VINES: Distributed Algorithms for a Web-Based Distributed Network Management System

Glenn Mansfield[1], Elias Procópio Duarte Jr.[2], Masako Kitahashi[3] and Shoichi Noguchi[4]

[1] Cyber Research Inc., Sendai, Japan
[2] Tokyo Institute of Technology, Tokyo, Japan &
Federal University of Paraná, Curitiba, Brazil
[3] (ex) Fuji-Xerox Information Systems, Tokyo, Japan
[4] Nihon University, Koriyama, Japan

E-mail: glenn@tia.ad.jp, elias@inf.ufpr.br

Abstract. Rapid growth in networks and related services have resulted in demands for more effective, accessible, intelligent and scalable network management systems. Not only managers but also users need access to network management information. Traditional network management architectures are utterly inadequate to meet these requirements. In this work we describe VINES - a next generation network management system that builds on state-of-the-art technolgy and latest results. VINES uses a new web-based management architecture, that is intelligent and distributed. A set of *Management Domain Servers*, implementing an elaborate access control mechanism, allows access to management information without compromising security. To realize the new architecture a new knowledge base framework is designed, highly effective detection and diagnosis algorithms are employed, and the resilience of the management system to network failures is ensured. The results of an experimental prototype are discussed.

1 Introduction

The Internet has undergone a sea-change in the past few years. Its scope has rapidly expanded from universities, laboratories, and computer related industries to schools, businesses, homes and the man on the street. Network Operations and Management technology requires some radical changes to keep pace with the developments. In this work we describe VINES, a distributed network management system that utilises new concepts and algorithms based on state-of-the-art technology to solve some of the problems encountered in the area of network management.

The major constraints of present network management systems are as follows:

- The management architectures are non-hierarchical and essentially centralized. The access to management information is generally closed and confined to users of Network Management Stations (NMSs) in the sense that

users without appropriate management tools cannot access management information [even if it was made accessible]. Of course, "open"ness cannot be achieved without putting in place effective access control and security mechanisms.
- The Management systems are generally inflexible. The look, feel and functions get fixed at the design stage. Users of the system have little or no control over the design of the system or of adapting the system to their local/personal requirements/likings.
- As in any management, in Network Management the quality and effectiveness of management is directly related to the quality of information made available to the management systems. Present management systems in general suffer from the lack of access to an important component of information, i.e. the Network Configuration Information. Generally, this information component is not used or if it is used, generally it is of dubious content.
- Though present network management systems in general provide access to management information in the network, there is a serious lack of algorithms to process and add value to this information. Without any added value or insight provided - an ordinary user can make very little sense out of the management information.
- Network Management Systems are mission critical. Yet in general they succumb to the very faults they are supposed to be watching out for. To serve its purpose, a network management system needs to be fault-tolerant.

There have been approaches to solve some of these problems. AIMS[1] has firmly established network configuration information as an indispensable component of management information. The object oriented approach of IPANEMA[2] has laid the ground for flexibilty in user-system design. New developments in networking particularly the web and platform independent applications have opened up new avenues in "open" network management. A substantial amount of work has been done on algorithms for effective network management and useful results have been published [4, 10]. In this work we describe VINES - a next generation network management system that builds on state-of-the-art technolgy and latest results. We discuss the results that have been obtained by using the experimental prototype.

In VINES, the **V**ersat**I**le **NE**twork Management **S**ystem, the sources of information, the services and the algorithms are distributed. It also uses effective information and algorithms to achieve intelligence, accuracy, and reliability. In VINES, mid-level managers are introduced [there may be more than one mid-level manager serving the same management domain(MD)] to effect distributed management. The "open"ness is achieved using platform-independent (Java-like) programming technology to make the tools accessible along with the data (information).

The rest of the paper is organized as follows. In section 2 a brief outline of VINES is presented. The new information components made available to VINES are discussed in section 3. The intelligent algorithms used are discussed in section 4, followed by a discussion on the distributed system-level diagnosis algorithms

in section 5. In section 6 we discuss the parallels with the self-defense mechanism in biological systems that forms the basis of the distributed and fault-tolerant fault management paradigm in VINES, followed by a short presentation on the status of the project in section 7.

