DISTRIBUTED FAULT-TOLERANCE FOR LARGE MULTIPROCESSOR SYSTEMS+

J. G. Kuhl* and S. M. Reddy**

University of Iowa, Iowa City, Iowa 52242

Abstract

Techniques for dealing with hardware failures in very large networks of distributed processing elements are presented. A concept known as distributed fault-tolerance is introduced. A model of a large multiprocessor system is developed and techniques, based on this model, are given by which each processing element can correctly diagnose failures in all other processing elements in the system. The effect of varying system interconnection structures upon the extent and efficiency of the diagnosis process is discussed, and illustrated with an example of an actual system.

Finally, extensions to the model, which render it more realistic, are given and a modified version of the diagnosis procedure is presented which operates under this model.

I. Introduction

As advancing LSI technology continues to push basic component sizes toward the processor level and beyond, increased interest is being shown in the possibility of constructing computing systems realized as very large networks of distributed processing elements. A number of such systems have been proposed in the literature[1-4], with estimates of the number of processors employed ranging up to 10^5 to 10^6. We will refer to these distributed multiprocessor systems as DMP systems. A typical processing element (PE) for a DMP system might consist of a main processor, a memory, and a communications processor to facilitate communications with other processing elements in the system. A PE might also include various peripheral devices such as disk or magnetic bubble storage. Each PE is connected to a certain number (though probably not nearly all) of the other PE's in the system via communication facilities, which are typically taken to be hard, dedicated links.

An interesting feature of these large DMP networks is that they must, by necessity, be truly distributed and hostless. By this it is meant that no central facility is available to provide control, coordination, or mediation among the processors. The very large numbers of PE's employed in the proposed DMP systems would simply overwhelm any attempt at control by a central host computer. The truly distributed nature of these systems implies that the individual PE's must be able to make localized decisions regarding basic operating system functions such as scheduling, resource allocation, etc., based only on information obtained via normal communcations with other processors in the system. For a detailed example of how a fully distributed operating system might be implemented in a DMP environment, the reader is referred to ref. 1.

It must be expected that as the number of PE's employed in a system increases, the probability of hardware failures occurring within the system will also increase. In fact, for an extremely large multiprocessor system, the probability of failure affecting one or more PE's at a given time might be quite significant. This may be especially true in light of the fact that the large number of PE's employed in a very large system implies that the individual PE's must be relatively low in cost, and hence probably of only average reliability. It might be argued that since individual PE's do not constitute a scarce resource in a large DMP system, the loss of one, or even a few of them, due to failure might easily be tolerated.

There is, however, something of a paradox in this argument. It is true that the loss of a PE due to failure might not, in itself, deal a serious blow to the performance of the system. However, a single faulty PE, allowed to continue operating unchecked, could potentially act in a manner which would seriously jeopardize the entire system, e.g. by passing erroneous data to other, unsuspecting PE's. Thus, although the value of an individual PE to the system might be relatively small, the responsibility imposed on the PE to operate correctly might be very large. This implies the need for some facility to detect failures of PE's and somehow remove or isolate faulty processors from the system. Due to the truly distributed nature of DMP systems it is probably the case that these facilities must also be distributed throughout the system. In fact, it may be reasonable to view the failure handling facilities as constituting a portion of each PE. In this sense, it becomes the responsibility of the individual PE's to somehow detect failures within themselves or other PE's. Several observations can be made based on this fact:

* Department of Computer Science

** Division of Information Engineering

+ The research reported was supported by an IBM pre-doctoral fellowship [JGK] and by Air Force Office of Scientific Research under Grant No. AFOSR-78-3582 [JGK and SMR].

i) All responsibility for detection of failures of a PE and subsequent removal of the failed PE from the system, probably should not be assigned to any single PE. The rationale for this observation is that if such responsibility is localized, then failure of the mechanism implementing the responsibility may render the system unable to protect itself from subsequent failure of the PE being watched over by this mechanism. In other words, although the localized failure detection mechanism has responsibility only for the PE it watches, the entire system is dependent upon its correct operation.

ii) Due to the truly distributed nature of such systems, a PE must reach any conclusions as to the health of other PE's in the system, based solely on analysis of information obtained via normal communication links. A PE can probably directly test only its neighbors (i.e. those PE's to which it is directly connected via a communication link). For information concerning non-neighbors a PE must rely on indirect or second-hand knowledge. Also, any action taken by a PE in response to the detection (or supposed detection) of a failure in another PE must involve only use of these normal links. This limits the types of actions which the system can take in response to a failure of a PE. In fact, any synchronized or voted effort among PE's to physically or logically remove a faulty processing element from the system is probably not possible due to the autonomy of the processors, the lack of tight synchronization, and the limited nature of the communication facilities. Furthermore, any physical facility for removing a faulty processor from the system, even if it were to act only in response to a voted effort among several PE's would constitute the same sort of potential single point failure problem for the system as discussed above.

