Analysis and Experimental Evaluation of Comparison-Based System-Level Diagnosis for Multiprocessor Systems

Hongying Wang and *Douglas M. Blough**
Dept. of Electrical and Computer Engineering
University of California
Irvine, CA 92717

Leon Alkalaj
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

Abstract

In this paper, a comparison-based model for system-level fault diagnosis that generalizes both the classical PMC model and the Maeng/Malek comparison model is studied. A new necessary and sufficient condition for a system to be t-diagnosable under this model is proven. Also, a class of systems that uses the minimum number of communication links to obtain a given degree of diagnosability is presented. Next, a distributed diagnosis algorithm is presented that can reduce the number of tests necessary for diagnosis when the number of faults is relatively small. To demonstrate the practicality of our diagnosis approach, a fault table based diagnosis algorithm suitable for relatively small systems has been implemented in the COmmon Spaceborne Multicomputer Operating System (COSMOS). A simulator for the JPL MAX multicomputer system running COSMOS was used to test the algorithm and evaluate its performance. The results show that the algorithm diagnoses all fault situations with low latency and very little overhead.

1 Introduction

Distributed or loosely-coupled multiprocessor and multicomputer systems have been developed for many critical applications in military, commercial, and scientific computing. Due to their critical nature, fault tolerance is a requirement in these applications. Multiprocessor systems are well-suited for fault tolerance because of their inherent redundancy. In this paper, we investigate how this redundancy can be used to diagnose faults in multiprocessor systems.

The area of fault diagnosis for multiprocessor systems has been researched extensively since the seminal work on this problem by Preparata, Metze, and Chien [1]. The model proposed in [1] is referred to as the PMC model. Since the introduction of the PMC model, several different testing models have been considered. Among those, comparison-based models, proposed initially by Malek [2] and Hakimi and Chwa [3], have been considered to be the most practical for multiprocessor systems. In these first comparison-based models, it was assumed that system tasks are duplicated on two processors in the system and their results compared by a central observer. This central observer then performs the diagnosis using the outcomes of these comparisons. In [4], Maeng and Malek extended Malek's comparison-based model to allow the comparisons to be carried out by the processors themselves. We refer to the model of [4] as the MM model. In the MM model, the processor performing the comparison must be distinct from the two processors being compared. In [5], Sengupta and Dahbura generalized the MM model to allow the comparator processor to be one of the two processors being compared. It is this model, referred to as the generalized comparison model, that we consider for multiprocessor system fault diagnosis in this paper.

The generalized comparison model corresponds closely to a natural redundancy mechanism utilized, for example, in the dataflow multicomputer operating system COSMOS [6], the Space Shuttle Computer [7], the Software-Implemented Fault Tolerance (SIFT) Computer [8], and the Multicomputer Architecture for Fault Tolerance (MAFT) [9]. For this reason, we believe the generalized comparison model to be extremely practical. As evidence of this, we have implemented and tested a diagnosis algorithm based on the generalized comparison model in COSMOS and we have evaluated the performance of this algorithm using a simulator for the JPL MAX multicomputer system [10]. The results of this evaluation, which are presented in Section 4, show that this approach is capable of diagnosing all fault situations with low latency and very little overhead. To our knowledge, this represents the first experimental validation of system-level diagnosis in a multiprocessor environment.

Before presenting the results of these experiments, we consider some of the fundamental theoretical problems in system-level diagnosis under the generalized

*This research was supported in part by the National Science Foundation under Grant CCR-9010547 and by the California Space Institute under Grant CS-54-92.

0363-8928/94 $3.00 © 1994 IEEE

model. In Section 3, we investigate the interconnection structures and comparison assignments required to diagnose up to t faulty processors in a system. We first present a class of systems which uses the minimum number of communication links possible to achieve this goal. We then provide a necessary and sufficient condition for a system (specified in terms of its communication graph) to be capable of diagnosing up to t faults. This characterization enables the maximum diagnosability of a system to be determined efficiently. We also prove a necessary and sufficient condition for a comparison assignment to allow t-fault diagnosis. This is the classical problem of t-diagnosability. However, our characterization does not yield an efficient algorithm for checking t-diagnosability and so this important problem remains open in the generalized model. Also in Section 3, we present a distributed diagnosis algorithm for use in the generalized comparison model that is efficient with respect to the number of tests required when the number of faults is small. Finally, in Section 4, we present our experimental results.

2 Preliminaries

For the purpose of fault diagnosis, a multiprocessor system is modeled by two graphs, a communication graph and a test (or comparison) graph. A multiprocessor system is a collection of processors, where each processor is connected to other processors in the system via communication links. This can be represented by an undirected communication graph $G(S) = G(V, E)$, where V is a set of vertices and E is a set of edges. Each vertex in the graph represents a processor in the system, and each edge represents a communication link between two processors. Tests (or comparisons) are done via communication links.

In the PMC model, it is assumed that each processor in the system tests a subset of its neighboring processors. The test graph $T(S)$ is a directed graph in which a directed edge (u_i, u_j) represents a test of processor u_j by processor u_i through edge (u_i, u_j) in the communication graph. The test outcome, denoted as a_{ij}, is 0 if u_i passes u_j, and is 1 if u_i fails u_j. Since faulty processors are not reliable, it is assumed in the PMC model that test outcomes produced by faulty processors can be 0 or 1 independent of the status of the tested processor. It is further assumed that a fault-free tester always correctly determines the status of a processor, *i.e.*, it produces a test outcome of 0 if the tested processor is fault-free and an outcome of 1 if the tested processor is faulty.

