INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

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

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

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

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

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9023586

Probabilistic multiprocessor and multicomputer diagnosis

Lee, Sunggu, Ph.D.

The University of Michigan, 1990

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

PROBABILISTIC

MULTIPROCESSOR AND MULTICOMPUTER

DIAGNOSIS

by

Sunggu Lee

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer Science and Engineering)
in The University of Michigan
1990

Doctoral Committee:

Professor Kang G. Shin, Chair
Associate Professor John R. Birge
Professor John P. Hayes
Professor Keki B. Irani
Assistant Professor Pinaki Mazumder

RULES REGARDING THE USE OF

MICROFILMED DISSERTATIONS

Microfilmed or bound copies of doctoral dissertations submitted to The University of Michigan and made available through University Microfilms International or The University of Michigan are open for inspection, but they are to be used only with due regard for the rights of the author. Extensive copying of the dissertation or publication of material in excess of standard copyright limits, whether or not the dissertation has been copyrighted, must have been approved by the author as well as by the Dean of the Graduate School. Proper credit must be given to the author if any material from the dissertation is used in subsequent written or published work.

To my parents

ACKNOWLEDGEMENTS

The author wishes to thank Professor Kang G. Shin for his guidance and encouragement during the course of his graduate study at The University of Michigan. Many thanks go to my committee members for their helpful comments and suggestions. I also wish to thank my fellow graduate students at the Real-Time Computing Laboratory and Beverly J. Monaghan for their help and stimulating discussions.

Financial support provided by the National Aeronatics and Space Administration under Grants NAG-1-492 and NAG-1-296 is gratefully acknowledged.

Finally, I would like to thank my family and especially my wife for their support and encouragement.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

CHAPTER 1

INTRODUCTION

The progress made in the design of powerful single-chip computers (and even single-chip multiprocessors) has led to the construction of increasingly sophisticated multiprocessor and multicomputer systems with tens of thousands of processors. Even if the MTTF (mean time to failure) of a processor is 10 years, a system with 10,000 processors can expect to have 952 failing processors in one year, assuming an exponential failure arrival process. Thus, it is imperative that such systems be provided with good fault-tolerance capabilities. In addition, in order to maintain a highly reliable system, faulty processors must be diagnosed and periodically removed (either physically or by reconfiguration) from the system. The problem of identifying the faulty processors in such a large system is an extremely difficult task, especially since it is possible for faulty processors to accuse non-faulty processors of being faulty and since processors can be intermittently faulty.

1.1. Objectives

This dissertation addresses the general problem of diagnosing faulty processors in a multiprocessor/multicomputer system. The system is modeled by a directed graph in which the vertices correspond to processors and the edges correspond to testing assignments. The fault *syndrome* is defined to be a binary labeling of the directed edges in which each label represents the result of the corresponding inter-processor test. Commonly referred to as *system-level diagnosis*, diagnosis using this type of system model was first addressed by

Preparata *et. al.* [52]. Over the past 20+ years, numerous theoretically interesting models and results have been presented for this type of diagnosis [1, 13, 14, 32, 47, 56, 57, 62]. However, very few practical approaches have come from these efforts, mainly because of the overly restrictive assumptions that were made in order to analytically obtain good results. Thus, the following seven objectives have been identified for designing useful and practical solutions to the general diagnosis problem addressed.

1. It has to be optimal or close to optimal in terms of diagnostic accuracy (percentage of correct diagnoses),
2. it has to be efficient (quadratic or lower computational complexity),
3. the testing required to obtain the fault syndrome has to be efficient,
4. it has to be implementable on all types of system topologies,
5. it has to be able to handle both intermittent and permanent faults,
6. it has to be implementable in a distributed manner with low communication and processing overhead, and
7. the requirements imposed by the solution have to be practical.

1.2. Overview of Dissertation

In addressing the multiprocessor/multicomputer diagnosis problem, an *incremental diagnosis strategy* is proposed, in which testing is conducted for a predetermined interval of time and then diagnosis is performed using these test results and the fault coverages obtained. If the quality of diagnosis obtained is unacceptable, then further testing is requested to obtain higher fault coverage values. The results of the diagnosis can be used to guide the testing process. This cycle of testing and diagnosis is repeated until a diagnosis of sufficiently high quality is achieved or the time allotted for diagnosis expires. In the latter case, it is desirable to produce the best diagnosis given the information available at the time. In general, the time required for testing will be much longer than the time required for diagnosis. If a sufficiently good diagnosis can be obtained even with low fault coverage values, then overall testing and diagnosis time can be substantially reduced. These concerns are addressed by the seven objectives listed in the previous section.

Optimal probabilistic diagnosis algorithms are presented. If the entire syndrome information is used, then optimal diagnosis is an NP-hard problem [43]. However, it is possible for each node to use only part of the syndrome information to arrive at a diagnosis of itself, in which case the diagnosis component must be assumed to be part of an ultra-reliable hardcore. Several categories of diagnosis are defined based on the type of fault syndrome information used in the diagnosis. For each such category defined, Bayesian probability analysis is used to derive optimal probabilistic diagnosis algorithms. One reason that this categorization is important is because the amount of communication overhead required to implement distributed diagnosis algorithms is significantly different for each category. Also, because these algorithms must be implemented as part of an ultra-reliable hardcore, simplicity of implementation is important, and the algorithms in each category differ in the calculations required.

It is possible to achieve higher diagnostic accuracy by using *multiple* rather than a single fault syndrome. In *multiple syndrome diagnosis*, it is assumed that the testing is conducted in stages and a fault syndrome is collected after each stage. The proposed multiple syndrome diagnosis method is dependent on the use of a *comparison-testing* method, in which a test between two processors is actually a comparison of the outputs of the processors on identical tasks. A special method of obtaining the multiple fault syndromes, termed *multiple syndrome testing*, is also used. As will be shown, it is possible to achieve significantly higher diagnostic accuracy than single syndrome diagnosis methods by using multiple syndrome testing, even when the total time devoted to testing is the same. In diagnosis using multiple syndromes, it is important that the fault syndrome obtained in a later testing stage be partially dependent on the fault syndrome obtained from an earlier testing stage. The main reason that higher diagnostic accuracy is possible with multiple syndrome testing is that certain "patterns" of fault syndromes implicate certain sets of processors as being faulty. It is not possible to achieve this effect when only one fault syndrome is used. Based on Bayesian probability analysis, an

optimal multiple syndrome probabilistic diagnosis algorithm is derived.

Finally, it is important to be able to implement all of the proposed algorithms efficiently in a distributed manner. In distributed diagnosis, the syndrome information that each processor needs must be reliably communicated to it. Since processors can be intermittently faulty, one cannot use a technique such as that used in [34] in which a non-faulty processor only sends information to those processors that pass its test. Instead, the communication problem must be viewed as a *Byzantine Generals* [41] type of problem, and multiple copies of messages must be sent along disjoint paths. If each processor uses the entire syndrome information in the diagnosis, then an all-to-all reliable broadcast operation is required. This results in an extremely high communication overhead because of the requirement of sending multiple copies of messages along disjoint paths. However, if each processor uses purely "local" syndrome information in the diagnosis, then the communication problem is considerably simplified. Our analysis shows that it is possible to execute distributed versions of algorithms that use a summarized form of the global syndrome information by only requiring a relatively small number of reliable communication steps between adjacent processors. The problem of distributing the testing information is then reduced to a reliable multicast problem, in which each processor must reliably send messages to its immediate neighbors.

New all-to-all reliable broadcast and nearest-neighbor reliable multicast algorithms, which are designed to work with *virtual cut-through* switching [38] on regular meshes and hypercubes, are presented. When routing a message using virtual cut-through, instead of buffering the message in its entirety at an intermediate node, the message is forwarded directly from incoming to outgoing channels, referred to as a *cut-through* operation. Only a few bytes of routing information have to be examined (using an "on-line" buffer) to determine the outgoing channel of the message. If the outgoing channel is busy, the message must then be buffered in intermediate storage as in traditional store-and-forward routing. The all-to-all

reliable broadcast and nearest-neighbor reliable multicast algorithms presented are designed such that the maximum number of cut-throughs can be achieved.

1.3. Outline of Dissertation

This dissertation is organized in the following manner. In Chapter 2, the necessary background for the dissertation is presented. Some of the definitions and notation used in the dissertation are introduced. Then the testing and probability models used are presented, and diagnosis methods are categorized based on the types of syndrome information used in the diagnosis. A brief survey of related diagnosis methods in the literature is also presented. From this survey, it is concluded that a probabilistic diagnosis method is the most appropriate in addressing the seven objectives outlined in Section 1.1. Practical methods for performing the inter-processor tests required for multiprocessor/multicomputer diagnosis are also identified in this chapter.

In Chapter 3, an algorithm is presented for finding the most probable diagnosis given the entire syndrome. Given that a certain type of syndrome information is being used, it is proven that the optimal diagnosis algorithm is the one which finds the most probable diagnosis given the available syndrome information. Thus, the algorithm presented in Chapter 3 is the globally optimum diagnosis algorithm. Since the problem of finding the most probable diagnosis given the entire syndrome is an NP-hard problem, this solution is $O(2^{|F|})$, where F is the set of faulty processors. A set of methods based on Bayesian probability analysis is used to reduce the effort of searching for the most probable diagnosis. This set of methods for efficiently conducting the search for the most probable diagnosis permits the diagnosis algorithm to be used on systems with up to several hundred faulty processors. In practice, the optimum diagnosis algorithm should be used in conjunction with a fast probabilistic diagnosis algorithm (such as one of the algorithms presented in later chapters) to produce the "best" diagnosis

whenever possible and to produce a "good" diagnosis in all other cases.

Chapter 4 presents a series of probabilistic diagnosis algorithms that are the optimal single syndrome diagnosis algorithms for the categories of diagnosis in which restricted fault syndrome information is available. The optimal diagnosis algorithms are compared with each other and with several related diagnosis algorithms in the literature using probabilistic analysis and simulation results. Based on this analysis, a simple near-optimal diagnosis algorithm which does not use probability parameters is also presented.

Chapter 5 addresses optimal multiple syndrome probabilistic diagnosis. Fussell and Rangarajan [30] presented a multiple syndrome diagnosis algorithm based on the use of two threshold values. Probability analysis is used to derive the optimum values for these two thresholds. The resulting diagnosis algorithm is the optimal multiple syndrome diagnosis algorithm. Probability analysis is also used to demonstrate that higher diagnostic accuracy can be achieved with multiple fault syndromes.

Chapter 6 addresses the distributed implementation of the diagnosis algorithms presented by solving the problems of all-to-all reliable reliable broadcast and nearest-neighbor reliable multicast using virtual cut-through switching. The all-to-all reliable broadcast algorithm works on a class of regular interconnection networks that includes regular mesh and hypercube structures. Nearest-neighbor reliable multicast using virtual cut-through is presented as a type of "mosaic-fitting" problem, in which differently shaped pieces must be brought together such that no empty spaces remain. Specific solutions are presented for the square mesh, hexagonal mesh, and binary hypercube structures.

The dissertation concludes with Chapter 7, where the main contributions of this dissertation are summarized and important future work is identified.

CHAPTER 2

PRELIMINARIES

This chapter presents the notation, testing model, and probability model used in the dissertation, describes practical system-level testing methods, categorizes diagnosis methods, and provides a brief survey of related work.

2.1. Notation and Definitions

In large multiprocessor and multicomputer systems, a point-to-point interconnection network is commonly used to connect the set of processing nodes. The system S is represented by an undirected graph $G = (V, E)$ in which the set of vertices (or nodes) V corresponds to the set of N processing nodes and the set of edges E corresponds to the set of communication links in the interconnection network. The nodes in V are denoted by u_i $(0 \leq i \leq N-1)$ and the edges in E are denoted by e_{ij} $(0 \leq i, j \leq N-1)$. If directed communication links are used, each edge corresponds to 2 communication links. Thus, given an undirected graph G, the corresponding directed graph $G^{dir} = (V, E^{dir})$ is defined to be the graph G with every undirected edge replaced by two directed edges (one in each direction). Reliable broadcast and multicast algorithms are described with respect to G^{dir}.

For diagnosis purposes, the set of testing assignments in S is represented by a directed graph $G_T = (V_T, E_T)$, called the *testing graph*. G_T is assumed to be a subgraph of G^{dir} in which $V_T = V$ and $E_T \subseteq E^{dir}$. Each edge $e_{ij} \in E_T$ represents the fact that u_i tests u_j. Test

outcomes are represented by binary variables a_{ij} such that $a_{ij} = 1$ if u_j fails u_i's test and $a_{ij} = 0$ if u_j passes u_i's test. a_{ij} is undefined if u_i does not test u_j. A *fault set* (or *diagnosis*) $F \subseteq V_T$ is the set of nodes such that each $u_i \in F$ is postulated to be faulty and each $u_j \in V_T - F$ is postulated to be fault-free. A (fault) *syndrome* SD is a function from E_T to $\{0, 1\}$. The function SD is defined such that for all $e_{ij} \in E_T$, $SD(e_{ij}) = a_{ij}$. Given a syndrome SD, a diagnosis is said to be *correct* iff the set of nodes diagnosed to be faulty is the same as the actual fault set.

The following additional definitions and conventions are used. The set of nodes that a given node u_i tests will be denoted by $\Gamma(u_i)$. Likewise, the set of nodes that test u_i is denoted by $\Gamma^{-1}(u_i)$ and $\Gamma^{-1}(u_i) = \Gamma_1^{-1}(u_i) \bigcup \Gamma_0^{-1}(u_i)$, where $\Gamma_k^{-1}(u_i) = \{ u_j \in \Gamma^{-1}(u_i) : a_{ji} = k \}$, $k = 0, 1$. The *one-condensation of G*, denoted by $G_1 = (V_1, E_1)$, is the subgraph of G_T with $E_1 = \{e_{ij} \in E_T : a_{ij} = 1\}$ and $V_1 = \{u_i \in V_T : u_i$ is the endpoint of an edge in $E_1\}$. Given a node $u_i \in V_T$, $d(u_i) = |\{u_j \in \Gamma^{-1}(u_i) : a_{ji} = 1\}|$. In discussing probabilities of events where the set of basic events is clear, the notation $P(x)$ is used to denote the probability of the event x. Section 2.4 introduces the probability model and notation used when more formal treatment is required.

2.2. Testing Model

The testing model used in this dissertation is the "partial tester" model A_{pT} [28]. In this testing model, the result of a non-faulty node testing another non-faulty node is the only completely reliable test result. The probability parameters that describe the possible values of a_{ij} given different fault statuses of u_i and u_j are given in Table 2.1. The fault status of u_k ($k = i$ or j) is denoted by δ_k (for u_k is faulty) and $\bar{\delta}_k$ (for u_k is non-faulty). Let $fs_i \in \{\delta_i , \bar{\delta}_i\}$ and $fs_j \in \{\delta_j , \bar{\delta}_j\}$. All possible combinations of fs_i, fs_j, and $a_{ij} = m$ values are shown along with $P(a_{ij} = m \mid fs_i , fs_j)$ in each of these situations, where $m = 0$ or 1. Thus, p_{ij} is the

probability that a non-faulty node u_i will correctly diagnose a faulty node u_j. Hence, under a permanent fault model and assuming that all possible faults within a node are equally likely, p_{ij} is the fault coverage of the test applied by u_i on u_j. For ease of notation, p_{ij} will be referred to as fault coverage even when intermittent faults are permitted. r_{ij} and s_{ij} are the probabilities of a faulty node correctly diagnosing a non-faulty and faulty node, respectively. As explained in [7], r_{ij} and s_{ij} can model the extent to which a faulty node u_i can pass judgement on u_j. These two parameters are useful in modeling the behavior of faulty nodes. Finally, f_i will denote the prior fault probability of u_i. In part of the probability analysis, the use of average probability parameter values will be required. The average values of probability parameters will be denoted by the corresponding letters without subscripts. Thus, for example, f and p will refer to the average f_i and p_{ij} values, respectively.

fs_i	fs_j	m	$P(a_{ij} = m \mid fs_i, fs_j)$
$\bar{\delta}_i$	$\bar{\delta}_j$	0	1
$\bar{\delta}_i$	$\bar{\delta}_j$	1	0
$\bar{\delta}_i$	δ_j	1	p_{ij}
$\bar{\delta}_i$	δ_j	0	$1 - p_{ij}$
δ_i	$\bar{\delta}_j$	0	r_{ij}
δ_i	$\bar{\delta}_j$	1	$1 - r_{ij}$
δ_i	δ_j	1	s_{ij}
δ_i	δ_j	0	$1 - s_{ij}$

Table 2.1: Probability parameters.

2.3. Methods for Inter-processor Testing

There are several possible methods for conducting the inter-processor tests required by system-level diagnosis. The most generally accepted method is *comparison-testing* [13]. In comparison-testing, in order for node u_i to test another node u_j, identical application tasks are executed on the two nodes. The test result is a 1 if and only if u_i and u_j produce different results. This also results in a test for node u_i by u_j. Thus, comparison-testing only requires

an undirected testing graph model. NMR (N-Modular Redundancy) [61] can be viewed as an implementation of comparison-testing. In NMR, the nodes are grouped into clusters of N nodes each. Within a cluster, all of the nodes execute the same task and a voter votes on the outputs of the nodes. Rangarajan and Fussell [55] investigated the use of NMR for diagnosis.

Another practical method of system-level testing is *on-line processor monitoring*. In the on-line monitoring method of [4], a simple hardware device is used to repeatedly capture a block of data from the tested node and then analyzed on the testing node for the presence of an error. Indications of abnormal behavior in the captured data block signal a faulty processor. Alternatively, one could consider creating a system task which snoops on the packets coming in and going out of a node. The task examines the packet formats and checks whether they conform to the standard. The error rates detected in the packets and the traffic volume can also be monitored to decide whether the node from which the packets originate is faulty. Also, to detect communication link failures and node crashes, a status monitor task can be established on each node to periodically exchange data with neighboring nodes.

Finally, a low-level self-testing method can be used [40]. In this method, each node performs a self-test and stores the result of the self-test in a special fault status register. When a node wishes to test another neighboring node, it simply reads the special fault status register of the node to be tested. This method requires that the fault status register and the mechanism for reading it be part of an ultra-reliable hardcore. Also, the reader will note that this method simply relegates the testing problem to a lower level.

This dissertation uses a directed testing graph model so that all inter-processor testing methods can be included into the testing model. However, comparison-testing will be assumed whenever examples using the average probability parameter are needed. Also, all simulations reported in Chapters 4 and 5 were done assuming comparison-testing.

2.4. Categorization of Diagnosis Algorithms

Methods for diagnosis can be categorized based on the type of syndrome information used to identify faulty nodes. Looking at the diagnosis algorithm executed on each node, the maximum amount of information that can be used by each node is the entire syndrome. A diagnosis method which uses the entire syndrome is defined as a *category 1 diagnosis* method. The optimal diagnosis algorithm in this category is the algorithm which finds the most probable fault set given the syndrome. This diagnosis algorithm was used in [7] and [43]. This category of diagnosis is inherently inefficient because the entire syndrome must be reliably communicated to each node.

Suppose instead that each node is aware of only the part of the syndrome that directly implicates it as faulty or non-faulty, i.e., each node u_i is only aware of the a_{ji} values such that $u_j \in \Gamma^{-1}(u_i)$. This is referred to as *local* syndrome information. It is possible to summarize this local syndrome information as the ordered pair $(d(u_i), |\Gamma^{-1}(u_i)|)$, where $d(u_i)$ is as defined above. In *category 2 diagnosis*, faulty nodes are identified one at a time, using local syndrome information and the identity of the nodes diagnosed as faulty in previous steps. Thus, since the calculations of all of the nodes have to be examined to determine the order in which faulty nodes are identified, a summarized form of the global syndrome is effectively being used in the diagnosis. This method approximates the category 1 method by looking at individual nodes instead of subsets of nodes. *Category 2A diagnosis* is similar to category 2 diagnosis except that summarized local syndrome information is used. The diagnosis methods of [5, 19] fall into this latter category. In *category 3 diagnosis*, local syndrome information is used to identify each node as faulty or non-faulty independently of the other nodes in the system. In *category 3A diagnosis*, summarized local information is used to diagnose each node. The diagnosis method of [5] can be easily modified to fall into this category. Table 2.2 summarizes this categorization.

Category	Syndrome Information Used
1	All
2	Local & nodes previously diagnosed as faulty
2A	Summarized local & nodes previously diagnosed as faulty
3	Local
3A	Summarized local

Table 2.2: Categories of diagnosis and syndrome information used.

The categories of diagnosis defined differ in the amount of communication overhead required for distributed diagnosis and the computational complexity, diagnostic accuracy, and ease of hardware implementation of the diagnosis algorithms. Category 1 diagnosis requires the highest level of communication overhead, has exponential computational complexity, has the highest level of diagnostic accuracy, and is the most difficult to implement in hardware. Category 2 diagnosis requires a moderate amount of communication overhead, has linear to quadratic computational complexity, has the second highest level of diagnostic accuracy, and is difficult to implement in hardware. Category 2A diagnosis is the same as category 2 diagnosis except that it has slightly lower diagnostic accuracy and is much simpler to implement in hardware. Category 3 diagnosis requires the least amount of communication overhead, has linear computational complexity, has lower diagnostic accuracy than category 2 or 2A, and has a very simple hardware implementation. Finally, category 3A diagnosis differs from category 3 only in that it has slightly lower diagnostic accuracy.

2.5. Probability Model

A probability model is defined by defining a probability space, which is a triple (Ω , Θ , P), where Ω is the sample space, Θ is the event space, and P is a probability measure. However, for each category of diagnosis, the type of syndrome information used in the diagnosis is different. Thus, for each category of diagnosis, a different probability space is used in talking about the probability of certain types of syndromes and fault sets being present.

Consider category x diagnosis, where x can be any of the categories defined above. Suppose the syndrome information used in diagnosing node $u_i \in V_T$ is denoted by SD_i. SD_i is a restricted form of the information present in the syndrome SD. For a given node u_i, its fault status set is defined as $Status\_Set_i = \{\delta_i , \bar{\delta}_i\}$. Let us consider an arbitrary category x diagnosis algorithm A. In order for the diagnosis by A to be correct, A's diagnosis of each node $u_i \in V$ must be correct. Let A_i denote the part of A which diagnoses the fault status of node u_i, and define as basic events the pairs (SD_i , fs_i), where SD_i is a "partial syndrome" and $fs_i \in Status\_Set_i$. The set of all possible SD_i's will be denoted by SD_i^{all}. The diagnosis of u_i by A when executed on a syndrome containing the partial syndrome SD_i is denoted by $Diag_{A_i}(SD_i) \in Status\_Set_i$.

For category x diagnosis on node u_i, the sample space $\Omega_i^x = \{(SD_i , fs_i) : SD_i \in SD_i^{all} , fs_i \in Status\_Set_i\}$, the event space Θ_i^x is all possible subsets of Ω_i^x, and the probability measure P_i^x is defined for category x diagnosis such that it is a legitimate probability measure. Although not explicit in this notation, the probability measure P_i^x is also dependent on the testing graph $G = (V, E)$. Given a testing graph G and a diagnosis algorithm A, let $Correct_G(A_i) = \{(SD_i , fs_i) : Diag_{A_i}(SD_i) = fs_i\}$. For a testing graph G, the probability of correct diagnosis of u_i by A is

$$P_i^x(Correct_G(A_i)) = \sum_{SD_i \in SD_i^{all}} P_i^x(SD_i , Diag_{A_i}(SD_i)) \quad .$$

2.6. Overview of Related Work

System-level diagnosis methods can be broadly classified into complete-test, incomplete-test, and probabilistic methods.

2.6.1. Complete-Test Methods

Complete-test methods assume that tests executed by one unit on another are always complete in the sense that all possible faults are covered by the tests. Preparata *et. al.*'s system-level diagnosis model (referred to as the *PMC model*) and method [52] is representative of the efforts in this class of methods. In this diagnosis model, it is assumed that a fault-free unit u_i will always correctly determine a faulty unit u_j which it is assigned to test. It is further assumed that there is an upper bound t on the maximum size of allowable fault sets. A system S is said to be t–diagnosable if all faulty units within the system can be identified without replacement provided the number of faulty units does not exceed t [52]. There exist polynomial-time algorithms for determining t–diagnosability [62] and for solving the diagnosis problem [15, 17, 50]. Other efforts in this class of methods include t/s–diagnosability [27], t/t–diagnosability [14, 37], and p–t–diagnosability [18, 29, 47, 63, 64].

There are several problems with these complete-test methods. The major problem is their assumption that all tests are complete. This requirement is impossible to meet in incremental diagnosis, where diagnosis begins before the testing is complete, and in diagnosis with intermittently faulty units. Another problem with the above methods is that they limit the types of testing graphs allowed. The types of testing graphs required bear no relation to the types of interconnection networks in use or being proposed for distributed systems. Finally, a problem with the PMC model is that it requires a large number of testing links [37].

2.6.2. Incomplete-Test Methods

There are several diagnosis methods which deal with diagnosis in the presence of possibly incomplete tests, i.e., tests with less than 100% fault coverage. Assuming possibly incomplete tests is equivalent to permitting intermittent faults from a system-level diagnosis viewpoint [5]. This is because if a unit u_j is intermittently faulty, then any fault-free unit u_i

which tests u_j may evaluate u_j to be faulty or fault-free, depending upon the status of u_j during the test. Additionally, u_j can arbitrarily evaluate any units that it tests since it is faulty. The same statement can be made if u_j is assumed to be permanently faulty but the tests performed upon u_j are incomplete.

Mallela and Masson [48] introduced the problem of diagnosable systems for intermittent faults. They defined a system to be t_i–diagnosable if, when no more than t_i units are intermittently faulty, a fault-free unit will never be diagnosed as faulty and the diagnosis may be incomplete but never incorrect [48]. They characterized t_i–diagnosable systems and gave an exponential time complexity procedure for determining $\hat{t}_i$, the least upper bound of t_i. Yang and Masson [65] gave a linear time diagnosis algorithm for t_i–diagnosable systems. A distributed implementation of this diagnosis algorithm was described in [66]. In [49], Mallela and Masson generalized the t_i–diagnosability model to hybrid fault situations, which they defined to be explicitly bound combinations of permanently and intermittently faulty units. Butler [9] presented an algebraic method for finding all possible sets of intermittently faulty and permanently faulty units given a syndrome. His method extends the fault pattern generation technique of [1] to the hybrid fault case.

A major problem with the above models is that, even with Yang and Masson's efficient algorithm [65], the probability that a single faulty unit can be diagnosed from a syndrome approaches 0 as the total number of units approaches infinity. In general, diagnosis algorithms for intermittent faults which do not use probability parameters tend to be too conservative and only identify units which are definitely faulty given the syndrome and a fault set size limit t_i. Also, t_i–diagnosable systems require even more testing links than the PMC model with $t = t_i$. Additionally, Butler's algebraic method [9] is an exponential complexity procedure which is impractical even for systems with more than about 20 units.

2.6.3. Probabilistic Methods

Several probabilistic methods have been introduced for solving the system-level diagnosis problem. Blount [7] used probability parameters and the "0-information tester" model [28] to define a mapping from syndromes to fault patterns, which is used as the basis of the diagnosis. Blough *et. al.* [5] presented a linear-time algorithm which utilizes a relatively small number of tests and achieves correct diagnosis with high probability. Dahbura *et. al.* [19] gave a similar quadratic-time heuristic diagnosis algorithm. Both of these algorithms are heuristic algorithms which use the number of 1-links (e_{ij} such that $a_{ij} = 1$) incident on a unit as the basis for classifying the unit as faulty or fault-free. However, in Blough *et. al.*'s algorithm, the probability of 100% correct diagnosis can be shown to approach 1 as $N \rightarrow \infty$ provided that $|\Gamma^{-1}(u_i)|$ grows faster than $\log N$ for all $u_i \in V$. Recently, Fussell and Rangarajan [30] presented a fast probabilistic algorithm which uses multiple syndromes and produces asymptotically correct diagnosis provided the number of *tests* (as opposed to testers) conducted on each unit grows faster than $\log N$. This algorithm requires the use of multiple syndromes and is limited to syndromes formed using comparison-testing.

Given fault syndromes generated from tests with incomplete fault coverage, the probabilistic methods in general have much better performance (in terms of diagnostic accuracy) than the other types of diagnosis methods which were examined. The reason for this is that the probabilistic diagnosis algorithms are designed to work well given almost "any" type of fault syndrome. There is no upper bound on the number of permitted faulty units. There is also no restriction on the structure of the testing graph. In addition, *if* it turns out that the structure of the testing graph satisfies certain desirable properties on connectivity, Blough *et. al.*'s algorithm is guaranteed to asymptotically produce 100% correct diagnosis as N grows to infinity. Given these considerations, it is concluded that a probabilistic diagnosis method is the most practical for the system-level diagnosis problem considered in this dissertation.

CHAPTER 3

MOST PROBABLE DIAGNOSIS

This chapter presents a diagnosis algorithm, referred to as the *MPD algorithm*, which uses the entire syndrome information and guarantees the most probable diagnosis based on Bayesian probability analysis. Since determining the most probable diagnosis given global syndrome information is an NP-hard problem, this diagnosis method has exponential computational complexity. However, in the MPD algorithm, several methods based on probabilistic analysis are used to constrain the search for the most probable diagnosis. The most important among these is a method for dynamically determining an upper bound on the size of the fault set which must be considered. Because of these probabilistic methods, the MPD algorithm can be used on systems with up to several hundred nodes.

It is shown how this diagnosis algorithm can be used as part of an *incremental diagnosis strategy*, in which diagnosis begins before the testing is complete. The performance of the MPD algorithm is compared with a probabilistic diagnosis algorithm which can asymptotically produce 100% correct diagnosis on certain types of systems.

3.1. Introduction

In general, the time required for testing will be much longer than the time required for diagnosis. If a sufficiently good diagnosis can be obtained even with low fault coverage values, then testing time can be substantially reduced. Also, since the tests used in system-level diagnosis are typically comparisons of tasks or on-line monitoring of one processor by

another, the fault coverages obtained can be expected to be quite low (assuming a realistic low-level fault model). This justifies the use of a sophisticated diagnosis algorithm if the diagnosis produced is more accurate and the time required for diagnosis is still reasonable.

In addition, it is desirable to be able to use the diagnosis method as part of an *incremental diagnosis strategy*, in which testing is conducted for a predetermined interval of time and then diagnosis is performed using these test results and the fault coverages obtained. If the quality of diagnosis obtained is unacceptable, then further testing is requested to obtain higher fault coverage values. The results of the diagnosis can be used to guide the testing process. This cycle of testing and diagnosis is repeated until a diagnosis of sufficiently high quality is obtained or the time allotted for diagnosis expires. In the latter case, the diagnosis algorithm should still give the best diagnosis given the information available at the time.

Based upon a survey of the currently available methods for system-level diagnosis in the literature (Section 2.6), it was concluded that a probabilistic diagnosis algorithm was the most practical and appropriate for diagnosis using inter-processor tests with low fault coverage. Among the probabilistic diagnosis algorithms, a few have the property that as the system size grows to infinity, the probability of correct diagnosis approaches 1 given certain constraints on the set of tests conducted [5, 30]. However, it can be proven that the optimal diagnosis algorithm is the one that guarantees that the most probable diagnosis is made [6]. Thus, it is highly desirable to be able to find the most probable diagnosis.

In this chapter, a diagnosis algorithm for finding the most probable diagnosis based on Bayesian probability analysis is presented. This algorithm, referred to as the MPD algorithm, is the optimal category 1 diagnosis algorithm (proven in Chapter 4). It is shown in Section 3.3 that finding the most probable diagnosis is an NP-hard problem. However, the search for the most probable diagnosis can be made to be exponential with respect to the number of *faulty* nodes. Our most significant contribution is the discovery of methods which are found

to make most probable diagnosis practical even for very large systems with up to several hundred nodes using currently available computers. By contrast, Blount's algorithm [7], which also finds the most probable diagnosis, cannot be used on systems with more than about 20 nodes due to the computational complexity of the algorithm. Through simulations, the performance of our diagnosis algorithm is compared to Blough *et. al.*'s algorithm [5], which is a fast probabilistic diagnosis algorithm with desirable asymptotic properties. This comparison supports the argument that the most probable diagnosis algorithm should be used in conjunction with a fast probabilistic diagnosis algorithm to achieve good diagnostic results. The MPD algorithm works with any general interconnection network and any set of probability parameters. As will be shown, because inter-processor tests with less than 100% fault coverage are premitted, the MPD algorithm is also applicable to the diagnosis of intermittent faults.

3.2. Background

In addition to the notation introduced in Chapter 2, the following definitions and conventions are used. Each fault set F is a *basic event* in a formal probability model. Likewise, each syndrome SD is a basic event. In particular, this means that $P(F)$ denotes the probability of the basic event F (*not* the probability of the event set F).

Definition 3.1: A diagnosis (or fault set) $F \subseteq V_T$ is said to be *consistent* if and only if there does not exist an edge $e_{ij} \in E_T$ such that $a_{ij} = 1$ and $u_i, u_j \in V_T - F$.

Definition 3.2: A set $F \subseteq V_T$ is said to be *minimally consistent* if F is consistent and no proper subset of F is consistent.

Definition 3.3: A *set of minimal consistent diagnoses* is defined as $MCD = \{ F \subseteq V_T \mid F \text{ is minimally consistent} \}$.

Among the probabilistic diagnosis methods surveyed in Section 2.6, Blount's method [7] is the only one which guarantees that the most probable diagnosis is found. However,

Blount's method [7] is a complete enumeration algorithm which is not only $O(2^{N+|E|})$ but always requires exactly $2^{N+|E|}$ iterations. By contrast, the MPD algorithm is $O(2^{|F|})$, where F is the set of faulty nodes. Because of this difference, our diagnosis algorithm can be used on testing graphs with an order of magnitude more nodes. The MPD algorithm should be used in conjunction with a fast probabilistic algorithm such as Blough *et. al.*'s algorithm or one of the algorithms in Chapter 4 in order to produce the "best" diagnosis whenever possible and a "good" diagnosis in all other cases. Blough *et. al.*'s diagnosis algorithm is chosen because of its desirable property of asymptotically correct diagnosis given certain conditions on the testing graph. Fussell and Rangarajan's algorithm [30] also has the property of achieving asymptotically correct diagnosis. However, their algorithm uses multiple syndromes, which is a somewhat different problem addressed in Chapter 5 and [44].

Blough *et. al.*'s (BSM) algorithm is an $O(|E|)$ approximation algorithm for the method used to diagnose mixtures of permanent and intermittent faults in [16]. This algorithm compares $d(u_i) \equiv |\{u_j \in \Gamma^{-1}(u_i) : a_{ji} = 1\}|$ to a threshold value κ_i, which is the assumed maximum number of faulty nodes testing u_i. Converting the formula for κ_i in [5] into our testing model and using our notation:

$$\kappa_i = \frac{1}{2} \sum_{u_j \in \Gamma^{-1}(u_i)} \left[f_j + p_{ji}(1 - f_j) \right] .$$

Algorithm BSM:

0. Let $F \leftarrow \emptyset$ be the set of diagnosed faulty nodes;
1. **For** each $u_i \in V_T$, let $D_i \leftarrow d(u_i) - \kappa_i$;
2. **While** $\max_{u_i \in V_T - F} D_i = D_j > 0$ **do**
   $F \leftarrow F \cup \{u_j\}$;
   for each $e_{jm} \in E_T$ such that $a_{jm} = 0$ **do**
   $D_m \leftarrow D_m + 1$;

3.3. Bayesian Analysis

Given a fault set F and a syndrome SD, the posterior probability of F given SD is

$$P(F|SD) = \frac{P(SD|F)\,P(F)}{P(SD)} = \frac{P(SD|F)\,P(F)}{\sum_{F' \subseteq V_T} P(SD|F')\,P(F')} \quad . \tag{3.1}$$

Thus, if $P(SD|F)$ and $P(F)$ can be determined, then the posterior probabilities of possible fault sets can be derived. $P(F)$ is the prior probability of the fault set F and can be easily determined if the probability of failure of each node is assumed to be independent of the probability of failure of the other nodes. Specifically,

$$P(F) = \prod_{u_i \in F} f_i \prod_{u_i \in V_T - F} (1 - f_i) \; . \tag{3.2}$$

$P(SD|F)$ is slightly more complicated but can be determined from the syndrome, the fault set, and the probability parameters discussed in Section 3.2.1. Given a binary variable τ, let $\bar{\tau}$ denote the complement of τ. Then,

$$P(SD|F) = \prod_{e_{ij} \in E_T} \Big[\; \bar{\delta}_i \bar{\delta}_j \bar{a}_{ij} \; + \; \bar{\delta}_i \delta_j (\, a_{ij} p_{ij} + \bar{a}_{ij} (1 - p_{ij}) \,) \Big.$$

$$\Big. + \; \delta_i \bar{\delta}_j (\, \bar{a}_{ij} r_{ij} + a_{ij} (1 - r_{ij}) \,) \; + \; \delta_i \delta_j (\, a_{ij} s_{ij} + \bar{a}_{ij} (1 - s_{ij}) \,) \; \Big] . \tag{3.3}$$

Note that only one of the additive terms within the brackets will be non-zero for each edge $e_{ij} \in E_T$. The following example illustrates the calculation of the probabilities described.

Example 3.1: $D_{\delta t}$ design with $\delta = 1$ and $t = 2$ (Fig. 3.1). A system S belongs to a design $D_{\delta t}$ when $e_{ij} \in E_T$ if and only if $j - i = \delta m$ modulo N, where $N =$ the number of nodes in S and $m = 1, 2, \cdots, t$ [52]. Let $F^0 = \{u_0 , u_1\}$ and SD^0 be the syndrome shown in Fig. 3.1. Then,

$$P(F^0) \;=\; f_0 f_1 (1 - f_2) (1 - f_3) (1 - f_4)$$

$$P(SD^0|F^0) \;=\; (1 - s_{01})\, r_{02}\, (1 - r_{12})\, r_{13}\, p_{30}\, (1 - p_{40})\, p_{41}$$

$$P(F^0|SD^0) = \frac{P(SD^0|F^0)\,P(F^0)}{\sum_{F' \subseteq V_T} P(SD^0|F')\,P(F')} .$$

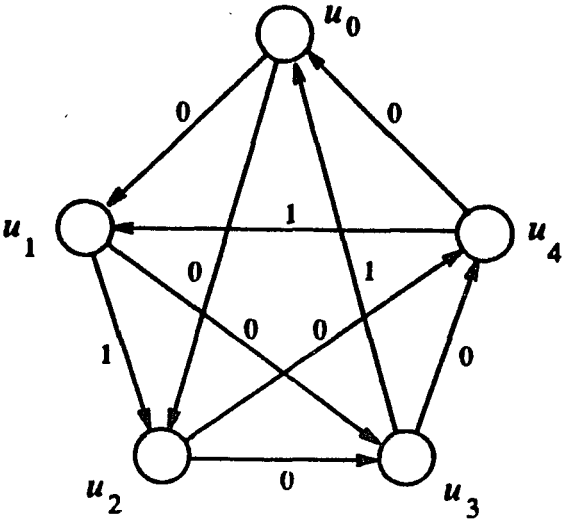


Figure 3.1: D_{12} testing graph of Example 3.1.

