Chapter 2

Thesis Background

This chapter introduces dependable distributed system concepts, basic terminologies about hypercube interconnection networks, fault tolerance and concepts of all types of routing (i.e., unicast, multicast and broadcast). Fault-tolerant routing in hypercube interconnection networks needed to frame the thesis. First, Section 2.1 describes the taxonomy of dependable computing systems. Section 2.2 outlines some basic concepts of fault tolerance and specifically focuses on network fault tolerance. Then, in Section 2.3, we present some basic description of interconnection networks. Section 2.4 describes properties of hypercube network. The last Section 2.5 explains fault tolerance methods for hypercube topology in various modes.

2.1 Dependability in Distributed Systems

Computing systems are described by five key properties: functionality, usability, performance, cost, and dependability. Dependability is defined as a computing system property which integrates availability, reliability, safety, confidentiality, maintainability. It delivers the service that can justifiably be trusted [7]. A taxonomy of the concepts of dependability is shown in Figure 2.1 which consists three parts: the attributes, the means, and the threats.

2.1.1 Distributed Systems Terminology

Availability is the ability of a computing system that can be able to deliver the correct service at any instance of time within a given time interval. *Reliability* is the ability of

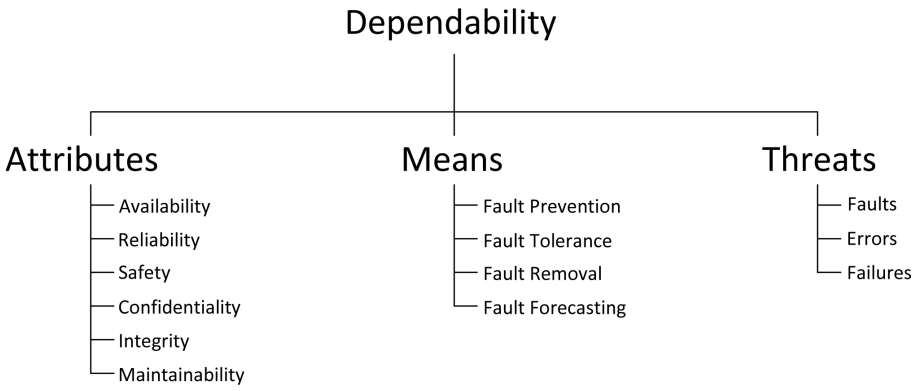

Figure 2.1: The dependability tree.

a computing system to perform failure-free operation under given conditions for a given time interval. *Safety* of a computing system deals with the avoidance of catastrophic consequences on the user or its environment. The absence of unauthorized disclosure of information leads to *confidentiality*. The absence of improper alterations of information or system state leads to *integrity*. *Maintainability* refers to how easily a failed system can be repaired and modified [77]. Associating three attributes availability and integrity together with confidentiality, with respect to authorized actions, leads to security.

The development of a dependable computing system required the combined utilization of a set of four methods. These are–

1. Fault *prevention* is the avoidance of introduction of faults.

2. Fault *tolerance* provides correct services in the presence of faults.

3. Fault *removal* minimises the number of faults.

4. Fault *forecasting* estimates the presence, the future incidence, and the likely consequences of faults.

Dependability is the ability to avoid failures that are more frequent and more severe than is acceptable. Actually, *Dependability* is a composition of the concepts of availability and reliability and it is the quality of service (QoS) provided by a computing system.

2.1.2 Dependable High Performance Computing Systems

Future High Performance Computing (HPC) systems will require the concurrent use and control of millions of processing elements, storage devices and networking. Since HPC systems continue to increase in scale, the key factor of success of massively parallel computers will highly depend on the ability to provide reliability and availability at Petascale/Exascale. The Message Passing Interface (MPI) is the de facto message passing library standard in HPC systems. MPI primarily addresses the message-passing parallel programming model. It provides an inflexible fault model in which a process fault enforce failures to all processes within a communicator. A dynamic research area fault-tolerant MPI programming has been adopted by the HPC group.

The current HPC systems uses checkpointing and rollback for fault tolerance. The problem is when level of error rate increases then these techniques becomes less efficient [19]. Today, the HPC systems is used for large numerical problems such as huge matrices with floating point numbers, but the requirement of the HPC systems is also essential in the field of *symbolic computing* in future. These types of problems always face the issue of long run time required by HPC systems. So, fault tolerance also needs to involve in large scale symbolic computation.

2.2 Fault Tolerance

Various factors may affect the performance of a computing system. So, the evaluation of faults and fault tolerance is highly essential. The most applicable terms in fault tolerance are availability, reliability and dependability. We have already provided their basic definitions in previous section. The theoretical concepts of these factors are discussed in next subsection.

2.2.1 Fault Tolerance Terminology

Before discussing about fault tolerance, understanding the differences among faults, error and failure is also important since occurrence of faults may degrade the system's performance. A fault is a notion of defect or bug within the system, while error is a deviation from the desired service of the system. The presence of faults does not imply that there is error in the system. A failure occurs when the behaviour of the system

can not longer perform as required by its specifications. Thus, a fault causes an error, which leads to a failure. The relations among them is shown in Figure 2.2

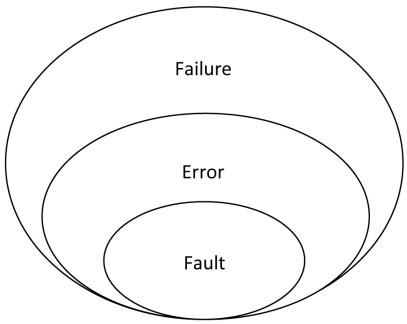

Figure 2.2: The relationship among fault, error and failure.

If a computing system is made of interdependent components, then occurrence of fault in a single component leads to an error and entire system may cause a failure. If a computing system is made of independent components, then occurrence of fault in a single component will cause a failure of this particular component only and remaining components of the computing system will be operational with some performance degradation. In this case, faulty component can be replaced.

Fault tolerance is the ability of the computing system to avoid failures and provide the service. Basically, fault tolerance has two attributes:– increase system availability and increase system reliability. Availability is the ability of a process to be in a state to perform a required function at a given instant of time or at any instant of time within a given time interval, assuming that the external resources, if required, are provided. Reliability is the ability of a process to perform a required function under given conditions for a given time interval that a system can run without failure.

Failures can be divided into two categories with respect to interconnection networks– link failures (failures of communication channels) and node failures (failures of processing element or its associated router). Failure of a node in a network causes effect on incident edges adjacent with that node. The concepts of node and link failures are shown in Figure 2.3. Fault tolerance can be defined as the ability of the system to provide a service for avoiding the failure, even in the presence of errors generated by a fault. The fault tolerance concept may determine the system performance by increasing reliability and availability of the system.

