VCube: A Provably Scalable Distributed Diagnosis Algorithm

Elias P. Duarte Jr., Luis C. E. Bona and Vinicius K. Ruoso
Dept. Informatics
Federal University of Parana
Curitiba, PR 81531–990
Email: {elias,bona,vkruoso}@inf.ufpr.br

***Abstract*—VCube is a distributed diagnosis algorithm for virtually interconnecting network nodes. VCube presents several logarithmic properties, and is a logical hypercube when all nodes are fault-free. VCube is dynamic in the sense that nodes can leave and rejoin the system as they become faulty and are repaired. The topology re-organizes itself and keeps its logarithmic properties even if an arbitrary number of nodes are faulty. Fault diagnosis is based on tests. All fault-free nodes of a system with N nodes detect an event with a latency of at most $log_2^2 N$ testing rounds. In this work we specify the algorithm and show that the worst number of tests executed is $Nlog_2 N$ per $log_2 N$ rounds. Besides the correctness proofs, experimental results are also given.**

I. Introduction

As computer systems have become critical for organizations and individuals, it is important to design systems that are resilient to faults. System monitoring is the first step towards resilience: before anything else it is important to be able to efficiently detect if and which components of the system are faulty. System-level diagnosis is a practical approach for monitoring systems that consist of several units [7], [6], [9]. Monitoring is based on tests executed among the system units (also called nodes). Several approaches have been proposed for building effective testing assignments, i.e. determining which units should test which other units. Diagnosis is completed based on the set of test results obtained, which is called the system *syndrome*. The output produced by a diagnosis algorithm consists of a list specifying which units are faulty and which are fault-free.

Distributed diagnosis [4] allows the construction of fault-tolerant monitoring systems: the units not only execute tests but also receive test results from other units, build the syndrome locally, and diagnose the system. In other words, distributed system-level diagnosis allows the fault-free nodes of a system of N nodes to determine the state of every node, either as faulty or fault-free.

VCube is a distributed diagnosis algorithm to virtually interconnect network nodes. The system is assumed to be dynamic in the sense that fault-free nodes can leave the system as they become faulty (crash) and then re-join the system as they are repaired. When all nodes are fault-free, the topology is a perfect hypercube and thus presents several logarithmic properties. However as nodes become faulty (crash) the remaining fault-free nodes re-organize themselves in order to keep the logarithmic properties. Repairs also cause a re-organization of the topology.

VCube allows the fault-free nodes of a system of N nodes to learn about new events in at most $log_2^2 N$ testing rounds. In this work we prove that maximum number of tests executed per $log_2 N$ testing rounds is $Nlog_2 N$. Furthermore a strategy to disseminate diagnostic information is defined which allows nodes to obtain as much new information as possible at each test executed. Experimental results show that this strategy reduces the average latency significantly.

The rest of this paper is organized as follows. Section 2 gives an overview of system-level diagnosis in general, and hierarchical diagnosis in particular. In Section 3 VCube is specified, and its correctness is proved. Section 4 presents experimental results obtained with simulation. Section 5 points to related work. The conclusion follows in Section 5.

II. Hierarchical Distributed Diagnosis

In 1967 Preparata, Metze and Chien proposed the first system-level diagnosis model for solving the problem of determining which units of a computer system are faulty and which are working as expected. That model was named after the authors initials: the PMC model [1]. Diagnosis is based on the execution of tests: system nodes are capable of testing each other, and of reporting the results of the tests executed. A node can be in one of two states, fault-free (the node is working perfectly according to its specification) and faulty (the node's behavior departs from what is expected). A fault-free node is able to execute tests and report test results correctly. The test is actually defined as a "testing procedure", which is often technology-dependent and may involve the execution of several instructions.

In 1974 Hakimi and Amin [2] charaterized the PMC model. In particular, they proved the *diagnosability* of a system under this model. A system is t-diagnosable if diagnosis can be successfully completed even if up to t nodes are faulty. They showed that a system is t-diagnosable if no two units test each other, $N \geq 2t + 1$, and each unit is tested by at least t other units.

