A Scalable On-Line Multilevel Distributed Network Fault Detection/Monitoring System Based on the SNMP Protocol

Ming-Shan Su
Dept. of Computer Science and Technology
Southeastern Oklahoma State University
Durant, OK 74701-0609, USA
msu@sosu.edu

K. Thulasiraman
School of Computer Science
University of Oklahoma
Norman, OK 73019-0631, USA
thulasi@ou.edu

Anindya Das
Dept. of Electrical and Computer Engineering
University of Western Ontario
London, Ontario, Canada N6A 5B9
adas@eng.uwo.ca

Abstract-**Traditional centralized network management solutions do not scale to present-day large-scale computer/communication networks. They suffer from certain other drawbacks too: a single point of failure and hence lack of fault tolerance, and heavy communication costs associated with the central manager. It has been recognized that decentralization/distributed solutions can solve some of the problems associated with centralized solutions [3]. For this reason, there has been considerable interest in the recent past on distributed/decentralized network management applications. Our work in this paper has been motivated by this research trend in network management. We present the design and evaluation of an SNMP-based distributed network fault detection/monitoring system. The design involves the integration of our recently developed ML-ADSD algorithm [11], [16] for diagnosis of faults in a distributed system of processors into the SNMP framework. The ML-ADSD algorithm uses the multilevel paradigm and is scalable in the sense that only minor modifications will be required to adapt the algorithm to networks of varying sizes. The system allows processors to fail and/or recover during the process of diagnosis. Thus the system has fault tolerance capability. We demonstrate the application of the system by implementing it on an Ethernet network of 32 machines. Our results establish that the diagnosis latency (or time to termination) is much better than the latency of earlier solutions. Also, the bandwidth utilization of our system is very insignificant, thereby demonstrating the practicality of deployment of the system in a real network environment. Thus in this work we have successfully integrated three modern disciplines: network management, distributed computing and system level diagnosis.**

I. Introduction

As computer networks have grown into complex, enterprise-wide systems, the management of operations and their associated risks has gained increased urgency. The aim of network management is to manage (monitor and control) the network devices over a computer network through the exchange of messages between devices and hopefully to maintain the health of the network through such a framework.

Typically, in these manager-agent approaches, one node (device) is designated as a central network management station (NMS) or manager and the rest of the nodes as regular nodes (Agent). The manager periodically polls the agents and then uses the data collected to locate fault conditions or take preventive measures. If the network size increases, then one or more nodes are designated as NMS nodes with dual roles - manager and agent – and to balance loads and to avoid traffic congestion at the central network manager. Those new managers then poll information from agents under their respective domains and the central network manager polls information from those new managers.

Current Network Management applications are largely based on the Simple Network Management Protocol (SNMP). SNMP was developed by IETF in 1988 for the purpose of managing heterogeneous network devices over a TCP/IP-based computer network and has been widely adopted by industry on network applications [1], [2].

The SNMP protocol is a request-and-response protocol. It defines the functions of the basic operations of SNMP and the format of the messages exchanged by management systems and agents. The features of SNMP include its standardization, universal support, extensibility, portability, and simplicity.

Operational data collected by agents is kept in the Management Information Base (MIB). The MIB represents the management information to be monitored and controlled in a TCP/IP-based network.

The drawbacks of centralized approaches to network management include a single point of failure, lack of scalability, and high communication costs associated with the central manager [3]. If a manager fails, node information associated with its domain is lost because there is no automatic substitution mechanism provided for by the network management system. Some decentralized approaches that resolve many of these issues such as management by delegation [4], active networks [5], and mobile agents [6] have been proposed. Other related works of interest to the work in this paper may be found in [7], [8] and [9].

In this paper we propose a distributed SNMP-based solution to the fault detection/monitoring problem. The paper is organized as follows.

In section II we provide an overview of the area of system-level diagnosis emphasizing the need for distributed diagnosis [10]. In section III we provide a description of our multilevel distributed diagnosis algorithm called the ML-ADSD algorithm presented in [11]. In section IV we describe the essential features of a distributed network fault detection/monitoring system based on the SNMP protocol and report results of our extensive experiments obtained by implementing the fault detection/monitoring system on a network of 32 machines. We conclude, in section V, with some observations and pointing to certain applications of this work in other areas besides network fault monitoring.

