Conf-930461--5

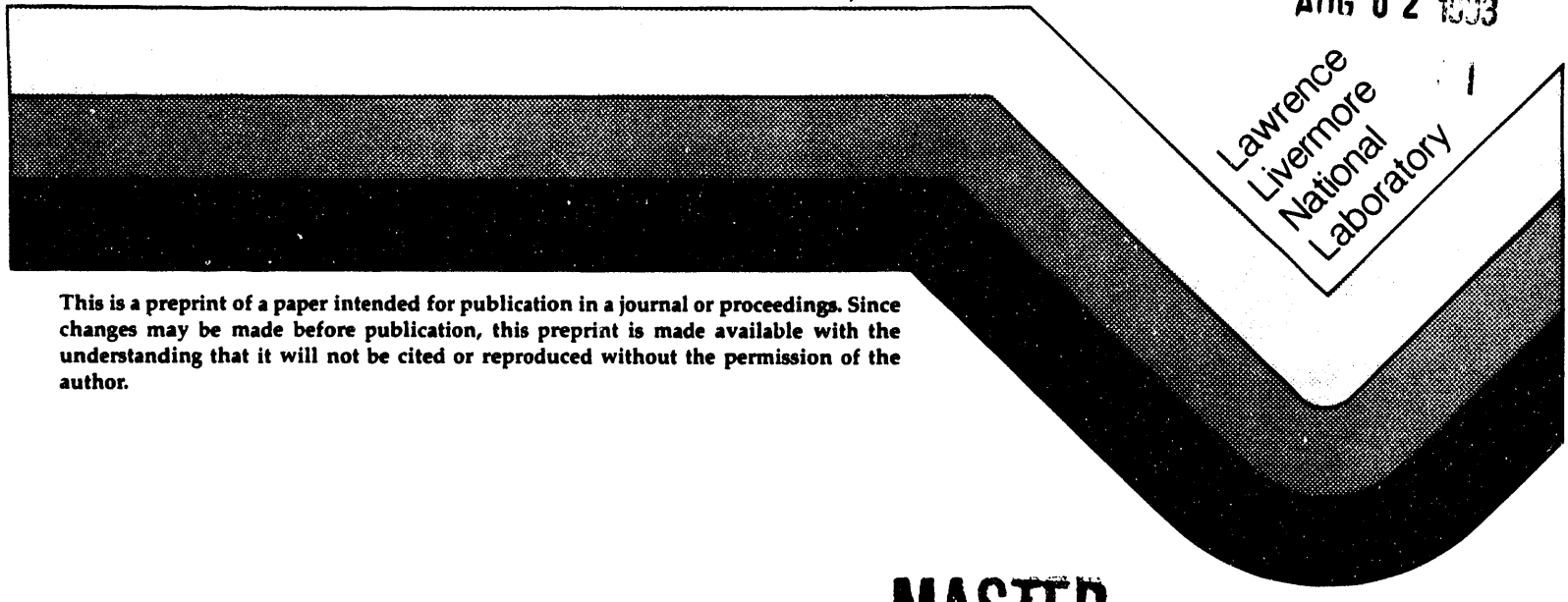
UCRL- JC - 112556
PREPRINT

Applying IEEE Storage System Management Standards at the National Storage Laboratory

Steven Louis and Susan W. Hyer
Lawrence Livermore National Laboratory
Livermore, CA

This paper was prepared to be submitted to
Twelfth IEEE Symposium on Mass Storage Systems
Monterey, CA
April 25-29, 1993

December 4, 1992



Applying IEEE Storage System Management Standards at the National Storage Laboratory

Steven Louis and Susan W. Hyer
Lawrence Livermore National Laboratory
Livermore, California

Abstract

Since its inception in 1990, the IEEE Storage System Standards Working Group has identified storage-system management as an area in need of further development. The pressing need for standards in storage-system management arises from the requirement to exchange management information and to provide control in a consistent, predictable manner between the components of a storage system. An appropriate set of management standards will allow multiple vendors to supply storage management subsystems or applications that are integral to or compatible with new storage systems conforming to future IEEE standards.

