The Multi-Level Paradigm for Distributed Fault Detection in Networks with Unreliable Processors

K. Thulasiraman[1], Ming-Shan Su[2], Vakul Goel[1]
[1]University of Oklahoma, School of Computer Science, Norman, USA, {thulasi, vakul}@ou.edu
[2]Southeastern Oklahoma State University, Dept. of Computer Science and Technology, Durant, USA, msu@sosu.edu

ABSTRACT

In this paper, we study the effectiveness of the multilevel paradigm in considerably reducing the diagnosis latency of distributed algorithms for fault detection in networks with unreliable processors. This study is based on our work in [6] that generalizes the algorithm in [3].

1. Introduction

In 1967, Preparata, Metze and Chien [1] proposed a model called the PMC model and a framework, called ***System-Level Diagnosis***, for diagnosing faults in a network of processors. The PMC model allows faulty processors also to test. Thus some of the results could be unreliable. In the more than three decades following this pioneering work, several issues arising from the application of this framework have been investigated and resolved. Diagnosis algorithms based on the PMC model are assumed to be executed on a single highly reliable supervisory processor. A single supervisory processor is a bottleneck in a system with a large number of processing elements. Distributed diagnosis algorithms exploiting the inherent parallelism available in a multiprocessor system are desirable. This led Kuhl and Reddy [2] to pioneer the area of distributed system level diagnosis. Major algorithmic contributions in this area include those by Bianchini and Buskens [3], Rangarajan and Dahbura [4], and Duarte and Nanya [5]. While the algorithms in [3] and [5] assume the existence of a logical fully connected network, the algorithm in [4] is applicable to networks of arbitrary topologies. The diagnosis latency of Bianchini and Buskens' algorithm [3] (called the ADSD algorithm) is $O(N)$ testing rounds, where N is the number of processors (machines) in the network and "*testing round*" refers to the period of time in which every fault free node has tested another node as fault free and has obtained the diagnostic information from that node or has tested all other nodes as faulty. Diagnosis latency refers to the time (measured in terms of testing rounds) taken for all fault free nodes to reach a consensus view of the fault/fault-free status of the whole network, after the last fault event (failure or recovery). The recent work in [6] discusses a multilevel distributed diagnosis algorithm and discusses the effectiveness of the multilevel paradigm in distributed diagnosis. An application of this work in the design of a distributed network management system is presented in [7]. The work reported in the present paper extends and refines the preliminary results of [8]. In our presentation certain details are omitted due to space limitations. Complete discussion may be found in [6].

2. ML-ADSD: Multi-Level Adaptive Distributed System-Level Diagnosis

In this section we propose the design of a multi-level adaptive distributed diagnosis algorithm for fully connected networks, which generalizes the ADSD algorithm of Bianchini and Buskens. This algorithm will be called the **ML-ADSD algorithm**. We assume a logically connected complete network G of N nodes. The nodes are first partitioned into p clusters. Thus each cluster has N/p nodes. We assume that both N and p are powers of 2. A pictorial tree description of this algorithm is shown in Figure 1. We assume that each cluster at level 1 has at least one fault free node. We denote the last level by M.

2.1 Algorithm Specification

- **Level-*i* Clusters and Leaders**

At level 1 of the tree representation are the p original clusters, to be called the **level-1 clusters**. In general, at level i there are $p/2^{i-1}$ clusters, each containing at most two nodes which are leaders of two level-(i-1) clusters. In view of our assumption that each cluster has at least one fault free node, all clusters at levels greater than one contain exactly two nodes. In each cluster the node with the smallest id will be called the leader of that cluster. In all clusters at levels greater than 1 the leader will be called the left node and the other node will be called the right node of that cluster. Also, if $M < \log p + 1$, then at level M there will be one cluster of $p/2^{M-2}$ nodes, one for each cluster at level $M-1$. Each cluster at level $i < M$ may be viewed as the root of the subtree with 2^{i-1} level-1 clusters as leaves. Specifically, consider the jth cluster (counted from left) in level i. This cluster may be viewed as representing the level-1 clusters numbered from $j2^{i-1}$ to $(j+1)2^{i-1}-1$. Each one of the two nodes in this cluster will then represent half of this group of the level-1 clusters. That is, the left node in this cluster will represent the level-1 clusters numbered from $j2^{i-1}$ to $2^{i-2}(2j+1)-1$, and the right node will represent the clusters numbered from $2^{i-2}(2j+1)$ to $(j+1)2^{i-1}-1$. We shall refer by $P_{l,j}(i)$ the set of level-1 clusters represented by the left node in cluster j at level i, and by $P_{r,j}(i)$ the set of clusters represented by the right node in jth cluster at level i. See Figure 2 for a pictorial explanation of these definitions. If $M < \log p + 1$, then the level-1 clusters represented by the jth node at level M are those at the leaves of the subtree rooted at this node and will be denoted by $P_j(M)$.

