1

Decentralizing Control and Intelligence in Network Management[1]

Kraig Meyer, Mike Erlinger, Joe Betser, Carl Sunshine
The Aerospace Corporation
P.O. Box 92957, Los Angeles, CA, 90009, USA. Phone: +1 310-336-8114. Email: kmeyer@aero.org

Germán Goldszmidt, Yechiam Yemini
Computer Science Department, Columbia University
450 Computer Science Building, Columbia University, New York, NY, 10027, USA.
Phone: +1 212-939-7123. Email: german@cs.columbia.edu

Abstract

Device failures, performance inefficiencies, and security compromises are some of the problems associated with the operations of networked systems. Effective management requires monitoring, interpreting, and controlling the behavior of the distributed resources. Current management systems pursue a platform-centered paradigm, where agents monitor the system and collect data, which can be accessed by applications via management protocols. We contrast this centralized paradigm with a decentralized paradigm, in which some or all intelligence and control is distributed among the network entities. Network management examples show that the centralized paradigm has some fundamental limitations. We explain that centralized and decentralized paradigms can and should coexist, and define characteristics that can be used to determine the degree of decentralization that is appropriate for a given network management application.

Keywords

Network Architecture and Design, Management Model, Distributed Processing, Client-Server.

1 INTRODUCTION

Some experts in the field of network management have asserted that most, if not all, network management problems can be solved with the Simple Network Management Protocol (SNMP) [3]. This stems in part from the belief that it is nearly always appropriate to centralize control and intelligence in network management, and that SNMP provides a good mechanism to manage networks using a fully centralized management paradigm.

[1]This work was sponsored in part by ARPA Projects A661 and A662. The views expressed are those of the authors and do not represent the position of ARPA or the U.S. Government. This paper approved for public release–distribution unlimited.

A. S. Sethi et al. (eds.), *Integrated Network Management IV*

In this paper, we explore a number of different applications currently being used or developed for network management. We show that there are real network management problems that cannot be adequately addressed by a fully centralized approach. In many cases, a decentralized approach is more appropriate or even necessary to meet application requirements. We describe such an approach and start to build a taxonomy for network management applications. We specifically identify those characteristics that can be used to determine whether an application is more suitably realized in a centralized or decentralized network management paradigm. From the outset, it should be noted that many, if not most, network management applications can be realized in either paradigm. However, each application has characteristics that make it more suitable to one of the two approaches, or in some cases to a combination of both.

The remainder of this paper briefly lists what these characteristics are, discusses several categories of applications that have these differing characteristics, and analyzes some example applications. The next section describes two contrasting paradigms for network management: centralized and decentralized. Section 3 describes application characteristics that can be used to determine which paradigm is appropriate, along with some typical applications. Section 4 looks at four examples of decentralized applications in more depth. Finally, section 5 provides a conclusion and discussion of future work.

2 NETWORK MANAGEMENT MODELS

Basically, a network management system contains four types of components: Network Management Stations (NMSs), agents running on managed nodes, management protocols, and management information. An NMS uses the management protocol to communicate with agents running on the managed nodes. The information communicated between the NMS and agents is defined by a Management Information Base (MIB).

2.1 Centralized SNMP Management

The Internet-standard Network Management Framework is defined by four documents ([3], [6], [8], [9]). In the Internet community, SNMP has become the standard network management protocol. In fact, SNMP has become the accepted acronym for the entire Internet-standard Network Management Framework. Despite this, it should be noted that SNMP itself need not be bound to the paradigm that has developed around it. SNMP can be used as a reasonably general and extensible data-moving protocol.

To encourage the widespread implementation and use of network management, a minimalist approach has driven SNMP based network management. As noted in [10], "The impact of adding network management to managed nodes must be minimal, reflecting a lowest common denominator." Adherence to this "axiom" has resulted in a network management paradigm that is centralized, usually around a single NMS. Agents tend to be simple and normally only communicate when responding to queries for MIB information.

The centralized SNMP paradigm evolved for several reasons. First, the most essential functions of

network management are well-realized in this paradigm. Agents are not capable of performing self-management when global knowledge is required. Second, all network entities need to be managed through a common interface. When many of these entities have limited computation power, it is necessary to pursue the "least common denominator" strategy mentioned above. Unfortunately, in many cases this strategy does not allow for data to be processed where and when it is most efficient to do so.

