On Distributed System Management

Germán Goldszmidt
Distributed Computing and Communication Lab
Computer Science Department,
Columbia University

Abstract

Device failures, performance inefficiencies, improper allocation of resources, security compromises, and accounting are some of the problems associated with the operations of distributed systems. Effective management requires monitoring, interpreting and controlling the behavior of the distributed system resources, both hardware and software. 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. Some of the fundamental limitations of this paradigm include limited scalability, micromanagement, and semantic heterogeneity. We propose an alternative model, Management by Delegation, and contrast its properties via an application example, evaluating the health of a Distributed System.

1 Introduction

As organizations are increasingly dependent on distributed systems they become more exposed to the inherent risks of such complex systems. Failures, performance inefficiencies, resource allocation, security compromises, and accounting are some of the problems associated with the operations of these systems. Management methodologies and tools are required to help the administrators handle those problems. Effective management requires monitoring, interpreting, and controlling the behavior of the distributed system resources, both hardware and software. Unfortunately, the distributed systems of today have not been designed to be manageable.

Users of large distributed systems want to take advantage of the special capabilities of different computational environments [1]. To fulfill such requirements, a distributed system must be equipped with mechanisms and policies that allow for efficient management of its resources.

An integrated management environment should enable managers to make timely management decisions. To be effective, managers must overcome the volume and complexity of information that characterize large heterogeneous distributed systems. Given the plethora of manageable resources, the hodgepodge of administrative approaches, inconsistent tools, and inadequate facilities [2], system management has become a difficult task.

Interest in management has been stimulated by the growing complexity of networks and increased dependence on their operations. Effective management of distributed systems has, therefore, emerged in recent years as a most central need. Despite some recent progress, however, manageability remains an elusive goal.

System management must be tailored to the different requirements of each system type and installation. Different applications and system configurations have different requirements and need different strategies [3]. Dynamic network configuration based on security, performance, or recovery considerations [4], is an example. To meet their functional requirements, system management processes must adapt, customize, and refine the capabilities of the available re-

[0] The IBM contact for this paper is Germán Goldszmidt, IBM Research, POBox 704, Yorktown Heights, New York, 10598

sources [5].

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. The platform-centered paradigm separates these applications, logically and physically, from the data they require and from the devices they need to control. This paradigm derives from traditional networked environments, where devices lacked resources to execute non-trivial management software, management data and functions were relatively simple, and organizations could devote the human resources needed to handle operations. Within large, heterogeneous distributed systems where devices incorporate significant processing power while human resources are scarce, the situation is different.

Management Standards Developments of management protocol standards (SNMP [6] and CMIP [7]) have served as the key driving forces of the field. These standards share the platform-centered paradigm, even if their approaches and details vary greatly. As described in Section 2, the platform-centered paradigm subsumed by the standards has serious limitations. The technologies that we are developing address these limitations and are orthogonal to the choice of methods to collect, organize or access management data.

Most distributed management applications are designed and implemented following the traditional client server process interaction paradigm, e.g., RPC [8]. This paradigm enforces a rigid association of functionality with servers. Services cannot be modified without recompilation, reinstallation, and reinstantiation of the server process. Static fixing of service interfaces induces an abnormal distribution of functionality between requesters and providers of management services. Server interfaces enable clients to obtain assistance only for predefined, fixed tasks.

Management by Delegation We introduced a more flexible paradigm, Management by Delegation (MbD) [9, 10]. This framework is based on *elastic servers* [11, 12], whose functionality can be extended at execution time by *delegating* new functional procedures to them. It allows flexible distribution of management responsibilities in a distributed environment.

Lack of this ability forces applications to engage in abnormally fine grained and complex process interactions. Managers must micromanage agents through primitive steps, resulting in ineffective distribution of management responsibilities, failure-prone management bottlenecks, and limitations for real-time responsiveness.