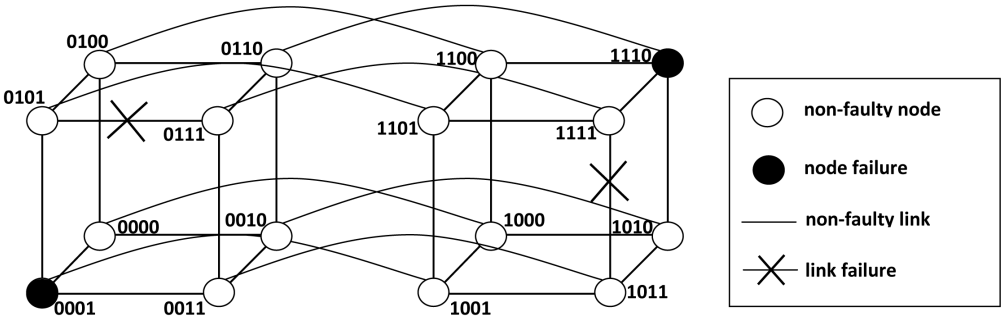

Figure 2.3: Network failure terminology.

Failure occurrence can result in substantial physical damage, economic losses, or threats to human life in a vital systems. The failure in a mission-critical system may result in the failure of some goal-oriented activity, such as a navigational system for aircraft. A business-critical system failure may result in very high costs for the business using that system, such as a computerised accounting system.

Faults may be transient and permanent depending on their duration along the time. A transient fault vanishes after a short period of time, whereas permanent faults do not disappear over the time. Two fault models can be considered for handling with permanent faults in a system and they are static model and dynamic model. In the static model, when a fault is noticed, stopped all running processes in the system and a process is set up for recovery from faults. This process computes the alternative paths for avoiding failed component. After this, the network is resumed. The static fault model requires checkpointing techniques, which allows processes to resume their work from a previous known state.

In the dynamic model, when a fault is noticed, there is no need to stop applications and network traffic and carry out the system with appropriate action to handle the faulty component. For instance, after detecting a faulty component in the system, the source node select an alternative path that does not use any faulty component. The dynamic fault model does not use any checkpointing technique to handle the faults and this model kept the system in working mode without stopping the network traffic.

In this thesis, we consider that all faults are permanent and use only dynamic fault model to handle them.

2.2.2 Fault-Tolerant Routing

A lot of studies on fault tolerant routings involving the surviving route in a network, where the diameter of the surviving network is a measure of the worst case time to complete a broadcast. Dolev et al. [41] studied the effects of faulty nodes and edges on the diameter of the surviving route network. Broder et al. [18] applied the concept to the product graph, a Cartesian product of component graphs. They derived the fault tolerant properties of the product graphs from the analysis of the surviving route networks on the component graphs. Peleg and Simons [97] further studied a group of graphs such that the diameter of the surviving graph is bounded by a constant. They developed a routing method called the kernel construction, which can be used to construct disjoint paths between two given nodes. Based on the concept of surviving route graph, Marko and Vaccaro [40] studied the fault tolerance of star and hypercube networks. In addition, Rescigno [105] applied the kernel construction approach to study randomized parallel routing in star networks. Rescigno's algorithm is randomized, thus, it does not always guarantee the maximum number of node-disjoint paths.

Also, there are related studies that utilize node or edge-disjoint paths on other fault-prone communication networks such as optical and mobile networks. In such networks, connectivity is related to the concept of network survivability that deals with a mechanism to protect resources against failures. A common approach to recover from failures is to provide an alternate path. In optical routing, link failures are common. Choi et al. [33] utilized edge-disjoint paths to construct backup paths for failed edges. In mobile networks, multiple-path methods have been studied for designing routing algorithms to deliver messages with high success rates and low flooding rates. Stojmenovic and Lin [113] showed that desirable success rates and flooding rates can be achieved by using c disjoint paths, where c is a small constant. Other routing schemes that utilize multiple paths have been proposed in [30, 60, 80, 125, 131]. Specifically, Goyal and Caffery [50] suggest an approach that builds upon connectivity concepts in graph theory.

Designing fault-tolerant routing algorithms is a challenging issue, when dealing with faults in a computing system. Faults degrade the system's performance, since it is a abnormal behaviours. A natural solution to the problem of faults in interconnection

networks is adaptive routing algorithms because they are able to dynamically use alternative paths to avoid faults. Most of the routing algorithms are based on fault tolerance model, the attributes of the model are: *failure type* (link or node) and *failure mode* (static or dynamic). Complexity and power of fault-tolerant routing algorithms are closely related and dependent on failure mode. So, it is the most important attributes for designing the fault-tolerant routing algorithms.

In static failure mode, all faults are static, permanent, and known in advance, so designing of routing algorithms could be simple. This failure mode still have a major drawback because static routing systems can not respond to network changes and for this reason they are not considered for today's hardware technology. Static routing is also known as non-adaptive routing. In dynamic failure mode, failures may occur at any time and position during system operation. In this mode, the routing algorithms do not need to know the position of faults in advance. Dynamic routing is also known as adaptive routing. Dynamic routing is most popular in comparison of static routing because it aids in avoiding congestion and improves performance of the network.

The advantages of static routing are as follows:–

- The routes are always fixed and hence the routing overhead is minimum.
- The routing is dependent on network topology, i.e., static in nature.
- Routing is same for datagram and virtual circuit type of services.

The major disadvantages of static routing are as follows:–

- Lack of flexibility.
- The system is not robust in case of link failure or node failure and can not recover from failures.
- Congestion may occur on a particular route.

So, The popularity of dynamic routing is mainly due to the above reasons.

Actually there are different types of interconnection networks and every interconnection networks have different types of combinations of network. Designing fault-tolerant routing algorithms for these interconnection networks is a complex problem. Many studies have been conducted to this problem through fault-tolerant routing algorithms

for meshes, rings, torus, hypercubes. Interconnection networks generates random faults which increases the complexity of dynamic fault tolerance. So we need a fault-tolerant routing algorithm which must be able to circumvent faults. Each time the routing algorithm changes a dimension or direction to avoid failures.

2.3 Interconnection Networks

The objective of analysing parallel architectures and their combinational properties is to find better topologies for massively parallel computing. Ideally a topology should have the following features:

- Communication,
- Low hardware cost,
- Capabilities for efficient applications,
- Fault tolerance, and
- Extensibility.

The major criterion for processor organization is diameter, which is the largest distance between two nodes in a graph. So, it can be used to measure the maximum communication delay. Degree, links, cut-width can be used to measure the hardware cost. Bisection width is the minimum number of edges that must be removed in order to divide the graph into two halves. So, it can be used to measure the capabilities for efficient applications. For the connectivity n-wide diameter, fault diameter can be used to measure the capabilities of fault tolerance. A computer is considered properly extensible if its infrastructure need not be changed when the computer's capacity is increased. Extensibility will not only result in lower direct costs, but design costs of a computer will also be lower.

Currently, interconnection networks play an important role for computing system's performance. Massively Parallel Processing (MPP) systems such as Cray Titan, IBM Blue Gene/P, Fujitsu K Computer and many more supercomputers use high-speed interconnection networks. The interconnection network is responsible for reliable and quick communication among the processing elements (PEs) in any parallel computer. It

consists of a series of nodes and links that interact with each other for communications. The interconnection network is a requirement of any parallel computer because parallel system shows high performance by providing reliable and quick communication over the networks. Lots of different interconnection networks for parallel computers have been proposed [67].

2.3.1 Network Topologies

An important feature of an interconnection network is its topology. A topology can be defined as the arrangement of nodes and channels in the interconnection network and their pattern through links. As defined by Duato et al. [42], network topologies can be divided into three different classes: shared medium networks, direct networks, and indirect networks.