To avoid these problems a notion of fault-tolerance for DMP systems is proposed, known as distributed fault-tolerance. As its name implies, distributed fault-tolerance is an attempt to allow diagnosis and subsequent isolation of failed processing elements in a manner which does not centralize the failure handling facilities of the system in any sense.

The requirements for fully distributed fault-tolerance will be as follows: each nonfaulty processor in the system must be able to independently achieve correct diagnosis of all failures in the system. Once a processing element has diagnosed a failure of another processor, then it can "discriminate" against the faulty processor simply by refraining from further dealings with that processor. Since all nonfaulty processors are aware of the failure and refuse to interact with the faulty processor, it is effectively isolated from the rest of the system.

In the remainder of this paper we will develop some basic results relating to achieving distributed fault-tolerance in DMP systems. We will not be concerned with specific implementation details such as how a particular distributed operating system for a DMP system might enforce its recovery strategy. Rather, we will focus on the crucial issue in achieving distributed fault-tolerance--namely, the ability of processing elements to correctly diagnose the condition of other processing elements in the system.

In the next section, some basic considerations will be discussed, including the meaning of testing and associated architectural considerations, development of a graph-theoretic model for DMP systems, and a discussion of some crucial issues in achieving distributed fault tolerance. The third section will formalize the idea of diagnosis of the system by a processing element. A distributed algorithm for accomplishing the diagnosis will be given. In addition, conditions will be established on the type of interconnection structure required for a DMP system to achieve a given measure of distributed fault-tolerance, and the relationship between the interconnection structure and the efficiency of the diagnostic process will be discussed. Section IV will provide a somewhat more realistic look at distributed fault- tolerance by relaxing some of the assumptions of section III. Section V will present conclusions.

II. Fault Model

The previous section discussed the need for the PE's in a DMP system to possess the ability to detect the presence of failures in other PE's. This essentially implies that a PE be able to "test" other PE's in the system. Testing of one PE by another can be defined as the application of certain inputs to the PE under test, followed by an evaluation, by the testing PE, of the responses to these inputs. Clearly, for a complex processing element, the number of inputs and resulting responses necessary for a thorough test would be very large, if not completely unreasonable. Furthermore such a test might require a level of intimacy between the tester and PE under test, which is not achievable via only the normal communication links.

Fortunately the severity of the testing problem can be lessened by incorporation of fault-tolerance techniques into the architecture of the PE's. First of all, a PE can be made to tolerate and recover from certain failures within itself, by employing well understood design techniques[5-7]. These failures, then, would not even need to be identified as failures at the system level. Furthermore, a PE could be constructed in such a manner as to aid other processing elements in the detection of failures. One way to accomplish this would be to employ self-testing circuit design techniques, together with checkers monitoring various data and control paths to detect failures during normal operation. (Note however that it is not desirable to localize the entire responsibility failure detection, and subsequent processor isolation, within the PE itself, since failure of this facility would then render the entire system vulnerable as discussed in the previous section.) A simple, dedicated "watchdog" processor within the element could monitor the checker outputs constantly, and store any error indications. Then the act of one PE testing another would reduce to a simple matter of a testing PE interrogating the watchdog in the PE under test for any error indications.

In fact, the watchdog of a PE might even initiate the testing by causing a message to be broadcast to PE's assigned to test that PE, noting the occurance of an exceptional circumstance. In either case the testing PE could periodically perform testing to determine the health of the watchdog in the other PE, but the complexity of the watchdog should be small, so any such testing would probably be fairly simple in nature. Of course, other schemes to aid diagnosis are possible, but the important point is that when we speak of one PE as "testing" another, we are not necessarily implying a complex or time consuming action.

A PE is capable of directly testing only those PE's with which it shares a direct communication link (we will refer to two directly connected PE's as neighbors). In order to learn of the condition of non-neighbors, a PE must rely on indirect methods. Specifically, the processor must depend on the results of tests performed on non-neighbors being forwarded to it through the network. It is this issue of dissemination of testing information through a DMP network and the subsequent use of this information by the various PE's in developing their diagnoses, which will be the central issue in the subsequent discussion.