In the MM model, a system task is duplicated on two different processors u_j and u_k, and their results are sent to a third processor u_i for comparison via edges (u_j, u_i) and (u_k, u_i) in the communication graph. If the two results match, the comparison outcome, denoted as C^i_{jk}, is 0. Otherwise, the comparison outcome is 1. It should be noted that in this model, a processor can not compare itself with other processors. Hence, the three processors involved in a single comparison must be distinct. Similar to the PMC model, it is assumed in this model that comparison outcomes produced by faulty comparators can be 0 or 1 independent of the statuses of the two processors being compared. It is further assumed that a faulty processor always produces incorrect results for each of its given tasks and that it is extremely unlikely that two faulty processors produce the same incorrect results when assigned the same task and inputs. Hence, if the comparator is fault-free, then the comparison outcome will be 0 if both processors being compared are fault-free and the comparison outcome will be 1 if at least one of the compared processors is faulty.

The major difference between the generalized comparison model and the MM model is that a processor is allowed to compare itself with other processors in the generalized comparison model. A task is assigned to two different processors u_j and u_k, and their results are sent to processor u_i for comparison via edges (u_j, u_i) and (u_k, u_i) in the communication graph. A comparator can be any processor in the system that is adjacent to the two compared processors. The comparator can also be one of the processors u_j or u_k. The comparison graph can be represented by a hypergraph $M(S) = G(V, C)$, where V is the set of processors and C is the set of comparison edges. Each element of C can be represented by $(u_i, \{u_j, u_k\})$ where u_i is the comparator, and $\{u_j, u_k\}$ is the processor pair to be compared. The comparison outcome has the same meaning as in the MM model. When the comparator u_i is different from processors u_j and u_k, the situation is exactly the same as in the MM model. Table 1 shows the special case where the comparator is one of the two processors being compared. This table shows the four possible states of processors u_i and u_j, the corresponding test outcome under the PMC model, and the comparison outcome under the generalized comparison model. The table shows that this special case of the generalized comparison model is equivalent to a test in the PMC model.

The following definitions are central to the discussion in the remainder of the paper. A syndrome is a complete collection of test (or comparison) outcomes. A system is t-diagnosable if all faulty processors in the system can be correctly identified, provided that the

u_i	u_j	C_{ij}^i	a_{ij}
fault-free	fault-free	0	0
fault-free	faulty	1	1
faulty	fault-free	x	x
faulty	faulty	x	x

Table 1: Two-processor test outcomes for the generalized model (C_{ij}^i) and the PMC model (a_{ij})

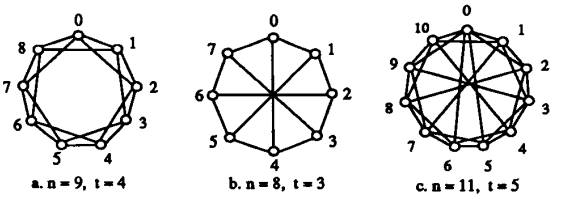

Figure 1: Optimal connection structure $S_{t,n}$

number of faulty processors present does not exceed t. The definition of t-diagnosability can be applied to the PMC model, the MM model, or the generalized comparison model.

3 Analytical Results

There are several interesting questions related to t-diagnosability. One is how to design a system to meet a given diagnosability requirement. Another is given a test (or comparison) graph, what is the maximum value of t for which the system is t-diagnosable. The t-diagnosability problem can also be formulated in terms of a communication graph instead of a comparison graph. In this section, we study the diagnosability problem under the generalized comparison model from these directions. We also compare the PMC model, the MM model, and the generalized model in terms of the number of tests (or comparisons) and the number of communication links required for diagnosis. Finally, we give a distributed diagnosis algorithm under the generalized comparison model.

3.1 Optimal Interconnection

The following defines an optimal generalized t-diagnosable system.

Definition 1 *A generalized t-diagnosable system with n processors is said to be optimal if the system has the minimum number of communication links.*

Hence, optimality is defined in terms of the number of communication links required in a t-diagnosable system. In general, many optimal designs may exist. Here we consider a class of designs in which the communication connection at each processor is nearly identical. These systems are denoted by $S_{t,n}$ [11]. Depending on the parities of t and n, there are three cases:

Case 1: t even. Let $t = 2r$. Then $S_{2r,n}$ has vertices $0, 1, \ldots, n-1$ and two vertices u_i and u_j are joined if $i - r \leq j \leq i + r$ (where addition is taken modulo n). For example, $S_{4,9}$ is shown in Figure 1a.

Case 2: t odd, n even. Let $t = 2r + 1$. Then $S_{2r+1,n}$ is constructed by first drawing $S_{2r,n}$ and then adding edges joining vertex u_i to vertex $u_{i+(n/2)}$ for $1 \leq i \leq n/2$. For example, $S_{3,8}$ is shown in Figure 1b.