Elasticity can substantially improve the performance of distributed applications, in terms of computational cycles and communication bandwidth. Managers can define management programs in special purpose management scripting languages. MbD can store and instantiate delegated scripts and provides a concurrent runtime environment, where the scripts can execute asynchronously without requiring the manager's intervention. A delegation protocol allows a manager to transfer programs, create process instances, and control their execution.

Compressing Management Information Most network management systems are passive and offer little more than interfaces to raw or partly aggregated and/or correlated data in MIBs [13]. Developing effective technologies to support compression of real-time management information is a central problem of network management. A *health function* [14] provides such efficient compression by combining managed data linearly into a single index of network state. Health functions cannot be included as part of a static MIB design, as they may vary from site to site and over time. Nor can they be usefully computed at centralized management platforms, since this can result in excessive polling rates, lead to errors due to perturbations introduced by polling, and miss the goal of compressing data at their source. Instead, we use the MbD paradigm to support flexible and effective evaluation of health functions.

The rest of this paper is organized as follows. In Section 2 we review some of the management problems intrinsic to the platform-centered paradigm. Section 3 describes the characteristics of management by delegation,

and Section 4 presents some of the new challanges introduced by this approach. Section 5 discusses the application of the MbD paradigm to compress management information, and we conclude the paper in Section 6.

2 Limitations of Current approaches

This section reviews central barriers on manageability, intrinsic to the platform-centered paradigm. SNMP is used as a representative platform-centered protocol.

Platform-centered management is unscalable, as the rates at which management data may be accessed and processed typically exceed the network/platform capacity. If the size (number of devices), complexity (number of managed variables), or speed of the network increases or if management communications rates are bounded, the system quickly becomes unmanageable.

In high-speed networks, the rates at which management data must be tracked by the platform may exceed its processing capacity. In networks where management bandwidth availability is limited, the data collection rates may easily exceed the bandwidth available.

During failure times, furthermore, centralized management will tend to increase data access rates, at a time when the system is least capable to handle them. The likelihood of management failures will increase, due to delay or loss of management communications, at a time when fast and reliable response is most needed. Management applications will perform worst when they need to perform best.

Scalability is even more seriously bounded by the rates at which operators can handle data/alerts. The more data are collected and processed, the more taxing will be the task of operators, charged with interpretation of the data and symptoms. If the data collected could be interpreted by software applications instead of human operators, the problem would be simplified. Platform-centered management results in significant barriers on the development of such applications, such as the problem of micromanagement [10].

Management applications are, therefore, seriously limited in processing non-trivial tasks. In particular, the current paradigm is unsuitable to address manageability of high-speed (Gb/s) networks. The time scale over which the behaviors of such networks need to be monitored and management actions invoked is too short. A platform-centered paradigm could neither handle the enormous data rates involved nor access devices control functions fast enough. Flexible decentralization of management responsibilities is therefore essential for such networks.

Platform-centered management leads to significant barriers on manageability due to semantic heterogeneity. Even if management data can be brought to the platform, they are of little use for applications, in the absence of a uniform semantic model for their interpretation. Management protocols unify the syntax of data access, not its semantics.

The semantics of operational behaviors can vary greatly among resources. The manner in which devices are operated and controlled is often inseparable from their competitive advantages. A growing number of *private* MIBs have been introduced, each containing diverse device management features.

This explosive semantic heterogeneity of managed data renders the task of developing non-trivial management applications very difficult. In the absence of such applications, platform-centered management has been reduced to *"core-dumping"* cryptic device data on users screens and platform databases. This does not provide an adequate model for managing a distributed system. Typically, users can neither interpret the data, nor afford the costs of applications development and support. Organizations need networks that mostly manage themselves, minimizing the burden on their technical staffs.

3 Management by Delegation

We propose a new paradigm for distributed system management, capable of addressing the fundamental limitations of the platform-centered paradigm. Instead of bringing device data to platform applications, applications are

delegated to devices and executed in the environment of the data and functions that they need to access. The primary management responsibilities are dynamically shifted to the devices themselves.