A. *Shared medium networks*: In this network, all communicating devices share the communication medium. These kind of networks were used in the first parallel computers but failed very soon due to limited bandwidth, poor performance and scalability problems. Buses are commonly used network within the circumstance of shared medium networks. The examples of shared medium networks are local area networks (LAN) such as Ethernet, Fast Ethernet or Fiber Distributed Data Interface (FDDI).

B. *Direct networks*: Every node includes a switch/router in the network and each nodes directly connected to other nodes, which handles message communication among each nodes. For this reason, it is also known as *router-based networks*. Each node has its own local memory, processor, and other supporting devices. Because direct networks consist links only, routing decisions have to be made in the nodes. In the parallel computer, normally a node is not directly connected to all other nodes, a message passing from a source node to a destination node may require various steps by intermediate nodes to reach its destination node. These steps are called *hops*.

Direct networks have been a very popular interconnection network and used for building parallel computers. Most direct networks have an orthogonal topology, where nodes can be arranged into a n-dimensional space. Every links can be arranged in a single dimension which must produce a displacement. Lots of different network topologies have been proposed and for detailed discussion of these topologies can be found in [47, 82]. Direct networks can be defined by a graph $G(N, C)$, where N represents

the vertices or the set of nodes of the graph and C represents the edges or the communication channels of the graph. The examples of direct network topologies are mesh, torus, ring, hypercube, etc. The classification of direct network topologies is shown in Figure 2.4.

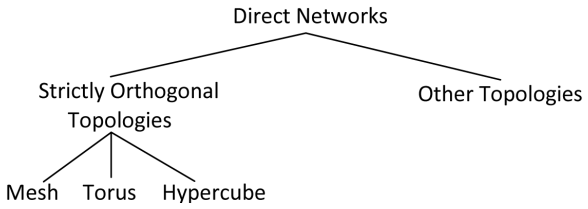

Figure 2.4: Classification of direct network topologies.

C. *Indirect networks*: In these networks, each processing node is connected to a network of switches through links. Each node has an adapter which is used for connecting switch and each switch has a number of ports. Each port has one input link and one output link. In indirect networks, each source provides exactly one path (*single path networks*) or multiple paths (*multipath networks*) to each destination. Lots of different network topologies have been proposed and for detailed discussion of these topologies can be found in [3, 111].

An indirect network can also be defined by a $G(N, C)$, where N represents the set of switches of the graph and C represents the set of unidirectional or bidirectional links between the switches in the graph. The examples of indirect network topologies are generalized-cube network, crossbars, multistage networks, irregular networks, etc.

The working of a router in a direct network and a switch in an indirect network is the similar. A direct network is equivalent to an indirect network in which every switch is connected to a single end node. A crossbar network has a single switch connecting all the end nodes. A multistage network corresponds to the case in which the switches are organized in several stages. So that switches belonging to intermediate stages are not connected with any end node. This mingled view allows us to apply the same strategies to route messages through the network. Some common topological functions of direct and indirect networks are as follow:

- *Degree*: The number of edges (links or channels) of each node (switch or router).
- *Diameter*: The maximum shortest distance between any two nodes.

- *Regularity*: All the switches/routers have the same degree.
- *Symmetry*: The network architecture appears identical from any switch/router perspective.
- *Connectivity*: The minimum number of links/switches requirement to disconnect the network.

2.3.2 Routing

Routing is a path determined by a message travelling from a source node to a destination node. There may be multiple paths between source-destination pair, the routing algorithms are responsible for them. Although the existing routing algorithms are rather large, Duato et al. [42] classified routing algorithms into four main classes– *number of destinations*, *routing decisions*, *adaptivity* and *implementation*. The taxonomy of routing algorithms is given in Figure 2.5. The summary of these classifications are given below—

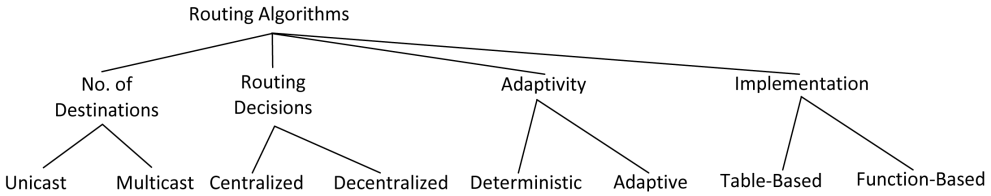

Figure 2.5: Taxonomy of routing algorithms.

Number of destinations– Basically two types of routing algorithms are available in the literature. These are *unicast* routing (when message have a single destination node), and *multicast* routing (there are one or more source nodes and a set of destination nodes).

Routing decisions– This is based on who will take the routing decisions and where they will be taken. *Centralized* controller or *decentralized* manner can be responsible for taking the decisions. In the centralized manner, there are two possibilities for taking the decisions–either by source-based routing or distributed routing. Distributed routing is based on by selecting source node, destination nodes and the path between them. It is also called a *multiphase* routing.

Adaptivity– This is the most important criterion for choosing a path between source-destination pair in a routing algorithm. *Deterministic* routing algorithms always select the similar path between a source-destination pair, even in the presence of multiple possible paths. The network load and the availability of network resources do not influence the routing of a message. *Oblivious* routing algorithms do not consider any information about the present state of the network for routing. *Adaptive* routing algorithms take the decision by adopting the condition of the network. For supporting adaptive routing in the network, it has multiple paths between a source and a destination.

Implementation– Routing algorithms can be implemented by routing tables or logic/ arithmetic routing functions, which store the path information for each source-destination pair. There are two types of routing algorithm–either deterministic or adaptive. The broad explanation of both routing approaches are given in the next subsection with example.

When a set of messages has a cyclic dependency on resources, then routing deadlock occurs. Due to the problem of deadlocks in direct networks, most routing algorithms have been proposed for direct networks to avoid deadlock situations. In this thesis, therefore we focus on routing algorithms for direct networks.

2.3.2.1 Deterministic Routing

Dimension-Order Routing (DOR) is the most common deterministic routing scheme used in the direct networks. In DOR, a message traverses the network in proper sequence over an ordered set of dimensions of path. *XY* routing and *e-cube* routing are the two common examples of DOR.

Mesh networks use the XY routing algorithm in which a message always routes in the X direction first. If it reaches to destination node, then the message will be routed in the Y direction (or vice versa). This routing scheme gives deadlock-free message delivery because there is no possibility of cyclic dependences [38]. Let us suppose a mesh network shown in Figure 2.6, and assume XY routing (first dimension X, then dimension Y). Suppose the source node is 3 and destination node is 14. A message will be routed from source node 3 to destination node 14 through the intermediate nodes 2, 6 and 10 as shown in the figure. If any node and/or link is faulty in the network on that path (e.g., the node 6 and/or link between nodes 10 and 14 is faulty), then

the message will stop. An alternative path is available of the same length through nodes 7, 11 and 15, but this path can not be taken due to the XY routing algorithm approach. Therefore, XY routing limits the number of paths a message can take and thus increases the possibility of message blocking, but insures deadlock free environment in the network [92].