An early, practical application of IEEE storage-system-management work is being pursued at the National Storage Laboratory (NSL), a recently-formed industrial collaboration at Lawrence Livermore National Laboratory. The NSL's purpose is to develop advanced hardware and software technologies for high-performance, distributed storage systems. Since storage-system management is of critical concern, it is being explored in depth at the NSL. Work was initiated to define basic management requirements and develop generalized graphical-user-interface tools using remote-procedure-call mechanisms to implement the NSL's conceptual management framework. Several constraints were imposed on the development of early versions of this work to maintain compatibility with the NSL's underlying UniTree-based software architecture and to provide timely prototypes and proof of concept.

The project leverages the on-going standards work of the IEEE Storage System Standards Working Group (SSSWG) and utilizes the ideas of previous, related efforts, including existing ISO management standards. It also explores some of the relationships and interactions between IEEE storage-system management and more well known management methods for distributed systems and networks. The early application of storage-system-management standards has the immediate benefit of providing basic management capabilities for the high-performance storage systems under development at the NSL. It will also have longer-term benefits by providing "real-life" storage-system-management requirements to the IEEE SSSWG for validation of evolving standards.

Introduction

The development of new computing technologies, driven by the need for continued productivity in scientific and commercial communities, has resulted in dramatic increases in data storage requirements. Advances in massively parallel processors, memory capacities, distributed computing capabilities over high-speed networks, and new generations of applications dictate a re-evaluation of traditional storage architectures, and a search for innovative, cost-effective storage solutions. Experts in storage systems are now describing an emerging paradigm shift from CPU-centered to network-centered storage-system architectures [1]. Commercial vendors have started to recognize that the phenomenon of explosive growth coupled with an emerging, diverse, distributed application environment will require consistent, shared, network storage devices with simplified administration and automated management [2].

Searching for new network-oriented approaches to storage was the catalyst in the formation of the National Storage Laboratory [3]. The NSL is an industry-driven collaboration organized to investigate new high-performance storage-system technologies. One of the key points of the NSL's architectural configuration is the ability to distribute the individual components of a storage system and allow communication between these components through high-speed switching fabrics. As new network technology begins to blur the distinction between local and remote components of a storage system, effective management becomes crucial.

The need for effective management of distributed processes and open systems has long been recognized in the network domain. ISO (the International Organization for Standardization) has developed a series of standards for systems management related to their OSI (Open Systems Interconnection work. These standards include a management framework and architecture [4,5], management information services and protocols [6,7], a management information model [8], and a guideline for the definition of managed objects [9]. In parallel with the OSI efforts, the Internet developed and adopted SNMP (Simple Network Management Protocol) in 1988 [10]. SNMP has now become a widely-accepted protocol standard for network

management. An enhanced follow-on SNMP protocol, called the Simple Management Protocol (SMP) and developed to address shortcomings of the original SNMP, is currently undergoing evaluation by the Internet Engineering Task Force [11].

Other organizations, including standards bodies developing portable operating systems and open, distributed-processing frameworks [12], have addressed distributed system management concerns. Although most of these efforts are network or operating systems oriented approaches to management, a few have touched upon areas likely to have an impact on storage-systems management. An example is UNIX International's System Management Working Group. This organization has published several documents describing storage device management [13] and hierarchical storage management [14]. They state that effective storage management is critical because it improves data processing in several areas: performance, availability, space utilization, device installation, and user productivity. They conclude that inefficient storage is costly in terms of machine and human resources and, eventually, money.

The many developments in network and systems management have led to a diversity of sometimes incompatible methods of managing distributed resources. A highly-visible attempt to address this problem is the Open Software Foundation's Distributed Management Environment (DME) architecture [15]. DME provides functions and services that unify and support both network and systems management. It will also provide support for future, distributed storage-system management. The DME specifies an object-oriented infrastructure for distributed management applications and will support Internet and OSI management protocols. It will also have a unified management user interface that integrates all types of management interactions, and will include distributed notification services.

Still another set of global specifications for common network management methods is the Network Management Forum's Omnipoint. In its first release, Omnipoint 1 include nearly one hundred specifications that presents an envelope or wrapper around a number of existing management standards. It is expected to provide a common implementation profile for many of these standards. A follow-on specification, called Omnipoint 2, will have increased breadth of functionality across many industry focus groups and sector profiles.