A delegator entity (manager) uses a delegation protocol to download and control a delegated management program (DP). The DP is maintained and executed by an MbD-Kernel *elastic server* residing in a device. Multiple delegated Management Program Instantiations (DPI) may be spawned by management clients.

DPIs may include entire management protocol agents (e.g., SNMP, CMIP), or subsets thereof, supporting appropriate interfaces to respective managers. The delegation protocol supports downloading of DPs, instantiation, pausing/resuming DPI execution, and termination of DPIs. Control and access of DPIs may be shared by multiple clients. The MbD-Kernel environment contains interfaces to the device operations, permitting access by DPIs to local data and functions. These interfaces need not be standardized, nor do they have to be uniformly accessible to all DPIs. The MbD-Kernel supports access control to restrict the resources that may be accessible to different DPIs. A prototype MbD-Kernel supporting all the above functionality has been implemented [15].

A typical scenario of the use of delegation is shown in Figure 1. A management delegator processes (D) is transferring a delegated program (DP_1) to an elastic server (ES_1). If the delegated program is accepted by the elastic server, the delegator receives a handle that can be used to instantiate the DP.

For example, a network manager may delegate a procedure that implements a service for bulk retrieval of SNMP MIB data. This routine may later be invoked by the delegating manager and by other clients of the MbD-server to retrieve data from network devices that support the SNMP protocol. Managers may also delegate programs that will be instantiated as processes, for example, to monitor the status of the devices.

Figure 2 shows a snapshot of activity between delegators and an elastic server. Delegators have created multiple instances (DPI_1, DPI_2, etc.) of delegated processes. Associations have been established between delegators, elastic servers, and DPIs. DPIs may communicate between themselves (DPI_1 and DPI_2), with other clients (DPI_3 and client b), and use the delegation protocol to communicate with remote elastic servers (DPI_2). They can also invoke delegated procedures, (e.g. DPI_4 invokes DP_A).

A DP may be delegated to devices at booting time, and instantiated and executed at the device. The device will assume autonomous responsibilities to handle port failures. Communications occur during delegation time, but not a single communication is required during stress time.

Suppose, that it is desirable for an operator to control the decisions to partition ports via an SNMP management workstation. The procedure to partition ports, as well as the following fault-detection/notification program, could be delegated at boot time.

```
{
Upon symptom_event
   diagnostic_test(symptom_event),
   notify(manager_id, symptom_event)
}
```

The delegated fault-detection program could be instantiated at boot-time and generate specific TRAPs to the SNMP management workstation. These traps can then be used to invoke appropriate GETs of port_failure table entries, indicating which ports are in trouble. Operators can make decisions concerning partitioning of ports and invoke the partition_port procedure accordingly by instantiating it.

A few observations can be generalized from this example.

Distribution of Load. Automation of management functions via applications software can significantly reduce the load on operations centers. In networks where management communications bandwidth is scarce, device autonomy may be increased. In high-speed networks where fast reaction is needed, management programs responsible for such control functions may be located as close as possible to managed elements. Delegation and instantiation time may be conveniently scheduled (e.g., for

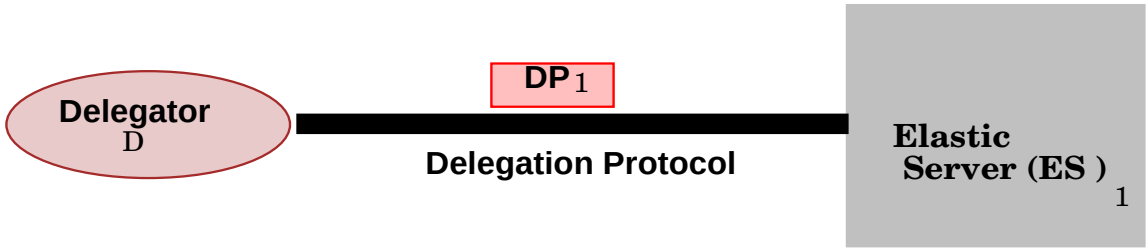

Figure 1: Delegating to an Elastic Server

boot time) leaving only minimal communications for stress times.