2 A brief outline of VINES

2.1 The VINES Object Model

VINES has effectively used the object oriented approach to offer the user desired design-flexibility. In the VINES object model a *network* is the media for transmitting information. The media may be a line (physical circuit/virtual circuit), a coaxial-cable, ether, or a collection of interconnected networks. *Network elements* are equipment with one or more *network interfaces* whereby it is possible to exchange information with the network. *Network elements* with multiple interfaces e.g. gateways, routers, bridges/repeaters etc. maybe used to *connect* networks. Thus *networks* may have zero or more *(sub)-networks* and *network elements*, each having at least one interface on the media. A *network-object* represents a network in VINES.

Besides communication properties, a *network-object* has management properties. The management properties determine how (the user wishes) the object will be managed. It includes, management information pertaining to the network object, and the decision procedures which will use the management information and the services related to the management information this includes graphical display of the information.

2.2 The Management Architecture of VINES

Management Domain (MD) The user (Network Manager) can conceptually carve out his/her target management domain (MD) from an existing network. Automated procedures aid the user in this activity by generating the required configuration information. The user may also customize the management system by (re-) defining the properties of the network objects in the management domain (MD).

Management Domain Server (DS) The management information of a MD (the union of all the MIBs in all the network elements in the MD) is serviced by one or more MD-servers (DS). These DSs serve as the mid-level between the clients (network managers) and the network elements.

Each DS serves a particular *image* of the MD. For example in fig. 2 the multiprotocol MD consisting of three network elements is separated into images MD1 and MD2 corresponding to protocols P1 and P2 respectively. The provision of multiple images allows distribution of the management service load. A DS is given the definition of the MD it is serving and a list of services it will provide to clients.

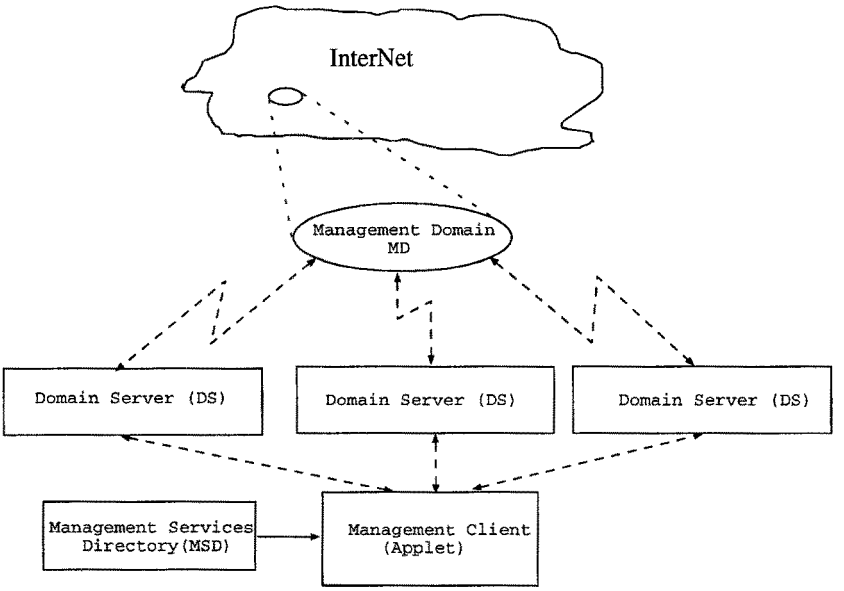


Fig. 1. The VINES Management Architecture

Management Services Directory In the distributed autonomous management architecture of VINES the Management Services Directory (MSD) provides the glue necessary to hold the distributed architecture together. The MSD contains a list of the DSs serving a MD and the list of services provided by each DS. An example of a service is the graph of the number of input packets at a particular interface of a particular host. The MSD maps a user requirement for a particular service related to a particular set of MOs on a particular Network element, to a DS.

VINES Clients In VINES a user attempts to manage the network (to get management information about a MD) via a client. A client is a simple application (*applet*) that is downloaded from the network. At startup the client will contact a Management Service Directory to find out which DS is the most appropriate to service the user request. The client will then contact the DS which in turn accesses the MIBs of the agents in the NMSs and provides the requested service - if the access is permitted.