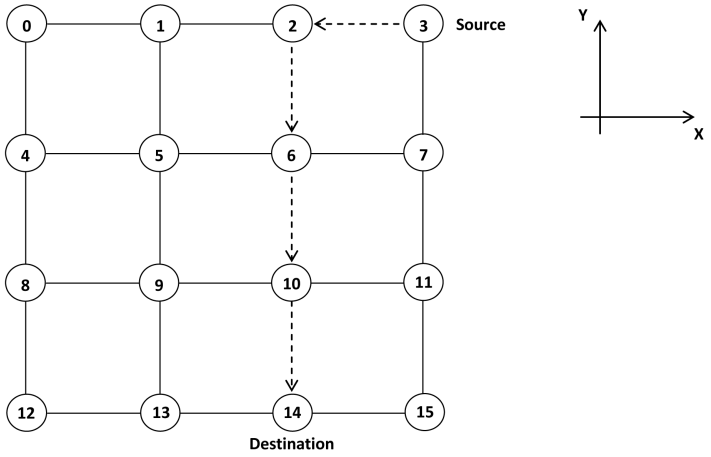

Figure 2.6: XY routing in a mesh with $N = 16$. Messages are routed in a dimension-ordered fashion.

Hypercube networks use the e-cube routing algorithm in which a message always routes in the same order (e.g., dim_0, then dim_1, then dim_2, then dim_3, ...). Consider the hypercube network shown in Figure 2.7. Suppose the transfer of a message from source node 4 to destination node 11 through the intermediate nodes 5, 7 and 3 is shown in a hypercube with $N = 16$ using the e-cube algorithm. If at any particular instant of time, the network resource is not available, then path will be blocked and the message has to wait. Although the alternative paths exists in the network (e.g., through intermediate nodes 12, 8 and 10). Nevertheless, by using e-cube algorithm in the network insures that cyclic dependency does not occur, so that deadlocks are avoided [37].

2.3.2.2 Adaptive Routing

A possible taxonomy of a adaptive routing protocols is shown in Figure 2.8 [48].

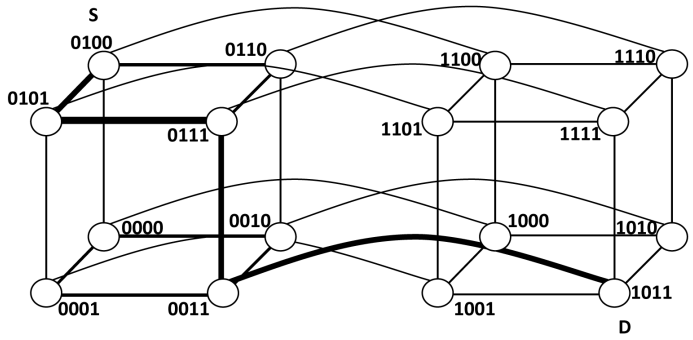

Figure 2.7: e-cube routing in a hypercube with $N = 16$. Messages are routed in a dimension-ordered fashion.

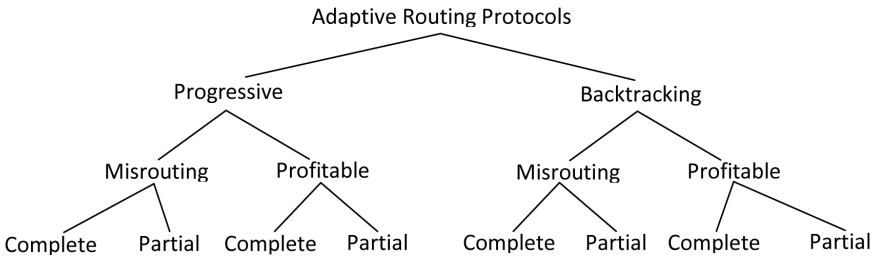

Figure 2.8: A taxonomy for adaptive routing protocols.

In a *progressive protocol*, routing decisions can not be reversed and have limited power to backtrack even message might end up being blocked. When a routing decision is prepared in a *backtracking protocol*, it can be reversed if they go to the blocking of a message. Therefore, if a message arrives at a blocked network resource (e.g., a node or a link), the message will track back an alternative path using history information. This method is useful for circuit switching or packet switching direct networks with bidirectional links between nodes and ensure that no path is explored more than once in a network.

A *profitable protocol* or *minimal routing protocol* will always select a network resource in such a way that the message should reach nearer to its destination. If a message finds a faulty node and/or link in the network, then it can choose the other node and/or link in such a way that results in a minimal length of the path. A *misrouting protocols* or *non-minimal routing protocols* gives the preference to misroute over message blocking. Hence, a message can choose longer path over minimum path, when routing a message from source node to destination node.

Completely adaptive routing protocol can use all paths in its class, while *partially adaptive* can use only subset of paths in its class to avoid deadlock situations.

2.4 Hypercube Basics

Since the network topology plays a vital role in system's performance, many possible options have been proposed like star network, hypercube network, cube connected cycle network, butterfly network, and many more networks. The most popular network topology is the hypercube network [39]. Hypercube has gain the popularity due to many of its elegance properties, including regularity, high symmetric (both node and link), small diameter, strong hierarchical structure, structural recursion, and maximal fault-tolerance.

Hypercube network is the example of direct networks, which consists a set of nodes and each node connected to one another by communication links. Each node has its own processor and local memory. Router is a common component which handles the message communication among nodes and each router direct associated with its neighbour routers. So, for this reason, it is also called router-based networks. Direct network is very much popular interconnection network for developing parallel computing systems. Duato et al. [42] stated that direct networks are the graph $G(V, E)$, where V is the set of vertices or the set of precessing nodes and E is the set of edges or the set of communication channels.

Definition 2.4.1. An n-dimensional hypercube H_n is defined recursively as follows:

$$H_n = \begin{cases} H_0 & : \ n = 0 \\ K_2 \times H_{n-1} & : \ n > 0 \end{cases} \tag{2.1}$$

Where H_0 is a trivial graph with one node, K_2 is a complete graph with two nodes, and $\times$ is the product operation of two graphs.

According to the equation 2.1, 2-dimensional cube is made of using two 1-dimensional cube, 3-dimensional cube is made of using two 2-dimensional cube, 4-dimensional cube is made of using two 3-dimensional cube, and so on.

Definition 2.4.2. The Hamming distance between two processing elements addresses $A_{n-1}A_{n-2}\ldots A_1A_0$ and $B_{n-1}B_{n-2}\ldots B_1B_0$ in hypercube H_n is defined as follows:

$$H(A, B) = \sum_{i=0}^{n-1} A_i \oplus B_i \tag{2.2}$$

Coordinate Sequence (CS) determines the path from the source node s to the destination node d. Actually, CS contains all the dimensions of source-destination pair. One of the important property of hypercube is the availability of many disjoint paths between any pair of nodes [85]. For any pair of nodes A, B of H_n, (A, B) is an edge in H_n iff node ids A and B differ at k bit positions, that is, $H(A, B) = k$; outcomes that there are k disjoint paths of length k and $n - k$ disjoint paths of length $k + 2$.

Example 2.4.1. The 4-dimensional hypercube shown in Figure 2.9. Suppose $s = 0101$ and $d = 1110$, then Hamming distance $H(0101, 1110) = 3$. Hence, there are 3 paths of length 3 and one path of length 5.