Even when management data is brought to an NMS platform, it is frequently not processed by applications in a meaningful way. Network management protocols unify the syntax of managed data access, but leave semantic interpretation to applications. Since the semantic heterogeneity of managed data has grown explosively in recent years, the task of developing meaningful management applications has grown more onerous. In the absence of such applications, platform-centered management often provides little more than *MIB browsers*, which display large amounts of cryptic device data on user screens. As first noted in the introduction to [7], it is still the case that "most network management systems are passive and offer little more than interfaces to raw or partly aggregated and/or correlated data in MIBs."

The rapid growth in the size of networks has also brought into question the scalability of any centralized model. At the same time, the computational power of the managed entities has grown, making it possible to perform significant management functions in a distributed fashion.

Contemporary management systems, based on the platform-centered paradigm, hinder users from realizing the full potential of the network infrastructure on which their applications run. This paradigm needs to be augmented to allow for decentralized control and intelligence, distributed processing, and local interpretation of data semantics.

2.2 Decentralized Management by Delegation

Management by Delegation (MBD) [13] utilizes a decentralized paradigm that takes advantage of the increased computational power in network agents and decreases pressure on centralized NMSs and network bandwidth. MBD supports both temporal distribution (distribution over time) and spatial distribution (distribution over different network devices). In this paradigm, agents that are capable of performing sophisticated management functions locally can take computing pressure off of centralized NMSs, and reduce the network overhead of management messages.

At the highest level of abstraction, the Decentralized MBD paradigm and Centralized SNMP paradigm appear the same, as both have an NMS communicating with agents via a protocol. But the MBD model supports a more distributed management environment by increasing the management autonomy of agents. MBD defines a type of distributed process, Elastic Process [4], that supports execution time extension and contraction of functionality. During its execution, an elastic process can absorb new functions that are delegated by other processes. Those functions can then be invoked by remote clients as either remote procedures or independent threads in the scope of the elastic process.

MBD provides for efficient and scalable management systems by using delegation to elastic agents. Instead of moving data from the agent to the NMS where it is processed by applications, MBD moves the applications to the agents where they are delegated to an elastic process. Thus, management

responsibilities can be shifted to the devices themselves when it makes sense to do so.

Decentralization makes sense for those types of management applications that require or can take advantage of spatial distribution. For example, spatial distribution may be used to minimize overhead and delay. There is also an entire class of management computations, particularly those that evaluate and react to transient events, that must be distributed to the devices, as they can not be effectively computed in an NMS. Decentralization also allows one to more effectively manage a network as performance changes over time. The ability to download functions to agents and then access those functions during stressed network conditions reduces the network bandwidth that would be consumed by a centralized paradigm.

3 DISTRIBUTING NETWORK MANAGEMENT APPLICATIONS

The two paradigms of network management presented in the previous section might be viewed as contrasting, competing, possibly even incompatible models. The reality is that the SNMP (or centralized) paradigm and the MBD (or decentralized) paradigm are really just two points on a variety of continuous scales. An ideal network management system should be able to handle a full range of network management functions, for example using MBD's elastic processes to distribute management functionality in those cases where distribution is more efficient, but using SNMP's centralized computation and decision making when required. In this way, MBD should be seen as augmenting, rather than competing with, SNMP efforts. In fact, the SNMP community has already recognized the value of distributable management, with a manager-to-manager MIB [2] and some preliminary work on NMS-to-agent communications via scripts.

As previously mentioned, most of the early network management applications were well-suited to centralized control, which explains the success that the centralized SNMP paradigm has had to date. Some newer and evolving applications require a decentralized approach. A good example of an application that requires decentralization is the use of RMON (remote monitoring) probes [12]. RMON probes collect large amounts of information from their local Ethernet segment, and provide an NMS with detailed information about traffic activity on the segment. These probes perform extensive sorting and processing locally, and provide summary and table information via SNMP through a specially formatted MIB. Although this application uses SNMP for data transfer, in actuality, RMON is a realization of an application in the decentralized paradigm.