II. System-Level Diagnosis: An Overview

Rapid advances in semiconductor technology have made possible the development of very large digital systems comprising hundreds of thousands of components or units. Yet it is impossible to build such systems without defects. As the size of a system grows, it is more likely to develop faults both in the manufacturing process and during the operation period. Testing of such systems becomes extremely difficult due to their large sizes. This problem may be further aggravated by possible geographical distribution of units.

In 1967, Preparata, Metze and Chien [10] proposed a model called the PMC model and a framework, called *System-Level Diagnosis*, for dealing with this problem. The PMC model allows faulty processors also to test. Thus some of the test results could be unreliable. This makes the problem of diagnosis challenging and interesting. In the more than three decades following this pioneering work, several issues arising from the application of this framework have been investigated and resolved. Many of these results have profound theoretical and practical implications. Most of the recent research efforts in system-level diagnosis have focused on enhancing the applicability of system-level diagnosis based approaches to practical scenarios. One such application is: On-line distributed diagnosis and application to diagnosis of a networked cluster of machines. Our work in this paper is concerned with the latter, namely, on-line distributed diagnosis.

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 [12] to pioneer the area of distributed system-level diagnosis. Major algorithmic contributions in this area include those by Bianchini and Buskens [13], Rangarajan and Dahbura [14] and Duarte and Nanya [15]. While the algorithms in [13] and [15] assume the existence of a logical fully connected network, the algorithm in [14] is applicable to networks of arbitrary topologies. The diagnosis latency of Bianchini and Buskens' algorithm (called the ADSD algorithm) is $O(N)$ testing rounds whereas the diagnosis latency of the Hi-ADSD algorithm of Duarte and Nanya is $O(log^2N)$ testing rounds. Here, N is the number of processors (machines) in the network and diagnosis latency refers to the time taken for all fault free nodes to reach a consensus view of the fault/free-free status of the whole network. It is important to stress the fact that processors could fail and/or recover during the process of diagnosis. So a fault event could be either a failure or recovery from a failure.

III. Multi-Level Adaptive Distributed Diagnosis For Fault Detection

In this section we briefly describe the design of a prototype multi-level adaptive distributed diagnosis algorithm for fully connected networks, which generalizes the ADSD algorithm of Bianchini and Buskens. This algorithm, called the ML-ADSD algorithm, has been motivated by the need to reduce the diagnosis latency of the ADSD algorithm as well as the message transmission overhead. Reduction of the diagnosis latency was also the key motivation of Duarte and Nanya's [15] hierarchical distributed diagnosis algorithm. Complete details of the ML-ADSD algorithm may be found in [11] and [16].

The main features of the ML-ADSD algorithm are: on-line, multi-level divide-and-conquer partition strategy; adaptive on the next testing assignment; distributed on execution; no upper bound on the number of faulty nodes; autonomous leader election; easy control on synchronization; and less message transmission overhead. Also, as noted earlier processors could fail and/or recover during the process of diagnosis. So a fault event refers to a processor failure or recovery from failure.

In the algorithm, we assume a logically complete network G of N nodes, n_0, n_1,..., n_{N-1}. The nodes are first listed in sequential order and partitioned into p clusters of equal size. Thus each cluster has (N/p) nodes. Also it is assumed that each node is able to correctly test and determine the status of other nodes based on the PMC fault model. Link failures are not permitted

A pictorial tree description of this algorithm is shown in Figure 1. To simplify the presentation we assume that each cluster at level 1 (the original p clusters) has at least one fault free node. At level 1 of the tree representation are the p original clusters. At level 2 there are $p/2$ clusters. Each