MbD, in contrast with platform-centered management, can flexibly control the granularity of manager-agent interactions and thus avoid micro-management. In cases where a program needs to pursue interactions at fine granularity, parts of the program or the entire program can be delegated and accessed by the manager when needed.

Heterogeneity. Management programs may be designed to handle the specific operational environment and distinct features of specific resources. They require neither a uniform semantic model of device data nor adaptation to different platform environments. A device vendor may equip the device with appropriate libraries of management programs and users can delegate and instantiate those as necessary or write their own.

MbD simplifies the problems of heterogeneity and eliminates barriers to management applications development. A device vendor can develop management application programs maintaining minimal components that can be easily ported.

The exposure of these programs to heterogeneity can be handled via carefully defined minimal interfaces. Contrast this with the task of developing platform applications (e.g., for fault management) that can handle the semantic heterogeneity of different devices. Therefore, management applications could be developed with relative ease by both vendors and users while accommodating heterogeneity and changes in the network environment.

Communications among clients and management programs may be supported via existing application-layer protocols. The problem of accomplishing applications communications standards is simplified to that of defining appropriate APIs, encapsulating a minimal functional interface. In contrast, SNMP requires defining a special MIB to accommodate variation of applications models, e.g., RMON [16].

Cost Tradeoffs MbD more suitably reflects emerging network tradeoffs and resource realities. Devices are increasingly equipped with substantial hardware resources (e.g., some hubs incorporate powerful RISC CPUs). These resources permit devices a great degree of management sophistication, far exceeding the simple device models that the platform-centered paradigm envisions.

The costs associated with operations dwarf those involved in procuring additional hardware resources within devices. All these contribute to dramatic changes in the nature of future networks, where device-centered management may provide a simpler, more flexible, scalable, and inexpensive management paradigm than the platform-centered management paradigm.

Interoperability. MbD can inter-operate with current management protocols. It may also extend their capabilities, e.g., SNMP MIB entries may be flexibly programmable via delegated scripts. MbD provides a complementary management paradigm to current management protocols.

For instance, MbD was used in conjunction with SNMP managers, in the port-failure handling example above. A delegated diagnosis program generated an event notification TRAP and recorded its finding in an appropriate MIB