2.2 Data Structures

In the ML-ADSD algorithm, a node $n_{i,j}$ (ith node in cluster j) uses an array called TESTED_UP$_{i,j}$ to update the testing results. TESTED_UP$_{i,j}$ is a two-dimensional array of size (N/p x p). Also the row index i represents the node id in its cluster and the column index j represents the cluster id. In addition, entry TESTED_UP$_{i,j}[u][k] = v$ at node $n_{i,j}$ means that node i in cluster j has received a diagnosis message from a neighbor node (which it has tested as fault-free) indicating node u in cluster k has tested node v in cluster k and found node v as fault-free. Also, an entry TESTED_UP$_{i,j}[i][j] = u$ means that node i itself has tested node u and found node u as fault-free in cluster j. TESTED-UP array is used to detect a cycle or equivalently a leader at level 1 and another array LEADER array is used to detect cycle at other levels.

Level-*i* Testing Rounds

Level-1 testing is the same as in the ADSD algorithm of [3]. If $M = \log p + 1$, then nodes at level i perform level-i testing as follows. The node in the cluster j at level i representing $P_{l,j}(i)$ will test the nodes in the level-1 clusters in the set

0-7803-7761-3/03/$17.00 ©2003 IEEE

$P_{r,j}(i)$ until it finds the smallest fault free node, if such a node exists, and updates in its TESTED_UP array the status of the nodes in these clusters. The other node in the cluster j at this level representing $P_{r,j}(i)$ will test the nodes in the level-1 clusters in the set $P_{l,j}(i)$ until it finds the smallest fault free node , if it exists, and updates in its TESTED_UP array the status of nodes in these clusters. If $M < log\ p + 1$, then nodes at level M perform the level-M testing as follows. The j^{th} node in this cluster will test the nodes not in $P_j(M)$ until it finds the smallest fault free node in a cluster greater than its cluster (modulo p) and updates the TESTED_UP array with the status of nodes in other clusters.

ML-ADSD Algorithm

Initially all nodes are regular (that is non-leader) nodes and are at level 1. During each round each node v executes the following.
Let k be the current level of node v.
CASE 1: If $k == 1$, then do the following.
- Execute the level-1 testing algorithm.
- If node v is a leader, then set status(v) = *leader*
- If node v is not a leader then update the TESTED_UP array with the status of nodes in other clusters.

CASE 2: If $k > 1$, then do the following.
- Execute the level-k testing algorithm.

CASE 2.1 If v detects a cycle using the LEADER array then do the following.
- If k is not the last level then do:
  (a) If v is not a leader then change the level of v to 1 and End testing round.
  (b) If v is a leader at level k, then change the level of v to k+1 and End testing round.
- If k is the last level, then change the level of v to 1 and End testing round.