The NSL has begun to investigate some of the specifics of storage-system management and its vital importance to the successful utilization of new storage-system hardware and software technologies. The NSL effort builds from much of the extant work in network and systems management, and follows the guidelines of the evolving IEEE Mass Storage Reference Model [16,17].

Storage-System Management

The availability of adequate storage-system-management tools is now recognized as an important aspect of the increasingly complex hierarchical storage systems developed over the past decade [18]. Simply stated, the role of storage-system management is to monitor and control the available resources of a storage system in ways that conform to the particular management policies of a given site. A subtle assumption is that site policies for storage are usually constructed to make optimal use of resources, but there is no strict requirement that efficient resource utilization be the driving force behind such policies. For example, sub-optimal usage may result when site policies provide to a particular group of users (e.g., those with the most money or power) an over-allocation of resources to the detriment of the general user base and the storage system as a whole. The IEEE SSSWG takes the position that the intent of standards for storage-system management is to provide a generalized framework for the design of useful management subsystems and applications, rather than to accommodate or force specific management facilities or policies.

The basic role of storage-system management within the IEEE SSSWG model is shown in Figure 1. Storage-system management communicates with all servers in the storage system and can communicate with clients of the storage system as well. Other external processes may also communicate with the storage-system-management component of the model to affect various management operations and behavior within the storage system. Specific kinds of storage-system-management services have been described in Version 5 of the IEEE Mass Storage Reference Model. Some of the functions provided by storage-system management may be included as part of a server's standard client interface. We assume that there will be various levels of authorization associated with storage-system-management functions available through a server's client interface. Without an adequate authentication and authorization mechanism, potentially dangerous operations could be performed by an arbitrary client, leading to detrimental results.

Some management capabilities, such as administrative configuration control or migration and de-fragmentation, may not be provided via a server's general client interface. These kinds of management operations may be performed by the external processes of Figure 1 and may be developed outside the scope of an IEEE storage system.

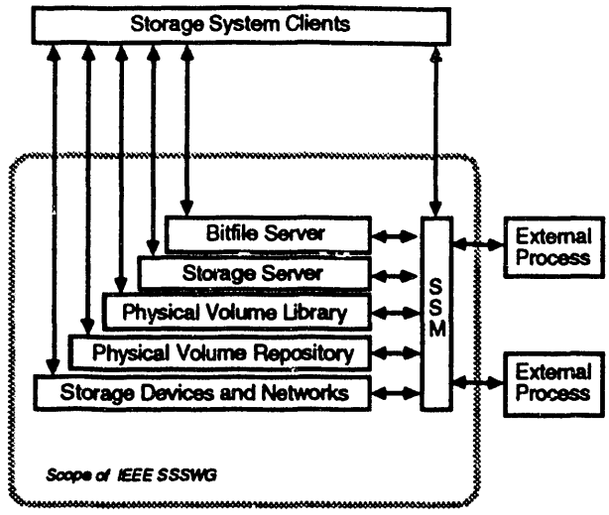


Figure 1. IEEE storage-system management

Storage-system-management operations can be performed in a variety of ways. The amount of automation inherent in the performance of an operation can range from none to completely automated:

- Operations that require a manual decision to initiate, followed by a manual analysis process, followed by a manual corrective action, if necessary. An example is an operator-initiated job that prints out a list of I/O errors for a specified group of hardware devices over a particular time.

- Operations that are automatically initiated, but require a manual analysis process and manual corrective action, if necessary. An example is an automatic alert message warning of a high number of I/O errors for an individual device or group of devices. The device remains operational until an administrator decides it should be taken off-line or put into a less-than-fully-operational mode (e.g., read-only).

- Operations that are automatically initiated and analyzed, but require a manual corrective action, if necessary. An example is an automatic decision to change the state of a device because a known threshold of errors has been reached. It may still require further manual corrective actions to access and protect data on the failing device.

- Operations that are automatically initiated, automatically analyzed, and automatically provide a corrective action, if necessary. Examples are dynamic device load balancing, volume shuffling and clustering due to suspect or failing conditions, and automated reconfiguration of hierarchical data placement and migration algorithms [19].