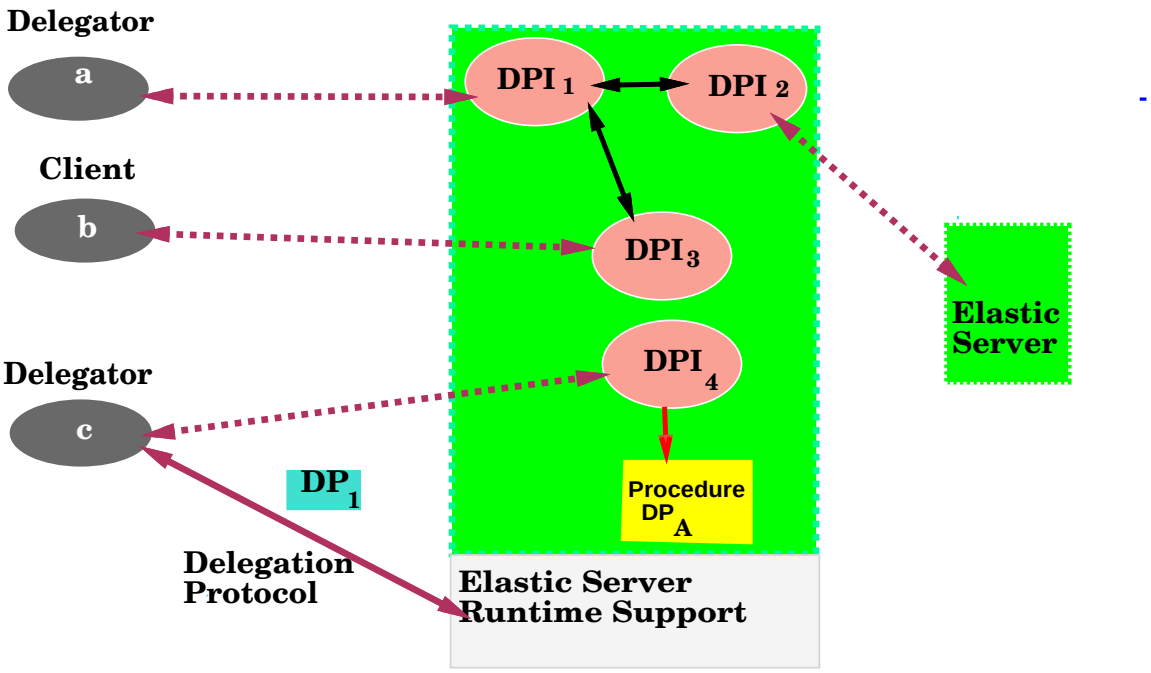

Figure 2: Delegated Programs in an Elastic Server

table. The manager accessed the SNMP agent, incorporated within the MbD-Kernel, to read these MIB variables.

Entire protocol agents may be incorporated within an MbD-Kernel as delegated programs. SNMP MIBs can be flexibly programmed to provide arbitrary device information. Contrast this with current MIBs, which have been rigidly defined to provide certain information and limit the ability of managers to obtain other information.

As another example, RMON is an SNMP MIB that permits remote configuration of pre-programmed monitoring processes. MbD, in contrast, permits flexible delegation of monitoring processes dynamically and recording the values which they compute within appropriate MIB locations for access by SNMP managers. Thus, MbD may enable programmable versions of RMON.

Delegation may also be used to accomplish conversions among different management protocols and inter-manageability. For example, an SNMP device may be scripted to support CMIP accesses. Inter-manageability is likely to be, in the near future, a source of major difficulties in managing internets.

Autonomy. Finally, MbD can be used to greatly improve autonomy and survivability of distributed systems. As the network grows, or involves more complex devices, management responsibilities may be delegated to devices to maximize their autonomy, and/or organized in a hierarchy to support regional autonomy. Use of MbD enables devices to acquire autonomous management capabilities, conditional on the network status. A device can use the status of its environment to instantiate appropriate management programs, reflecting different levels of autonomy. For example, when communications are lost with managing entities, the device may activate management programs that permit it fully autonomous management.

4 MbD Kernel and Delegation Protocol

A number of important issues must be addressed in the design and implementation of the MbD kernel:

Event-driven management. Management typically involves handling of exceptional events. Exception-handling mechanisms in programming languages are essentially synchronous with computations. That is, the occurrence of an exception (e.g., divide by zero, resource depletion) can be associated with certain computation steps. Management events, on the other hand, are defined in terms of the statistics of behavior traces, observed by processes external to the MbD kernel. The kernel must provide, therefore, efficient structures to allow management software to access external observation operations, detect and handle management events. MbD involves real-time computations in resource-constrained environments. Management programs must be capable of adapting to these resource constraints.

Recovery. Crash recovery of management systems is a relatively uncharted area. Unlike traditional database systems requiring strict restoration of the data to a consistent state, management systems can tolerate a great degree of inconsistency. As a result, crash recovery can be simplified and accelerated.

Portability Portability of the MbD kernel means more than the mere ability to move the software among different processing environments. The kernel needs to interface to local device monitoring/control processes, varying among devices and implementations. How to provide a uniform abstraction of these monitoring/control processes and appropriate interfaces that may be supported in real-time leads to non-trivial questions.

Inter-manageability, too, requires the definition of a uniform model of management that may be mapped to different protocols. An MbD-Kernel providing SNMP and CMIP interfaces may be implemented (inefficiently) via two independent agents. Alternatively, a common management database structure may be established that can be mapped to both protocols interfaces.

