packet $p_l(i)$ to processor $s_l(1)$
and processor $s_i(j)$, for $j \neq 0$, having retained a copy of any packet just received, sends this packet on to processor $s_i(j + 1)$;

* for $i' := \beta_r$ downto 1 do
  processor $s(0)$ sends packet $p_r(i')$ to processor $s_r(1)$ and processor $s_r(j')$, for $j' \neq 0$, having retained a copy of any packet just received, sends this packet on to processor $s_r(j' + 1)$;

enddo

To show that this algorithm is optimal, note that $s_i(\beta_l)$ and $s_r(\beta_r)$ are adjacent. There are two cases to consider.

Case (i) $k$ is odd. The time taken by the algorithm is

$$\beta_l = \beta_r = \left[k^n/2\right] = (k^n - 1)/2 = [(k^n - 1)/2].$$

Case (ii) $k$ is even. The time taken by the algorithm is

$$\max(\beta_l, \beta_r) = \beta_l = \left[k^n/2\right] = k^n/2 = [(k^n - 1)/2].$$

The time taken by this algorithm is $[(k^n - 1)/2]$ which is optimal since the source processor must send out the $k^n - 1$ different packets over two incident links in each unit of time.

Consider now the single-node scatter algorithm for multi-port I/O $k$-ary $n$-cube model, where the size of each message the source processor $s(0)$ wishes to send is $M$ packets where $M \geq n$. The algorithm divides each message into $n$ parts and distributes these parts over the $n$ link-disjoint Hamiltonian cycles. The algorithm then performs in parallel the single-node scatter for the 2-port I/O on each Hamiltonian cycle. The algorithm is as follows:
processor s(0) divides each message it intends to send into n parts, namely $P_1, P_2, \ldots, P_n$ each of size at most $[M/n]$ packets;

let the ith part of the jth message be denoted $p_{(i,j)}$, $1 \leq i \leq n, 1 \leq j \leq k^n - 1$;

let $S_i = \bigcup_{j=1}^{k^n-1} p_{(i,j)}$, for $i = 1, 2, \ldots, n$;

let $P_{(i,d)}$ be the set containing the dth packet of each element in $S_i$;

each processor generates the n link-disjoint Hamiltonian cycles of the $Q_n^k$, namely $HC_1, HC_2, \ldots, HC_n$ as detailed above;

for $d := 1$ to $[M/n]$ do

processor s(0) sends every packet in $P_{(i,d)}$
along $HC_i$, for $i = 1, 2, \ldots, n$, using the single-node scatter algorithm for the 2-port I/O model as described above;

endfor

The algorithm clearly achieves its objectives. The total time taken to complete the single-node scatter algorithm for the 2-port I/O model is $[(k^n - 1)/2]$. Therefore, the total time taken to complete our single-node scatter algorithm for the multi-port I/O model is

$$\left[\frac{k^n - 1}{2}\right] \left[\frac{M}{n}\right] \approx T_{SNS}.$$
Chapter 8

Conclusions

8.1 Summary

The major objective of this thesis is to examine the capability of the $k$-ary $n$-cube interconnection network $Q^k_n$, for $k \geq 3$ and $n \geq 2$, of simulating other popular networks and to develop schemes for some common communication algorithms for this network. We have shown in Chapter 2 that the $k$-ary $n$-cube network captures the advantages of the mesh network and those of the binary hypercube: the $k$-ary $n$-cube is Hamiltonian; it can be constructed recursively from low dimensional cubes; the degree of each node is $2n$; the total number of links is $nk^n$; its diameter is $[k/2]n$; and it is both node- and link-symmetric. We have also shown that the $k$-ary $n$-cube $Q^k_n$ contains $k^{n-1}$ node-disjoint cycles each of length $k$ in each dimension and we have stated the $2n$ node-disjoint parallel paths between any two nodes. The $k$-ary $n$-cube $Q^k_n$ has a smaller degree than that of its equivalent hypercube (the one with at least as many nodes) and it has a smaller diameter than its equivalent mesh of processors. It can efficiently simulate other network topologies such as cycles, meshes,
tori, trees, and hypercubes.