CASE 2.2 If v does not detect a cycle at level k, then test
- If v is still the fault free node with the smallest id in the group of level-1 clusters at the leaves of the subtree rooted at v.
  - If yes, execute the level-1 testing algorithm and End round.
  - Otherwise, change the level of v to 1

2.2 Proof of Correctness and Diagnosis Latency of the ML-ADSD Algorithm

We first consider the two level algorithm. In this scheme shown in Fig. 3 there will be one cluster of p nodes at level 2. We can view the ADSD algorithm as a level-1 algorithm. Since all the nodes are at level 1 in the ADSD algorithm, in at most N testing rounds after the last fault event every fault free node will have correct and consistent fault status information of all the nodes in the network. But in the case of the multilevel algorithm some of the faulty nodes could be at level 2 and may recover to become fault free nodes while being at level 2. Until they return to level 1, the information they contain is not reliable and may not be correct. Also, they may not perform level-1 testing, thereby preventing the election of the leaders at level 1. So, in proving the correctness of the two-level algorithm we need to ensure that a faulty node after recovery does not prevent the election of leader at level 1 and also eventually returns to level 1. ML-ADSD algorithm takes care of this.

Without loss of generality, we assume that initially all nodes are fault free and are regular nodes. Our proof of correctness involves establishing that after executing a certain number of testing rounds after the occurrence of the last fault event, all the nodes will have the correct view of the fault status of all the nodes in the system. The algorithm may be viewed as consisting of three phases.

Phase 1: In this phase all the nodes identify their respective leaders. The nodes in each level-1 cluster acquire correct view of the fault status of all the nodes in that cluster. In order to elect the leader each node must identify, using the data in the TESTED_UP array, the cycle of fault free processors at level 1. To do so, each node must execute certain number of level-1 rounds after the last fault event. We need to consider several cases.

Case 1: No faults occur.

In this case, all nodes are regular nodes and as in the ADSD algorithm, in at most ***N/p* level-1 testing rounds** after the last fault event, all nodes in each level-1 cluster will get consistent and correct fault status information of all nodes in that cluster, and the leader of each cluster will be selected.

In the following, cluster refers to a level-1 cluster.

Case 2: A faulty node can recover only when it is at level 1.

Without loss of generality, let us assume that the last fault event is the recovery of a faulty node, say node k, in cluster 1. Let C_1, the cycle of fault free processors in cluster 1 just before the last fault event be as in Figure 4(a). Let i be the fault free node with the smallest id in C_1. In the worst case node k may also be in a cycle C_2 of faulty processors before its recovery. Cycle C_2 is shown in Figure 4(b). Since node k recovers in the last faulty event, the cycle of fault free processors after the recovery of node k will be as in Figure 4(c). This cycle needs to be identified by all the fault free nodes in cluster 1.

- Let us first assume that node k is less than i.

In the first testing round after the recovery of node k, node i will execute a level-1 testing and identify node i_1 as the next node in the cycle of fault free processors. Node i being the leader of cluster 1 before the recovery, the TESTED_UP$_i$ will indicate that node i is a leader and so it will change its status to "leader". This is because, in the TESTED_UP$_i$ array of node i, node k will not be identified as a fault free processor until after a certain number of testing rounds. Other nodes including node k also will execute one level-1 testing in this first round, but will not identify themselves as leaders. In the second round, being a leader node, node i will execute a level-2 testing .It will not find itself to be the fault free node with the smallest id in cluster. So, it will set its status to "regular". This completes the actions taken by node i in the second round after the recovery of node k. Other nodes in cluster 1 will execute one level-1 testing and remain as regular nodes. These nodes also will identify the next two nodes in the fault free cycle C_3. In particular, node i_1 will identify i_2 and i_3 as the next two nodes in C_3. In the third round, node i will execute one level-1 testing and identify itself as leader. After testing i_1 fault free and updating its TESTED_UP array, it will also identify nodes i_1, i_2, and node i_3 as the next three nodes in the fault free cycle C_3. Thus in an even round node i will execute one level-2 testing and in an odd round it will execute one level-1 testing. This will continue until node i identifies node k as the smallest fault free node in C_3 and node i recognizes that it is no longer the leader of its cluster. All other nodes will execute only level-1 testings. Thus, if node k is xth one after node i in the fault free cycle C_3, then these x nodes will be identified by node i in x or x+1 rounds after the recovery of node k, and the cycle C_3 of fault free processors will be identified by all the nodes in at most ***N/p* or *N/p*+1 testing rounds** after the last fault event. Also, at the end of these rounds, node k will be identified as

the leader of cluster 1. The above reasoning is applicable even if more than one node recovers in the last fault event.