Most storage system environments have a preponderance of management activities at the less automated end of this scale. A goal of successful storage-system management is to provide facilities and services that allow many operations to be at the more automated end of the scale without necessarily precluding operations at the manual end. The degree of management automation or "self-management" should be considered a matter of site policy and should not be rigidly dictated. From a practical point of view, the management of storage systems would be "easier" and "safer" if most of the management operations were more automatic. It should not be surprising that an examination of LLNL's experience with storage systems shows a strong correlation between the complexity of manual operations and the probability of human error. It is also true that the possibilities for catastrophic consequences increase dramatically in complex storage-system environments.

The value of storage-system-management services, tools, and applications corresponds to the ability of those services, tools, and applications to facilitate "self-managed" or automated operations. The goals of storage-system management can be summarized as follows:

- High fault tolerance
- High recoverability
- High resource utilization
- Balanced workloads
- Conformance to site policy
- Automated management

A storage system that meets these goals will be viewed by both users and administrators as available, reliable, efficient, effective, simple to manage, and simple to use.

Management Models

A storage-system-management capability depends upon the existence of an entity that describes specific aspects of a physical or logical resource of the system. The aspects described must be of interest to or needed by management applications. In most network and systems management models, these entities are called managed-object definitions and represent the management view of the resources of the storage system.

Most of the approaches in use today for modeling distributed applications and management of distributed network and system resources are object-oriented. The

use of object-oriented principles was widely accepted because of its facility in promoting modularity, extensibility and component reuse.

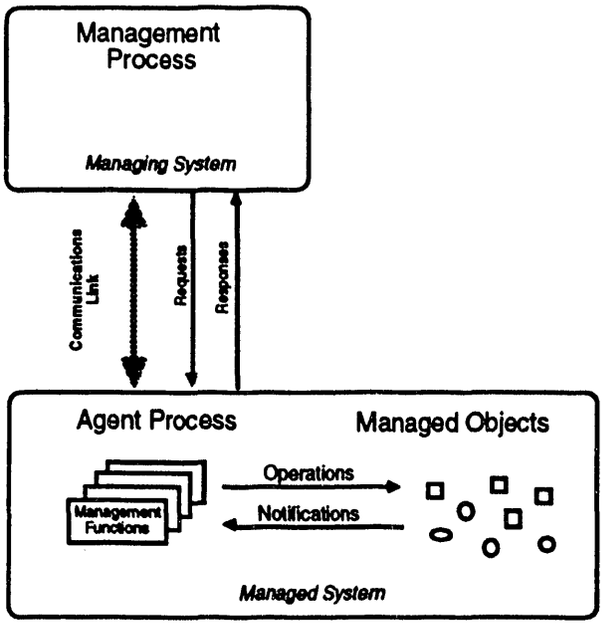


Figure 2. OSI Management Model.

The ISO Open Systems Interconnect management model defines a management interface between a manager and an agent and is shown in Figure 2. Applications issuing management requests and receiving notifications are viewed as managers. The requests are operations on agent-maintained managed objects and their attributes. Notifications can be responses to previous requests or can be asynchronously delivered descriptions of significant events. This model makes a distinction between a managed object and the physical resource that it represents by defining an interface to a managed resource that is independent of the implementation of the resource. The resource is visible to management only through this interface, called the managed-object boundary.

The object model defined by the Object Management Group specifies the interactions between client applications and the objects that it can utilize to perform services [20,21]. A logical entity called an object-request broker mediates the interaction between clients and server objects. Servers are always objects. Clients may be objects, but they may also be application programs or libraries as well. In the OMG model, the defined interfaces are the communication procedures between clients and the object request broker, or between the object request broker and server objects.

The OSI and OMG models have been adapted and refined by several organizations [22]. The OSF DME and others reconcile these two models by describing management solutions as a collection of cooperating objects that share information and provide application functions. In this combined approach, the strict distinction between manager and agent is not maintained. Objects contain management knowledge in terms of object state and behavior. Since the objects are addressable across a distributed network, the knowledge present in these objects is obtainable by a variety of application clients. A specialized implementation of the OMG's object request broker, called the management request broker, has been developed and optimized in OSF's DME for management services. Its relationship to applications and servers is shown in Figure 3.