The question remains, how does one characterize network management applications in such a way that one can determine whether they should be distributed? There are a number of metrics that can be used to judge whether a network management application is more appropriately realized in a centralized or decentralized paradigm. These metrics are illustrated in figure 1 and include the following:

- **Need for distributed intelligence, control and processing.** This scale runs from a low need for distribution (corresponding with centralized intelligence) to a high need for distribution, or decentralized intelligence. An application that requires fast decisions based on local information will need decentralized control and intelligence. Applications that utilize

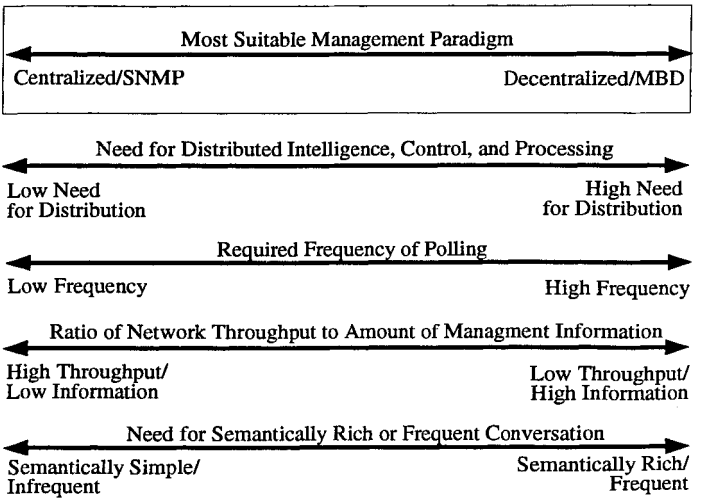

Figure 1: Metrics used to determine decentralization

large amounts of data may find it advantageous, though not always necessary, to perform decentralized processing. A specific example of this is an application that may need to use many pieces of data that can only be obtained by computing database views over large numbers of MIB variables. In this case, the application output may be very small, but the input to it may be an entire MIB.

- **Required frequency of polling.** The need for proximity to information and frequency of polling may dictate that computations be performed in local agents. This scale runs from a low frequency of polling to a high frequency of polling. An example of an application that requires a high frequency of polling is a health function that depends on an ability to detect high frequency deltas on variables.

- **Ratio of network throughput to the amount of management information.** At one end of this scale, the network in question has plenty of capacity relative to the amount of management information that needs to be sent through it. At the other end of the scale, there is a large amount of management information–so much that it conceivably could saturate the lower throughput network. An example of an application with a low throughput/high information ratio is the management of a large remote site via a low bandwidth link. Note that network throughput is affected not only by the amount of bandwidth available but also by the reliability of that bandwidth.

- **Need for a semantically rich and/or frequent conversation between manager and agent.** One end of this scale represents those applications that require only semantically simple and infrequent conversations, meaning that access to data is infrequent and simple

data types are all that need to be accessed. At the other end of this scale are applications that require frequent conversations and/or semantically rich interactions, meaning that complex data structures, scripts, or actual executables need to be passed to a remote server. An application that needs to download diagnostic code to agents on demand is an example of one that would require a semantically rich and frequent conversation.

3.1 Centralized Applications

From the discussion of these metrics, we can see that centralization is generally appropriate for those applications that have little inherent need for distributed control, do not require frequent polling or high frequency computation of MIB deltas, have high throughput resources connecting the manager and agent, pass around a small amount of information, and do not have a need for frequent and semantically rich conversations between the manager and agent.

Most network management applications that are currently being used fall into this category. One may argue that this is because the centralized (SNMP) paradigm is the only one that is realized in most commercial products, but in actuality this centralized paradigm was built because the most important network management needs fit these characteristics. The classic example of this is the display of simple MIB variables. Monitoring a router's interface status, or a link's up/down status, involves querying and displaying the value of a single or small number of (MIB) variables, and is well suited to centralized management.

The NMS network map is another example of a tool that requires input from a number of devices to establish current connectivity. Thus a decentralized approach would not provide the connectivity map that a centralized approach can quickly establish via an activity like ping.

3.2 Partially Decentralized Applications

"Partial Decentralization" is appropriate for applications that are bandwidth-constrained, but still require some degree of centralized administrative control. An example of a bandwidth-constrained application is the management of a west coast network by an east coast manager. If the networks are linked by a relatively low bandwidth link, it is desirable for all information about the west coast network to be collected locally by an agent on the west coast, and only summary information be passed back to the east coast. Another case of a "partially decentralized" application is when local networks are autonomous. A department administrator may manage a local network, passing only summary information up to the higher level network manager.