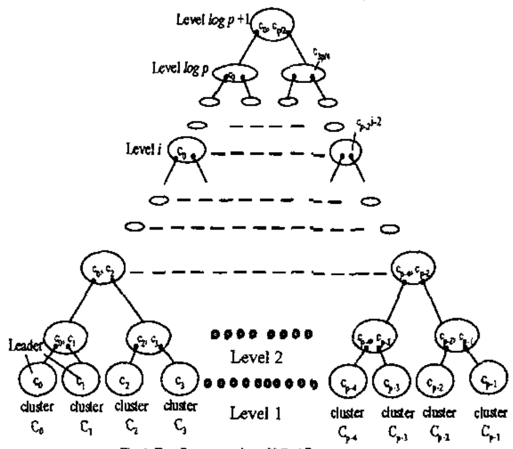

Fig. 1. Tree Representation of ML-ADSD Algorithm

level-2 cluster consists of at most 2 nodes, which are leaders of two 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.

Basically the algorithm proceeds as follows. Nodes execute testing rounds as in Biancinni and Buskens' algorithm [13]. They begin at level 1. After N/p testing rounds a leader will be determined for each cluster where p is the number of clusters defined by the user. These cluster leaders proceed to higher levels in subsequent testing rounds, exchange information with other leaders and collect status information of nodes in other clusters. When the leaders c_0 and $c_{p/2}$ reach the last level as in Fig. 1, they exchange information about the fault status information of all the nodes in the network. In subsequent testing rounds this information is transmitted to other leaders in the network. The information is eventually delivered to nodes in the original clusters through their leaders. Note that testing rounds are also performed by faulty nodes. This is what makes the proof of correctness challenging and interesting. We have proved in [11] and [16] the correctness of the algorithm and established the following results regarding diagnosis latency after the occurrence of the last fault event.

Theorem 1:

i) If the number of levels $M = \log p + 1$, the diagnosis latency of the M-Level ML-ADSD algorithm is at most $2\,N/p + (\log p + 4)\log p$ testing rounds.
ii) If the number of levels $M > 2$, then the diagnosis latency of the ML-ADSD algorithm is at most $2\,N/p + (M\text{-}2)(M\text{+}4) + p\,2^{(-M+3)} + 1$ testing rounds.

We have conducted extensive experimental evaluation of our algorithm as well as those in [13] and [15] and demonstrated the superiority of our ML-ADSD algorithm over the ADSD and the Hi-ADSD algorithms. Simulation results indicate that in all cases, the ML-ADSD algorithm is much better than the ADSD algorithm. In addition, the performance of the ML-ADSD algorithm can be tuned/improved, depending on the needs, by an appropriate choice of the number of clusters and the number of levels. The ML-ADSD algorithm is scalable in the sense that only some minor modifications will be required to adapt the algorithm to networks of varying sizes [16].

IV. SNMP-Based Implementation, Experimental Results And Discussion

In this section we present application of the ML-ADSD algorithm in the design of an SNMP-based LAN fault detection/monitoring system.

(a) SNMP-Based Implementation and Experimental Set Up: We implemented the fault detection/monitoring system on a 10/100 Mbps Ethernet LAN consisting of 32 machines (Pentium 100/133/166 MHz) running Microsoft Windows NT operating systems.

In the ML-ADSD algorithm, a node uses a two-dimensional array called TESTED_UP to update the testing results of the network. Based on SMI [1], we built the TestedUpMIB module for mapping the TESTED_UP array diagnosis information in the ML-ADSD algorithm.

Three major tasks are required to implement the management system at each node: *generating the MIB (TestedUpMIB), coding and setting up the Extension Agent (ML_ADSDAgt.dll), and coding and setting up the Manager (ML_ADSDMgr.exe).* As mentioned earlier, a node may have dual roles, as a network management station (NMS or manager) as well as a managed node (agent). In our implementation, we set up each machine to have this property too. We partitioned the 32 machines into 8 clusters of 4 machines each and we have used the 2-level diagnosis scheme of section III in the design of the fault detection system. The experimental set up is shown in Fig. 2.

Since robustness and diagnosis latency of the ML-ADSD algorithm are our main focus, we tested the fault detection system extensively with different combinations of parameter settings.