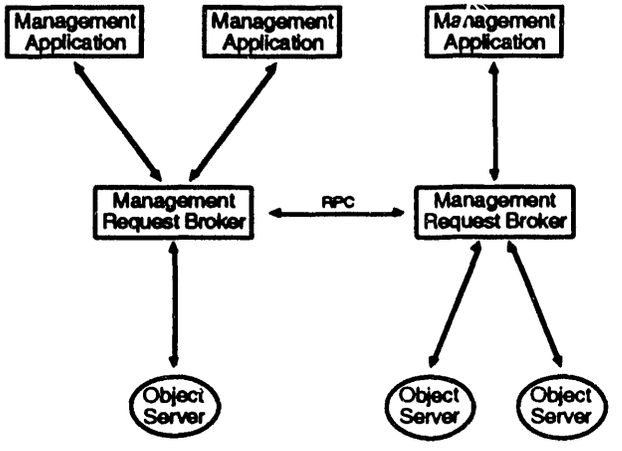


Figure 3. OSF DME Management Model.

The IEEE SSSWG used the OSI management model as the starting point for its storage-system-management work primarily because it represents an accepted set of international standards in the management area and most of its concepts have been included in newer vendor-driven efforts. As new approaches (e.g., OSF DME and NMF Omnipoint) progress in gaining wider vendor and user acceptance, the IEEE SSSWG and the NSL will adapt their work as appropriate.

NSL Requirements

A goal of the storage-system-management component of the NSL collaborative project is the development of standards-based management interfaces and protocols using modular managed-object definitions. A set of managed objects will be identified and defined together with the operations, attributes, and notifications necessary for the development of practical management capabilities. We expect the results of the storage-system-management work at the NSL to drive the IEEE SSSWG effort and influence its eventual direction.

The storage-system-management requirements for the NSL were generated, in part, by identifying and generalizing the specific management operations of storage systems at the two major computer centers at LLNL. The Livermore Computer Center currently runs the Livermore Distributed Storage System [23,24], the precursor of UniTree. The National Energy Research Supercomputer Center runs the Common File System developed at the Los Alamos National Laboratory [25]. These systems, along with most other modern storage systems, meet several management requirements including (but not limited to) purging, migration, space reclamation, and backup. It should be noted that since the storage system under development at the NSL is an enhanced version of UniTree (referred to as NSL-UniTree), initial managed-object definitions were heavily influenced by UniTree-specific requirements and capabilities.

The NSL requirements identification and managed-object definition tasks have been divided into two phases. In Phase One, only those requirements deemed necessary for the successful day-to-day operation of UniTree are identified. As stated above, the managed-object definitions for Phase One have been heavily influenced by the information readily obtainable from UniTree. In Phase Two, and subsequent phases, more general storage-system-management requirements will be examined, and managed objects will be defined beyond those provided by the UniTree architecture.

For Phase One NSL storage-system management, we have defined five different types of management clients, together with a set of security services to provide protection against unauthorized requests. The different client types in approximate order of capability level are:

- Site Manager
- Systems Personnel
- Operations Personnel
- Privileged Users
- Non-Privileged Users

Management operations performed by systems personnel may include general storage system configuration tasks, diagnosis and resolution of error conditions, and addition or deletion of various physical and logical components. In situations where it may be dangerous for multiple systems personnel to concurrently-initiate similar management operations, an individual Site Manager may be given sole responsibility or authority for certain tasks.

Some examples of management tasks for operations personnel are rote activities that must be performed from a centralized point, such as starting and stopping background support utilities like migration and reclamation procedures, or disabling devices for service or repair. At LLNL and the NSL, operations personnel are limited in what they can do, and normally are allowed to perform only a small subset of the management operations able to be performed by systems personnel. These limitations are, of course, site- and system-dependent policies and may vary according to the sophistication and technical level of the operators.

In many environments, a subset of the general user community is deemed to be "privileged" in that they are allowed to issue management operations beyond what is allowed to the general user community. Typically, these operations include the ability to examine and/or change attributes of the storage system normally not visible to a user (e.g., the ability to list or modify another user's directory). It should be noted that some of the abilities of privileged users may not necessarily be available to operations personnel.

A subset of management operations usually exists that are innocuous enough to be issued by anyone. These might include receiving dynamic status information or summarized statistical information about the storage system. Sometimes this information is available directly from the storage-system servers as part of the server's general client interface (see Figure 1).