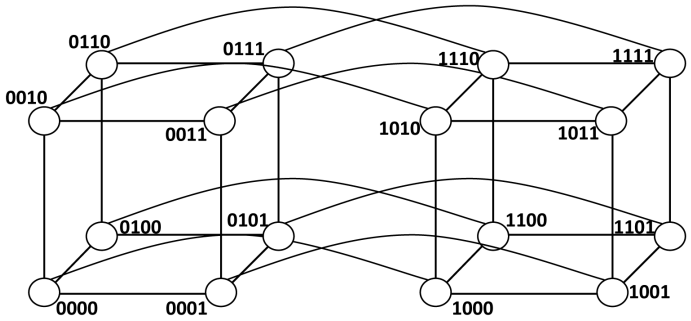

Figure 2.9: A 4-dimensional hypercube and its subcubes.

It has been shown that H_n can tolerate a large number of faulty nodes/links. If the faulty components (node/links) are less than n, then there is at least one path of length less than or equal to $k + 2$. The availability of redundant disjoint path in H_n makes it fault-tolerant interconnection network.

Definition 2.4.3. Node: can be either processor, memory, or switch.

Definition 2.4.4. Link: is the communication path between any two nodes.

Definition 2.4.5. Neighbour node: Two nodes are neighbours if there is a link between them. In the case of H_n, it is n.

Definition 2.4.6. Node Degree (d): is defined as the maximum number of the neighbours for a node. In the case of H_n, it is n.

Definition 2.4.7. Network Diameter (D): is defined as the maximum shortest path between any two nodes. The path length is measured by the number of links traversed. In the case of H_n, it is n.

Definition 2.4.8. Bisection width: is the minimum number of links that must be removed in order to divide the network into two equal halves. In the case of H_n, it is $N/2$ (where N is the number of nodes, and $N = 2^n$ or $n = log_2 N$).

Definition 2.4.9. Scalability: The increase in the complexity of the communication as more nodes are added.

Definition 2.4.10. Network Throughput: is defined as the total number of messages the network can handle per unit time.

Definition 2.4.11. Cost: is defined as the product of the degree (d) and the diameter (D) of a network. In the case of H_n, it is $d \times D$.

2.4.1 Subcube Reliability Computation in Hypercube Networks

The nodes of a graph correspond to processors and the edges correspond to communication links between processors. When an interconnection network is represented by such a graph, the fault tolerance is often measured by the node connectivity of the corresponding graph.

Let a graph can be denoted by $G(V, E)$, where $V(G)$ is the set of vertices and $E(G)$ is the set of edges in the network. Suppose $x, y \in V(G)$ and at least there is a single path $x - y$. Then the minimum number of vertices dividing x from y equals the maximum number of disjoint $x - y$ paths. So, the Menger's theorem [12] told that edge-disjoint and/or node-disjoint paths have been used to analyse the fault tolerance in different types of network topologies.

An n-dimensional hypercube or n-cube network consists of 2^n identical nodes that are labelled from 0 to $2^n - 1$. Any two nodes are connected by a link iff their labels differ by exactly one bit. Figure 2.9 shows the structure of a 4-cube network where two disjoint 3-cube, i.e., 0-subcube (consisting of eight nodes labelled with 0 in most significant bit position) and 1-subcube (consisting of eight nodes labelled with 1 in most significant bit position), are connected by eight cross links. Let H_n be the n-dimensional hypercube, then the relation

$$N = 2^n \tag{2.3}$$

Where N is the total number of nodes in H_n.

2.5 Fault Tolerance for Hypercube Interconnection Networks

There are two techniques available for fault tolerance hypercubes– they are classified as hardware vs software [5, 66]. Hardware techniques use redundant components (nodes and/or links) for reconfiguring a hypercube in order to replace faulty components. Software techniques use inherent redundancy, and symmetry for reconfiguring a hypercube [25, 51].

2.5.1 Hardware-based Fault Tolerance in Hypercube Networks

Hardware-based fault tolerance in hypercube networks uses the hardware switches to provide spare nodes utilization. In this approach, when node fails– replace it with a spare node.

2.5.2 Software-based Fault Tolerance in Hypercube Networks

2.5.2.1 Hypercube Free Dimension Technique

The free dimension technique can be used to accomplish fault tolerance in hypercubes [133]. If there is no set of faulty nodes throughout a dimension, it is called a free dimension in a hypercube H_n. In this technique, a hypercube is divided into a subcubes until each subcube contains not more than one faulty node. This division is done using free dimension. Figure 2.10 shows all the directions of the dimension in space.

In n-dimensional H_n, there exists at least one free dimension for any $f \leq n$ faulty nodes. Hence, H_n can be divided into $(n-f+1)$-dimensional subcubes such that each subcube generated by same set of dimensions comprises not more than one faulty node. As shown in Figure 2.11 with $f = 3$, it can be verified that H_4 can be divided into 3-dimensional subcubes and each subcube has not more than one faulty node. According to this findings, If $n > 3$ and $f \leq n$ then H_n is divided into two subcubes along some free dimensions. This technique degrade the hypercube performance about to 50% due to requirements of global knowledge of faulty nodes [100].

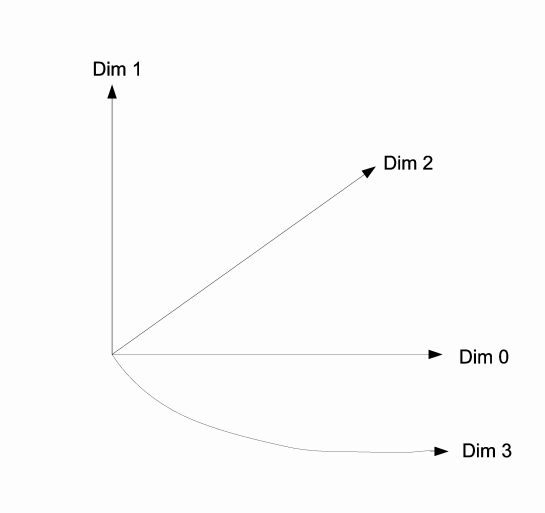

Figure 2.10: The structure of the dimension in the space.

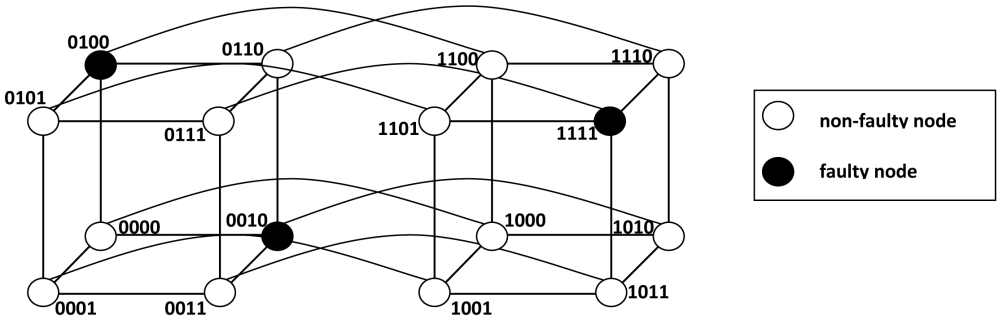

Figure 2.11: Hypercube division using free dimension.