Case 3: t odd, n odd. Let $t = 2r + 1$. Then $S_{2r+1,n}$ is constructed by first drawing $S_{2r,n}$ and then adding edges joining vertex u_0 to vertices $u_{(n-1)/2}$ and $u_{(n+1)/2}$ and vertex u_i to vertex $u_{i+(n+1)/2}$ for $1 \leq i < (n-1)/2$. For example, $S_{5,11}$ is shown in Figure 1c.

The following theorem, showing that the $S_{t,n}$ systems are optimal, was proven in [12].

Theorem 1 [Wang/Blough/Alkalaj, 1993]
A system $S_{t,n}$, with $n \geq 2t + 1$, is an optimal generalized t-diagnosable system.

3.2 Maximum Diagnosability

An interesting problem is to determine the maximum diagnosability of a system with a given communication graph. When all the possible tests (or comparisons) are performed, the system has the maximum diagnosability.

The following definition is used in Theorem 2 which characterizes the maximum diagnosability of a system in the generalized comparison model. It should be noted that the proof of this theorem is similar to Sullivan's diagnosability proof in [14].

Definition 2 *Let $G(S) = G(V, E)$ be an undirected communication graph with $Z \subseteq V$. Then, $N(Z) = \{v \in V | \exists z \in Z : (v, z) \in E\} - Z$, i.e., the set of vertices in $V - Z$ which share an edge with at least one vertex in Z. $N(Z)$ is referred to as the neighbor set of Z.*

Theorem 2 [1] *A system S with an undirected communication graph $G(S)$ in which all possible comparisons*

[1] For the proofs of this theorem and Theorem 4, the reader is referred to [13].

are performed is generalized t-diagnosable iff for all $Z \subseteq V$ *with* $Z \neq \emptyset$, $|N(Z)| + |Z|/2 > t$.

Theorem 3 gives a characterization of t-diagnosability in the PMC model that is due to Sullivan [14] and is based on an earlier characterization by Allan, Kameda, and Toida [16]. This result certainly applies to a test graph obtained from a communication graph in which every possible test is performed. The characterization uses the following definition.

Definition 3 *Let* $G(V, E)$ *be a directed graph with* $Z \subseteq V$. *Then,* $\Gamma^{-1}(Z) = \{v \in V | \exists z \in Z$ *with* $(v, z) \in E\} - Z$, i.e., *the set of vertices in* $V - Z$ *which have an edge to at least one vertex in* Z.

Theorem 3 [Sullivan, 1987] *A directed graph* $G(V, E)$ *is t-diagnosable (in the PMC model) iff for all* $Z \subseteq V$ *with* $Z \neq \emptyset$, $|\Gamma^{-1}(Z)| + |Z|/2 > t$.

As stated previously, the maximum diagnosability of a system is achieved when all possible tests are conducted. In this situation, $\Gamma^{-1}(Z)$ for a set Z in the testing graph is equal to $N(Z)$ for the same set Z in the communication graph. Hence, the condition of Theorem 3 is identical to the condition of Theorem 2 in this case. This implies that the maximum diagnosabilities of a system under the PMC model and the generalized comparison model are the same. This has several interesting consequences. First, it implies that, if all possible 2-way comparisons are done in a system, then the addition of any or all 3-way comparisons does not increase the system's diagnosability. So, from a diagnosis perspective, the 3-way comparisons do not provide any additional information that is not already provided by 2-way comparisons. However, we will see in later sections that the use of 3-way comparisons allows a given level of diagnosability to be achieved more efficiently, *i.e.*, using fewer comparisons. The second consequence is that the maximum diagnosability of a system under the generalized comparison model can be determined efficiently. Since the maximum diagnosabilities are the same under the two models, the maximum diagnosability under the generalized comparison model can be found by running Sullivan's diagnosability algorithm on the PMC testing graph containing all possible tests.

3.3 Classical Diagnosability

It is important to know the diagnosability of a system when all the possible comparisons are carried out. This gives the maximum number of faults that are diagnosable in the system. However, in many practical situations, only a subset of the comparisons are available for diagnosis and the diagnosability of the system must be determined. This is the classical problem of t-diagnosability. Another reason for studying classical t-diagnosability is that it may not be necessary to have all the possible comparisons to achieve the maximum diagnosability of a system. An example, as shown in the previous section, is when all 2-way comparisons are performed, then all 3-way comparisons are redundant because they do not improve the diagnosability of the system at all.

In [5], Sengupta and Dahbura proved the first necessary and sufficient condition for a system to be t-diagnosable under the generalized comparison model. Their result is based on a generalization of the Hakimi and Amin condition [15]. To date, the complexity of checking the t-diagnosability of a system (specified in terms of a comparison assignment) is unknown for the generalized model. Since the Allan, Kameda, and Toida characterization [16] was used by Sullivan to construct an efficient diagnosability algorithm [14], we are interested in studying similar characterizations in the generalized model. Accordingly, Theorem 4 provides a new characterization for t-diagnosability in the generalized model that is based on a generalization of the Allan, Kameda, and Toida condition. Before stating this theorem, we give the following definition.

Definition 4 *Let* $M(S) = G(V, C)$ *be a directed hypergraph with* $Z \subseteq V$, *where* V *is the vertex set and* C *is the comparison edge set. Then,*

1. $N_1(Z) = \{v \in V - Z | \exists z \in Z$ *with* $(v, \{v, z\}) \in C\}$, i.e., *the set of processors in* $V - Z$ *which compare themselves with at least one processor in* Z.

