# Multi-Cluster Adaptive Distributed System-Level Diagnosis Algorithms
## – IEICE Technical Report FTS 95-73 –

**Elias Procópio Duarte Jr.** **Takashi Nanya**

Graduate School of Information Science, Tokyo Institute of Technology
2–12–1 Ookayama Meguro-ku Tokyo 152, Japan

{elias, nanya}@cs.titech.ac.jp

***Abstract.*** *System-level diagnosis is used to determine the state of a system consisting of $N$ units which may be faulty or fault-free. Each unit tests a subset of all others, and fault-free units perform tests and report test results reliably. In adaptive diagnosis the tests each unit performs are adaptively selected based on previous test results; and in distributed diagnosis the units themselves perform the diagnosis, instead of a centralized observer. In this paper we present new adaptive distributed system-level diagnosis algorithms, that by grouping units in logical clusters improve the diagnosis latency of current algorithms, while still requiring the same order of diagnostic messages. Two algorithms, ADSD with Intersections and Hierarchical ADSD are presented and analyzed. Applications of these algorithms, including network fault management are considered.*

**Keywords:**

System-Level Diagnosis, Distributed Algorithms, Adaptive Diagnosis

## 1. Introduction

Consider a system consisting of $N$ units, which can be faulty or fault-free. The goal of system-level diagnosis is to determine the state of those units. For almost 30 years researchers have worked on this problem, and the first model of diagnosable systems was introduced by Preparata, Metze, and Chien, the *PMC Model* [1]. In the PMC model units are assigned a subset of the other units to test, and fault-free units are able to accurately assess the state of the units they test. The set of all tests makes up a testing graph, i.e., a directed graph in which vertices are the system's units and an edge from vertice $i$ to vertice $j$ corresponds to a test performed by unit $i$ on unit $j$.

The collection of all test results is called the *syndrome* of the system. The problem of diagnosis is to obtain the state of the system from a given syndrome. The PMC model assumes the existence of a *central observer* that, based on the syndrome, can diagnose the state of all the units. For a given testing assignment the diagnosability of a system may be limited by the number of faulty units, and determining this number is called the *diagnosability problem*. Preparata *et al.* showed that a system is $t$-diagnosable if $N \geq 2t + 1$, and each unit is tested by at least $t$ other units. Later, Hakimi and Amin [2] showed that this result holds if no two units test each other.

Early system-level diagnosis algorithms assumed that all the tests had to be decided in advance. The tests were then executed, and from the obtained results, it was

determined which units were faulty. Those algorithms focused on finding properties of the testing graph which would allow the observer to identify the faulty units from the tests corresponding to the testing graph's edges.

An alternative approach, which requires fewer tests, is to assume that each unit is capable of testing any other, and to issue the tests adaptively, i.e. the choice of the next tests depends on the results of previous tests, and not on a fixed pattern. This approach was called *adaptive* [3]. Early adaptive system-level diagnosis results assumed the existence of the previously mentioned central observer. Furthermore, a bound on the number of faulty nodes was imposed for the system to achieve correct diagnosis.

Adaptive system-level diagnosis algorithms proceed in testing rounds, i.e., the period of time in which each unit has executed at least one test successfully. To evaluate adaptive algorithms two measures are normally used: the total number of tests required per testing round and the diagnosis delay, i.e., the number of testing rounds required to determine the state of the units.

Kuhl and Reddy [4, 5], introduced distributed system-level diagnosis, in which fault-free nodes reliably receive test results through their neighbors, and each node independently performs consistent diagnosis. They proposed the SELF distributed system-level diagnosis algorithm, that although fully distributed, was non-adaptive, i.e. each unit had a fixed testing assignment, and the number of faulty units in the system could not exceed $t$. We will use alternatively the word node for unit, and network for system.