By enumerating all possible fault sets and calculating their posterior fault probability values, the most probable fault set F^* can be directly determined. However, a large subset of 2^{V_T} can be removed from consideration altogether by the following two properties that result from the definition of a consistent diagnosis and Equations (3.1) — (3.3).

Property 1: Every superset of a consistent diagnosis is a consistent diagnosis.

Property 2: If F is an inconsistent diagnosis and $P(SD) > 0$, then $P(F|SD) = 0$.

A straightforward diagnosis algorithm would be to calculate $P(F|SD)$ for all consistent diagnoses $F \subseteq V_T$ and identify the fault set F^* with the maximum posterior probability. The probability calculations in this method are essentially the same as in Blount's diagnosis method [7]. However, this method is $O(2^N)$ as opposed to $O(2^{N+|E|})$ for Blount's method. In addition, this algorithm produces complete probabilistic information for every possible con-

sistent diagnosis or fault set. This information can be used to not only obtain the most probable diagnosis but also to determine the effect of including any other node into the postulated fault set. Thus, if a subset of nodes $A \subseteq V_T$ is replaced, then $\sum_{F \subseteq A} P(F|SD)$ is the probability that all of the faulty nodes have been replaced.

Table 3.1 shows the result of applying this algorithm to the D_{12} graph of Example 3.1 with $f_i = 0.4$, $p_{ij} = 0.9$, and $r_{ij} = s_{ij} = 0.5$ for all nodes $u_i \in V_T$ and edges $e_{ij} \in E_T$. Note that given two consistent fault sets A and B such that $A \subset B$, it is still possible for $P(B|SD)$ to be greater than $P(A|SD)$ (compare 5-th and 6-th fault sets in Table 3.1). This is contrary to the minimality criterion used by many other authors, e.g. [9], to determine the most likely diagnosis given a syndrome. Since this complete enumeration diagnosis algorithm is $O(2^N)$, it is not suitable for large systems. (Recall however that Blount's method [7] is worse than this algorithm by a factor of $2^{|E|}$.) In Section 3.4, a more efficient and practical diagnosis method is described.

Rank	Probability	Fault Set
1	0.5962	{ 0 , 1 }
2	0.0994	{ 0 , 1 , 2 }
3	0.0994	{ 0 , 2 , 4 }
4	0.0663	{ 1 , 3 }
5	0.0341	{ 0 , 1 , 2 , 3 , 4 }
6	0.0184	{ 0 , 1 , 2 , 4 }
7	0.0184	{ 0 , 2 , 3 , 4 }
8	0.0184	{ 0 , 1 , 2 , 3 }
9	0.0110	{ 2 , 3 , 4 }
10	0.0110	{ 0 , 1 , 3 }
11	0.0110	{ 0 , 1 , 4 }
12	0.0110	{ 1 , 2 , 3 }
13	0.0020	{ 0 , 1 , 3 , 4 }
14	0.0020	{ 1 , 2 , 3 , 4 }
15	0.0012	{ 1 , 3 , 4 }

Table 3.1: Posterior probabilities for Example 3.1.

If one only wishes to find the most probable fault set given a syndrome, then it is possible to search significantly fewer fault sets than in the case where exact posterior probability values must be determined. However, a polynomial time algorithm probably does not exist for even this simpler problem because it is an NP-hard problem. This is formalized in the following theorem.

Theorem 3.1: The problem of finding the most probable fault set given a fault syndrome is NP-hard.

Proof: A fault set of size k $(1 \leq k \leq N)$ can be made to be less probable than any fault set of size l $(k < l \leq N)$ simply by making $f_i = \varepsilon > 0$ sufficiently small for all nodes $u_i \in V_T$. It is always possible to obtain a sufficiently small ε since $\Gamma(u_i)$ is finite for all $u_i \in V_T$ and all probability values range from 0 to 1. Then, the most probable diagnosis will always be a consistent fault set of minimum size. However, obtaining a minimum size consistent fault set is equivalent to obtaining a minimum size vertex cover of the *one-condensation* of the testing graph (refer to Section 3.5). Thus, since obtaining a minimum size vertex cover is a known NP-hard problem, the problem of finding the most probable fault set given a syndrome is NP-hard. **Q.E.D.**

Due to this NP-hardness result, the best that can be done is to produce a good expected-case algorithm. An alternative solution is to use a good heuristic algorithm to produce a diagnosis F_D. The latter solution is the approach taken in most previous work on probabilistic diagnosis. In this case, since F_D will not be guaranteed to be the most probable diagnosis, the percentage of correct diagnoses will be lower. In the next section, a more efficient method of finding the most probable diagnosis is described.

3.1. Most Probable Diagnosis Algorithm

The problem of finding the most probable fault set F^* can be viewed as a search problem. The search space is the fault set lattice shown in Fig. 3.2 with $\varnothing$ at the bottom and V_T at the top of the lattice.

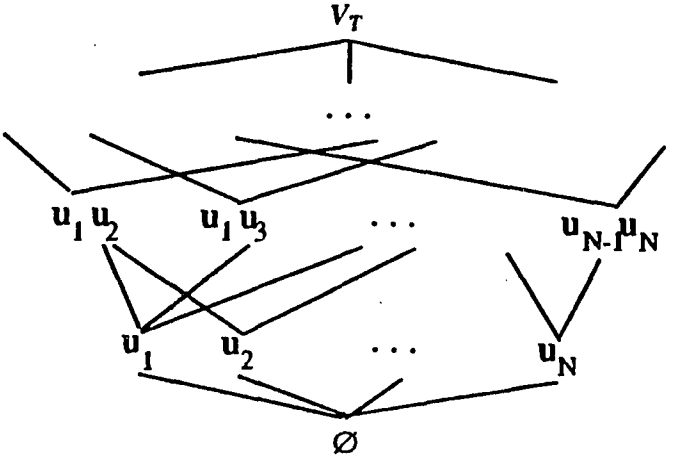


Figure 3.2: Fault set lattice.

It is desirable to conduct the search in such a manner that the fewest possible number of fault sets are searched before concluding that one of the fault sets searched is F^*. The most significant contribution of this chapter is the discovery of methods (referred to as heuristics) which are found to make most probable diagnosis practical in large testing graphs. Although many heuristics were found for limiting the number of fault sets which must be searched in order to find F^*, experiments showed that only some of the heuristics were useful in reducing total diagnosis time.

3.1.1. Useful Heuristics

Heuristics which were found to be useful in reducing total diagnosis time were of two types. The first type limited the sizes of possible fault sets. The second type could be used in

a preprocessing stage to determine those nodes which were definitely members of F^* and those nodes which were definitely not members of F^*.

In a large testing graph, the 1-links will tend to be sparse and widely distributed. Given a testing graph $G = (V_T, E_T)$ and a syndrome SD, let us extract the digraph $G_1 = (V_1, E_1)$ such that $e_{ij} \in E_1 \longleftrightarrow e_{ij} \in E_T$ and $e_{ij} = 1$. Also, $u_i \in V_1 \longleftrightarrow e_{kl} \in E_1$ such that $i = k$ or $i = l$. G_1 is called the *one-condensation* of G. Clearly, every fault set in MCD, the set of minimal consistent diagnoses, will consist of members of V_1. Based on Property 1, it is known that every consistent diagnosis is either a member of MCD or a superset of a member of MCD. Heuristic #1 is to count the number of connected components (or simply components) of G_1. Let σ denote the number of components of G_1. σ is a crude lower bound on the size of the smallest consistent fault set. Assuming that fault sets are searched from smaller fault sets to larger fault sets, the search can start from all subsets of V_1 of size σ.

Heuristic #2 is to impose a static upper bound t' on the size of possible fault sets. Since f_i values will tend to be small, F^* will tend to be near the bottom of the fault set lattice. Thus, a method of constraining the search from the top is extremely useful in reducing total search time. Also, with large numbers of faulty nodes, the syndrome produced is more arbitrary because the test evaluation results of more faulty nodes are used. With this heuristic, our algorithm will only guarantee the most probable diagnosis if the number of faulty nodes is less than t'.

Dynamic Upper Bound

In many cases, it is found that a dynamic upper bound on the size of possible fault sets can be used which is smaller than t'. Heuristic #3 is to determine a dynamic upper bound $\hat{t}$ on the maximum size of F^*.

Some preliminaries are required. Let pc_i be the probability contribution of u_i and its outward-bound testing links, e.g., $pc_i = (1 - f_i)\, p_{ij}$ given $u_i \in V_T - F$, $u_j \in F$, $\Gamma(u_i) = \{u_j\}$, and $a_{ij} = 1$. Then,

$$P(F, SD) = P(SD|F)\, P(F) = \prod_{u_i \in V_T} pc_i \ .$$

$P(F, SD)$ is the "unnormalized" posterior probability of F. Since only unnormalized posterior probabilities need to be compared when searching for the most probable diagnosis, $P(F, SD)$ values will be referred to simply as probability values.

To dynamically determine the upper bound on the sizes of the consistent fault sets which must be considered, let us first consider an upper bound on the maximum probability value that any fault set of size t or greater can attain. Let us sort $\{f_i : 1 \le i \le n\}$ in decreasing order. If there exists a positive integer k such that $f_k > 0.5$ and $f_{k+1} \le 0.5$, then let $M = \max\{k, t\}$; otherwise, let $M = t$. Then $\hat{U}_t = \prod_{i=1}^{M} f_i \prod_{i=M+1}^{N} (1 - f_i)$ is a definite upper bound. Since f_i values will typically be very small in large systems, $\hat{U}_t$ should suffice for most situations.

Nevertheless, it is possible to obtain a smaller upper bound $\hat{P}_t$ by considering the maximum probability contribution of each node in its faulty and fault-free state. For each $u_i \in V_T$, let $\hat{p}_i^{\,1} = f_i \prod_{u_k \in \Gamma(u_i)} \left[a_{ik} \hat{s}_{ik} + \bar{a}_{ik} \hat{f}_{ik} \right]$, $\hat{p}_i^{\,0} = (1 - f_i) \prod_{u_k \in \Gamma(u_i)} \left[a_{ik} p_{ik} + \bar{a}_{ik} \right]$, and $ratio_i = \hat{p}_i^{\,0} / \hat{p}_i^{\,1}$. Let us sort $\hat{p}_i^{\,1}$, $\hat{p}_i^{\,0}$, and $ratio_i$ in increasing order of $ratio_i$ (this means that when $ratio_i$ is swapped with $ratio_j$ during the sorting process, $\hat{p}_i^{\,1}$ and $\hat{p}_i^{\,0}$ are also swapped with $\hat{p}_j^{\,1}$ and $\hat{p}_j^{\,0}$, respectively). Let $ratio_{N+1} = 2.0$. If there exists a positive integer α such that $ratio_\alpha < 1$ and $ratio_{\alpha+1} \ge 1$, then let $M = \max\{\alpha, t\}$; otherwise, let $M = t$. Then, $\hat{P}_t = \prod_{i=1}^{M} \hat{p}_i^{\,1} \prod_{i=M+1}^{N} \hat{p}_i^{\,0}$.

Although the method described above may seem rather complex, it is an intuitively simple method. $\hat{p}_i^{\,1}$ and $\hat{p}_i^{\,0}$ are simply upper bounds on pc_i given that $u_i \in F$ and $u_i \in V_T - F$, respectively. Since a node must either be faulty or fault-free, one of $\hat{p}_i^{\,1}$ and $\hat{p}_i^{\,0}$ must be included in $\hat{P}_t$ for each node u_i. Finally, determining whether to postulate u_i as faulty or fault-free for purposes of calculating $\hat{P}_t$ is dependent on which option results in a larger value of $\hat{P}_t$. The results of numerous experiments show that $\hat{P}_t$ is a fairly tight upper bound. This is due to the fact that all of the available syndrome information is used in the calculation of the upper bound.

Theorem 3.2: $\hat{P}_t$ is an upper bound on the maximum probability value that any fault set of size t or greater can attain.

Proof: Let us assume that the process described earlier has been used to calculate $\hat{P}_t$ but that there exists a syndrome SD in which the probability value of a fault set of size $\geq t$ is greater than $\hat{P}_t$. Let $P(F, SD)$ be this probability value and let pc_i values be defined as above. Sort the values pc_i in decreasing order.

Each pc_i $(1 \leq i \leq N)$ has a corresponding $u_k \in V_T$ upon which its calculation is based. An analogous statement can be made for each $\hat{p}_i^{\,1}$ and $\hat{p}_i^{\,0}$. Let $\beta = \hat{P}_t$. For each i $(1 \leq i \leq M)$, let j be such that pc_j and $\hat{p}_i^{\,1}$ correspond to the same node $u_k \in V_T$. For each such u_k found, if $u_k \in V_T - F$, let $\beta = \beta \dfrac{\hat{p}_i^{\,0}}{\hat{p}_i^{\,1}}$. For each x $(M+1 \leq x \leq N)$, let y be such that pc_y and $\hat{p}_x^{\,0}$ correspond to the same node $u_z \in V_T$. For each such u_z found, if $u_z \in F$, let $\beta = \beta \dfrac{\hat{p}_x^{\,1}}{\hat{p}_x^{\,0}}$. Let us consider the following three cases.

Case 1: $M = |F|$

In this case, all factors in β are such that $\hat{p}_i^{\,1}$ corresponds to a node $u_j \in F$ and $\hat{p}_i^{\,0}$ corresponds to a node $u_k \in V_T - F$. Thus, $P(F, SD) \leq \beta$. Also, for each $\hat{p}_i^{\,1}$ that

was multiplied out from $\hat{P}_t$, there exists a $\hat{p}_j{}^0$ which was also multiplied out. For each such pair,

$$\frac{\hat{p}_i{}^0}{\hat{p}_i{}^1} \cdot \frac{\hat{p}_j{}^1}{\hat{p}_j{}^0} \leq 1$$

since the first ratio is $\leq$ the inverse of the second ratio (because $i \leq M < j$). Therefore, $\beta \leq \hat{P}_t$ and it follows that $P(F, SD) \leq \beta \leq \hat{P}_t$.

Case 2: $M > |F|$

In this case, there exist factors $\hat{p}_i{}^1$ in β such that $\hat{p}_i{}^1$ corresponds to a node $u_j \in V_T - F$. However, there do not exist factors $\hat{p}_i{}^0$ in β such that $\hat{p}_i{}^0$ corresponds to a node $u_j \in F$. Also, note that by the same argument as in case 1, $\beta \leq \hat{P}_t$. Let $\beta' = \beta$. For each $\hat{p}_i{}^1$ that corresponds to a node $u_j \in V_T - F$, let $\beta' = \beta' \frac{\hat{p}_i{}^0}{\hat{p}_i{}^1}$. Since $M > |F| \geq t$, this multiplicative factor is < 1 and $\beta' < \beta$. Also, $P(F, SD) \leq \beta'$. Therefore, $P(F, SD) \leq \beta' < \beta \leq \hat{P}_t$.

Case 3: $M < |F|$

In this case, there exist factors $\hat{p}_i{}^0$ in β such that $\hat{p}_i{}^0$ corresponds to a node $u_j \in F$. However, there do not exist factors $\hat{p}_i{}^1$ in β such that $\hat{p}_i{}^1$ corresponds to a node $u_j \in V_T - F$. Also, note that by the same argument as in case 1, $\beta \leq \hat{P}_t$. Let $\beta' = \beta$. For each $\hat{p}_i{}^0$ that corresponds to a node $u_j \in F$, let $\beta' = \beta' \frac{\hat{p}_i{}^1}{\hat{p}_i{}^0}$. Since $t \leq M < |F|$, this multiplicative factor is ≤ 1 and $\beta' \leq \beta$. Also, $P(F, SD) \leq \beta'$. Therefore, $P(F, SD) \leq \beta' \leq \beta \leq \hat{P}_t$. Thus, in all cases, $P(F, SD) \leq \hat{P}_t$. However, this contradicts the assumption that $P(F, SD) > \hat{P}_t$. **Q.E.D.**

The dynamic upper bound $\hat{t}$ can be determined as follows. Recall that the search is conducted from smaller fault set sizes to larger fault set sizes. Before any fault sets of size t are

searched, the maximum $P(F, SD)$ value obtained for all $F \subseteq V_T$ $(|F| < t)$ is compared against $\hat{P}_t$. At any time, if the largest $P(F, SD)$ value is greater than $\hat{P}_t$, then the search can be stopped and $\hat{t} = t - 1$.

Preprocessing Heuristics

In many cases, with some simple preprocessing, it is possible to determine that certain nodes are definitely members of F^* and that certain nodes are definitely not members of F^*.

The first preprocessing heuristic uses the fact that a static upper bound t' has been assumed on the sizes of possible fault sets. For a bound-size fault set model, Dahbura and Masson [16] introduced the following theorem.

Theorem 3.3: [16] For a digraph $G(V_T, E_T)$ and a syndrome produced by a fault set containing at most t' faulty nodes, a node u_i is diagnosable as faulty if and only if

$$|\Gamma_1(u_i) \bigcup \Gamma_1^{-1}(u_i)| + vc(V_T - \Gamma_1(u_i) - \Gamma_1^{-1}(u_i) - u_i) > t' .$$

The quantity $vc(D)$, $D \subseteq V_T$, is defined as the size of the minimum vertex cover set of the subgraph with D as its vertex set and the 1-links of G between members of D as its edge set.

This theorem is not directly implementable because determining the minimum size vertex cover set of a general graph is an NP-hard problem. Thus, as an approximation, the following condition is used to determine if u_i can be diagnosed to be faulty:

$C1$: $$|\Gamma_1(u_i) \bigcup \Gamma_1^{-1}(u_i)| + \sigma - 1 > t' .$$

Condition $C1$ is valid as $\sigma - 1$ is a lower bound on the size of the minimum size vertex cover set of the remainder of G_1. Heuristic #4 applies $C1$ to all $u_i \in V_1$ to find the nodes which are definitely faulty, assuming that the number of faulty nodes is not more than t'.

Heuristic #5 applies a check on the probability parameters to quickly determine the fault status of some of the nodes. Each node $u_i \in V_T$ must be either faulty or fault-free. Let c_i be

the probability contribution of u_i and its outward and inward-bound testing links. Let $\hat{c}_i^{\,1}$ and $\underline{c}_i^{\,1}$ ($\hat{c}_i^{\,0}$ and $\underline{c}_i^{\,0}$) be upper and lower bounds on c_i given that u_i is faulty (fault-free). For each $e_{ij} \in E_T$, let $\hat{r}_{ij} = \max\{r_{ij}\ ,\ (1 - r_{ij})\}$, $\hat{s}_{ij} = \max\{s_{ij}\ ,\ (1 - s_{ij})\}$, $\underline{r}_{ij} = \min\{r_{ij}\ ,\ (1 - r_{ij})\}$, $\underline{s}_{ij} = \min\{s_{ij}\ ,\ (1 - s_{ij})\}$, $\hat{\rho}_{ij} = \max\{p_{ij}\ ,\ s_{ij}\}$, $\hat{\omega}_{ij} = \max\{(1 - p_{ij})\ ,\ (1 - s_{ij})\}$, $\underline{\rho}_{ij} = \min\{p_{ij}\ ,\ s_{ij}\}$, and $\underline{\omega}_{ij} = \min\{(1 - p_{ij})\ ,\ (1 - s_{ij})\}$. Then $\hat{c}_i^{\,1}$, $\hat{c}_i^{\,0}$, $\underline{c}_i^{\,1}$, and $\underline{c}_i^{\,0}$ can be calculated as follows:

$$\hat{c}_i^{\,1} = f_i \prod_{u_k \in \Gamma(u_i)} \left[a_{ik}\hat{s}_{ik} + \bar{a}_{ik}\hat{r}_{ik} \right] \prod_{u_k \in \Gamma^{-1}(u_i)} \left[a_{ki}\hat{\rho}_{ki} + \bar{a}_{ki}\hat{\omega}_{ki} \right], \tag{3.4}$$

$$\hat{c}_i^{\,0} = (1 - f_i) \prod_{u_k \in \Gamma(u_i)} \left[a_{ik}p_{ik} + \bar{a}_{ik} \right] \prod_{u_k \in \Gamma^{-1}(u_i)} \left[a_{ki}\ (1 - r_{ki}) + \bar{a}_{ki} \right]. \tag{3.5}$$

$$\underline{c}_i^{\,1} = f_i \prod_{u_k \in \Gamma(u_i)} \left[a_{ik}\underline{s}_{ik} + \bar{a}_{ik}\underline{r}_{ik} \right] \prod_{u_k \in \Gamma^{-1}(u_i)} \left[a_{ik}\underline{\rho}_{ik} + \bar{a}_{ik}\underline{\omega}_{ik} \right], \tag{3.6}$$

$$\underline{c}_i^{\,0} = (1 - f_i) \prod_{u_k \in \Gamma(u_i)} \bar{a}_{ik}\ (1 - p_{ik}) \prod_{u_k \in \Gamma^{-1}(u_i)} \bar{a}_{ik}r_{ik}\ . \tag{3.7}$$

If a subset of nodes $A \subseteq V_T$ are known to be faulty and a subset of nodes $B \subseteq V_T$ are known to be fault-free, then the parameters $\hat{p}_i^{\,1}$, $\hat{p}_i^{\,0}$, $\hat{c}_i^{\,1}$, $\hat{c}_i^{\,0}$, $\underline{c}_i^{\,1}$, and $\underline{c}_i^{\,0}$ must be modified to take this new knowledge into account. This is a straightforward modification of the Eqs. (3.4) — (3.7).

Heuristic #5 is based on iteratively using Eqs. (3.4) — (3.7) to determine the maximum and minimum probability contributions of each node in its faulty and fault-free state. For each node $u_i \in V_T$, if $\underline{c}_i^{\,0} > \hat{c}_i^{\,1}$, then u_i can be diagnosed to be fault-free. Likewise, if $\underline{c}_i^{\,1} > \hat{c}_i^{\,0}$, then u_i can be diagnosed to be faulty. In our numerical experiments, this heuristic was found to be most useful in eliminating from consideration most of those nodes in $V_T - V_1$. An assumption made by many previous authors is that no node is suspected of being faulty if it is not evaluated to be faulty by another node *and* it does not evaluate another node to be faulty [5, 19]. However, this assumption is not always justified. Suppose that it has somehow been

determined that a node u_k is faulty with very high probability. Now, if a node u_i not in V_1 evaluates u_k to be fault-free with very high confidence (high p_{ik} value), this makes u_i a very suspicious node. The condition check $\underline{c_i}^0 > \hat{c}_i^1$ is meant to ensure that this special case does not apply to the node u_i.

Summary and Remarks

The above five heuristics were found to be easily implementable and useful in reducing total search time. Heuristic #1, counting the number of components in G_1, was used to determine a starting point for the diagnosis search. Heuristic #2, imposing a static upper bound t' on the size of possible fault sets, was necessary to keep the computation time down in worst-case type situations. Heuristics #1 and #2 were also used to aid in the implementation of Heuristic #4. Heuristic #3, dynamically determining an upper bound on the size of F^*, was found to be a very useful heuristic which frequently allowed the search to stop before fault set size t' was reached. Heuristic #4, which applies Condition $C1$ to quickly determine faulty nodes, was found to be especially useful when the number of faulty nodes was close to t' and σ was large. Finally, Heuristic #5, applying a check on the probability parameters to quickly determine faulty and fault-free nodes, was useful in determining many definitely fault-free nodes when low fault parameter values were used.

Besides the above five heuristics, there are several other heuristics which could reduce the number of fault sets to be searched. Heuristic #6 is based on an idea due to [51] and uses the observation that certain inconsistent fault sets larger than σ need not be searched at all. Suppose A is an inconsistent fault set. Then, when searching supersets of A of size $|A| + 1$, it is only necessary to look at sets of the form $A \cup \{u_k\}$, where $u_k \in V_1$ and u_k covers at least one 1-link that A does not cover. A is an inconsistent fault set because it does not cover all of the 1-links in the syndrome. Thus, any superset of A which does not cover more 1-

links than A will also be inconsistent.

Heuristic #7 is based on searching within each component of G_1 independently of the other components of G_1. Let T_1 be the number of vertices in the largest component of G_1. *MCD* can be constructed with an algorithm of computational complexity $O(T_1^{\sigma} + 2^{T_1})$ since *MCD* can be constructed using the minimal size vertex covers of the components of G_1.

Heuristics #6 and #7 were found to entail a significant amount of bookkeeping. For all cases attempted in our experiments, it was found that the use of these two heuristics increased the diagnosis time required. Although Heuristic #6 might look promising at first and was the basis for guiding the diagnosis search in [51], because so few fault set sizes were searched using our method with Heuristics #1 through #5, the extra time required to generate the fault sets nullified the savings of generating fewer fault sets using Heuristic #6.

3.4.2. Description of Most Probable Diagnosis (MPD) Algorithm

The most probable diagnosis (MPD) algorithm is described below. The only assumption made is that the fault parameters for each vertex and edge in the testing graph are independent of the fault parameters for all other vertices and edges.

Algorithm MPD:

0: Given a fault syndrome SD and a testing graph $G = (V_T, E_T)$,
1: Let $G_1 = (V_1, E_1)$ be the one-condensation of G, and $V_0 \leftarrow V_T - V_1$;
2. Perform preprocessing (Heuristics #4 and #5) to determine nodes which
   are definitely faulty (set A) and nodes which are definitely fault-free (set B);
3: Let $min\_size = \sigma$ (number of components of G_1);
4: Let $V_0 \leftarrow V_0 - B$;
5: Calculate $\hat{p}_i^{\,1}$ and $\hat{p}_i^{\,0}$ for each element $u_i \in V_T$;
6: Let $sz \leftarrow min\_size$, $\hat{P}_{min\_size} \leftarrow 1.0$, $\hat{P}_{N+1} \leftarrow 0.0$, $f\_set \leftarrow \emptyset$, $max\_prob \leftarrow 0.0$,
   and $max\_set \leftarrow \emptyset$;
7: **While** $max\_prob < \hat{P}_{sz}$ and $sz \leq t'$ **do**
   begin
      for each consistent set $D \subseteq V_1$ with $D \supseteq A$ and $|D| = sz$ **do**
         begin
            $f\_set \leftarrow f\_set \cup D$;
            if $P(D, SD) > max\_prob$ **then**

```
                begin
                   max_prob ← P(D, SD);
                   max_set ← D;
                endif
            endfor;
         sz ← sz + 1;
      endwhile,
8:  Let sz ← min_size + 1;
9:  While max_prob < P̂_sz and sz ≤ t' do
      begin
         for each D = F ∪ E_T ⊇ (F ∈ f_set, E_T ∈ V_0, |D| = sz), do
            if P(D, SD) > max_prob then
               begin
                  max_prob ← P(D, SD);
                  max_set ← D;
               endif;
         sz ← sz + 1;
      endwhile
10: Return max_set.
```

Theorem 3.4: Given that the syndrome SD is produced from a set of faulty nodes of size not more than t', Algorithm MPD returns the most probable diagnosis given the syndrome.

Proof: From Theorem 3.3, it is known that Heuristic #4 is a valid preprocessing heuristic. Also, from the definitions of $\hat{c}_i^{\,1}$, $\hat{c}_i^{\,0}$, $\underline{c}_i^{\,1}$, and $\underline{c}_i^{\,0}$, Heuristic #5 is a valid preprocessing heuristic. By Properties 1 and 2, σ is a valid lower bound on the fault set lattice that must be searched. Since $\hat{P}_t$ is a valid upper bound on the maximum probability that any fault set of size t or greater can attain, using the condition $max\_prob < \hat{P}_{sz}$ to consider searching sets of size sz or greater is valid. Thus, the minimum of t' and $\hat{t}$ is a valid upper bound on the fault set lattice that must be searched. Since all consistent fault sets between the lower bound and upper bound are searched by the MPD algorithm, the most probable fault set must be found during the search. **Q.E.D.**

The MPD algorithm is conceptually a very simple algorithm. It uses σ as the lower bound, the minimum of $\hat{t}$ and t' as its upper bound, and searches all of the consistent fault

sets between the lower and upper bounds in the fault set lattice. In searching the fault sets, those nodes whose statuses are already determined through preprocessing (Heuristics #4 and #5) are removed from consideration.

3.5. Simulation Results

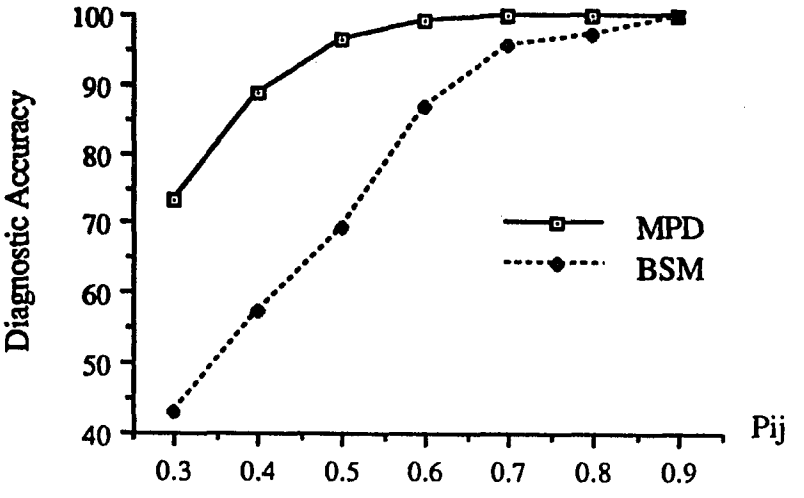
Numerous simulations were run with the BSM and MPD algorithms on several types of testing graphs. Since similar comparative results were obtained in all cases, only a representative set of results are presented in this section. In evaluating diagnosis algorithms, certain values were assumed for the probability parameter values and the set of faulty nodes and syndromes were generated randomly using these probability parameters. 1000 diagnoses were performed to evaluate each diagnosis algorithm with each set of probability parameter values.

In order to demonstrate the practical use of the MPD algorithm on a large system, BSM and MPD were executed on a Q_8, a 256-node hypercube of dimension 8. Each node in the hypercube was assigned to test 4 of its neighbors and to be tested by 4 of its neighbors in a symmetric fashion. Fig. 3.3(a) shows the simulation results for the MPD and BSM algorithms with $f_i = 0.01$ and p_{ij} values ranging from 0.3 to 0.9. Comparison-testing was used in these simulations, resulting in $r_{ij} = 1 - p_{ij}$ and $s_{ij} = 1 - (1 - p_{ij})^2$ [19]. Experiments were also conducted to determine the effect of inaccurate probability parameter values on the diagnostic accuracy of the MPD and BSM algorithms. Although the same probability parameter values as in the previous simulations were used by the diagnosis algorithms, an error of -30% in each probability parameter value was introduced in generating the set of faulty nodes and the fault syndromes. Fig. 3.3(b) shows the results of these simulations for a Q_8.

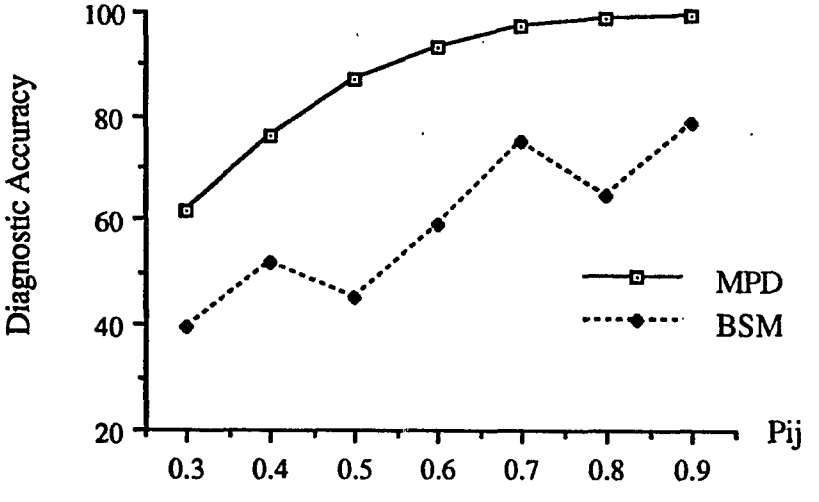
Tables 3.2 and 3.3 show the results for the MPD algorithm with $f_i = 0.01$, $r_{ij} = s_{ij} = 0.5$, and $p_{ij} = 0.6$. Table 3.2 shows the simulation results for 1000 syndromes randomly generated assuming the above probability parameter values. Table 3.3 shows the simu-

lation results for 1400 iterations with the number of faulty nodes chosen to produce useful diagnostic accuracy statistics. A static upper bound of $t' = 7$ was used. All CPU times are given in average CPU milliseconds on a Sun 3/280 running SunOS 3.5.

The hypercube Q_8 was found to be an excellent interconnection structure for diagnostic purposes. Even with $p_{ij} = 0.6$, very accurate diagnoses were possible, as demonstrated by Fig. 3.3 and Table 3.2. From Fig. 3.3(a), it can be seen that the MPD algorithm produces significantly better results than the BSM algorithm when $p_{ij} < 0.9$. The MPD algorithm produces good results when $p_{ij} \geq 0.5$ even with inaccurate probability parameter values. The reason for the jagged diagnostic accuracy curve of the BSM algorithm in Fig. 3.3(b) is that the BSM algorithm is based on the use of a threshold on the number of one-links incident on each node. With inaccurate probability parameter values, an incorrect threshold may be chosen, resulting in lower diagnostic accuracy.



(a)



(b)

Figure 3.3: Simulation results of MPD and BSM on a Q_8 with $f_i = 0.01$ assuming (a) no probability parameter errors and (b) 30% parameter errors.

# Faults	Avg CPU Time	# Instances	# Correct	% Correct
1	2101	212	210	99.06
2	2131	270	270	100.00
3	2202	221	214	96.83
4	2458	174	167	95.98
5	2537	72	68	94.44
6	6809	31	29	93.55
7	2119	16	16	100.00
8	849	4	0	0.00
Cumulative	2367	1000	974	97.40

Table 3.2: Results of Algorithm MPD on a Q_8 with $f_i = 0.01$, $r_{ij} = s_{ij} = 0.5$, and $p_{ij} = 0.6$.

# Faults	Avg CPU Time	# Instances	# Correct	% Correct
1	2067	200	198	99.00
2	2136	200	197	98.50
3	2155	200	196	98.00
4	2374	200	194	97.00
5	3313	200	188	94.00
6	4601	200	193	96.50
7	9706	200	186	93.00
Cumulative	3765	1400	1352	96.57

Table 3.3: Statistical results of Algorithm MPD on a Q_8 with parameter values of Table 3.2.

In the MPD algorithm, a rather surprising result was that the processing time required was fairly uniform regardless of the number of faulty nodes. This is explained by two offsetting factors. With a smaller number of faulty nodes, the diagnosis is easier and the dynamic upper bound $\hat{t}$ is quickly reached, resulting in fast diagnosis. With a larger number of faulty nodes, the diagnosis is more difficult but σ tends to be larger, making Heuristic #4 more useful in quickly diagnosing some of the faulty nodes and thus making the rest of the diagnosis easier.

Ideally, the static upper bound t' should be chosen based on simulation results that show that diagnostic accuracy falls to unacceptable levels when the number of faulty nodes is more than t'. In order to use such simulation results, the simulation should produce the same number of instances of each fault set size, instead of using the f_i values to produce the set of

faulty nodes. This type of simulation result is shown in Table 3.3 for the situation in Table 3.2. Note that with 7 faulty nodes, the diagnostic accuracy has fallen to 93% from a diagnostic accuracy of 99% for 1 faulty node, thus justifying the choice of $t' = 7$. With a larger t' value, Heuristic #4 becomes less useful, thus making the diagnosis in all situations more difficult and time-consuming. Thus, another consideration in choosing t' is the desired average diagnosis time. Finally, t' can be chosen based on the distribution of the number of faulty nodes.

After a diagnosis is done, it is desirable to be able to obtain an estimate of the accuracy of the diagnosis. $P(F^*|SD)$ is the desired accuracy value. However, if a Bayesian analysis is used, then even if F^* is known, all of the consistent fault sets must still be searched in order to determine the exact value of $P(F^*|SD)$. This is because every consistent fault set can potentially contribute a positive amount to the calculation of $P(SD)$, which is necessary to calculate $P(F^*|SD)$. Peng and Reggia [51] describe a systematic method of making a lower bound estimate of $P(F^*|SD)$ without searching all of the consistent fault sets. However, their method, when applied to the multiprocessor diagnosis addressed in this chapter, results in extremely poor lower bound estimates because so few consistent fault sets are searched by our diagnosis method. Thus, a much more practical method is to generate simulation results such as those in Table 3.3 to estimate the accuracy of a diagnosis based on factors such as $|F^*|$. For example, for the situation in Table 3.2, if a diagnosis implicating 3 faulty nodes is found, an estimate of the diagnostic accuracy is 98%.

3.6. Conclusion

Probabilistic diagnosis methods can be used as part of an incremental diagnosis strategy. The choice of which diagnosis method to use will be dependent on the accuracy of diagnosis required and the time available for diagnosis. To achieve the maximum level of diagnostic

accuracy, a diagnosis method that finds the most probable diagnosis given the syndrome must be used. This chapter has presented the MPD algorithm, which guarantees that the most probable diagnosis is found (given enough time and a sufficiently large static upper bound) and is suitable for systems with up to several hundred nodes. This diagnosis algorithm should be used in conjunction with a fast probabilistic algorithm such as one of the algorithms in Chapter 4 in order to produce "good" diagnoses in those fault situations where the MPD algorithm is unable to produce the most probable diagnosis (because there are too many faulty nodes — resulting in the static upper bound being reached). Our simulation results indicate that the MPD algorithm can produce significantly better diagnosis than the BSM algorithm, which can produce asymptotically correct diagnosis. Good diagnostic accuracy is achievable with the MPD algorithm even when inaccurate probability parameter values are used. The main contribution in this chapter is the discovery of methods which are found to make most probable diagnosis practical in large testing graphs.

CHAPTER 4

OPTIMAL SINGLE SYNDROME DIAGNOSIS

This chapter presents optimal single syndrome diagnosis algorithms for the categories of diagnosis in which a restricted form of the syndrome information is used. Probability analysis is used to derive optimal diagnosis algorithms for the categories 2, 2A, 3, and 3A defined in Chapter 2. By analyzing the behavior of the optimal diagnosis algorithms, it is shown that the algorithms work well even when probability parameters are inaccurate or unknown. Probability analysis also results in new bounds on the required number of tests for which asymptotically correct diagnosis is possible as the system size grows to infinity. Finally, simulations and probability analyses are used to compare the performance of the optimal algorithms with several other previous probabilistic diagnosis algorithms and with each other.

4.1. Introduction

In addressing the diagnosis of multiprocessor and multicomputer systems, most of the system-level diagnosis techniques based on the PMC model [52] are insufficient because of the objectives stated in Chapter 1. Several authors [48,49,65] have addressed the problem of diagnosis of intermittent faults in t_i–diagnosable systems, in which if no more than t_i nodes are intermittently faulty, a non-faulty node will never be diagnosed as faulty [48]. However, because a node is identified as faulty only if there is sufficient evidence to definitely identify it as faulty given the upper bound t_i on the number of faulty nodes, these methods rarely achieve correct diagnosis (according to our definition in Chapter 2). Other authors [5,19]

have presented fast probabilistic diagnosis algorithms which achieve correct diagnosis with high probability given intermittently faulty nodes. The method presented by Rangarajan and Fussell [30] degenerates to Blough *et. al.*'s method when only one syndrome is used in the diagnosis.