2. *Suppose* (Z_1, Z_2) *is a partition of* Z. *Then,*

   (a) $N_2(Z_1) = \{u \in V - Z - N_1(Z) | \exists v, w \in Z_1$ *with* $(u, \{v, w\}) \in C\}$, i.e., *the set of processors in* $V - Z - N_1(Z)$ *that compare two processors in* Z_1. *Similarly,* $N_2(Z_2) = \{u \in V - Z - N_1(Z) | \exists v, w \in Z_2$ *with* $(u, \{v, w\}) \in C\}$, i.e., *the set of processors in* $V - Z - N_1(Z)$ *that compare two processors in* Z_2. *Let* $N_2(Z) = N_2(Z_1) \cup N_2(Z_2)$.

   (b) $G_3(Z) = (N_3(Z), E_3(Z))$ *where* $N_3(Z) = \{u \in V - Z - N_1(Z) - N_2(Z) | \exists v \in Z$ *and* $w \in V - Z - N_1(Z) - N_2(Z)$ *with* $(u, \{v, w\}) \in C$ *or* $(w, \{u, v\}) \in C\}$, i.e., *the set of processors in* $V - Z - N_1(Z) - N_2(Z)$ *which compare a processor in* Z *to a processor in their own set, or which are compared to a processor in* Z *by a processor in their own set,*

and $E_3(Z) = \{(u,v)\} \in N_3(Z) | \exists w \in Z$ *with* $(u, \{v,w\}) \in C\}$.

Theorem 4 *A system* $M(S) = G(V,C)$ *is generalized t-diagnosable iff for all* $Z \subseteq V$ *with* $Z \neq \emptyset$, *and for all* Z_1, Z_2 *that partition* Z, $|N_1(Z)| + |N_2(Z)| + CMVCS(G_3(Z)) + max(|Z_1|, |Z_2|) > t$, *where* $CMVCS(G_3(Z))$ *denotes the cardinality of a minimum vertex cover set of* $G_3(Z)$.

It is clear that the major difference between Theorem 4 and the Allan, Kameda, and Toida theorem is that Theorem 4 depends on the partition of Z. Despite this difference, there are some similarities. Whether an efficient diagnosability algorithm exists for the generalized comparison model based on Theorem 4 is still an open question.

3.4 Comparison of the Three Models

In Section 2, we showed that the generalized comparison model generalizes both the PMC model and the MM model. It is therefore interesting to compare the performances of the three different testing models.

Consider, as an example, a 2-diagnosable system. It was proven in [12] that a system must have at least 5 processors to be 2-diagnosable in the generalized comparison model. The same result was proven previously in both the PMC model and the MM model. The best performances of the three models are shown in Table 2 in terms of the number of links and the number of tests (or comparisons) required in a 2-diagnosable system.

We can see from Table 2 that the generalized model uses fewer links with the same number of comparisons as the MM model. In addition, fewer comparisons are used in the generalized comparison model than tests in the PMC model. For example, for a system with five nodes, the best the MM model can do is using a completely connected graph with 10 links and 15 comparisons. The generalized comparison model can diagnose up to two faults with only 5 links and 10 comparisons. For a system with six processors, the PMC model needs 6 links and 12 tests. The generalized model can accomplish diagnosis using 6 links and only 10 comparisons. Meanwhile, the MM model has to have 10 links and 10 comparisons to reach the same diagnostic result. Each comparison in the MM model needs two links because the three processors involved in a comparison are distinct. In the generalized model, one link is used when a comparator compares itself with another processor, thus saving links. In the PMC model, only one test can be performed on each directed edge, while in the generalized model,

No. of	PMC		MM		Generalized	
proc.	links	tests	links	comparisons	links	comparisons
5	5	10	10	15	5	10
			15	9	7	9
6	6	12	10	10	6	10
			11	9	10	9
7	7	14	9	10	7	10
8	8	16	13	10	8	10
9	9	18	9	9	9	9

Table 2: Best performances of three models with 2 faults

two processors can be compared through two different links at one comparison, resulting in fewer comparisons. Hence, the flexibility of the model allows minimization of either the number of comparisons or the number of links in the system for a small number of faults. This makes efficient diagnosis possible for these systems.

In [1], it was proven that a t-diagnosable system needs at least $\lceil nt/2 \rceil$ links in the PMC model. In [12], it is shown that the same is true in a generalized t-diagnosable system. Theorem 1 shows that this lower bound is achievable in the generalized comparison model. This, combined with Theorems 2 and 3, implies that it is achievable under the PMC model as well. Therefore, both the PMC model and the generalized model can meet a given diagnosability requirement using the minimum number of communication links. This is illustrated by Table 2 for the cases shown. The table also indicates that the minimum number of communication links can not always be achieved under the MM model. With regard to the number of tests or comparisons used when n is much larger than t, the number of comparisons in the MM model and the generalized model is approximately half of the number of tests in the PMC model. This is the case in Table 2 when $n = 9$. When t is larger, the reduction in number of comparisons when going from the PMC model to the generalized model is smaller.

3.5 Distributed Diagnosis

In the classical system-level diagnosis research, it is assumed that a central observer exists. This central observer receives all the test outcomes in the PMC model, or the comparison outcomes in the generalized comparison model, and performs diagnosis based upon these outcomes. In [17], a distributed diagnosis model was proposed by Kuhl and Reddy. They gave the following definition.