Security & Access Control The *Distributed* and *Open* architecture of VINES makes it imperative to have an effective security and access control mechanism in place.

In VINES two levels of control are enforced. First the access privelege to the management information service is verified. There are three levels of authentications:

- **Strongly authenticated:** This is the authentication level required for Super-User/Administrator priveleges and implies that the user has been authenticated beyond doubt.
- **Weakly authenticated:** The user is known to be *known* e.g. the user has a verifiable identification but is not known or registered locally.
- **Unauthenticated:** The user is without any identification.

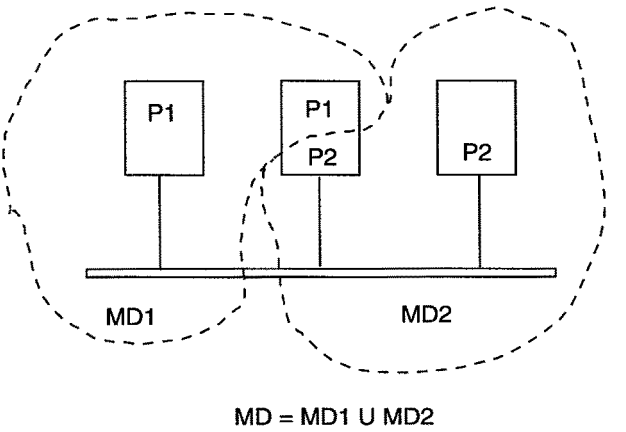


Fig. 2. Images in VINES

For every level of authentication a set of communities or parties will be assigned depending on the level of trust and reliability required for the corresponding management action.

At the second level the access to the managemet information is verified by using the *community* scheme in SNMP or the *party* scheme in SNMPv2.

Thus if the service access control check for the management service and the information access control check are both positive, then the corresponding service and information is made available to the client.

In the VINES architecture the DS effectively shields the network elements from public access for management information retrieval purposes, thereby bolstering the security of the network. DSs may be split or merged to form new Management Domains. Furthermore, multiple DSs may serve the same MD. This ensures the scalability of the management architecture in the global perspective.

3 Information Components

In VINES, the management information sources and services are distributed among the DSs. Each DS caters to a part of the MIB-tree and related services. This makes the management system rugged and resilient to faults.

Information pertaining to dynamically varying MOs is fetched periodically from the network. Quasi-static information components are refreshed at longer intervals or on user demand.

Among the information components, Network Configuration Information is an essential element for effective network planning, administration and managment. It provides the basic information that will be used to create the network map. The map is necessary for an overview of the physical/logical network topology, to serve as an important input for management and administration of routes, traffic flow, network problems and policy and, as a pool of information for the public. Apart from interconnectivity information the other desirable information elements are policy related information, network name and address related information, network administration and management related information.

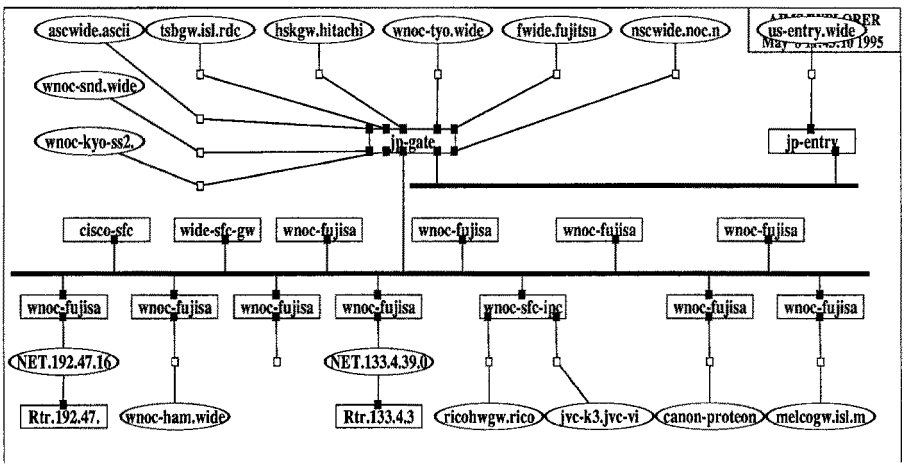


Fig. 3. Automatically synthesised network map.