- Next consider the situation when k is greater than i.

In this case, in the first round after the last event, node i will execute a level-1 testing round and may find itself as leader. In the second and subsequent testing rounds, being a leader and the fault free node with the smallest id, node i will execute a level-1 testing as well as a level-2 testing and will remain at level 2. On the other hand, all other nodes in cluster 1 will execute only level-1 testings. Thus, in at most ***N/p* rounds** after the last event, node i and all other nodes in cluster 1 will identify node i as their leader

Case 3: Faulty nodes at level 2 may recover

Let the last fault event be the recovery of node k at level 2.

(a) Let us first consider the situation when node k is not the fault free node with the smallest id in cluster 1 after the last fault event. In the first round after its recovery, node k will execute a level-2 round and will return to "regular" node status. In the second round, it will execute one level-1 testing and may find itself as a leader because, possibly, it was in a cycle of faulty nodes before the last fault event. In the third round it will execute one level-2 testing and again return to "regular" node status. In the fourth round it will execute one level-1 testing and may again find itself to be a leader. This sequence of alternation between "regular" and "leader" status will continue until at most ***N/p* rounds** are executed. At the end of these rounds, node k and other nodes will identify the cycle C_3.

(b) Let us next consider the situation when node k is the fault free node with the smallest id in cluster 1 after it recovers in the last fault event. In this case, in each round after the last fault event, the node k will execute both a level-1 testing and a level-2 testing and will remain at level 2. All other nodes will execute level-1 testings in these rounds. So, in at most ***N/p* rounds** after the last event, every node in cluster 1 will identify node k as the leader.

Summarizing, in **at most *N/p* or *N/p* +1** rounds after the last fault event all nodes will identify the leader nodes and acquire correct view of the status of all the nodes in their respective clusters.

Phase 2: During this phase, in **at most *p* rounds** the fault free leaders of the clusters identify the ring of leader nodes at level 2 using the LEADER array, update their TESTED_UP arrays with the status information of all nodes in clusters other than their own.

Phase 3: During this phase consisting of **at most *N/p* Level-1 rounds**, the information at the fault-free leader nodes will be propagated to the remaining nodes in their respective cluster.

Theorem 1: The diagnosis latency of the two-level adaptive distributed diagnosis algorithm is **at most $2(N/p) + p + 1$ rounds.//**

We next consider the multilevel (level > 2) algorithm. Again we view the ML-ADSD algorithm as consisting of three phase. First we assume that the number of levels $M = \log p + 1$.

Case 1: No faults occur

Phase 1: In at most ***N/p* level-1 rounds** after the last fault event, all nodes in each level-1 cluster will get consistent and correct fault status information of all nodes in that cluster, and the leader of each cluster will be selected.

Phase 2: The leaders (left nodes of the two clusters at level $\log p$ of the first cluster (0^{th} cluster) and the $N/2\text{-}1^{th}$ cluster) will reach the last level in $2(\log p\text{-}1)$ testing rounds, and then perform two testing rounds. At the end of these ***2logp* rounds**, these two nodes will have correct status information of all the nodes in the network.

- While the left nodes of the two clusters at level $\log p$ move to the last level, the right nodes of these clusters move to level 1. After performing one testing round at level 1, these right nodes will move to level $\log p$ after $2(\log p\text{-}2)$ rounds and perform two additional rounds at that level to collect information from the left nodes in their respective clusters the correct fault status information of all nodes in the network. In all, they perform **$2(\log p\text{-}1) + 1$ rounds** to collect correct fault status information of all nodes in the network.
- Continuing as above, in general, the right nodes of the clusters at level i perform $2(i\text{-}1) + 1$ rounds to collect the correct status information of all nodes in the network. Note that $2 \le i \le \log p$.