In our experiments we used two parameters: waiting time (in seconds) between testing intervals and the maximum number of faults during each run of an experiment An experiment corresponds to a setting of the above parameters. Each experiment consists of 20 runs of the diagnosis algorithm with 20 randomly generated fault events injected during a run. The faults were injected by a tester. In our experiments, we have used the fail-stop model. In this model a failed machine is considered dead and so does not respond to any test message.

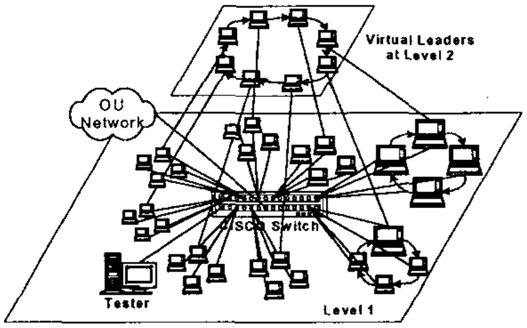

Fig. 2 Experimental setup

(b) Diagnosis Latency: Recall that diagnosis latency refers to the time taken for all fault free nodes to reach a consensus view of the fault/free-free status of the whole network. The

diagnosis latency is recorded after the last fault event (failure or recovery) in each run. The average diagnosis latency over the 20 runs of an experiment gives the average diagnosis latency for that experiment. The results of 42 experiments are reported in Table 1.

Since ML-ADSD is a generalization of the ADSD algorithm, we also implemented and tested the ADSD algorithm on similar settings. The ADSD implementation results are shown in Table 2. The comparison of diagnosis latency (average of latencies for different settings) is shown in Fig. 3.

Simulation results for network sizes from 32 to 1024 nodes on ML-ADSD, ADSD, and Hi-ADSD can be found in [11] and [16]. From these simulation results and the experimental results in Tables 1 and 2, we have found that in all cases, the diagnosis latency of the ML-ADSD algorithm is much better (smaller) than the latency of the ADSD algorithm. Also, from Table 1 we can see that in general the more faults in the system, the less is the time for all the fault-free nodes to diagnose those faults.

(c) Bandwidth Utilization: Having shown the feasibility of implementing our work in a real network, we next proceeded to measure the bandwidth utilization for various settings by running the ML-ADSD algorithm. For this we used a CISCO Catalyst 2948G Switch, with 48 10/100TX(RJ-45) ports, to connect to all the 33 machines (including the tester) as in Fig. 2. Since all the machines are running in a distributed and independent manner, a tester is used to inject the events (faulty or recovery). Every two seconds the tester also compares the TestedUpMIB in each fault free machine after the last fault event, calculates and records the diagnosis latency when all the fault free machines have reached a consensus on the fault status of all the machines in the network. Note that a tester is not required in the real management system and we used it here just for experimental purpose.

We measured the bandwidth utilization in bits per second on each machine based on every 3 minutes granularity. We have found that the bandwidth required by the algorithm is very small. Table 3 shows bandwidth utilization in bits per second under the no-faults condition. The bits generated by the tester are not included. This is not significantly different from the utilization with faults.

For instance, in the setting which corresponds to 4 seconds waiting time and at most 4 faults, the average bandwidth utilization on each machine is 4,619 bits per second (including the background traffic) and 5,091 bits per second for the leader machines running the ML-ADSD algorithm. The background traffic on average is 2,492 bits per second and consists of the traffic generated by the tester, some unicast, non-unicast, multicast, and broadcast packets. However, the background traffic will fluctuate a little bit daily. In other words, the average bandwidth utilization per machine is only 0.026% or 0.0026% on a 10/100 Mbps Ethernet network and this shows the practicability of running ML-ADSD over a real network.

Though our system is feasible for deployment in a real network environment, in a completely shared media WAN such as those using multi-point frame relays the aggregate traffic of about 160 Kilo bits per second generated by the ML-ADSD algorithm may impact the performance of networks with low bandwidth capacity. However, such networks are becoming outdated and are being replaced by point-to-point frame relay networks.