For this work, we define three basic types of management processes:

- Monitoring Functions
- Control Functions
- Alarm Notifications

This is an oversimplification of the full range of management operations that will eventually be present in the NSL and it is likely that these types will later be expanded and more fully defined.

Monitoring functions are management operations that provide the capability to a management client to receive management information about the storage system. A typical example is information about the state of physical and logical storage devices and volumes. Displays, or other mechanisms for information communication, are needed for major storage-system component and servers and should be capable of providing basic summary information for known activity within the server. Displays are used at the NSL primarily because the NSL's storage-system-management implementation is based on a graphical user interface (GUI) environment. Mechanisms are also needed to monitor resource utilization, general configuration, and specific policies relevant for each server.

Management operations that allow a management client to change the state or other characteristic of some component in the storage system are classified as control functions. Typical examples include changing the administrative or operational state of storage devices and volumes, and setting threshold levels for internal counters that may trigger an internal process to start. Interactive displays or other mechanisms are needed to allow authorized management clients to easily perform these control operations.

Alarm notifications (usually asynchronous) are the result of significant events occurring within the storage system and are differentiated here from other kinds of asynchronous events that convey simple monitoring information. Alarm notifications, while not necessarily due to fault conditions, are intended to convey important events affecting the operational health of the system. An alarm usually indicates that some immediate or eventual management action needs to be initiated. Some typical examples are error conditions on file transfers, security violations, inconsistent directory information, and the exceeding of pre-defined thresholds. A communication mechanism is needed to display and/or record the alarms, alerting appropriate personnel. Additional information is usually provided with the alarm concerning alarm type, the probable cause, the severity level, and possible actions that might be taken.

Implementation

The initial NSL storage-system-management implementation is based on the Sammi Graphical User Environment. Sammi is a UNIX-based system from Kinesix that connects with management applications and can control or be controlled by those applications. It was chosen as the top layer in the NSL storage-system-management architecture primarily for its ability to provide straightforward, fast prototyping and an easy-to-use GUI environment. Sammi is also compatible with the IBM RISC 6000 workstations in use at the NSL and is able to act as a well-behaved X-client, using X-Windows and Motif Widgets. The overall NSL storage-system-management architecture is shown in Figure 4.

The top portion of Figure 4, which contains the displays, runtime environment, and application programming interface, is actually the Kinesix Sammi Graphical User Environment and can be conceptually viewed as equivalent to one of the External Process boxes in Figure 1. Windowed displays are developed using a special Sammi Format Editor that produces format files interpreted by the runtime environment at display time. User access to displays can be controlled by system administration security policies. The bottom portion of Figure 4 is part of the NSL-UniTree storage-system-management development work and corresponds to what is contained within the shaded outline box of Figure 1. The Data Server Process is a mechanism to allow the Sammi environment to act as a client to the Storage System Manager. Other (non-Sammi) clients have the ability to interact with the Storage System Manager through varying types of application processes. In the current NSL implementation, however, only a Sammi-specific Data Server Process is used and the Sammi runtime environment interacts with a single NSL-UniTree storage system. The NSL-UniTree environment is not precluded from interacting with multiple external management processes.

In the NSL implementation, the ability for the separate components to efficiently communicate comes from a common knowledge of NSL-UniTree managed-object definitions. Communication between the Sammi Runtime Environment and its peer application (the Data Server) is through industry-standard Remote Procedure Calls (RPCs). The Storage System Manager also communicates with the Data Server and with the UniTree management interface routines through RPCs.