In addition to the fact that optimal diagnosis using all of the fault syndrome information is NP-hard, it is desirable to restrict the amount of fault syndrome information used in the diagnosis because of the communication requirements. Consider an algorithm which is executed at each node and makes a diagnosis using all of the fault syndrome information. This implies that the testing results of every non-faulty node must be reliably communicated to every other non-faulty node. In the presence of intermittently faulty nodes, several identical copies of the message from any node to any other node must be sent along disjoint paths to guarantee reliable communication. Thus, the communication requirements of this reliable all-to-all broadcast operation are immense. Because of this, it is desirable to be able to perform the diagnosis at each node based on limited fault syndrome information.

In this chapter, diagnosis using limited fault syndrome information is addressed and optimal probabilistic diagnosis algorithms for several categories of diagnosis methods are derived. This chapter is restricted to diagnosis using a single syndrome. Although it is sometimes possible to obtain better diagnosis results by using multiple syndromes [55], in order to derive more general results, single syndrome diagnosis is addressed. Multiple syndrome diagnosis can be based on single syndrome diagnosis methods. In Section 6.2, it is shown how diagnosis algorithms such as those described in [5] and [19] can be efficiently implemented in a distributed manner. The resulting distributed diagnosis algorithms have linear computational complexity and require low communication overhead.

Probabilistic analysis is also used to obtain new bounds on the required number of tests for which asymptotically correct diagnosis is possible as the system size grows to infinity. In

[6], Blough showed that for regular testing graphs with N nodes, the probability of correct diagnosis of any diagnosis algorithm approaches zero as $N \rightarrow \infty$ if the number of test conducted grows slower than $N \log N$. He also described a diagnosis algorithm in [6] and [5] for which the probability of correct diagnosis approaches one as $N \rightarrow \infty$ if the number of tests conducted grows faster than $N \log N$. In this chapter, these bounds are improved by showing that when the number of tests conducted grows as $N \log N$, then certain types of probabilistic diagnosis algorithms can achieve 100% correct diagnosis as $N \rightarrow \infty$ under certain conditions on the fault coverage and prior fault probability parameter values. In addition, if certain other conditions on the fault coverage and prior fault probability are not satisfied, then it can be shown that no diagnosis algorithm (of a certain type to be specified) can achieve 100% correct diagnosis as $N \rightarrow \infty$.

Probabilistic diagnosis algorithms introduced by other researchers [5, 19] have been based on simply trying to achieve 100% correct diagnosis as the system size grows to infinity, given certain restrictions on the interconnection network structure. However, under these same restrictions, there are many probabilistic diagnosis algorithms that achieve 100% correct diagnosis as the system size grows to infinity. This point is addressed by introducing probabilistic diagnosis algorithms that achieve the optimal level of diagnostic accuracy given that certain types of syndrome information are being used. However, the optimal diagnosis algorithms introduced may appear to have the disadvantage that they are dependent upon the accurate characterization of the behavior of non-faulty and faulty nodes with probability parameters. Nevertheless, by analyzing the behavior of our optimal diagnosis algorithms given reasonable ranges of probability parameter values, it is shown that the optimal diagnosis algorithms perform as well as the best alternative algorithms in the literature when probability parameters are incorrectly specified.

The probabilistic analysis methods introduced in this chapter permit us to compare probabilistic diagnosis algorithms introduced by previous researchers with each other and with our optimal algorithms. The analysis shows that Dahbura *et. al.*'s (DSK) algorithm [19] is superior to Blough *et. al.*'s (BSM) algorithm [5]. Both algorithms perform worse than the optimal category 2 and 2A algorithms. Simulation results are also used to support the analytic results. Based on probabilistic analysis of the DSK algorithm, a modified form of the DSK algorithm that uses only relative prior fault probability parameter information is introduced. This algorithm, called the DSK* algorithm, performs significantly better than DSK and achieves close to optimal performance in many instances.

4.2. Background

There are a few diagnosis algorithms in the current literature which are closely related to the diagnosis algorithms presented in this chapter. The MPD algorithm, described in Chapter 3 and in [43], guarantees the most probable diagnosis given a syndrome. Since the MPD algorithm results in the highest achievable level of diagnostic accuracy, it is used to evaluate the performance of other diagnosis algorithms. Blough *et. al.*'s (BSM) diagnosis algorithm and Dahbura *et. al.*'s (DSK) diagnosis algorithm are category 2A probabilistic diagnosis algorithms. The MPD, BSM, and DSK diagnosis algorithms will be compared to the diagnosis algorithms presented in this chapter and to each other.

In comparing these algorithms, the testing models used in each of the algorithms must be unified. The MPD algorithm uses the testing model introduced in Chapter 2. The BSM algorithm uses a testing model in which the behavior of a faulty node is completely unspecified, i.e., r_{ij} and s_{ij} parameter values are not known. The DSK algorithm is based on a comparison-testing model. To convert the undirected testing graph model required by comparison-testing into a directed testing graph model, each undirected testing edge between

nodes u_i and u_j is replaced by two directed edges e_{ij} and e_{ji}. $a_{ij} = a_{ji} = 1$ if nodes u_i and u_j produce different results for the same task. In this case, the parameter $r_{ji} = 1 - p_{ij}$. In the DSK algorithm, it was assumed that faulty nodes could produce M different types of outputs with uniform probability, with M being a large number. Thus, $s_{ij} = 1 - (1 - p_{ij})^2 - (p_{ij}M^{-1})^2 \approx 1 - (1 - p_{ij})^2 = 1 - r_{ji}^2$. Note also that $p_{ji} = p_{ij}$, $r_{ij} = r_{ji}$, and $s_{ji} = s_{ij}$.

The DSK algorithm is a simple algorithm which repeatedly selects and removes from the updated syndrome an arbitrary node which is incident on the greatest number of one-links until no one-links remain in the syndrome.

Algorithm DSK:

0. Let $G' = G_1$ ($G' = (V', E')$ with $V' = V_1$ and $E' = E_1$);
   Also, let $F \leftarrow \varnothing$ be the set of diagnosed faulty nodes;
1. **For** each $u_i \in V'$, let $d'(u_i) \leftarrow d(u_i)$;
2. **While** $E' \neq \varnothing$ **do**
   choose $u_k \in V'$ such that $d'(u_k) = \max_{u_i}\{d'(u_i)\}$;
   for each $e_{km} \in E'$ such that $a_{km} = 1$ **do**
   $d'(u_m) = d'(u_m) - 1$;
   $F \leftarrow F \cup \{u_k\}$ and $G' \leftarrow G' - \{u_k\}$;

In the DSK algorithm, one node u_k is chosen arbitrarily from among the nodes u_i with the maximum value of $d'(u_i)$.

The BSM algorithm is described in Section 3.2. Blough [6] showed that the BSM algorithm has a time and space complexity of $O(|E_T|)$ and achieves 100% correct diagnosis as $N \rightarrow \infty$ for systems containing a number of edges growing faster than $N \log N$. In Section 4.4.1, it is shown that the DSK algorithm is also $O(|E_T|)$ and has the same property of asymptotically correct diagnosis.

4.3. Optimal Probabilistic Diagnosis Algorithms

In this section probability analysis is used to derive optimal diagnosis algorithms for category 2, 2A, 3, and 3A probabilistic diagnosis. Each category of probabilistic diagnosis is defined by the type of syndrome information used in the diagnosis. Given a certain type of syndrome information, the optimal diagnosis algorithm is to make the most probable diagnosis at each node. This was shown formally for category 1 diagnosis by Blough [6]. The same property is now shown for general categories of diagnosis. Consider category x diagnosis. Let $OPTx$ denote the algorithm which makes the most probable diagnosis for each node u_i given SD_i.

Theorem 4.1: For any category x diagnosis algorithm A, P({A produces correct diagnosis}) $\leq$ P({$OPTx$ produces correct diagnosis}).

Proof: Consider a testing graph G and an arbitrary category x diagnosis algorithm A. A produces correct diagnosis if and only if the diagnosis of each node is correct. Let A_i denote the diagnosis of node u_i by A. For an arbitrary node $u_i \in V_T$,

$$
\begin{aligned}
P_i^x(Correct_G(A_i)) &= \sum_{SD_i \in SD_i^{all}} P_i^x(SD_i\ ,\ Diag_{A_i}(SD_i)) \\
&= \sum_{SD_i \in SD_i^{all}} P_i^x(Diag_{A_i}(SD_i) \mid SD_i)\ P_i^x(SD_i) \\
&\leq \sum_{SD_i \in SD_i^{all}} P_i^x(Diag_{OPTx_i}(SD_i) \mid SD_i)\ P_i^x(SD_i) \\
&= P_i^x(Correct_G(OPTx_i)) \quad .
\end{aligned}
$$

Then, since $P_i^x(Correct_G(A_i)) \leq P_i^x(Correct_G(OPTx_i))$ for all nodes $u_i \in V_T$, the theorem follows. **Q.E.D.**

Thus, the strategy used for deriving an optimal diagnosis algorithm is based on the calculation of the posterior fault probability for each node given the syndrome information for that category of diagnosis.

Several assumptions are made in our probability analysis. It is assumed that the probability parameter values of different nodes are independent. f_i, p_{ij}, r_{ij}, and s_{ij} probability parameter values are assumed to be greater than 0 and less than 1. Since values arbitrarily close to 0 or 1 can be chosen, this is a reasonable and non-restrictive assumption. The probability of a faulty node having a one-link incident on it is assumed to be greater than the probability of a non-faulty node having a one-link incident on it. This last assumption is required if any probabilistic system-level diagnosis is to work.

4.3.1. Diagnosis Using Purely Local Syndrome Information

Given an arbitrary node $u_i \in V_T$ and a node $u_j \in \Gamma^{-1}(u_i)$, let $A_{ji} = P(a_{ji} = 1 \mid \delta_i)$ and $B_{ji} = P(a_{ji} = 1 \mid \bar{\delta}_i)$. It is assumed that $A_{ji} > B_{ji}$. Using the parameters of our testing model,

$$A_{ji} = (1 - f_j)\, p_{ji} + f_j s_{ji} \quad ,$$

$$B_{ji} = f_j(1 - r_{ji}) \quad .$$

Let the local syndrome information used by node $u_i \in V_T$ be denoted by LS_i. Then the probability measure P_i^3 is defined as

$$P_i^3(LS_i \ , \delta_i) = f_i \prod_{u_j \in \Gamma_1^{-1}(u_i)} A_{ji} \prod_{u_j \in \Gamma_0^{-1}(u_i)} (1 - A_{ji}) \ ,$$

$$P_i^3(LS_i \ , \bar{\delta}_i) = (1 - f_i) \prod_{u_j \in \Gamma_1^{-1}(u_i)} B_{ji} \prod_{u_j \in \Gamma_0^{-1}(u_i)} (1 - B_{ji}) \ .$$

It can be checked that P_i^3 is a legitimate probability measure. It follows that

$$P_i^3(\delta_i \mid LS_i) = \frac{P_i^3(LS_i \ , \delta_i)}{P_i^3(LS_i \ , \delta_i) + P_i^3(LS_i \ , \bar{\delta}_i)}$$

$$= \frac{1}{1 + \dfrac{1 - f_i}{f_i} \displaystyle\prod_{u_j \in \Gamma_1^{-1}(u_i)} \left[\frac{B_{ji}}{A_{ji}}\right] \prod_{u_j \in \Gamma_0^{-1}(u_i)} \left[\frac{1 - B_{ji}}{1 - A_{ji}}\right]} \quad . \tag{4.2}$$

To make the most probable diagnosis for u_i based on u_i's local syndrome information, u_i must be diagnosed to be faulty if and only if Eq. (4.2) > 0.5. Thus, the optimal category 3 probabilistic diagnosis algorithm is given as:

Algorithm OPT3:

For all nodes $u_i \in V_T$, **do**
1. calculate posterior fault probability for u_i using Eq. (4.2);
2. if $P(\delta_i \mid LS_i) > 0.5$, then label u_i as faulty;
   otherwise, label u_i as non-faulty;

In category 3A probabilistic diagnosis, the strategy is to determine the fault status of each node based on summarized local syndrome information, i.e., the number of neighbors of the node that test it to be faulty. The main advantages of using summarized local syndrome information (instead of local syndrome information) are that the resulting diagnosis algorithms are simpler, less dependent on the accuracy of probability parameter values, and implementable as constant-time distributed algorithms. In the probability analysis for the category 3A probabilistic diagnosis on an arbitrary node $u_i \in V_T$, it is assumed that the testing graph is regular (has constant node-degree) and that the average probability parameter values of nodes in $\Gamma^{-1}(u_i)$ are being used. Then, going through a similar process as the previous analysis, one can write the posterior fault probability of u_i given $z = d(u_i)$ one-links directed into u_i out of a maximum of $\gamma = |\Gamma^{-1}(u_i)|$ (denoted by z *one–links* : γ) as

$$P_i^{3A}(\delta_i \mid z \text{ one–links} : \gamma) = \frac{1}{1 + \frac{1-f_i}{f_i}\left[\frac{B}{A}\right]^z \left[\frac{1-B}{1-A}\right]^{\gamma - z}}, \tag{4.3}$$

where $A = (1-f)p + fs$ and $B = f(1-r)$.

Implicit in the above analysis is the fact that the partial syndrome information used in category 3A probabilistic diagnosis is denoted by z *one–links* : γ (note that $z = d(u_i)$). Also, although the probability measure P_i^{3A} has not been formally defined, it is easy to see what $P_i^{3A}(z \text{ one–links} : \gamma, \delta_i)$ and $P_i^{3A}(z \text{ one–links} : \gamma, \bar{\delta}_i)$ must be by referring to the analysis

for P_i^3 and Eq. (4.3). A similar procedure will be used in describing category 2 and 2A probabilistic diagnosis (the notation for the partial syndrome information used and the definition of the formal probability measure will be obvious from the discussion).

Eq. (4.3) is an increasing function of z since $A > B$. By setting Eq. (4.3) equal to $\frac{1}{2}$ and solving for z, a threshold value z_{th_i} can be determined. When $z > z_{th_i}$ ($z < z_{th_i}$), node u_i is likely to be faulty (fault-free) since $P(\delta_i \mid z \text{ one-links} : \gamma) > 0.5$ ($P(\bar{\delta}_i \mid z \text{ one-links} : \gamma) > 0.5$). Solving for z_{th_i},

$$z_{th_i} = \frac{\log\left[\frac{1-f_i}{f_i}\right]}{\log\left[\frac{A(1-B)}{(1-A)B}\right]} + \gamma \frac{\log\left[\frac{1-B}{1-A}\right]}{\log\left[\frac{A(1-B)}{(1-A)B}\right]} . \tag{4.4}$$

This results in the following optimal category 3A probabilistic diagnosis algorithm.

Algorithm OPT3A:

For all nodes $u_i \in V_T$, **do**
1. calculate z_{th_i} using Eq. (4.4);
2. if $d(u_i) > z_{th_i}$, then label u_i as faulty;
   otherwise, label u_i as non-faulty.

4.3.2. Category 2 and 2A Probabilistic Diagnosis

The first node that is identified to be faulty should be the one with the highest posterior fault probability. Once the first node is identified to be faulty on the basis of Eq. (4.2), the node with the next highest posterior fault probability must be identified. However, the fact that one node has already been identified to be faulty can be used to update the posterior fault probability of adjacent nodes. Suppose u_k has previously been identified to be faulty. Let $u_i \in \Gamma(u_k)$ be an arbitrary as-yet undiagnosed node. Then, from Table 2.1, it is known that $A_{ki} = P(a_{ki} = 1 \mid \delta_i) = s_{ki}$ and $B_{ki} = P(a_{ki} = 1 \mid \bar{\delta}_i) = 1 - r_{ki}$. In general, if the nodes in

$H_{i1} \subseteq \Gamma_1^{-1}(u_i)$ and $H_{i0} \subseteq \Gamma_0^{-1}(u_i)$ have previously been identified to be faulty, then

$$P_i^2(\delta_i \mid LS_i\ , H_{i1}\ , H_{i0}) =$$

$$\frac{1}{1 + \frac{1-f_i}{f_i} \prod_{u_j \in \Gamma_1^{-1}(u_i) - H_{i1}} \left[\frac{B_{ji}}{A_{ji}}\right] \prod_{u_j \in \Gamma_0^{-1}(u_i) - H_{i0}} \left[\frac{1-B_{ji}}{1-A_{ji}}\right] \prod_{u_j \in H_{i1}} \left[\frac{1-r_{ji}}{s_{ji}}\right] \prod_{u_j \in H_{i0}} \left[\frac{r_{ji}}{1-s_{ji}}\right]} \quad (4.5)$$

Relying purely on probabilistic information, the process of identifying faulty nodes should stop when there does not exist any node with a posterior fault greater than 0.5. This can result in a fault set F which is not a vertex cover of G_1, the one-condensation of the testing graph G. A fault set which is not a vertex cover of G_1 can not have produced the syndrome for which the diagnosis is made. Thus, a better stopping condition is to continue identifying faulty nodes until the resulting fault set is a vertex cover of G_1. (In effect, a little bit more syndrome information is being used.) By careful analysis [43], it can be shown that except under extremely extraordinary circumstances, the most probable fault set is a subset of V_1, the node set of G_1. The fault set found by using Eq. (4.5) and this stopping condition is also typically a subset of V_1. The following is the optimal category 2 probabilistic diagnosis algorithm.

Algorithm OPT2:

0. Let $F \leftarrow \varnothing$ be the set of diagnosed faulty nodes;
   let $G_2 = (V_2, E_2)$ be a copy of G_1;
1. **For all** $u_i \in V_T$ **do**
   - use Eq. (4.5) with $H_{i1} = H_{i0} = \varnothing$ to calculate posterior fault probabilities;
2. **While** $E_2 \neq \varnothing$, **do**
   2a. Let u_k be the node with highest posterior fault probability;
   2b. $F \leftarrow F \cup \{u_k\}$;
   2c. **For all** nodes $u_j \in \Gamma(u_k)$ **do**
   - update posterior fault probability of u_j using Eq. (4.5);
   2d. Update E_2 by removing all links to and from u_k;

The main advantage in going from a category 2 to a category 2A probabilistic diagnosis method is that the resulting diagnosis is less susceptible to inaccurate probability parameter

values. In category 2A probabilistic diagnosis, the first node to be identified as faulty is the node with the highest posterior fault probability as calculated using Eq. (4.3). But then, the equation for updating the posterior fault probability of nodes adjacent to previously identified faulty becomes similar to Eq. (4.5). If the nodes in $H_{i1} \subseteq \Gamma_1^{-1}(u_i)$ and $H_{i0} \subseteq \Gamma_0^{-1}(u_i)$ have previously been identified to be faulty, then

$$P_i^{2A}(\delta_i \mid z \text{ one-links} : \gamma, H_{i1}, H_{i0}) =$$

$$\frac{1}{1 + \frac{1-f_i}{f_i}\left[\frac{B}{A}\right]^{z-|H_{i1}|}\left[\frac{1-B}{1-A}\right]^{\gamma-z-|H_{i0}|}\prod_{u_j \in H_{i1}}\left[\frac{1-r_{ji}}{s_{ji}}\right]\prod_{u_j \in H_{i0}}\left[\frac{r_{ji}}{1-s_{ji}}\right]} \quad (4.6)$$

Thus, Algorithm OPT2A is the same as Algorithm OPT2 with Eq. (4.5) replaced by Eq. (4.6) in Steps 1 and 2c.

4.4. Analysis

4.4.1. Analysis of DSK Algorithm

In this section, it is shown that for regular testing graphs, the DSK algorithm has the same property of asymptotically correct diagnosis as the BSM algorithm. It is also shown that the DSK algorithm has the same computational complexity as the BSM algorithm. Finally, the DSK algorithm is analyzed to show that its diagnostic accuracy can be improved by using prior fault probability information. For simplicity of analysis, average values of probability parameter values will be used. Let $\gamma = |\Gamma^{-1}(u_i)|$, which is a constant given a regular testing graph. Then, κ, the average κ_i value, is equal to $\frac{\gamma}{2}\left[f + p(1-f)\right]$.

Computational Complexity

Dahbura *et. al.* [19] showed that the DSK algorithm has computational complexity $O(N^2)$. However, using a proof similar to [6], it can be shown that the DSK algorithm is also

$O(|E_T|)$.

Theorem 4.2: Algorithm DSK has a time and space complexity of $O(|E_T|)$.

Proof: Referring to the description of Algorithm DSK in Section 4.2, Steps 0 and 1 (the initial calculation of $d(u_i)$ values) can clearly be done in $O(|E_T|)$ time and space. The loop in Step 2 can be implemented in the following manner. A data structure of $N-1$ buckets labeled 0 through $N-1$ is constructed and each processor is placed in the bucket corresponding to its initial $d(u_i)$ value. Next, a processor from the largest non-empty bucket is selected and placed in F. For each processor added to F, the bucket assignments of the processors that it tests are changed. Then this process is repeated for the next non-empty bucket — the process stops when the only non-empty bucket is bucket 0. Since processors can be added to F at most once, each testing list is traversed at most once. Finally, by using doubly-linked lists within the buckets and keeping pointers from processors to their positions in the buckets, a processor can be found and moved in constant time. Thus, the loop in Step 2 can be implemented in $O(|E_T|)$ time and space. **Q.E.D.**

Asymptotic Performance of DSK

If the testing graph is regular, the reader will note that algorithms BSM and DSK become similar. However, there are two important differences between the two algorithms that make the following theorem concerning the asymptotic performance of DSK non-trivial. First, when a node u_j is placed in the set F in BSM, its outward-bound 0-links are changed to 1-links, while in DSK, u_j's outward-bound 1-links are changed to 0-links. Second, BSM terminates when the maximum "updated" $d(u_i)$ value is not more than a fixed threshold value κ, while DSK continues until the maximum "updated" $d(u_i)$ value is 0.

Theorem 4.3: Given a regular testing graph $G(V_T, E_T)$ with $|\Gamma^{-1}(u_i)| = \gamma$ for all $u_i \in V_T$, if $\gamma \geq \omega(N) \log N$, where $\omega(N) \to \infty$ as $N \to \infty$ and $p > \frac{f}{1-f}$, then $P(\{\text{DSK}$

produces correct diagnosis}) $\rightarrow 1$ as $N \rightarrow \infty$.

Proof: Let us divide the operation of DSK into two parts, DSK1 and DSK2. In DSK1, Algorithm DSK is executed until $\max_{u_i}\{d(u_i)\} < \kappa$ for all $u_i \in V'$. In DSK2, the rest of Algorithm DSK is executed.

We now prove that Algorithm DSK1 approaches 100% correct diagnosis as $N \rightarrow \infty$. Consider the following two cases: 1) a non-faulty node is tested by more than κ faulty nodes, and 2) a faulty node is not failed by more than κ non-faulty nodes. Clearly, given either of these two cases, it is possible for DSK1 to produce an incorrect diagnosis. Suppose neither Case 1 nor Case 2 holds. Then, for every non-faulty node u_i, $d(u_i) \leq \kappa$, and for every faulty node u_j, $d(u_j) > \kappa$ on each iteration of Step 1 of the DSK1 algorithm. Thus, DSK1 will produce a correct diagnosis if neither Case 1 nor Case 2 holds. Let the random variables X and Y denote the number of nodes that satisfy Case 1 and Case 2, respectively. Then, $P(\{\text{DSK1 produces correct diagnosis}\}) \geq 1 - E[X] - E[Y]$. By using Chernoff bounds (Section 5.5), Blough [6] showed that $E[X]$ and $E[Y]$ approach zero as $N \rightarrow \infty$ when $p > f/(1-f)$ and $|\Gamma^{-1}(u_i)| \geq \omega(N) \log N$. Thus, $P(\{\text{DSK1 produces correct diagnosis}\}) \rightarrow 1$ as $N \rightarrow \infty$.

Since DSK = DSK1 if DSK1 has caught all of the faulty nodes that exist, $P(\{\text{DSK} = \text{DSK1}\}) \rightarrow 1$ as $N \rightarrow \infty$. Therefore, $P(\{\text{DSK produces correct diagnosis}\}) \rightarrow 1$ as $N \rightarrow \infty$. **Q.E.D.**

Probabilistic Analysis of DSK

By analyzing the DSK algorithm, it is possible to make the choice of u_k in Step 2 of the algorithm more intelligently based on prior fault probability values. Intuitively, if $f_j > f_i$ and both nodes u_i and u_j have the same number of one-links incident on them, then it is better to choose u_j over u_i. Probabilistic analysis supports this observation. However, if $f_j > f_i$ and

$d(u_j) < d(u_i)$, the choice to be made is not immediately clear. In this case, the current syndrome points to u_i but prior knowledge points to u_j. The choice depends on the extent to which the current syndrome is believed over the prior knowledge. Bayesian probability analysis provides us with an answer to this problem.

Given a node u_i with z one-links incident on it out of a maximum of γ (denoted by z *one–links* : γ),

$$P(\delta_i \mid z \text{ one–links} : \gamma) = \frac{P(z \text{ one–links} : \gamma \mid \delta_i)\, P(\delta_i)}{P(z \text{ one–links} : \gamma)} \quad ,$$

$$P(z \text{ one–links} : \gamma \mid \delta_i)\, P(\delta_i) = f_i \left[\sum_{j=0}^{\gamma} \binom{\gamma}{j} (1-f)^j f^{\gamma-j} \sum_{k=0}^{\min\{j,z\}} \binom{j}{k} p^k (1-p)^{j-k} \right.$$

$$\left. \binom{\gamma-j}{z-k} s^{z-k} (1-s)^{\gamma-j-(z-k)} \right]$$

$$= f_i\, A(z) \quad ,$$

$$P(z \text{ one–links} : \gamma) = f_i\, A(z) + (1-f_i) \left[\sum_{j=z}^{\gamma} \binom{\gamma}{j} f^j (1-f)^{\gamma-j} \binom{j}{z} (1-r)^z r^{j-z} \right]$$

$$= f_i\, A(z) + (1-f_i)\, B(z) \quad .$$

Then using these equations, one obtains

$$\frac{P(\delta_m \mid z \text{ one–links} : \gamma)}{P(\delta_l \mid y \text{ one–links} : \gamma)} = \frac{f_m\, A(z) \left[f_l\, A(y) + (1-f_l)\, B(y) \right]}{f_l\, A(y) \left[f_m\, A(z) + (1-f_m)\, B(z) \right]}$$

$$= \frac{f_m\, f_l\, A(z)\, A(y) + f_m\, (1-f_l)\, A(z)\, B(y)}{f_m\, f_l\, A(z)\, A(y) + f_l\, (1-f_m)\, A(y)\, B(z)} \quad . \tag{4.7}$$

If $y = z$, $f_m > f_l$ implies that Eq. (4.7) > 1, which in turn implies that u_m should be chosen as faulty over u_l.

Since the first terms in both the numerator and denominator of Eq. (4.7) are the same, Eq. (4.7) depends on the second terms. The crossover point at which $f_m > f_l$ implies that u_m

is more likely to be faulty than u_l occurs at

$$\frac{f_m\ (1-f_l)}{f_l\ (1-f_m)} = \frac{A(y)\ B(z)}{A(z)\ B(y)}\ . \tag{4.8}$$

Thus, Eq. (4.8) tells us when u_m should be chosen as faulty over u_l even though $d(u_m) < d(u_l)$. Intuitively, it can be observed that the greater the difference $y - z$, the greater the ratio of $f_m/(1-f_m)$ to $f_l/(1-f_l)$ must be for u_m to be chosen as faulty over u_l. Thus, if u_m would not be chosen as faulty over u_l with $y = z + 1$, then u_m would also not be chosen as faulty over u_l with $y = z + k$ $(1 < k \leq \gamma - z)$. Eq. (4.8) was calculated for the case $y = z + 1$ using all sets of parameter values used in our experiments. Empirically, it was found that the equation is insensitive to the parameters z and γ. This can also be seen by using Eq. (4.3) as the posterior fault probability equation. Table 4.1 shows Eq. (4.8) calculated for the case $y = z + 1$ with several f and p values. As can be seen from the table, the values of Eq. (4.8) for $y = z + 1$ are typically fairly large.

	$f = 0.0484$	$f = 0.0099$	$f = 0.0050$
$p = 0.3$	30.5	145.3	286.7
$p = 0.4$	35.4	169.4	334.3
$p = 0.5$	42.3	203.0	401.0
$p = 0.6$	52.7	253.5	501.0
$p = 0.7$	69.9	337.7	667.7
$p = 0.8$	104.3	506.1	1001.0
$p = 0.9$	207.6	1011.1	2001.0

Table 4.1: Eq. (4.8) with $y = z + 1$ for various values of f and p.

The somewhat surprising conclusion is that unless the differences in the prior fault probabilities are extremely large, prior fault probabilities should only be used to break ties and *not* to override current syndrome information, i.e., one should not choose u_m over u_l when $d(u_m) < d(u_l)$ and $f_m > f_l$ for most cases.

Based on this conclusion, a new diagnosis algorithm called the DSK* algorithm, based on the DSK algorithm, is designed as described below.

Algorithm DSK:*

0. Let $G' = G_1$ ($G' = (V', E')$ with $V' = V_1$ and $E' = E_1$);
   Also, let $F \leftarrow \emptyset$ be the set of diagnosed faulty nodes;
1. **For** each $u_i \in V'$, let $d'(u_i) \leftarrow d(u_i)$;
2. **While** $E' \neq \emptyset$ **do**
   let R be the set of nodes u_l such that $d(u_l) = \max_{u_i}\{d(u_i)\}$;
   choose the node u_k with the highest f_i value in R;
   for each $e_{km} \in E'$ such that $a_{km} = 1$ **do**
   $d'(u_m) = d'(u_m) - 1$;
   $F \leftarrow F \cup \{u_k\}$ and $G' \leftarrow G' - \{u_k\}$;

The DSK* algorithm has $O(N^2)$ time and space complexity. DSK* is identical to DSK except for Step 2 of the DSK algorithm, in which the choice of node u_k is changed slightly. As will be seen in Section 4.6, this apparently small change results in a significant improvement in diagnostic accuracy.

4.4.2. Analysis of Behavior of Optimal Algorithms

Optimal diagnosis algorithms have been derived for category 3, 3A, 2, and 2A probabilistic diagnosis. However, all of these algorithms depend upon the use of several probability parameters. Some of these probability parameters such as s_{ij} values are difficult to determine accurately. The parameter p_{ij} ($= 1 - r_{ji}$ with comparison-testing) is also difficult to determine accurately since intermittent faults can occur. This means that A_{ji} and B_{ji} also cannot be accurately determined. Although it is possible to estimate f_i values based on manufacturers' reliability estimates of components and standard failure rate models (e.g., exponential or Weibull distributions), the accuracy of such f_i estimates is limited. Therefore, it is necessary to show that our optimal diagnosis algorithms perform well even when inaccurate probability parameter estimates are used. A basic assumption used in our analysis is that the probability parameter values of different nodes do not vary extremely widely. For instance, it is assumed

that fault coverage values p do not vary by more than a factor of about 10.

Consider category 3 and 3A probabilistic diagnoses. If the probability parameter values of different nodes do not vary extremely widely, then Algorithm OPT3A approximates the behavior of Algorithm OPT3 since Eq. (4.3) approximates Eq. (4.2). Now, the behavior of Algorithm OPT3A depends on the threshold value z_{th_i} chosen for each node u_i. Suppose another threshold value c_i is chosen for u_i. The behavior of Algorithm OPT3A in diagnosing u_i does not change as long as $\lfloor z_{th_i} \rfloor = \lfloor c_i \rfloor$. Table 4.2 shows the values of z_{th_i} given three different f values and 7 different p values with $\gamma = 10$. The reader will note that when $\lfloor z_{th_i} \rfloor$ values are considered, a fairly wide range of p and f values will result in identical OPT3A algorithm behaviors. As γ is increased, the range of p and f values for which this is true will become smaller since z_{th_i} is a linear function of γ. However, it is also true that as γ is increased, the effect of small differences in z_{th_i} threshold values on the diagnosis produced become smaller. As will be seen in Section 4.5, this is because the number of nodes u_i with intermediate $d(u_i)$ values (for which diagnosis is difficult) decreases as γ is increased. Thus, even with inaccurate probability parameter values, the OPT3A algorithm performs almost well as when the correct probability parameter values are used.

p	f = 0.0484	f = 0.0099	f = 0.0050
0.3	1.91	1.64	1.57
0.4	2.27	1.89	1.79
0.5	2.65	2.17	2.04
0.6	3.06	2.49	2.33
0.7	3.54	2.86	2.67
0.8	4.10	3.32	3.10
0.9	4.87	3.99	3.73

Table 4.2: Threshold values z_{th_i} with different f and p values.

The analysis for category 2 and category 2A probabilistic diagnosis is a bit more involved. First, it can again be stated that if the probability parameter values of different

nodes do not vary extremely widely, then Algorithm OPT2A approximates the behavior of Algorithm OPT2 since Eq. (4.6) approximates Eq. (4.5). Now it will be shown that the OPT2A algorithm performs as well or better than the DSK* and DSK algorithms, even with inaccurate probability parameter estimates, if the estimates are within a factor of about 30 of the correct values.

From Eq. (4.6), it is known that the initial posterior fault probability of each node is directly related to the number of one-links incident on it. Let u_m and u_l be two arbitrary nodes with z and y one-links incident on them, respectively. Then, calculating the ratio of the posterior fault probabilities of u_m to u_l and simplifying (note that $\varnothing$, $\varnothing$ refers to the values of the sets H_{i1} and H_{i0}),

$$\frac{P_i^{2A}(\delta_m \mid z \text{ one-links} : \gamma, \varnothing, \varnothing)}{P_i^{2A}(\delta_l \mid y \text{ one-links} : \gamma, \varnothing, \varnothing)} = \frac{1 + \dfrac{1-f_l}{f_l}\left[\dfrac{B}{A}\right]^y \left[\dfrac{1-B}{1-A}\right]^{\gamma-y}}{1 + \dfrac{1-f_m}{f_m}\left[\dfrac{B}{A}\right]^z \left[\dfrac{1-B}{1-A}\right]^{\gamma-z}} .$$

If this ratio is greater than 1, then u_m should be chosen as faulty over u_l. This condition can be simplified by requiring that the following condition be satisfied:

C1: $$\frac{f_m\,(1-f_l)}{f_l\,(1-f_m)} > \left[\frac{A\,(1-B)}{B\,(1-A)}\right]^{y-z} .$$

Thus, if $y = z$ and $f_m > f_l$, then u_m will be chosen as faulty over u_l. Suppose $f_m > f_l$ but $z < y$. Condition C1 tells us when u_m should be chosen as faulty over u_l even though $z = d(u_m) < d(u_l) = y$. The right side of condition C1 increases with $y - z$. Even when $y = z + 1$, the right side of condition C1 is typically a fairly large number. For example, suppose $p = 0.6$ and $f = 0.01$. Assuming comparison-testing is used, $A \approx p = 0.6$ and $B = fp = 0.006$. Thus, the right side of condition C1 = 248.5. This says that the ratio of f_m to f_l must be greater than at least 248.5 for us to choose u_m as faulty over u_l when

$y = z + 1$. Calculating the right side of condition C1 for all sets of p and f values used in our simulations, the smallest value obtained was 30.5 with $f = 0.0484$ and $p = 0.3$. Thus, it can be concluded that when f values of different nodes do not vary extremely widely (by more than a factor of approximately 30), using Eq. (4.6) to calculate initial posterior fault probability values will result in the same initial diagnosis as DSK*.

Now, consider what happens after several nodes have been identified to be faulty. If diagnosis is based purely on the number of one-links incident on a node, then it is desirable to be able to compare a node adjacent to several previously identified faulty nodes with nodes not adjacent to any previously identified faulty nodes. If this comparison is made based on the number of one-links incident on that node, one would like to estimate Eq. (4.6) by an equation of the form $P_i^{2A}(\delta \mid z \ one\text{-}links : \gamma, \varnothing, \varnothing)$ even when $H_{i1} \neq \varnothing$ and/or $H_{i0} \neq \varnothing$. If this can be done, then it will be seen that the OPT2A algorithm identifies faulty nodes in order from nodes with large $d(u_i)$ values to nodes with small $d(u_i)$ values.

Consider a node u_i and suppose that $H_{i1} \subseteq \Gamma_1^{-1}(u_i)$ and $H_{i0} \subseteq \Gamma_0^{-1}(u_i)$ have previously been identified to be faulty. Let $h_{i1} = |H_{i1}|$ and $h_{i0} = |H_{i0}|$. The BSM algorithm effectively says to make the approximation

$$P_i^{2A}(\delta_i \mid y \ one\text{-}links : \gamma, H_{i1}, H_{i0}) \approx P_i^{2A}(\delta_i \mid y + h_{i0} \ one\text{-}links : \gamma, \varnothing, \varnothing)$$

and the DSK algorithm effectively says to make the approximation

$$P_i^{2A}(\delta_i \mid y \ one\text{-}links : \gamma, H_{i1}, H_{i0}) \approx P_i^{2A}(\delta_i \mid y - h_{i1} \ one\text{-}links : \gamma, \varnothing, \varnothing) \ .$$

Which is the better approximation? Is it possible to make an approximation which is better than both of them?

Let $C(z, \gamma, h_{i1}, h_{i0})$ represent the denominator of Eq. (4.6) minus one. Then,

$$C(y\ ,\gamma\ ,1\ ,0) = C(y\ ,\gamma\ ,0\ ,0)\left[\frac{1-r}{s}\right]\left[\frac{A}{B}\right]\ , \tag{4.9}$$

$$C(y\ ,\gamma\ ,1\ ,0) = C(y-1\ ,\gamma\ ,0\ ,0)\left[\frac{1-r}{s}\right]\left[\frac{1-A}{1-B}\right]\ , \tag{4.10}$$

$$C(y\ ,\gamma\ ,0\ ,1) = C(y+1\ ,\gamma\ ,0\ ,0)\left[\frac{r}{1-s}\right]\left[\frac{A}{B}\right]\ , \tag{4.11}$$

$$C(y\ ,\gamma\ ,0\ ,1) = C(y\ ,\gamma\ ,0\ ,0)\left[\frac{r}{1-s}\right]\left[\frac{1-A}{1-B}\right]\ . \tag{4.12}$$

Thus, in the two cases of h_{i1} and h_{i0} pairs considered, the BSM approximation would say that (Eqs. (4.9) and (4.11))

$$\left[\frac{1-r}{s}\right]\left[\frac{A}{B}\right] \approx \left[\frac{r}{1-s}\right]\left[\frac{A}{B}\right] \approx 1\ ,$$

and the DSK approximation would say that (Eqs. (4.10) and (4.12))

$$\left[\frac{1-r}{s}\right]\left[\frac{1-A}{1-B}\right] \approx \left[\frac{r}{1-s}\right]\left[\frac{1-A}{1-B}\right] \approx 1\ .$$

Using our previous example with $A = 0.6$, $B = 0.006$, $p = 0.6$, and assuming comparison-testing,

$$\left[\frac{1-r}{s}\right]\left[\frac{A}{B}\right] = 71.43\ ,\quad \left[\frac{r}{1-s}\right]\left[\frac{A}{B}\right] = 47.62\ ,$$

$$\left[\left[\frac{1-r}{s}\right]\left[\frac{1-A}{1-B}\right]\right]^{-1} = 3.479 \quad \text{and} \quad \left[\left[\frac{r}{1-s}\right]\left[\frac{1-A}{1-B}\right]\right]^{-1} = 5.218\ .$$

Thus, for this example, it appears that the DSK approximation is far superior to the BSM approximation but that even the DSK approximation is not that good. However, from our previous analysis, it is known that for this example, nodes with z *one–links* : γ and $z+1$ *one–links* : γ have the same posterior fault probability only if $\frac{C(z\ ,\gamma\ ,0\ ,0)}{C(z+1\ ,\gamma\ ,0\ ,0)} = 248.5$. Thus, both estimates are "safe". However, every time h_{i1} or

h_{i0} is increased, the error in the estimate increases by the appropriate factor, e.g., 3.479 for the DSK approximation with increasing h_{i1}. Assuming that f values do not vary by more than a factor of about 30, the DSK approximation is safe with $h_{i1} \leq 4$, $h_{i0} \leq 3$, and $h_{i1} + h_{i0} \leq 6$.