In Chapter 3, we have given a recursive structure of $k$-ary Gray codes and have exactly classified when a cycle of length $m$, where $3 \leq m \leq k^n$, can be embedded in $Q^k_n$. Our analysis yields an algorithm for generating a cycle of length $m$ in $Q^k_n$, when one exists, thus answering a question posed in [17].

In Chapter 4, we have described a technique for embedding a large cycle in a faulty $k$-ary $n$-cube $Q^k_n$. In particular, we have shown that in a $k$-ary $n$-cube $Q^k_n$, where $k \geq 3$ and $n \geq 2$, with $\nu$ faulty nodes and $\lambda$ faulty links where $\nu + \lambda \leq n$, there exists a cycle of length at least $k^n - \nu \omega$, where $\omega = 1$ if $k$ is odd and $\omega = 2$ if $k$ is even. Also, we have extended our main result to obtain embeddings of meshes and tori in such a faulty $k$-ary $n$-cube.

In Chapter 5, we have developed a technique for embedding a Hamiltonian cycle in a $k$-ary $n$-cube with at most $4n - 5$ faulty links where every node is incident with at least two healthy links. Our result is optimal as there exist $k$-ary $n$-cubes with $4n - 4$ faults (and where every node is incident with at least two healthy links) not containing a Hamiltonian cycle. We have shown in this chapter that the same technique can be easily applied to the hypercube. We have also shown that the general problem of deciding whether a faulty $k$-ary $n$-cube contains a Hamiltonian cycle is NP-complete, for all (fixed) $k \geq 3$.

In Chapter 6, we have developed some efficient communication algorithms for the $k$-ary $n$-cube network. In particular, we have developed and analysed routing, single-node broadcasting, multi-node broadcasting, single-node scattering, and total exchange. All our algorithms, except single-node broadcasting when $k$ is odd, are optimal for the one-port
I/O $k$-ary $n$-cube model. When $k$ is odd, the single-node broadcasting is optimal for the 2-port I/O model.

In Chapter 7, we have shown how Hamiltonian cycles of the $k$-ary $n$-cube network can be exploited to develop fault-tolerant multi-node broadcast and single-node scatter communication algorithms for the one-port I/O $k$-ary $n$-cube model. We have also shown in this chapter how the link-disjoint Hamiltonian cycles of the $k$-ary $n$-cube can be used to develop multi-node broadcast and single-node scatter algorithms for machines that support multi-port I/O model.

### 8.2 Future Research

There has been much less work done in embeddings of popular interconnection networks into $k$-ary $n$-cubes. Our principal open problem is to describe embedding of meshes and hypercubes of arbitrary dimensions into their optimum $k$-ary $n$-cubes.

We have shown in Section 2.4 that a binary tree of height $h$ (where the root is at height 0) can be embedded into a $Q^k_h$, for $k \geq 3$ and $h \geq 2$. However, the number of nodes in a tree of height $h$ is $O(2^h)$ and the number of nodes in a $Q^k_h$ is $O(k^h)$. We plan to explore ways for a more efficient embedding of a tree into a $Q^k_h$.

Whilst we have established in Chapter 4 and 5 the existence of a long cycle in a faulty $k$-ary $n$-cube $Q^k_n$, we have as yet to develop efficient algorithms for generating these long cycles. In fact, some work has been done on algorithms to find long cycles in a $k$-ary $n$-cube with faulty links [86] and in a hypercube with faulty nodes [23] and faulty links [59] but even this scenario has not been as thoroughly researched as it might have been.
In Chapter 6, we have exhibited efficient algorithms for the problems of routing, single-node broadcasting, multi-node broadcasting, single-node scattering and total exchanging when we assume that there is one-port I/O communication using store-and-forward routing. We would like to consider these problems for multi-port I/O communication using wormhole routing. Also, we would like to know how we can cope with these problems in faulty $k$-ary $n$-cubes.

We have developed in Chapter 7 an efficient algorithm for generating two link-disjoint Hamiltonian cycles in the two dimensional $k$-ary $n$-cube $Q^k_n$ and have shown how these cycles can be used to develop efficient algorithms for multi-node broadcast and single-node scatter when multi-port I/O communication is allowed. However, the problem of developing an efficient algorithm to generate $n$ link-disjoint Hamiltonian cycles in the $n$ dimensional $k$-ary $n$-cube $Q^k_n$ is yet to be considered.
Bibliography