Definition 5 *A system is said to be distributed t-diagnosable if each fault-free processor can correctly identify the faulty or fault-free status of every other processor in the system in a distributed fashion.*

In [12], we presented a distributed fault diagnosis algorithm Distributed_Generalized_Diag, which is an adaptation of the Kuhl/Reddy algorithm for the generalized comparison model. Whereas, in the Kuhl/Reddy algorithm, each processor must test each of its neighbors, our algorithm has each processor compare pairs of neighboring nodes. If the comparison outcome for a pair is 0, then both processors are verified to be fault-free with one comparison instead of two tests. If the comparison outcome is 1, then the processor compares itself with one or both of the processors in the pair to determine their statuses. In this manner, far fewer comparisons can be done when the number of faults is small. The following theorem, proved in [12], shows that Algorithm Distributed_Generalized_Diag correctly diagnoses all systems that are distributed t-diagnosable.

Theorem 5 [Wang/Blough/Alkalaj, 1993]
In the generalized comparison model, a system S with comparison graph C employing Algorithm Distributed_Generalized_Diag, is distributed t-diagnosable if $\kappa(C) \geq t$, where $\kappa(C)$ is the connectivity of C.

If $0 \leq \lceil tt'/n \rceil \leq \lfloor t/2 \rfloor$, where t' is the actual number of faults, then the following is an upper bound on the total number of comparisons used by Algorithm Distributed_Generalized_Diag [13].

$$n\lceil \frac{t}{2} \rceil + 2(n-t')\lceil \frac{tt'}{n} \rceil$$

We can calculate numerical upper bounds on the number of comparisons required from the above formula. Table 3 shows the upper bound on the number of comparisons using Algorithm Distributed_Generalized_Diag in the generalized model and the number of tests used in the Kuhl/Reddy distributed algorithm in the PMC model for $S_{t,n}$ systems. For example, in a system $S_{7,15}$, regardless of how many faults occur, the PMC model needs at least $nt = 105$ tests. In the generalized model, when $t' = 4$, the upper bound is 104; when $t' = 2$, the upper bound is 86. In the situation where no fault occurs, the number of comparisons is at most 60, which is slightly more than half of the number of tests required by the PMC model. In most practical situations, faults are rare and the distributed generalized diagnosis algorithm will be quite efficient.

It should be noted that adaptive distributed diagnosis which has been studied by Bagchi and Hakimi [18] and Bianchini and Buskens [19], greatly improves the performance of distributed diagnosis using the PMC model in terms of the number of tests and message overhead. We are currently studying application of the 3-way comparison approach to distributed adaptive diagnosis. While it is not clear whether the number of tests can be reduced further, we expect there will be other benefits to this approach such as reduced diagnostic latency.

n	t	Kuhl/Reddy Tests (any $t' \leq t$)	Comparisons in generalized model				
			$t' = 4$	$t' = 3$	$t' = 2$	$t' = 1$	$t' = 0$
15	7	105	104	104	86	86	60
13	6	78	75	75	61	61	39
11	5	55	61	61	51	51	33

Table 3: Comparison between distributed diagnosis in the generalized model and the Kuhl/Reddy algorithm on $S_{t,n}$ systems.

4 Implementation and Experimentation

Despite a large number of theoretical research results achieved for system-level diagnosis, practical application and implementation of those results remain few. In [19], Bianchini demonstrated the practical implementation of on-line distributed system-level diagnosis theory in a local area network. Prior to the work described in this section, no practical implementation of system-level diagnosis had been undertaken in a multiprocessor system. In the following, we describe the results of experiments done on our comparison-based diagnosis approach using a simulator for the JPL MAX multicomputer [10].

4.1 Overview of the Simulation Environment

MAX is a fault-tolerant multicomputer system developed at JPL for real time control applications. Figure 2 shows a MAX system consisting of several interconnected modules. Each module is an independent computer that communicates with other modules via two networks, the globalbus and the meshwork. To maintain consistency with the remainder of the paper, we henceforth refer to MAX modules as processors. The globalbus is a serial access broadcast bus that is used to control application execution. Only one processor may obtain access to the bus at a time. The globalbus

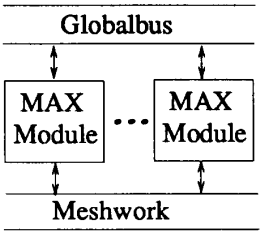

Figure 2: MAX configuration

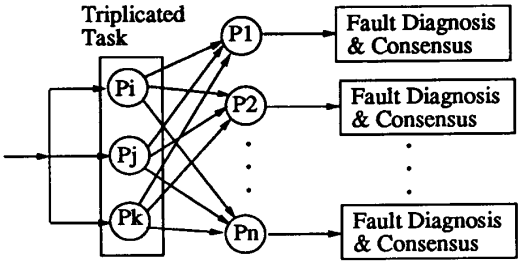

Figure 3: Fault diagnosis and consensus tasks in COSMOS

implements a reliable broadcast feature. The meshwork is a high speed configurable direct network used to transfer data between processors. Each processor can circuit switch among its neighbors so that data received on one meshwork port can be transmitted on another port without CPU intervention. Our testbed is a simulator for the MAX multicomputer system. The simulator simulates the MAX hardware, running the COmmon Spaceborne Multicomputer Operating System (COSMOS).