Before proceeding, however, we seek to formalize the problem by introducing a graph theoretic model of a DMP system. In doing so, we will draw from a body of literature broadly known as system level fault diagnosis.[8-12] System diagnosis, first introduced by Preparata et al.,[8] concerns itself with the diagnosis of failures in digital systems comprised of interconnected units, with each unit possessing the capability to test any other unit. The work assumes the presence of a centralized, failure-proof facility to collect and process test results in order to produce the diagnosis, and hence is not directly applicable to the DMP systems considered here. However, many of the techniques used in developing the theory will be useful.

A DMP system will be modeled by an undirected graph with each node representing a processing element and an edge connecting two nodes representing a communication link directly connecting the corresponding two PE's. Such a graph will be called the interconnection graph of the system. In addition, the assignment of testing duties in the system will be shown by a directed graph in which a directed arc is drawn from a node i to a node j to indicate that processor P_i is assigned to test processor P_j. This graph will be called the testing graph of the system. Obviously, the presence of a directed arc between two nodes of the testing graph implies the presence of an edge connecting those nodes in the interconnection graph. (If each PE is assigned to test all of its neighbors, then the testing graph and the interconnection graph will be identical.)

Tests are assumed to be pass/fail in nature. The outcome of a test performed on P_j by P_i will be denoted by a_i^j. If P_i finds P_j to be fault-free, then we will denote this by $a_i^j = 0$. If P_i determines P_j to be faulty, then $a_i^j = 1$. An unknown or unpredictable test result will be denoted by $a_i^j = x$. It is assumed that all tests performed by nonfaulty PE's are valid. That is to say the test outcome will always accurately reflect the condition of the PE under test. The outcome of tests performed by faulty PE's is unpredictable and hence a faulty PE may determine a fault-free PE to be faulty or a faulty PE to be fault free. (An alternative fault model,[9] assumes that all tests performed upon a faulty PE must fail even if the test is performed by a faulty unit. However a faulty unit still might identify a fault-free unit as faulty. For the results in this paper it does not matter which fault model is assumed.)

We will further assume that a faulty PE may alter or destroy any messages routed through it, including messages carrying diagnostic information. The assumption that a faulty PE may reach incorrect conclusions concerning tests which it performs, and may corrupt information passing through it, is known as the principle of test invalidation,[9] and will be an important consideration in the sequel.

III. Fault-Tolerant Systems--Part 1

In this section we show the conditions under which the individual PE's in a DMP system can perform correct diagnosis of the condition of the other PE's in the system. Throughout the section we will work under the following assumption which will be relaxed in the next section:

Assumption A: It is assumed that testing in the system is conducted in "rounds", with a round of testing consisting of the execution of all tests implied by the testing graph. It is further assumed that from the time a round of testing has begun until the time when all nonfaulty PE's have completed their diagnoses, no failures occur in the system.

The assumption assures that the entire diagnosis procedure analyzes a static "snapshot" of the condition of the system at a fixed point in time. While the assumption is somewhat unrealistic, it allows some simple insight into how the diagnostic procedure must proceed. With this insight, we will later be able to relax the assumption.

As mentioned earlier, a PE can directly ascertain the condition of only those PE's which it tests. For information concerning other PE's, it must rely on indirect data passed to it via a neighbor. This PE can in turn pass the information along to another neighbor to aid that PE in making its diagnosis. Note that since the fault model allows that a faulty PE may reach incorrect conclusions about the tests it performs, and since it may alter or destroy information passed to it, a PE can only trust information received from a neighbor which the PE has personally tested and found to be fault-free. Based on these assumptions the following definition is made.

Definition 1: A DMP system is said to be t-fault self-diagnosable iff each fault-free PE can correctly identify all faulty and fault-free PE's in the system, provided that no more than t PE's are faulty.

Next we present an algorithm by which the PE's of a DMP system can perform diagnosis and prove conditions under which this algorithm provides t-fault self-diagnosability.