It is possible to make a better estimate than both the DSK and BSM approximations. Let us call this the BETTER approximation. Suppose the situation in which h_{i0} or h_{i1} is increased by one at each "step" is considered. Then, a better approximation strategy is to make the DSK approximation for several steps, followed by the BSM approximation for one step, followed by the DSK approximation for several steps, etc. The reason for this is that the DSK approximation underestimates the denominator of Eq. (4.6) and the BSM approximation overestimates the denominator of Eq. (4.6), with the overestimation several times more severe than the underestimation. By analyzing Eq. (4.6) and the DSK and BSM approximations, it is possible to determine exactly when each approximation should be used in the BETTER approximation.

Let us summarize the behavior of the OPT2A algorithm as compared to the DSK* and DSK algorithms. Suppose probability parameter values used by the OPT2A algorithm are wrong. The initial diagnosis made by the OPT2A algorithm is still the same as the DSK* algorithm. Assuming that the incorrect probability parameter values are within a reasonable range of the correct probability parameter values (e.g., within a factor of about 30), the DSK approximation (which is also used in the DSK* algorithm) is worse than the BETTER approximation. Thus, in diagnosing faulty nodes after the initial diagnosis, the OPT2A algorithm performs as well as or better than the DSK* algorithm. To summarize, the OPT2A performs as well or better than the DSK* algorithm (which in turn performs as well or better than the DSK algorithm) when probability parameter values used are within a reasonable range of the correct probability parameter values.

4.5. Theoretical Results

In this section, new bounds are derived for the asymptotically correct diagnosability of category 2, 2A, 3, and 3A diagnosis algorithms.

In our asymptotic analysis, it will be assumed that all probability parameter values used are average values. A γ-regular testing graph will also be assumed. Suppose that a category 3A probabilistic diagnosis algorithm is being used with the threshold value z_{th}. Let the random variable Y denote the number of faulty nodes that are tested to be faulty by $\leq z_{th}$ other nodes. Let the random variable X denote the number of non-faulty nodes that are tested to be faulty by $> z_{th}$ other nodes. In the following analysis, $\log N$ will refer to $\log_a N$.

Lemma 4.1: Given $\gamma = \log N$ and $z_{th} = c_1 \log N$, $\lim_{N\to\infty} E[Y] = 0$ if $A > 1 - (2a)^{\frac{1}{c_1 - 1}}$.

Proof: $P(d(u_i) = z \mid \delta_i) = \binom{\gamma}{z} A^z (1-A)^{\gamma - z}$. Thus,

$$E[Y] = \sum_{u_i \in V_T} f_i \sum_{k=0}^{\gamma} \binom{t}{k} A^k (1-A)^{\gamma - k}$$

$$= Nf \sum_{k=0}^{\gamma} \binom{\gamma}{k} A^k (1-A)^{\gamma - k} .$$

$$\lim_{N\to\infty} E[Y] = \lim_{N\to\infty} Nf \sum_{k=0}^{c_1 \log N} \binom{\log N}{k} A^k (1-A)^{\log N - k}$$

$$= f \lim_{N\to\infty} a^{\log N} (1-A)^{(1-c_1)\log N} \sum_{k=0}^{c_1 \log N} \binom{\log N}{k} A^k (1-A)^{\log N - k}$$

$$\leq f \lim_{N\to\infty} a^{\log N} (1-A)^{(1-c_1)\log N} \sum_{k=0}^{\log N} \binom{\log N}{k}$$

$$= f \lim_{N\to\infty} (1-A)^{(1-c_1)\log N} a^{\log N} 2^{\log N}$$

$$= f \lim_{N \to \infty} \left[(1 - A)^{(1 - c_1)} \, 2a \right]^{\log N}$$

$$= 0$$

since $A > 1 - (2a)^{\frac{1}{c_1 - 1}}$ implies that $\left[(1 - A)^{1 - c_1} \, 2a \right] < 1$. **Q.E.D.**

Lemma 4.2: Given $\gamma = \log N$ and $z_{th} = c_2 \log N$, $\lim_{N \to \infty} E[X] = 0$ if $B < (2a)^{\frac{-1}{c_2}}$.

Proof: $P(d(u_i) = z \mid \delta_i) = \binom{\gamma}{z} B^z \, (1 - B)^{\gamma - z}$. Thus,

$$E[X] = \sum_{u_i \in V_T} (1 - f_i) \sum_{k = z_{th} + 1}^{\gamma} \binom{\gamma}{k} B^k \, (1 - B)^{\gamma - k}$$

$$= N \, (1 - f) \sum_{k = z_{th} + 1}^{\gamma} \binom{\gamma}{k} B^k \, (1 - B)^{\gamma - k} \quad .$$

$$\lim_{N \to \infty} E[X] = \lim_{N \to \infty} N \, (1 - f) \sum_{k = z_{th} + 1}^{\gamma} \binom{\gamma}{k} B^k \, (1 - B)^{\gamma - k}$$

$$\leq \lim_{N \to \infty} N \, (1 - f) \sum_{k = z_{th}}^{\gamma} \binom{\gamma}{k} B^k \, (1 - B)^{\gamma - k}$$

$$= \lim_{N \to \infty} N \, (1 - f) \sum_{k=0}^{\gamma - z_{th}} \binom{\gamma}{k} (1 - B)^k \, B^{\gamma - k}$$

$$= \lim_{N \to \infty} N \, (1 - f) \sum_{k=0}^{(1 - c_2) \log N} \binom{\log N}{k} (1 - B)^k \, B^{\log N - k} \quad ,$$

which is the same form as $\lim_{N \to \infty} E[Y]$ with f replaced by $(1 - f)$, c_1 replaced by $(1 - c_2)$, and A replaced by $(1 - B)$. With these replacements, the condition $A > 1 - (2a)^{\frac{1}{c_1 - 1}}$ becomes $B < (2a)^{\frac{-1}{c_2}}$. Therefore, it follows that $\lim_{N \to \infty} E[X] = 0$. **Q.E.D.**

Theorem 4.4: Given $\gamma = \log N$, Algorithms OPT3, OPT3A, OPT2, OPT2A, DSK, DSK*, and MPD all achieve 100% correct diagnosis as $N \to \infty$ if $A > 1 - (2a)^{\frac{1}{1 - c}}$ and

$B < (2a)^{\frac{-1}{c}}$, where c and the threshold value z_{th} (if one exists) are chosen appropriately.

Proof: By Lemmas 4.1 and 4.2, if the conditions of the theorem are satisfied, then as $N \rightarrow \infty$, every non-faulty node u_i has $d(u_i) \leq c \log N$ and every faulty node u_j has $d(u_j) > c \log N$. The threshold value z_{th_i} chosen in Algorithm OPT3A is a linear function of γ (refer to Eq. (4.4)). Thus, clearly the theorem holds for OPT3A. Because Algorithm OPT2A uses the same initial posterior fault probability equation as OPT3A, the theorem holds for OPT2A. Since Algorithms DSK and DSK* identify faulty nodes in order from larger $d(.)$ values to smaller $d(.)$ values, the theorem holds for DSK and DSK*. The theorem also holds for Algorithm MPD since MPD has higher diagnostic accuracy than any other diagnosis algorithm [6]. Likewise, since Algorithms OPT3 and OPT2 have higher diagnostic accuracy than OPT3A and OPT2, respectively, Theorem 4.4 holds for OPT3 and OPT2. **Q.E.D.**

The above theorem places a lower bound on A and an upper bound on B such that asymptotically correct diagnosis can be obtained when the number of required tests grows as $\log N$. It is also possible to determine a fairly close upper bound on A and lower bound on B such that no category 3A probabilistic diagnosis algorithm can achieve asymptotically correct diagnosis when $\gamma = \log N$. Also, from the same analysis, it will be evident that all other diagnosis algorithms will probably not be able to do much better.

Lemma 4.3: Let $\gamma = \log N$ and $z_{th} = c \log N$. Then (1) $\lim_{N \rightarrow \infty} E[Y] = \infty$ if $(0.5 \leq c \leq 1$ and $[A^c(1-A)^{1-c}2a] > 1)$ or $(0 \leq c < 0.5$ and $[A^c(1-A)^{1-c}a] > 1)$, and (2) $\lim_{N \rightarrow \infty} E[X] = \infty$ if $(0 \leq c \leq 0.5$ and $[B^c(1-B)^{1-c}2a] > 1)$ or $(0.5 < c \leq 1$ and $[B^c(1-B)^{1-c}a] > 1$.

Proof: Suppose $0.5 \leq c \leq 1$. Then,

$$\lim_{N\to\infty} E[Y] = Nf \sum_{k=0}^{c \log N} \binom{\log N}{k} A^k (1-A)^{\log N - k}$$

$$\geq f \lim_{N\to\infty} a^{\log N} A^{c \log N} (1-A)^{(1-c)\log N} \sum_{k=0}^{c \log N} \binom{\log N}{k}$$

$$\geq f \lim_{N\to\infty} a^{\log N} A^{c \log N} (1-A)^{(1-c)\log N} \left[c \sum_{k=0}^{\log N} \binom{\log N}{k} \right]$$

$$= fc \lim_{N\to\infty} A^{c \log N} (1-A)^{(1-c)\log N} a^{\log N} 2^{\log N}$$

$$= \infty \quad \text{if } \left[A^c (1-A)^{1-c} 2a \right] > 1 .$$

Now suppose that $0 \leq c < 0.5$. Then,

$$\lim_{N\to\infty} E[Y] \geq f \lim_{N\to\infty} a^{\log N} A^{c \log N} (1-A)^{(1-c)\log N} \sum_{k=0}^{c \log N} \binom{\log N}{k}$$

$$\geq f \lim_{N\to\infty} a^{\log N} A^{c \log N} (1-A)^{(1-c)\log N}$$

$$= \infty \quad \text{if } \left[A^c (1-A)^{1-c} a \right] > 1 .$$

From the proof of Lemma 4.2, it is known that $\lim_{N\to\infty} E[Y]$ has the same form as $\lim_{N\to\infty} E[X]$ with f replaced by $(1-f)$, c replaced by $(1-c)$, and A replaced by $(1-B)$. When these replacements are made, part (2) of the theorem follows. **Q.E.D.**

Theorem 4.5: Given $\gamma = \log N$, all category 3A probabilistic diagnosis algorithms produce 0% correct diagnosis as $N \to \infty$ if there exists a value c $(0 \leq c \leq 1)$ such that either part (1) or part (2) of Lemma 4.3 is satisfied.

Proof: Consider any category 3A probabilistic diagnosis algorithm. Let z_{th} be the threshold value used by the algorithm. Suppose z_{th} is not a linear function of $\gamma = \log N$. If z_{th} grows slower than a linear function, then as $N \to \infty$, it is possible to choose a constant c between 0 and 1 (depending on B) such that $z_{th} < c \log N$ and part (2) of Lemma 4.3 is satisfied for any B value. If z_{th} grows faster than a linear function, then as $N \to \infty$, it is pos-

sible to choose a constant c between 0 and 1 (depending on A) such that $z_{th} > c \log N$ and part (1) of Lemma 4.3 is satisfied for any A value.

Now suppose z_{th} is a linear function of $\gamma = \log N$. Then, as $N \rightarrow \infty$, there exists a value c $(0 \leq c \leq 1)$ such that $(c - \varepsilon)\log N \leq z_{th} \leq (c + \varepsilon)\log N$ for any $\varepsilon > 0$. Thus, either $E[Y] \rightarrow \infty$ or $E[X] \rightarrow \infty$. **Q.E.D.**

Theorem 4.5 only places an upper bound on A and a lower bound on B such that no category 3A probabilistic diagnosis algorithm is able to produce correct diagnosis as $N \rightarrow \infty$. However, if part (1) of Lemma 4.3 is satisfied for a sufficiently small c value or if part (2) of Lemma 4.3 is satisfied for a sufficiently large c value, then it is likely to be the case that no diagnosis algorithm can achieve correct diagnosis as $N \rightarrow \infty$. To see the reason for this observation, suppose part (1) of Lemma 4.3 is satisfied for $c = 0$. This says that as $N \rightarrow \infty$, the number of faulty nodes with no one-links incident on them goes to infinity. It is unlikely that any diagnosis algorithm will be able to correctly diagnose *all* of these faulty nodes. Alternatively, suppose part (2) of Lemma 4.3 is satisfied for $c = 1$. Then as $N \rightarrow \infty$, the number of non-faulty nodes which are tested to be faulty by all of their testing neighbors goes to infinity. It is unlikely that any diagnosis algorithm will be able to correctly diagnose *all* of these non-faulty nodes. Simulation results with the MPD algorithm in Section 4.6 support these observations.

Let us use examples to illustrate the utility of the bounds developed in this section. Let us assume comparison-testing is being used. Suppose $\gamma = \log N$ with $a = 2$, i.e., a hypercube network (hypercubes are formally defined in Section 6.3.1). Let us assume a category 3A probabilistic diagnosis algorithm with threshold value $z_{th} = c \log N$ and $c = 0.5$. Then, sufficient conditions for asymptotically correct diagnosis are $A > 0.9375$ and $B < 0.0625$. Necessary conditions for asymptotically correct diagnosis are $A > 0.9330$ and $B < 0.0670$. Thus, $p = 0.937$ and $f = 0.05$ works but $p = 0.927$ and $f = 0.075$ does not. If $c = 0.25$,

sufficient conditions for asymptotically correct diagnosis are $A > 0.8425$ and $B < 0.0039$. Necessary conditions are $A\ (1-A)^3 < 2^{-4}$ and $B\ (1-B)^3 < 4^{-4}$. Thus, $p = 0.842$ and $f = 0.004$ works but $p = 0.495$ and $f = 0.009$ does not.

All of the bounds developed have been for testing graphs with $\gamma = \log N$. If γ grows faster than $\log N$, it has already been shown in [6] that 100% correct diagnosis can be achieved as $N \rightarrow \infty$ as long as $A > B$. In addition, Blough [6] has shown that if γ grows slower than $\log N$, no diagnosis algorithm can achieve 100% correct diagnosis as $N \rightarrow \infty$.

4.6. Simulation Results

Simulations were conducted to evaluate the performance of the diagnosis algorithms studied. However, since *optimal* diagnosis algorithms have been designed for each category of probabilistic diagnosis studied, the results of simulations are not as important as they would be for heuristic diagnosis algorithms. This is pointed out because it was already shown in [45] that the DSK* algorithm achieves diagnostic accuracy very close to that of the globally optimal MPD algorithm for all testing graphs on which simulations were attempted. Therefore, our simulation results will not show that our optimal category 2 or 2A probabilistic diagnosis algorithms do much better than DSK*. Due to the random nature of the simulations, it is possible for OPT2 and OPT2A to perform worse than DSK* in some cases. However, given a sufficiently large enough testing graph and appropriate probability parameter values, it is expected that Algorithms OPT2 and OPT2A *will* perform significantly better than Algorithm DSK*.

The simulations were conducted on a Sun 3/280 for hypercubes of dimension six through ten. The same experimental setup as in [45] was used. In assigning prior fault probability values, an exponential failure arrival rate was assumed. For each node u_i, a time value τ_i, corresponding to the length of time u_i has been in the system, was generated from a uniform

distribution over the interval $[0, T]$. Then $f_i = 1 - e^{-\lambda\tau_i}$ was assigned, where $\lambda = \text{MTTF}^{-1}$ is the mean failure arrival rate. $T = 10^3$ hours and the three MTTF values 10^4, 5×10^4, and 10^5 hours were used, resulting in λT values of 0.1, 0.02, and 0.01 respectively. The same simulation results can be obtained by decreasing T and increasing λ by the same factor. Thus, this can model components which have been in the system for different lengths of times and components which have different MTTF values (their τ_i values can be appropriately adjusted). The $E[f_i]$ and $E[|f_j - f_i|]$ values calculated are shown in Table 4.3 below. The equations for $E[f_i]$ and $E[|f_j - f_i|]$ are

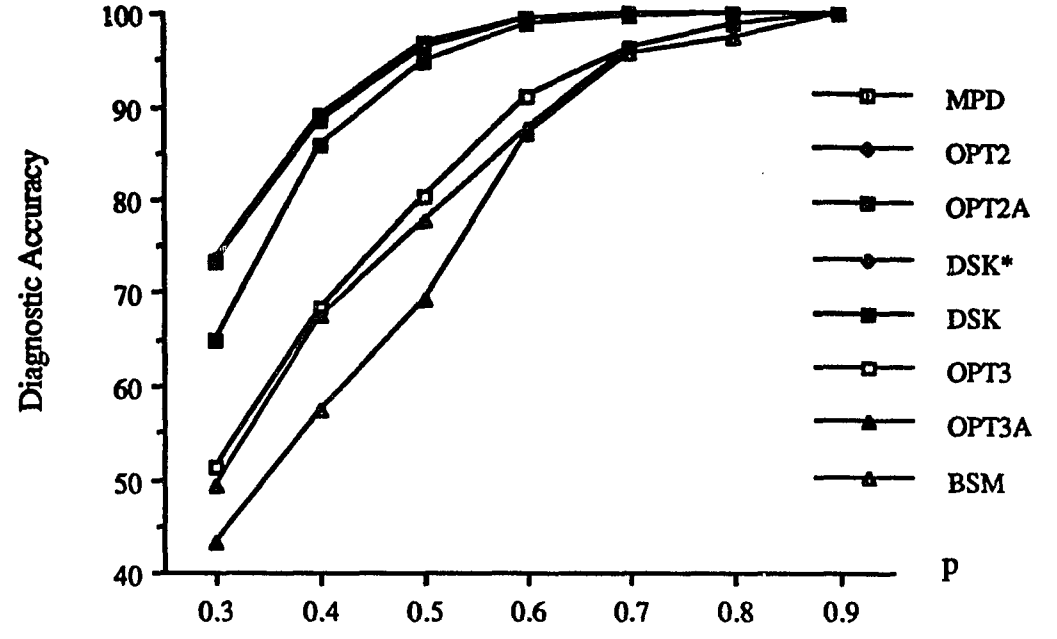
$$E[f_i] = 1 + \frac{e^{-\lambda T} - 1}{\lambda T},$$

$$E[|f_j - f_i|] = \frac{2}{\lambda T}\left[1 + e^{-\lambda T}\right] + \frac{4}{(\lambda T)^2}\left[e^{-\lambda T} - 1\right].$$

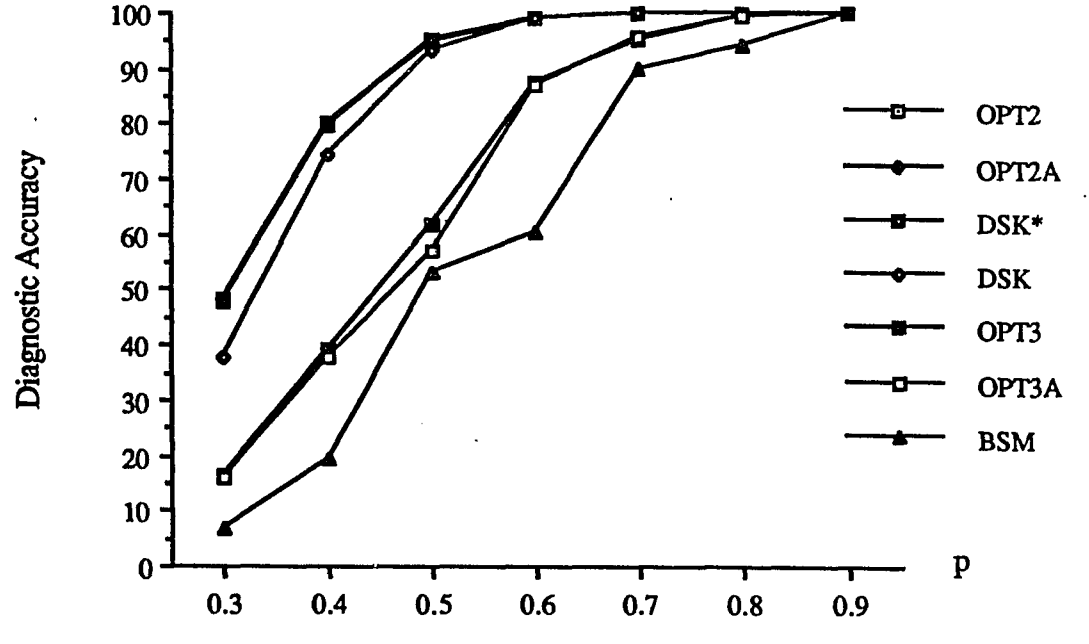
MTTF	$E[f_i]$	$E[\|f_j - f_i\|]$
10,000	0.0484	0.0317
50,000	0.0099	0.0066
100,000	0.0050	0.0033

Table 4.3: Prior fault probability mean and mean difference values.

For each of the four diagnosis algorithms BSM, DSK, DSK*, and MPD, 1000 syndromes were produced and diagnosed assuming p values of 0.3 to 0.9 (in 0.1 increments). Each node in the hypercube was assigned to test each of its immediate neighbors. To accommodate all diagnosis algorithms, comparison testing was implicitly assumed and the method described in Section 4.2 was used to calculate r and s values and to produce the directed testing graph. Let Q_r denote a hypercube of dimension r. The diagnosis algorithm MPD was only executed on hypercubes Q_6 through Q_8 with MTTF = 50K hours because of its high computational cost. Figs. 4.1(a) and 4.1(b) show the results of the simulations on a Q_8 and Q_{10} with MTTF = 50K hours. Similar results were obtained for all hypercube dimensions and



(a)



(b)

Figure 4.1. Diagnostic accuracy on (a) Q_8 and (b) Q_{10} with MTTF = 50K hours.

MTTF values used. From Figs. 4.1(a) and 4.1(b), it is hard to see how the DSK*, OPT2, OPT2A, and MPD algorithms compare because their diagnostic accuracy values are so close together. Therefore, in Table 4.4, the relative performance of the DSK, DSK*, OPT2A, and OPT2 algorithms with respect to the globally optimal algorithm MPD for a Q_8 with MTTF = 50K hours and p values from 0.3 to 0.9 is shown. Denoting the the diagnostic accuracy of Algorithm A by $DA(A)$, the formula for the relative performance of Algorithm A is $\frac{DA(A) - DA(MPD)}{DA(MPD)} \times 100\%$.

p	0.3	0.4	0.5	0.6	0.7	0.8	0.9
DSK	-11.4%	-3.6%	-2.1%	-0.6%	-0.2%	0.0%	0.0%
DSK*	-0.1%	-0.6%	-0.2%	0.0%	0.0%	0.0%	0.0%
OPT2A	-0.2%	-0.4%	-0.5%	0.0%	0.0%	0.0%	0.0%
OPT2	+0.1%	+0.1%	0.0%	0.0%	0.0%	0.0%	0.0%

Table 4.4: Diagnostic accuracy with respect to MPD for Q_8 with MTTF = 50K hours.

Experiments were also conducted to show what happens when incorrect probability parameter values are used. Although the same probability parameter values as in the previous simulations were used by the diagnosis algorithms, an error of -30% in each probability parameter value was introduced in generating the set of faulty nodes and the fault syndromes. Fig. 4.2 shows the results of these simulations for a Q_8 with MTTF = 50,000 hours. Our analysis in Section 4.4.2 showed that even when inaccurate probability parameter values are used, the OPT3 and OPT3A algorithms perform almost as well as they would with the correct probability parameter values and the OPT2 and OPT2A algorithms perform as well or better than the DSK* and DSK algorithms. The simulation results shown in Fig. 4.2 support our analytical results.

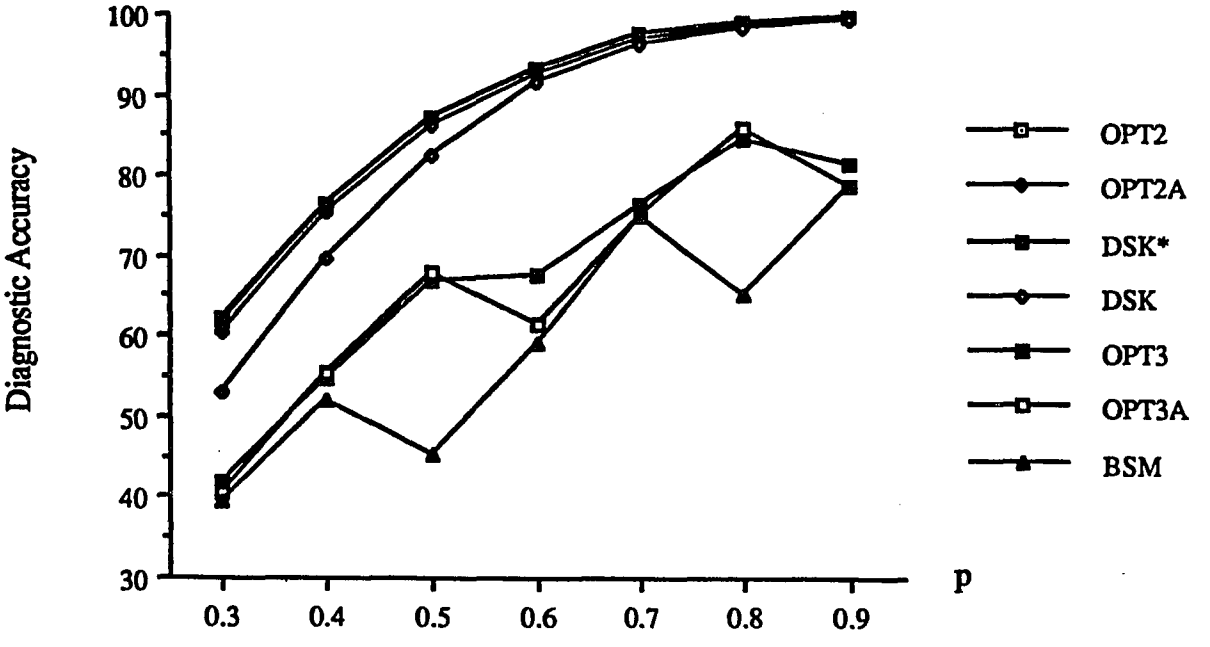


Figure 4.2. Diagnostic accuracy with 30% parameter errors on Q_8 with MTTF = 50K hours.

4.7. Conclusion

In this chapter, several important theoretical and practical results concerning the diagnosis of a multiprocessor/multicomputer system based on inter-processor tests with imperfect fault coverage have been presented. Necessary and sufficient conditions on probability parameter values have been shown such that 100% correct diagnosis can be achieved as $N \rightarrow \infty$ in systems where the number of tests per processor grows as $\log N$. Optimal diagnosis algorithms have been derived for four important categories of diagnosis defined previously in Chapter 2. It is shown that these optimal diagnosis algorithms perform well even when inaccurate probability parameter estimates are used.

Probabilistic analysis has been used to compare the diagnosis capabilities of Dahbura *et. al.*'s (DSK) probabilistic diagnosis algorithm, Blough *et. al.*'s (BSM) probabilistic diagnosis algorithm, and our optimal probabilistic diagnosis algorithms. This analysis shows that the DSK algorithm is superior to the BSM algorithm because it approximates the optimal category 2A diagnosis algorithm better than the BSM algorithm. The comparative analysis is supported by our simulation results. Based on the comparative analysis, the DSK* algorithm, which is a modified form of the DSK algorithm, has been derived. The DSK* algorithm performs significantly better than the DSK algorithm and achieves close to optimal performance in many instances. The DSK* algorithm has an advantage over the optimal category 2A algorithm in that it is only dependent on relative prior fault probability values.

CHAPTER 5

OPTIMAL MULTIPLE SYNDROME DIAGNOSIS

This chapter addresses diagnosis using multiple fault syndromes. It is shown that by using multiple syndromes, significantly better diagnosis can be achieved than by using a single syndrome, even when the amount of time devoted to testing is the same. A *multiple syndrome diagnosis* algorithm, in which testing is conducted in stages and a fault syndrome is collected after each testing stage, is derived which is *optimal* in the level of diagnostic accuracy achieved (among diagnosis algorithms of a certain type to be defined) and produces good results even with sparse interconnection networks and inter-processor tests with low fault coverage. Furthermore, upper and lower bounds are proven on the number of fault syndromes required to asymptotically produce 100% correct diagnosis as $N \rightarrow \infty$. Our solution and another multiple syndrome diagnosis solution (by Fussell and Rangarajan [30]) are evaluated both analytically and with simulations.

5.1. Introduction

The previous two chapters addressed the optimal diagnosis of multiprocessor/multicomputer systems based on the analysis of a single syndrome. What happens if multiple syndromes are used? Is diagnostic accuracy improved by using multiple syndromes instead of a single syndrome if the same amount of time is devoted to testing in both cases? In [55], Rangarajan and Fussell gave examples to show that the additional information available when multiple syndromes are used permits correct diagnosis in some fault situations

where an "accumulated" single syndrome results in incorrect diagnosis. These examples were dependent on the assumption that faulty processors will sometimes give correct results. However, a more fundamental reason for the better diagnostic capability possible with multiple fault syndromes is the testing method used. In Section 5.3, it is shown that by using a special method of testing to generate the multiple syndromes, significantly higher diagnostic accuracy can be achieved than by using only a single syndrome.

This chapter addresses multiple syndrome diagnosis for a variant of category 3A diagnosis assuming that comparison-testing is being used as the inter-processor testing method. Although it is possible to derive multiple syndrome diagnosis algorithms for all of the categories defined in Chapter 2, it is not always beneficial to do this. For instance, with category 1 diagnosis, it has already been shown that most probable diagnosis is NP-hard even with a single syndrome. In addition, an extremely high communication overhead is required to reliably distribute the syndrome information to all of the nodes. These problems are compounded when multiple syndromes are used. For category 2 and 2A diagnosis, because of the particular testing method used (described in Section 5.3), the diagnostic accuracy achievable with multiple syndrome diagnosis is no higher than for categories 3 and 3A. The calculations for category 3 diagnosis are significantly more complex than for category 3A diagnosis with no appreciable increase in diagnostic accuracy. Finally, the multiple syndrome diagnosis method developed is limited to comparison-testing because of the way in which the multiple syndromes are formed (described to Section 5.3).

Several authors [5, 19] have presented fast probabilistic diagnosis algorithms which achieve correct diagnosis with high probability given intermittently faulty processors. Blough *et. al.* [5] showed that they could asymptotically achieve 100% correct diagnosis in an N processor system as $N \rightarrow \infty$ provided that $\alpha(N) \log N$ tests were performed on each processor, where $\alpha(N) \rightarrow \infty$ arbitrarily slowly as $N \rightarrow \infty$. In Blough *et. al.*'s method, the number

of tests on processor u_i is equivalent to the number of processors testing u_i. Fussell and Rangarajan [30] improved on [5] by showing that the same asymptotic result can be obtained for systems with lower connectivity (e.g., meshes or rings) if each pair of processors conducts multiple tests and the number of *these* tests on each processor grows faster than $\log N$. Fussell and Rangarajan's algorithm can be viewed as a multiple syndrome diagnosis algorithm.

This chapter improves upon Fussell and Rangarajan's (FR) algorithm [30] by deriving a multiple syndrome diagnosis algorithm which is *optimal* in the level of diagnostic accuracy achieved. Since multiple syndromes can be formed in many different ways and since many different types of syndrome information can be used in the diagnosis, a specific category of multiple syndrome diagnosis (of which the FR algorithm is a member) is defined and our analysis is restricted to this category. Our diagnosis algorithm is provably *optimal* among all multiple syndrome diagnosis algorithms which use the same type of syndrome information as the FR algorithm. In addition, our optimal multiple syndrome diagnosis algorithm has the same desirable asymptotic properties as the FR algorithm. Upper and lower bounds on the number of tests required for asymptotically correct diagnosis are shown.

5.2. Background

In this chapter, inter-processor testing is assumed to be done by comparison-testing, in which a test between two processors u_i and u_j is actually a comparison of the outputs of two identical tasks. Thus, the testing graph is assumed to be an undirected graph (the edges $e_{ij} \in E_T$ are assumed to be undirected edges representing comparison tests). For an undirected testing graph, $\Gamma(u_i) = \Gamma^{-1}(u_i)$. Thus, $\Gamma(u_i)$ refers to all of the testing neighbors of u_i and $d(u_i) = |\{u_j \in \Gamma(u_i) : a_{ij} = 1\}|$.. Also, for comparison-testing, fault coverage p has a slightly different meaning. Given a node $u_i \in V_T$ and a test task t_k, the fault coverage p_{ik} is the probability that u_i produces an incorrect result for task t_k given that u_i is faulty. Finally,

in this chapter, the actual set of faulty nodes which are to be diagnosed is denoted by F'.

The testing methods used in single and multiple syndrome diagnoses are referred to as *single syndrome testing* and *multiple syndrome testing*, respectively. Most of the previous work on diagnosis based on comparison-testing have assumed a single syndrome testing method. In single syndrome testing, it is assumed that the comparison test between a node u_i and another node $u_k \in \Gamma(u_i)$ is independent of any other comparison test. If two nodes u_i and u_j execute and compare more than one task, then $a_{ij} = a_{ji} = 1$ if any of the task outputs are different for the two nodes. The accumulated syndrome formed in this manner is the syndrome used by a single syndrome diagnosis algorithm

In multiple syndrome testing, testing is done in stages, and in each testing stage, it is assumed that the same task is used in the comparison tests between a node u_i and nodes in $\Gamma(u_i)$. A "new" fault syndrome is formed after each testing stage using the same testing graph. In a single testing stage, each processing node is assigned at most one task to execute. Thus, all nodes in the same connected component of the testing graph must execute the same test task in a testing stage. In diagnosis based on multiple-syndrome testing, the fact that the syndromes obtained in later testing stages are partially dependent on the syndromes obtained in earlier testing stages can be used. The number of testing stages used in multiple syndrome testing is denoted as R.

To obtain an efficient and practical diagnosis algorithm, algorithms in which each node u_i is only aware of the results of its tests with its neighbors, referred to as local syndrome information in Chapter 2, are considered. For multiple syndrome diagnosis, summarized local syndrome information for a node u_i can be written as $\{d^k(u_i) : 0 \leq k \leq R\}$, where $d^k(u_i) = d(u_i)$ for testing stage k. There are two dimensions to the syndrome information: one dimension is $d^k(u_i)$ for a fixed k and the other dimension is the number of testing stages in which $d^k(u_i)$ is greater than a fixed threshold. Given an integer m, *m-threshold local*

syndrome information is defined as $|\{k : 0 \leq k \leq R \text{ and } d^k(u_i) > m\}|$. *Category 3, 3A,* and *3AM* diagnosis are defined as diagnosis using local, summarized local, and m-threshold local (for any fixed m) syndrome information, respectively. This categorization is shown in Table 5.1, which is an extension of the categorization of Chapter 2. The diagnosis algorithm derived in this chapter is the optimal category 3AM multiple syndrome diagnosis algorithm.

Category	Syndrome Information	Interpretation
3	Local	a_{ji} values for $u_j \in \Gamma(u_i)$
3A	Summarized local	$\{d^k(u_i) : 0 \leq k \leq R\}$
3AM	m-threshold local	$\|\{k : 0 \leq k \leq R \text{ and } d^k(u_i) > m\}\|$

Table 5.1: Categorization of diagnosis using local syndrome information.

The FR algorithm [30] is characteristic of category 3AM diagnosis. In the FR algorithm, testing is conducted in stages and two thresholds kv_i and sv_i are used. In testing stage i, it is assumed that all processors execute the same test task t_i. Let $T = \{t_1, \cdots, t_R\}$ be the set of R test tasks executed on all processors and M be the number of distinct faulty results for a test task, where all tasks are treated identically.

Algorithm FR:

0. Let $F \leftarrow \emptyset$ be the set of diagnosed faulty nodes;
1. **For** each $u_j \in V_T$ **do**
   $kv_j \leftarrow |\Gamma(u_j)| - 1$;
   $sv_j \leftarrow R - k_r R(1 - p(1 - \frac{p}{M})^{|\Gamma(u_j)|})$, where $1 \leq k_r \leq 2$;
2. **For** each $t_i \in T$ **do**
   for each $u_j \in V_T$ **do**
   if $d(u_j) > kv_j$
   then $L(i, j) = 1$;
   else $L(i, j) = 0$;
3. **For** each $u_j \in V_T$ **do**
   if $\sum_{t_i \in T} L(i, j) > sv_j$
   then $F \leftarrow F \cup \{u_j\}$;

In Step 1 of the description of the FR algorithm, kv_j is chosen to be $|\Gamma(u_j)| - 1$ and a range of values is indicated as being acceptable for the choice of sv_j. These thresholds were simply

chosen in order for the the algorithm to satisfy desirable asymptotic properties. The authors proved that as $N \rightarrow \infty$, the diagnostic accuracy of the FR algorithm asymptotically approaches 100%. An earlier algorithm by the same authors [55] can be considered to be the same as the FR algorithm with $sv_j = 0$ for all $u_j \in V_T$.

5.3. Analysis of Multiple Syndrome Testing

The two main differences between multiple and single syndrome testing are the use of multiple versus single syndromes and the constrained manner in which the syndromes are formed in multiple syndrome testing. In [55], examples are given to show that the additional information available when multiple syndromes are used permits correct diagnosis in some fault situations where an "accumulated" single syndrome results in incorrect diagnosis. In this section, it is shown that the way in which syndromes are formed in multiple and single syndrome testing also results in a significant difference in diagnostic capability.

Suppose all of the syndromes associated with the multiple testing stages in a multiple syndrome testing method are used to form an updated syndrome. The single syndrome generated using this process has the property that for a given node $u_i \in V_T$, all of the tests e_{ik} for $u_k \in \Gamma(u_i)$ use the same set of tasks in their testing. However, the syndrome used in single syndrome testing has no such restriction. In single syndrome testing, it is assumed that the test $e_{ik} \in E_T$ is independent of all of the other tests in E_T. As will be shown shortly, this "small" difference in testing method results in a significant difference in diagnostic capability.