Our fault diagnosis procedure has been implemented at the operating system level, *i.e.* in COSMOS. COSMOS [6] is an operating system intended to serve the computing needs of a large variety of NASA flight missions. COSMOS uses a coarse-grain dataflow paradigm, with each dataflow node representing a task in the system. Tasks are the basic unit of concurrency in COSMOS. Multiple tasks can be scheduled for execution in parallel. A graph of inter-task dependencies is referred to as a dataflow task graph. The execution of a task graph is based on the data driven principle, *i.e.* a task is enabled for execution when all its necessary inputs become available. In addition, tasks are stateless as dictated by the dataflow paradigm. COSMOS provides several degrees of fault tolerance. Triple modular redundancy, double modular redundancy with a shadow processor, or double modular redundancy may be specified for critical tasks at the system level. A user can specify which tasks should be executed redundantly and to what degree (duplication or triplication). Output token checksums from redundant tasks are broadcast and COSMOS performs comparison or 3-way voting on the checksums in all processors. Double modular redundancy offers only fault detection, which is sufficient for tasks that can be rolled back and retried. Since triple modular redundancy masks one fault, such redundancy is used when tasks must proceed without interruption. Double modular redundancy with a shadow processor allows micro-rollback after fault detection. COSMOS is targeted for loosely-coupled multicomputer systems with reliable broadcast. MAX is an example of such a system.

Our additions to COSMOS are a diagnosis task and a fault consensus task that are executed on every processor. Figure 3 shows the triple modular redundancy mechanism offered in COSMOS and its interaction with the fault diagnosis and consensus tasks. As shown in Figure 3, checksums from the redundant tasks are broadcast on the globalbus to all processors for voting. The comparisons done by a processor on the checksums from a set of redundant tasks form the syndrome used by our diagnosis procedure, which is described in the next section. Note that, because COSMOS already broadcasts redundant task checksums to all processors, no extra messages are generated for diagnosis purposes.

After producing a new diagnosis result, a processor tries to reach consensus on this result with the other fault-free processors. If fault consensus is achieved, then system reconfiguration proceeds. In this situation, the faulty processors are logically isolated. If the fault-free processors can not reach consensus, no reconfiguration is done. The system operates as before until further diagnosis results are available and all fault-free processors agree. The diagnosis and consensus tasks are handled in the operating system and therefore are transparent to the applications programmer. This relieves the programmer from the responsibility for system-level fault diagnosis and reconfiguration.

4.2 Fault Diagnosis and Consensus Tasks

MAX and COSMOS are designed for space applications where the number of processors is relatively small. Typically, only 1 or 2 faults must be tolerated in these applications. Therefore, a table look-up technique can be used to perform diagnosis. In

Fault set	Comparison results		
	C^1_{12}	C^1_{10}	C^1_{20}
$\emptyset$	0	0	0
{0}	0	1	1
{2}	1	0	1

Table 4: The comparison table for u_1 in a system with 3 processors

Time Stamp	Comparison results		
	C^1_{12}	C^1_{10}	C^1_{20}
10	x	x	1
15	x	1	1

Table 5: The syndrome for u_1 in a system with 3 processors

our approach, each processor stores a comparison table and performs diagnosis under the assumption that it is fault-free. The table stored in each processor is an $m \times q$ matrix, where m is the number of possible fault sets of size no greater than t in which the processor itself is fault-free, and q is the number of possible comparisons performed by the processor. For example, Table 4 shows the comparison table stored in processor u_1 in a system with 3 processors u_0, u_1, and u_2. Each processor also maintains a syndrome, which is an array of length q, each entry of which is initialized to the uncertain state, denoted by x.

The syndrome is updated whenever a new comparison outcome is available. Table 5 shows the syndrome stored in processor u_1 in the same 3-processor system. Whenever the syndrome is changed, the diagnostic task of each fault-free processor tries to match the syndrome to a unique row in its comparison table. If a unique row is matched, the fault set corresponding to that row becomes the diagnosis result of the processor. If no fault set can be uniquely identified, *i.e.*, the available syndrome matches several fault sets, then the diagnostic task waits for further comparison outcomes before producing a diagnosis result.

As an example, suppose in a 3-processor system, a task is duplicated on processors u_0 and u_2, and u_0 is faulty. Processor u_2 can diagnose u_0 as faulty after comparing the checksum results of this task. However, processor u_1 receives the two checksums and has the comparison result $C^1_{02} = 1$. At this point, u_1 has the syndrome shown in Table 5 at time-stamp 10. As can be seen from Table 4, u_1 can not produce a diagnosis result because the syndrome can be matched to two different fault sets, {0} and {2}. If another task is then duplicated on processors u_0 and u_1, then the syndrome is updated to the one shown in Table 5 at time-stamp 15. Since this syndrome uniquely matches the fault set {0} in Table 4, processor u_1 diagnoses that u_0 is faulty.