Security. Although security of management is clearly an important issue, it has only recently received significant attention. Management capabilities increase the risks of security compromises. Protocol analyzers, for example, can compromise communications by carefully analyzing protocol frame structures. Malicious attacks can target device controls and bring an entire network down. Even simple statistical observations can be used to identify critical network resources and target them.

Security in the context of MbD is particularly challenging. Because of its increased functionality, MbD results in an increased security exposure as compared with centralized management. The current MbD implementation supports a simple model of capabilities for DPs. The basis for comprehensive security in MbD will require a combination of authentication, data integrity, and data confidentiality.

However, simply providing these features is insufficient to secure MbD. In MbD, subjects will be trusted to download and execute scripts in remote entities. What these scripts are permitted to do will vary, depending on the identity of the subject making the request. A framework is needed to identify subjects in the distributed management system and to associate privileges with the subject identity. These privileges will vary; for example, one subject may be permitted only to run a simple script to obtain a specific piece of information, while another may be permitted to perform complicated calculations, using significant CPU resources. Still others will be given complete control of the remote entity.

Separate Administrative Domains. In some applications of MbD, subjects will receive information from components in domains that they do not trust. These mutually suspicious components may need to cooperate to resolve management problems. Although the security features described in the previous paragraph can ensure that the source of information is identified (data origin authentication) and that

the information is not modified while passing through the network (data integrity), the information itself may not be entirely trustworthy.

A subject receiving this information needs a way to associate a level of trust with the information, based on the source of the information. For example, a subject would not want to disable a router interface based on information that was received from an untrusted domain.

5 Health of A Distributed System

A distributed system needs to support effective management decisions based on vast amounts of real-time operational data. Such decision processes can be viewed as methods to compress management data into a simpler decision. Often, much of the compression needed to evaluate management decisions is accomplished through manual processes.

One method to compress operational data is to compute index functions, reducing a large number of observed operational variables to a single indicator of the system state. This is similar to the use of different indexes to reflect the state of the securities market (e.g., Dow Jones index) and the economy (e.g., inflation index). Such indexing typically uses linear aggregation of a large number of variables, each providing a different microscopic measure of state. We call such a linear weighted function of management variables a *health* function.

In SNMP, observations of operational variables are accomplished via counters and gauges. A counter represents a cumulative (integral) of an operational variable. Typically, however, only the change in the counter versus its value provides a useful indication of the network state. For example, the MIB-II [17] counter representing `ifInOctets` accounts for the total number of bytes received by an interface since device initialization. Health functions will, typically, utilize a linear combination of such rates at which status indicators vary.

It is often useful to combine MIB variables to more useful status indicators. For example, the utilization of an interface at time `t=sysUpTime` can be defined as

$$\mathrm{U(t)} = \frac{(\mathrm{ifInOctets} + \mathrm{ifOutOctects}) * 8}{(\mathrm{ifSpeed} * \mathrm{SysUptime} * 100)}$$

where `IfInOctets (IfOutOctets)` gives the total number of bytes received (sent). This measure provides an average sense of utilization over a time window since boot-time. A useful indication of the instantaneous network state is provided by the derivative $u(t) = U'(t)$. This derivative may be approximated by frequent sampling of the respective managed variables and computations of the changes in U.

Similarly to utilization, one can establish measures of instantaneous error rates to capture additional micro-state indications. For example, the percentage of input errors to packets delivered can be evaluated as

$$\mathrm{E(t)} = \frac{\mathrm{ifInErrors}}{(\mathrm{ifInUcastPackets} + \mathrm{ifInNUcastPkts})}$$

A health function can be used to linearly aggregate these micro-measures. For example, the state of the network seen by a hub, including multiple interfaces, can be indexed by a health function $\mathcal{H}(\vec{e}, \vec{u}) = \vec{A}\vec{e} + \vec{B}\vec{u}$ where $\vec{e}$ and $\vec{u}$ represent the evaluation of e and u at all managed entities and $\vec{A}$ and $\vec{B}$ are weight vectors.