In the begining of the 1980's two diagnosis approaches were proposed based on the PMC model which highly extended its capabilities: Adaptive Diagnosis and Distributed Diagnosis. In Adaptive Diagnosis [3] nodes execute tests in rounds, and each

978-1-4799-7562-4/14 $31.00 © 2014 IEEE
DOI 10.1109/ScalA.2014.14

node determines which tests it should execute based on the results of previously executed tests. In the original PMC model and all variants that appeared before adaptive diagnosis was proposed, a static testing assignment was decided beforehand, and nodes executed the same tests whenever diagnosis was executed. In Distributed Diagnosis, besides executing tests, system nodes receive test results from other nodes, collecting the syndrome (defined as the set of all test results) and diagnosing the system. In other words: the fault-free nodes themselves determine which nodes are faulty and which are fault-free in the system. In the original PMC model and all other models that appeared before Distributed Diagnosis was proposed, the test results are sent to a central observer which is an external entity assumed to never fail and which completes the diagnosis of the system. Note that although in the original PMC model the fault model assumed was similar to the Byzantine fault model [10], most distributed diagnosis algorithms assume the crash fault model in practice.

In 1992 Bianchini and Buskens [5] put together these two models and proposed the Adaptive Distributed System-level Diagnosis (Adaptive-DSD) algorithm. Nodes running Adaptive-DSD are assigned sequential identifiers from 0 to $N-1$, where N is the number of nodes in the system. A given fault-free node i running Adaptive-DSD executes tests sequentially on nodes $i+1, i+2, ...$ until it tests another fault-free node. Node 0 follows node $N-1$. In this way fault-free nodes form a cycle in the testing graph, defined as graph $G = (V, E)$ where V is the set of nodes and E is the set of tests executed in the latest testing round, i.e. there is an edge from node i to node j in E if node i tested node j in the most recent testing round. Upon testing a fault-free node, the tester obtains diagnostic information about every node in the system that it has not tested in the current round. The latency of a diagnosis algorithm is defined as the number of testing rounds it takes for every fault-free node to correctly diagnose that a fault-free node has become faulty or vice-versa. Adaptive-DSD presents a latency of N testing rounds in the worst case, and guarantees that the number of tests executed per round is at most N.

The Hierarchical Adaptive Distributed System-level Diagnosis algorithm [8] is a distributed diagnosis algorithm that groups nodes in progressively larger clusters, employing a divide-and-conquer testing strategy. Clusters are sets of nodes, the number of nodes in a cluster is a power of 2. Initially assume that the number of nodes in the system N is a power of 2 - note that this is not a requirement for running the algorithm. A tester executes tests periodically at a testing interval. Initially the cluster with 1 (2^0) node is tested. In the next interval the cluster with 2 (2^1) nodes is tested. Then in the third interval the cluster with 4 (2^2) nodes is tested, and so on until the cluster with N/2 ($2^{\log_2 N-1}$) nodes is tested. Then the first cluster is tested again in the next interval, and the process is repeated. When all nodes are fault-free the testing assignment is equivalent to a hypercube. However this is a logical topology, and the underlying system is assumed to be fully connected, i.e. any node is able to test any other node if this is required.

When a tester finds a faulty node it continues executing tests until a fault-free node is found, or all nodes in the cluster are tested faulty. A testing round occurs when all fault-free nodes have executed their assigned tests. Nodes are not assumed to be synchronized with each other. Diagnosis is completed in at most $log_2^2 N$ testing rounds, and the maximum number of tests executed in a testing round is $N^2/4$.

Hi-ADSD with Detours [11] was proposed in order to guarantee that the maximum number of tests executed remains a logarithmic function. When a faulty node is tested on a given cluster, the tester checks if there are *detours* to the remaining nodes of that cluster. Extra tests are only executed if there are no detours available. A detour is a path to a given node in the testing graph from outside the cluster. The length of a detour is never larger than the length of the path between the same endpoints in the hypercube (i.e. in the system in which all nodes are fault-free). Remember: in Hi-ADSD a tester only obtains information about a node when its cluster is tested. Nodes executing Hi-ADSD with Detours, on the other hand, can obtain information about a given node from other clusters. Although simulation results confirm that the number of tests executed is a logarithmic function, this bound has not been formally proved.

Hi-ADSD with Timestamps [13] was specified taking advantage of the fact that there are several paths from a tester to a tested node. Instead of using one such path as Hi-ADSD and Hi-ADSD with Detours do, this algorithm allows the tester to obtain diagnostic information from *all* paths with size less than or equal than the length of the path between the same endpoints in the hypercube. However, if the tester obtains information about a given node from two or more tested nodes, it is important to determine which information is most recent. Timestamps are defined as state counters which a node keeps for every other node [12]. Initially a timestamp is set to zero, which corresponds to a fault-free state. As an event is detected, the timestamp is incremented. Thus an even timestamp corresponds to a fault-free node, and an odd timestamp corresponds to a faulty node. A tester only updates diagnostic information when the timestamp obtained is greater than the known timestamp for a given node. Simulation results have shown that the mean latency of the algorithm is substantially reduced in comparison with the original Hi-ADSD.