Gathering and maintaining up-to-date network configuration database is a challenging task which has been nicely tackled in VINES. Network-layer entities have the requisite configuration information which enables them to route

the packets around the network. Therefore, network-level entities are the primary source of network configuration information. To extract network configuration information, VINES employs standard protocols and techniques. The Internet Standard Simple Network Management Protocol(SNMP)[5] is used to fetch configuration information from the various standard Management Information Bases. Fig. 3 shows a map that is automatically synthesised from the network in the VINES framework. Other sources of information about network elements, routing, policy and administration are the *Routing Arbiter Database* (RADB)[6], RWHOIS[7], WHOIS[8], DNS[9].

4 Algorithms

4.1 The Management Information Knowledge Base (MIKB)

To add-value to the management information accessed from the network, VINES employs an extension of the Management Information Base (MIB). The MIB essentially contains the syntactic definition of Managed Objects (MOs). In VINES a Management Information Knowledge Base (MIKB) is used. The MIKB contains the semantic image of the MIB. Essentially it describes the significance of the value of a MO, as seen by a knowledgable person. E.g. if the value of an ifOperStatus (interface operational status) MO is 1 it *means* the corresponding interface is **operational** whereas if the value is 2 it *means* that the interface is **down**. The MIKB is used to detect symptoms of abnormalities in a network. By defining the MIKBs corresponding to all the standard MIBs, the built-in capability of locally *watching* out for fault symptoms is realised in VINES.

4.2 The Divide & Conquer Algorithm for Symptom Isolation

In general an NMS collects information about the network. From the perspective of fault management, this information will be analyzed for *symptoms* or indications of abnormal health. In VINES the MIKB is used to decide whether the status of a Managed Object (MO) or, the combined status of a set of MOs is indicative of abnormal health or otherwise [1]. For example, if the number of ICMP destination unreachable packets exceeds a threshold, say *alpha*, an *alarm* or an *event* may be triggered indicating an abnormality in the network and calling for action/intervention. More often than not, this event is the consolidated effect of several symptoms. Each ICMP destination unreachable packet indicates that some destination is unreachable and, as such, is a symptom of a potential network fault. It is the task of the NMS to detect the presence of symptoms, isolate and identify each symptom and, diagnose and control the fault indicated by the symptoms. Symptoms have their own respective characteristics. The *amplitude* of a symptom is a measure of the observable which will be thresholded to decide whether the observable represents an abnormality. The *persistence* of a symptom is a measure of the duration of the symptom and the *frequency* of a symptom is the number of times the symptom has been observed in a given period.

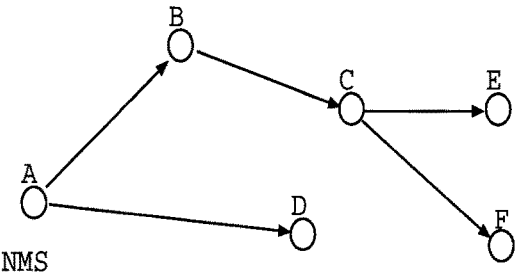
The existence of multiple faults in a network complicates the analysis of symptoms. Some faults and the corresponding symptoms are persistent. If there are persistent symptoms manifested in an aggregated MO, (e.g. ICMP destination unreachable packets), fresh symptoms are likely to get obscured. It is likely that new events are not triggered as the event is already *ON*. Dynamically adjusting the threshold to the "normal" state of the network is a non-trivial problem. Thus some or maybe most of the symptoms occurring in the presence of a persistent symptom are likely to remain hidden, unnoticed. In such situations, the simple thresholding mechanism is utterly inadequate to detect fresh abnormalities and set the corresponding alarms.