To get a direct comparison, let us compare the difference in diagnostic capability between single and multiple syndrome testing when only one syndrome is used in the multiple syndrome testing method. For simplicity of analysis, average parameter values will be used and a regular testing graph with node-degree γ will be assumed. The probability analysis for single syndrome testing has been done in [46]. Given a node u_i and any node $u_j \in \Gamma(u_i)$, let

$A = P(a_{ji} = 1 \mid \delta_i) = p(1-f) + f(1-\frac{p}{M})) + (1-p)(fp) = (1-f)\,p + fp(2-p-\frac{p}{M})$

and $B = P(a_{ji} = 1 \mid \bar{\delta}_i) = fp$. For single syndrome testing, the probability of having $z = d(u_i)$ one-links incident on u_i out of a maximum of γ links (denoted z *one–links* : γ) given that u_i is faulty and non-faulty are

$$P(z \text{ one–links} : \gamma \mid \delta_i) = \binom{\gamma}{z} A^z (1-A)^{\gamma - z} \quad , \tag{5.1a}$$

$$P(z \text{ one–links} : \gamma \mid \bar{\delta}_i) = \binom{\gamma}{z} B^z (1-B)^{\gamma - z} \quad . \tag{5.1b}$$

It follows (after simplification) that

$$P(\delta_i \mid z \text{ one–links} : \gamma) = \frac{1}{1 + \frac{1-f_i}{f_i}\left[\frac{B}{A}\right]^z \left[\frac{1-B}{1-A}\right]^{\gamma - z}} \quad . \tag{5.1c}$$

For multiple syndrome testing with a single syndrome, the probability that there are z one-links incident on a node $u_i \in V_T$ given that u_i is non-faulty and faulty are:

$$P(z \text{ one–links} : \gamma \mid \bar{\delta}_i) = \sum_{j=z}^{\gamma} \binom{\gamma}{j} f^j (1-f)^{\gamma - j} \binom{j}{z} p^z (1-p)^{j-z} \tag{5.2a}$$

$$= h(z) \quad ,$$

$$P(z \text{ one–links} : \gamma \mid \delta_i) = p \sum_{j=0}^{z} \binom{\gamma}{j} (1-f)^j f^{\gamma - j} \binom{\gamma - j}{z - j} (1-\frac{p}{M})^{z-j} (\frac{p}{M})^{\gamma - z}$$

$$+ (1-p)\, h(z) \tag{5.2b}$$

$$= p\, g(z) + (1-p)\, h(z) \quad .$$

Thus, the posterior fault probability of u_i given z one-links incident on u_i is

$$P(\delta_i \mid z \text{ one–links} : \gamma) = \frac{f_i \left[p\, g(z) + (1-p)\, h(z)\right]}{f_i \left[p\, g(z) + (1-p)\, h(z)\right] + (1-f_i)\, h(z)} \tag{5.2c}$$

$$\approx \begin{cases} 1 & \text{if } z = \gamma, \\ f_i(1-p)/(1-f_i p) & \text{if } z \neq \gamma. \end{cases}$$

The approximation holds if M is large and $\gamma \leq 4$. Then, Eq. (5.2c) is close to a delta function with a spike at $z = \gamma$ since f (prior fault probability) values are typically fairly small. If $\gamma > 4$, then Eq. (5.2c) becomes close to a step function. In general, Eq. (5.1c) is a much more smoothly increasing function of z than Eq. (5.2c).

Let us consider the testing time required for single versus multiple syndrome testing. Suppose that it takes τ units of time to execute each task and that each task has the same level of fault coverage p. Then, for multiple syndrome testing using a single syndrome, it takes τ units of time to obtain the syndrome since each node executes at most one task. In single syndrome testing, $\tau\,\gamma$ units of time are required for testing since each node executes γ tasks (in order to make γ comparison tests). Thus, using the same amount of testing time, γ test tasks (treated as one "large" test task) can be executed in one testing stage of the multiple syndrome testing method, thereby achieving an effective fault coverage of $1 - (1-p)^{\gamma}$.

Fig. 5.1 shows the distributions of $z = d(u_i)$ given u_i faulty and u_i non-faulty for both multiple syndrome testing (Eqs. (5.1a) and (5.1b)) and single syndrome testing (Eqs. (5.2a) and (5.2b)) using $M = 1000$, $\gamma = 4$, $p = 0.4$, and two different values of f. The fault coverage value p used for single syndrome testing is actually $1 - (1-p)^{\gamma} = 0.87$ since this level of fault coverage can be obtained in the same amount of testing time required to achieve fault coverage of p for multiple syndrome testing. Naturally, a high level of diagnostic accuracy can be achieved if the syndrome information perceived when u_i is faulty is drastically different from the syndrome information perceived when u_i is non-faulty. From our analysis, it can be seen that the syndrome used in multiple syndrome testing fits this mold much more closely than the syndrome used in single syndrome testing.

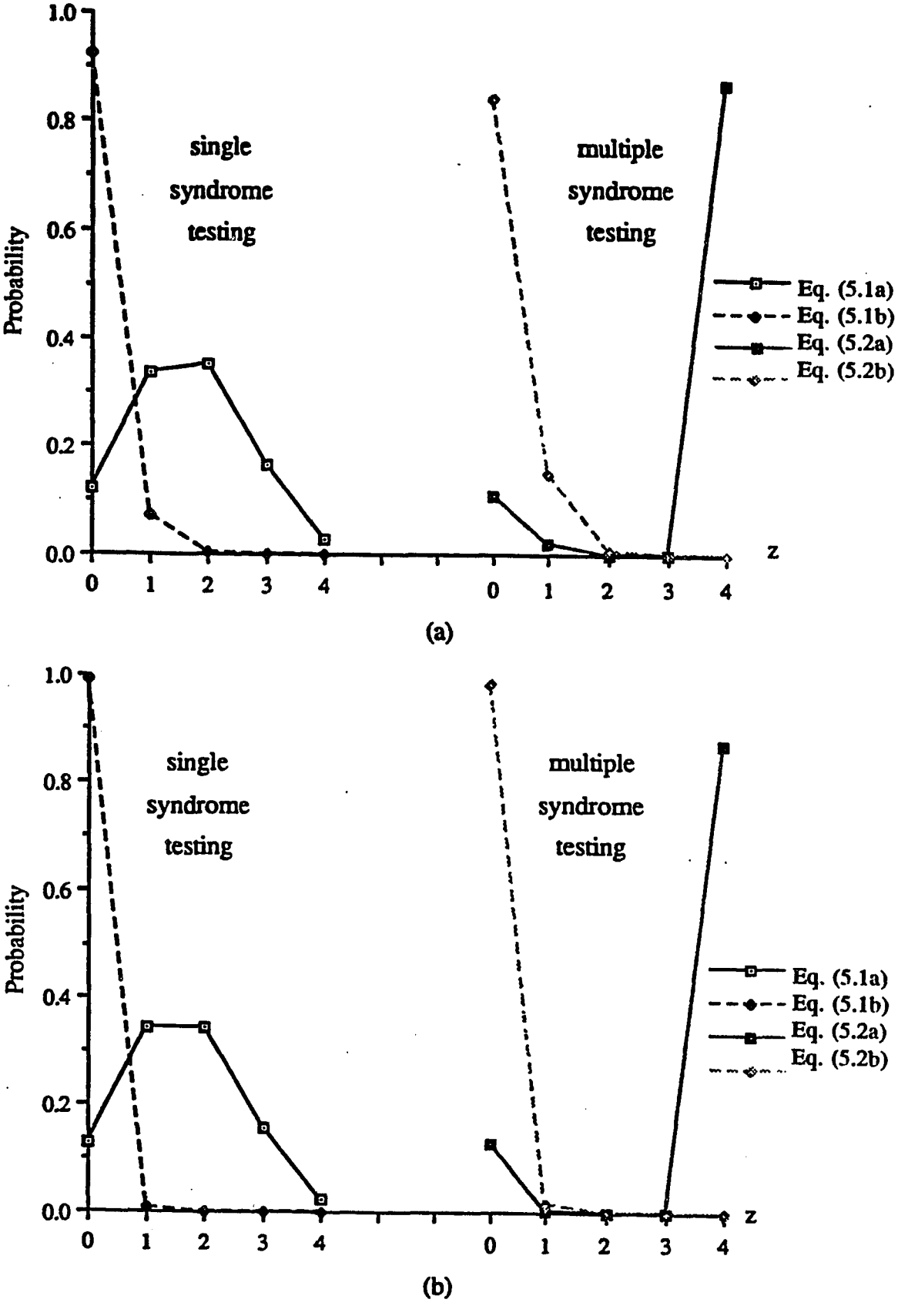


Figure 5.1. Probability distributions with (a) $f = 0.0484$ and (b) $f = 0.0050$.

5.4. Optimal Multiple Syndrome Diagnosis Method

In Theorem 4.1 of Chapter 4, it was proven that for any category of diagnosis, the optimal diagnosis method is to make the most probable diagnosis at each node given the type of syndrome information available. Thus, to derive the optimal category 3AM multiple syndrome diagnosis algorithm, the posterior probability of a node $u_i \in V_T$ being faulty given the syndrome information for u_i must be analyzed. Category 3AM multiple syndrome diagnosis is based on m-threshold local syndrome information for any fixed m. Under the assumption that the test evaluation by a non-faulty processor is at least as good as the test evaluation by a faulty processor, the posterior fault probability of u_i is a non-decreasing function of $d^k(u_i)$ and also of $|\{k : 0 \le k \le R \text{ and } d^k(u_i) > m\}|$ for any m. Then, the optimal category 3AM multiple syndrome diagnosis algorithm is based on the selection of two thresholds. The first threshold z_{th_i} is for the number of one-links incident on u_i during a single testing stage. The second threshold H_{th_i} is for the number of testing stages in which a given node passes the first threshold. In the FR algorithm, $z_{th_i} = kv_i = \gamma - 1$ and $H_{th_i} = sv_i$. The thresholds kv_i and sv_i were simply chosen so that desirable asymptotic properties of the algorithm could be proven.

Optimal threshold values can be obtained by calculating posterior fault probabilities. The optimal choice for z_{th_i} is obtained using Eq. (5.2c). The optimal z_{th_i} value, denoted $z^*_{th_i}$, is equal to z such that $P(\delta_i \mid z \text{ one-links} : \gamma) \le 0.5$ and $P(\delta_i \mid z+1 \text{ one-links} : \gamma) > 0.5$. Although calculation of Eq. (5.2c) for large values of γ is computationally expensive, since γ is at most the node-degree of the processor interconnection network, very large γ values will not be needed for most practical partially-connected systems. Also, from the analysis done in Section 5.3 (approximation for Eq. (5.2c)), it is evident that when $\gamma \le 4$, $z^*_{th_i} = \gamma - 1$. Since Eq. (5.2c) is a monotonically non-decreasing function of z, one can obtain $z^*_{th_i}$ for $\gamma > 4$ by evaluating Eq. (5.2c) for several values of z near γ.

Probability analysis is now used to derive $H^*_{th_i}$, the optimal value of H_{th_i}. For simplicity of analysis, it will be assumed that $\gamma \leq 4$ so that $z^*_{th_i} = \gamma - 1$. (The changes required in the analysis when $\gamma > 4$ is discussed at the end of this section.) For a single syndrome,

$$P(\gamma \text{ one-links} : \gamma \mid \delta_i) = \begin{cases} A_1 \approx p + (1-p)\,p^\gamma & \text{if } \Gamma(u_i) \subseteq F' \\ A_2 \approx p & \text{otherwise} \end{cases}, \tag{5.3}$$

$$P(\gamma \text{ one-links} : \gamma \mid \bar{\delta}_i) = \begin{cases} B_1 = p^\gamma & \text{if } \Gamma(u_i) \subseteq F' \\ B_2 = 0 & \text{otherwise} \end{cases}. \tag{5.4}$$

The approximations in Eq. (5.3) hold since M is assumed to be large, that is,

$$A_1 = p(1 - \frac{p}{M})^\gamma + (1-p)\,p^\gamma \approx p + (1-p)\,p^\gamma ,$$

$$p(1 - \frac{p}{M})^{\gamma-1} \leq A_2 \leq p \quad \Rightarrow \quad A_2 \approx p .$$

If $\Gamma(u_i) \subseteq F'$, then A_1 and B_1 are the probabilities of having greater than $z^*_{th_i}$ one-links incident on u_i given that u_i is faulty and non-faulty, respectively. A_2 and B_2 are the same probabilities when $\Gamma(u_i) \subseteq V_T - F'$. Next, the probabilities of having H syndromes in which $d(u_i) > z^*_{th_i}$ (denoted *H passes*) given that u_i is faulty and non-faulty are

$$P(H \text{ passes} \mid \delta_i) = f^\gamma \binom{R}{H} A_1^H (1-A_1)^{R-H} + (1-f^\gamma) \binom{R}{H} A_2^H (1-A_2)^{R-H} , \tag{5.5}$$

$$P(H \text{ passes} \mid \bar{\delta}_i) = f^\gamma \binom{R}{H} B_1^H (1-B_1)^{R-H} + (1-f^\gamma) \binom{R}{H} B_2^H (1-B_2)^{R-H} . \tag{5.6}$$

Finally, the posterior fault probability of u_i given that $d(u_i) > z^*_{th_i}$ for H syndromes is

$$P(\delta_i \mid H \text{ passes}) = \frac{P(H \text{ passes} \mid \delta_i)\,P(\delta_i)}{P(H \text{ passes} \mid \delta_i)\,P(\delta_i) + P(H \text{ passes} \mid \bar{\delta}_i)\,P(\bar{\delta}_i)} . \tag{5.7}$$

The value of H at which $P(\delta_i \mid H \text{ passes}) = 0.5$ is the optimal H_{th_i} value, $H^*_{th_i}$. It is difficult to determine $H^*_{th_i}$ directly because of the form of Eq. (5.7). However, since $H^*_{th_i}$ is to be used as a threshold for an integer quantity, it is only necessary to determine $\bar{H}_{th_i} = \lfloor H^*_{th_i} \rfloor$.

Eq. (5.7) is a monotonically non-decreasing function of H. Thus, $\bar{H}_{th_i}$ can be determined by calculating Eq. (5.7) for several values of H. This process is made simpler if a close upper bound for $H^*_{th_i}$ can be calculated. Denoting this upper bound by $\hat{H}_{th_i}$,

$$\hat{H}_{th_i} = \frac{\log\left[\frac{1-f_i}{f_i}\right]}{\log\left[\frac{A_1(1-B_1)}{(1-A_1)B_1}\right]} + R\,\frac{\log\left[\frac{1-B_1}{1-A_1}\right]}{\log\left[\frac{A_1(1-B_1)}{(1-A_1)B_1}\right]} . \tag{5.8}$$

Theorem 5.1: $\hat{H}_{th_i}$ is an upper bound for $H^*_{th_i}$.

Proof: Fig. 5.2 shows the distributions of three random variables Z_0, Z_1, and Z_2. Z_0 denotes the number of testing stages (out of R) that a faulty node u_i has $d(u_i) > z^*_{th_i}$. Thus, the distribution for Z_0 has Eq. (5.5) as its probability mass function. Z_1 is the binomial random variable with parameters R and A_1. Likewise, Z_2 is the binomial random variable with parameters R and A_2. Since $A_1 \geq A_2$, it is clear that the distribution for Z_1 is strictly to the right of the distribution for Z_2. Also, from the form of Eq. (5.5), it is evident that the distribution for Z_0 must lie in between the distributions for Z_1 and Z_2. This relationship is shown graphically in Fig. 5.2. Since $B_1 \geq B_2$, the distribution for Eq. (5.6) can be shown to reside between two analogous random variable distributions. Thus, the distributions for Eqs. (5.5) and (5.6) are both shifted to the right when A_2 and B_2 are replaced by A_1 and B_1, respectively. When these replacements are made, $\hat{H}_{th_i}$ is the value of H at which $\left[P(H \text{ passes} \mid \delta_i)\,P(\delta_i)\right] = \left[P(H \text{ passes} \mid \bar{\delta}_i)\,P(\bar{\delta}_i)\right]$. **Q.E.D.**

To calculate $\bar{H}_{th_i}$, the following procedure needs to be executed.

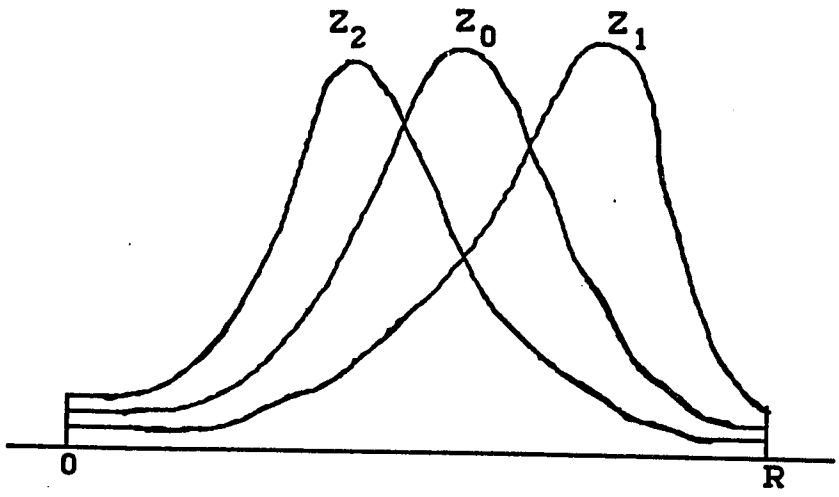


Figure 5.2. Distributions of random variables Z_0, Z_1, and Z_2.

Procedure Calc_H:

1. Calculate $\hat{H}_{th_i}$ using Eq. (5.8);
2. **For** H from 0 to $\min\{R\ , \lfloor \hat{H}_{th_i} \rfloor + 1\}$ **do**
   calculate $P(\delta_i \mid H\ passes)$ using Eq. (5.7);
3. $\bar{H}_{th_i} \leftarrow H'$ such that $P(\delta_i \mid H'\ passes) \leq 0.5$ and $P(\delta_i \mid H' + 1\ passes) > 0.5$.

Algorithm OPTM, the optimal category 3AM multiple syndrome diagnosis algorithm, is essentially the same as the FR algorithm except that the thresholds are chosen differently. This, however, is a crucial difference since the performance of the algorithm hinges upon the choice of thresholds. In the description of the FR algorithm in Section 5.2, it was assumed that all processors execute the same set of tasks $T = \{t_1\ , \cdots\ , t_R\}$. Since all tasks are treated identically in our analysis, the OPTM algorithm is also described under this assumption.

Algorithm OPTM:

1. **For** each $u_j \in V_T$ **doparallel**
   calculate $z^*_{th_j}$ using Eq. (5.2c);
   calculate $\bar{H}_{th_j}$ using procedure Calc_H;
2. **For** each $u_j \in V_T$ **doparallel**
   for each $t_i \in T$ **do**
   if $d(u_j) > z^*_{th_j}$
   then $L(i, j) = 1$;
   else $L(i, j) = 0$;
3. **For** each $u_j \in V_T$ **doparallel**
   if $\sum_{t_i \in T} L(i, j) > \bar{H}_{th_j}$
   then u_i is faulty;
   else u_i is non-faulty;

Although the OPTM algorithm is the optimal category 3AM diagnosis algorithm, it is not the optimal category 3 or even category 3A diagnosis algorithm. Since the effort required to obtain and use the syndrome information for category 3 or 3A is only slightly more than for category 3AM, it might seem worthwhile to derive the optimal category 3 and 3A multiple syndrome diagnosis algorithms. However, the diagnostic accuracy achieved by the OPTM

algorithm is very close to the best possible even when category 3 or 3A syndrome information is used.

To see the reason for this, let us refer to the analysis of Section 5.3 and Fig. 5.1. Given a non-faulty node u_i, it is most likely to not have any one-links incident on it. Given a faulty node u_i, if it fails a test task t_j (in other words, t_j *covers* the fault), then it is most likely to have γ one-links incident on it; otherwise, if u_i passes t_j, then it is most likely to not have any one-links incident on it. Therefore, in all cases, u_i will most likely either have γ one-links or zero one-links incident on it. Given this observation, the syndrome information used in category 3AM contains almost all of the important category 3A syndrome information. In simulations using the experimental setup to be described in Section 5.6, out of 80,000 syndromes generated for a 100-node torus-wrapped square mesh, there was exactly one syndrome in which a node u_i had neither $d(u_i) = \gamma$ nor $d(u_i) = 0$. Thus, the OPTM algorithm produced the optimal category 3A diagnosis over 99.998% of the time. Also, as explained in [46], the optimal category 3A diagnosis algorithm approximates the behavior of the optimal category 3 diagnosis algorithm since *average* probability parameter values are used in the analysis for category 3A. In summary, the OPTM algorithm is the optimal category 3AM multiple syndrome diagnosis algorithm and the "near-optimal" category 3A and category 3 multiple syndrome diagnosis algorithm.

In deriving the thresholds for the OPTM algorithm, the assumption that $\gamma \leq 4$ has been used. If $\gamma > 4$, then it is possible that $z^*_{tlh_i} < \gamma - 1$. In that case, the A_1, A_2, B_1, and B_2 values used in Eqs. (5.3) - (5.8) must be changed. Let $\Delta_i \equiv |\Gamma(u_i) \cap F| = x$. Then,

$$A_1' = P(d(u_i) > z^*_{tlh_i} : \gamma \mid \delta_i \, , \Delta_i = \gamma) \, , \cdots \, , A_{\gamma+1}' = P(d(u_i) > z^*_{tlh_i} : \gamma \mid \delta_i \, , \Delta_i = 0) \quad \text{and}$$

$$B_1' = P(d(u_i) > z^*_{tlh_i} : \gamma \mid \overline{\delta}_i \, , \Delta_i = \gamma) \, , \cdots \, , B_{\gamma+1}' = P(d(u_i) > z^*_{tlh_i} : \gamma \mid \overline{\delta}_i \, , \Delta_i = 0)$$

should be used instead of A_1, A_2, B_1, and B_2. A_1 and B_1 are replaced by A_1' and B_1', respec-

tively. A_2 is replaced by A_2' through $A_{\gamma+1}'$ and B_2 is replaced by B_2' through $B_{\gamma+1}'$. Note that Eqs. (5.3) and (5.4) will now have more additive terms. It may be possible to combine some of the A_k' or B_k' values as was done in the analysis for the case of $z^*_{tlh_i} = \gamma - 1$.

5.5. Asymptotic Analysis

One of the main desirable aspects of the FR algorithm was that it was shown to asymptotically achieve 100% correct diagnosis as $N \rightarrow \infty$ if $\gamma \geq 2$ and R grows faster than $\log N$. But, when $\bar{H}_{tlh_i}$ is calculated as in the previous section, it is noted that $\bar{H}_{tlh_i}$ is not necessarily one of the permitted values for sv_i in the FR algorithm (refer to Section 5.2). However, it is possible to directly prove that the OPTM algorithm also asymptotically achieves 100% correct diagnosis as $N \rightarrow \infty$. Let $\alpha(N)$ be any function of N such that $\lim_{N\rightarrow\infty} \alpha(N) = \infty$. In this section, it is proven that the OPTM algorithm asymptotically achieves 100% correct diagnosis as $N \rightarrow \infty$ if $\gamma \geq 2$ and $R \geq \alpha(N) \log N$. It is also proven that no category 3AM multiple syndrome diagnosis algorithm achieves 100% correct diagnosis as $N \rightarrow \infty$ if $R \leq \dfrac{\log N}{\alpha(N)}$.

For the asymptotic analysis of the OPTM algorithm, the following corollary [6] to a theorem proved by Chernoff [11] is needed.

Corollary 1: Let Z be a binomial random variable with parameters n and q. Then

$$P(Z \leq cnq) \leq e^{-(1-c)^2 nq/2}, \quad 0 < c \leq 1 ,$$

$$P(Z \geq cnq) \leq e^{-(c-1)^2 nq/3}, \quad c \geq 1 .$$

For the purposes of analysis, let us assume that $H^*_{tlh_i} = H^*_{tlh}$ for all $u_i \in V_T$ (the proofs also work when this does not hold). Since H^*_{tlh} is equivalent to $\bar{H}_{tlh}$ when used as a threshold, the analysis will be done assuming that H^*_{tlh} is the threshold used in the OPTM algorithm. Also, let a be the base of the logarithm unless otherwise specified. Let the random variable Y denote the number of faulty nodes u_i for which $d(u_i) > z^*_{tlh_i}$ for $\leq H_{tlh}$ syndromes. Let the

random variable X denote the number of non-faulty nodes u_i for which $d(u_i) > z^*_{th_i}$ for $> H_{th}$ syndromes. For a multiple syndrome diagnosis situation, if there are no nodes which fit the requirements for random variables Y or X, then OPTM produces correct diagnosis. However, if there is any node that fits the requirement for random variable Y or X, then OPTM does not produce correct diagnosis. Thus,

$$1 - E[X] - E[Y] \leq \mathrm{P}(\{\text{OPTM produces correct diagnosis}\}) \leq 1 - \max\{E[X], E[Y]\}.$$

Lemma 5.1: If $R \geq \alpha(N) \log N$, where $\lim_{N\to\infty} \alpha(N) = \infty$, and $RB_1 \leq H^*_{th} \leq RA_2$, then $\lim_{N\to\infty} E[X] = \lim_{N\to\infty} E[Y] = 0$.

Proof: In the following, let $m = \lfloor H^*_{th} \rfloor$. The comments refer to Fig. 5.2.

$$E[X] = \sum_{u_i \in V_T} P(> H^*_{th} \text{ passes} \mid \bar{\delta}_i)\, P(\bar{\delta}_i)$$

$$\leq \sum_{u_i \in V_T} \left[\sum_{k=m+1}^{R} \binom{R}{k} B_1^k (1 - B_1)^{R-k} \right] (1 - f_i) \qquad \{\text{distribution shifted right}\}$$

$$= (1 - f) N \sum_{k=m+1}^{R} \binom{R}{k} B_1^k (1 - B_1)^{R-k}$$

$$\leq (1 - f) N \sum_{k=m}^{R} \binom{R}{k} B_1^k (1 - B_1)^{R-k}$$

$$= (1 - f)\, a^{\log N}\, P(X \geq c_1 R B_1)$$

$$\leq (1 - f)\, a^{\log N}\, e^{-(c_1 - 1)^2 RB_1/3} \quad \text{if } c_1 = \frac{m}{RB_1} \geq 1.$$

Since $H^*_{th} \geq m \geq RB_1$ and $R \geq \alpha(N) \log N$, $\lim_{N\to\infty} E[X] = 0$. Similarly, for $E[Y]$,

$$E[Y] = \sum_{u_i \in V_T} P(\leq H^*_{th} \text{ passes} \mid \delta_i)\, P(\delta_i)$$

$$\leq \sum_{u_i \in V_T} \left[\sum_{k=0}^{m} \binom{R}{k} A_2^k (1 - A_2)^{R-k} \right] f_i \qquad \{\text{distribution shifted left}\}$$

$$= f \sum_{k=0}^{m} \binom{R}{k} A_2^k (1 - A_2)^{R-k}$$

$$= f \; a^{\log N} \; P(X \leq c_2 R A_2)$$

$$\leq f \; a^{\log N} \; e^{-(1-c_2)^2 R A_2/2} \qquad \text{if } c_2 = \frac{m}{RA_2} \leq 1 \text{ and } c_2 > 0.$$

Since $(H_{th}^* \leq RA_2) \Rightarrow (m \leq RA_2)$ and $R \geq \alpha(N) \log N$, $\lim_{N \to \infty} E[Y] = 0$. **Q.E.D.**

Theorem 5.2: If $R \geq \alpha(N) \log N$, where $\lim_{N \to \infty} \alpha(N) = \infty$, then P({OPTM produces correct diagnosis}) $\rightarrow 1$ as $N \rightarrow \infty$.

Proof: Let the random variables Z_0, Z_1 and Z_2 be as defined in the proof of Theorem 5.1. As shown in Fig. 5.2, Z_0 is sandwiched in between Z_2 and Z_1. Since Z_1 and Z_2 are binomial random variables, $E[Z_1] = RA_1$ and $E[Z_2] = RA_2$. Thus, $RA_2 \leq E[Z_0] \leq RA_1$. Likewise, if the random variable W_0 denotes the number of syndromes for which a non-faulty node u_i has $d(u_i) > z_{th_i}^*$, then $RB_2 \leq E[W_0] \leq RB_1$. Then, from Eqs. (5.3) and (5.4),

$$B_1 = p^\gamma \leq p(1 - \frac{p}{M})^{\gamma - 1} \leq A_2$$

since M has been assumed to be a large number.

Now, let D be a category 3AM multiple syndrome diagnosis algorithm with $z_{th_i} = z_{th_i}^*$ and $H_{th} = 0.5\, R(B_1 + A_2)$. Then, since $RB_1 \leq H_{th} \leq RA_2$, D produces correct diagnosis as $N \rightarrow \infty$ provided the conditions of the theorem are satisfied. Thus, for any $\varepsilon > 0$, there exists an $N' > 0$ such that P({D produces correct diagnosis}) $> 1 - \varepsilon$. H_{th}^* is the optimal H_{th} threshold value. Therefore, P({OPTM produces correct diagnosis}) > P({D produces correct diagnosis}) $> 1 - \varepsilon$. Since this holds for any $\varepsilon > 0$, the theorem follows. **Q.E.D.**

Theorem 5.2 could also have been proven by using the fact that the FR algorithm is in category 3AM since the OPTM algorithm is the optimal category 3AM multiple syndrome

diagnosis algorithm and Theorem 5.2 was proven for the FR algorithm [30]. However, a more direct proof of Theorem 5.2 has been provided in this section.

Lower bounds are now determined for the number of testing stages required for asymptotically correct category 3AM multiple syndrome diagnosis. In [6], Blough essentially proved that if $\leq \log N/\alpha(N)$ tests are performed on each processor, where $\lim_{N\to\infty} \alpha(N) = \infty$, then no category 3AM diagnosis algorithm which uses a single syndrome can achieve asymptotically correct diagnosis as $N \to \infty$. It follows that if γ is constant and $R \leq \log N/\alpha(N)$, where $\lim_{N\to\infty} \alpha(N) = \infty$, then no category 3AM multiple syndrome diagnosis algorithm can achieve asymptotically correct diagnosis as $N \to \infty$. However, what happens when $R = \log N$? The following theorem answers this question.

Theorem 5.3: If $A_1 < 1 - \frac{1}{a}$ (recall that a is the base of the logarithm) and $R = \log N$, then for any category 3AM multiple syndrome diagnosis algorithm D, P({D produces correct diagnosis}) $\to 0$ as $N \to 0$.

Proof: Suppose that $A_1 < 1 - \frac{1}{a}$ and $R = \log N$. Then,

$$\lim_{N\to\infty} E[Y] \geq \lim_{N\to\infty} Nf\ (1 - A_1)^R = f \lim_{N\to\infty} \left[(1 - A_1)\, a\right]^{\log N} = \infty .$$

Thus, as $N \to \infty$, P({OPTM produces correct diagnosis}) $\to 0$ and likewise for any other category 3AM multiple syndrome diagnosis algorithm. **Q.E.D.**

As an example of the use of Theorem 5.3, if $a = 2$ (as in a hypercube structure), Theorem 5.3 tells us that one must have $A_1 \geq 0.5$, which implies that $p > 0.4$ (with $\gamma \geq 2$). It is unlikely that such a high p (fault coverage) value can be obtained using a single test task of short duration. A slightly higher upper bound for A_1 can be obtained by using a closer lower bound approximation for $E[Y]$. In summary, it appears unlikely that any category 3AM multiple syndrome diagnosis algorithm can achieve asymptotically correct diagnosis when

$R = \log N$ unless a is very small, implying a quickly growing logarithm function, or very long test tasks are used.

5.6. Simulations

Simulations were conducted to evaluate the performance of the diagnosis algorithms studied. In assigning prior fault probability values, an exponential failure arrival rate was assumed. For each node u_i, a time value τ_i, corresponding to the length of time u_i has been in the system, was generated from a uniform distribution over the interval $[0, T]$ for some T. Then $f_i = 1 - e^{-\lambda\tau_i}$ was assigned to u_i, where $\lambda = \text{MTTF}^{-1}$ is the mean failure arrival rate.

The simulated experiments were conducted on a Sun 4/280 for a 100-node torus-wrapped square mesh and a 300-node TMR structure. A TMR structure is a 2-regular graph in which nodes are clustered into completely connected components of size 3 each. $T = 10^3$ hours and MTTF values 10^4 and 10^5 hours were used, resulting in λT values of 0.1 and 0.01 respectively. Given MTTF values of 10^4 and 10^5, $E[f_i] = 0.0484$ and 0.0050, respectively.

For all diagnosis algorithms evaluated, 1000 fault situations were produced and diagnosed assuming p values of 0.1 to 0.5 (in 0.1 increments) and the two MTTF values given above. For each fault situation, $R = 8$ testing stages were used, resulting in 8 syndromes. The OPTM algorithm was compared with the FR algorithm and the OPT3A algorithm, which is the optimal category 3A single syndrome diagnosis algorithm [46]. Assuming that it takes τ units of time to execute a single test task, multiple syndrome testing requires $R\tau$ time units while single syndrome testing requires $\gamma\tau$ time units. Thus, for single syndrome diagnosis, if the same amount of time is devoted to testing, it is possible to use test tasks which are R/γ times as long as those used in multiple syndrome diagnosis. Therefore, in the simulations for the OPT3A algorithm, $p' = 1 - (1 - p)^{R/\gamma}$ was used as the fault coverage value.

In the simulations for the FR algorithm, values for kv_i and sv_i must be chosen. The FR algorithm specifies that $kv_i = \gamma - 1$ but indicates that a range of values is acceptable for sv_i (refer to Section 5.2). If the equation for sv_i in Section 5.2 is used, it is possible to get a negative value for sv_i. Since a negative sv_i threshold value implies that all nodes in V_T will be diagnosed to be faulty, this possibility is discounted. Then the modified equation for sv_i is $\max\{0\ ,\ R - 2R(1 - p(1 - \frac{p}{M})^{\gamma})\} \leq sv_i \leq R - R(1 - p(1 - \frac{p}{M})^{\gamma})$. In our simulations, sv_i was chosen to be the value halfway between the lower and upper bounds for sv_i. M was chosen to be 1000. Tables 5.2 and 5.3 show the values of $\bar{H}_{th}$ and sv_i for the torus-wrapped square mesh and TMR structure, respectively. $\bar{H}_{th}$ values shown in Tables 5.2 and 5.3 are for both $f = 0.0484$ and $f = 0.0050$ unless otherwise specified. sv_i and sv_i^{mid} (used in the simulations) are independent of f. In Table 5.3, threshold values for $p = 0.7$ and $f = 0.0484$ are shown to demonstrate that $\bar{H}_i$ and sv_i values do diverge.

	$\bar{H}_{th}$	sv_i	sv_i^{mid}
$p = 0.1$	0	0 — 0.80	0.40
$p = 0.2$	0	0 — 1.60	0.80
$p = 0.3$	0	0 — 2.40	1.20
$p = 0.4$	0	0 — 3.19	1.60
$p = 0.5$	0	0 — 3.99	2.00

Table 5.2: Threshold values for torus-wrapped square mesh ($\gamma = 4$).

	$\bar{H}_{th}$	sv_i	sv_i^{mid}
$p = 0.1$	0	0 — 0.80	0.40
$p = 0.2$	0	0 — 1.60	0.80
$p = 0.3$	0	0 — 2.40	1.20
$p = 0.4$	0	0 — 3.20	1.60
$p = 0.5$	0	0 — 4.00	2.00
$p = 0.7$	1 ($f = 0.0484$)	3.18 — 5.59	4.38

Table 5.3: Threshold values for TMR structure ($\gamma = 2$).

Figs. 5.3 and 5.4 show the results of the simulations for the torus-wrapped square mesh and TMR structure, respectively. In all cases, the OPTM algorithm performs significantly better than the OPT3A algorithm, with the difference more acute when p is small. The FR algorithm performs the same as the OPTM algorithm for $p = 0.1$ and $p = 0.2$, but then quickly falls off in accuracy as different $\lfloor sv_i^{mid} \rfloor$ threshold values are used. Similar results were obtained for all simulations attempted.

5.7. Conclusion

In this chapter, an optimal category 3AM (and near-optimal category 3A and category 3) multiple syndrome diagnosis algorithm has been derived. Using probability analysis, multiple syndrome testing is shown to be more effective than single syndrome testing. Our simulation results support the probability analysis. It is proven that Algorithm OPTM, the optimal category 3AM multiple syndrome diagnosis algorithm, achieves 100% correct diagnosis in an N processor system as $N \rightarrow \infty$ provided that $R \geq \alpha(N) \log N$ testing stages are used, where $\alpha(N) \rightarrow \infty$ arbitrarily slowly as $N \rightarrow \infty$. It is also shown that no category 3AM can achieve asymptotically correct diagnosis as $N \rightarrow \infty$ if γ is constant and $R \leq \log N/\alpha(N)$. If γ is constant, the computational complexity of the OPTM algorithm is $O(R)$, which is the minimum possible for any multiple syndrome diagnosis algorithm since R testing stages are required.

The OPTM algorithm requires each processor to execute identical tasks with its neighbors, send and receive the results of the tasks from its neighbors, compare the results received with its own results, and execute a diagnosis procedure to determine whether it should diagnose itself to be faulty or non-faulty. The diagnosis procedure must be executed by a diagnostic component which is either *ultra-reliable* (part of the *hard-core* of the processor) or operates in a "fail-safe" mode. In the former case, the diagnosis algorithm is required to be simple. The thresholds used by the OPTM algorithm can be precomputed. Thus, the OPTM

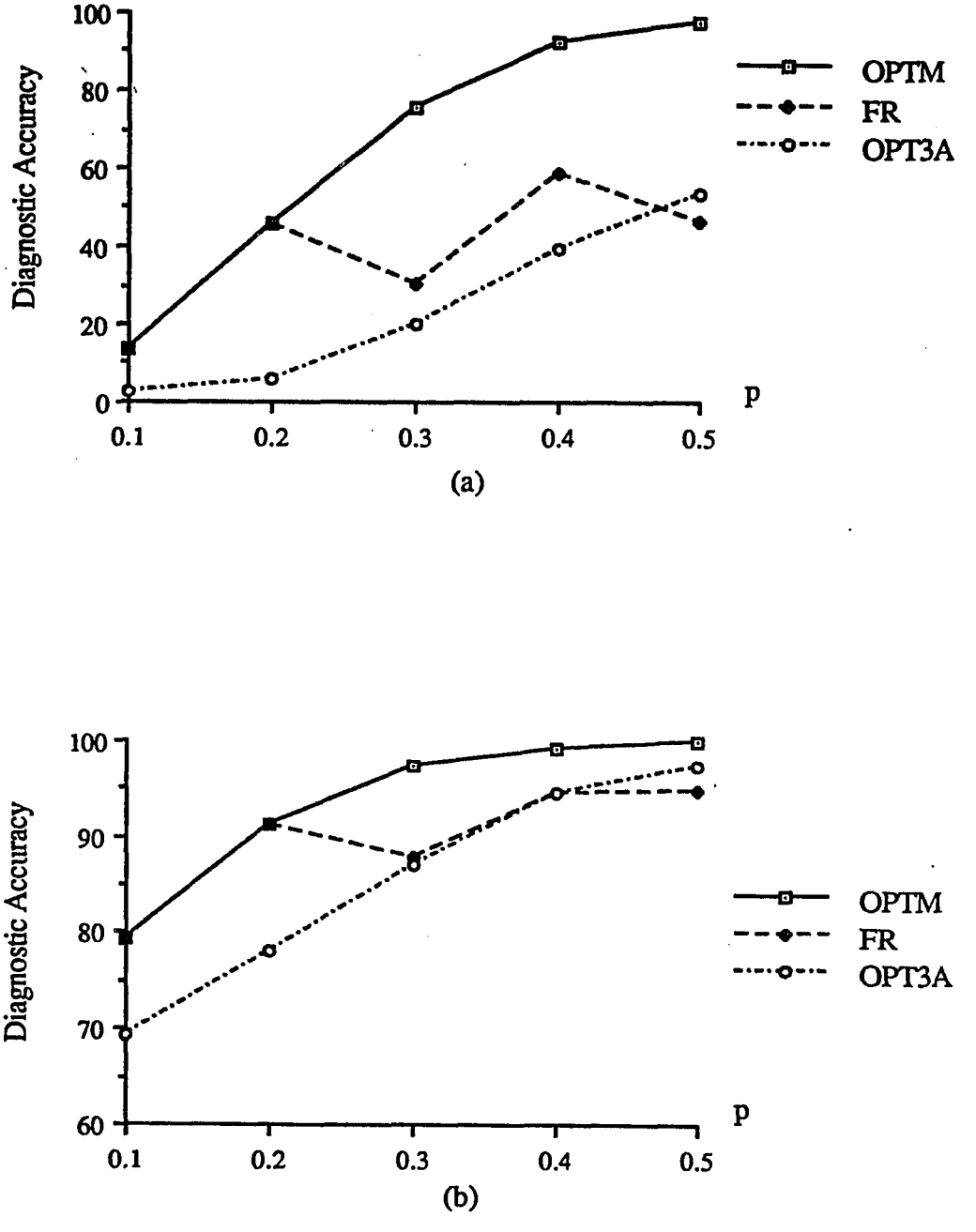


Figure 5.3. Accuracy with square mesh and MTTF of (a) 10K and (b) 100K hours.