Table 1. The diagnosis latency (in seconds) of two-level ML-ADSD

ML-ADSD		Faults 1	4	5	6	7	8	Avg
Time between test intervals	2	25.80	28.70	26.40	27.20	24.80	25.80	26.45
	3	37.20	38.10	36.50	37.50	38.80	36.30	37.40
	4	43.40	44.90	47.60	46.55	46.40	44.15	45.50
	5	66.85	61.90	61.80	57.80	58.85	59.85	61.18
	6	72.85	68.35	69.45	66.95	65.25	62.35	67.53
	7	77.15	75.55	72.25	81.65	75.55	73.00	75.86
	8	92.30	93.10	84.35	84.90	87.90	81.55	87.35

Table 2. The diagnosis latency (in seconds) of ADSD algorithm

ADSD		Faults 1	4	5	6	7	8	Avg
Time between test intervals	2	39.95	36.70	35.90	34.50	34.45	31.25	35.46
	3	57.00	53.55	51.80	49.90	47.55	45.75	50.93
	4	66.40	65.45	64.15	60.15	59.05	59.05	62.38
	5	106.80	95.95	94.50	86.95	86.55	81.20	91.99
	6	119.15	103.55	98.40	94.55	93.70	88.70	99.68
	7	123.45	115.95	112.90	109.10	103.85	103.15	111.40
	8	141.85	127.05	125.85	116.70	113.00	115.35	123.30

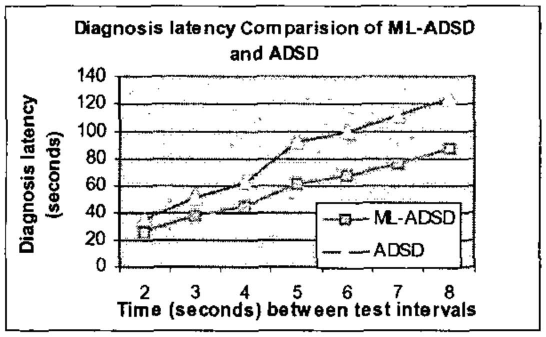

Fig 3. Diagnosis latency (in seconds) comparison of ML-ADSD and ADSD algorithms.

Table 3. The Bandwidth utilization (in bits per second) of ML-ADSD and ADSD (with no faults and tester's packets excluded)

Waiting Time	2s	3s	4s	5s	6s	7s	8s
ML-ADSD	5793.21	4942.09	4073.45	3721.23	3579.03	3364.77	3243.82
ADSD	5479.87	4596.48	3998.64	3684.47	3490.13	3381.45	3359.65

V. Summary

Centralized approaches to network management suffer from certain major shortcomings: lack of scalability, single point of failure and hence lack of fault tolerance and excessive communication overhead around the central manager. Use of decentralization/distributed solutions can overcome some of these problems. For this reason decentralized approaches for network management applications have been discussed in the literature [3 – 9].

Following this research trend in network management we have presented in this paper the design and evaluation of a prototype SNMP-based distributed network fault detection/monitoring system. The design involves the integration of our recently developed ML-ADSD algorithm [11], [16] for diagnosis of faults in a distributed system of processors into the SNMP framework. The ML-ADSD algorithm uses the multilevel paradigm and is scalable in the sense that only minor modifications [16] will be required to adapt the algorithm to networks of varying sizes. Furthermore, the performance of the ML-ADSD algorithm can be tuned/improved, depending on the needs, by an appropriate choice of the number of clusters and the number of levels. Our distributed network fault detection/monitoring system allows processors to fail and/or recover during the process of diagnosis. Thus the system has fault tolerance capability. We demonstrated the application of the system by implementing it on a Ethernet network of 32 machines. Our results establish that the diagnosis latency (or time to termination) is much better than the latency of earlier solutions. Also, the bandwidth utilization of our system is very insignificant, thereby demonstrating the practicality of deployment of the system in a real network environment. Thus in this work we have successfully integrated three modern disciplines: network management, distributed computing and system-level diagnosis.