Phase 3: Finally, the nodes in all level-1 clusters will collect from their respective leaders the correct fault status information of all the nodes in the network in at most ***N/p* level-1 testing rounds**

Combining all these rounds, all nodes will have correct fault status information of all the nodes in the network in at most $(N/p + [2 \log p + 2 (\log p - 1) + 2 (\log p - 2) + \ldots\ldots\ldots + 2] + \log p - 1 + N/p)$

$= 2\ N/p + (\log p + 1)(\log p) + \log p - 1$

$= 2\ N/p + (\log p + 2) \log p - 1$ testing rounds.

Case 2: Only Faulty nodes at level 1 recover

As in Case 2 in the discussion in the previous section, all nodes will identify their cluster leaders in Phase 1 in at most N/p or $N/p + 1$ rounds. The other phases will proceed as in Case 1 above. So the diagnosis latency in this case is one more than that for the first case.

Case 3: Faulty nodes at levels greater than one may recover

In this case, an additional at most 2 $\log p$ testing rounds will be required for all the fault free nodes to return to level 1. So, in this case the diagnosis latency is **$2\ N/p + (\log p + 4) \log p$ rounds**.

Theorem 2: If $M = \log p + 1$, the diagnosis latency of the M-Level ML-ADSD algorithm is at most **$2\ N/p + (\log p + 4) \log p$ testing rounds.//**

Proceeding as above, we can determine the diagnosis latency for the M-level ML-ADSD algorithm as in Theorem 3.

Theorem 3: The diagnosis latency of the M-level algorithm (M > 2) is at most

$2\ N/p + (M\text{-}2)(M+4) + p2^{(-M+3)} + 1$ testing rounds.

Using similar arguments we can show that the total number of tests performed by the ML-ADSD algorithm is at most $N^2(2/p+(2(\log p(\log p+4)/N))$. Using these results we can compute the optimum value of p which achieves the best performance in terms of diagnosis latency and number of tests.

3. Simulation

We simulated the ML-ADSD algorithm and performed extensive testing for the case $M = \log p+1$ levels. We have used the discrete event simulation language SSS [9]. From the results in Table 1 it can be seen that the ML-ADSD algorithm is superior to the ADSD algorithm both in terms of the diagnosis latency as well as the number of tests performed. This work thus demonstrates the effectiveness of the multi-level paradigm in distributed fault detection.

4. Summary

In this paper we presented the design of a multilevel distributed diagnosis for fault location in networks with unreliable processors. Our work demonstrates the effectiveness of the multi-level

paradigm in reducing the diagnosis latency and the number of tests performed

5. References

[1] F. P. Preparata, G. Metze, R. T. Chien, "On the Connection Assignment Problem of Diagnosable Systems", IEEE Transactions on Electronic Computers, Vol. EC-16, No. 6, December 1967, pp. 848-854.

[2] J.G. Kuhl, S. M. Reddy, "Fault-Diagnosis in Fully Distributed Systems", Proceedings of the 11th International Symposium on Fault Tolerant Computing, 1981, pp. 100-105.

[3] R. Bianchini, R. Buskens, "Implementation of On-Line Distributed System-Level Diagnosis Theory", IEEE Transactions on Computers, Vol. 41, No. 5, May 1992, pp. 616-626.

[4] 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, 1995, pp. 312-333.

[5] E. P. Duarte Jr., Takashi Nanya, "A Hierarchical Adaptive Distributed System-Level Diagnosis Algorithm", IEEE Transactions on Computers, Vol. 47, No. 1, Jan. 1998, pp. 34-45.

[6] M.-S. Su, *Multilevel distributed diagnosis and the design of a distributed network fault detection system based on the SNMP protocol*, Ph.D. thesis, School of Computer Science, University of Oklahoma, 2002.

[7] M.-S. Su, K. Thulasiraman, and A. Das, "A Scalable On-Line Multilevel Distributed Network Fault Detection/Monitoring System Based on the SNMP Protocol", IEEE GlobeCom2002, Nov. 2002, Taipei, Taiwan.

[8] M.-S. Su, K. Thulasiraman, and A. Das, "A multi-level adaptive distributed diagnosis algorithm for fault detection in a network of processors", Proc. 39th Annual Allerton Conf. on Communication, Control, and Computers, 2001.