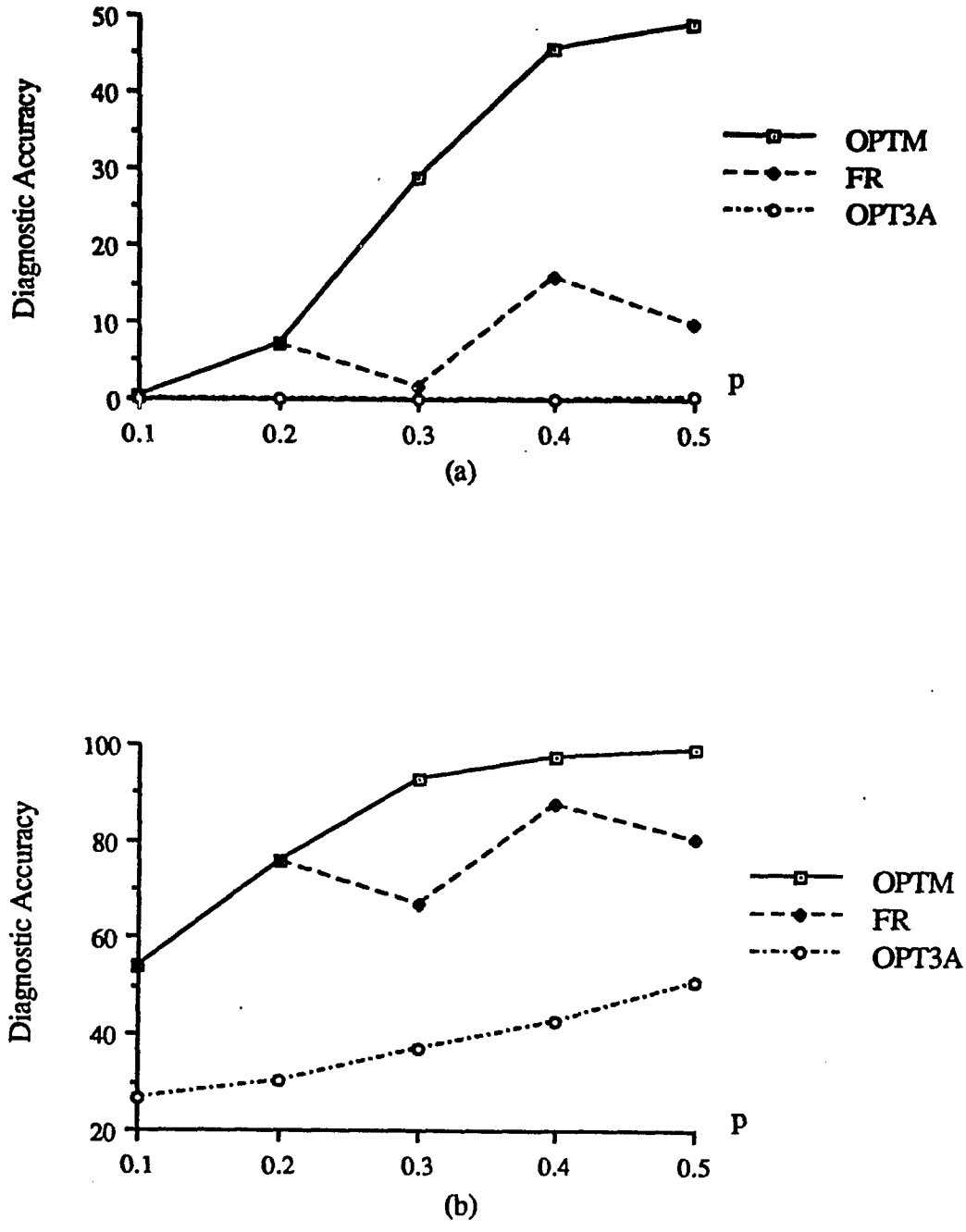


Figure 5.4. Accuracy with TMR structure and MTTF of (a) 10K and (b) 100K hours.

algorithm simply requires the diagnostic component to accumulate integer quantities and compare them against precomputed threshold values. The diagnostic component can therefore be a very simple digital circuit.

CHAPTER 6

DISTRIBUTED DIAGNOSIS

This chapter addresses the distributed implementation of the algorithms presented in the previous chapters.

6.1. Introduction

There are three main aspects to the distributed diagnosis problem: getting the syndrome information to all of the nodes, performing the diagnosis in a distributed manner, and getting the diagnosis decisions to all of the nodes. However, given certain assumptions, the last aspect of the distributed diagnosis problem may not be necessary. During normal operation, if the diagnostic component of a given node knows that the node is faulty, then any time a task is sent to that node, the diagnostic component can intervene and refuse to accept the task. Thus, if the diagnosis information that each node has is *consistent* with the diagnosis information of all other nodes (e.g., each node only makes a diagnosis of itself), then it may not be necessary to exchange diagnosis information. Therefore, this chapter addresses only the first two aspects of the distributed diagnosis problem: getting the syndrome information to all of the nodes and performing the distributed diagnosis.

For a category 3 or 3A probabilistic diagnosis algorithm, executing the algorithm in a distributed manner is very simple. Each node executes its assigned inter-processor tests, using comparison-testing or an alternate method. Each time a node u_i tests another node u_j, u_i tells u_j its test result. Then, each node executes its diagnosis algorithm to diagnose itself as faulty

or non-faulty. Since purely local syndrome information is being used, no other communication is necessary.

For category 2 and 2A probabilistic diagnosis, distributed implementation is not so straightforward. The difficulty lies in the fact that this type of algorithm effectively uses a summarized form of the entire syndrome information. Thus, the testing result at one corner of the system interconnection network may have to be sent to the opposite corner of the interconnection network. Part of the proposed solution to this problem involves making all required communication occur between nodes which are adjacent in the interconnection network. The number of such communication steps required (besides the initial reliable communication between neighboring nodes to determine each other's testing results) is at most $2|F|$, where F is the set of nodes diagnosed as faulty by the diagnosis algorithm.

Category 1 diagnosis requires the entire syndrome to be reliably communicated to every non-faulty node. Unlike categories 2 and 2A, there is no other choice but to send multiple copies of each node's broadcast message to every other node in the system. Once each node has all of the syndrome information, it can simply execute the diagnosis algorithm independently of the other nodes. However, in order to guarantee that each node diagnoses the same set F as the set of faulty nodes, it is normally required that every node must agree on the test results of faulty as well as non-faulty nodes. Since nodes can be intermittently faulty and no assumptions have been made regarding the manner in which nodes can be faulty, this implies that the communication problem must be viewed as a Byzantine Generals problem [41].

For the distributed implementation of the diagnosis algorithms presented, it is now clear that solutions to the all-to-all reliable broadcast and nearest-neighbor reliable multicast problems are required. Efficient solutions to these problems are presented in Sections 6.3 and 6.4. Note that one possible all-to-all reliable broadcast method is to use x iterations of nearest-neighbor reliable multicast, where x is the diameter of the interconnection network. However,

this solution is not as efficient as the one presented in Section 6.3. In the next section, a method is shown for efficiently implementing distributed versions of category 2 and 2A algorithms using only nearest-neighbor reliable multicast. The computational complexity analysis in this chapter is done under the assumption that a node can use all of its incoming and outgoing links concurrently. This assumption is not unrealistic as it is satisfied by the HARTS routing controller chip [23].

6.2. Distributed Implementation of Category 2 and 2A Algorithms

The method for the distributed implementation of category 2 and 2A diagnosis algorithms is illustrated by describing the distributed implementation of the DSK* algorithm. To implement other category 2 or 2A probabilistic diagnosis algorithms, the evaluation function $e(.)$, the procedure for updating the evaluation function when nodes are identified as faulty, and the stopping condition must be appropriately modified. The following is the distributed version of the DSK* algorithm.

Algorithm DDSK:*

1. **for** each $u_i \in V$, **doparallel**
   let $e(u_i) = d(u_i) + f_i$;
   send message $<e(u_i)>$ to the nodes in $\Gamma(u_i)$;
2. **for** each $u_i \in V$, **doparallel**
   if $e(u_i) \geq 1$ and $\forall u_j \in \Gamma^{-1}(u_i)$ $\left[(e(u_i) > e(u_j)) \text{ or } (e(u_i) = e(u_j) \text{ and } (i > j))\right]$ **then**
   begin
   declare u_i to be faulty;
   $e(u_i) = 0$;
   for each $u_k \in \Gamma(u_i)$, **doparallel**
   begin
   u_i sends $<e(u_i)$, *faulty*$>$ to u_k;
   if $a_{ik} = 1$ then
   begin
   $e(u_k) = e(u_k) - 1$;
   u_k sends $<e(u_k)>$ to $\Gamma(u_k)$;
   endif
   endfor
   endif;
3. **repeat** Step 2 until $e(u_i) < 1$ for all $u_i \in V$;

The execution of the DDSK* algorithm is very simple and requires a minimal amount of communication overhead. As defined above, $e(u_i) \geq 1$ if and only if $d(u_i) > 0$. Thus, the stopping condition in Step 3 simply checks whether all one-links in the syndrome have been accounted for. In Step 2, the algorithm checks whether node u_i can be considered to be a local maximum. The reason for the second condition within the brackets in Step 2 is so that two adjacent nodes do not both consider themselves to have locally maximum $e(.)$ values. All communication is done between nearest neighbors. In each iteration of Step 2, messages are propagated at most two hops. Effectively, this two-hop message propagation allows the global diagnosis algorithm to be implemented using local calculations. The following theorem states that DDSK* is a correct distributed implementation of DSK*.

Theorem 6.1: If $f_i < 1$ for all $u_i \in V$, Algorithm DDSK* always produces the same diagnosis as Algorithm DSK*.

Proof: Given an arbitrary syndrome, let F_1 be the diagnosis produced by DSK* and let F_2 be the diagnosis produced by DDSK*. For simplicity in analysis, let us assume that $e(.)$ values are different for all nodes. If $e(.)$ values are ever identical, the ordering implied in Step 2 of DDSK* can be used to break ties.

Suppose there exists a node $u_k \in F_1$ that is not a member of F_2. Without loss of generality, let u_k be the first such node chosen by DSK*. Then u_k had to have the globally maximum updated $e(.)$ value in the DSK* algorithm. All nodes u_m chosen in previous steps by DSK* must also have been chosen by DDSK*. Thus, since the updating procedure is the same for DSK* and DDSK*, u_k has the globally maximum updated $e(.)$ value in the DDSK* algorithm. This implies that $e(u_k)$ is also locally maximum, and thus, $u_k \in F_2$ because of Step 2 of DDSK*. Therefore, $F_1 \subseteq F_2$.

Suppose there exists a node $u_k \in F_2$ that is not a member of F_1. By Step 2 of DDSK*, it is known that u_k has a locally maximum $e(.)$ value and that $e(u_k) \geq 1$. Note that $e(.)$ values are never increased as nodes are identified to be faulty in both DSK* and DDSK*. Let u_m be the node with the globally maximum $e(.)$ value. $e(u_m) > e(u_k) \geq 1$. Thus, u_m must be chosen to be faulty by DSK*. Since a globally maximum $e(.)$ value is also a locally maximum $e(.)$ value, u_m must also be chosen to be faulty by DDSK*. Since u_k is itself a local maximum, while the $e(.)$ values of u_k's neighbors may be decreased, $e(u_k)$ cannot be decreased by the above choice of u_m. Eventually, all such nodes u_m will be exhausted. At that time, u_k will have a globally maximum $e(.)$ value, Thus, at that time, u_k will be chosen as faulty by DSK*, and thus, $u_k \in F_1$. Therefore, $F_2 \subseteq F_1$. **Q.E.D.**

The computational complexity of the DDSK* algorithm is $O(|F'|)$, where F' is the set of faulty nodes. Thus, since the number of faulty nodes is typically much less than the number of nodes in the system, this is an efficient distributed algorithm.

6.3. All-to-All Reliable Broadcast

In addition to distributed diagnosis, all-to-all (ATA) reliable broadcast is essential for implementing several key fault-tolerant algorithms for clock synchronization [39, 42, 54] and distributed agreement [22, 41]. In these algorithms, each non-faulty node must be able to correctly deliver its message to all of the other non-faulty nodes in the system. Given a regular interconnection network with N nodes and connectivity γ, this can be done if every non-faulty node sends its message to every other node through γ disjoint paths. All algorithms described in this section are of this type. Using this type of method, $\gamma - 1$ faulty nodes/links can be tolerated if signed messages are used; $\min\{\lceil\frac{\gamma}{2}\rceil - 1, \lceil\frac{N}{3}\rceil - 1\}$ faulty nodes/links with unsigned messages. To disrupt communication between nodes u and v with $\geq \gamma$ faulty nodes/links, there must be at least one faulty node or link in every disjoint path from u to v.

Thus, using this type of method, the probability of correct operation is high even when $\geq \gamma$ faulty nodes/links are present.

In this section, an efficient solution for ATA reliable broadcast is presented that works on a class of interconnection networks which includes regular meshes and binary hypercubes. Regular meshes [10] and binary hypercubes [3, 59] have drawn considerable attention in recent years as an interconnection topology for the processors of a distributed computing system.

Most previous work on reliable broadcast [53] and ATA reliable broadcast [26] has implicitly assumed a store-and-forward routing method. These algorithms can be described by executing the broadcast in several steps and specifying the point-to-point communication patterns that occur at each step. Then the objective is to minimize the total time required for the broadcast operation by using a minimal number of steps with minimal-length messages transmitted at each step. Ramanathan and Shin's reliable broadcast (RS) algorithm [53] requires $\gamma + 1$ steps on a hypercube of dimension γ if each node can simultaneously use all of its outgoing links. Fraigniaud's ATA reliable broadcast (FRS) algorithm for hypercubes [26] simply involves executing the RS algorithm at each node in lock step. In every step after the first, each node must merge two messages from the previous step before sending the larger message in the current step. In the last step, the message formed after merging can be made a little bit shorter by removing the portion of the message that would be returned to the originator of that portion of the message. The time required for the FRS algorithm is $(\gamma + 1)\,\tau_S + (2^\gamma - 1)\,L\,\tau_L$, where τ_S is the message startup time, L is the message length, and τ_L is the propagation time per unit length message. To achieve this time, 100% of the link capacity must be used for the entire duration of the ATA broadcast operation.

Recently, there has been work on broadcast algorithms that take advantage of the faster communication possible using *virtual cut-through* switching [36]. In virtual cut-through [38] and *wormhole routing* [21], instead of storing a message completely in a node and then

forwarding it to the next node, the head of the message is advanced directly from incoming to outgoing channels. Only a few control bytes are buffered (in a small on-line FIFO buffer) at each node to determine the outgoing channel for the message. If all outgoing channels are busy, then the entire message is buffered (in a much larger intermediate storage buffer) in the case of virtual cut-through and prevented from moving forward in the case of wormhole routing. Deadlock-free wormhole routing is addressed in [21]. Special routing controller chips have been designed for wormhole routing [20] and virtual cut-through [23]. The operation of advancing a message immediately from incoming to outgoing channels is referred to as *cut-through*. When a message is being advanced in this manner, it is possible for the node to also *receive* the message by copying the portions of the message as they pass through the FIFO buffer [36]. In this section, it is assumed that all messages are transmitted in fixed sized packets. It is also assumed that different sizes can be used for packets of different types.

Kandlur and Shin's reliable broadcast (KS) algorithm [36] is an efficient algorithm for a regularly wrapped hexagonal mesh topology which uses virtual cut-through. In broadcast algorithms that use virtual cut-through, besides minimizing the number of send-receive operations on the longest path, it is desirable to maximize the number of send-receive operations that can be implemented with cut-through. In the KS algorithm, the longest path has $2 \times$ (diameter of mesh) send-receive operations, among which all but three can use cut-through. The analysis done in [36] shows that for a single reliable broadcast operation, the KS algorithm is much faster than an algorithm based on the use of edge-disjoint Hamiltonian cycles (HC's).

ATA reliable broadcast is an extremely expensive operation. For any interconnection network with N nodes, $(N-1)\,\tau$ is the minimum time required, where τ is the time to transmit a packet from a node to its neighbor. To achieve this minimum time, 100% of the link capacity must be used for the entire duration of the ATA reliable broadcast operation. As

an example, if N = 10,001 and τ = 1 millisecond, the entire operation requires 10 seconds, during which time the network is completely congested with the ATA reliable broadcast.

The ATA reliable broadcast operation should have minimal adverse effect on the normal operation of the system. At the same time, for applications such as clock synchronization, timely execution of the ATA reliable operation is required [54]. These two requirements are contradictory since a lower total execution time T_{all} implies a higher link utilization, H, by the ATA reliable broadcast operation. Thus, it is desirable to be able to adjust H and T_{all} values depending on the priority of the task requiring the ATA reliable broadcast. One method for reducing H is to execute the RS algorithm (for hypercubes) or the KS algorithm (for hexagonal meshes) for each node in turn, with the reliable broadcast for one node starting when the previous node finishes. This method reduces H at the expense of increasing the required execution time T_{all} by a factor of N. An alternate method is to have several nodes execute the reliable broadcast operation at any given time. However, if this is done using the KS or a similar reliable broadcast algorithm, it is possible for two ATA reliable broadcast packets to contend for the same communication link, thus eliminating the possibility of both packets using cut-through.

In heavy network traffic, the advantages of virtual cut-through are minimal because most send-receive operations will be forced to use store-and-forward instead of cut-through. In fact, as will be shown in Section 6.3.4, in heavy network traffic, a solution to the ATA reliable broadcast problem using store-and-forward routing has lower execution time than any solution that uses virtual cut-through. Thus, an ATA reliable broadcast algorithm using virtual cut-through routing should be optimized for low to moderate network traffic and have minimal adverse effect on the normal operation of the system.

In the ATA reliable broadcast solution presented, for implementation reasons, the algorithm interleaves the nodes that send a broadcast message along a particular HC and executes

the required broadcast operation in several stages. By adjusting the interleaving distance η, it is possible to decrease H and increase T_{all} by the same factor. This solution can be used essentially unchanged on a large class of interconnection topologies including hypercubes and meshes. Under a model suitable for analyzing virtual cut-through, it is proven that this solution with the choice of minimum η results in the minimum value of T_{all} when there is no other network traffic. Under normal network traffic conditions, this solution with a suitable choice of η compares favorably to several other possible ATA reliable broadcast algorithms. Under heavy network traffic conditions, a store-and-forward solution results in lower T_{all} than any virtual cut-through solution. However this comes at the expense of significantly delaying normal network traffic. The advantages of the proposed solution are most evident when the probability of cut-through is high, broadcast messages are short, and the interconnection network is large.

6.3.1. Interconnection Networks

Recall from Chapter 2 that G is the undirected graph corresponding to the physical interconnection structure of the system and that G^{dir} is the graph G with every undirected edge replaced by two directed edges (one in each direction). A graph is said to be *γ-regular* if all nodes in the graph have degree γ. A *regular* graph is one that is *γ-regular* for some γ [8]. A graph G is said to belong to the class Λ if the following two conditions are satisfied.

LC1: G is *γ-regular* for an even integer γ.

LC2: There are $\frac{\gamma}{2}$ undirected edge-disjoint HC's in G.

The ATA reliable broadcast algorithm described in this chapter can be used on any graph belonging to the class Λ. Note that if G belongs to the class Λ, then γ is the connectivity of G by condition LC1.

An m-dimensional hypercube, denoted by Q_m, has $N = 2^m$ nodes and $m2^{m-1}$ edges. If directed communication links are used, Q_m^{dir} has $m2^m$ directed edges. A Q_m is recursively defined as: (1) Q_0 is a single point, and (2) $Q_m = K_2 \times Q_{m-1}$, where K_2 is a complete graph of 2 nodes and $\times$ denotes the product operation on two graphs [8]. Fig. 6.1 shows examples of Q_1, Q_2 and Q_3. Each node in a Q_m is uniquely represented by an m-bit address such that the addresses of adjacent nodes differ in exactly one bit. The bits in an address are referred to in right to left order from 0 to $m - 1$. Two adjacent nodes which differ in the i-th bit will be said to be in *direction* i $(0 \le i \le m - 1)$ with respect to each other.

Hypercubes of even dimension belong to the class Λ. Condition LC1 is satisfied because a Q_m is m-regular with degree $\gamma = m$. Condition LC2 is satisfied by Theorem 6.2 below. The proof of Theorem 6.2 is a constructive one; thus, it can be used to construct the m undirected edge-disjoint HC's in a Q_{2m}. As an example, two undirected edge-disjoint HC's in a Q_4 are shown in Fig. 6.2(a) (although Fig. 6.2(a) shows a square mesh, it is also a Q_4 since a Q_4 can be redrawn as a 4×4 torus-wrapped square mesh). Let C_k denote an undirected cycle of length k. Then Theorem 6.2 can be proven with the aid of the following two lemmas. Reference [24] gives the construction method for products of two cycles and reference [25] gives the construction method for products of three cycles.

Lemma 6.1: [24] $C_k \times C_l$ $(k, l \ge 3)$ can be decomposed into 2 undirected HC's.

Lemma 6.2: [24,25] $C_k \times C_l \times C_r$ $(k, l, r \ge 3)$ can be decomposed into 3 undirected HC's.

Theorem 6.2: [33] A Q_{2m} contains m undirected edge-disjoint HC's.

Proof: The theorem can be proven by induction on $2m$. For the induction basis, a Q_2 is a cycle and a Q_4 has two undirected edge-disjoint HC's by Lemma 6.1. Assume a Q_{2k} contains k undirected edge-disjoint HC's for all $k \le m$. We must show that a Q_{2m+2} contains

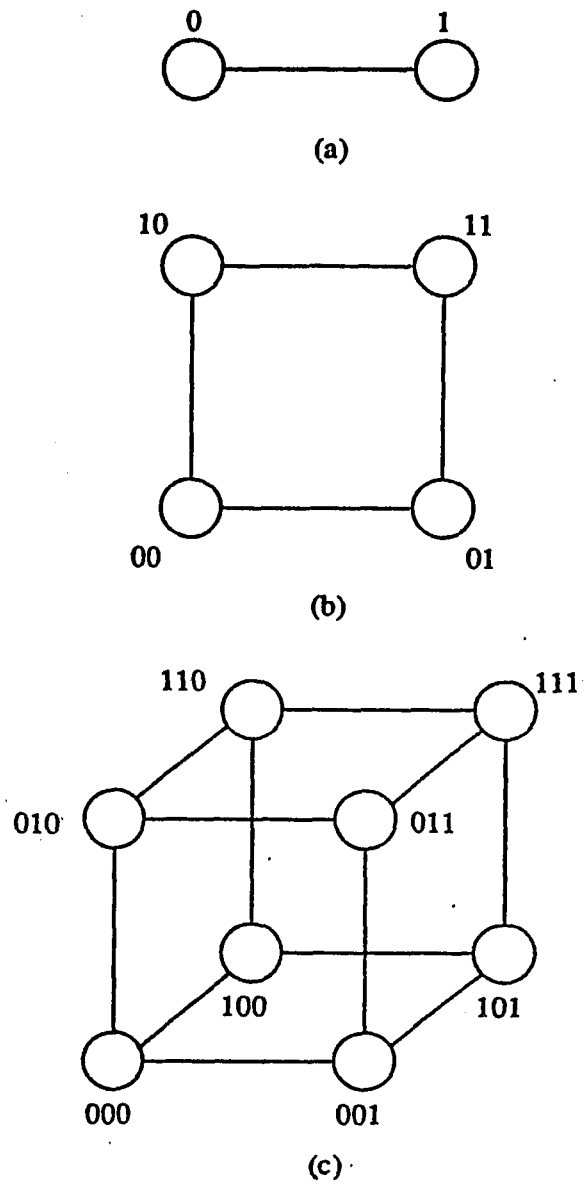


Figure 6.1. Illustrative examples of hypercubes (a) Q_1, (b) Q_2, and (c) Q_3.

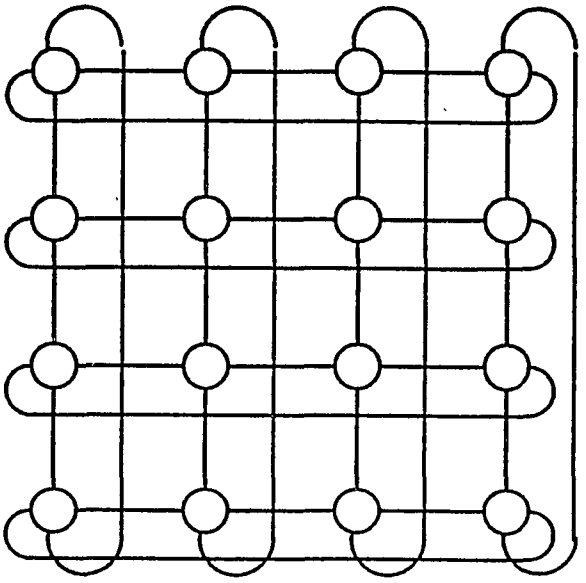
$m + 1$ undirected edge-disjoint HC's.

If $m + 1$ is even, then decompose the Q_{2m+2} into two Q_{m+1}'s. By induction, each Q_{m+1} contains $(m + 1)/2$ undirected edge-disjoint HC's. Create $(m + 1)/2$ graphs by multiplying (applying the product operation on) the i-th HC's in the two decomposed hypercubes, for all $1 \leq i \leq (m + 1)/2$. By Lemma 6.1, each product graph contains two undirected edge-disjoint HC's. Then, since all product graphs are edge-disjoint, there exist $(m + 1)$ undirected edge-disjoint HC's in the Q_{2m+2}.

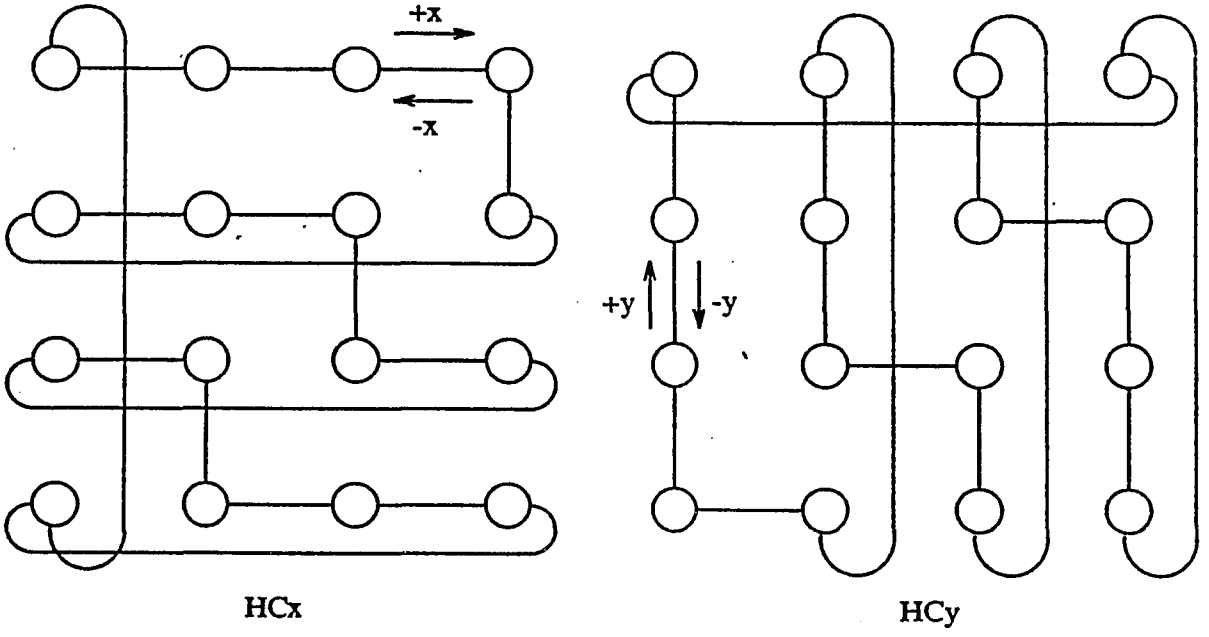
If $m + 1$ is odd, then decompose the Q_{2m+2} into a Q_m and a Q_{m+2}. Create $m/2 - 1$ graphs by multiplying the i-th HC's in the two decomposed hypercubes, for all $1 \leq i \leq m/2 - 1$. By Lemma 6.1, each product graph again contains 2 undirected edge-disjoint HC's. There remains one unused HC in Q_m and 2 unused HC's in Q_{m+2}. From Lemma 6.2, it is known that the product of these 3 cycles can be decomposed into 3 HC's. Thus, there are $(m + 1)$ undirected edge-disjoint HC's in the Q_{2m+2}. **Q.E.D.**

The torus-wrapped square mesh, shown in Fig. 6.2(a), belongs to the class Λ. SQ_m is defined to be a torus-wrapped square mesh of size m, where m is the number of nodes in a single row or column. Since the degree of every node in a SQ_m is 4, condition LC1 is satisfied with $\gamma = 4$. Fig. 6.2(b) shows two undirected edge-disjoint HC's in a SQ_4, thus satisfying condition LC2. A similar pattern can be used to find two undirected edge-disjoint HC's for any SQ_m.

A hexagonal mesh (hex-mesh) has the general structure shown in Fig. 6.3(a), which is an unwrapped hex-mesh of size 3. In order to achieve regularity and homogeneity such that identical hardware, software and protocols can be applied uniformly over the network, it is required that the nodes on the hexagonal periphery be wrapped around systematically.



(a)



(b)

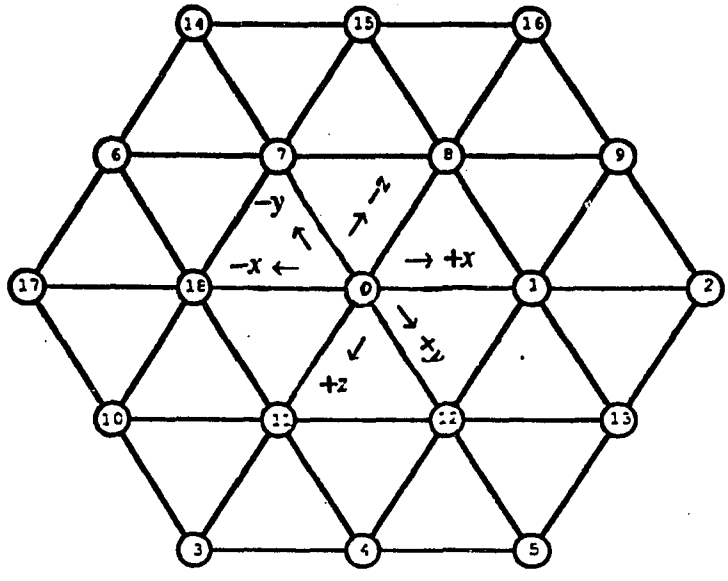
Figure 6.2: Torus-wrapped square mesh (a) with all links and (b) decomposed into HC's.

A general systematic method for wrapping, called *C-type wrapping* is defined in [10]. The six oriented directions in a hex-mesh are shown in Fig. 6.3(a). A hex-mesh of size m can be partitioned into $2m - 1$ rows with respect to any of the three directions +x, +y, and +z. For notational purposes, when a hex-mesh is partitioned into $2m - 1$ rows with respect to any direction and the hex-mesh is rotated such that the corresponding direction points to the right, the rows are referred to as rows 0 through $2m - 2$ from the top to the bottom. In addition, rows 0 to $m - 2$ and rows m to $2m - 2$ are referred to as the *upper* and *lower* parts of the hex-mesh, respectively. $[a]_b \equiv a \ mod \ b$, where a and b are integers. C-type wrapping is defined in [10] as follows: Wrap a hex-mesh of size m in such a way that for each of the three ways of partitioning the nodes into rows, the last node in row i is connected to the first node in row $[i+m-1]_{2m-1}$.

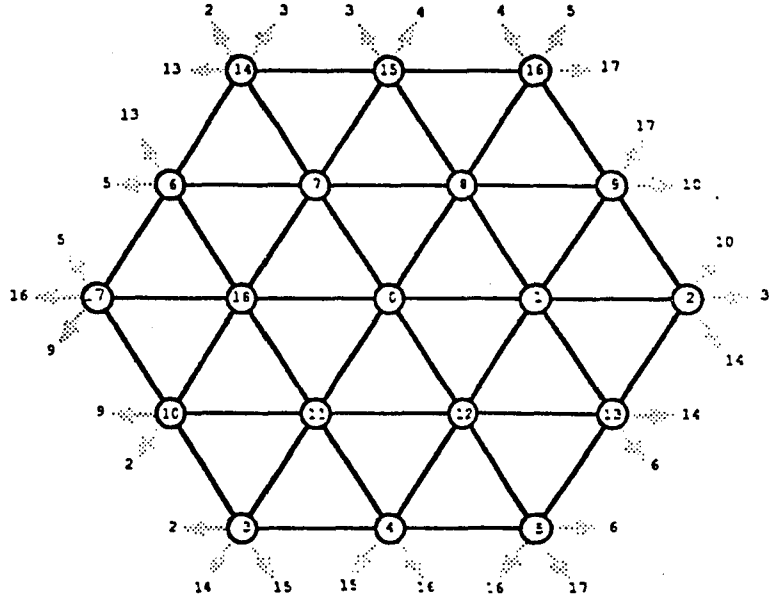
Several topological properties of hex-meshes with the C-type wrapping are given in [10]. Fig. 6.3(b) illustrates a C-type wrapped hex-mesh of size m, denoted as HM_m. A HM_m is a homogeneous processing surface, i.e., all nodes are topologically identical. The diameter of HM_m is $m - 1$, and there are $3m(m - 1) + 1$ nodes in a HM_m [10]. The edges in any direction of a HM_m have a very useful property. Theorem 6.3 states that the edges in any direction of a HM_m describes a HC. Based this theorem, it is known that there are three undirected edge-disjoint HC's in a HM_m. Thus, since a HM_m is also γ-regular with $\gamma = 6$, it belongs to the class Λ.

Theorem 6.3: The edges in any direction of a HM_m describes a HC.

Proof: Without loss of generality, let us consider the +x direction. Referring to the definition of a C-type wrapping, we know that a HC will result if, starting from any row i and applying the definition repeatedly, all rows are visited and no row is visited twice before row i is encountered again. Checking this condition for row $i = 0$, repeated application of the C-type wrapping definition results in the sequence of row visits



(a)



(b)

Figure 6.3: Hex-mesh of size 3 (a) without wrapping and (b) with C-type wrapping.

$$0 \rightarrow m-1 \rightarrow 2m-2 \rightarrow m-2 \rightarrow 2m-3 \rightarrow \cdots \rightarrow 2m - \frac{k+1}{2} \rightarrow m - \frac{k}{2} \rightarrow \cdots$$

where k denotes the order in which the row is visited. Except for the first two rows visited, all rows of the form $2m - \frac{k+1}{2}$ are in the lower part of the hex-mesh and all rows of the form $m - \frac{k}{2}$ are in the upper part of the hex-mesh given $k \leq 2m - 1$. Thus, if $k \leq 2m - 1$, it is clear that no row is visited twice. Setting $m - \frac{k}{2} = 0$, we get $k = 2m$. Thus, on the $2m$-th row visit, row 0 is encountered again, and no row is visited twice before $k = 2m$. Thus, all $2m - 1$ rows must have been visited. **Q.E.D.**

6.3.2. Proposed Solution

The proposed ATA reliable broadcast solution is described for all interconnection networks in the class Λ. Let G be the undirected graph representing any such interconnection network. G is a γ-regular graph (γ even) with $\frac{\gamma}{2}$ undirected edge disjoint HC's. In G^{dir}, there are γ directed HC's $HC_1, HC_2, \cdots, HC_\gamma$. For a given node v, $next_i(v)$, and $prev_i(v)$ denote the nodes immediately following and preceding node v in directed HC HC_i. Let us arbitrarily designate a node as N_0. For any node v, $ID_i(v)$ is the distance from N_0 to v when traversing HC_i. The proposed solution is described below.

IHC Algorithm:

```
For i = 0 to η − 1 do
   begin
      for j = 1 to γ doparallel
         for every node v doparallel
            if ( [ID_i(v)]_η = i ) then
               v sends its message to next_j(v);
      for j = 1 to N − 1 do
         for k = 1 to γ doparallel
            for every node v doparallel
               begin
                  receive message from prev_k(v);      {*}
                  if ( j < N − 1 ) then
```

```
            relay message to next_k(v);      {*}
        end;
    end.
```

The IHC algorithm is performed in η stages. In a single stage, every η-th node in HC_j is permitted to start a message along HC_j for every j $(1 \leq j \leq \gamma)$. η can be considered as the *interleaving distance*. Once messages have been started along directed HC's, they keep flowing for $N - 1$ hops along the cycles in which they started. If there are no other messages in the network, then the two steps indicated by "{*}" correspond to a single cut-through operation at each node. In discussing the IHC algorithm, the outermost **for** loop is referred to as the η stages of the algorithm and the transmission of packets in cycle HC_j during stage i is referred to as a HC_j^i-cycle.

We now address the implementation of the IHC algorithm using virtual cut-through. Let us use an architecture similar to the routing controller chip [23] for HARTS [10], a 19-node (HM_3) version of which is currently being built at the Real-Time Computing Laboratory. A crucial feature in the HARTS routing controller chip is that all incoming and outgoing links (receivers and transmitters) *can* be used simultaneously. Thus, such a capability is also assumed in our architecture, shown in Fig. 6.4.