2.5.2.2 Spanning Tree Approach

Spanning tree approach is used to configure a hypercube in order to avoid faulty nodes. This approach starts by constructing spanning tree (ST) of the hypercube. This approach requires two reconfiguration phase. In the first phase, all faulty nodes are deleted from the ST. In the second phase, new ST is constructed by reconnecting the children(s) of deleted node. Figure 2.12 shows the ST of Figure 2.9.

Since node (1010) is faulty, then it should be deleted from ST and its children be reconnected as shown in Figure 2.13. The reconfiguration phase defines new parents and new children in order to circumvent faulty node(s). In Figure 2.13 the new children of node (1100) become node (1110) and (0100) while the parent of node (0010) becomes

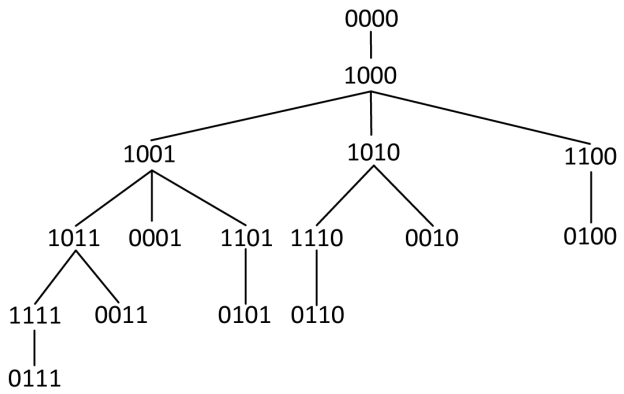

Figure 2.12: A Spanning tree (ST) of the hypercube: First phase.

node (0011) [119].

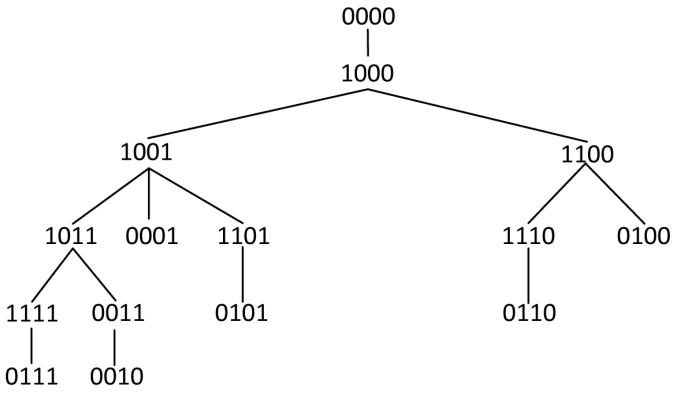

Figure 2.13: The modified ST under faulty node: Second reconfiguration phase.

The strategy of communicating nodes together is based on cubic graph. Each node has communication only with its neighbours and communicates with far nodes via its neighbours. The hypercube network has the number of nodes neighbours which is $log_2\ N$, N is the number of the nodes and the longest way between two nodes which is $log_2\ N$.

2.5.3 Fault-Tolerant Routing Techniques in Hypercube Networks

The interesting properties of hypercube topology is very much useful for parallel computing and therefore, motivated to researchers to solve classical problems e.g., image and signal processing [23], traveling salesman problem (TSP) [88], finding node-disjoint

shortest paths [73], node-to-set disjoint path routing [79]. In a parallel systems, when the size of the network grows, the probability of node or link faults occur more frequently. So, maintaining reliability of hypercube based parallel systems is highly desirable. Several fault-tolerant routing strategies for any interconnection network [104] has been proposed to resolve the faults when transmitting the data. There are many variants of the hypercube topology like crossed cubes [20], twisted cubes [21], folded cubes [45] and hierarchical cubes [125]. Hence, all these variants of hypercube also increase the importance of hypercube routing.

First time by Saad and Schultz [107] studied parallel routing on hypercube networks without faulty nodes. Madhavapeddy and Sudborough [85] developed an algorithm that constructs node-disjoint paths between disjoint source-destination pairs in the hypercube networks. Gu and Peng [57] also proposed an efficient algorithm for the pairwise disjoint paths between disjoint source-destination pairs. Latifi et al. [79] provided a simple algorithm that constructs disjoint paths between one node and a set of nodes. Krishnamoorthy and Krishnamurthy [71] considered the problem of determining the diameter of hypercube networks with faulty nodes. Under the constraint that each non-faulty node must have at least one non-faulty neighbour. Latifi [78] studied the node-disjoint paths in hypercube networks with faulty nodes. Latifi used the node-disjoint paths as the method to derive the diameter of the hypercube network with faulty nodes.

When a fault occur in a hypercube network, the *Coordinate Sequence* (CS) or *e-cube* routing algorithm may fail to route the message decently. So, the researchers have been introduced a number of adaptive algorithms to route the correct message in the presence of faulty node and/or link for a hypercube networks [4, 6, 8, 9, 26, 27, 32, 81, 100, 126]. Several authors dealt with fault-tolerant unicast routing [52, 54, 58, 123], Other authors dealt with fault-tolerant multicast routing [31, 74, 76, 83, 91, 110, 127, 129]. Some authors dealt with fault-tolerant broadcast routing [22, 96, 101, 124, 128].

2.5.3.1 Depth-First Search based Routing

Hamming distance is a technique to find the optimal path of equal length between the source and destination nodes. In this approach, each node to know the information of its own links. If the number of faulty components (nodes and/or links) is less than n, then this algorithm routes the message successfully. This type of algorithms are based

on *Coordinate Sequence* (CS) whose function is to send the message from a source node to a destination node by changing the dimension each time. In the first step, the protocol follow the *e-cube* pattern for routing in respect of smallest dimension. If smallest dimension is blocked, then node tries next higher dimension to route the message as defined by the CS. If there is no such dimension available in the CS, then there is a chance of misrouting [27].

Example 2.5.1. Consider the 4-dimensional cube shown in Figure 2.9. Since DFS algorithm can tolerate less than n faulty components, so we have consider faulty links. Faulty links are shown by cross on line in Figure 2.14. Suppose source node is $S = 0110$ and destination node is $D = 1001$. The CS of the message at the S is CS=$\{0, 1, 2, 3\}$. The message start at S and reached D as follow:

$$0110 \xrightarrow{0} 0111 \xrightarrow{1} 0101 \xrightarrow{3} 1101 \xrightarrow{0} 1100 \xrightarrow{1} 1110 \xrightarrow{2} 1010 \xrightarrow{0} 1011 \xrightarrow{1} 1001.$$

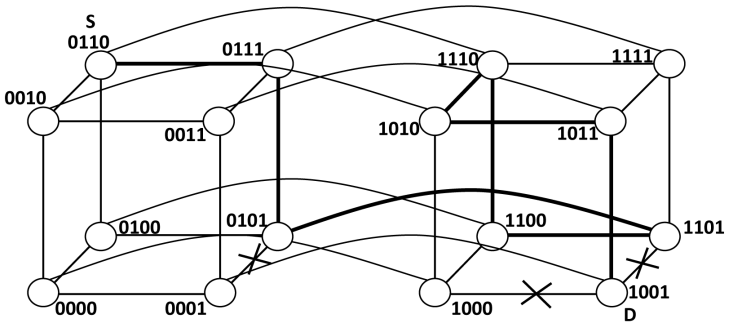

Figure 2.14: Routing in 4-cube using DFS.