In order to maintain a consistent application state in COSMOS, the fault-free processors must agree on the faulty processors in the system and simultaneously remove them from the active processor list. This is done with a fault consensus procedure that makes use of the reliable broadcast feature in MAX. Fault consensus is a system task running on each processor. The fault diagnosis task broadcasts its diagnosis result whenever it produces a new result. Fault consensus is activated when a diagnosis result is received. The consensus procedure is as follows: Let V be the set of all activated processors in the system, and let M and $\overline{M}$ be subsets of V with $\overline{M} = V - M$. If all processors in M diagnose that all processors in $\overline{M}$ are faulty and $|M| > |V|/2$, then all processors in $\overline{M}$ are faulty and consensus is reached. Note that consensus among a simple majority of processors is sufficient because the MAX reliable broadcast protocol prevents "two-faced" behavior of faulty processors. In addition, consensus will be reached on all fault-free processors at the same time because the reliable broadcast protocol ensures that messages are received in the same order by all processors. If consensus is reached among fault-free processors, then system reconfiguration is done to prevent scheduling further tasks on faulty processors. Otherwise, the diagnosis information is not complete and the fault consensus task waits for further diagnosis results.

A comment is in order concerning the scheduling of redundant tasks in COSMOS. In COSMOS, any processor with available resources can execute any task that is ready to fire. This leads to randomness in the scheduling of redundant tasks. This randomness ensures that, in time, a sufficient set of comparisons will be done to allow diagnosis to take place. Diagnostic latency could be reduced by a more precise scheduling scheme. However, this scheduling scheme would be much more complex and would constitute a fundamental change to COSMOS. Furthermore, our experimental results show that diagnostic latency is low even with the random scheduling scheme.

4.3 Experimental Results

Our experimental results use the diagnosis and consensus procedures and simulation environment described

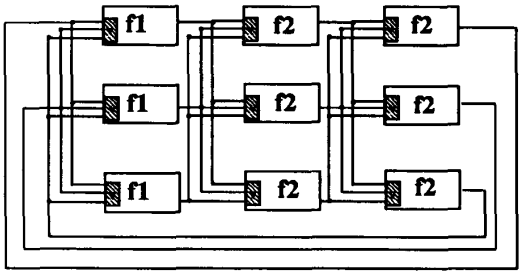

Figure 4: Dataflow task graph for a test application

in the previous sections. In order to test our approach, we must also run an application. Figure 4 shows the application used for this purpose. In this application, all tasks in the first stage execute function $f1$ while all tasks in the second and third stages execute function $f2$. Function $f1$ receives an input token containing a value and its square, increments the value by 2, and calculates its square. Both the updated value and its square form the output token. Function $f2$ increments its received value by 1, and outputs both the increased value and its square. Voting is done on the input values to each task and is handled internally in COSMOS.

A simulation consists in placing initial tokens on the inputs to the first stage tasks and then simulating the execution of the task graph on MAX/COSMOS. During the simulation, fault injection is done by setting a processor to be logically faulty. If a processor is faulty, it always outputs a result different from a fault-free processor given the same task and inputs. Fault injections are done according to a Poisson process for each processor. After a fault situation is diagnosed and consensus is reached, all processors are reset to the fault-free state. This allows the simulation to proceed indefinitely with faults occurring periodically and being diagnosed. The application runs in an error-free state during the entire simulation despite the presence of faulty processors.

Figure 5 shows the diagnostic overhead resulting from simulations of a 6 processor system which allows diagnosis of up to 2 faults. This figure shows the percentage of globalbus messages that are generated by our diagnosis and consensus tasks averaged over five runs with 10,000 simulation cycles in each run. Note that our tasks do not use the meshwork at all so, aside from the minimal CPU time used in their executions, this is the only overhead incurred by our approach. The overhead exhibits some transient behavior but quickly approaches a steady-state value. The steady-state overhead is about 0.3% when the MTTF is 2.7×10^{-6} hour. The steady-state overhead when the MTTF is 5.5×10^{-6} hour is even lower, being only about 0.1%. It should be noted that these MTTF values are much lower than would be expected in real applications. Higher MTTF values would result in even lower overheads for our approach.

The failure rates were chosen to yield a relatively large number of faults during a simulation of reasonable length. The failure rates are also high enough to produce some occurrences of two faults. All fault simulations involved either one or two faults and were correctly diagnosed by our procedure. In the case where two faults occur, the application can not proceed since triple-modular redundancy can mask only a single fault. This situation is detected at the time of voting and is tolerated by rolling back the application. This is done using the checkpoint and rollback feature of COSMOS.

Another quantity of interest is the diagnostic latency. We measured both the average detection latency, *i.e.*, the time between fault occurrence and detection, and the average diagnostic latency, *i.e.*, the time between fault detection and diagnosis consensus. For an MTTF of 2.7×10^{-6} hour, the average detection latency was 0.9 ms and the average diagnostic latency was 2.7 ms. For an MTTF of 5.5×10^{-6} hour, the average detection latency was 1 ms and the average diagnostic latency was 1.2 ms.

In summary, the simulation results show that our diagnosis procedure correctly diagnoses all fault situations. The overhead of the procedure is quite low despite the use of high failure rates to reduce simulation length. In addition, the diagnostic latency of our approach is also quite low.