This category of applications also includes those that can be decentralized for the purpose of bandwidth and processor conservation. It may be possible to greatly reduce the amount of bandwidth or centralized processing required by having an agent perform a local calculation over a large amount of data, then reporting the result–a small amount of data–back to the centralized manager. This algorithm may be repeated on each subnet of a large network, effectively breaking one large calculation into many small calculations. Some applications of RMON and health functions fit this profile. Some applications for the management of stressed networks also fit this profile.

Some degree of decentralization is highly desirable for the applications in this category. This may

be accomplished by building a midlevel SNMP manager local to the variables being monitored, or by using elastic processes in the MBD paradigm. The SNMP solution is less general in that each midlevel manager must include both agent and NMS capabilities.

3.3 Decentralized Applications

Further analysis of the aforementioned metrics shows that decentralization is most appropriate for those applications that have an inherent need for distributed control, may require frequent polling or computation of high frequency MIB deltas, include networks with throughput constraints, perform computations over large amounts of information, or have a need for semantically rich conversations between manager and agent.

An example in this class is a health function that requires an ability to detect high frequency deltas on a set of MIB variables. A second example may be the management of a satellite or disconnected subnet, where a subnet manager is required to obtain data, make decisions, and change application or network characteristics even when that manager is isolated from the central, controlling manager. Finally, an application may have a need to download diagnostics and control information into a network element dynamically, in an attempt to isolate a problem.

Depending on the generality required, the SNMP manager-to-manager MIB may not be sufficiently general to allow for adequate delegated control for these applications. If frequent reprogrammability is a requirement, decentralization is the logical choice.

4 EXAMPLES OF DECENTRALIZED APPLICATIONS

We have identified four examples of network management applications that should be realized in a decentralized network management paradigm. These include Distributed Intrusion Detection, Subnet Remote Monitoring, Subnet Health Management, and Stressed Domain Management. What is presented below is a description of the activity and an analysis of its requirement for a decentralized approach. Current research efforts are involved in determining quantitative values for centralized and decentralized approaches to these applications.

4.1 Management of Distributed Intrusion Detection

Intrusion detection refers to the ability of a computer system to automatically determine that a security breach is in the process of occurring, or has occurred at some time in the past. It is built upon the premise that an attack consists of some number of detectable security-relevant system events, such as attempted logons, file accesses, and so forth, and that these events can be collected and analyzed to reach meaningful conclusions. These events are typically collected in an audit log, which is processed either in real time or off-line at a later time.

Intrusion detection requires that many potentially security-relevant events be recorded, and thus enormous amounts of audit data are a necessary prerequisite to successful detection. Simply recording all of the audit records results in a large amount of Input/Output (I/O) and storage overhead.

For example, if all audit events are enabled on a Sun Microsystems workstation running Multilevel Secure SunOS, it is possible for a single machine to generate as much as 20 megabytes of raw data per hour, although 1-3 megabytes is more typical [11]. Once the audit records are recorded, they must all be read and analyzed, increasing I/O overhead further and requiring a large amount of CPU processing. Audit data generally scales linearly with the number of users. As a consequence, expanding intrusion detection to a distributed system is likely to result in network congestion if all audit data must be sent to a central location. The CPU requirements scale in a worse than linear fashion: Not only must analysis be performed on each machine's local audit log, but correlation analysis must be performed on events in different machines' local logs. As a result, there is a high motivation to keep processing distributed as much as possible, and to keep the audit record format as standardized as possible.

Historically, the management of distributed intrusion detection has not been addressed in any standardized way. Banning [1] suggests that a list of an audit agent's managed objects should be stored in a MIB, and an audit agent should be managed using a standardized protocol such as CMIP [5]. However, to date, no intrusion detection systems have been widely fielded that perform this function.

Intrusion detection is an excellent candidate application for decentralized management. There is a high motivation for decentralized intelligence and processing because it is very clear that centralized processing won't scale, and that network bandwidth won't accommodate all audit data being sent to a centralized point. Further, there may be a need for a semantically rich conversation between distributed monitors, as they may need to pass relatively complicated structures that are hard to predefine in a MIB.