Algorithm SELF*: Let S be a DMP system with testing graph T. Let the PE's of S, and the corresponding nodes of T, be denoted $P_0, P_1, \ldots, P_n$. Let TESTED-BY (P_i) denote the set of all PE's tested by P_i (i.e., those PE's reached by an arc from P_i in T) and let TESTERS- OF (P_i) denote the set of all PE's which are assigned to test P_i. Messages for the purpose of exchanging diagnostic information between PE's will be of the form $D_0;D_1$ where D_0 and D_1 are sets of integers in the range [0,n]. Each P_i will compute a fault vector $F_i = f_i^1 f_i^2 \ldots f_i^n$ where $f_i^j = 1$ if P_i concludes that P_j is faulty and $f_i^j = 0$ if P_i determines P_j to be fault-free. Assume that a round of testing has been completed (recall that the result of a test of P_j performed by P_i is denoted $a_i^j \in \{0,1\}$). Then each P_i can compute its fault vector F_i according to the following algorithm.

```
1.  Initialize sets $D_0$, $D_1$ to null sets. Let
    $f_i^i = 0$ and let $f_i^j$ be unspecified for $j \neq i$.
2.  For each $P_r \in$ TESTED_BY $(P_i)$ do
    2.1 Let $f_i^r = a_i^r$
    2.2 If $a_i^r = 0$
        then add r to set $D_0$
        otherwise add r to set $D_1$.
3.  Broadcast message $D_0$; $D_1$ to all
    $P_r \in$ TESTERS_OF $(P_i)$.
4.  For each message, say $D_0'$; $D_1'$, received
    from a neighbor $P_j$, where $P_j \in$ TESTED_BY
    $(P_i)$ and $a_i^j = 0$, until $F_i$ is completed**
    do
    4.1 Reinitialize $D_0$ and $D_1$ to null.
    4.2 for each $k \in D_0' \cup D_1'$ such that $f_i^k$ is
        not yet specified do
        4.2.1 If $k \in D_0'$ then do
              4.2.1.1 add k to $D_0$
              4.2.1.2 let $f_i^k = 0$
        4.2.2 otherwise do
              4.2.2.1 add k to $D_1$
              4.2.2.2 let $f_i^k = 1$.
    4.3 If $D_0 \cup D_1$ is not null
        then broadcast message $D_0$; $D_1$ to
        all processors $P_t$ such that
        $P_t \in$ TESTERS_OF $(P_i)$ - $\{P_j\}$ and
        $a_t^i \neq 1$.
5.  Let any still unspecified positions of
    $F_i$ be 0.
```

*A preliminary version of this algorithm appeared in ref. 12 and a similar algorithm was proposed independently in ref. 11, through not in the same context.

**The precise meaning of "until F_i is completed" will be discussed shortly.

Note that step 4 of the algorithm involves a loop to be repeated "until F_i is completed." When considering t-fault self-diagnosability a fault vector F_i should be considered completed when it is fully specified or when it contains t ones. In the latter case, step 5 of the algorithm assures that any still unspecified entries will be set to 0.

We can establish a measure of the degree of diagnosability provided to a DMP system by the above algorithm, as follows.

Definition 2: The connectivity, k(G), of a directed graph G, is the minimum number of nodes of G whose removal from G results in a graph which is not strongly connected.

Theorem 1: A DMP system S, with testing graph T, and employing algorithm SELF, is t-fault self-diagnosable iff $k(T) \geq t$.

Proof:

Necessity: Note that a PE makes use of only its own test results or information received from a neighbor known to be fault-free. Hence information from a fault-free processor P_j can reach another fault-free processor P_i only by passing through a sequence of fault-free processors, say $P_j = P_{k_0} \rightarrow P_{k_1} \rightarrow \ldots \rightarrow P_{k_r} = P_i$, $r \geq 1$, where P_{k_i} tests $P_{k_{i-1}}$, $0 < i \leq m$, i.e., only if there is a path from P_i to P_j in T-{all faulty nodes}. If $k(t) < t$ then there is some set of fewer than t faults such that removal of the corresponding nodes from T, will leave some remaining node P_i with no path to some other remaining node P_j and hence no results from tests on P_j can reach P_i and f_i^j will remain unspecified.

Sufficiency: Since PE's accept information only from neighbors which they have tested and found to be fault free, a fault-free PE will never use erroneous information, i.e., will never incorrectly diagnose another PE. Hence it remains only to show that $k(T) \geq t$ is sufficient for fault-free PE's to fully specify their fault vectors. Let the number of failures in S be $p \leq t$. For $p < t$, there is clearly a path connecting any two nodes in T-{all faulty PE's} and hence the diagnosis can be completed. For $p = t$, let P_i be any fault-free PE and P_j any faulty PE. Since $k(T) \geq t$ implies at least t disjoint paths from P_i to P_j in T, there must be at least one such path which contains only fault-free PE's (other than P_j itself). Thus P_i will correctly diagnose P_j as faulty. Since there are t faulty PE's, F_i will eventually contain t ones, causing the remainder of F_i to be correctly filled with zeros.