5 Conclusion

We have considered a generalized comparison model for multiprocessor system fault diagnosis. It combines the advantages of the PMC model by allowing a processor to compare other processors with itself, and of the MM model by performing comparison of a processor pair on a system task. A new necessary and sufficient condition for t-diagnosability in this generalized model was proven. However, both the problems of t-diagnosis and t-diagnosability remain open in this model. We also gave a distributed diagnosis algorithm which reduces the overhead of testing when small numbers of processors are faulty. Practical implementation of system-level diagnosis and consensus using the generalized comparison model was demonstrated in COSMOS. The results have shown that diagnosis can be accomplished with low latency and very

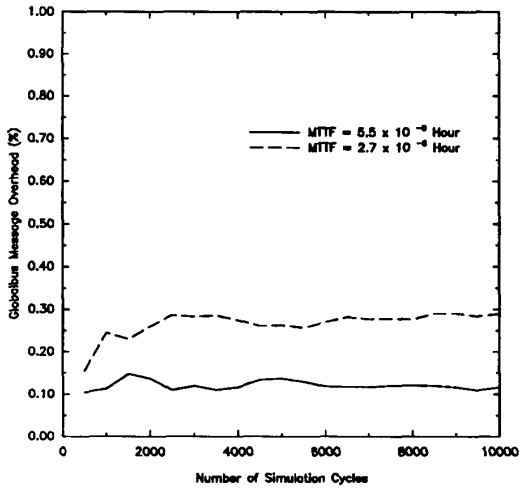

Figure 5: Message overhead of diagnosis and consensus in COSMOS

little overhead.

References

[1] F.P. Preparata, G. Metze and R.T. Chien, "On The Connection Assignment Problem of Diagnosable Systems", *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 848-854, Dec. 1967.

[2] M. Malek, "A Comparison Connection Assignment for Diagnosis of Multiprocessor Systems", *Proc. of the 7th Int. Symp. on Comput. Arch.*, pp. 31-36, 1980.

[3] K.Y. Chwa and S.L. Hakimi, "Schemes for Fault-Tolerant Computing: A Comparison of Modularly Redundant and t-Diagnosable Systems", *Info. and Control*, vol. 49, pp. 212-238, 1981.

[4] J. Maeng and M. Malek, "A Comparison Connection Assignment for Self-Diagnosis of Multiprocessor Systems", *Proc. of the 11th Int. Symp. Fault Tolerant Comput.*, pp. 173-175, 1981.

[5] A. Sengupta and A.T. Dahbura, "On Self-Diagnosable Multiprocessor Systems: Diagnosis by the Comparison Approach", *IEEE Trans. on Comput.*, vol. 41, pp. 1386-1396, Nov. 1992.

[6] B.F. Lewis, *et al.*, "COSMOS Multicomputer Operating System and Development Environment Functional Specification", *NASA Technical Memorandum*, Caltech, JPL, Aug. 1992.

[7] J. Sklaroff, "Redundancy Management Technique for Space Shuttle Computers," *IBM Journal of Research and Development*, vol. 20, pp. 20-28, Jan. 1976.

[8] J. Wensley, *et al.*, "SIFT: Design and Analysis of a Fault-Tolerant Computer for Aircraft Control," *Proc. of the IEEE*, vol. 66, pp. 1240-1255, Oct. 1978.

[9] R.M. Kieckhafer, *et al.*, "The MAFT Architecture for Distributed Fault Tolerance," *IEEE Trans. Comput.*, vol. 37, pp. 398-405, April 1988.

[10] B.F. Lewis and R.L. Bunker, "MAX: An Advanced Parallel Computer for Space Applications", *Proc. of the 2nd Int. Symp. on Space Info. Systems*, Sep. 1990.

[11] J.A. Bondy and U.S.R. Murty, *Graph Theory with Applications*, New York: Elsevier North Holland, Inc., 1976.

[12] H. Wang, D.M. Blough and L. Alkalaj, "A Comparison Model for System-Level Fault Diagnosis and its Application to the COmmon Spaceborne Multicomputer Operating System ", *Proc. of the 1993 Pacific Rim Int. Symp. on Fault-Tolerant Systems*, pp. 35-40, 1993.

[13] H. Wang, D.M. Blough and L. Alkalaj, "Analysis and Experimental Evaluation of Comparison-Based System-Level Diagnosis for Multiprocessor Systems", *Technical Report* ECE-93-07, ECE Dept. UC, Irvine, Nov., 1993.

[14] G.F. Sullivan, *The Complexity of System Level Fault Diagnosis and Diagnosability*, Ph.D. dissertation, Yale Univ., 1987.

[15] S.L. Hakimi and A.T. Amin, "Characterization of the Connection Assignment of Diagnosable Systems," *IEEE Trans. Comput.*, vol. C-23, pp. 86-88, Jan. 1974.

[16] F.J. Allan, T. Kameda and S. Toida, "An Approach to the Diagnosability Analysis of a System," *IEEE Trans. Comput.*, vol. C-23, pp. 1040-1042, Oct. 1975.

[17] J.G. Kuhl and S.M. Reddy, "Distributed Fault-Tolerance for Large Multiprocessor Systems", *Proc. of the 7th Int. Symp. on Comput. Arch.*, pp. 23-30, 1980.

[18] A. Bagchi and S.L. Hakimi, "An Optimal Algorithm for Distributed System Level Diagnosis," *Digest of the 21st Int. Symp. on Fault-Tolerant Comput.*, pp. 214-221, 1991.

[19] R. Bianchini and R. Buskens "Implementation of On-Line Distributed System-Level Diagnosis Theory," *IEEE Trans. Comput.*, vol. 41, pp. 616-626, 1992.