4.2 Subnet Remote Monitoring (RMON)

As previously mentioned, RMON [12] provides a framework in which remote monitoring probes collect information from local Ethernet segments, and provide this data to NMSs. RMON has in fact taken a hybrid centralized/decentralized approach to management. The RMON agent is responsible for collecting data from the local segment and performing calculations over that data (e.g., determining which stations are generating the largest amount of traffic). On a busy network, this may include maintaining a station table of over 3000 nodes along with packet counts. It is impractical, and inefficient, to download this entire station table to the management station for centralized processing. The entire transaction could easily take minutes, which is likely too slow to be meaningful.

In the RMON MIB a form of distributed processing was used in the creation of the Host Top N function. The Host Top N MIB group provides sorted host statistics, such as the top 20 nodes sending packets, or an ordered list of all hosts according to the number of errors they sent over the last 24 hours. Both the data selected and the duration of the study is defined by the user via the NMS. Once the requested function is setup in the agent, the NMS then only queries for the requested statistics.

Using a pure centralized approach for the Top N transmitting stations,[2] the NMS would have to

[2] Assume that a sort will be performed based on the number of packets transmitted by each station.

request statistics for all the hosts that have been seen on that subnet. Two such sets of requests would have to be made to determine the Top N: one to get a baseline count for each station and one to get the count for each station after a time, t. The difference between the two sets of requests would then be sorted by the NMS for the Top N display.

Assuming that statistics for only one station can be requested in each SNMP message, the total number of SNMP messages is 2 times the number of stations (ns) with a total SNMP cost of: 2 * ns * SC, where SC is the cost of an SNMP message.

If instead, the RMON approach is taken, the Top N function is distributed to the agent and the costs are greatly decreased. In this situation there are two costs. The first cost corresponds to the request that a Top N function be performed for some number of stations N < ns over some period t; the second is the cost of gathering the sorted statistics. Assuming that the set up costs (selection criteria and time period) can be established in two SNMP messages, the cost for a distributed top N function is: 2 * SC + N * SC. In the worst case, N = ns, decentralization costs (2 + ns) * SC. Thus whenever NS > 2, the decentralized approach of RMON is superior–costs less–than the usual centralized approach.

4.3 Management of Subnet Health Applications

Subnet health management is another application that requires some degree of decentralization. One of the difficult problems in a large network is the determination of the health of a subnet, where health is a dynamic function of a number of network traffic parameters. RMON is designed to provide data for the management of subnets. In a network of many subnets, e.g., a corporate network, the SNMP centralized paradigm puts a processing burden on the NMS and a data transfer burden on the network.

Subnet health can be determined using either the centralized or distributed paradigm. In a lightly loaded network, it is acceptable for the NMS to query all the subnets for information. The returned information can then be filtered by the management station to determine subnet health. The problem with this centralized paradigm arises in a loaded or congested network, especially when the amount of information being returned is large. When the network is loaded, the additional traffic generated by querying the subnets for large volumes of data can be significant. Thus the decentralized approach becomes necessary. This is a case where a large amount of information is needed relative to the throughput or bandwidth available on the network.

In the centralized approach the management station has the requirement to make some evaluation of subnet health by first gathering data and second, correlating that data. The decentralized approach localizes the gathering and correlation activities, so the local subnet then has the responsibility only to report its health based on some known health function.

The determination of whether subnet health is a centralized or decentralized activity is made not by the activity itself, but by variables affecting that activity. Thus, it is not the activity of gathering data and evaluating health that determines centralization. Rather, the effects of the network traffic on such gathering and the effects of such gathering on network traffic determine the choice between centralized and decentralized paradigms. This determination should be made dynamically by the NMS, which is able to determine and modify the balance of centralized versus decentralized activity.

The following steps might be taken:

- Using ping or a predefined health function, the NMS determines whether a centralized or decentralized approach should be used.
- If conditions favor a centralized approach, the NMS would request from the RMON agent all data that might be needed for various application tools. This is essentially the current approach.
- If a decentralized approach is determined to be needed, the NMS would request results from predefined RMON agent health functions.
- Based on these health functions, additional health data may be requested and/or new health functions downloaded to the agent. Each health function would put additional emphasis on agent health evaluation.