Q.E.D.

In fact, it is easily seen from the proof of Theorem 1 that the condition $k(T) \geq t$ is necessary for t-fault self diagnosability regardless of what algorithm is used to perform the diagnosis.

The following example illustrates the operation of the algorithm.

Example 1: Figure 1a shows a potential interconnection graph G for a 5 processor DMP system (called a D_{12} system[8]) and Figure 1b indicates a possible testing graph T for the system. It is easily verified that $k(T) = 2$ and hence the system is 2-fault self-diagnosable. Suppose P_0 and P_1 have failed. Then the results of a round of testing would show $a_3^0 = a_4^0 = a_4^1 = 1$, a_0^1, a_0^2, a_1^2, and a_1^3 would be unpredictable, and all other test results would be 0. Hence at the completion of step 2 of SELF, the fault-vectors of the fault-free PE's would be: F_2 = uu000, F_3 = 1uu00, F_4 = 11uu0, where u indicates the entry is unspecified. Now P_2 will receive messages {4}; {0} from P_3 and { }; {0,1} from P_4 and will update F_2 to 11000. P_3 will receive message { }; {0,1} from P_4 and update F_3 to 11u00 and then to 11000 while ignoring any message from P_0. Since P_4 has identified two faulty PE's, F_4 will be completed as F_4 = 11000.

It is interesting to note that the level of diagnosability achievable in a DMP system is tied solely to the connectivity of its testing graph. This implies that in designing a DMP system which is to have a given level of self-diagnosability, the designer need only insure that the processor interconnection structure is rich enough to support a testing graph of the required connectivity.

If all PE's in a DMP system are assigned to test all neighbors, then the interconnection and testing graphs of the system will be identical. It is reasonable then to ask why we consider the testing and interconnection graphs as separate entities. One reason for this is the fact that there may be arcs in the interconnection graph which can be eliminated without reducing the connectivity. These arcs essentially represent "wasted" tests which do not add to the diagnosability of the system. Furthermore, the connectivity of the interconnection graph may imply a higher level of diagnosability than is really required. In such a case, constructing a testing graph with lower connectivity will reduce both the testing load placed on the individual PE's, and the number of diagnostic messages which must flow through the system to accomplish the diagnosis.

However, other considerations affect the efficiency with which the diagnostic process can be completed. As noted in the proof of the theorem, a processor P_i must receive diagnostic information about another processor P_j via a series of fault-free PE's which lie on a path from P_i to P_j in the testing graph. Thus the time required for the information to reach P_i, and hence for P_i to reach a diagnostic decision concerning P_j, is proportional to the length of the shortest such path. We can formalize this notion as follows:

Definition 3: The distance from a node j of a directed graph is the length of the shortest path from i to j in the graph.

Definition 4: The t-diagnostic distance from a node i to a node j in a directed graph is the maximum of the distances from i to j in T-Q, where Q ranges over all subsets of t-1 nodes of T-{i,j}.

Definition 5: The t-diagnostic diameter of a directed graph is the maximum of the t-diagnostic distances between any two nodes in the graph.

With these definitions, we can state that the time required for a fault-free PE, in a t-fault self-diagnosable system, to produce a complete diagnosis is proportional to the largest t-diagnostic distance from the PE to any other PE in T. And hence the efficiency of the diagnostic process for the system as a whole is proportional to the t-diagnostic diameter of T. Hence in choosing a testing graph for a particular system, and indeed for choosing the system interconnection structure itself, consideration should be given to keeping the t-diagnostic diameter small. This may imply a trade-off between minimizing the number of tests required and keeping the diameter at a reasonable level.

We end this section with an example of the application of the results of this section to a realstic system.

Example 2: One of the system interconnection structures which has been deemed well suited for very large DMP systems, is the n-cube.[1] An n-cube DMP system consists of 2^n processing elements. Each processor is assigned an "address," which is a unique n bit binary vector whose corresponding decimal value is in the range 0 to $2^n - 1$. A processor is connected, via a communication link, to each processor whose address differs from its own in exactly one bit position. Figure 2 shows an interconnection graph for a 5-cube DMP system. It is well known that the connectivity of an n-cube is n. Hence the maximum level of self-diagnosability achievable is the n-fault level. The n-diagnostic distances in an n-cube can be determined as a consequence of the following theorem.

Theorem 2: In an n-cube, let i and j be any two distinct nodes whose addresses differ in d positions ($1 \le d \le n$). Then there are n distinct paths of length $\le d + 2$ connecting i and j.