In 1984, Hosseini, Kuhl and Reddy, [6] extended the SELF algorithm, introducing the NEW-SELF algorithm, which also has a fixed inter-node test assignment, but is executed on-line, permitting faulty nodes to reenter the network after being repaired. NEW-SELF ensures the accuracy of test-results by restricting the forwarding of testing results to fault-free nodes. For correct diagnosis, NEW-SELF requires that every fault-free node receive all test results from all other fault-free nodes. To reduce the amount of network resources required for diagnosis, the EVENT-SELF algorithm was proposed [7], which uses event-driven techniques to improve both the diagnosis latency and the impact of the algorithm on network performance.

The Adaptive Distributed System-Level Diagnosis algorithm, *Adaptive DSD*, was introduced by Bianchini and Buskens [8, 9]. Adaptive DSD is at the same time distributed and adaptive, each fault-free node uses the minimal number of messages per testing round, i.e., one message, to achieve consistent diagnosis in at most $N$ testing rounds.

In this paper we present a new approach to Adaptive Distributed System-Level Diagnosis, in which nodes are grouped in logical clusters, so that using the same order of the number of messages diagnosis latency can be reduced. The algorithms assume no link faults, a fully-connected network and impose no bounds on the number of faults.

Besides the PMC fault model, there are many other fault models, see for example [12] for a survey of probabilistic diagnosis. Diagnosis of link faults were treated in [10]. Work on general topology networks has received a great deal of attention recently, e.g. [11]. Furthermore, there are many other completely different approaches to fault diagnosis, that may be useful according to one's specific needs [13, 14].

The paper is organized as follows. Next section presents the Adaptive DSD algo-

rithm in sufficient detail to permit its comparison to our approach. Section 3 introduces the cluster approach to diagnosis, and two algorithms: (1) *ADSD with Intersections* (Adaptive Distributed System-level Diagnosis with Intersections), which has diagnosis latency of up to $O(\sqrt{N})$ testing rounds; and (2) *Hi-ADSD*, (Hierarchical Adaptive Distributed System-level Diagnosis), in which each node uses one message testing round to achieve diagnosis in at most $O(log^2 N)$ steps. In section 5 possible applications of these methods, like network fault management, are considered and followed by concluding remarks.

## 2. The Adaptive-DSD Algorithm

Consider a system S consisting of the triple (V(S),E(S),T(S)), where V(S) is the set of nodes or vertices of S; E(S) is the set of edges linking two nodes of S; T(S) is a testing digraph, where an edge $(n_i, n_j)$ means that $n_i$ tests $n_j$. Consider also a fault situation F(S), which is the set of the states of each node in V(S), faulty or fault-free. $S$ is assumed to be fully connected, and each node in $V(S)$ is assigned a unique identifier, from 0 to $N-1$.

A fault-free node is capable of performing tests and reporting the results of those tests reliably. Faulty nodes may distribute erroneous test results. Diagnostic messages containing test results flow between neighboring nodes and reach non-neighboring nodes through intermediate nodes. Each fault-free node achieves independent consistent diagnosis based on the diagnostic messages it receives. There is no bound on the number of faulty-nodes for fault-free nodes to achieve correct diagnosis.

The Adaptive Distributed System-level Fault Diagnosis (*Adaptive-DSD*) algorithm, [8, 9] is executed at each node of the system at predefined *testing intervals*. Each time the algorithm is executed, a fault-free node will test other nodes until it finds another fault-free node. A *testing round* is defined as the period of time in which all nodes of the system have executed Adaptive DSD at least once. After one testing round, V(S) has the format of a ring, as shown in figure 1. It is very important to keep in mind the difference between *testing*, *testing interval* and *testing round* to understand the algorithm.

Whenever a fault-free node detects that a fault has occurred, the logical testing network, i.e. the logical ring, rearranges itself around the fault so it stays connected. The algorithm does not consider link faults. In the example shown in figure 1, nodes 1, 4, and 5 are faulty, and the rest are fault-free. Node 0 tests node 1 and finds it faulty; so it goes on and tests node 2, which is fault-free, it then stops testing. Node 2 then tests node 3, and so on.

Each node $i$ that executes the algorithm has an array called $TESTED-UP_i$, that contains $N$ entries, indexed by the node identifier. The entry $TESTED-UP_i[k]=j$ means that the node $i$ has received diagnostic information from a fault-free node specifying that node $k$ has tested $j$ to be fault-free. This means that each cell contains the identification of the node tested by the node whose identifier is the cell's index. An entry $TESTED-UP_i[j]$ is "arbitrary" if the node $j$ is faulty.