Symptoms of faults aggregate and are manifested in the aggregate traffic characteristcs generally observed by a traffic monitor. Isolation of symptoms manifested in the aggregate traffic characteristcs is a challenging task. Symptoms get obscured by other symptoms. At times there are too many symptoms clouding the symptom space, making the task of symptom isolation practically impossible. VINES employs a powerful technique, the *divide and conquer technique*[10], wherein symptoms are iteratively isolated from the aggregate observable. This provides a tractable mechanism for symptom isolation, fault detection and analysis. The symptom isolation technique makes it possible to use a simple thresholding mechanism for detecting abnormalities. It is implemented using the popular SNMP-based RMON technology. Using dynamically constructed filters to supress already detected symptoms in the observed aggregate, fresh symptoms are isolated. Experimental results show a significant improvement in the fault management capability and accuracy. The RMON technolgy permits distribution of the network fault symptom isolation technique.

4.3 Intelligent Algorithms Using Configuration Information

In VINES configuration information is used very effectively. In the following we give two examples.

- *Intelligent polling:* To monitor the operational status of the nodes in a network, generally one is required to poll all the network elements in the network. In VINES the network configuration information is used to compute *long paths*. The nodes at the end of *long paths* are polled. In fig. 4 the *long paths* are A-B-C-E, A-B-C-F and A-D. By polling the leaf nodes D, E and F an attempt is made to establish that all nodes in the network are operating. If there is a response then the inference is that all the nodes on the path are operational. Of course if the polling 'times out' - the inference is: one of the nodes along the path is not operational. The problem node is detected by shortening the path by one arc at a time.
- *Fault diagnosis* In general, faults cascade in a network. There is a sudden spurt in destination unreachable messages. In general, the reason is a faulty router that connects several networks. The network configuration information is used to detect such conditions. In fig. 4 if both nodes E and F are unreachable then there is good case to check the connecting node C.




```
LongPaths: A-B-C-E , A-B-C-F, A-D
```

Fig. 4. Configuration information for intelligent polling and diagnosis.

5 Distributed System-Level Diagnosis Algorithms

Network fault management systems are mission-critical, for they are most needed during periods when part of the network is faulty. Distributed system-level diagnosis offers a practical and theoretically sound solution for fault-tolerant fault monitoring. It guarantees that faults don't impair the fault management process.

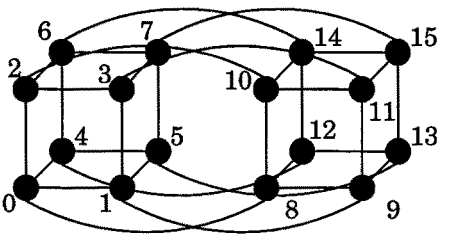


Fig. 5. Testing assignment of Hi-ADSD when 16 nodes are fault-free.

In VINES a new algorithm for distributed network fault management is employed. This algorithm is a combination of the Hierarchical Distributed System-Level Diagnosis (Hi-ADSD) algorithm for LAN fault detection [11] with the algorithm for diagnosis of non-broadcast networks described in [4]. We initially review each of these two algorithms and conclude presenting their combination, as employed by VINES.

Consider a system composed of N nodes that can be faulty or fault-free. The purpose of distributed system-level diagnosis is to have each fault-free node of the system determine the state of the other nodes. The Hierarchical Adaptive Dis-

tributed System-level Diagnosis (*Hi-ADSD*) algorithm is a fully distributed algorithm that allows every fault-free node in a fully connected network to achieve diagnosis in at most $(log_2 N)^2$ testing rounds. Nodes are mapped into progressively larger logical clusters, so that tests are run in a hierarchical fashion. When all nodes are fault-free the testing topology is like a hypercube, as shown in figure 6. Each node executes its tests independently of the other nodes, i.e. tests are run asynchronously. All the information that nodes exchange is diagnostic information. The algorithm assumes no link faults, a fully-connected network and imposes no bounds on the number of faults. The algorithm has been implemented using SNMP.