Current network management paradigms do not support the flexibility and decentralization required to compute health functions effectively. A health function could not be usefully incorporated as part of a static SNMP MIB design. The specific function used and its parameters may vary among installations, device configurations, and even time of the day.

Similarly, a health function could not be usefully incorporated as part of an OSI managed object. While the OSI management framework permits encapsulation of functions within managed objects and their remote invocation by managing entities, these functions must be statically bound to a managed object at its design time. Designers of managed objects may not usefully provide configuration/installation/site/time-independent health functions.

Nor can a health function be usefully computed by managers at centralized management

platforms. First, the rates of polling required to aggregate the variables used may far exceed platform processing capability. In the example above, suppose a device is polled every 0.1 seconds. Suppose too, that the total number of interfaces aggregated by health functions is n (say n=200). The aggregated polling rate is then 10n (e.g., 2000) SNMP requests per second. Second, the very goal of compression is to reduce data volume at the source. Third, polling through the network introduces random perturbations in approximating temporal derivatives of managed variables, leading to errors and potential hazards in decisions.

A static approach to collect operational data defines the collection of observations a priori and independently of their use. Since the need for operational data cannot always be predicted, many observations are collected and stored for potential access by management applications. This results in the collection of large amounts of data that are never used. Furthermore, in most MIBs, applications cannot control the execution of the observations. Thus, the exact meaning of the observations depends on the MIB implementor's design decisions.

Standard network management approaches require a priori knowledge of what algorithms are mapped into statically defined objects. For example, the OSI Workload Monitoring Function [18] specifies metric objects for determining resource performance and utilization. The parameters that define what constitutes a *healthy* network depend on the particular installation configuration, usage, and administrative policies. These parameters vary among different systems and during different times within the same network. Therefore, health functions cannot be statically defined, but should be dynamically bound to agents when needed.

Delegating health functions to MbD agents enables compression of data at the source. It permits flexible changes in health functions to reflect specific behaviors at different sites and times. It allows the computations of health functions to accomplish great precision, through direct access of operational values and minimal observation delays. It improves scalability of management by reducing the rates of polling needed and restricting it mostly to times when problems are identified via aggregated health measures. It permits local evaluation of health functions and decisions by device agents, when management platforms have difficulties in accessing devices (during critical stress times). Consequently, we use MbD to compute health functions [14].

Delegating health functions to distributed agents enables direct observation of network behaviors at sufficient precision. Health functions may be dynamically changed to reflect varying behavior patterns at different times. By maintaining health indicators locally, vast amounts of real-time data can be significantly compressed.

To study the application of MbD a simple health application for an Ethernet has been prototyped. It consists of manager processes, a health process, and a collection of observers. Manager processes can dynamically reconfigure the distributed application. The observers compute observations such as operational state, interface utilization, rate of collisions, and interface errors. They also perform corrective actions. The health process receives reports from the observers and evaluates a higher-level abstraction ($\mathcal{H}$) of the state of the distributed system.

A manager process can specify (and later modify) a list of observers with which health will communicate and the relative weights of their evaluations in the overall score or index. Health may report its evaluated index to managers, by answering an explicit request, via setting a private SNMP MIB variable or as event reports.

Diagnostic procedures and corrective actions can be dynamically delegated to be executed when necessary, without manager intervention. For example, if the level of an Ethernet utilization becomes too high, an observer process may automatically disconnect (temporarily) the device which is the source of the most packets from the network.

6 Conclusions

Current management systems pursue a platform-centered paradigm, which is unscalable and introduces micro-management.

Elastic servers introduce a new paradigm of interaction between application components on a distributed environment. This paradigm addresses the fundamental limitations of platform-centered management. The elastic server model supports flexible distribution of functions and dynamic allocation of resources. It provides a software framework that enhances the expressive power of client/server interactions.