In the Distributed Virtual Hypercube Algorithm (DiVHA) [14] each node keeps a local graph representing the set of tests executed in the system. As a node tests a cluster it checks in its local graph which information can be obtained from previously tested clusters. The number of tests executed is proved to be logarithmic in the worst case, but it requires running several graph searches in order to guarantee that there is only one tester per node per cluster. DiVHA was employed to maintain a network overlay in which parallel and distributed applications were executed using a grid environment.

In the next section we present VCube a new hierarchical adaptive distributed system-level diagnosis algorithm that

presents a significant improvement over the other algorithms: VCube allows nodes to use several paths to obtain diagnostic information (detours); it is able to determine which information is newer (timestamps); and also guarantees that the number of tests executed is logarithmic in the worst case, but uses a simple and elegant way to assign tests based on ordering the candidate testers of each node of each cluster.

III. VCube: A Provably Scalable Distributed Algorithm

VCube employs a hierarchical testing strategy which organizes the system nodes in progressively larger clusters. In this section we initially give a new expression for computing hierarchical clusters, after which the algorithm is specified.

A. The $c_{i,s}$ Function

The list of ordered nodes tested by node i in a cluster of size 2^{s-1}, is denoted by $c_{i,s}$, $s = 1..log_2N$. In the first testing interval, the tested cluster for node i is $c_{i,1}$; in the next testing interval, the tested cluster is $c_{i,2}$, and so on until the tested cluster is c_{i,log_2N}. Then, in the next interval the tested cluster is again $c_{i,1}$ and the the same process continues. When a fault-free node is tested, the tester obtains diagnostic information from the tested node.

Below we give a new compact expression for computing $c_{i,s}$, $i = 0..N-1$, in which $\oplus$ corresponds to the exclusive or operation.

$$c_{i,s} = i \oplus 2^{s-1} \parallel c_{i\oplus 2^{s-1},k} \mid k = 1, 2, \ldots, s-1 \qquad (1)$$

This function can be described as follows. First the hypercube neighbor of node i in cluster s is computed. The identifiers of these two nodes differ only in one bit, the bit that is equal to one in 2^{s-1}. The remaining nodes in the cluster are the nodes in clusters 1, 2, ... , s-1 of the hypercube neighbor, i.e. $c_{i\oplus 2^{s-1},1}$, $c_{i\oplus 2^{s-1},2}, \ldots, c_{i\oplus 2^{s-1},s-1}$

Table I shows the $c_{i,s}$ function results for a system with 3 dimensions. Consider for example node 0 and its third cluster $c_{0,3}$. Initially function $c_{0,3}$ returns the hypercube neighbor of node 0 in the cluster, i.e. node $0 \oplus 2^2 = 4$. In this cluster the hypercube node that is adjacent to node 0 (000_2) is the node for which only the third least significant bit is different, 4 (100_2). After that the other nodes in the cluster are those in $c_{4,1}$, and $c_{4,2}$. As $c_{4,1} = (5)$, and $c_{4,2} = (6, 7)$, then $c_{0,3} = (4, 5, 6, 7)$.

Another example is $c_{6,3}$; the hypercube neighbor of node 6 in this cluster is node 2. Thus $c_{6,3} = (2, c_{2,1}, c_{2,2})$. The same reasoning can be applied to systems with higher dimensions. For instance consider a system with $1,024$ nodes, thus $s = 1, 2, \ldots, 10$, and $c_{6,10} = (2, c_{2,1}, c_{2,2}, \ldots, c_{2,9})$.

B. VCube: Specification

VCube employs a new testing strategy is based on the following rule: before node i executes a test on node $j \in c_{i,s}$, it checks whether it is the first fault-free node in $c_{j,s}$ (please note that the index is ***j***). This strategy is different from that employed by Hi-ADSD. A fault-free node running Hi-ADSD executes tests on a cluster until it tests another node as fault-free, or all nodes are tested faulty. In this way, several nodes may execute tests on the same nodes of a given cluster. Thus in Hi-ADSD the number of tests executed is quadratic in the worst case; we prove that VCube employs a logarithmic number of tests in worst case.