There is situation of misrouting when message moving at node 1101 and 1010. The total path length is 8 as shown in the figure. In this way, we have seen that DFS algorithm can route the message successfully between any pair of non-faulty nodes when the number of faulty elements is less than n in H_n.

Since DFS algorithm does not consists global information about faulty components and increase the path length during message passing between any pair of non-faulty nodes. So, Sasama et al. [108] proposed a modified algorithm which provides the shortest path between any pair of non-faulty nodes and increases the probability of successful routing messages to the destination. This modified algorithm insure the shortest path routing in hypercube H_n having less faulty components with respect

to n (dimension of the hypercube). The improvement of this algorithm over DFS algorithm shown in the Figure 2.15. It route the messages from source node $S = 0110$ to destination node $D = 1001$ using shortest path and length of the path is 4.

$0110 \xrightarrow{0} 0111 \xrightarrow{2} 0011 \xrightarrow{3} 1011 \xrightarrow{1} 1001.$

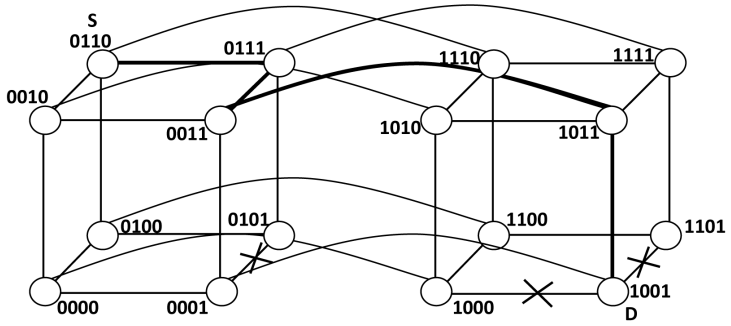

Figure 2.15: Shortest path routing in 4-cube.

2.5.3.2 Heuristic-based Routing

DFS algorithm has been improved by Abd-El-Barr et al. [2]. They provide a heuristic-based routing algorithm which uses five passes. The first phase is similar to the DFS algorithm. In this phase, if the message is not routed, then algorithm begins the second phase. If the time interval elapses, the source send the same message after specified time period. Heuristic algorithm requires the dimension for routing. In the second phase, they route the message through the lowest possible dimension. In the third phase, route the message in the highest possible dimension. In the fourth phase, misrouting of the message is possible in the highest possible dimension. In the last phase, routing is done only having all dimensions in the CS.

Table 2.1 demonstrates the routing results between DFS algorithm and heuristic-based routing algorithm. We have taken four different sources and four different destinations in faulty hypercube under link fault model. It is clear from the Table 2.1 that particularly selected source and destination, the DFS algorithm was unable to route the message while the heuristic algorithm was successful under the same faulty links conditions.

S	D	Faulty links	DFS routing	Heuristic routing
0	15	2–10, 8–10, 12–13	0–1–3–2–6	0–2–8–9–13–15
2	9	1–9, 9–11, 10–11	2–3–11–3–2–0–1	2–6–14–12–8–9
3	10	1–9, 3–11, 9–11	3–1–0–4–6	3–1–0–4–12–13–15–11–10
9	4	3–7, 5–13, 3–11	9–8–12–13–15–14–12–8	9–11–10–14–6–4

Table 2.1: Results of heuristic routing algorithm.

2.5.3.3 Node Safety based Routing

A node is unsafe state when either it has two or more faulty neighbour nodes or if it has three or more faulty or unsafe nearest neighbours. Figure 2.16 shows a 4-dimensional cube which identifies unsafe nodes. The main objective of introducing node safety is selecting feasible paths for routing message in the presence of faulty components in H_n. This technique is used to avoid routing messages through unsafe nodes.

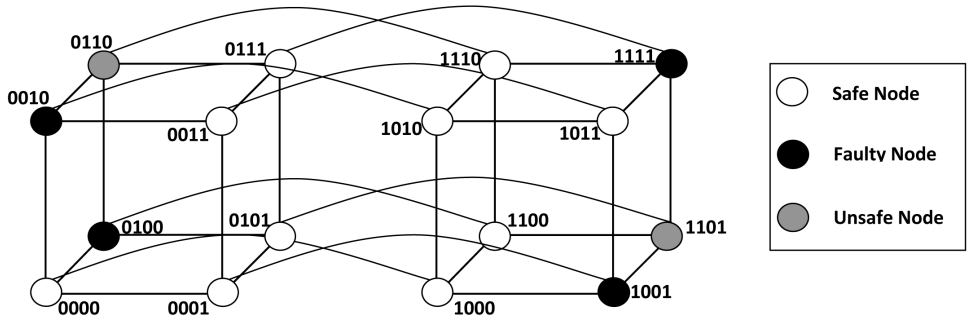

Figure 2.16: Example of node status in 4-cube.

For increasing the number of possible feasible paths, this technique reduces the number of unsafe nodes in H_n. In the Figure 2.16, there are two disconnected sets of safety nodes:– set 1: 0001, 0011, 0101, 0111 and set 2: 1000, 1010, 1100, 1110 and each set makes a subcube of dimension 2. If every node in the H_n is faulty and/or unsafe, then a faulty hypercube is called *fully unsafe*. The routing is expected to be very typical in a *fully unsafe* hypercube. If an n-dimensional H_n having n faulty nodes, then it is called *fully unsafe*. If the number of faulty nodes in an n-dimensional H_n is less than n, then the n-cube may be safe. If a node has k faulty neighbours such that $2 \leq k \leq n$, then every non-faulty node in k-subcube is unsafe.

Example 2.5.2. Consider the faulty 4-dimensional hypercube as shown in the Figure 2.17. In the figure, there are six unsafe nodes, i.e.$\{0000, 0110, 1010, 1011, 1100, 1110\}$. Among these only node 1110 is strongly unsafe, remaining nodes are ordinary unsafe nodes. Routing is done over the feasible minimum length path from the strongly unsafe node $S = 1110$ to the ordinary unsafe node $D = 0000$. This algorithm can also deal faulty links.

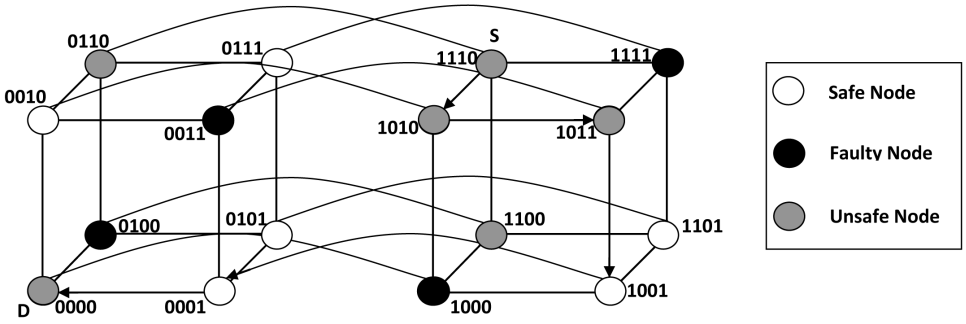

Figure 2.17: Example of strongly and ordinary unsafe nodes in 4-cube.