Proof: It can be shown constructively that there are d paths of length d and $n - d$ of length $d + 2$. However, the proof is somewhat lengthy and hence is omitted.

Corallary 1: The n-diagnostic diameter of an n-cube is $n + 1$.

The above results indicate that the n-cube interconnection structure has good properties for efficient n-fault self-diagnosis. If a lower level of diagnosability, say $t < n$, is desired, various testing graphs with reduced connectivity can be constructed yielding tradeoffs between number of tests required and the t-diagnostic diameter of the graph. For instance it can be shown constructively that for any $t \le n$ a t-connected subgraph of the n-cube can be found which employs exactly $t2^n$ arcs, the graph theoretic minimum. However, this subgraph tends to result in extremely large t-diagnostic diameters. On the other hand, it can also be shown that a class of t-connected subgraphs of the n-cube exist which require $t2^n + (n - t)2^{n-t}$ arcs which have t-diagnostic diameter of less than or equal to $n(t + 1) - (t^2 - t) + 1$, thus providing fairly good diagnosis efficiency. An example of such a two-connected testing graph for a 5-cube interconnection structure is shown in Figure 3. (Note that the graph is shown as an undirected graph. An edge in this graph should be interpreted as implying a test in each direction.) If one-fault diagnosability is desired, testing subgraphs retaining diameter n can be constructed which require $\le 3n\ 2^{n-2}$ arcs.

IV. Fault-Tolerant Systems--Part II

Assumption A of the previous section is clearly unrealistic. The PE's in a DMP system are not tied to any common clock and hence cannot possibly hope to synchronize a round of testing tightly enough to guarantee a static snap-shot of the system. In fact it is probably unreasonable to assume that testing will be done in rounds at all. Instead, it is likely that each individual PE will perform its testing duties in a more or less independent fashion, according to some locally maintained schedule.

Hence the fault vector cannot realistically be viewed as an entity which is calculated at discrete time intervals by each PE in the system. Instead, the diagnosis procedure must be seen as a constantly evolving event, with fault vectors being updated (not recalculated) as new failures occur and new diagnostic information becomes available.

In this section, the diagnostic algorithm of the previous section will be modified to reflect this more realistic view of the diagnosis procedure. We begin with an example illustrating the difficulties to be encountered.

Example 3: Consider again the system in Figure 1 and assume the following scenario. At time t_0 the system is free of failures. Between t_0 and some time t_1, P_4 tests P_0, P_4 tests P_1, and P_3 tests P_0 with all test outcomes zero. At time t_1, P_0 fails. The faulty P_0 tests P_1 and P_2 and incorrectly determines P_2 to be faulty. Now subsequent to this at some time t_2, P_1 also fails and, after failing, P_1 tests P_2 and P_3 and also incorrectly finds P_2 faulty. All other test results in the system will be zero, regardless of when performed. Now if P_0 and P_1 are operating reasonably correctly following their incorrect diagnosis of P_2, the fault vectors computed by algorithm SELF could be F_0 - 001uu, F_1 = u010u, F_2 = uu000, F_3 = 0uu00, F_4 = 00uu0, with $a_3^0 = a_4^0 = a_4^1 = 0$. If P_0 and P_1 send the diagnostic messages, implied by the algorithm, properly then it is easy to verify that the fault vectors finally computed by the nonfaulty PE's will be F_2 = 00000, F_3 = 00100, F_4 = 00100. So in fact all three nonfaulty PE's have failed to compute proper fault vectors and in fact P_3 and P_4 have actually diagnosed a fault-free PE as faulty. The scenario put forth in the example is admittedly contrived. However if one considers a very large system in which diagnostic information propagating at various places represents the results of tests which occurred at substantially different points in time, the potential for trouble is evident.

To a certain extent, the problem is unresolvable. If it takes some amount of time, say Δt, for test information concerning P_j to reach P_i, then any opinion which P_i might hold about P_j is outdated at least by an amount of time Δt. Hence P_i cannot be aware of a failure of P_j which has occurred less than Δt time units ago. This is a fact which must simply be lived with, and it does not pose a serious problem since the information regarding P_j should reach P_i eventually.

However the possibility that a PE may make a diagnosis which is not just outdated, but actually incorrect, is intolerable. By analyzing the example, it is seen that the incorrect diagnosis occurs when a PE after being tested and found fault free by other PE's suffers a failure and subsequently disseminates invalid diagnostic information to PE's unaware of the failure.