After node i tests node $j \in c_{i,s}$ as faulty-free, it obtains new diagnostic information from tested node j. This information is not restricted to the cluster: the tester may obtain *any* new information that the tested node has - in comparison with its own local information. As a tester may obtain information about a given node from different tested nodes, the algorithm employs timestamps to determine which information is newer. Initially every node is assumed to be fault-free and the corresponding timestamp is zero. As an event is detected, i.e. a fault-free node has become faulty or vice-versa, the corresponding timestamp is incremented. Thus an even timestamp corresponds to a fault-free node and an odd timestamp to a faulty node.

```
Algorithm VCube executed at node i
repeat
    for s ← 1 to log2N do
        for all j ∈ c_{i,s} | i is the first faulty-free node ∈ c_{j,s} do
            test(j)
            if j is fault-free then
                if t_i[j] mod 2 = 1 then t_i[j]++
                obtain diagnostic information from j
            else
                if t_i[j] mod 2 = 0 then t_i[j]++
        sleep until next testing interval
forever
```

Algorithm 1: VCube: the algorithm.

VCube executed by fault-free node i is given in pseudocode in Algorithm 1. Node i keeps timestamp array $t_i[0..N-1]$. After obtaining information from node j tested as fault-free, node i only updates a local timestamp entry when the obtained value is greater than the current one. An extra advantage of this strategy is that the average latency of the algorithm is much improved, as testers can obtain new diagnostic information about every system node from any tested node.

In order to avoid transferring the complete t_j array as node i tests node j as fault-free, it is possible to implement a simple solution in which node j only sends new information, i.e. information that has changed since the last time node j was tested by node i.

C. Correctness

Theorem 1: The number of tests executed by all nodes running VCube is at most $Nlog_2N$ per log_2N testing rounds.

Proof: Consider the tests executed at node j. For each $c_{j,s}$ (s=1, .., log_2N) node j is tested by the first fault-free node in $c_{j,s}$. Each fault-free node in $c_{j,s}$ uses the same ordered list of nodes containing itself to check whether it is the first fault-free node on that list and must test node j. Thus each node is tested at most log_2N times per log_2N testing rounds. As there are N nodes, the total number of tests is at most $Nlog_2N$. ■

TABLE I
THE $c_{i,s}$ FUNCTION FOR AN 8 NODE SYSTEM.

s	$c_{0,s}$	$c_{1,s}$	$c_{2,s}$	$c_{3,s}$	$c_{4,s}$	$c_{5,s}$	$c_{6,s}$	$c_{7,s}$
1	1	0	3	2	5	4	7	6
2	2,3	3,2	0,1	1,0	6,7	7,6	4,5	5,4
3	4,5,6,7	5,4,7,6	6,7,4,5	7,6,5,4	0,1,2,3	1,0,3,2	2,3,0,1	3,2,1,0

In order to see that the same latency bounds proved in [8] hold for the algorithm with the new testing strategy, it suffices to see that a fault-free node may take up to $log_2 N$ testing rounds to test a given cluster, and the minimum distance from any node to any other node is at most $log_2 N$, thus the latency is $log_2^2 N$ testing rounds in the worst case.

IV. EXPERIMENTAL EVALUATION

In this section we present experimental results obtained with simulation. The simulator was built using the Simulation Programming Language (SMPL). The experiments were executed on a 512 node system. Each node starts testing a random cluster. We report results for the average number of tests and average latency. The latency is the number of testing rounds required by all fault-free nodes to detect an event. In all graphics the number of tests reported were computed as follows. After a scheduled event occurred and was diagnosed by every fault-free node, we counted the number of tests executed in the next $\log_2 N$ testing rounds. Results were computed from executions of the algorithm on more than 7,000 random fault situations.

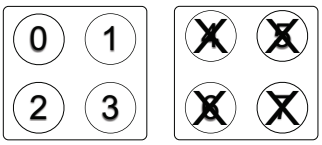

Fig. 1. A fault situation that causes the worst case number of tests of Hi-ADSD.