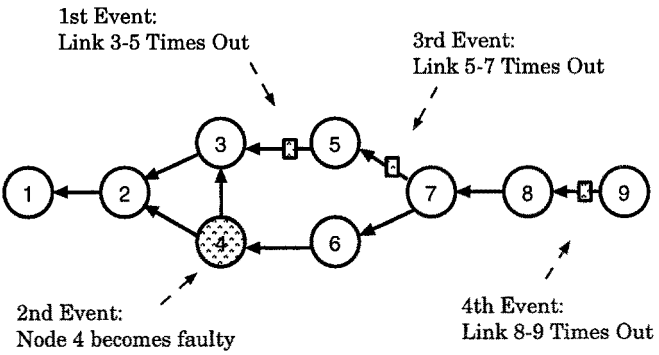


Fig. 6. A series of events occur in the network.

The algorithm presented in [4] for diagnosis of non-broadcast networks is necessary, as VINES is used to manage an internet, composed of a collection of networks and routers. In the algorithm, nodes test links periodically, and disseminate link time-out information to all its fault-free neighbors in parallel. Upon receiving link time-out information a node computes which portion of the network has become unreachable. This approach is closer to reality than previous algorithms, for it is impossible to distinguish a faulty node from a node to which all routes are faulty. The diagnosis latency of the algorithm is optimal, as nodes report events in parallel, and latency is proportional to the diameter of the network. The dissemination step includes mechanisms to reduce the number of redundant messages introduced by the parallel strategy. The algorithm is implemented within the SNMP framework.

Consider the example system in figure 6. Initially all links and nodes are fault-free. Each node starts testing links as depicted by arrows, and exchange test information with neighbors. Eventually each node receives information about all links. As the events depicted occur, some nodes detect that neighbors are unreachable, and information is disseminated to the whole network.

VINES uses an integrated approach for internet fault monitoring. This ap-

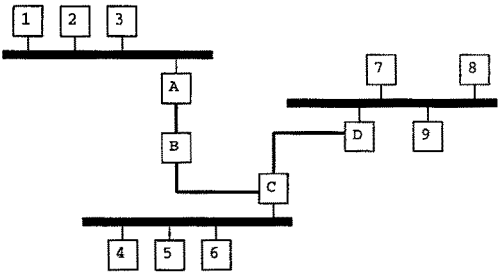


Fig. 7. A small internet.

proach is achieved by running Hi-ADSD for diagnosis on broadcast networks (LAN's), together with the non-broadcast network diagnosis algorithm. Consider the small internet in figure 7. Nodes with identifiers from 1 to 9 are connected to broadcast networks. Node A, node C, and node D have a link to a broadcast network, and to a non-broadcast network. Node B takes part only in the non-broadcast network. For the two algorithms to run cooperatively, it is sufficient that nodes only on a broadcast network run an algorithm for diagnosis on the LAN to which it belongs; nodes on a non-broadcast network run the algorithm for diagnosis on that network; nodes that are on a broadcast network, but also have a link to another network must run both a LAN diagnosis algorithm, and a WAN diagnosis algorithm. This means these nodes execute tests according to the two algorithms, and also carry the necessary data structures to hold information about the entire system. In this way, a truly fault-tolerant network fault monitoring system can be deployed, in which any fault-free node can diagnose the whole system.

6 Lessons from Biology

The distributed and fault-tolerant fault management paradigm of VINES has borrowed many ideas from the self defense mechanism of biological systems. The abnormality detection mechanism is distributed among the components of the system so that the basic health-check is done locally. Once an abnormality is detected the corresponding *Object for Diagnosis and Repair* (ODR) is sent to counter the abnormality. The ODRs are distributed in the network (body) amongst the various repositories (glands). A broadcast mechanism is used to announce the abnormality, followed by a mechanism for generating the ODR, and then the ODR is sent to neutralise the abnormality. In our case ODRs are made available in a distributed fashion in the network in several repositories. The set of the ODRs represent the complex network fault detection & diagnosis related knowledge. Establishing, algorithmically, the correspondence between a detected abnormality and an ODR is a challenging task. In the algorithm proposed in this paper we have utilised a novel MIB-based naming scheme. The premises on which this scheme is based are as follows:

- *All abnormalities can be detected in terms of MOs*: If there are some abnormalities that cannot be detected in terms of MOs- the MIB base can be expanded to include the MOs which can be used to detect the abnormalities.
- *The detection of abnormality can be done locally using the MIKB*: Since the MIKB contains the knowledge about how to interpret the values associated with MOs - the detection can be done locally by using the MIKB in conjunction with the MIB.