The solution to the problem is to establish a policy of "mutual distrust" among the PE's. This policy can be stated as follows: If a processor P_i is notified by a neighbor P_j that a third processor P_k has failed, P_i should believe this information only after verifying that P_j is still operating correctly. In other words, before accepting any message indicating a new failure in the system, a PE must retest the neighbor from which the message was received. In this way a nonfaulty PE can never erroneously determine another PE to have failed.

This may seem to imply the need for a great deal of extra testing in the system, but in fact no extra testing at all may be required. Instead of immediately broadcasting newly received diagnostic information as in SELF, a PE could keep track of the new information locally. Then the next time a neighbor tested this PE, it could also interrogate the PE for any new diagnostic information.

Of course, if it is desired to have new failure information disseminated as quickly as possible in order to keep fault vectors as up to date as possible, then the need for extra testing may be implied. We will pursue this latter approach by presenting a new diagnostic algorithm which acurrately reflects the nature of the diagnostic process as presented in this section.

Algorithm SELF2: Assume the same notation as in the previous section except that diagnostic message will consist of a single integer "d" in the range 0 through n which indicates that the PE sending the message has determined that P_d is faulty. Further assume that at some initial start-up time, all fault vectors are initialized to the all zero vector. Assume that each processor P_i performs its tests according to some local schedule. Then each time P_i is scheduled to test a neighbor P_j, it should proceed as follows:

```
1.   Perform the test of $P_j$ to obtain $a_i^j$.
2.   If $a_i^j = 1$
     then do
     2.1  Set $f_i^j$ to 1.
     2.2  Broadcast message "j" to all $P_k$ such
          that $P_k \in$ TESTERS_OF $(P_i)$ and $f_i^k = 0$.
```

Each time P_i receives a diagnostic message "d" from neighbor P_j, P_i should perform the following:

```
1.   If $f_i^j = 0$ and $j \in$ TESTED_BY $(P_i)$ and
     $f_i^d = 0$ then do
     1.1  Perform the test of $P_j$ to obtain $a_i^j$.
          1.2.1  If $a_i^j = 0$
                 then do
                 1.2.1.1  Set $f_i^d$ to 1.
                 1.2.1.2  Broadcast message
                          "d" to all $P_k$ such
                          that $P_k \in$ TESTERS_OF
                          $(P_i) - \{P_j\}$ and
                          $f_i^k = 0$.
          1.2.2  Otherwise do
                 1.2.2.1  Set $f_i^j$ to 1.
                 1.2.2.2  Broadcast message "j" to
                          all $p_k \in$ TESTERS_OF $(P_i)$
                          such that $f_i^k = 0$.
```

Space considerations prevent proof of the correctness of this algorithm but its correctness should be fairly evident from the preceeding discussion. It should be reiterated that the testing of a PE does not necessarily constitute a long or tedious process but rather, if sufficient diagnostic capability is incorporated in each PE, might involve an action as simple as examining a few checker outputs. Clearly the viability of the algorithm just presented is greatly influenced by the architecture of the PE's and the ease with which they can detect failures in one another.

The following example will illustrate the operation of the algorithm.

Example 4: Consider one last time, the system in Figure 1. Assume that at time t_0 the system is free of failures. At time t_1, suppose P_0 fails and proceeds to erroneously diagnose P_2 to be faulty. Now P_0 may initiate a message to P_3 and P_4 indicating P_2 has failed. P_3 and P_4 will, upon receiving this message, initiate tests of P_0 and will find P_0 to have failed thus causing f_3^0 and f_4^0 to be set to 1. P_3 will broadcast message "0" to P_1 and P_2 and P_4 will do the same to P_2 and P_3. Now P_2 will perform a test of either P_3 or P_4, depending on which message it receives first, and upon passage of the test will set f_1^0 to 1. P_1 will do the same and set f_1^0 to 1. Hence the attempt of P_0 to declare P_2 faulty has been thwarted and in the process P_0 itself has been diagnosed as faulty. The reader can easily verify that any scenario of dual failures, regardless of their times of occurrence would also be properly dealt with.

V. Conclusions

This paper has considered the problem of achieving fault-tolerance in large, fully distributed multiprocessor systems. It has been argued that each PE in such a system must be able to produce an independent diagnosis of the health of other PE's in the system. Techniques for accomplishing this distributed diagnosis have been presented. The level of diagnosis provided by these procedures, and the efficiency with which this diagnosis can be accomplished, has been related to the processsor interconnection structure of the system as well as to the architecture of the individual processing elements.