Figure 1 shows the fault situation that leads of the worst case number of tests of Hi-ADSD. Figure 2 shows the average number of tests executed for that fault situation. This situation occurs when all nodes of a cluster of size $N/2$ are faulty simultaneously. In the experiment, nodes become faulty one after the other, until all nodes of the cluster are faulty. The choice of the node to become faulty at a given time instant was random. The figure clearly shows that while the number of tests executed by Hi-ADSD reaches $N^2/4$; VCube employs at most $Nlog_2 N$ tests.

Figure 3 shows the average latency for the same fault situations for which the number of tests executed are shown in Figure 2. The average latency drops significantly for VCube in comparison with Hi-ADSD. The reason is that as timestamps are used, a node can obtain information about new events from any other node tested fault-free. When the number of faulty nodes reaches $N/2$, the latency of Hi-ADSD also reduces significantly. The reason is the number of tests executed sigficantly increases.

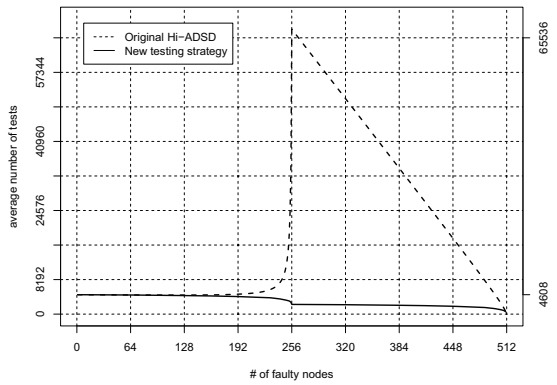

Fig. 2. Number of tests: worst case for Hi-ADSD and VCube.

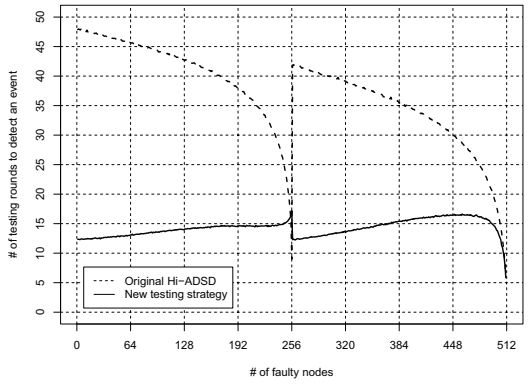

Fig. 3. Diagnosis latency for the fault situations of Figure 2.

Figure 4 shows the average number of tests executed for both Hi-ADSD and VCube when randomly chosen nodes become faulty. In this case both algorithms virtually employ the same number of tests. However the average latency for diagnosing these fault situations drops significantly for the new algorithm as shown in Figure 5. The reasons for this improved latency are the same discussed for the experiment above.

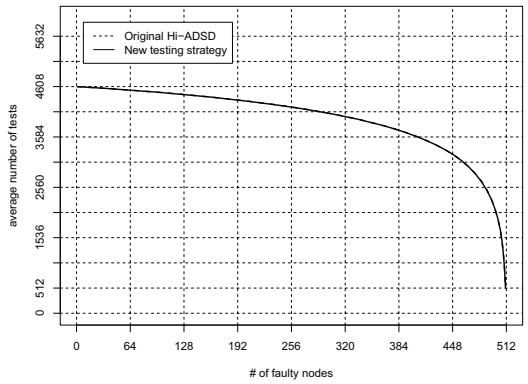

Fig. 4. Number of tests: randomly chosen nodes become faulty.

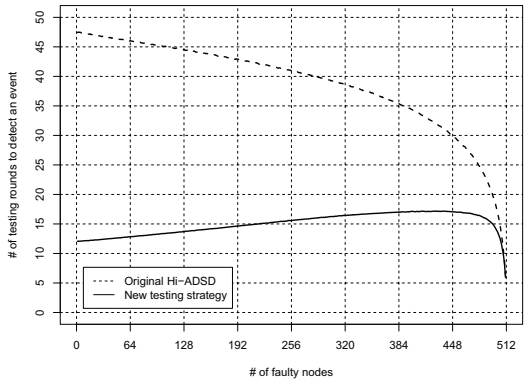

Fig. 5. Diagnosis latency for the fault situations of Figure 4.

V. Related Work

Among the several applications of Hi-ADSD, network fault management was the first. The algorithm was implemented using the Internet standard Simple Network Management Protocol (SNMP) [21]. The resulting system [20] is a fault-tolerant fault monitoring strategy that can be deployed in LANs running the TCP/IP protocols. Another hierarchical distributed diagnosis algorithm that was also implemented with SNMP is ML-ADSD (Multi-Layer ADSD) that employs a tree-based topology in which the Adaptive-DSD algorithm is run on each layer. Another algorithm that uses the concept of several layers is presented in [24]. This algorithm is configurable, and explores the trade-off that exists between the latency and number of tests employed in order to achieve the best result for particular systems.