[9] M. Pollastshek, Programming Discrete Simulations – Tools for Modeling the Real World, R&D Books, 1996.

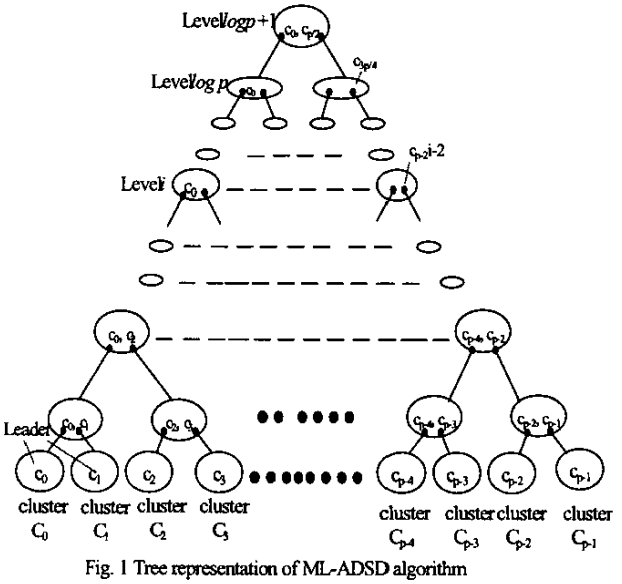

Fig. 1 Tree representation of ML-ADSD algorithm

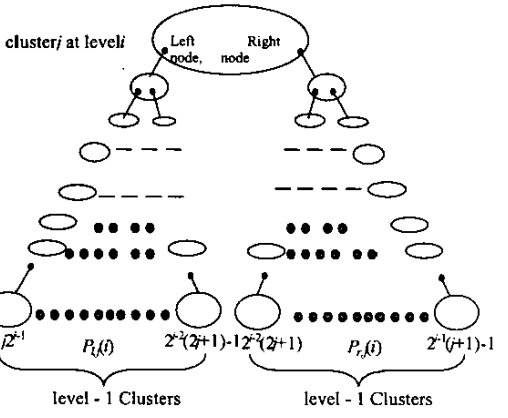

Fig. 2 Subtree representation at the node of cluster j at level i

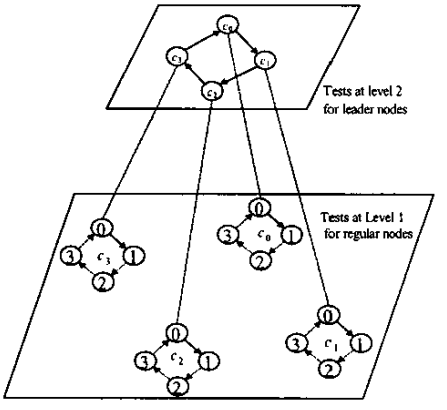

Fig. 3 A two-level scheme

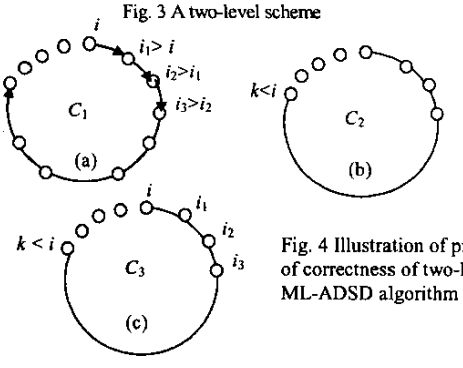

Fig. 4 Illustration of proof of correctness of two-level ML-ADSD algorithm

Number of Nodes	p x N/p	Avg. # of Tests	# Testing Rounds (diagnosis latency)
64 nodes	1 cluster	2417	39
	8 by 8	897	14
128	1 cluster	7957	63
	8 by 16	2659	20
256	1 cluster	34790	137
	16 by 16	6483	24
512	1 cluster	154009	306
	32 by 16	16588	32
	16 by 32	19381	37

Table 1. ML-ADSD algorithm vs. ADSD algorithm (1 cluster)