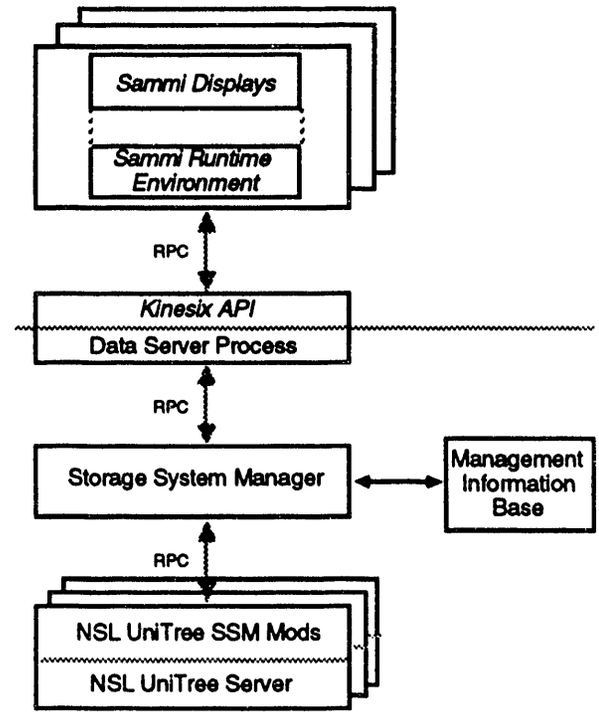


Figure 4. NSL management configuration

For each Sammi display, which may in turn contain several individual smaller displays of managed-object attribute values, a structured request is sent from the Sammi Runtime Environment through the Data Server to the Storage System Manager. The request acts a get/set data operation or as a registration process to receive data at regular intervals or as asynchronous events. If the Sammi display is subsequently terminated

because it is no longer needed, a corresponding de-registration process occurs for the information that was being received. In between the registration and de-registration, the Storage System Manager and management interface modifications in NSL-UniTree are responsible for keeping information about active attribute groups or other information types.

The Storage System Manager is able to intercept and restructure management requests sent to it by the Data Server if it can determine that those requests do not need or should not go to the NSL-UniTree management interface routines. A request may not need to be sent to UniTree because the Storage System Manager has the ability or the responsibility to formulate the response. Some management information may not be available directly from UniTree but may instead be contained in the Storage System Manager's Management Information Base (MIB). For example, if an attribute for a managed object does not have a direct counterpart in UniTree, but its value can be determined by the Storage System Manager by making multiple requests to UniTree over time, that attribute value can be summarized or calculated in the MIB. In a similar manner, the MIB may contain information previously retrieved from UniTree and kept current by the Storage System Manager through periodic updates from UniTree. The MIB shown in Figure 4 is an abstraction that represents a holding place or repository for current, saved, summarized, and calculated data. In Phase One its actual implementation is as a Storage System Manager data structure in memory, but could just as easily be implemented as a local storage device.

In the Phase One NSL implementation, UniTree-managed object definitions have been constructed so the need to intercept requests and form responses in the Storage System Manager is minimized. Most monitor and control information will flow directly to and from UniTree. This simplifies the communication interactions between the architectural components. The Phase One managed objects define only the attributes directly retrievable from NSL-UniTree. The number of managed objects for Phase One was kept small because of the decision to model only those management concerns necessary for successful day-to-day operations. Formal inheritance and containment hierarchy definitions for managed-object classes were not attempted for Phase One. Subsequent phases will allow more complex managed-object definitions. Valid Phase One directives include getting and setting attribute values, registering and de-registering to receive asynchronous events or data at regular intervals, and performing specific operations such as initiation of a full or partial backup. Some typical attributes for an object describing resource utilization in a UniTree disk or tape server for Phase One are total headers, number of used headers, total data space, and free data space. Both the disk and tape servers have alarms for resource exhaustion. A strong effort was made to develop the UniTree storage-system-management interface modifications in a modular fashion to allow extensibility for Phase Two and later phases.

Conclusions

The extensive work already accomplished and work in progress for network and systems management significantly overshadows efforts in storage-system management. Many of the ideas concerning management of widely-distributed processes have been well explored and it is becoming more apparent that the computing industry is moving toward consensus for common management frameworks that cover many disjoint areas, including storage systems. With that in mind, we have taken some initial steps within the NSL to investigate how to manage storage systems modeled after the IEEE Mass Storage System Reference Model. The fluidity and speed of progress inherent in evolving distributed management environments strongly suggest that we be guided in future enhancements by the above mentioned consensus developing within both industry and standards organizations. We also encourage these industry and standards organizations to look toward efforts such as the NSL and the IEEE SSSWG to help guide them in developing effective management tools for storage systems.