Diagnosing complex systems deployed in multi-tier environments is the focus of the strategy presented in [23]. The authors propose a probabilistic diagnosis model for arbitrary failures in components of distributed applications executed in the Internet. The monitoring system observes the message exchanges between components and performs a probabilistic diagnosis of the component that was the root cause of a failure. This idea was extended in [22], in which a hierarchical monitor framework is proposed that consists of several individual monitors. Each monitor builds a causal graph between the components based on their communication and uses a rule base of allowed state-transition paths to diagnose the failure. Tests are rule-based and are assumed to have imperfect coverage.

Diagnosis strategies have also been applied to wireless networks. Note that this type of network requires a different system-model in comparison with that adopted by Hi-ADSD. Hi-ADSD assumes a fully connected network in which any node can test any other node. In a wireless network the system is partially connected, thus a multi-hop model has to be assumed, in which some nodes cannot reach others, and have to learn about their states through still other nodes. In [28] the authors implemented the RDZ algorithm for diagnosis of general topology networks in a wireless ad hoc network. A hierarchical system-level diagnosis algorithm is described in [18] which is based on a clustering strategy and is also applied to identify faults in wireless ad hoc networks. In [17] a comparison-based diagnosis strategy is presented that is based on neural networks and performs well even with an incomplete syndrome.

Another related type of diagnosis algorithm is that for physical hypercubes. In [15] the authors propose a hierarchical adaptive diagnosis algorithm for hypercubes that is not distributed, i.e. the syndrome is collected by a central observer that diagnoses the system. More recently [19] a scalable algorithm based on the comparison approach was proposed for not only the hypercube but also several variants, including crossed cubes, generalized twisted cubes, recursive circulants, among others. This algorithm is based on the MM* model which is also not distributed.

A two-phase hierarchical diagnosis algorithm is presented in [25]. In the first phase consists of local preliminary diagnosis which is executed by nodes called “diagnosers”. In the second phase diagnosers communicate in order to share information with the eventual purpose of reaching global consistency. A hierarchical testing graph consisting of several independent hypercubes is proposed in [27]. This approach results in a regular testing graph with degree $t + 1$, where t corresponds to the diagnosability of the algorithm. The algorithm is non-adaptive, i.e. nodes only execute the pre-defined tests, and non-distributed, i.e. test results are sent to a central observer that diagnoses the system. Another hierarchical diagnosis algorithm based on this model is presented in [26], but in this case not only the hypercube topology is considered but also array and tree-based topologies. This algorithm employ a voting strategy to complete diagnosis that allows incomplete tests and guarantees diagnosis as long as the majority of tests report precise results.

VI. Conclusion

In this paper we presented VCube, a distributed diagnosis algorithm which presents several logarithmic properties,

including the latency – i.e. the time it takes for a fault-free node to diagnose an event – and the number of virtual edges, which correspond to tests – at most $Nlog_2N$ tests are executed per log_2N testing rounds. Nodes running the algorithm form a hypercube like topology, that maintains its logarithmic properties even if an arbitrary number of nodes become faulty. Furthermore as nodes obtain diagnostic information from multiple sources, the algorithm presents a lower average latency.

Our future work is focused on developing applications on top of VCube. The virtual topology is a practical, resilient approach to allow nodes of a distributed system to offer robust services.

ACKNOWLEDGMENT

This work was partially supported by grant 309143 / 2012-8 from the Brazilian Research Agency (CNPq).

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] 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.

[5] 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.

[6] G. Masson, D. Blough, and G. Sullivan, "System Diagnosis," in *Fault-Tolerant Computer System Design*, ed. D.K. Pradhan, Prentice-Hall, 1996.

[7] A.T. Dahbura, "System-level diagnosis: A perspective for the third decade," in *Concurrent Computations*, Springer, 1989.

[8] E.P. Duarte Jr., and T. Nanya, "A Hierarchical Adaptive Distributed System-Level Diagnosis Algorithm," *IEEE Transactions on Computers,* Vol. 47, pp. 34-45, 1998.