We now point out certain potential applications of our work. In a mobile environment, when a node a loses communication with another node b because of mobility, node a can interpret this as failure of node b. With this interpretation of failure, it appears that we can use the algorithm of [14] to study the location management problem in a mobile environment. But in such an extension, an important difference needs to be taken into account: node b, while viewed by node a as a failed node, may appear as a recovered node to another node c which has gained communication with node b because of mobility. This important but challenging modification to distributed diagnosis offers opportunities for further extension of our work. For related works see [17], [18].

Our work can also be extended and applied in the distributed system load balancing area. For example, while the ML-ADSD algorithm is doing fault detection it can also do performance management. For instance, it can carry other system MIB variable information (e.g., CPU usage, Hard disk/Memory space usage, User logging data, etc.) at the same time, which can be used to balance load on different machines in a network.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support and facilities provided by the school of Computer Science and the IT department at the University of Oklahoma in carrying out our experiments. In particular the authors thank CS Director Dr. John Antonio and IT staff Mr. Nate Wahlmeier for helpful discussions and suggestions.

REFERENCES

[1] M. Rose and K. McCloghrie, "Structure and identification of management information for TCP/IP-based internets," RFC 1155, 1990.

[2] M. Rose, *The Simple Book: An Introduction to Internet Management*, 2nd edition, PTR Prentice-Hall, Inc, 1994.

[3] G. Goldszmidt and Y. Yemini, "Decentralizing control and intelligence in network management", *Proc. the 4th Int'l Symp. on Integrated Network Management*, Santa Barbara (1995).

[4] G. Goldszmidt and Y. Yemini, "Distributed management by delegation", *Int'l Conf. on Distributed Computing Systems*, 1995.

[5] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden. "A survey of active network research", *IEEE Communications Magazine*, vol. 35, no. 1, January 1997, pp. 80-86.

[6] A. Bieszcad, B. Pagurek, and T. White, "Mobile agents for network management", *IEEE Communications Surveys*, 4Q (1998).

[7] J. Martin-Flatin and S. Znaty, "Annotated typology of distributed network management paradigms", *Technical Report* SSC/1997/008, Ecole Polytechnique Fdrale de Lausanne (1997).

[8] R. Subramanyan, J. Miguel-Alonso, and J.A.B. Fortes. "A scalable SNMP-based distributed monitoring system for heterogeneous network computing". *SC2000*. Dallas, Texas, Nov. 2000.

[9] P. Simões, L. M. Silva, and F. Boavida, "Integrating SNMP into a mobile agent infrastructure", *Distributed Systems, Operations and Management (DSOM)*, Zurich, Switzerland, 1999, pp. 148-163.

[10] F. P. Preparata, G. Metze, and R. T. Chien, "On the connection assignment problem of diagnosable systems", *IEEE Trans. on Electronic Computers*, vol. ec-16, no. 6, Dec. 1967, pp. 848-854.

[11] 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.

[12] J.G. Kuhl and S. M. Reddy, "Fault-diagnosis in fully distributed systems", *Proc. the 11th Int'l Symp. on Fault Tolerant Computing*, 1981, pp. 100-105.

[13] R. Bianchini and R. Buskens, "Implementation of on-line distributed system-level diagnosis theory", *IEEE Trans. on Computers*, vol. 41, no. 5, May 1992, pp. 616-626.

[14] S. Rangarajan, A. Dahbura and E. Ziegler, "A distributed system-level diagnosis algorithm for arbitrary network topologies", *IEEE Trans. on Computers*, vol. 44, 1995, pp. 312-333.

[15] E. Duarte Jr. and T. Nanya, "A hierarchical adaptive distributed system-level diagnosis algorithm", *IEEE Trans. on Computers*, vol. 47, no. 1, Jan. 1998, pp. 34-45.

[16] 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.

[17] Y. Breitbart, M. Garofalakis, C. Martin, R. Rastogi, S. Seshadri et al, "Topology discovery in heterogeneous IP networks", *Proc. IEEE INFOCOM*, 2000, pp 265-274.

[18] W. Theilmann and K. Rothermel, "Dynamic distance maps of the internet", *Proc. IEEE INFOCOM*, 2000, pp. 275-284.