The information contained in the array $TESTED-UP$ of a given node gets transmitted to other nodes. When a node $i$ finds node $j$ to be fault-free, it saves this information in $TESTED-UP_i[i]$. In the next round of testing, this test data of $i$ is taken by its first fault-free predecessor, and so on, until all nodes get the information. In this

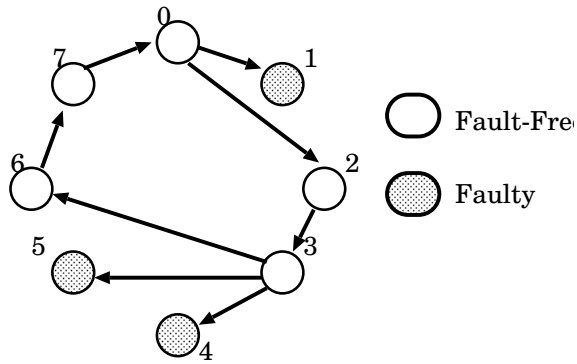


**Figure 1. Example of test assignment in Adaptive DSD.**

way, the diagnostic information in the $TESTED - UP$ array is forwarded to nodes in the reverse direction of the testing network. The total number of tests per testing interval in this algorithm is low, equal to $N$, the number of nodes. The Adaptive-DSD algorithm is given below:

```
Adaptive-DSD
t = i
REPEAT
  t = (t+1) MOD N
  request t to forward TESTED_UPi to i
UNTIL (i tests t as fault-free)
TESTED_UPi[i] = t
FOR j = 1 TO N-1 DO
  IF i != j
  THEN TESTED_UPi[j] = TESTED_UPt[j]
End Adaptive DSD;
```

From the information in $TESTED - UP_i$ a node $i$ has to diagnose the state of the entire system, for this task another algorithm, called *Diagnose* is employed. This algorithm uses the array $STATE_i$ to store the state of the system. The $k^{th}$ element of $STATE_i$ represents the state of node $k$, as determined by node $i$. *Diagnose* first initializes the state of all nodes as faulty. It then sets its own state to fault-free. In the second iteration of the loop it sets the state of the node which it had directly determined to be fault-free. In the next iteration, it sets the state of that node which this one had detected as fault-free. This continues until it reaches the node which had determined that $i$ itself was fault-free. By this time, the loop has been completed and the algorithm terminates. The algorithm can be found in [9].

Since at each testing interval the array $TESTED - UP$ flows backward one step, and the largest possible path in the cycle is of size $N$, after no more than $N$ intervals, all fault-free nodes will set their $i^{th}$ element of $TESTED - UP$ to $j$. It follows that the diagnosis latency of Adaptive DSD is $N$ testing rounds.

A diagnosis latency of $N$ testing rounds may be unacceptable for many real systems. For example, consider $N = 200$, and a testing interval of 30 seconds: it takes around 1:40 hours for all the nodes to diagnose a given fault situation. In the original papers, Bianchini *et al.* [8, 9], suggest the use of event-driven mechanisms to reduce the latency,

like employing multicast or broadcast just after a new situation is identified. As in an specific systems it may be undesirable or even impossible to introduce these extra event-driven mechanisms, in the next section we introduce new adaptive distributed system-level algorithms that reduce the diagnosis latency without extra event-driven mechanisms, while requiring the same order of the number of diagnostic messages.

## 3. Multi-Cluster Diagnosis

In the Adaptive DSD algorithm each node sees all others in a linear list. For example, if all nodes are fault-free, information about node $N-1$ will reach node $0$ after $N$ testing rounds. Using such a linear testing strategy it is impossible to achieve diagnosis in less than $N$ testing rounds. But applying a divide-and-conquer strategy adaptive distributed system level diagnosis can be achieved in less than $N$ testing rounds.