As a corollary to the above, an the MO naming scheme may be used to associate ODRs with the detected abnormalities. In the network when an abnormality is detected the search for the corresponding ODR will start. Multiple ODRs are likely to arrive due to the redundancy in the open-network. The selection of the most appropriate ODR from amongst arivees may be done on a first come first use basis, on the date of creation of the ODR, or based on past experience with ODRs from a particular source. The last would involve a learning algorithm which is a matter for future research.

7 Status and Conclusion

The prototype of VINES has been tested in a modular fashion to establish the proof-of-concept. The tests have been carried out on medium to large-size operational networks (WIDE, TOPIC, TAINS). The results have been very encouraging.

- Network Configuration has been automatically generated for arbitrary networks (where access control to the SNMP-MIBS were permitted). Using powerful layout algorithms - visually pleasing network maps have been produced. Further the Network configuration information generation time was reasonable enough to permit periodic reruns to ensure the reliability and uptodateness of the information.
- The "divide and conquer" algorithm applied to the TOPIC backbone network has established its efficacy beyond doubt. The number of fault-symptoms detected improved by $> 300\%$. Not only that, the fault-symptom isolation technique makes the network more *manageable* than when the fault-symptoms are seen in aggregation.
- Distributed System-Level Diagnosis Algorithms make the fault management process fault-tolerant. VINES combines two algorithms, Hi-ADSD for LAN diagnosis, and an algorithm for non-broadcast diagnosis to form a new solution for efficient internet diagnosis.
- The system offers the network manager great flexibility in customizing the management in accordance with the local conditions and personal likings.

The architecture is scalable in the global context. Provisions of elaborate access control measures and the DS allow the selective sharing of management information without compromising security of the network.

VINES is now under development. We expect that VINES will set a new direction for network management systems.

References

1. G.Mansfield, M.Murata, K.Higuchi, K.Jayanthi, B.Chakraborty, Y.Nemoto, S.Noguchi, "An SNMP-based Expert Network Management System",IEICE Trans. COMMUN.,Vol.E75-B,NO.8, pp.701-708, August 1992.
2. M. Kitahashi, S. Noguchi, "Object Oriented Analysis for Network Management" Journal of the Information Processing Society of Japan, vol-35, No 6, June 1994.
3. G.Mansfield, M.Ouchi, K.Jayanthi, Y.Kimura, K.Ohta, Y.Nemoto, "Techniques for automated Network Map Generation using SNMP", Proc. of INFOCOM'96, pp.473-480, March 1996.
4. E. P. Duarte Jr., G. Mansfield et. al. "Non-Broadcast Network Fault-Monitoring Based on System-Level Diagnosis" Proceedings of ISINM97, San Diego, May 1997.
5. William Stallings: "SNMP, SNMPv2, and CMIP - the Practical Guide to Network Management Standards -", Addison-Wesley Publishing Company 1993.
6. T. Bates, E. Gerich, et.al. "Representation of IP Routing Policies in a Routing Registry" ripe-181, http://www.ra.net/RADB.tools.doc/ripe-181.html, October 1994.
7. S. Williamson, M. Kosters, "Referral Whois Protocol (RWhois)" RFC1714, December 1994.
8. E. Feinler, K. Harrenstien, M. Stahl, "NICNAME/WHOIS" RFC0954, January 1985.
9. P. Mockapetris, "Domain names - concepts and facilities" RFC1035, January 1987.
10. K. Ohta, T. Mori, H. Sone, G. Mansfield, Y. Nemoto, et. al. "Divide and Conquer Technique for Network Fault Management" Proceedings of ISINM97, San Diego, May 1997.
11. E.P. Duarte Jr., T. Nanya "Hierarchical Adaptive Distributed System-Level Diagnosis Applied for SNMP-based Network Fault Management" Proceedings of the 15^{th} IEEE Symposium on Reliable Distributed Systems, Niagara-on-the-Falls, October 1996.