[9] E.P. Duarte Jr., R.P. Ziwich, and L.C.P. Albini, "A Survey of Comparison-Based System-Level Diagnosis," *ACM Computing Surveys*, Vol. 43, pp. 1-56, 2011.

[10] L. Lamport, "The Weak Byzantine Generals Problem," *Journal of the ACM*, Vol. 30, pp. 668-676, 1983.

[11] E.P. Duarte Jr., L.C.P. Albini, A. Brawerman, and A.L.P, Guedes, "A hierarquical distributed fault diagnosis algorithm based on clusters with Detours," *The 6th IEEE Latin American Network Operations and Management Symposium*, pp. 1-6, 2009.

[12] 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.

[13] E.P. Duarte Jr., A. Brawerman, and L.C.P. Albini, "An algorithm for distributed hierarchical diagnosis of dynamic fault and repair events," *IEEE International Symposium on Cluster Computing and the Grid*, pp. 299-306, 2000.

[14] L.C.E. Bona, K.V.O. Fonseca, E.P. Duarte Jr., and S.L.V. Mello, "HyperBone: A Scalable Overlay Network Based on a Virtual Hypercube," *IEEE International Symposium on Cluster Computing and the Grid*, pp. 1025-1030, 2008.

[15] C. Feng and L.N. Bhuyan, and F. Lombardi, "Adaptive system-level diagnosis for hypercube multiprocessors," *IEEE Transactions on Computers*, Vol. 45, pp. 1157-1170, 1996.

[16] T. Ye, and S. Hsieh. "Fault diagnosis for hypercube-like networks," *The 2nd International Conference on Applied Informatics and Computing Theory*, pp. 205-209, 2011.

[17] M. Elhadef, and A. Nayak, "Comparison-Based System-Level Fault Diagnosis: A Neural Network Approach," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 23, pp. 1047-1059, 2012.

[18] P.M. Khilar, and S. Mahapatra, "A Novel Hierarchical Clustering Approach for Diagnosing Large-Scale Wireless Adhoc Systems," *International Journal of Computers and Applications*, Vol. 31, pp. 260-267, 2009.

[19] T. Ye, and S. Hsieh, "A scalable comparison-based diagnosis algorithm for hypercube-like networks," *IEEE Transactions on Reliability*, in-press.

[20] E.P. Duarte Jr., L.C.E. Bona, "A Dependable SNMP-based Tool for Distributed Network Management," *The IEEE/IFIP International Conference on Dependable Systems and Networks*, *Dependable Computing and Communications Symposium*, pp. 279-284, 2002.

[21] D. Harrington, R. Presuhn, and B. Wijnen, "Request for Comments: 3411," 2002.

[22] G. Khanna, M.Y. Cheng, P. Varadharajan, S. Bagchi, M.P. Correia, and P.J. Verissimo, "Automated Rule-Based Diagnosis Through a Distributed Monitor System," *IEEE Transactions on Dependable and Secure Computing*, Vol. 4, pp. 266,279, 2007.

[23] G. Khanna, I. Laguna, F.A. Arshad, and S. Bagchi, "Distributed Diagnosis of Failures in a Three Tier E-Commerce System," *The 26th IEEE International Symposium on Reliable Distributed Systems*, pp. 185-198, 2007.

[24] P. Chandrapal, and P. Kumar, "A Scalable Multi-level Distributed System-Level Diagnosis," *Distributed Computing and Internet Technology, Lecture Notes in Computer Science*, Vol. 3816, pp. 192-202, 2005.

[25] R. Su, W.M. Wonham, "Hierarchical Fault Diagnosis for Discrete-Event Systems under Global Consistency," *Discrete Event Dynamic Systems*, Vol. 16, pp. 39-70, 2006.

[26] T. Fukuta, H. Masuyama, and T. Sasama, "Hierarchical System-Level Diagnosis Based on a Voting Scheme for Identical-Unit-Interconnection Systems," *Sixth International Conference on Networking*, pp. 34, 2007.

[27] H. Masuyama, and T. Miyoshi, "A Non-Adaptive Distributed System-Level Diagnosis Method for Computer Networks," *The 1st International Conference on E-Business and Telecommunication Networks*, pp. 161-168, 2003.

[28] D. Phelps, M. Su, and K. Thulasiraman, "Distributed testing and diagnosis in a mobile computing environment," *The 6th International Wireless Communications and Mobile Computing Conference*, pp. 1268-1272, 2010.