In some ways the above is a dynamic escalation from the centralized paradigm to the decentralized paradigm based on health functions. The goal of the NMS is to determine subnet health with minimal impact on the network as a whole.

4.4 Management of Stressed Networks

An additional application that is well-suited towards distributed management is the management of stressed networks. Networks in stressed conditions have a number of properties that require different management strategies from unstressed networks. For the purpose of this paper, network stress is defined as sustained operation at high utilization, and includes highly saturated network segments or devices. Related characteristics of such networks include longer delays, reduced effective connectivity, and less predictable responses. Network stress may be caused by failure of network components, causing phenomena such as loss of connectivity, increased packet traffic, and unexpected routing. A common characteristic of stress is that if left unattended, problems tend to escalate, and network resources become less available. The unstable stress phenomena are the most critical to address. Algorithms used for stressed region management must have the following characteristics:

- **Local Autonomy of Algorithm.** The algorithm must have good distributivity, provide most information locally, and only require low management bandwidth outside of the local domain.
- **Stress Containment using Routing.** Routing must be able to bypass problematic regions. Routing algorithms must be very distributed, with routing tables at each domain, and must react to changes in traffic patterns. In stress, there should be alternate routes known locally, but remote verification of reachability is required.
- **Local Network Domain Stabilization.** If the source of a problem is local, the local domain should be able to make decisions to contain and correct problems locally. If a stress source is external, outside consultation is required.

- **Gradual and Graceful Degradation.** Management algorithms should function and network services should continue—albeit with worse performance—as network stress grows. This typically requires a distributed architecture, with low dependency on remote resources and high dependence on local autonomy.
- **Stress Prediction.** Distributed health monitoring allows for local domains to anticipate stress conditions before they actually occur. Countermeasures may be taken locally or may require interaction between domains.

A basic technique for stress monitoring involves the correlation of MIB variables reflecting local stress (such as retransmissions, packet lengths, and timeouts). These correlations should be done on a domain-by-domain basis, for efficient collection of data from neighboring nodes, and thus computations would be distributed. This may also naturally lead to distributed control and decentralization. Local managers would conduct cross-correlations on a regular basis, and patterns of stress could be established and trigger stress alarms for that domain. Similarly, higher level managers would conduct cross-correlations of domain-manager information, to establish "regional" stress propagation, and devise policies and strategies to combat escalating stress. All these activities are very likely to be distributed in a hierarchical fashion among network domains.

A need for distributed control, bandwidth limitations, and other characteristics of stress management indicate that decentralization may provide significant benefits in effectively managing network and system stress.

5 CONCLUSIONS AND FUTURE WORK

We have described two network management paradigms, SNMP and MBD, that have historically represented conflicting views of how networks should be managed. We have shown that the centralized approach associated with SNMP and the decentralized approach of MBD are actually just two points on a continuous scale of network management approaches. We have started building a taxonomy for network management applications and identified a number of characteristics that can help to determine whether a given network management application should be realized in a centralized paradigm, a decentralized paradigm, or some hybrid of the two. Finally, we have focused on four specific examples of network applications and explained why none of them is best realized in a strict, fully-centralized network management paradigm.

We plan to continue to investigate network management approaches through a series of experiments directed at quantifying the choice of network management paradigm. We believe that the costs associated with the various paradigms can be used by applications to dynamically choose among centralized, decentralized, or hybrid approaches to network management. The experiments should also provide additional input to extend the list of characteristics that effect the choice of network management paradigm.

References

[1] D. Banning, et. al. *Auditing of Distributed Systems.* Proceedings of the 14th National Computer Security Conference, pages 59–68, Washington, D.C., October 1991.

[2] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser. *Manager-to-Manager Management Information Base.* Request for Comments 1451, April 1993.

[3] J. Case, M. Fedor, M. Schoffstall, and J. Davin. *A Simple Network Management Protocol (SNMP).* Request for Comments 1157, May 1990.

[4] G. Goldszmidt. *Distributed System Management via Elastic Servers.* Proceedings of the IEEE First International Workshop on Systems Management, pages 31–35, Los Angeles, California, April 1993.

[5] International Standards Organization (ISO). *9596 Information Technology, Open Systems Interconnection, Common Management Information Protocol Specification,* May 1990.