2.5.3.4 Reachability based Routing

Kaneko [68] improve the node safety technique by given reachability concepts. A node is *fully reachable* with respect to Hamming distance h, if every non-faulty node is apart from that node by h is reachable through path length h of that node. Every non-faulty node is a safe node with respect to Hamming distance 1. Based on this, a non-faulty node is a safe node with respect to h if it is adjacent to at least $n - h - 1$ safe nodes with respect to distance $h - 1$.

Example 2.5.3. Consider the 4-cube shown in Figure 2.18. In this figure, there are five faulty nodes 0001, 0100, 1100, 1101, 1110. Node 1111 is fully reachable with respect to distance 2, since the Hamming distance from 1111 to the nodes 0011, 0101, 0110, 1001, and 1010 is 2. It should be noted that node 1111 is not safe with respect to distance 2.

The performance of this algorithm has been compared with the algorithm of node safety. It has been shown that this algorithm can find routing paths without including any faulty nodes.

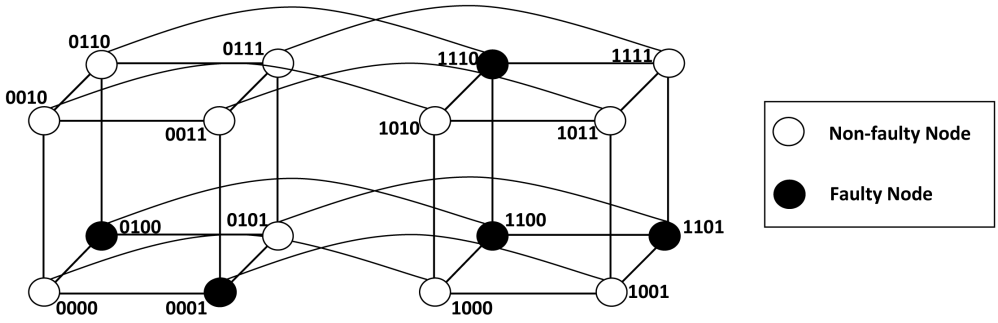

Figure 2.18: Example of node reachability in 4-cube.

2.5.4 Multicasting/Broadcasting in Faulty Hypercube

In multicasting, sending a message from a given node in a healthy hypercube– the source node has the capability to decide which of its neighbour nodes should receive the message. Upon receiving the message, each node matches its own address with the destination address. If matched, then the node will route the message to the local processor and remove the address from the list. If the list gets empty, then the receiving node is a leaf node and no more forwarding is required. Otherwise, receiving node forward the message to its descending neighbours in the multicast tree. At each forward node, the binary addresses of all destination nodes are calculated for voting to select the dimensions. The maximum number of ones in the dimension position decides the selected dimensions. Lan et al. [75] presented two types of multicasting algorithms which guarantee successful routing of messages in faulty hypercube. In the first type, each non-faulty node was assumed to has maximum one faulty node. In the second type, each non-faulty node have at least two faulty nodes. These algorithms uses n-bit status vector S, where each bit represents the position of neighbouring node such that 0 indicates the corresponding neighbouring node is faulty while 1 indicates the corresponding neighbouring node is non-faulty.

Example 2.5.4. This example shows the procedure of designing a multicast tree in a 5-dimensional hypercube H_5. The source node s, the destinations nodes d, faulty node F at dimension 1 and status vector S are set as follows:

$s = 01100 : 12$

$d = \{d_1 = 01000 : 8, d_2 = 01011 : 11, d_3 = 01101 : 13, d_4 = 10111 : 19, d_5 = 11000 : 20, d_6 = 11110 : 26\}$

Non-faulty case, $s \oplus d$=result	Dimension 1 is faulty	Row sum
$01100 \oplus 01000 = 00100$	00100	1
$01100 \oplus 01011 = 00111$	00101	2
$01100 \oplus 01101 = 00001$	00001	1
$01100 \oplus 10111 = 11011$	11001	3
$01100 \oplus 11000 = 10100$	10100	2
$01100 \oplus 11110 = 10010$	10000	1
Column sum	31303	

Table 2.2: The resulting relative addresses with respect to source node 12 (01100)

$F = 01110 : 14$
$S = 11101$

Table 2.2 shows the resulting relative addresses at the source node $s = 01100$, it has been computed by doing *XOR* operation between the source node and the destination nodes addresses assuming that dimension 1 is faulty. Table 2.2 authorize that bit position 0, 2, and 4 have column sum=3. Any one of them can be selected. For the sake of simplicity, suppose bit position 0 is selected. Since rows 2, 3, and 4 are having 1 at bit position 0; it has been used to form a destination sub-list. The neighbour is $01100 \oplus 01011$ and the sub-list is $\{01101, 10111, 01011\}$. Now, row 2, 3, and 4 are reset to all zeros and the new table is constructed based on the next sub-list as {11000, 11110}. This procedure is repeated in order to construct the multicast tree shown in Figure 2.19.

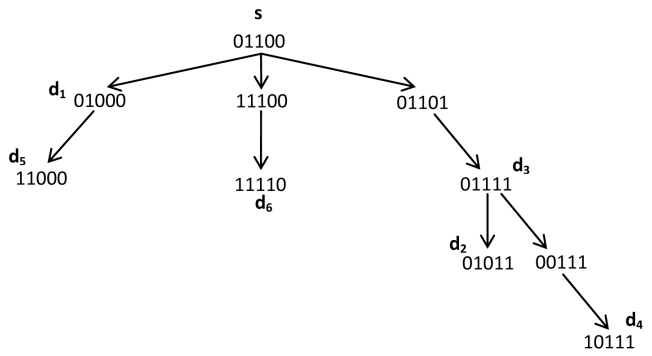

Figure 2.19: Multicast tree in H_5 with reference to Table 2.2.

If the set of all faulty nodes is known to n-dimensional hypercube, then broadcasting a message on H_n in the presence of up to $(n-2)$ faulty nodes and/or links is exactly n time units. The solution of broadcasting a message is the use of minimal number of messages, i.e. $2^n - 1$. The algorithm is based on constructing paths for a faulty hypercube. The paths are the collection of path that connect a source node to all other nodes in a hypercube. Then these paths partitioned into a number of sub-collections $P(x)$, one individual path for each destination node x. A path collection is defined by the following properties:

1. For each node x, the sub-collection $P(x)$ comprises at least $(n-1)$ paths from the source node to node x.
2. For each node x, the paths are mutually node-disjoint in the $P(x)$.
3. The paths in each $P(x)$ are arranged in an increasing order of path length.
4. The maximum path length in $P(x)$ is n.
5. For each path in the $P(x)$ the prefix of this path conducting from the source node to node y.

In this algorithm, there are up to $(n-2)$ faulty nodes and/or links. A path is called injured path if it contains a faulty node and/or link. Rodrigues [106] proposed an algorithm that constructs virtual spanning trees connecting the nodes, through which messages are disseminated. These spanning trees are dynamically built embedded on a virtual hypercube-like topology. Nodes can fail by crashing, and up to $n-1$ nodes may crash at any given time.