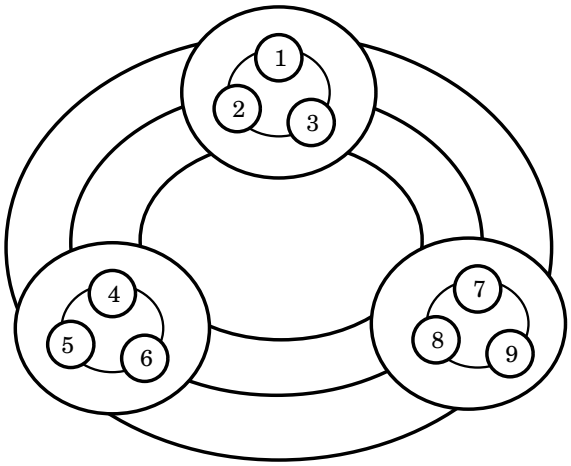


**Figure 2. Organizing nodes in logical clusters**

Figure 2 shows a possible structure for clustering nodes of network. In this figure clusters are made up of three nodes, and we have a total of three clusters. In this way if Adaptive DSD is executed at two levels, diagnosis could be, in principle achieved in 6 testing rounds, instead of 9. Many different algorithms are possible, one just has to change the number of nodes in a cluster or the testing strategy for each node.

Whenever a node tests a cluster it must select which information it wants from that cluster or the specific node being tested. In Adaptive DSD, as all nodes are part of only one big cluster, and each node always tests only one other node, this problem doesn't exist. The basic data structure of a multi-cluster algorithm cannot be a unidimensional array as in Adaptive DSD, and needs to reflect the strategy of each algorithm.

Among the many possible cluster-based diagnosis algorithms two are presented in the next two subsections: the *ADSD with Intersections* and the *Hierarchical ADSD*.

### 3.1. Adaptive-DSD with Intersections

In this section we introduce *ADSD with Intersections*, an algorithm for cluster-based adaptive distributed system-level diagnosis. In ADSD with Intersections, nodes are grouped in logical clusters, and each cluster executes tests forming a ring, like in Adaptive DSD. But now, all clusters have a point of intersection, so that if there are a total of $p$ clusters,

it takes $N/p$ testing rounds to diagnose the state of all $N$ nodes. At each testing interval, all nodes still test one fault-free node, but the node at the intersection tests $p$ nodes, one at each cluster.

The main data structure employed is a two-dimensional $TESTED - UP_i[cluster, j]$ array. And for a fixed $cluster$, $TESTED - UP_i[cluster, j]$ is treated much in the same way as the unidimensional $TESTED - UP_i[j]$ of Adaptive DSD.

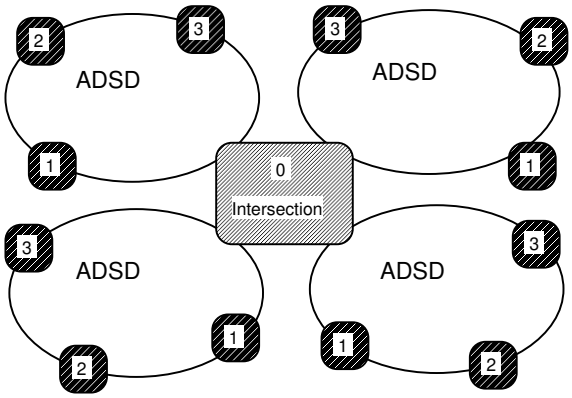


**Figure 3. Multiple simultaneous clusters run Adaptive-DSD with intersection.**

Within a cluster, whenever one node tests another node as fault-free, it gets all its information regarding tests in all clusters. The node at the intersection proceeds in a different way: for each cluster it tests it only gets information about that specific cluster. In this way, at each test it gets only the new testing information of each cluster.

The algorithm provides a mechanism for network nodes to identify the intersection node, and also a replacement if the current one becomes faulty. There is an array that identifies an order of priority of nodes that should become the intersection, this array is called PRIORITY.

All the nodes at the clusters have identifiers that go from 1 to $p$, and the intersection has the logical address 0. Whenever a node tests another node that is not the intersection it simply gets all the information of all clusters.