Acknowledgments

We wish to acknowledge the contributions of Tracy Tran and John Nguyen of IBM Federal Systems Company in Houston for their work on the Sammi graphical user environment, Data Server Process and Storage System Manager. We also thank Gregg Hommes of NERSC for his contributions to the NSL-UniTree storage system management modifications. This paper describes work performed under auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

References

1. Coleman, S. and R. W. Watson, "The Emerging Paradigm Shift in Storage System Architectures", *to be published in the Proceedings of the IEEE*, April 1993.

2. Kondoff, A. and T. M. Ravi, *HP Distributed Information Storage Architecture*, Hewlett-Packard Information Architecture Group, October 1991.

3. Coyne, R. A., H. Hulen and R. W. Watson, "Storage Systems for National Information Assets", *Supercomputing '92 Proceedings*, Minneapolis, MN, November 1992.

4. ISO/IEC 7498-4, *Information Processing Systems — Open Systems Interconnection — Basic Reference Model — Part 4: Management Framework*, 1989.

5. ISO/IEC 10040, *Information Technology — Open Systems Interconnection — Systems Management Overview*, 1991.

6. ISO/IEC 9595, *Information Technology—Open Systems Interconnection — Common Management Information Service Definition*, 1991.

7. ISO/IEC 9596-1, *Information Technology—Open Systems Interconnection — Common Management Information Protocol Specification*, 1991.

8. ISO/IEC 10165-1, *Information Technology—Open Systems Interconnection — Structure of Management Information Part 1: Management Information Model*, 1991.

9. ISO/IEC 10165-4, *Information Technology—Open Systems Interconnection — Structure of Management Information Part 4: Guidelines for the Definition of Managed Objects*, 1991.

10. Case, J., M. Fedor, M. Schoffstall and C. Davin, "Simple Network Management Protocol (SNMP)", *Internet RFC 1157*, May 1990.

11. Case, J., K. McCloghrie, M. Rose and S. Waldbusser, "Introduction to the Simple Management Protocol (SMP) Framework", *Internet Engineering Task Force Draft Document*, October 1992.

12. ISO/IEC 10746-1, *Basic Reference Model of Open Distributed Processing Part 1: Overview and Guide to Use, Working Draft*, 1992.

13. UNIX International, *Storage Device Management*, UI System Management Working Group, June 1992.

14. UNIX International, *Requirements for Hierarchical Storage Management*, UI System Management Working Group, June 1992.

15. Open Software Foundation, *OSF Distributed Management Environment (DME) Architecture*, May 1992.

16. Coleman, S. and S. Miller, *Mass Storage Reference Model Version 4*, IEEE Technical Committee on Mass Storage Systems and Technology, May 1990.

17. Coleman S. and D. Isaac, *Mass Storage System Reference Model Version 5*, IEEE Storage System Standards Working Group, April 1993.

18. Collins, M. W. and T. McLarty, "Mass Storage System Reference Model System Management", *Digest of Papers, Ninth IEEE Symposium on Mass Storage Systems*, IEEE Computer Society Press, October 1988.

19. Buck, A. L. and R. A. Coyne, "Dynamic Hierarchies and Optimization in Distributed Storage Systems", *Digest of Papers, Eleventh IEEE Symposium on Mass Storage Systems*, IEEE Computer Society Press, October 1991.

20. Object Management Group, *The Common Object Request Broker: Architecture and Specification*, Draft OMG Document Number 91.8.1, August 1991.

21. Object Management Group, *The Object Model*, Draft OMG Document Number 91.9.1, September 1991.

22. Quinn, P. and G. Preoteasa, "Reconciling Object Models for Systems and Network Management", *UniForum*, 1991.

23. Hogan, C., et al., "The Livermore Distributed Storage System: Requirements and Overview", *Digest of Papers, Tenth IEEE Symposium on Mass Storage Systems*, IEEE Computer Society Press, May 1990.

24. Foglesong, J., et al., "The Livermore Distributed Storage System: Implementation and Experiences", *Digest of Papers, Tenth IEEE Symposium on Mass Storage Systems*, IEEE Computer Society Press, May 1990.

25. Collins, M. W.and C. W. Mexal, "The Los Alamos Common File System", *Tutorial Notes, Ninth IEEE Symposium on Mass Storage Systems*, IEEE Computer Society Press, October 1988.

END

DATE FILMED

9/30/93