In high bandwidth communication networks, it has been observed that a large portion (about 80%) of the communication latency is spent in the processing at the transmitters and receivers [35]. (The delay caused by the actual transmission accounts for only 20% of the latency.) Virtual cut-through can be seen as an attempt to eliminate much of the processing (i.e., store-and-forward) at the intermediate nodes between a source and a destination. Thus, when an incoming message cuts through a node, only a very small amount of processing (with the aid of special hardware) is required to determine where and how to advance the message. In Fig. 6.4, the length of the FIFO buffer is determined by the minimum number of bytes that

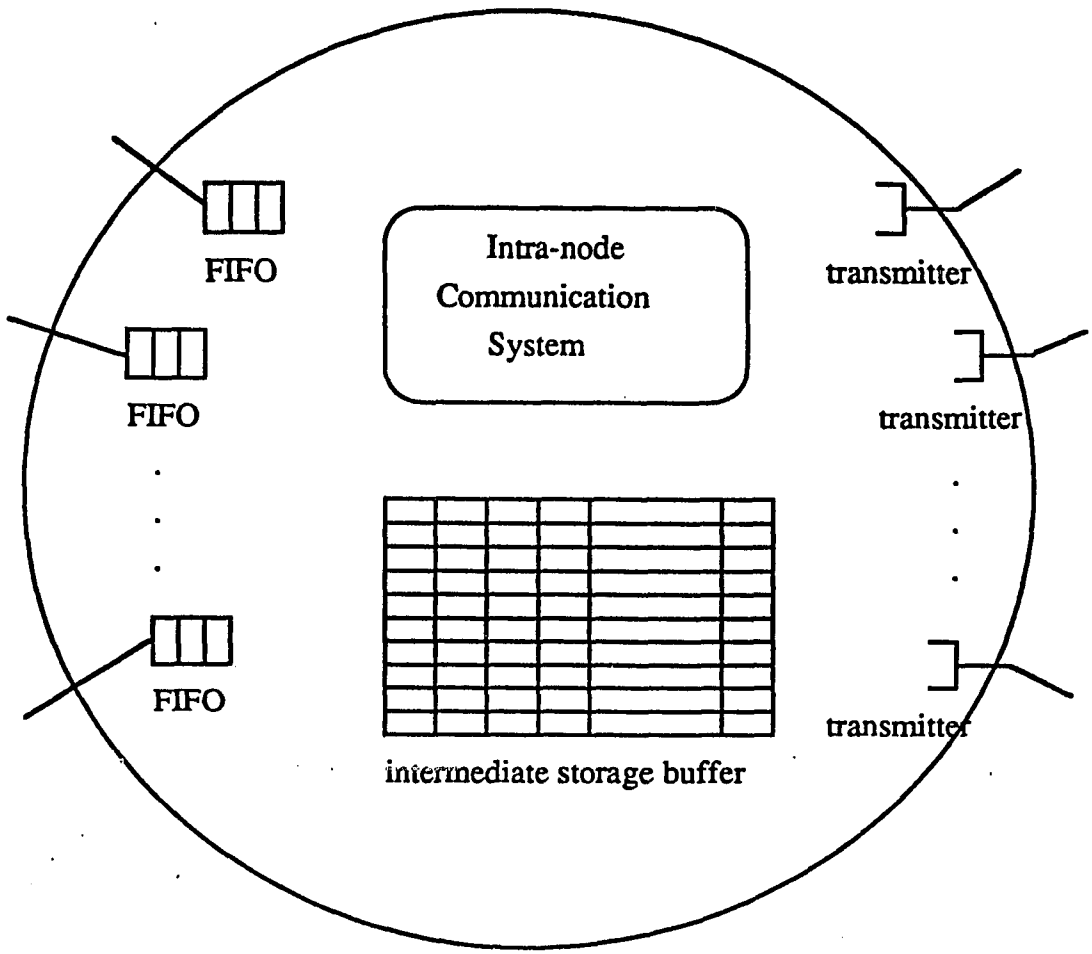


Figure 6.4: Node architecture to support virtual cut-through.

must be seen and the additional delay required to determine the outgoing transmitter. Thus, the FIFO buffer has a length of at least three words: one word for start-of-packet, one word for the type of packet (control, low/high priority, broadcast, etc.), and one word for a routing tag to be used in routing decisions. Many applications in which ATA reliable broadcast is required, such as clock synchronization and distributed diagnosis of intermittently faulty processors, have very short messages that must be broadcast (such as a single clock value or γ bits). For other applications, the messages can be split into fixed size packets. Thus, for the IHC algorithm, a packet size of $\mu \times B_{FIFO}$ is used, where B_{FIFO} is the size of the FIFO buffers at the receivers and μ is a small integer greater than 1.

Under ideal conditions, in which all nodes can operate perfectly synchronously and there are no other messages in the network, the IHC algorithm can be executed with $\eta = \mu$ (assuming that $[N]_\mu = 0$). In this case, all possible cut-throughs in the IHC algorithm can occur. If the conditions are less than ideal, then an interleaving distance η that is greater than μ must be used. With $\eta > \mu$, strict synchronization is not required and normal network traffic can be accommodated. The choice of η depends on the degree of link utilization (by the ATA reliable broadcast operation) desired and the degree of synchronization available. Note that even if nodes start sending packets in a cycle HC_j "out of sequence" (possibly due to synchronization inaccuracies), it merely affects the amount of time required and does not affect the correct execution of the algorithm. Also, if normal network traffic or synchronization inaccuracies cause one HC_j^i-cycle to complete before the other HC_k^i-cycles ($k \neq j$), then the nodes on cycle HC_j can start on stage $i + 1$ immediately.

When a packet arrives at a node v on cycle HC_j, it is possible that the transmitter for HC_j is busy. In this case, the packet is stored in the intermediate storage buffer at node v. If the next packet arrives on cycle HC_j before the transmitter for HC_j is made available, then it is also blocked. This packet is then queued behind the first packet. This queue of HC_j

broadcast packets is built up as long as the transmitter for HC_j is busy (since broadcast packets are short, space should not a problem). When the transmitter for HC_j becomes available, the packets in the HC_j broadcast queue are transmitted one after the other with the spacing between packets determined by the capabilities of the transmitters and receivers. If any new HC_j broadcast packet arrives during this time, it is permitted to pass by if possible, and queued otherwise.

There are several ways in which nodes can determine when to stop relaying packets flowing in any given cycle HC_j. One method is to count the number of packets which have been passed. However, this can result in one extra "send" back to the originator of the packet. To avoid this last "send", the suggested method is to add the address of the last node w (with respect to node v) to node v's packet (in the space normally reserved for the routing tag). Then, instead of counting packets, node w simply checks the address of each HC_j broadcast packet to determine if it should stop relaying the packet.

6.3.3. Reliable Broadcast with Virtual Cut-Through

An ATA reliable broadcast algorithm can be described by first presenting the reliable broadcast algorithm for a single node and then showing how all nodes execute this reliable broadcast algorithm. Two methods have previously been proposed for converting a reliable broadcast algorithm into an ATA reliable broadcast algorithm. In the first method, every node executes the reliable broadcast algorithm concurrently and in lock step [26]. In general, this uses 100% of the available link capacity and may require merge and possibly even split operations. These operations can only be done with store-and-forward routing, i.e., it is not possible with virtual cut-through switching.

In the second method, each node executes the reliable broadcast algorithm in turn, with the reliable broadcast for one node starting when the previous node finishes. This method

leaves a certain amount of unused link capacity (for use by other tasks) and can be executed with virtual cut-through if the individual reliable broadcast operations can use virtual cut-through. This method can be considered as an alternative to the proposed solution. Thus, the VRS algorithm, which is the RS algorithm modified to use virtual cut-through, is described. The KS and VSQ algorithms are described next. The VSQ algorithm is a virtual cut-through reliable broadcast algorithm for a SQ_m. These three algorithms can form the basis for ATA reliable broadcast in hypercubes, hex-meshes, and square meshes. VRS-ATA (KS-ATA, VSQ-ATA) is defined to be the ATA reliable broadcast algorithms in which the VRS (KS, VSQ) algorithm is executed for each node in turn, with one node starting the VRS (KS, VSQ) algorithm when the previous node completes.

Hypercube Algorithm

The RS algorithm is based on the *recursive doubling* algorithm for broadcast in a hypercube. The recursive doubling algorithm is illustrated with Example 6.1.

Example 6.1: Suppose that node 0 wishes to broadcast in the Q_3 in Fig. 6.1. This is accomplished as follows:

Step 1: Node 0 sends its packet to node 1 (direction 0).

Step 2: Nodes 0 and 1 simultaneously send the packet to nodes 2 and 3, respectively (direction 1).

Step 3: Nodes 0 through 3 simultaneously send the packet to nodes 4 through 7, respectively (direction 2).

In the RS algorithm, the node that wishes to broadcast a packet sends a copy of the packet to all of its neighbors. Each of its neighbors then simultaneously execute the recursive doubling algorithm. This algorithm requires $\gamma + 1$ (2γ) steps if each node can use all (only

one) of its outgoing communication links at a time. It was proven that if the RS algorithm is used, each node receives γ copies of the packet through γ disjoint paths in the fault-free situation [53]. The execution of the RS algorithm is illustrated with Example 6.2.

Example 6.2: Suppose node 0 wishes to reliably broadcast a packet to all other nodes in a Q_4. The send-receive operations that occur at each step of the algorithm are shown in Table 6.1. The send-receive operations shown in bold in Step 5 of the algorithm (column 8) can be optionally omitted since they simply return copies of the packet to their originator.

	Column Number							
Step #	1	2	3	4	5	6	7	8
1	0→1 0→2 0→4 0→8							
2	1→3 2→6 4→12 8→9							
3	3→7 6→14 12→13 9→11	1→5 2→10 4→5 8→10						
4	7→15 14→15 13→15 11→15	5→13 10→11 5→7 10→14	3→11 6→7 12→14 9→13	1→9 2→3 4→6 8→12				
5	15→14 15→13 15→11 15→7	13→12 11→9 7→3 14→6	11→10 7→5 14→10 13→5	9→8 3→1 6→2 12→4	7→6 14→12 13→9 11→3	5→4 10→8 5→1 10→2	3→2 6→4 12→8 9→1	**1→0** **2→0** **4→0** **8→0**

Table 6.1: Communication patterns for Example 6.2.

To convert the RS algorithm into an efficient virtual cut-through algorithm, it is necessary to convert the maximum number of store-and-forward operations into cut-through operations. When a node u receives a packet from direction i and then immediately sends it on in direction $[i+1]_m$, u will be said to have *forwarded* the packet. When a node u sends a packet

in direction i and later sends a copy of the same packet in direction $[i+1]_m$, u will be said to have *redirected* the packet. Clearly, every time a node forwards a packet, a cut-through operation can be used at that node. Also, every time a node has to initiate or redirect a packet, it cannot be done with cut-through. In Table 6.1, the send-receive operations have been separated into columns. In a given column, all of the send-receive operations except the first and last ones are operations in which the packet is being forwarded to the next node. Whenever a new column is started, a redirection is taking place.

In the VRS algorithm (RS algorithm modified to use virtual cut-through), all send-receive operations in which a packet is forwarded is implemented as a cut-through operation. All send-receive operations in which a packet has to be redirected is implemented as a store-and-forward operation. The longest path in the VRS algorithm consists of $\gamma - 1$ store-and-forward operations and 2 cut-through operations (since there is no redirection of the packets received in Step 2 of Table 6.1).

C-wrapped Hex-mesh

The KS algorithm is an efficient virtual cut-through reliable broadcast algorithm for a HM_m. When a node v wishes to perform a reliable broadcast, it initiates a copy of its broadcast packet in all six directions. The pattern of cut-through and store-and-forward operations are identical for each of the six directions. This pattern is shown for one direction in Fig. 6.5. By inspection, it can be seen that the longest path consists of 3 store-and-forward operations and $2m - 5$ cut-through operations. Assuming $m \geq 2$, the number of cut-throughs required is no more than $2\sqrt{\frac{N+5}{3}} - 5$ since $3m(m-1) + 1 = N$.

Torus wrapped Square Mesh

The VSQ algorithm is a virtual cut-through reliable broadcast algorithm for a SQ_m, an $m \times m$ torus-wrapped square mesh. This algorithm is similar to the KS algorithm. When a node v wishes to perform a reliable broadcast, it initiates a copy of its broadcast packet in each of the four possible directions. Then the pattern of cut-through and store-and-forward operations are identical for each of the four directions. When designing this pattern for one direction, it must be insured that the patterns for any of the other directions do not "interfere" with the patterns for that direction. Interference would correspond to two arrows pointing in the same direction over the same link, which would mean that one packet could block the progress of another. A pattern satisfying this condition for one direction is shown in Fig. 6.6. The longest path length consists of 3 store-and-forward operations and $2\sqrt{N} - 6$ cut-through operations.

6.3.4. Comparative Analysis

In this section, the IHC algorithm is compared with other possible ATA reliable broadcast algorithms on the basis of (1) generality, (2) effect on normal system operation, and (3) execution time. In this analysis, it is necessary to distinguish between the utilization of the available communication links by the ATA reliable broadcast operation and by all other tasks. The former is referred to by the link utilization parameter H and the latter is referred to by ρ. $\rho = 0$ ($\rho = 1$) represents an unloaded (completely congested) system. In the context of this chapter, H is referred to as link utilization and ρ is referred to as normal network traffic.

Generality

In terms of generality, the IHC algorithm is clearly superior to any of the VRS-ATA, KS-ATA, VSQ-ATA, and FRS algorithms since the IHC algorithm can be executed on any topology in the class Λ while all of the other algorithms are each restricted to one topology.

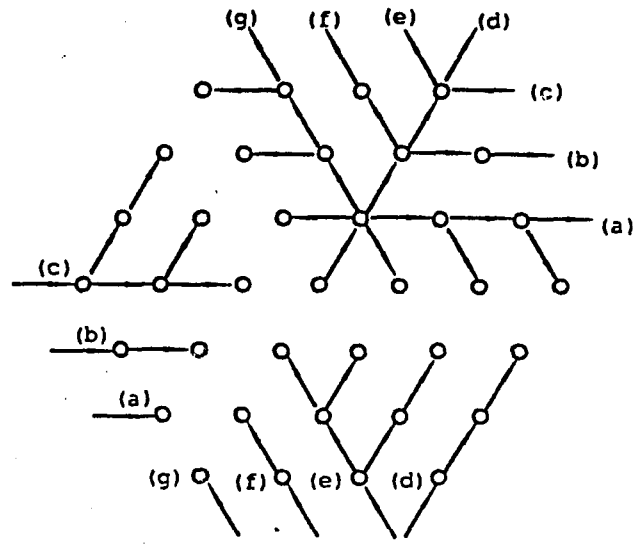


Figure 6.5: KS algorithm initiated in one direction.

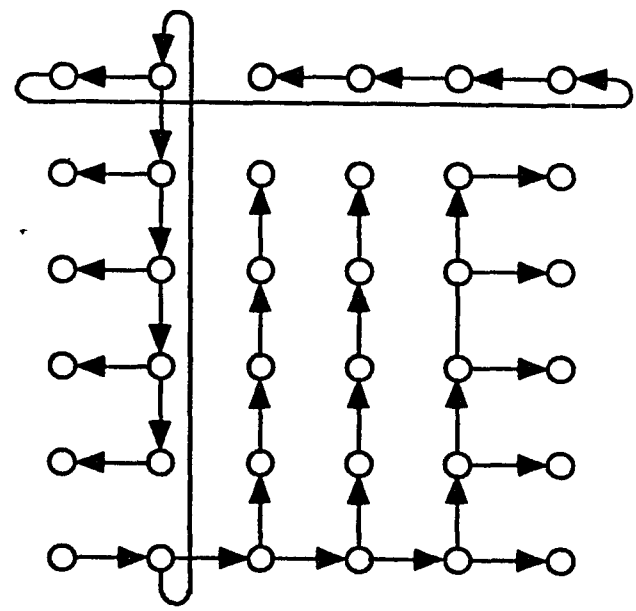


Figure 6.6: VSQ algorithm initiated in one direction.

However, if there is a regular, homogeneous topology for which one wishes to design an ATA reliable broadcast algorithm, then using the same method as in the KS-ATA algorithm, one can design an efficient reliable broadcast algorithm and simply have each node execute the reliable broadcast algorithm in turn. Alternatively, using the same method as in the FRS algorithm, one can have all of the nodes execute the reliable broadcast algorithm in lock step, with messages being merged and split as required. Thus, in this sense, the IHC algorithm is more general *only* because it does not require a new reliable broadcast algorithm to be developed for each new topology.

The ideas presented in the IHC algorithm can be used in a more general manner to design alternative ATA reliable broadcast algorithms. To illustrate, note that when the IHC algorithm is used, the γ disjoint paths taken from a source to a destination are not necessarily the *shortest* paths. The RS algorithm has the property that the γ disjoint paths taken from the source to any other node are the γ shortest such disjoint paths. This is desirable because the probability of successfully sending a packet from a source to a destination is dependent on the length of the path taken.

Let us design an interleaved ATA reliable algorithm which is based on the RS algorithm. To do this, the RS algorithm is first converted to the VRS algorithm. However, referring to Table 6.1, the send-receive operations are coordinated such that Column k is executed after Column $k-1$ for all k $(2 \leq k \leq 7)$. Denoting node k's broadcast packet as M_k, Table 6.2 shows the entire sequence of send-receive operations for all M_k $(0 \leq k \leq 7)$ in a Q_3. If all nodes execute their send-receive operations in lock step, then store-and-forward routing must be used. However, assuming that $\mu = 2$, virtual cut-through can be used if nodes 0, 3, 5, and 6 initiate their broadcast packets in the first stage and nodes 1, 2, 4, and 7 initiate their broadcast packets in a subsequent second stage. In general, for a Q_m, the nodes initiating broadcast packets in the first stage should be such that the distance between any two nodes is two. If

the maximum number of such nodes is chosen for the first stage, then the remaining nodes should also be separated from each other by at least two hops. Thus, two stages are sufficient for any Q_m. This algorithm, denoted as the VRS-IHC algorithm, is similar to the IHC algorithm in that the ATA reliable broadcast is performed in stages to prevent contention for the same communication link by two or more broadcast packets at any time.

Step #	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
1	0→1 0→2 0→4	1→0 1→3 1→5	2→3 2→0 2→6	3→2 3→1 3→7	4→5 4→6 4→0	5→4 5→7 5→1	6→7 6→4 6→2	7→6 7→5 7→3
2	1→3 2→6 4→5	0→2 3→7 5→4	3→1 0→4 6→7	2→0 1→5 7→6	5→7 6→2 0→1	4→6 7→3 1→0	7→5 4→0 2→3	6→4 5→1 3→2
3	3→7 6→7 5→7	2→6 7→6 4→6	1→5 4→5 7→5	0→4 5→4 6→4	7→3 2→3 1→3	6→2 3→2 0→2	5→1 0→1 3→1	4→0 1→0 2→0
4	7→6 7→5 7→3	6→7 6→4 6→2	5→4 5→7 5→1	4→5 4→6 4→0	3→2 3→1 3→7	2→3 2→0 2→6	1→0 1→3 1→5	0→1 0→2 0→4
5	1→5 2→3 4→6	0→4 3→2 5→7	3→7 0→1 6→4	2→6 1→0 7→5	5→1 6→7 0→2	4→0 7→6 1→3	7→3 4→5 2→0	6→2 5→4 3→1
6	5→4 3→1 6→2	4→5 2→0 7→3	7→6 1→3 4→0	6→7 0→2 5→1	1→0 7→5 2→6	0→1 6→4 3→7	3→2 5→7 0→4	2→3 4→6 1→5
7	3→2 6→4 5→1	2→3 7→5 4→0	1→0 4→6 7→3	0→1 5→7 6→2	7→6 2→0 1→5	6→7 3→1 0→4	5→4 0→2 3→7	4→5 1→3 2→6

Table 6.2: Communication patterns using the VRS-IHC algorithm on a Q_3.

Effects on Normal System Operation

The ATA reliable broadcast operation affects normal system operation by imposing extra load on the processing nodes, increasing the link utilization factor, and delaying the transmission of packets created by other normal system tasks. When a cut-through occurs at a node, the node processor does not have to examine and process the packet going through the node. Thus, given the same number of send-receive operations, virtual cut-through algorithms have

an advantage over store-and-forward algorithms in the amount of processing required by the node processor. However, in the FRS algorithm, only $\gamma N(\gamma + 1)$ send-receive operations are required as opposed to $\gamma N(N - 1)$ for the virtual cut-through algorithms. Thus, if the normal network traffic is high, the FRS algorithm may incur less processing cost than the virtual cut-through algorithms. However, it should also be noted that the FRS algorithm requires messages to be merged and split. Depending on the implementation, such operations may incur an expensive processing cost. Among the virtual cut-through algorithms, the IHC algorithm requires the least number of store-and-forward operations if there is no normal network traffic. Even with normal network traffic, with a suitable choice of η, the IHC algorithm requires the least number of store-and-forward operations. This is elaborated on shortly.

A high level of link utilization for the ATA reliable broadcast operation implies a proportionately large delay in transmitting packets created by other (normal) tasks. Two important aspects of this delay are the *expected* and *maximum* delays experienced in transmitting packets created by normal tasks. The expected delay experienced by a normal task's packet is an increasing function of H. In general, the link utilization H is a function of time. Let H_{max} denote the maximum link utilization. For the IHC algorithm, $H_{max} = \frac{\mu}{\eta}$. For the VRS-ATA algorithm, $H_{max} = 0.5$ since $\frac{N}{4}$ nodes can be sending packets of length $\mu \geq 2$ over 2-hop routes simultaneously in all directions in the worst case (refer to Table 6.1). For the KS-ATA and VSQ-ATA algorithms, $H_{max} \geq 6(\sqrt{(N + 5)/3} - 1)/N$ and $H_{max} \geq 4/\sqrt{N}$, respectively, since that many (times N) send-receive operations can occur simultaneously. Thus, with a suitable choice of η, the IHC algorithm has lower link utilization than any of the other algorithms (recall that $H_{max} = 1.0$ for the FRS algorithm). Also, the IHC algorithm has an advantage over all the other algorithms considered in that H_{max} can be flexibly adjusted by changing η. Fig. 6.7 shows a plot of H_{max} versus η with $N = 1000$.

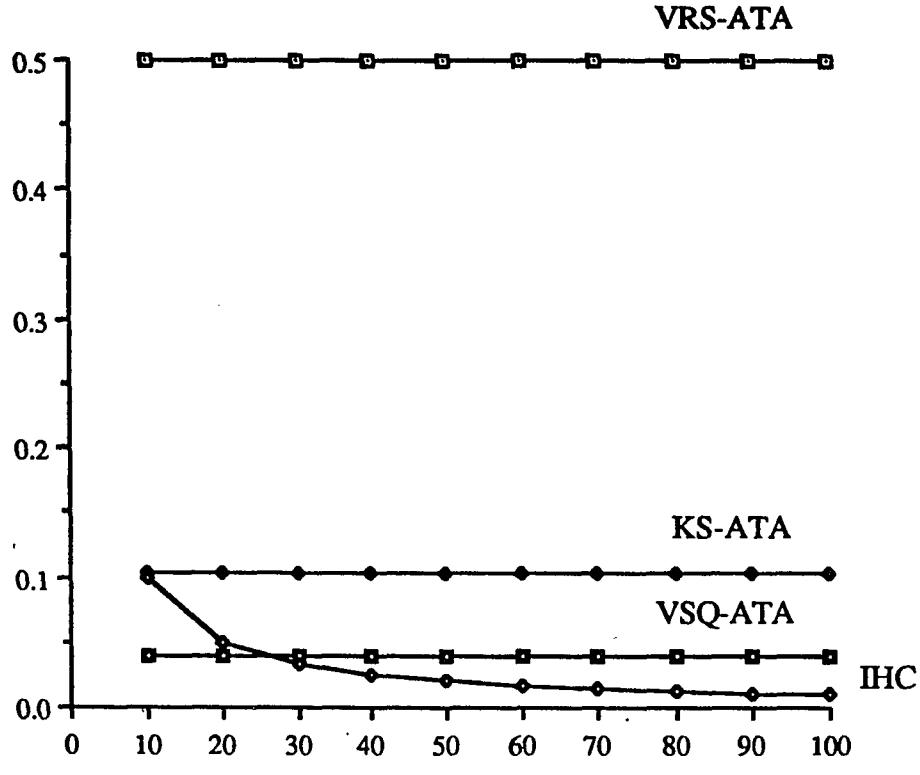


Figure 6.7: H_{max} as a function of η with $N = 1000$.

The maximum delay experienced by a packet is dependent on the number of times that it has to wait for a broadcast packet and the length of the broadcast packet it has to wait for. In routing a normal system task packet P_{nst}, the length of the path chosen is an upper bound on the number of times P_{nst} has to wait for broadcast packets. With random network traffic, it is not possible to control the number of times P_{nst} has to wait for other packets. However, the length of broadcast packets is fixed in the virtual cut-through algorithms and variable in the FRS algorithm. In fact, in the FRS algorithm, the broadcast messages sent in the final stage are of length $(\frac{N}{2} - 1)\mu B_{FIFO}$. In the virtual cut-through algorithms, all broadcast messages are of length μB_{FIFO}. Thus, in terms of the expected and maximum delays experienced by P_{nst}, the FRS algorithm is the least desirable.

Execution Time

In sending a packet from a node v to its neighbor w, there are several time parameters of interest. If the packet cuts through node v, then the delay experienced is denoted by α, which is proportional to B_{FIFO}. If the packet is stored into the intermediate storage buffer before being transmitted, then the delay experienced is $\tau_S + L\tau_L$. Since a packet has fixed length and fits into exactly μ FIFO buffers, $L\tau_L = \mu\alpha$. If the packet has to wait because the transmitter to node w is busy, then the additional queueing delay experienced is denoted by D.

Let us consider the ATA reliable broadcast operation with $\rho = 0$. The times required by the IHC, VRS-ATA, KS-ATA, SQ-ATA, and FRS algorithms are shown in Table 6.3. Note that for the KS-ATA algorithm, m refers to the dimension of a HM_m, a C-wrapped hex-mesh. The IHC algorithm performs better than all of the other virtual cut-through algorithms if $\eta \leq \min\{2\log_2 N + 1\ ,\ 2\sqrt{\frac{N+5}{3}} - 3\ ,\ 2\sqrt{N} - 4\}$. Since $\eta \geq \mu$ and μ can be assumed to

be a small fixed constant, this condition is easily achieved. If $\eta = \mu$, the IHC algorithm is also faster than the FRS algorithm. Theorem 6.4 further states that the IHC algorithm with $\eta = \mu$ has the optimum execution time among all algorithms that deliver γ copies of every node's packet to every other node using γ disjoint paths.

Algorithm	Execution Time
IHC	$\eta(\tau_S + \mu\alpha + (N-2)\alpha)$
VRS-ATA	$N((\gamma-1)(\tau_S + \mu\alpha) + 2\alpha)$
KS-ATA	$N(3(\tau_S + \mu\alpha) + (2m-5)\alpha)$
VSQ-ATA	$N(3(\tau_S + \mu\alpha) + (2\sqrt{N}-6)\alpha)$
FRS	$(\gamma+1)\tau_S + (N-1)\mu\alpha$

Table 6.3: Execution times with $\rho = 0$.

Theorem 6.4: When $\rho = 0$, Algorithm IHC with $\eta = \mu$ delivers γ copies of every node's packet to every other node along γ disjoint paths in the minimum possible time.

Proof: Since there are N nodes, each of which must receive γ copies of packets from all other nodes, a total of $\gamma N(N-1)$ packets must be sent and received. This is the exact number of send-receive operations in the IHC algorithm. With $\eta = \mu$, every directed link in G^{dir} is being used during every step of the algorithm. When using virtual cut-through, communication time is minimized if the minimum number of intermediate store-and-forward operations are used. All intermediate send-receive operations in the IHC algorithm can be implemented with cut-through. The IHC algorithm with $\eta = \mu$ executes the minimum number of send-receive operations, among which all but the initial sends are cut-through operations. It also uses the maximum link capacity for the entire duration of the algorithm. Thus, the theorem follows. **Q.E.D.**

Now let us consider the general case of $\rho > 0$. Consider sending a packet from a node v to all other nodes on the cycle HC_j. If the packet cuts through all $N-2$ intermediate nodes, the amount of time required is $\tau_S + \mu\alpha + (N-2)\alpha$. However, if the packet is buffered and

has to wait at i of the $N-2$ intermediate nodes, then the time required is $\tau_S + \mu\alpha + i(\tau_S + \mu\alpha + D) + (N-2-i)\alpha$. Assuming that the network is uniformly loaded and that the utilization of the links (by other tasks) is ρ, the probability that a packet has to be buffered when it requests a link is ρ. Since the probability of having to wait for a link is independent of the probability of having to wait for any other link, the number of times a packet has to wait for a link follows a binomial distribution. Thus, the expected number of times that a packet has to wait for a link on a path of length $N-1$ is $\rho(N-1)$.

In the IHC algorithm, since there are up to N/η packets in a cycle HC_j at any given time, it is possible that a broadcast packet has to be buffered at a node because another broadcast packet which was previously buffered is being transmitted. The probability of this occurrence is dependent on ρ and η. Denoting this probability as ρ', the effective network traffic perceived by broadcast packets in the IHC algorithm is $\rho + \rho'$.

Let $\mu' B_{FIFO}$ denote the length of P_{nst}, a packet created by a normal system task. Let P_1 and P_2 be two consecutive broadcast packets in an HC_j cycle. The initial separation between P_1 and P_2 is $\frac{\eta}{\mu} B_{FIFO}$. For P_2 to be buffered because P_1 is being transmitted, P_1 must previously have been buffered in a node u_k. P_1 could itself have been buffered because another broadcast packet was being transmitted. However, the first such buffering of a broadcast packet must have been due to a packet P_{nst}. Thus, suppose P_1 was buffered due to P_{nst}. Let τ_{res} be the time required to reserve an outgoing transmitter. If priority is given to P_1 and the buffering and subsequent redirection of P_1 is done efficiently, P_1 will have completely left node u_k after $\mu'\alpha + \mu\alpha + 2\tau_{res}$ time units. This says that P_2 will not have to wait for P_1 at node u_k if $\eta > \mu + \mu' + 2\tau_{res}/\alpha$. Also, since the time for a packet to cut through a node includes τ_{res}, it is clear that $\tau_{res} < \alpha$. Therefore, given these considerations, we make the assumption that $\rho' < \rho$, which is reasonable if $\eta \geq \mu + \mu'$. Then, since a broadcast packet

travels $N-1$ hops in a HC_j^i-cycle, the expected number of times it is buffered is $(\rho+\rho')(N-1) < 2\rho(N-1)$.

We consider the maximum expected execution time of a single computational thread of an ATA reliable broadcast algorithm, referred to simply as the maximum expected execution time, as a measure of the performance of the algorithm. For a single stage in the direction of cycle HC_j, the IHC algorithm has expected time $(1-\rho-\rho')(\tau_S+\mu\alpha+(N-2)\alpha) + (\rho+\rho')(N-1)(\tau_S+\mu\alpha+D)$. Since the next stage for cycle HC_j can start immediately after this stage completes, the IHC algorithm in direction HC_j has expected time $\eta(1-\rho-\rho')(\tau_S+\mu\alpha+(N-2)\alpha) + \eta(\rho+\rho')(N-1)(\tau_S+\mu\alpha+D)$. Since the computation in each of the γ directions is similar, this is the maximum expected execution time of the IHC algorithm.

In the VRS-ATA, KS-ATA, and VSQ-ATA algorithms, there are N stages of reliable broadcasts. Thus, the maximum expected execution time is N times the average delivery time for a packet in the longest path of the reliable broadcast. The average delivery time in the longest path of the VRS algorithm is $(1-\rho)((\gamma-1)(\tau_S+\mu\alpha)+2\alpha) + \rho(\gamma+1)(\tau_S+\mu\alpha+D)$. In a single reliable broadcast of the KS-ATA algorithm, the longest path has expected time $(1-\rho)(3(\tau_S+\mu\alpha)+(2m-5)\alpha) + \rho(2m-2)(\tau_S+\mu\alpha+D)$, where $m \geq \sqrt{\frac{N+5}{3}}$ if $m \geq 2$. In a single reliable broadcast of the VSQ-ATA algorithm, the longest path has expected time $(1-\rho)(3(\tau_S+\mu\alpha)+(2\sqrt{N}-6)\alpha) + \rho(2\sqrt{N}-3)(\tau_S+\mu\alpha+D)$. Table 6.4 shows the maximum expected execution times of the ATA reliable broadcast algorithms considered.

Algorithm	Maximum Expected Execution Time
IHC	$\eta(1-\rho-\rho')(\tau_S+\mu\alpha+(N-2)\alpha)+\eta(\rho+\rho')(N-1)(\tau_S+\mu\alpha+D)$
VRS-ATA	$N(1-\rho)((\gamma-1)(\tau_S+\mu\alpha)+2\alpha)+N\rho(\gamma+1)(\tau_S+\mu\alpha+D)$
KS-ATA	$N(1-\rho)(3(\tau_S+\mu\alpha)+(2m-5)\alpha)+N\rho(2m-2)(\tau_S+\mu\alpha+D)$
VSQ-ATA	$N(1-\rho)(3(\tau_S+\mu\alpha)+(2\sqrt{N}-6)\alpha)+N\rho(2\sqrt{N}-3)(\tau_S+\mu\alpha+D)$
FRS	$(1-\rho)((\gamma+1)\tau_S+(N-1)\mu\alpha)+\rho(\gamma+1)(\tau_S+x\mu\alpha+D), \quad 1\le x\le N/2$

Table 6.4: Maximum expected execution times with $\rho > 0$.

Among the virtual cut-through algorithms, the IHC algorithm with a suitable choice of η has the best performance. Using the assumption that $\rho' < \rho$, the maximum expected execution time of IHC is less than $\eta(1-\rho)(\tau_S+\mu\alpha+(N-2)\alpha)+2\eta\rho(N-1)(\tau_S+\mu\alpha+D)$. Thus, IHC has the best performance if $2\eta \le \min\{\log_2 N+1,\ 2\sqrt{\frac{N+5}{3}}-3,\ \sqrt{N}+1\}$. We also need $\eta > \mu+\mu'$ for the assumption of $\rho' < \rho$.

If ρ is close to 1, the FRS algorithm performs better than all of the virtual cut-through algorithms. This can be seen from Table 6.4 by setting $\rho = 1$. The reason for this is that broadcast packets are merged into larger messages in the FRS algorithm. Thus, since there are fewer send-receive operations that have to wait for a communication link to become available, the FRS algorithm completes the ATA reliable broadcast faster. However, this ignores the cost of merging and splitting messages. To determine the value of ρ at which a virtual cut-through algorithm is better than the FRS algorithm, one should first include the cost of merging and splitting messages into an exact equation for the maximum expected execution time of FRS, and then compare maximum expected execution times.

6.4. Nearest-Neighbor Reliable Multicast

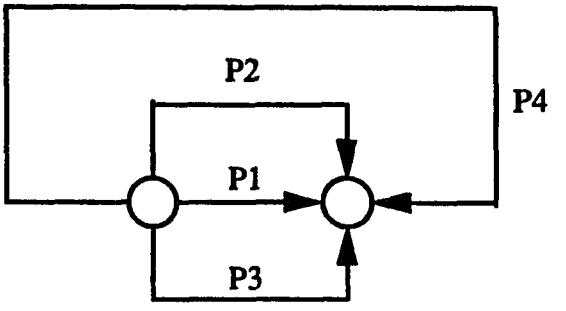
The nearest-neighbor reliable multicast (NNRM) algorithms presented are designed to take advantage of the faster communication possible using virtual cut-through. The main difference between NNRM and ATA reliable broadcast is that NNRM is a much less expen-

sive operation in terms of the required communication and processing overhead. In NNRM, it is only necessary to reliably send γ copies of each node's message to its immediate neighbors. In this section, efficient NNRM algorithms are presented for the SQ_m (torus-wrapped square mesh), HM_m (C-wrapped hex-mesh), and Q_m (hypercube).

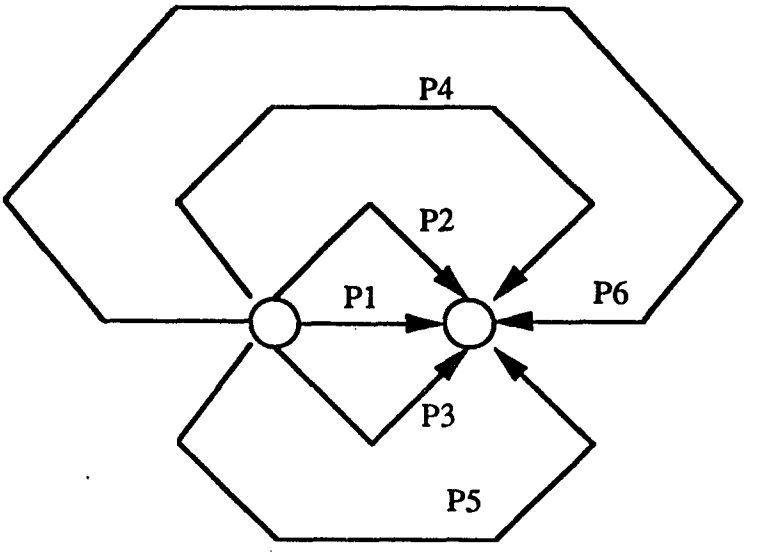
Consider a node u_i reliably sending a message to its neighbor u_j. In order to use the least number of send-receive operations, the set of disjoint paths used must be the set of γ shortest disjoint paths. The set of γ shortest disjoint paths between two adjacent nodes are shown for the square mesh, hex-mesh, and hypercube in Figs. 6.8(a) - (c). In Fig. 6.8(c), $m = \gamma$ is the dimension of the hypercube.

Consider the square mesh. Each node u_i in the square must send four copies of its message to each of its neighbors. The paths that must be followed between node u_i and its neighbor u_j are the four paths shown in Fig. 6.8(a). The problem then is to fit the sixteen paths emanating from each node u_i (four sets of four directed paths to each of u_i's neighbors) onto SQ_m^{dir}, the directed graph corresponding to SQ_m. At least four "stages" of multicast are required since the total number of paths is four times the total number of directed edges in SQ_m^{dir}. However, in order to implement the NNRM such that cut throughs are used, more "stages" of multicast are required since three of the four paths shown in Fig. 6.8(a) pass through intermediate nodes. When a pattern of paths (or partial paths) is laid down on a square grid, it is desirable to lay down the pattern such that no two paths require use of the same directed edge. If this condition is satisfied, then none of the paths contend for the same communication links, i.e., cut throughs can be achieved at all intermediate nodes on a path. This is the strategy used for designing efficient virtual cut-through NNRM algorithms for the square mesh, hex-mesh, and hypercube.