Whenever a node tests the node at the intersection, i.e., the node with id equal to 0 (zero), it accesses the PRIORITY array, and searches sequentially until it finds a working intersection, which can be the node itself.

In this way if the PRIORITY array is filled such that the original intersection is followed by the nodes that test that intersection and then the nodes that test the nodes that test the intersection and so on, as soon as the intersection becomes faulty, these nodes discover the new intersection. It should be noted that whenever a higher priority node recovers, receiving one message from each cluster is sufficient to obtain the state of the entire network. The algorithm is as follows:

```
ADSDwI(cluster, next);
 /* in node of address my_cluster, my_id  */
 /* initially called as
      ADSDwI(my_cluster, (my_id+1) MOD p) */

REPEAT
 IF next = 0
 THEN /* test intersection */
  REPEAT
   cluster := PRIORITY[next].cluster;
   next    := PRIORITY[next].id;
   IF (cluster = my_cluster) AND (next = my_id)
   THEN /* become intersection */
    FOR i := 1 TO N/p DO
     IF i <> my_cluster
      THEN ADSDwI (i,1);
   ELSE request TESTED_UP(cluster,next);
  UNTIL fault-free intersection is found;
 ELSE  request TESTED_UP(cluster,next);
 next := (next + 1) MOD p;
UNTIL next is tested as fault-free;
update cluster information consistently;
End ADSDwI;
```

As the diagnosis process starts, the intersection doesn't know the state of any node in any cluster. After $N/p$ testing rounds it will have received complete information about the state of all nodes at all clusters.

### 3.2. Optimal Number of Clusters

For a given system of $N$ nodes, it is important to determine which number of clusters, $p$, of which size, optimize the diagnosis latency of the system, while keeping the number of messages as low as possible. The total number of messages in the system per testing round is given by function $f(p) = p + N/p$. To find the number of clusters that minimize the number of messages we have to find a minimum of $f(p)$. This minimum is reached when $p = \sqrt{N}$, i.e., the best organization is to organize nodes in $\sqrt{N}$ clusters of size $\sqrt{N}$. Which gives an algorithm of diagnosis latency on the order of $O(\sqrt{N})$ testing rounds.

### 3.3. The Hi-ADSD Algorithm

In this section we present the Hierarchical Adaptive Distributed System-Level Diagnosis algorithm *(Hi-ADSD)*. In Hi-ADSD, nodes are grouped in clusters, that are themselves grouped in larger clusters, so that the testing graph forms a hierarchical structure, as shown in figure 4.

At each testing interval node $i$ tests a cluster, instead of a node. We assume that clusters have sizes of powers of 2, although this can be modified. Whenever we mention $logN$, we are refering to the logarithm base 2 of $N$. Initially a node tests the other node in its $2^1$ sized cluster. After that it goes on to test the $2^2$ sized cluster. To get information

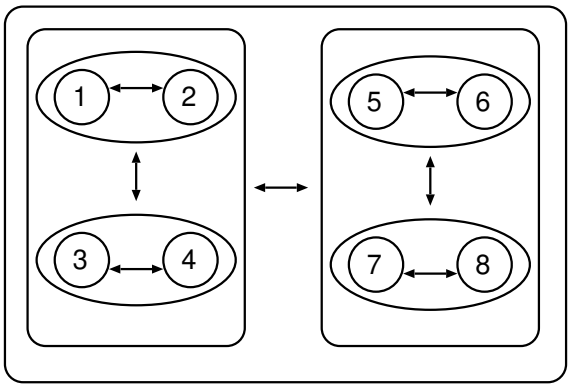


**Figure 4. A hierarchical approach to test clusters.**

about this cluster it executes tests adaptively until it finds a working node in the cluster. After it gets information about the cluster, it proceeds to the next, an so on, always from the $2^i$ to the $2^{i+1}$ cluster. The following figure shows the testing hierarchy for 8 nodes, from the viewpoint of node 1:

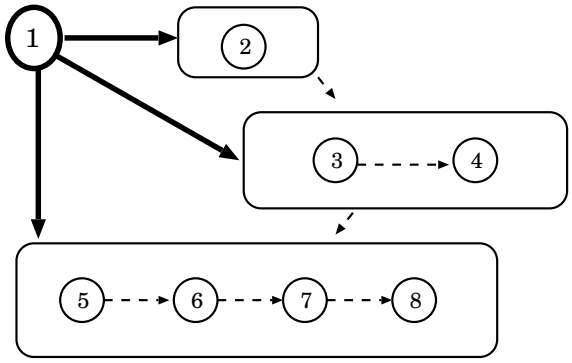


**Figure 5. Each node adaptively tests all clusters.**

Hi-ADSD uses a tree to store information about the tests in all clusters. To effectively diagnose the state of all nodes, it is sufficient to list all nodes in the tree. The following figure shows the tree for node 1, considering that all nodes are fault-free.

An outline of the algorithm is given below:

```
Algorithm Hi-ADSD;
/* at node i */
REPEAT FOREVER
  FOR cluster <- 1 TO logN
     j <- next_node(cluster,0);
     REPEAT
       test_node(j);
       IF j is non-faulty
          THEN cluster_info <- new_info
          ELSE j <- next(cluster,j);
     UNTIL j is non-faulty OR j=0 /* flag */;
     IF j=0 THEN result(cluster) <- NIL;
  END FOR;
END REPEAT;
END Algorithm.
```

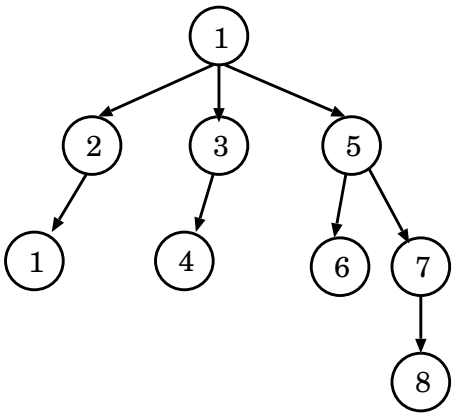


**Figure 6. A tree keeps all testing information.**

In Hi-ADSD whenever a faulty node becomes fault-free, it takes one testing round for that node to update its testing tree, which is initially empty. And during this period of time if the node is tested, the tester should take care not to copy the empty tree, which it should instead get from another fault-free node in that cluster.

We outline below a correctness and complexity proof of the algorithm.

**Lemma 1:** After $logN$ testing rounds each fault-free node will have tested all clusters in which there is a fault-free node.

*Outline of proof:* This follows from the definition of the algorithm, i.e., at a given testing interval node $i$ tests a cluster, and looks for a fault free node in that cluster. As there are $logN$ clusters to test, after $logN$ testing rounds, all nodes in all clusters will have executed the algorithm at least $logN$ times, having tested all clusters.

**Theorem 1:** The maximum shortest path in $T(S)$ is of length $t$ after $t$ testing rounds.

*Outline of proof by induction:* After one testing round, each node has tested 1 other fault-free node, and the maximum shortest path is 1. After 2 testing rounds, clusters have size 4, and, in these clusters, each node tests two other nodes, and gets information about the fourth node indirectly, through the other nodes. This makes up a path of length 2.

Assume that after $t$ testing rounds the maximum indirectness is $t$. We verify what happens after $t+1$ testing rounds. Now the cluster has size $2^{t+1}$, and there are two clusters of size $2^t$. As every node in each of these clusters will have an edge to the other cluster, the maximum shortest path from that node to a node in the other cluster is of size 1 + $t$, where $t$ is the maximum shortest path within each of the two clusters.

For example, see nodes 6 and 3 on figure 4. For 6 to get information about 3, 6 tests 2, which tests 4 which tests 3. In this system of 8 nodes, this path is the worst case, and has length $log8$.

**Theorem 2:** After at most $log^2N$ testing rounds, Diagnose executed at a fault-free node will correctly determine F(S).