Management applications must assist managers in making effective decisions. Health functions provide an efficient method to overcome the volume and complexity of data which characterize large heterogeneous distributed systems. The definition of what constitutes a *healthy* network cannot be standardized or fixed for all networks, since it is installation and time dependent. Thus, it is not sufficient to provide fixed definitions as part of specific programs or MIB variables. MbD supports a dynamic approach to defining observations. This dynamicity permits applications to configure observation processes to flexibly monitor information of their interests. Management decisions, such as to temporarily disconnect a device, are executed efficiently, without the need for manager platform intervention. Real-time operational data are effectively compressed at the MbD agent, and managers can define observation operators tailored to their changing needs.

Acknowledgments.

I would like to express my sincere thanks to my advisor professor Yechiam Yemini, for his valuable suggestions and encouragement.

References

[1] Andrzej Goscinski. *Distributed Operating Systems The Logical Design.* Addison Wesley, 1991.

[2] Matthias Autrata. OSF Distributed Management Environment. In *Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Santa Barbara, CA, October 1991.

[3] Wesley W. Chu. System Management Research via Behavior Characterization. In *IEEE First International Workshop on Systems Management*, Los Angeles, California, April 1993.

[4] C. Bisdikian, F. Patel, F. Schaffa, and M. Willebeek-LeMair. Dynamic Reconfiguration of Hub-based Networks. In *IEEE First International Workshop on Systems Management*, Los Angeles, California, April 1993.

[5] Miriam J. Masullo and Seraphin B. Calo. Policy Management: An Architecture and Approach. In *IEEE First International Workshop on Systems Management*, Los Angeles, California, April 1993.

[6] Jeffrey D. Case, Mark S. Fedor, Martin L. Schoffstall, and James R. Davin. A Simple Network Management Protocol (SNMP). RFC 1157, May 1990. DDN Network Information Center, SRI International.

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

[8] Patricia Gomes Soares. On Remote Procedure Call. In *Proceedings of the Second CASCON International Conference*, Toronto, Canada, November 1992.

[9] Yechiam Yemini, Germán Goldszmidt, and Shaula Yemini. Network Management by Delegation. In *The Second International Symposium on Integrated Network Management*, Washington, DC, April 1991.

[10] Germán Goldszmidt, Yechiam Yemini, and Shaula Yemini. Network Management: The MAD Approach. In *Proceedings of the IBM/CAS Conference*, Toronto, Canada, October 1991.

[11] Germán Goldszmidt. Elastic Servers in Cords. In *Proceedings of the Second CASCON International Conference*, Toronto, Canada, November 1992.

[12] Germán Goldszmidt. Distributed System Management via Elastic Servers. In *IEEE First International Workshop on Systems Management*, Los Angeles, California, April 1993.

[13] B.N. Meandzija, K.W. Kappel, and P.J. Brusil. Integrated Network Management and The International Symposia. In *The Second International Symposium on Integrated Network Management*, Washington, DC, April 1991.

[14] Germán Goldszmidt and Yechiam Yemini. Evaluating Management Decisions via Delegation. In *The Third International Symposium on Integrated Network Management*, San Francisco, CA, April 1993.

[15] Germán Goldszmidt and Yechiam Yemini. The Design of a Management Delegation Engine. In *Proceedings of the IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*, Santa Barbara, CA, October 1991.

[16] S. Waldbusser. Remote Network Monitoring Management Information Base. RFC 1271, November 1991.

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

[18] International Standards Organization - ISO. Information Processing - Open System Interconnection - Systems Management - Part 11: Workload Monitoring Function. Sydney, Australia, December 1991.

About the author

Germán Goldszmidt received his B.A. and M.S. degrees in Computer Science from the Technion, Israel Institute of Technology. Currently he is a Ph.D. candidate in Computer Science at Columbia University and a member of the Distributed Computing and Communication laboratory at the Center for Physical Science Research. Since 1988 he has been working at IBM's T.J. Watson Research Center with the Distributed Systems Software Technology group. His current research interests include distributed computing systems and languages, network and system management, and debugging of concurrent programs. He can be reached at the address Computer Science Dept., Columbia University, New York City, NY 10027. His Internet addresses are german@cs.columbia.edu and gsg@watson.ibm.com.