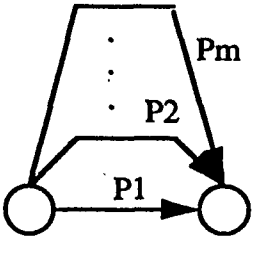
SQ-NNRM, the NNRM algorithm for the square mesh, is described by the number of multicast stages and the pattern of send-receive operations in each stage. The references to



(a)



(b)



(c)

Figure 6.8. Set of γ shortest disjoint paths between adjacent nodes for the (a) square mesh, (b) hex-mesh, and (c) hypercube.

paths are to the set of paths shown in Fig. 6.8(a). Algorithm SQ-NNRM is executed in 13 stages. In stage 1, each node uses path P1. By the end of stage 1, the first part of paths P2, P3, and P4 has already been traversed for every node. In stages 2 and 3, the rest of the path P2 must be traversed for every node. Likewise, in stages 4 and 5, the rest of path P3 must be traversed for every node. Finally, stages 6 through 13 are required to send the rest of path P4 for every node. In each stage, the partial paths are laid down in a pattern such that all of the directed edges are occupied by exactly one partial path. Fig. 6.9(a) shows the pattern for stage 1, Fig. 6.9(b) shows the pattern for stages 2 through 5, and Fig. 6.9(c) shows the pattern for stages 6 through 13.

HM-NNRM, the NNRM algorithm for the hex-mesh, can be described similarly to the SQ-NNRM algorithm. The references to paths are to the set of paths in Fig. 6.8(b). Algorithm HM-NNRM is executed in 35 stages. In stage 1, each node traverses path P1 using the pattern in Fig. 6.10(a). Again, by the end of stage 1, the first part of paths P2 through P6 has already been traversed for every node. The remaining parts of paths P2 and P3 for each node can be traversed with an identical pattern in stages 2 and 3. For the remainder of paths P4 and P5, eight stages of the pattern shown in Fig. 6.10(b) is required. Finally, for the remaining part of path P6, 24 stages of the pattern shown in Fig. 6.10(c) is required.

Q-NNRM, the NNRM algorithm for the hypercube, is similar to the SQ-NNRM algorithm since each set of four dimensions in the hypercube can correspond to the edges in a square mesh. As can be seen from Fig. 6.8(c), Q-NNRM requires only the patterns in Fig. 6.9(a) and (b). The references to paths are to the set of paths in Fig. 6.8(c). In stage 1, each node traverses the path P1 using the pattern in Fig. 6.9(a) replicated in each of the γ directions. After stage 1, the first parts of paths P2 through Pm have been traversed. The rest of the paths are traversed using the pattern of Fig. 6.9(b). For every set of four dimensions in the hypercube, four stages using the pattern of Fig. 6.9(b) is required. Thus, the total number

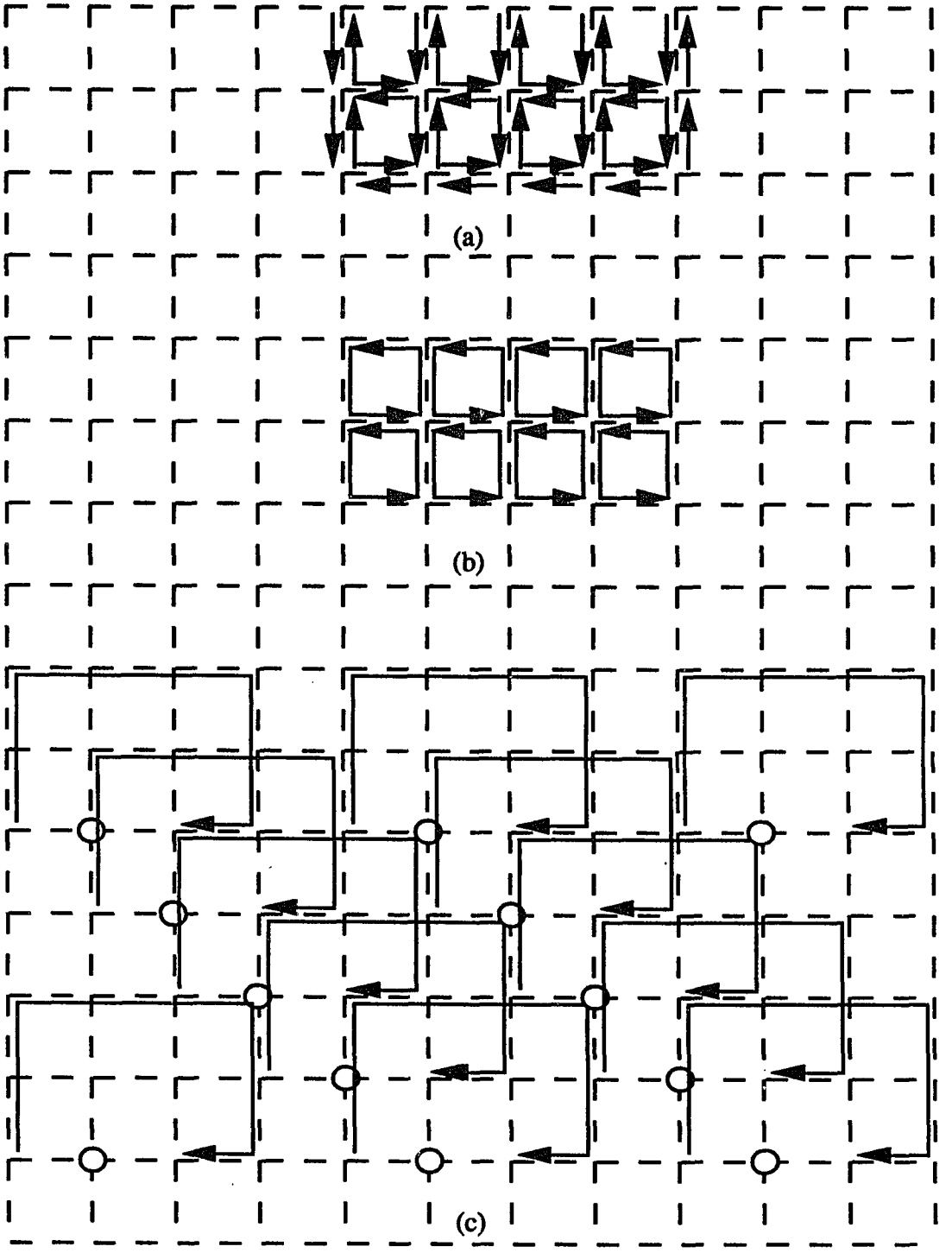


Figure 6.9: SQ-NNRM patterns for stages (a) 1, (b) 2 - 5, (c) 6 - 13.

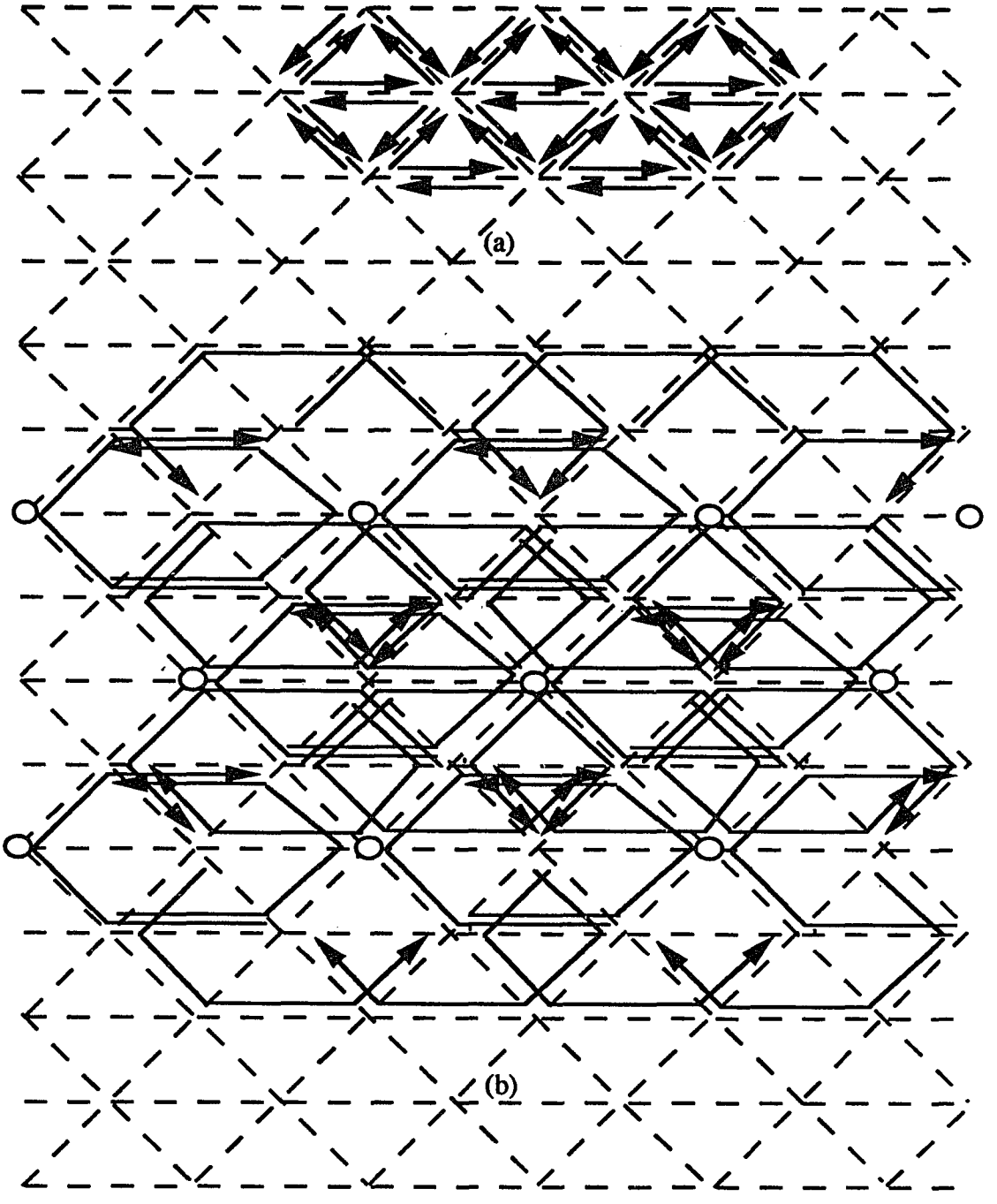


Figure 6.10: HM-NNRM patterns for stages (a) 1 - 3, (b) 4 - 11, and (c) 12 - 35.

(continued on next page)

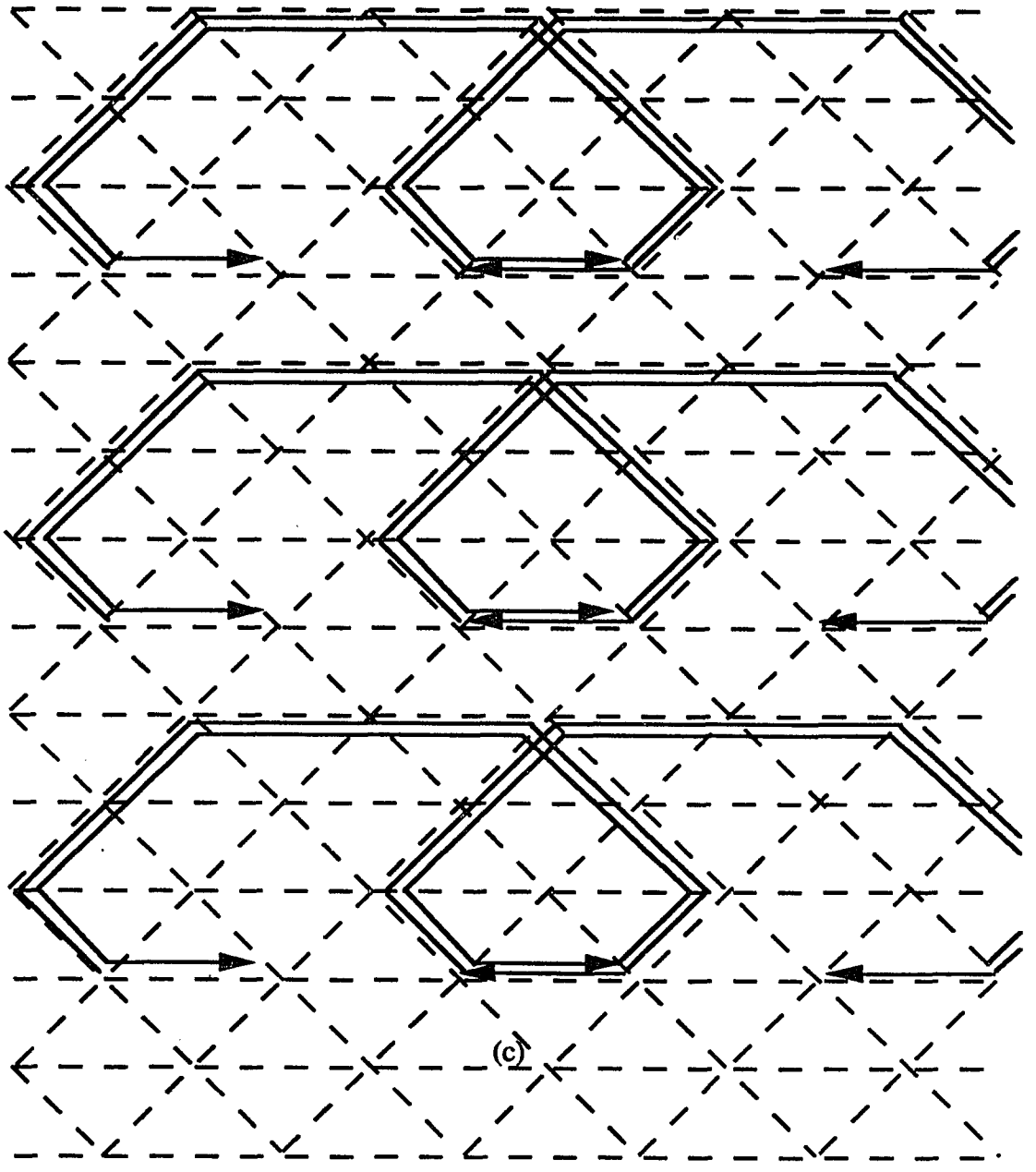


Figure 6.10: continued.

of stages required in an m–dimensional hypercube is $4\lceil m/4 \rceil + 1$.

Note that in the NNRM algorithms as well as in the ATA reliable broadcast algorithms, the times when different copies of a broadcast message arrive at a destination node may be different. There are several ways in which a receiver node can make use of redundant copies that arrive at different times. In [53, 60], several methods are shown for establishing a quorum such that a receiver node can verify a broadcast message before all of the copies of the message have been received. One way of establishing a quorum is to wait until $\lceil 2\gamma/3 \rceil$ copies of the broadcast message arrive before voting on the information contained in them. With this method, a maximum of $\lfloor \gamma/3 \rfloor$ faults can be tolerated. Alternatively, by maintaining a count of identical copies received, a quorum can be established when there are at least $\lceil \gamma/2 \rceil$ identical messages. With this latter method, a maximum of $\lfloor \gamma/2 \rfloor$ faults can be tolerated.

The NNRM algorithms presented in this section are efficient because the number of multicast stages required is small and the length of the longest path in a multicast stage is at most 8 (in the square mesh). Dally and Seitz quoted a time of 150 ns as the time required to perform a cut-through operation using the torus routing chip [20]. Then, assuming that there are no conflicts in using links and that the message startup time is 1 ms (as in the previous section), each multicast stage requires at most 1.0012 ms. For the square mesh and hex-mesh, the total time required for the NNRM operation is approximately 13 ms and 35 ms, respectively, given an arbitrarily large system. For a hypercube of dimension 16 (with 64K nodes), the total time required is approximately 17 ms. These times are substantially less than the execution times required for the ATA reliable broadcast (recall that 2.0 seconds were required for a 10,001 node system). Note that the calculations for the number of multicast stages and execution time required by the NNRM operation used the implicit assumption that the patterns shown in Figs. 6.9 and 6.10 could all fit onto the system topology. However, with a finite processing surface in which the edges at the periphery wrap around, some of the patterns may

not fit perfectly. In that case, the number of multicast stages (and consequently exection time) required should be multiplied by a factor of at most 2 (extra multicast stages are needed to cover the "irregular" parts of the processing surface).

6.5. Conclusion

Distributed diagnosis has been addressed in this chapter. Distributed implementation of diagnosis algorithms in categories 1, 2, 2A, 3, and 3A were all considered. For category 1, it was seen that an all-to-all reliable broadcast operation was required. For categories 2 and 2A, a method was presented for reducing the distributed diagnosis problem to a nearest-neighbor reliable multicast problem. Diagnosis algorithms in categories 3 and 3A can also be implemented with a nearest-neighbor reliable multicast solution.

The all-to-all (ATA) reliable broadcast problem was considered for a class of regular interconnection networks. Although good solutions to this problem exist if store-and-forward routing is assumed, better solutions can be found by taking advantage of the faster communication possible with virtual cut-through. When designing an ATA reliable broadcast algorithm that uses virtual cut-through, the objective is to ensure that all possible cut-throughs can be achieved when there is no other network traffic. This implies that nodes must coordinate their actions such that "collisions" of broadcast packets do not occur. One method for doing this to stagger the nodes that initiate reliable broadcasts and to finish the individual reliable broadcasts as fast as possible. Another method is to interleave the nodes that initiate reliable broadcasts. However, with this latter method, the reliable broadcasts by individual nodes should be performed such that two broadcast packets from the same or different nodes do not interfere with each other's progress.

In this chapter, a general method has been proposed for performing ATA reliable broadcast in an interleaved manner. The proposed solution, the IHC algorithm, compares favorably

to several other possible virtual cut-through and store-and-forward algorithms in terms of generality of usage, the effect on normal system operation, and execution time. Unlike the other algorithms studied, the IHC algorithm can be used on a large class of interconnection networks. In addition, using the ideas presented in the IHC algorithm, we can design interleaved ATA reliable broadcast algorithms based on efficient reliable broadcast algorithms. ATA reliable broadcast is an expensive operation which affects normal system operation by imposing extra load on the processing nodes and delaying the transmission of packets created by normal system tasks. Unless the normal network traffic is heavy, the virtual cut-through solutions incur less processing cost than the store-and-forward solutions since cut-through is handled at the link level with the aid of special hardware. Also, with a suitable choice of the interleaving distance η, the IHC algorithm introduces the least delay on the transmission of normal system task packets. In terms of execution time, the IHC algorithm is shown to be optimal if there is no other network traffic. In heavy network traffic, a store-and-forward algorithm that merges packets as they are being relayed has the best overall performance. In light to medium network traffic, the IHC algorithm has the best performance among the virtual cut-through algorithms considered if η is chosen such that

$$\mu + \mu' < \eta \leq \frac{1}{2} \min\{\log_2 N + 1 \, , \, 2\sqrt{\frac{N+5}{3}} - 3 \, , \, \sqrt{N} + 1\}.$$

Nearest-neighbor reliable multicast was addressed by identifying a set of minimal-length disjoint paths and then addressing the problem of fitting the paths onto the topological structure such that none of the paths contend for the same communication links. If this property is satisfied, then cut through can be used at all intermediate nodes in the paths. Specific solutions to the nearest-neighbor reliable multicast problem were presented for the square mesh, hex-mesh, and hypercube topologies. It was shown that these solutions required significantly less processing time than all-to-all reliable broadcast.

CHAPTER 7

DISCUSSION

7.1. Summary

This dissertation has addressed the general problem of locating faulty processors in a large multiprocessor/multicomputer system. The diagnosis is done on the basis of a fault syndrome consisting of a set of binary pass-fail tests conducted by processors on each other. The result of an inter-processor test can be incorrect because (1) the testing processor is faulty or (2) the tested processor is faulty but the test is not able to catch the fault. The diagnosis task is difficult because a consistent and highly probable set of faulty nodes must be identified based on a collection of incomplete tests.

The main results of this dissertation are:

- an algorithm for finding the most probable diagnosis given the fault syndrome,
- the derivation of probabilistic algorithms which are optimal in diagnostic accuracy given limited fault syndrome information,
- the development of an optimal multiple syndrome probabilistic diagnosis algorithm, and
- efficient methods for implementing all of the diagnosis algorithms presented in a distributed manner.

The optimal diagnosis algorithm is the one which finds the most probable diagnosis given the fault syndrome. However, the problem of finding the most probable diagnosis given

the entire syndrome information is shown to be an NP-hard problem. In Chapter 3, this problem is formulated as a search problem and methods are developed for conducting the search in an efficient manner such that the most probable diagnosis can be found in systems with up to several hundred nodes.

In Chapter 4, optimal probabilistic diagnosis algorithms are presented for diagnosis using limited fault syndrome information. Because optimal diagnosis given the entire syndrome is an NP-hard problem, previous researchers have proposed heuristic algorithms which are efficient and produce "good" diagnostic accuracy results. However, their algorithms were found to use a limited form of the available fault syndrome information. Given that the same type of fault syndrome information is being used, OPT2A (one of the algorithms presented in Chapter 4) is an optimal $O(N^2)$ diagnosis algorithm, where N is the number of nodes in the system. Diagnosis algorithms are categorized based on the type of fault syndrome information used in the diagnosis. For each such category defined, Bayesian probability analysis is used to derive an optimal probabilistic diagnosis algorithm. All algorithms derived (except for category 1: unrestricted syndrome information) have quadratic or lower computational complexity. Because these algorithms are based on the use of probability parameters, simulations were conducted to show that the algorithms perform well even with inaccurate probability parameter estimates. Furthermore, the probabilistic analysis methods developed permit us to evaluate heuristic diagnosis algorithms, such as those presented by previous researchers, based on how closely they approximate the optimal algorithm for that category.

In certain situations, it is possible to obtain a significantly better diagnosis by using multiple rather than a single fault syndrome. Chapter 5 presents a multiple syndrome probabilistic diagnosis algorithm which is optimal given limited fault syndrome information. Through analysis of probability distributions, it is shown that significantly better diagnosis can be achieved by using multiple rather than a single fault syndrome given a special inter-processor

testing method described in Chapter 5. Previous researchers presented heuristic diagnosis algorithms for multiple syndrome probabilistic diagnosis. Given that the same type of fault syndrome information is used, our algorithm is optimal in diagnostic accuracy.

Chapter 6 addresses distributed diagnosis. A method is shown for distributing a special type of global fault syndrome information with a small number of reliable communication steps between nodes which are physically adjacent in the interconnection network. Next, solutions are presented for all-to-all reliable broadcast, in which every node is required to reliably send a message to every other node, and nearest-neighbor reliable multicast, in which every node is required to reliably send a message to its nearest neighbors. The solutions presented are designed to take advantage of the faster communication possible with virtual cut-through switching.

In Chapter 1, seven objectives were identified in designing useful and practical solutions to the multiprocessor/multicomputer diagnosis problem. All of these objectives are satisfied to some degree by the algorithms developed in this dissertation. All of the diagnosis algorithms presented (MPD, OPT2, OPT2A, OPT3, OPT3A, DSK*, and OPTM) are probabilistic algorithms that can be implemented on all types of system topologies and can handle both intermittent and permanent faults. The reliable broadcast and multicast algorithms presented (IHC, SQ-NNRM, HM-NNRM, and Q-NNRM) permit these diagnosis algorithms to be implemented in a distributed manner with relatively low communication and processing overhead. The MPD algorithm is optimal in terms of diagnostic accuracy if all of the fault syndrome information is used. However, because the specific problem solved by the MPD algorithm is NP-hard, MPD can only be executed on systems with small numbers of faulty nodes.

The other diagnosis algorithms (OPT2, OPT2A, OPT3, OPT3A, DSK*, and OPTM) are "close to optimal" in terms of diagnostic accuracy. However, these algorithms are all efficient (quadratic or lower computational complexity) and achieve good diagnostic accuracy

even when inter-processor tests with fairly low fault coverage are used. Since good diagnosis results can be obtained with low fault coverage tests, a relatively short amount of time needs to be devoted to the testing required to obtain the fault syndrome. Finally, it was shown that all of the diagnosis algorithms presented work well even when inaccurate estimates are used for the probability parameters required.

In summary, the main contribution of this dissertation has been to show that it is possible to derive efficient and optimal probabilistic diagnosis algorithms if limited fault syndrome information is used.

7.2. Future Work

There are three main tasks worth further investigation. These include accurate probability parameter estimation, design of better and more general diagnosis algorithms (particularly for multiple syndrome diagnosis), and hardware experimentation.

While it has been shown that the diagnosis algorithms presented in this dissertation perform well even with inaccurate estimates of probability parameters, accurate probability parameter estimates are nevertheless required to obtain the best diagnosis results. Methods are needed to acquire accurate estimates of probability parameters such as prior fault probability of nodes and fault coverage of inter-processor tests. It is particularly difficult to obtain an accurate estimate of fault coverage since nodes can be intermittently faulty.

Next, better and more general diagnosis algorithms are desired, particularly for multiple syndrome diagnosis. In Chapter 5, it was shown that by using a special testing method and multiple fault syndromes, significantly better diagnosis can be achieved than traditional diagnosis methods based on single fault syndromes. However, the method described is restricted to the comparison-testing inter-processor testing method and is optimal only if limited fault syndrome information is used. It should be possible to derive better and more general multi-

ple syndrome diagnosis methods by using probabilistic analysis methods similar to those developed in this dissertation.

The most important future task is implementation of the proposed algorithms on a real system and measurement of their performances. Although simulations have been conducted to show that the diagnosis algorithms perform well given various combinations of probability parameters and system topologies, experimentation with actual multiprocessor or multicomputer systems is required to verify the practicality and usefulness of the probabilistic diagnosis methods. Such experimentation would be extremely difficult with a commercial multiprocessor/multicomputer system since realistic fault injection (for both intermittent and permanent faults) is required to be able to evaluate various inter-processor testing methods and high-level diagnosis methods. Thus, the experiments can be performed on HARTS, an experimental multicomputer currently being built in the Real-Time Computing Laboratory. Since the operating system (HARTOS) [36], interconnection network, and network interface processor are all being designed and built by us, we have the flexibility to make low-level changes to the multicomputer system and conduct realistic experiments.

The proposed experiments are divided into several parts. Our diagnosis methods are based on the results of system-level inter-processor tests. Several inter-processor testing methods, including comparison-testing and on-line processor monitoring, have been described in Chapter 2. The first part of the experiments will be aimed at evaluating the fault coverage obtained using various combinations of these testing methods. Based upon these initial experiments, several testing mechanisms can be selected to be used in subsequent experiments in which the effectiveness of various diagnosis methods can be evaluated.

Fault injection methods [2, 12, 31, 58] are required to inject faults for our experiments since the occurrence of natural faults are rare and not easily reproducible. Fault injection can be accomplished through hardware or software. Due to limited resources, software fault

injection methods are proposed for our experiments. The types of faults to be injected include memory faults, register faults, and communication faults. To inject memory faults it is necessary to alter the contents of memory at certain locations. In an operating environment with virtual memory, this can be achieved with operating system support by marking the particular blocks as unavailable and handling the resulting page faults appropriately. It is also possible to alter memory belonging to a particular task since the system task has knowledge of the internal data structures of the operating system. Register faults can be injected in a similar fashion. Communication faults, which are manifested as missing or corrupted messages, can be inserted using a monitor process in the network processor that screens packets and occasionally corrupts a few packets. The monitor process can also be used to suppress all packets originating from, or destined to, a particular process.

For these fault insertion and testing experiments, we must design and develop HARTOS system and application tasks. A fault injection support task is required at each node for the software fault injection. The implementation of these experiments also requires the development of several utilities and support programs, including utilities for specifying the types, locations, and the numbers of faults to be injected into the system. Tools are also needed to streamline the generation of the executable objects from a set of possible test tasks, given the parameters for the experiment.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Adham and A. D. Friedman, "Digital system fault diagnosis," *Design Automation and Fault-Tolerant Computing*, vol. 1, no. 2, pp. 115-132, February 1977.

[2] Jean Arlat, Yves Crouzet, and Jean-Claude Laprie, "Fault injection for dependability validation os fault-tolerant computing systems," *Digest of Papers, FTCS-19*, pp. 348-355, 1989.

[3] B. Becker and H. U. Simon, "How robust is the n-cube?," *Proc. 27th Ann. Symp. on Found. of Comput. Sci.*, pp. 283-291, October 1986.

[4] D. M. Blough and G. M. Masson, "Performance analysis of a generalized upset detection procedure," *Digest of Papers, FTCS-17*, pp. 218-223, 1987.

[5] D. M. Blough, G. F. Sullivan, and G. M. Masson, "Almost certain diagnosis for intermittently faulty systems," *Digest of Papers, FTCS-18*, pp. 260-265, 1988.

[6] D. M. Blough, *Fault Detection and Diagnosis in Multiprocessor Systems*, Ph. D. Dissertation, The Johns Hopkins University, Baltimore, Maryland, 1988.

[7] M. L. Blount, "Probabilistic treatment of diagnosis in digital systems," *Digest of Papers, FTCS-7*, pp. 72-77, 1977.

[8] J. A. Bondy and U. S. R. Murty, *Graph Theory with Applications*, New York, NY: North-Holland, 1976.

[9] J. T. Butler, "Diagnosis of intermittently faulty and permanently faulty processors in a multiprocessing system using three-valued functions," *12th Int'l Symp. on Multiple Valued Logic*, pp. 122-128, May 1982.

[10] M. S. Chen, K. G. Shin, and D. D. Kandlur, "Addressing, routing and broadcasting in hexagonal mesh multiprocessors," *IEEE Trans. Comput.*, vol. C-39, no. 1, pp. 10-18, January 1990.

[11] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Annals Math. Stat.*, vol. 23, pp. 493-507, 1952.

[12] Ram Chillarege and Nicholas S. Bowen, "Understanding large system failures - a fault injection experiment," *Digest of Papers, FTCS-19*, pp. 356-363, 1989.

[13] K. Y. Chwa and S. L. Hakimi, "Schemes for fault-tolerant computing: a comparison of modularly redundant and t-diagnosable systems," *Information and Control*, vol. 49, pp. 212-238, June 1981.

[14] K. Y. Chwa and S. L. Hakimi, "On fault identification in diagnosable systems," *IEEE Trans. Comp.*, vol. C-30, no. 6, pp. 414-422, June 1981.

[15] A. T. Dahbura and G. M. Masson, "Self-implicating structures for diagnosable systems," *Digest of Papers, FTCS-13*, pp. 332-335, 1983.

[16] A. T. Dahbura and G. M. Masson, "Greedy diagnosis of hybrid fault situations," *IEEE Trans. Comput.*, vol. C-32, no. 8, pp. 777-782, August 1983.

[17] A. T. Dahbura and G. M. Masson, "An $O(n^{2.5})$ fault identification algorithm for diagnosable systems," *IEEE Trans. Comp.*, vol. C-33, no. 6, pp. 486-492, June 1984.

[18] A. T. Dahbura, "An efficient algorithm for identifying the most likely fault set in a probabilistically diagnosable system," *IEEE Trans. Comp.*, vol. C-35, no. 4, pp. 354-356, April 1986.

[19] A. T. Dahbura, K. K. Sabnani, and L. L. King, "The comparison approach to multiprocessor fault diagnosis," *IEEE Trans. Comp.*, vol. C-36, no. 3, pp. 373-378, March 1987.

[20] W. J. Dally and C. L. Seitz, "The torus routing chip," *J. Distributed Computing*, vol. 1, no. 3, pp. 187-196, 1986.

[21] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. C-36, no. 5, pp. 547-553, May 1987.

[22] D. Dolev, "The Byzantine generals strike again," *J. Algorithms*, vol. 3, pp. 14-30, 1982.

[23] J. W. Dolter, P. Ramanathan, and K. G. Shin, "A microprogrammable VLSI routing controller for HARTS," *Proceedings of ICCD '89*, pp. 160-163, October 1989.

[24] M. Foregger, "Hamiltonian decompositions of products of cycles," *Discrete Math.*, vol. 24, pp. 251-260, 1978.

[25] M. Foregger, personal communication, December 1989.

[26] P. Fraigniaud, "Asymptotically optimal broadcast and total-exchange algorithms in faulty hypercube multicomputers," *Laboratoire de l'Informatique du Parallelisme*, Ecole Normale Superieure de Lyon, May 1989.

[27] A. D. Friedman, "A new measure of digital system diagnosis," *Digest of Papers, FTCS-5*, pp. 167-170, 1975.

[28] A. D. Friedman and L. Simoncini, "System-level fault diagnosis," *Computer*, pp. 47-53, March 1980.

[29] H. Fujiwara and K. Kinoshita, "Connection assignments for probabilistically diagnosable systems," *IEEE Trans. Comp.*, vol. C-27, no. 3, pp. 280-283, March 1978.

[30] D. Fussell and S. Rangarajan, "Probabilistic diagnosis of multiprocessor systems with arbitrary connectivity," *Digest of Papers, FTCS-19*, pp. 560-565, 1989.

[31] Ulf Gunneflo, Johan Karlsson, and Jan Torin, "Evaluation of error detection schemes using fault injection by heavy-ion radiation," *Digest of Papers, FTCS-19*, pp. 340-347, 1989.

[32] S. L. Hakimi and A. T. Amin, "Characterization of connection assignment of diagnosable systems," *IEEE Trans. Comp.*, vol. C-23, no. 1, pp. 86-88, January 1974.

[33] C. T. Ho, personal communication, November 1989.

[34] S. H. Hosseini, J. G. Kuhl, and S. M. Reddy, "On self-fault diagnosis of the distributed systems," *IEEE Trans. Comp.*, vol. 37, no. 2, pp. 248-251, February 1988.

[35] H. Kanakia and D. R. Cheriton, "The VMP network adapter board (NAB): high-performance network communication for multiprocessors," *SIGCOMM '88*, pp. 175-187, August 1988.

[36] D. D. Kandlur and K. G. Shin, "Reliable broadcast in hexagonal mesh multiprocessors," *submitted for publication*, November 1989.

[37] A. Kavianpour and A. D. Friedman, "Efficient design of easily diagnosable systems," *3rd USA-JAPAN Computer Conference*, pp. 251-257, 1978.

[38] P. Kermani and L. Kleinrock, "Virtual cut-through: a new computer communication switching technique," *Computer Networks*, vol. 3, no. 4, pp. 267-286, September 1979.

[39] C. M. Krishna, K. G. Shin, and R. W. Butler, "Ensuring fault tolerance of phase-locked clocks," *IEEE Trans. Comput.*, vol. C-34, no. 8, pp. 752-756, August 1985.

[40] J. G. Kuhl and S. M. Reddy, "Distributed fault-tolerance for large multiprocessor systems," *7th Int'l Symp. on Comp. Arch.*, pp. 23-30, May 1980.

[41] L. Lamport, R. Shostak, and M. Pease, "The Byzantine generals problem," *ACM Trans. Prog. Languages and Systems*, vol. 4, no. 3, pp. 382-401, July 1982.

[42] L. Lamport and P. M. Melliar-Smith, "Synchronizing clocks in the presence of faults," *J. ACM*, vol. 32, no. 1, pp. 52-78, January 1985.

[43] S. Lee and K. G. Shin, "Incremental incomplete-test system-level diagnosis," *submitted for publication*, June 1989.

[44] S. Lee and K. G. Shin, "Optimal multiple syndrome probabilistic diagnosis," *submitted for publication*, November 1989.

[45] S. Lee and K. G. Shin, "Comparison of probabilistic diagnosis schemes," *Real-Time Computing Laboratory*, August 1989.

[46] S. Lee and K. G. Shin, "Optimal and efficient probabilistic distributed diagnosis schemes," *submitted for publication*, November 1989.

[47] S. H. Maheswari and S. L. Hakimi, "On models for diagnosable systems and probabilistic fault diagnosis," *IEEE Trans. Comp.*, vol. C-25, no. 3, pp. 228-236, March 1976.

[48] S. Mallela and G. M. Masson, "Diagnosable systems for intermittent faults," *IEEE Trans. Comp.*, vol. C-27, no. 6, pp. 560-566, June 1978.

[49] S. Mallela and G. M. Masson, "Diagnosis without repair for hybrid fault situations," *IEEE Trans. Comp.*, vol. C-29, no. 6, pp. 461-470, June 1980.

[50] G. G. L. Meyer and G. M. Masson, "An efficient fault diagnosis algorithm for symmetric multiple processor architectures," *IEEE Trans. Comp.*, vol. C-27, no. 11, pp. 1059-1063, November 1978.

[51] Y. Peng and J. A. Reggia, "A comfort measure for diagnostic problem solving," *Information Sciences*, vol. 47, pp. 149-184, 1989.

[52] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems," *IEEE Trans. Electron. Comp.*, vol. EC-16, no. 6, pp. 848-854, December 1967.

[53] P. Ramanathan and K. G. Shin, "Reliable broadcast in hypercube multicomputers," *IEEE Trans. Comput.*, vol. 37, no. 12, pp. 1654-1657, December 1988.

[54] P. Ramanathan, D. D. Kandlur, and K. G. Shin, "Hardware-assisted software clock synchronization for homogeneous distributed systems," *IEEE Trans. Comput.*, vol. C-39, no. 4, April 1990 (in press).

[55] S. Rangarajan and D. Fussell, "A probabilistic method for fault diagnosis of multiprocessor systems," *Digest of Papers, FTCS-18*, pp. 278-283, 1988.

[56] J. D. Russell and C. R. Kime, "System fault diagnosis: closure and diagnosability with repair," *IEEE Trans. Comp.*, vol. C-24, no. 11, pp. 1078-1088, November 1975.

[57] J. D. Russell and C. R. Kime, "System fault diagnosis: masking, exposure, and diagnosability without repair," *IEEE Trans. Comp.*, vol. C-24, no. 12, pp. 1155-1161, December 1975.

[58] Z. Segall, D. Vrsalovic, D. Siewiorek, D. Yaskin, J. Kownacki, J. Barton, D. Rancey, A. Robinson, and T. Lin, "FIAT - fault injection based automated testing environment," *Digest of Papers, FTCS-19*, pp. 102-107, 1988.

[59] C. L. Seitz, "The cosmic cube," *Commun. ACM*, vol. 28, pp. 22-33, January 1985.

[60] K. G. Shin and J. W. Dolter, "Alternative majority voting methods for real-time computing systems," *IEEE Trans. Reliability*, vol. 38, no. 2, pp. 58-64, April 1989.

[61] D. P. Siewiorek and R. S. Swarz, *The Theory and Practice of Reliable System Design*, Digital Equipment Corporation, Bedford, MA, 1982.

[62] G. F. Sullivan, "A polynomial time algorithm for fault diagnosability," *Proc. Found. Comput. Sci.*, pp. 148-156, 1984.

[63] G. F. Sullivan, "System-level fault diagnosability in probabilistic and weighted models," *Digest of Papers, FTCS-17*, pp. 190-195, 1987.

[64] G. F. Sullivan, "An $O(t^3 + |E|)$ fault identification algorithm for diagnosable systems," *IEEE Trans. Comp.*, vol. 37, no. 4, pp. 388-397, April 1988.

[65] C. L. Yang and G. M. Masson, "A fault identification algorithm for t_i-diagnosable systems," *IEEE Trans. Comp.*,, vol. C-35, no. 6, pp. 503-510, June 1986.

[66] C. L. Yang and G. M. Masson, "A distributed algorithm for fault diagnosis in systems with soft failures," *IEEE Trans. Comput.*, vol. 37, no. 11, pp. 1476-1479, November 1988.