References

1. H. Sullivan, T. Bashkov, and D. Klappholz, "A large scale, homogenous, fully distributed parallel machine," in Proc. Fourth Annual Symposium on Computer Architecture, pp. 105-124, March 1977.

2. A. Despain and D. Patterson, "X-tree: A tree structered multiprocessor computer architecture," in Proc. Fifth Annual Symposium on Computer Architecture, pp. 144-151, April 1978.

3. L. Wittie, "MICRONET: A reconfigurable microcomputer network for distributed systems research," Simulation, Vol. 31, pp. 145-153, Nov. 1978.

4. C. Reams and M. Liu, "Design and simulation of the distributed loop computer network," in Proc. Third Annual Symposium on Computer Architecture, pp. 124-129, Jan. 1976.

5. A. Avizienis, "Fault-tolerant systems," IEEE Trans. Comput., Vol. C-25, pp. 1304-1312, Dec. 1976.

6. A. Avizienis et al., "The STAR computer: An investigation of the theory and practice of fault-tolerant computer design," IEEE Trans. Comput., Vol. C-20, pp. 1312-1321, Nov. 1971.

7. D. Siewiorek, M. Canepa, and S. Clark, "C. vmp: The architecture of a fault-tolerant multiprocessor," in Proc. Seventh International Symposium on Fault-Tolerant Computing, June 1977.

8. F. Preparata, G. Metze, and R. Chien, "On the connection assignment problem of diagnosable systems," IEEE Trans. Comput., Vol. EC-16, pp. 848-854, Dec. 1967.

9. F. Barsi, F. Grandoni, and P. Maestrini, "A theory of diagnosability of digital systems," IEEE Trans. Comput., Vol. C-25, pp. 585-593, Une 1976.

10. J. Russel and C. Kime, "System fault diagnosis," IEEE Trans. Comput., Vol. C-24, pp. 1078-1089, Nov. 1975 and pp. 1155-1161, Dec. 1975.

11. G. Meyer and G. Masson, "An efficient fault diagnosis algorithm for symmetric multiple processor architectures," IEEE Trans. Comput., Vol. C-27, pp. 1059-1063, Nov. 1978.

12. J. Kuhl and S. Reddy, "An approach to fault diagnosis and security in digital systems," in Proc. 1978 Conference on Information Sciences and Systems, pp. 269-273, John Hopkins University, March 1978.

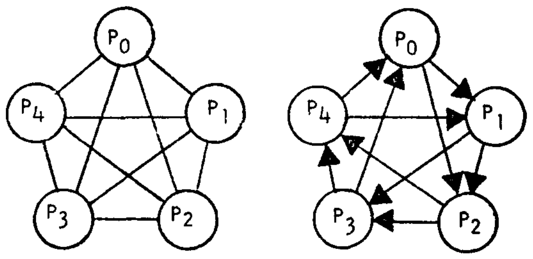

Figure 1: Interconnection and testing graphs for an example DMP system.

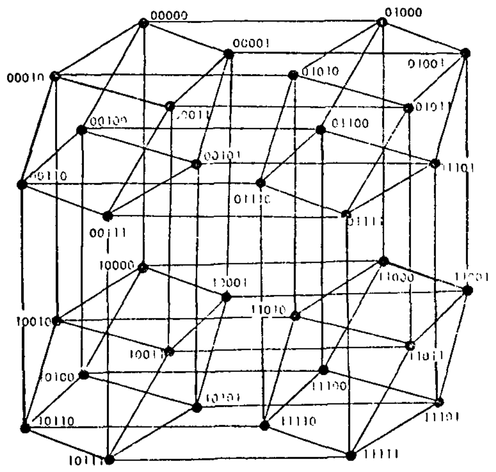
Figure 2. A 5-cube interconnection graph.

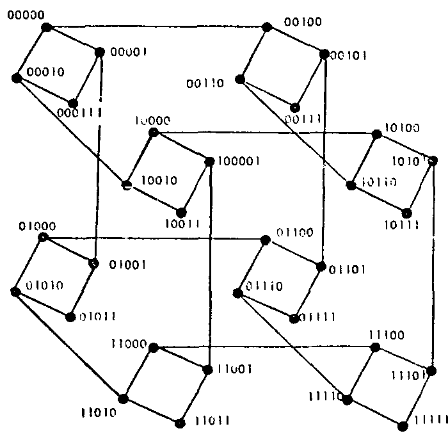

Figure 3: A 2-connected testing graph for system of Figure 2.