[6] K. McCloghrie and M. Rose. *Management Information Base for Network Management of TCP/IP-based internets: MIB-II.* Request for Comments 1213, March 1991.

[7] B.N. Meandzija, K.W. Kappel, and P.J. Brusil. Introduction to *Proceedings of the Second International Symposium on Integrated Network Management,* Iyengar Krishnan and Wolfgang Zimmer, editors. Washington, DC, April 1991.

[8] M. Rose and K. McCloghrie. *Structure and Identification of Management Information for TCP/IP-based Internets.* Request for Comments 1155, May 1990.

[9] M. Rose and K. McCloghrie. *Concise MIB Definitions.* Request for Comments 1212, March 1991.

[10] M. Rose. *The Simple Book, An Introduction to Management of TCP/IP-based Internets.* Prentice Hall, 1991.

[11] O. Sibert. *Auditing in a Distributed System: SunOS MLS Audit Trails.* Proceedings of the 11th National Computer Security Conference, Baltimore, MD, October 1988.

[12] S. Waldbusser. *Remote Network Monitoring Management Information Base.* Request for Comments 1271, November 1991.

[13] Y. Yemini, G. Goldszmidt, and S. Yemini. *Network Management by Delegation.* Second International Symposium on Integrated Network Management, pages 95–107, Washington, DC, April 1991.

Kraig Meyer is a Member of the Technical Staff at The Aerospace Corporation in El Segundo, CA. He has previously worked as a lecturer and research assistant at the University of Southern California, and as a Systems Research Programmer on the NSFNET project at the Merit Computer Network. His research interests include computer network security, protocols, and management. Kraig holds a BSE in Computer Engineering from the University of Michigan and an MS in Computer Science from the University of Southern California.

Mike Erlinger is a Professor of CS at Harvey Mudd College, and a member of the technical staff at The Aerospace Corporation. Mike has founded and chaired the CS department at Mudd, and has technical program support responsibilities at Aerospace, as well as a lead role in several of the research efforts, such as the Southern California ATM Network. He has also founded and chaired the RMON MIB WG within the IETF. Mike has worked for Micro Technology as Director of Network Products and previously for the Hughes Corporation. His interests are in the areas of network management, software engineering, system administration, and high speed networking.

Joe Betser is the founder and head of the Network and System Management Laboratory at The Aerospace Corporation. Dr. Betser provides the national space programs with ongoing technical guidance and also serves as an ARPA PI. Joe established research collaborations with Columbia University and several California centers active in high speed networking and ATM. His new work focuses on QOS for tele-medicine, tele-multi-media, and other imaging applications. Joe served on the program and organizing committees for NOMS, ISINM, MilCom, and other computer communications events, and in particular, has chaired the vendor program at ISINM'93. Joe holds a PhD and MS in CS from UCLA, and a BS with Honors from Technion, Israel Inst. of Tech.

Carl Sunshine has been involved in computer network research from the early development at Stanford University of the Internet protocols. He subsequently worked at The Rand Corporation, USC Information Sciences Institute, Sytek (now Hughes LAN Systems), and System Development Corporation (now Unisys). Dr. Sunshine's work encompassed a range of topics including network protocol design, formal specification and verification, network management, and computer security. Since 1988 he has been with The Aerospace Corporation, managing computer system research and development for a variety of space programs.

German Goldszmidt is a PhD candidate in Computer Science at Columbia University, where he is completing his dissertation, entitled "Distributed Management by Delegation". He received his BA and MS degrees in Computer Science from the Technion. His Master's thesis topic was the design and implementation of an environment for debugging distributed programs. Since 1988 he worked at IBM Research, where he designs and develops software technologies for distributed applications. His current research interests include distributed programming technologies for heterogeneous systems, and network and distributed system management.

Yechiam Yemini (YY) is a Professor of CS and the Director of the Distributed Computing and Communications Laboratory at Columbia University. YY is the Founder, Director, and Chief Scientific Advisor of Comverse Technologies, a public NY Company producing multimedia store-and-forward message computers. YY is also the Founder and Chief Scientific Advisor of System Management Arts (SMARTS), a NY startup specializing in novel management technologies for enterprise systems. YY is frequently invited to speak in the areas of computing, networks, distributed systems, and the interplay among these areas, and is the author of over 100 publications.