*Outline of proof:* We showed in theorem 1 that the length of the maximum shortest

path in $T(S)$ is $logN$, after $logN$ testing rounds. But each of these tests takes logN testing rounds to be executed, i.e., it takes a maximum of $log^2N$ testing rounds for a node to obtain information about its farthest peer in $V(S)$.

## 4. Conclusion

We have presented new adaptive distributed system-level diagnosis algorithms in which nodes are grouped in logical clusters, permitting a divide-and-conquer testing approach that reduces the diagnosis latency of current algorithms. In ADSD with Intersections a number of clusters executing Adaptive DSD have a point of intersection, that distributes tests from all clusters to all others. In Hi-ADSD nodes test cluster organized in a hierarchical fashion, and latency is reduced to $log^2N$ testing rounds.

There are many applications for system-level diagnosis algorithms, including parallel computer diagnosis, and network fault management. We have conducted experiments in which Adaptive DSD was implemented within an SNMP-based network management system [15], and are now working on an extension of that implementation for both Adaptive DSD with Intersections and Hi-ADSD.

## References


[1] F. Preparata, G. Metze, and R.T. Chien, "On The Connection Assignment Problem of Diagnosable Systems," *IEEE Transactions on Electronic Computers*, Vol. 16, pp. 848-854, 1968.

[2] S.L. Hakimi, and A.T. Amin, "Characterization of Connection Assignments of Diagnosable Systems," *IEEE Transactions on Computers*, Vol. 23, pp. 86-88, 1974.

[3] S.L. Hakimi, and K. Nakajima, "On Adaptive System Diagnosis" *IEEE Transactions on Computers*, Vol. 33, pp. 234-240, 1984.

[4] J.G. Kuhl, and S.M. Reddy, "Distributed Fault-Tolerance for Large Multiprocessor Systems," *Proc. 7th Annual Symp. Computer Architecture*, pp. 23-30, 1980.

[5] J.G. Kuhl, and S.M. Reddy, "Fault-Diagnosis in Fully Distributed Systems," *Proc. FTCS-11*, pp. 100-105, 1981.

[6] S.H. Hosseini, J.G. Kuhl, and S.M. Reddy, "A Diagnosis Algorithm for Distributed Computing Systems with Failure and Repair," *IEEE Transactions on Computers*, Vol. 33, pp. 223-233, 1984.

[7] R.P. Bianchini, K. Goodwin, and D.S. Nydick, "Practical Application and Implementation of System-Level Diagnosis Theory," *Proc. FTCS-20*, pp. 332-339, 1990.

[8] R.P. Bianchini, and R. Buskens, "An Adaptive Distributed System-Level Diagnosis Algorithm and Its Implementation," *Proc. FTCS-21*, pp. 222-229 , 1991.

[9] R.P. Bianchini, and R. Buskens, "Implementation of On-Line Distributed System-Level Diagnosis Theory," IEEE Transactions on Computers, Vol. 41, pp. 616-626, 1992.

[10] C.-L. Yang, and G.M. Masson, "Hybrid Fault-Diagnosability with Unreliable Communication Links," *Proc. FTCS-16*, pp. 226-231, 1986.

[11] S.Rangarajan, A.T. Dahbura, and E.A. Ziegler, "A Distributed System-Level Diagnosis Algorithm for Arbitrary Network Topologies," *IEEE Transactions on Computers*, Vol.44, pp. 312-333, 1995.

[12] S.Lee, and K.G. Shin, "Probabilistic Diagnosis of Multiprocessor Systems," *ACM Computing Surveys*, Vol.26, No.1, pp.121-139, 1994.

[13] R.A Maxion, and R.T. Olszewski, "Detection and Discrimination of Injected Network Faults," *Proc. FTCS-23*, pp. 198-207, 1993.

[14] M.A. Hiltunen, and R.D. Schlichting, "Understanding Membership," *Tech. Rep. TR95-07, University of Arizona*, July, 1995.

[15] E.P. Duarte Jr., and T. Nanya, "An SNMP-based Implementation of The Adaptive DSD Algorithm for LAN Fault Management," Proc. of the IEEE/IFIP NOMS96, *to appear*, Kyoto, Japan, April 1996.